universität wien

DISSERTATION

Titel der Dissertation

"Simulation of Automated Negotiation"

Verfasser

Mag. Michael Filzmoser

Angestrebter akademischer Grad

Doctor of Philosophy (PhD)

Wien, im März 2009

Studienkennzahl lt. Studienblatt:	094 146
Dissertationsgebiet lt. Studienblatt:	Management
Betreuer:	o. Univ.-Prof. Dr. Rudolf Vetschera

Gewidmet

Meiner Frau Kerstin und meinen Kindern Jade, Kimberly, Sebastian und Niklas.

Danksagung

Größter Dank gilt meiner Familie. Meiner Frau Kerstin und meinen Kindern Jade, Kimberly, Sebastian und Niklas danke ich für die vielen schönen Stunden die wir bisher miteinander verbringen durften und noch miteinander verbringen werden. Danke dafür, dass wir alle Probleme die sich uns gestellt haben gemeinsam immer meistern konnten und auch für die Nachsicht die ihr aufgebracht habt, wenn wegen der Arbeit an dieser Dissertation oder anderen beruflichen Verpflichtungen einmal weniger Zeit für euch blieb. Ihr seid der Grund meines Strebens und der Anreiz für jede Leistung. Meinen Eltern Marianne und Walter danke ich für ihre Unterstützung in allen Lebenslagen, ich kann nur hoffen eurem Beispiel bei meinen Kindern folgen zu können. Auch meinen Freunden danke ich für viele gesellige Abende, die – wenn auch produktiver – nie besser hätten verbracht werden können.

Weiters gilt mein Dank meinen KollegInnen am Lehrstuhl für Organisation und Planung des BWZ der Universität Wien. Sabine, Eva, Marita, Joe, Albert und Michele, danke für interessante Diskussionen, euren Rat und euer Feedback. Besonders bedanken möchte ich mich bei meinem Betreuer Univ.-Prof. Dr. Rudolf Vetschera. Nicht nur für Ihren unschätzbaren Rat bei wissenschaftlichen Fragen und Problemen möchte ich mich herzlich bedanken, sondern auch für Ihren Rückhalt und Ihr Vertrauen in meine Fähigkeiten, ich hoffe mit meinen Leistungen Ihre Erwartungen erfüllt zu haben.

Declaration

The work in this dissertation is based on research carried out at the Chair for Organization and Planning of the Department of Business Administration, Faculty of Business, Economics, and Statistics, at the University of Vienna, Austria. No part of this dissertation has been submitted elsewhere for any other degree or qualification and it is all my own work unless referenced to the contrary in the text.

Contents

List of Figures

List of Tables

List of Source Code

Chapter 1

Introduction

The present hype of agent-based methodologies in computer science and the potentials of software agents that interact autonomously over the Internet established the field of automated negotiation, which is supposed to be of considerable importance for research and practice in the future (Zeng and Sycara, 1998; Tu et al., 2000). Though in electronic business information, orders, and payments can be handled electronically, automation for the task of negotiating the final contract is still missing to a large extent, necessitating human intervention which increases transaction costs and diminishes the potential value of electronic business (Maes et al., 1999). Scholars argue that the evolution of agent-mediated electronic business will change the future of traditional business and lead to a radical reorganization of economic structures (Kontolemakis et al., 2004; Nwana et al., 1998). In agent-based systems on the other hand sophisticated interactions between autonomous software agents like biding, voting, and negotiation are largely not present (Kraus, 1997). Making negotiation an interaction mechanism available for autonomous systems could replace current primitive rules of encounter and thereby improve outcomes of such interactions (Rosenschein and Zlotkin, 1994; Kraus, 1997).

Albeit the youth of the field of automated negotiation one must not forget its roots in the more established field of negotiation, as automated negotiation basically is a special form of negotiation where the negotiation process is automated by autonomous software agents and protocols governing their interaction. In this introductory chapter we therefore first will outline the importance of negotiation in general, briefly review the different approaches to study negotiation and their major results, and discuss how the automation of negotiation blends in existing negotiation research. Basing on this discussion we determine the research questions of this dissertation – which originate from deficiencies in present simulation studies of automated negotiation – and the research methodology to approach them.

1.1 Importance of negotiation

In private as well as in professional life a tremendous amount of conflicts among individuals, among organizations, or between individuals and organizations exists. These conflicts are of diverse content and varying importance. One mechanism to cope with conflicts – among others like for instance traditions, regulations, judicature, arbitration, mediation, or fiats – is negotiation

(Raiffa, 1982). Dupont and Faure (2002) define negotiation as *'... a process of combining conflicting positions into a common position, under the decisive rule of unanimity'* (Dupont and Faure, 2002, p. 40). As mutual agreement (unanimity) on one out of the set of possible alternatives is necessary to settle a dispute at hand and conflicting interests over these alternatives between the negotiating parties exist, negotiation combines the dual and most often conflicting motivations of the individual (competitive) desire to maximize the own utility of the outcome and the collective (cooperative) desire to reach an agreement in a mixed-motive activity (Bartos, 1977; Lewicki et al., 1994; Kersten, 1997; Dupont and Faure, 2002; Kersten, 2007) so, just like Bartos (1977) states, *'... the true conflict in all negotiations is that between competitive individualism and cooperative collectivism.'* (Bartos, 1977, p. 572).

If the parties in a negotiation mutually settle on one of the possible alternatives they have reached an agreement, if they however fail to do so some party will break-off the negotiation so that no agreement is reached – a situation called stalemate, deadlock, or impasse. To reach an agreement the parties not only will exchange messages to inform and influence the opponent, but above all have to exchange tentative proposals for settling the conflict – so-called offers (Cross, 1965). The exchange of offers is argued to be the most important type of communication in negotiations. Thus scholars from the fields of management science, economics, and game theory model negotiation or bargaining[1] as the process of exchanging offers and counter-offers (Tutzauer, 1992).

Negotiations are prevalent in many areas such as labor-management disputes, law and court, conflicts between whole nations or single individuals, as well as between and within organizations. All of us are confronted with the need to negotiate not only in these contexts but also in our private lives, e.g. when a couple has to mutually agree on how to allocate household tasks, where to go for dinner, or what film to see at the cinema. The relevance of negotiation research and teaching is also reflected in the developments of the academic sector in the early 80's, when negotiation became the perhaps fastest growing area of teaching in management schools (Bazerman et al., 2000). Moreover it is estimated that about 20% of a manager's time is spent on various kinds of negotiations – with suppliers, customers, business partners and subordinates – and remaining tasks are strongly influenced by the outcomes of these negotiations ((King, 1981; Byrnes, 1987) cited by Wall and Blum (1991) and (Shea, 1983) cited by Foroughi (1998)).

1.2 Development of negotiation research

Hence, considering the significance of negotiations it is not surprising that negotiation – besides in anecdotal literature from professionals, who provide case- and experience-based advice, report best-practice, and derive principles for negotiating successfully, as e.g. Fisher and Ury (1981) – is a major element of the research agenda of many scientific fields, including economics and game theory, social psychology, behavioral decision research, negotiation analysis, and research on negotiation support systems.[2]

[1]We use the terms 'negotiation' and 'bargaining' synonymously throughout this dissertation to refer to the conflict resolution mechanism sketched above

[2]The following overview of research on negotiation not claims to be an exhaustive review – such a review would deserve a dissertation on its own – but only aims at giving a brief overview of the different approaches to study negotiation and their key results. Together with these short descriptions we provide additional literature for each

Economics as a social science focuses on problems of choice under scarcity of resources and the determination of economic value. Reviewing the development of research in bargaining until the 1950s economists criticize that for a long period of time economics did not include a theory of bargaining (Pen, 1952; Cross, 1965, 1969, 1977). With a focus on the development of the research in bilateral monopoly, an important application of bargaining in economics, Pen (1952) states that due to inadequate theories the price under conditions of bilateral monopoly was designated as 'indeterminate'[3] (Edgeworth, 1881; Pen, 1952; Harsanyi, 1956) and resulting from bargaining. Bargaining, however, was not investigated in more detail as it was not considered to be an economic problem, but to belong to the realm of psychology (Pen, 1952). In another domain of economics, however, first fruitful attempts come from Zeuthen (1930) and Hicks (1932), who focus on the domain of management-union wage bargaining (collective bargaining). While these models are domain-specific (dealing with wage rates or strike durations) and static, Cross (1965, 1969, 1977) proposes a general model to analyze bargaining as a dynamic process of learning under incomplete information, where the parties adapt their strategies when they learn that their expectations about their opponent's strategy are wrong.[4]

Also game theory – as a branch of applied mathematics that studies strategic situations where rational players choose different actions in an attempt to maximize their returns – provides mathematical modeling and analysis for research on negotiation. Von Neumann and Morgenstern's book 'The Theory of Games and Economic Behavior' (1944) is – though game theoretic analyses were already used by e.g. Bernoulli, Bertrand, Cournot, Edgeworth, or von Stackelberg before to investigate specific questions – in common opinion the cornerstone of modern and general game theory. In game theory the situation of interest is modeled as a game, where the parties ('players') have to chose separately from different alternative plans of action ('strategies') which then together determine the outcome ('payoff'). In their investigation of two-person bargaining problems von Neumann and Morgenstern could not determine an unique solution but only argue that the solution has to be on the individually rational range of the Pareto frontier. This, however, was found for bilateral monopoly by Edgeworth years before (Bishop, 1963) and therefore does not constitute a progress in the field. Nash was the first to defined the 'bargaining problem', as a situation in which individuals have the possibility of concluding a mutually beneficial agreement, but conflict of interest exists among the bargainers about which agreement to conclude and no agreement may be imposed on any individual without his approval by the other (Nash, 1950). A bargaining theory then is an exploration of the relation between the outcome of bargaining, the characteristics of the situation and the actions of the individuals. Such game theoretic approaches for the analysis of bargaining in general can be divided into axiomatic (from the realm of cooperative game theory) and strategic (from the realm of non-cooperative game theory) approaches.

In conjunction with his definition of the bargaining problem Nash (1950) also introduced the

of the research fields for the interested reader, a good general overview is provided by Teich et al. (1994).

[3]Indeterminate besides the two conditions that it (i) has to be on the contract curve – i.e. be one out of the set of alternatives where neither party's utility can be improved without at the same time reducing the utility of the other party – and (ii) has to be individually rational for both parties – i.e. the limits of the settlement zone on the contract curve are given by the parties utility of no agreement, as either party would otherwise prefer no agreement over agreement on such an alternative of lower utility (Edgeworth, 1881; Harsanyi, 1956).

[4]Moreover Cross was the first who emphasized the importance of time in bargaining, by arguing that if it is not important when an agreement is reached, why should it be important whether an agreement is reached at all (Cross, 1965) – an idea later intensely sized by game theorist in their strategic approaches to bargaining.

axiomatic approach to tackle it. The axiomatic approach derives solution functions for bargaining problems basing on 'desirable' properties of the outcome of bargaining described by axioms and the assumption of rationality of the players, which determine the solution to the bargaining problem without specifying the bargaining process (e.g. (Nash, 1950; Raiffa, 1953; Kalai and Smorodinsky, 1975; Rosenthal, 1976; Roth, 1977; Thomson, 1981; Gupta and Livne, 1988; Gupta, 1989)). The strategic approach, on the other hand, determines equilibrium outcomes based on formalized procedures for very specific bargaining problems. Strategic research in bargaining was introduced by Ståhl (1972) and pursued by Rubinstein (1982) – both using an alternating offer protocol and impatient players for dividing a 'pie' – i.e. a fixed surplus. Combining the two former another game theoretic approach to the bargaining problem proposed by Nash (1953) - and therefore termed 'Nash program' (Osborne and Rubinstein, 1990) - is to relate axiomatic solutions to the equilibrium of strategic models. This makes sense as each approach alone has its deficiencies, the axiomatic approach on the one hand omits the negotiation process, the strategic approach on the other hand relies on restrictive assumptions about the underlying problem and the players interactions, as Nash states: *'The two approaches to the problem, via the negotiation model or via the axioms, are complementary; each helps to justify and clarify the other.'* (Nash, 1953, p.129). Researchers have shown that such connections exists for: (i) Nash's demand game and the Nash solution (Nash, 1953), (ii) a version of Zeuthen's conflict-risk game and the Nash solution (Harsanyi, 1956), or (iii) a version of Rubinstein's (1982) alternating offers game and the Nash solution (Osborne and Rubinstein, 1990).

While in the early years emphasis was placed on the axiomatic approach of game theory in recent years the strategic approach is used more widely. With a focus on more realistic bargaining problems of multiple issues (Chatterjee and Lilien, 1984) and applying Rubinstein's (1982) alternating offer protocol, issues are considered separately rather than combining them to utility values for whole packages (Lang and Rosenthal, 2001). Game theorists for example started to analyze issue-by-issue bargaining under fixed agenda (Fershtman, 1990), agenda setting for issue-by-issue bargaining (Busch and Horstmann, 1999a), compare issue-by-issue and bundle offer protocols (Lang and Rosenthal, 2001; Inderst, 2000), and analyze the signaling of private information through agenda setting (Bac and Raff, 1996; Busch and Horstmann, 1999b).[5]

In contrast to game theory's symmetric normative approach to investigate how rational actors behave in separate interactive decision making, negotiation analysis is an asymmetric descriptive-prescriptive approach (Raiffa et al., 2002). Negotiation analysis (Raiffa, 1982; Young, 1991; Sebenius, 1992; Raiffa et al., 2002) advises, building on approaches from decision analysis, one focal negotiator – therefore asymmetric – how to behave in negotiations to improve his outcome based on a descriptive analysis of the opponent – rather than on assumption of rational behavior. Many of the deficiencies of game theory for analyzing negotiation – like for instance multiple equilibria, departure from rational behavior by actors, or lack of common knowledge (Sebenius, 1992) – are thereby circumvented. It is argued that negotiation analysis builds the bridge between prescriptive and descriptive methods in negotiation research so that both can inform each other to deepen our understanding of negotiation and improve negotiation theory building (Bazerman and Neale, 1991; Hausken, 1997).[6]

[5]Napel (2002) and Osborne and Rubinstein (1990) provide a good review of game theoretic approaches to negotiation.

[6]Good introductions to this strand of research are provided by Young (1991) and Raiffa et al. (2002).

In the 1960s and 70s social psychological studies of negotiation focused on the influences of individual differences between negotiators and situational characteristics of negotiations on negotiation outcomes. The large body of empirical studies on the influence of individual differences – demography and personality – however indicates that these factors to a large extent fail to explain variance in negotiation behavior and outcome (Rubin and Brown, 1975; Thompson, 1998; Bazerman et al., 2000). A later review of studies performed in the 1980s comes to the same result, that there is little evidence for the effects of individual traits on the negotiation (Wall and Blum, 1991). Moreover these analyses are – though interesting in general – of limited use for a negotiator as often the independent variables used in these studies like the personality of the negotiator or the context of the negotiation are not controllable.

From the perspective of behavioral decision research the negotiator is regarded as decision maker. Studies in this field focused on investigating how negotiators systematically deviate from rational behavior in negotiations, by comparing the actual behavior in experiments to rational and optimal behavior, in contrast to the former social psychological studies where such a benchmark was lacking. Bazerman et al. (2000) in a summary of the findings of studies from behavioral decision research on negotiation list a number of such systematic deviations from rational behavior. For example empirical studies evidence that negotiators are responsive to framing and anchoring – i.e. positively framed negotiators concede more and negotiators are inappropriately affected by provided anchors in their decision making during negotiations. Negotiators also often ignore the opponent's perspective and devaluate suggestions of the opponent. In making decisions they rely on easily obtainable information and are overconfident and over-optimistic about the likelihood of achieving outcomes that favor themselves. Moreover evidence was found that negotiators often falsely assume that their preferences are completely incompatible with those of the opponent (fixed pie assumption) and refuse to change their behavior though rationality would dictate this (conflict escalation).[7]

To aid negotiators in their task researchers developed negotiation support systems for negotiation preparation and training, and to facilitate the actual negotiation process. A negotiation support system is a computer system that supports the entire negotiation process through its main components: (i) a decision support system for each negotiating party, and (ii) an electronic communication channel between the negotiators (Lim and Benbasat, 1992; Delaney et al., 1997). With these components negotiation support systems are argued to be helpful in alleviating the negative impact of negotiator's cognitive limitations, cognitive biases, and socio-emotional problems which are the major stumbling blocks to successful negotiation (Foroughi et al., 1995; Foroughi, 1998). Though the general aim is the same for all negotiation support systems, i.e. the improvement of negotiation processes and thereby negotiation outcomes, they differ in the support philosophies followed by the developers (DeSanctis and Poole, 1994) and therefore in the system features that provide support. Just to give a few examples: The communication support implemented in `Negoisst` (Weigand et al., 2003; Schoop et al., 2003) focuses on the efficiency of the communication by providing a structured communication protocol. `Inspire` (Kersten and Noronha, 1999b) on the other hand provides decision support in eliciting and providing preference information (both in graphical and numerical form) to achieve efficient outcomes,

[7] An introductory review of work on negotiation in social psychology and behavioral decision research is provided by Bazerman et al. (2000).

or `Negotiator Assistant` (Druckman et al., 2002, 2004) tries by means of mediation – i.e. the analysis of the current state of the negotiation and provision of context-specific advice – to increase flexibility and thereby the prospects of reaching an agreement in case of stalemate. Empirical evidence suggests that negotiation support systems succeed in their mission as their usage leads to better outcomes than face-to-face negotiations or negotiations via e-mail (Delaney et al., 1997; Rangaswamy and Shell, 1997; Foroughi, 1998; Köszegi et al., 2006).

1.3 Contributions of automated negotiation

A closer look at the above mentioned research streams indicates that – despite of methodological differences – they all share the aim to improve negotiation outcomes. Anecdotal literature distributes best-practice knowledge acquired by expert negotiators. Game theory and economics demonstrate for specific situations how a negotiator should rationally behave and therefore offers normative advice on how to cope with a negotiation problem and which outcomes one can expect. Behavioral decision research tries to expose biases negotiators demonstrate when making decisions, which when respected can be avoided. Negotiation analysis provides systematical support for the preparation for negotiations, to reduce complexity and uncertainty of the problem, which in turn should improve the performance of bounded rational negotiators. Finally negotiation support systems take further this idea in facilitating the use of negotiation analytical tools (e.g. by preference elicitation and visualization or post-settlement phases to improve reached agreements) and the communication between negotiators. These improvement attempts are vital, as despite the importance of negotiations for economic activity and every day life empirical studies found that the performance of humans in conducting negotiations is rather poor.

> 'Often, disputants fail to reach an agreement when, in fact, a compromise does exist that could be to the advantage of all concerned. And the agreements they do make are frequently inefficient: they could have made others that they all would have preferred'. (Raiffa, 1982, p. 358)

So the increase of the proportion of settlements reached through negotiation in cases where alternatives exist that are preferred to an impasse by all parties involved and even incremental improvements in the efficiency of negotiated agreements – as they frequently occur in everyday life – could outrank improvement policies in other areas in terms of total welfare gains (Crawford, 1982). As mentioned there is evidence that the usage of negotiation support systems improves negotiation outcomes. However, still the bounded rational human users make the decisions of whether to accept an offer, make a counter-offer, or break-off negotiations, as well as whether or not to make use of the manifold functionalities of these systems (Bichler, 2000). Studies show for example that humans behave inconsistent with their preferences (elicited by the negotiation support system) even when these utilities are displayed during the negotiation process, or that they reject to enter a post-settlement phase though this could improve their outcome. Given the deficiencies of human negotiators to reach (Pareto-optimal) agreements, they also might make errors in their usage of negotiation support systems or not fully exploit the possibilities they offer.

Recent studies on negotiation support systems, that applied agent technology to support the users, found out that the users not only accept the software agent's support, but also demand additional functions for the supporting agent and partial automation of the negotiation process (Chen et al., 2005a). A logical further-development of these recent efforts would be that computer systems entirely assume the burden of negotiation by automating negotiations (Oliver, 1996). Such automated negotiations then are negotiations in which autonomous software agents, surrogating their human users, perform the necessary tasks and make the necessary decisions to conduct negotiations on their users' behalf and in their users' interest.

Besides this automation of negotiations for problems currently addressed by means of human negotiation, which should improve negotiation outcomes, a further area of application of automated negotiation is the coordination of software agents in autonomous systems. The automation of negotiation makes this coordination mechanism applicable to such automated systems that operate without human interference, and creates value if coordination by negotiation yields better outcomes than achieved under the currently used – often simplistic and rigid – interaction mechanisms.[8]

1.3.1 Automated negotiation in electronic business

Actually automated negotiation will only be applied instead of traditional negotiation between human agents if and where there are benefits of doing so (Blecherman, 1999). Such potential benefits of automated negotiation for electronic business can either result from lower transaction costs, improved negotiation outcomes, or a combination of both – also trade-offs are thinkable e.g. that the lower costs of automated negotiation outweigh inferior outcomes compared to traditional negotiations so that the net benefit of automated negotiation still exceeds that of negotiation between humans.

The transaction costs of negotiation, include both direct costs as lawyers' cost as well as indirect costs like opportunity costs of not being able to do other profitable activities due to the time and effort spend on negotiating (Kersten, 2007). Doubtlessly automation of negotiation reduces indirect opportunity costs associated with the negotiation mechanism compared to human negotiations as it surrogates human involvement and thereby reduces or avoids costs caused when humans perform this task. Moreover the direct costs for automated negotiation itself are negligible, it proceeds very quickly – the independence from the accessibility of humans, which

[8]Furthermore automated negotiation can be considered itself as an area of application of game theory. Recent research in experimental economics criticizes the application of game theory for human interactions due to detected economic anomalies (i.e. real phenomena that are inconsistent with the expectations of the economic paradigm) for many of the building blocks of bargaining theory such as ultimatum bargaining (Güth et al., 1982), sequential bargaining (Neelin et al., 1988), or equilibrium concepts (Ochs and Roth, 1989). Often formal models suffer from unreasonable assumptions, to keep them analytically solvable, so that results have limited relevance. Game theoretical bargaining models for example often assume common knowledge, perfect information, and perfect rationality of participants (Sebenius, 1992). Though neither humans nor computer programs are ideal game theory agents – as none of them are capable of unlimited reasoning power – it seams that game theory is more applicable to software than to human agents which are closer to game theory's idealization of an agent (Rosenschein and Zlotkin, 1994). Hence, game theory could provide valuable inputs for the determination of interaction protocols and software agent strategies for automated negotiation (Rosenschein and Zlotkin, 1994; Varian, 1995; Binmore and Vulkan, 1999). Even game theory's central notion of 'strategy' – as a specification of what to do in every situation during an interaction – takes a clear an unambiguous meaning when it becomes simply a computer program – and actually is a description of reality for computer programs – while humans would not choose such a fixed strategy before an interaction and follow it without alteration (Rosenschein and Zlotkin, 1994).

is disrupted by time zones or geographical distance, makes automated negotiation much faster (Bichler, 2000; Choi et al., 2001; Maes et al., 1999) than traditional negotiation – and therefore with minimal consumption of computer resources, any consumption of resources that remains is likely to involve resources that would otherwise be idle and for which therefore no real usage fees can be accounted (Cranor and Resnick, 2000). Even if, due to problem characteristics (e.g. in the case of complex or combinatorial problems), a significant amount of computer resources is necessary these resources still are relatively cheap compared to the cost of human interference and continuously become cheaper over time.

Concerning the utility of the outcome of automated negotiations, many scholars share the assumption that automated negotiations can achieve better outcomes than human negotiations (Oliver, 1996; Bichler, 2000; Sandholm, 1999; Choi et al., 2001; Chavez and Maes, 1996) as software agents are superior to human agents in dealing with complex problems.[9] The argumentation in favor of this assumption bases on the one hand on the effects of the delegation of negotiation task to software agents, as human agents often lack the experience and capabilities in negotiating or the willingness to negotiate (Maes et al., 1999; Choi et al., 2001; Lomuscio et al., 2003).[10] On the other hand it is supported by results of simulation studies that revealed that software agents achieve (nearly) Pareto-optimal results – while humans often do not achieve such results in experimental and field studies as discussed above.[11]

That automated negotiation is only used if this results in a greater net benefit – as the difference between the utility of outcome and the costs of the transaction mechanism – than traditional negotiation between humans is not only true for this specific comparison, but also holds for comparisons of and choice between transaction mechanisms in general. The higher costs associated with negotiation compared to simpler transaction mechanisms have to be compensated by correspondingly better outcomes so that not only a net benefit exists but that this net benefit also exceeds that of alternative transaction mechanisms. While in electronic business information, orders, and payments can be handled electronically, humans are still in the loop in all stages of electronic business and electronic business transactions are largely conducted through web catalogs – offering fixed package take-it-or-leave-it offers – or online auctions – over single issues (the price of the good or service in most of the cases) – rather than by negotiations (Deveaux et al., 2001; Lomuscio et al., 2003; Choi et al., 2001). Due to lower transaction costs, better outcomes, or both, the net benefit resulting from automated negotiations can be greater than

[9]Complexity in this context can only refer to the complexity of the negotiation problem as determined by the number of issues and the nature of options within these issues as well as the preferences of the negotiators over the possible agreements. Automated negotiation is not an adequate approach for socially complex problems, where relationships between the negotiators and emotions play an important role. In this case human agents as empathic beings will outperform software agents. Furthermore, improvements of outcomes can only be evaluated in economic dimensions of negotiation outcomes in automated negotiation, qualitative evaluations like satisfaction with the agreement and the negotiation process or feelings about the outcome's fairness are not applicable to automated negotiation though they play an important role in evaluating the outcome of negotiations between humans.

[10]There are not only benefits of using automated negotiation, one must not neglect the reduction of the human users' power automated negotiation unavoidably causes. It increases negotiation speed but simultaneously decreases channel richness and the problem dimensions that can be dealt with and therefore decreases the power of patient, informed, and creative negotiators. Automated negotiation has to compensate for these power reductions by better outcomes, lower transaction costs, or a combination of both so that a higher net benefit remains. (Blecherman, 1999).

[11]However without an actual comparison of the performance of automated negotiation and human negotiation for a given negotiation problem a direct verification of this assumption is not possible. Only few studies do such comparisons and come to puzzling results, finding that human negotiators reach better result than software agents or at least equivalent ones (Oliver, 1996; Bosse and Jonker, 2005).

that of alternative transaction mechanisms currently in use instead of negotiation between humans – like for example web catalogs or auctions used for low-involvement goods where the additional benefit of traditional negotiations would exceed its transaction costs – and therefore replace them (Maes et al., 1999; Bichler, 2000; Bichler and Segev, 2001; Kontolemakis et al., 2004). Furthermore, through automation one can handle business volumes deemed impossible without (Nwana et al., 1998; Kontolemakis et al., 2004; Lomuscio et al., 2003) as more opponents can be reached and dealt with and more information can be gathered (Deveaux et al., 2001). Given the increasing importance of the World Wide Web and the increasing share of electronic business on total transactions in combination with this argumentation – of new types and larger volumes of transactions possible to handle via negotiation through its automation – it is not surprising that a number of models for automated negotiation have already been proposed for business-to-business and business-to-customer electronic business (Fatima et al., 2004). We agree with Blecherman (1999) who argues: *'With both the shopping mall and the auction house being replicated electronically, can the electronic bazaar, complete with haggling, be far behind?'* (Blecherman, 1999, p.168).

1.3.2 Automated negotiation in autonomous systems

Problems of coordination and cooperation are not unique to economic interactions between humans, but exist at multiple levels of activity in a wide range of populations. For example animals interact – with limited language – to cooperate with each other and from communities, and likewise coordination and cooperation is a vital topic in machine-interaction (Rosenschein and Zlotkin, 1994; Kraus, 1997). Computers make more and more decisions in a relatively autonomous fashion, some of them embedded in autonomous systems i.e. in concert with other computers. These systems are not restricted to business interaction but require mechanisms for cooperation and coordination for many domains like electrical power management, airport landing coordination and evasion maneuver coordination for airplanes, workflow coordination between warehouse robots, freight transportation systems, and supply chain management systems, grid computing and data allocation in information servers, or meeting scheduling – to name only a few of such autonomous systems studied or proposed in recent literature (Rosenschein and Zlotkin, 1994; Kraus, 1997; Fink, 2004; Wollkind et al., 2004). Sometimes software agents control resources and can make decisions independent of other software agents, so no coordination is required even if cooperation with other software agents may improve the individual software agent's behavior or the overall behavior of the system they form. In other domains software agents cannot go about their business without coordination with other software agents as this interaction – to avoid interference or gain access to resources they control – is necessary to achieve their objectives. In this case software agents need to determine a mutually beneficial course of action acceptable to all participants over which neither has direct control due to the software agents' self-determination that allows them to decide what to do, as well as when and under what conditions their actions should be performed (Rosenschein and Zlotkin, 1994; Kraus, 1997; Jennings et al., 2001).

However, it seems that negotiation as a means of coordination and cooperation only takes place among humans, and that interactions between machines occur without such negotiation processes but in restricted environments with often simple and rigid rules governing the interaction –

which themselves of course are result of human negotiation. Though their general applicability for coordination and cooperation between software agents in autonomous systems, negotiations or other sophisticated interactions like bidding or voting to a large extent not occur among computers (Kraus, 1997). However, simplicity and rigidity of currently used interaction rules can easily result in outcomes that are undesirable from everyone's point of view. If the interaction rules are primitive and not allow to exploit opportunities of cooperation computers will act inefficiently, making wrong or bad decisions the users will suffer from (Rosenschein and Zlotkin, 1994). Automation of the negotiation mechanism removes human interference and makes it applicable for coordination and cooperation of software agents in autonomous systems that also operate without human interference. This, as argued, could improve existing systems' overall performance on the one hand, as the currently used often simplistic and rigid rules of interaction are replaced, and on the other hand enables the development of novel systems necessitating such high level coordination mechanisms.

1.4 Research question and method

Given the youth of the field it is not surprising that operative systems are not available yet – with exception of some experimental systems developed for academic purposes and applied under controlled conditions in experiments.[12] Due to this lack of operative systems for automated negotiation, researchers relay on the one hand on analytical approaches, applicable for very specific problems only, and on the other hand on simulation, for investigation of more complex and realistic settings, in evaluating possible system configurations for automated negotiation systems. Besides its applicability to complex problems, not solvable by means of analytical approaches, the use of simulation is furthermore appropriate as (i) the risks of implementation without testing are to high, and even simple algorithms having acceptable behavior in a specific restricted situation might be unpredictable in a more liberal environment (Bichler and Segev, 2001; Henderson et al., 2003), (ii) it is necessary to demonstrate the performance of software agents by some means to create user acceptance (Jennings et al., 2001), and finally (iii) software agent strategies from computer simulations can easily be used in operative systems necessitating only minor modifications and adaptations as they both are program code.

The central aim and research question of this dissertation is the suggestion and evaluation of system designs for automated negotiation in electronic business and autonomous systems. To address this research question we follow a three-step approach:

- *Literature review:* In a first step we perform a systematical literature review of studies dealing with automated negotiation and its simulation – identified by a keyword search in major scientific databases – to determine the state of the art in the domain of automated negotiation on the one hand, and to identify deficiencies of existing approaches for the practical use of their insights in implementations of operative systems.

[12]Examples for such relatively simple experimental systems applied to study automated negotiation include `Bazaar` (Zeng and Sycara, 1998), `Kasbah` (Chavez and Maes, 1996), `ADEPT` (Faratin et al., 1998), or `Tete-a-Tete` (Guttman et al., 1998) – consult Guttman et al. (1998) for a review.

- *Model development:* Building on this review we develop a conceptual model for an automated negotiation system, where we address the shortcomings identified by incorporating concepts from negotiation and bargaining literature that were not used in simulations yet – especially generic offer generation and concession strategies from negotiation literature and non-alternating protocols with possibilities for strategy interruption from game theoretic mechanism design literature.

- *Simulation and evaluation:* In a last step we implement the conceptual model as a simulation program and compare different system configurations for various measures of the outcome of negotiations. Here we use negotiation problems derived from negotiation experiments between humans as input to the systems, which not only enables comparisons of system configurations among each in a realistic environment, but also allows comparison of the performance of automated and traditional negotiation.

1.5 Structure

The structure of the remainder of this dissertation, after this introductory chapter providing the motivation, research questions, and the methods to tackle them, is provided in Figure 1.1. Chapters 2 and 3 give the methodological and thematical background for simulation of automated negotiation. Chapter 2 describes the scientific technique of simulation and explains the general course of a simulation study, Chapter 3 on the other hand describes automated negotiation, the necessary components of systems for automated negotiation, and systematically reviews the literature relevant to the research question of this dissertation – describing the state of the art of simulation of automated negotiation as well as identifying deficiencies of existing approaches.

Chapters 4 to 6 then follow the outlined general course of simulation studies. Model development and the resulting conceptual model are covered in Chapter 4. Chapter 5 explains the experimental design, the dependent and independent variables, as well as the statistical methods used in analyses. The results of these analyses, which concern the comparison of the performance of different configurations of systems for automated negotiation among each other and to human negotiation, the effects of the single components of the system, their interactions, and a discussion of these results can be found in Chapter 6. Finally Chapter 7 concludes the dissertation in summarizing the study and its results, as well as outlining the limitations of this study and possible areas of future research.

Chapters 4 to 6 are accompanied by appendices, that present additional information. Appendix A, supporting the chapter on the conceptual model, provides information on the data used as input and benchmark for the system and the source code of the implemented simulation program. Appendix B provides the results of the tests accomplished to determine the minimal necessary number of replications of a simulation run to achieve stable results, and Appendix C provides additional tables summarizing results and statistical test for Chapter 6.[13]

[13]The appendix also contains abstracts of the dissertation in English and German language and a curriculum vitae of the author.

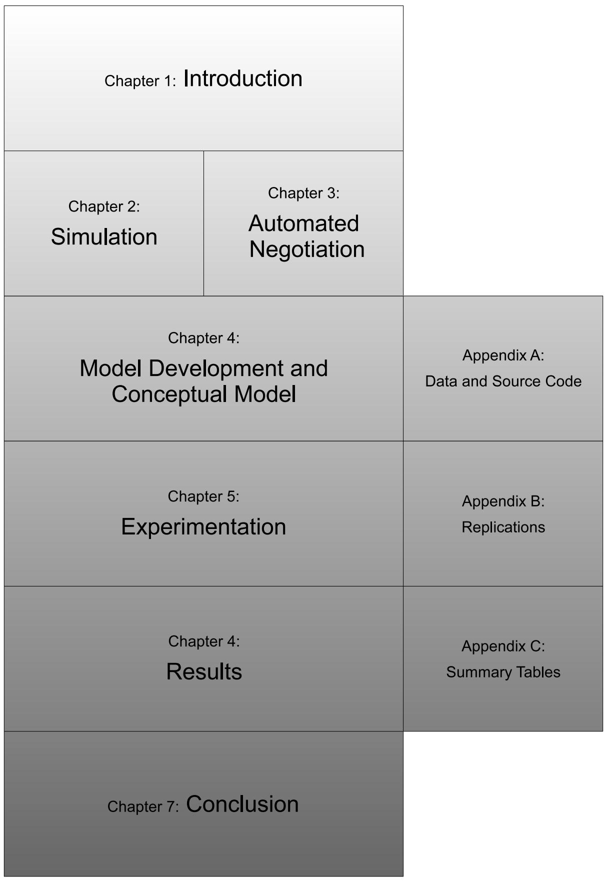

Chapter 1: Introduction
Chapter 2: Simulation
Chapter 3: Automated Negotiation
Chapter 4: Model Development and Conceptual Model
Appendix A: Data and Source Code
Chapter 5: Experimentation
Appendix B: Replications
Chapter 4: Results
Appendix C: Summary Tables
Chapter 7: Conclusion

Figure 1.1: Structure of the dissertation

Chapter 2

Simulation

Simulation is a scientific technique to investigate real-world systems in performing experiments on a computer. To study systems on a computer first the general system behavior and the system components' interactions have to be specified in form of mathematical or logical relationships, which constitutes a model of the real-world system of interest – or in the special case of agent-based simulation one specifies the individual behavior of the single agents in mathematical or logical form to investigate the emergent behavior of the system they constitute. This model itself is a system abstracting from the real-world system, but representing specific aspects of interest – depending on the goals and intentions of the simulation – and can be implemented as a simulation program that is understood by and runs on a computer (Law and Kelton, 1991; Pidd, 1992). If the model is a sufficiently accurate representation of the real-world system's aspects of interest and is correctly implemented in a simulation program, the results of experimentation with the simulation program permit conclusions about the real-world system i.e. the original (Page, 1991).[1]

The purpose of computer simulation in investigating a system can be (i) *system analysis* – the investigation of the behavior of an existing system if insights about the system's operations are required e.g. for comparison of different alternative policies or configurations for existing systems to chose the most appropriate –, (ii) *system postulation* – the formulation of hypotheses about the structure and internal relationships of a system and hypotheses testing by comparing the system's and the simulation's input-output relations –, and (iii) *system design* – the prediction of the performance and comparison of different system variants (with different specifications in controllable parameters) for non-existing systems before their actual implementation (Gordon, 1978).

However simulation is not the only method to study systems, but there are competing approaches that serve the same purpose. Besides using the simulation technique, systems can also be studied by means of experimentation with the actual real-world system or by analytically solving a model of the system. While experimentation with the actual system would cause no prob-

[1]To be implementable and executable on a computer, for a computer simulation all decisions must be predetermined by decision rules or strategies before the experimentation, as distinguished from simulation games where decisions are not fixed in advance but made by humans interacting with the computer program between its calculations (Müller, 1998).

lems of validity, such experiments are often costly, disruptive, and risky for the focal system. Experimentation with the actual system might also be difficult and time-intensive to conduct, ethnically questionable, not easily comparable due to changing environmental influence factors, and sometimes irreversible. The system's behavior might also be too slow or too fast thereby complicating experimentation. In these cases experimentation is better done with a model of the system rather than with the system itself (Law and Kelton, 1991; Page, 1991; Pidd, 1992). When using a mathematical-logical model to gain insights about a system the researcher need not rely on simulation techniques but could also derive an exact analytic solution from the model by mathematical methods such as algebra, calculus, or probability theory. This, however, implies that the model is simple enough to be mathematically traceable, which may require tremendous simplifications and rigid assumptions abstracting from the actual system of interest. If the system is too complex to realistically represent it for the study's purpose in an analytically solvable model, simulation of a more realistic model will be the better choice (Law and Kelton, 1991; Page, 1991; Pidd, 1992; Bichler and Segev, 2001).[2]

As a scientific technique simulation is a rather novel way to conduct research and can be used to incorporate the two established and classical methods of induction and deduction. While *induction* is the discovery of patterns in empirical data and *deduction* involves deriving consequences from specified assumptions, the simulation technique incorporates aspects of both. Like deduction it starts with a set of explicit assumptions, however unlike deduction it does not directly derive consequences from those assumptions, but instead generates data through experimentation that can be analyzed inductively. Unlike typical induction, however, the data analyzed comes from simulation experiments with a model of rigorously specified mathematical and logical relations rather than from direct measurement of the real-world (Axelrod, 1997). So while induction is appropriate to discover patterns in empirical data and deduction to derive consequences from assumptions computer simulation can be used to aid intuition (Axelrod, 1997). Just like Axelrod (1997) argues:

> Simulation is a way of doing thought experiments. While the assumptions may be simple, the consequences may not be at all obvious. The large-scale effects of locally interacting agents are called "emergent properties" of the system. Emergent properties are often surprising because it can be hard to anticipate the full consequences of even simple forms of interaction. (Axelrod, 1997, p. 24-25)

Arguments about the applicability of simulation for theory development by Davis et al. (2007)

[2] Especially the use of analytically solvable models to represent systems, when an adequate representation of the system for the research questions of the study calls for more sophisticated models, caused major criticism in recent years and a call for the use of alternative approaches such as simulation. Axelrod (1997) for example criticizes neoclassical economic models in which rational agents operating under powerful assumptions about the availability of information and the capability to optimize can achieve an efficient re-allocation of resources among themselves through costless trading. When agents use adaptive rather than optimizing strategies derivation of analytical solutions becomes often impossible. He argues that the rational choice assumption, dominant in economics and game theory, is not used because it is realistic or offers good advice to a decision maker but as it enables deduction and mathematical solution of models. The more realistic alternative to rational choice is adaptive behavior, where people may try to behave rationally but do not meet the requirements of information, foresight, and computational capacity rational models impose (Simon, 1955; Cyert and March, 1963). Such adaptive behavior can be achieved through learning at the individual level or through evolutionary mechanisms at the population level. However, the consequences of adaptive processes are often hard to deduce when many agents interact following rules with non-linear effects. In this case alternative methods such as simulation are necessary to investigate such more realistic models.

are in line with this potential to incorporate inductive and deductive methodology by means of using simulation techniques:

> Simulation is particularly suited to the development of simple theory because of its strengths in enhancing theoretical precision and related internal validity and in enabling theoretical elaboration and exploration through computational experimentation. In particular, simulation relies on some theoretical understanding of the focal phenomena in order to construct a computational representation. Yet simulation also depends on an incomplete theoretical understanding such that fresh theoretical insights are possible from the precision that simulation enforces and the experimentation that simulation enables. (Davis et al., 2007, p. 481)

Besides this application in scientific research for thought experiments and theory development, simulation is often also argued to be the last resort for investigating a system when realistic models are to complex to solve analytically by deduction or the collection of the necessary empirical data for induction is difficult or impossible, like e.g. in system design when studying non-existing systems (Law and Kelton, 1991; Pidd, 1992; Axelrod, 1997).

Beyond the academic area, simulation became popular from its application in real time and fast motion applications, first in the military sector (e.g. flight simulations for pilot training or combat simulations) and later in computer games (e.g. flight simulators or strategy games) (Liebl, 1992). Furthermore surveys indicate that the instruction in simulation techniques at universities is highly appreciated among students and practitioners, often ranked second only behind 'statistics', concerning its practical relevance for the later professional life (Law and Kelton, 1991).[3] As a scientific technique, however, simulation is most widely used in the areas of operations research and management science and proofed to be an useful and powerful tool for applications like designing and analyzing manufacturing systems, evaluating hardware and software requirements for a computer system, evaluating new military weapon systems or tactics, determining ordering policies for inventory systems, designing communication systems and message protocols for them, designing and operating transportation facilities – such as freeways, airports, subways, or ports –, evaluating structures and processes for service organizations – such as hospitals, post offices, or fast-food restaurants –, or analyzing financial or economic systems (Law and Kelton, 1991). Furthermore – and relevant for the topic of this dissertation as we simulate (computer) systems for automated negotiation – Maisel and Gnugnoli (1972) argue in accordance with other authors that simulation is the most potentially powerful, flexible, and adequate technique for all purposes of computer system evaluation in particular for the evaluation of new systems.

[3]Though their sources were quite old when their book was published – which again was some years ago – the assumption of Law and Kelton (1991), that the value and usage of simulation is increasing, still is valid. This is due to the fact that the forces that undermine the major drawbacks of simulation – which are the time-intensity of modeling and implementing large-scale and complex systems and simulation's high requirements of computer time and performance – still are present with the further development of simulation languages and decreasing costs of computing.

2.1 Simulation process

A typical simulation study passes through a number of phases with intermediate results as depicted in Figure 2.1.[4] Starting from an analysis of and data acquisition about the real-world system one can define a conceptual model, which then can be implemented as simulation program. For the correct accomplishment of the two steps modeling and implementation two controlling activities are necessary. Correct modeling is ensured by the validation of the conceptual model and the operational accuracy of the simulation program. On the other hand verification guarantees the correct implementation of the simulation program. Only after a verified simulation program based on a valid model is established this simulation program can be used in computational experiments to gain reliable simulation results. Modeling and programming, and the related control activities of validation and verification, are therefore two preconditions for the desired final experimentation with the simulation program (Law and Kelton, 1991; Pidd, 1992).

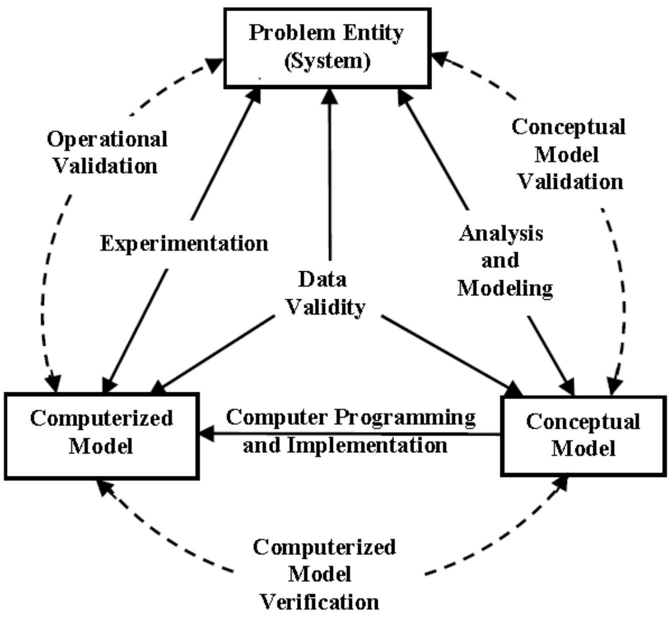


Figure 2.1: Simulation process (Sargent, 2005, p. 132)

After this general introduction of simulation and an overview over the general simulation process, the remainder of this chapter is structured according to the above mentioned phases and inter-

[4]Sargent (2005), who suggested this schema, defines its components and interactions as follows: *The problem entity is the system (real or proposed), idea, situation, policy, or phenomena to be modeled; the conceptual model is the mathematical/logical/verbal representation (mimic) of the problem entity developed for a particular study; and the computerized model is the conceptual model implemented on a computer. The conceptual model is developed through an analysis and modeling phase, the computerized model is developed through a computer programming and implementation phase, and inferences about the problem entity are obtained by conducting computer experiments of the computerized model in the experimentation phase. . . . Conceptual model validation is defined as determining that the theories and assumptions underlying the conceptual model are correct and that the model representation of the problem entity is 'reasonable' for the intended purpose of the model. Computerized model verification is defined as assuring that the computer programming and implementation of the conceptual model is correct. Operational validation is defined as determining that the model's output behavior has sufficient accuracy for the model's intended purpose over the domain of the model's intended applicability. Data validity is defined as ensuring that the data necessary for model building, model evaluation and testing, and conducting the model experiments to solve the problem are adequate and correct.* (Sargent, 2005, p.132-133)

mediate results. While the explanations in the following sections remain general to provide an overview, we will follow the procedure outlined below – making and justifying decisions between available options where necessary – to arrive at a simulation model for automated negotiation systems in Chapter 4.

2.2 System

The real-world not only consists of an accumulation of single objects, but objects are connected and interact through a network of relations. Whenever a subset of these objects and relations that act and interact toward the accomplishment of some logical end is restricted and studied, this subset can be conceived as a system (Law and Kelton, 1991). The structure and behavior of such systems is often not immediately comprehensible. The better the structure and the behavior of a system are understood, the more knowledge is available about the features of the system components (detailed knowledge) and about the interactive behavior of these components (structural knowledge), the higher the opportunities to successfully make purposeful interventions, as the effects of such interventions and their feasibility can be estimated and evaluated (Page, 1991).

The investigation of complex systems is the objective of the interdisciplinary approach of system analysis. System analysis investigates the structure and behavior of systems with the aim to provide decision makers with information about the consequences of different interventions in the system. Though the concept 'system' is very general, and applicable to ecological, economic, and many other phenomena, systems share basic features independent of their domain. Domain independent investigation of these basic features of the general structure and behavior of systems (see Figure 2.2) is the objective of system theory. System theory establishes the frame for system analysis as it is helpful to resort to general principles about the structure and behavior of systems when analyzing a specific system (Page, 1991).

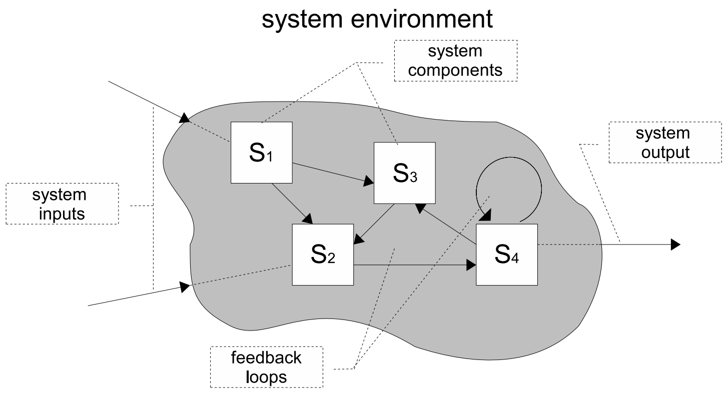


Figure 2.2: Notation in system theory (Page, 1991, p. 3)

Entities – also called objects or components – are parts of a system, which can be individually identified and processed and are not further divisible – or need not be further divided for the purpose and intention of the study. Entities can be classified in permanent entities, e.g. machines or humans, that remain in the system the whole time, and temporary entities, like jobs that pass through the system and cease to be individually of interest once they left it. From a different perspective one could also classify entities in those that process others and those that are processed by others, where the former then are active entities while the later constitute the set of passive entities (Pidd, 1992). To these entities attributes, which convey extra information about the entity, can be attached for closer specification of their features. Especially for simulation such attributes are useful for a variety of purposes, like to subdivide and distinguish entities of a class or to control the behavior of the entity (Pidd, 1992). If these attributes change over time they are called state variables. In a production system, for example, a specific machine could be conceived as a permanent (and active) entity with an occupancy attribute, which can take the values 'occupied' or 'idle', so that the occupancy attribute is a state variable of the system. The values of all state variables of a system at a specific point in time determine the system state. If the system state changes over time these changes represent the dynamic behavior of the system. As mentioned, entities are not isolated but are in relationship with other entities. Interactions, where the state of one entity causally influences the state of another entity, are the most important type of relations from the point of view of system analysis, as they determine the behavior of the system. The complexity of a system depends on the number of entities in the system and the number and type the relations between entities (Page, 1991).

The choice of the system of interest simultaneously fixes the border of the system in determining the elements and relations the system consists of i.e. what is inside and what is outside the system – where all left outside is the system's environment. If the system is not a closed one it has at least one relation to its environment – inputs from the environment to the system or outputs from the system to the environment. Though in a simulation study all inputs are controllable (as one can easily change their values) no matter whether or not they can in reality be set or changed at will (Law and Kelton, 1991; Kelton, 1999), for evaluating alternative policies of decision makers or alternative system configurations it is often reasonable to distinguish between controllable and uncontrollable inputs and focus advice on the former rather than the later (Kleijnen, 1987). Controllable inputs, or system parameters, are those factors that can be changed by the decision maker, like for instance priority rules for queues in front of a machine could be set to FIFO, LIFO, etc. On the other hand uncontrollable inputs, or environmental inputs, are those factor that cannot be influenced by the decision maker, like the interarrival time of jobs or their duration. Another way to categorize system inputs is the classification in quantitative factors – like again interarrival time of jobs, number of servers, probabilities of different job types – and logical/structural input factors – like whether failure or feedback loops are present, and whether a queue is processed FIFO or shortest-job-first (Kelton, 1999).

Entire systems can be classified according to the entities they consist of into in natural, artificial or technical, and – if humans are part of the system – social systems. Concerning the interaction of the system with its environment one can distinguish open and closed systems, where open system have at least one interaction with the system's environment (either system input or system output) while closed systems are independent of their environment as they have no such interactions. Furthermore systems can be divided into static and dynamic systems according to the system's behavior over time – i.e. whether or not the system state changes over time (Page, 1991).

2.3 Modeling and validation

The purpose of the modeling phase is to finally derive a model of the system of interest, i.e. a system representing another system appropriately for the focus of the study, which can be investigated and from which conclusions about the real-world system can be drawn. For simulation this implies that a model has to be developed that can be implemented as a computer program so that computer experiments can be conducted. Modeling therefore involves a reduction of complexity of the original system through either ignoring or neglecting some of the structure and behavior of the real-world system (i.e. its entities – where the boundaries of the system are set –, the entities' attributes, or the relations between entities) and approximations, in a way that important aspects of the real-world system for the purpose and intention of the study, remain appropriately represented in the model. Types of such approximations include (Bratley et al., 1987):

- *functional approximations:* like replacing nonlinear functions in the system by simpler (piecewise) linear ones. The simpler functions then should be close to the empirical ones at least in the regions the system is likely to operate.
- *distributional approximations:* unknown or approximately known empirical distributions of values are replaced by simpler theoretical ones such as normal or exponential distribution. The most extreme case would be to replace a random variable by a constant.
- *independence approximations:* assuming for simplification that various components (like e.g. random variables) are statistically independent.
- *stationarity approximations:* assuming that parameters that actually change are constant and do not vary over time, which can be justified if changes are limited and negligible, however, many phenomena are non-stationary so that the notion of steady state is not appropriate.
- *aggregation:* several of something are treated as one. Forms of this type of approximation include time intervals that are treated as single periods (temporal aggregation); divisions, firms, product lines etc. that are treated as one (cross-sectoral aggregation), several resources that are treated as one like e.g. a series of machines as one large machine, or computer systems with two CPUs as a computer system with one CPU of twice the speed (resource aggregation), different entities are grouped according to similar characteristics e.g. individuals of different age are grouped in age ranges with common death rate or working capacity (range aggregation).

As simulation is increasingly being used in problem solving and to provide advice for decision making, individuals affected by decisions based on simulation and decision makers using the information obtained from simulations are understandably concerned with whether the conceptual model, the simulation bases on, and therefore its results are 'correct'. This concern is addressed by validation, which is the substantiation that a conceptual model within its intended area of application possesses a satisfactory range of accuracy (Sargent, 2005).[5] Therefore, with a 'valid'

[5]A topic closely related to model validation is *model credibility*, which is concerned with developing in (potential) users the confidence they require in order to use a conceptual model and the results derived from it, through

model the decisions made basing on this model should be similar to those that would be made basing on physically experimenting with the system if this were possible. On the other hand, if a conceptual model is not valid, then any conclusions derived from the model will be of doubtful value only (Law and Kelton, 1991). We therefore agree with (Law, 2005, p. 24), who states that: *'[v]alidation should and can be done for all models, regardless of whether the corresponding system exists in some form or whether it will be built in the future'.* Model validation efforts can be performed by either the (i) modeling team itself, which is the most common approach, (ii) the user of the model, which also positively influences model credibility, or (iii) a third party other than the modeling team and the model user having expertise in the domain of the simulation. These experts then can either evaluate validation efforts of the modeling team and/or perform validation themselves.

Validity is a gradual term as there is no absolute 'yes' or 'no' answer to the question whether a model is valid, but the answer can only be a degree of validity, i.e. a conceptual model can serve its purpose better or worse (Liebl, 1992). Furthermore it is important to note that there exist no general theory about validation, but determining whether a conceptual model is valid is specific to the type of the model and the aim of the study. Due to the diversity of applications and types of models it is impossible to determine one standard procedure or one standard validity test.[6] Validation methods available for the investigation of existing real-world system – in case of system analysis or system postulation – for example cannot be used for validation in studies investigating systems that do not yet exist – the case of system design. Descriptive studies (system analysis) must be much more precise concerning the input and output relations while prescriptive simulations only have to correctly discriminate different system configurations according to their performance so that the concrete performance measures are not necessarily of importance in this later case (system postulation and design) (Liebl, 1992). Though a set of methods for validation is available[7], there exist no specific guidelines which approach to apply for a certain model and purpose of study (Liebl, 1992; Sargent, 2005).

The exercise of validation is not an one-time but actually a reoccurring task during the whole simulation process. It has to be performed when taking observations to determine the real system, when abstracting from the system to arrive at a conceptual model, when controlling whether the conceptual model is a good representation of the system for the intended application – and this for all versions of the model since there might be some intermediate ones before one finally ends up with a satisfactory valid conceptual model (Liebl, 1992; Sargent, 2005). As illustrated in Figure 2.1 there exist three steps during the simulation process, where validation plays an important role, first concerning the data used to build the model and to run the simulation program, which is referred to as *data validity*, second concerning how good the conceptual model

computer experiments, and which is achieved when the conceptual model and its results are accepted by the user of the model to be valid. Though often neglected in methodological discussions model credibility is maybe as important as validation for the actual implementation of conclusions and results of simulation (Law and Kelton, 1991; Sargent, 2005).

[6]General guidelines, however, are provided for example by the multistage validation approach (Naylor and Finger, 1967; Law and Kelton, 1991), which involves (i) developing a model with high face validity in building on knowledge from experts and existing theory, (ii) testing the assumptions of the model, like distributions of input variables, empirically, and (iii) determining the representativeness of the simulation output data by statistical comparison with the real-world system's output data. However this general approach is inappropriate for studies that investigate systems that do not exist yet (system design studies) as the last step cannot be performed due to the lack of a real-world system.

[7]Consult for example Balci (1994) or Sargent (2005) for an extensive list of approaches for validation of conceptual models for simulation.

represents the real-world system for the focus of the study, which is called *conceptual validation*, and third *operational validation*, which determines how close the outcomes of the simulation are to the outcomes of the real-world system.

2.3.1 Data validity

Gathering empirical data is of great importance in simulation studies as valid data about and from the real-world system is needed for both, the development of the conceptual model – where it is necessary to gain insights about entities, their attributes and their relations from observations and discussions with experts in the domain – and as input to run the simulation program (see Figure 2.1). However, it is also often difficult, costly, and maybe impossible to obtain appropriate, accurate, and sufficient data for modeling and validation and there is nothing that can be done about it other than maintaining scientific standards for data acquisition and storage (Law and Kelton, 1991; Sargent, 2005). Once collected empirical data on input variables can find different ways into the simulation (Law and Kelton, 1991):

1. The data values themselves are directly used in the simulation. For example, if the data represent service times, then one of the data values is used whenever a service time is needed in the simulation – this approach is called trace-driven simulation.

2. The data values themselves are used to determine an empirical distribution function. If these data represent service times, whenever a service time is needed in the simulation one is sampled from this empirical distribution.

3. Standard techniques of statistical inference are used to 'fit' a theoretical distribution – e.g. normal, exponential, or Poisson distribution – to the data and to perform hypothesis tests to determine the goodness of fit. If a particular theoretical distribution with certain values for its parameters is a good model for the service-time data, then one can sample from this distribution when a service time is needed in the simulation.

Law and Kelton (1991) state that the above list is in increasing order of desirability, i.e. that an appropriate theoretical distribution is preferable over an empirical distribution and trace-driven simulation for several reasons. Trace-driven simulation can only reproduce what has happened historically and seldom enough data is available for the desired amount of simulation runs, so that using an empirical distribution to sample random input variables is preferable to using the observed data itself as for continuous data any value between minimum and maximum of the observed data points and any desired number of values can be generated. If a well fitting theoretical distribution can be found for the observed data this is preferable to an empirical distribution to avoid irregularities especially when only a small number of observations is available, as values outside the range of observed data can be generated – i.e. extreme events – for the simulation, and as a theoretical distribution is a compact way of representing a set of data values by only few parameters, whereas considerably more space is needed to represent and store an empirical distribution on a computer. Furthermore, there might exist compelling reasons for using a certain theoretical distribution, though in this case observed data should be used to provide empirical support for the particular distribution chosen.

2.3.2 Conceptual validation

Conceptual validation also called 'white-box validation' (Pidd, 1992) or 'face validation' (Law and Kelton, 1991) concerns determining that the theories and assumptions underlying the conceptual model are correct, that the model is a good representation of the system of interest, and that the model's structure, logic, and mathematical and causal relationships are reasonable for the intended purpose of the conceptual model (Sargent, 2005). Conceptual validation therefore involves controlling the static and dynamic logic of the conceptual model for correctness. The static logic concerns the entities and their attributes, while the dynamic logic concerns the rules that govern the behavior of the entities which strongly affect the behavior of the system as a whole, and therefore the outcome of the simulation, which should mimic those which govern the operations of the entities in the real-world system (Pidd, 1992). In performing conceptual validation use of all available information about the system should be made, where information can come from: (i) conversations with experts in the domain, (ii) observation of the focal system or of similar systems, (iii) consideration of existing theory and relevant results from similar (already validated) simulation models, and (iv) own experience and intuition (Law and Kelton, 1991).

Conceptual validation assumes that the internal structures of both the conceptual model and the real-world system are well understood and takes place during the model development where it is applied to all of the entities and all of their relations. Here not the predictive power but the detailed internal workings of the model are focus of the validation (Pidd, 1992). Aspects to be considered for example cover the application of random numbers, which only should be used when the process which produces a behavior cannot be understood in any deterministic sense – or is not of relevance given the purpose of the conceptual model. In this case a theoretical distribution for these random numbers close to the observed data should be chosen. Moreover the correctness of conditions for the execution of activities ('if-then' rules), etc. has to be examined.

Methods to perform conceptual validation include the use of flow charts and graphical models or a set of equations to represent the conceptual model, structured walk-throughs and discussions of all submodules with domain experts, animation of the model behavior and tracing of entities and variables, as well as degenerate and extreme condition tests. A structured walk-through is a discussion of the model step by step with experts in the field, where the discussions only move on to the next step in the conceptual model when all concerns and questions of the audience were addressed. Animation is the graphical and dynamical display of a model's operations, like the move of parts through a factory, over time. Tracing is a technique that collects the intermediate values of variables during a model run or the state changes of entities on their way through the model to determine if the model's logic is correct. In degenerate tests the degeneracy of the model's behavior is tested by appropriate selection of input values and internal parameters, for example it can be tested whether the average number of jobs waiting in a queue of a single server continues to increase over time when the interarrival rate is larger than the service rate. Extreme condition tests investigate if the model structure and output is plausible for any extreme and unlikely combination of levels of factors in the system, for example if the in-process inventories are zero then the production outputs should be zero too (Pidd, 1992; Sargent, 2005).

2.3.3 Operational validation

Operational validation is the task of establishing that a conceptual model's output data closely – means with the accuracy required for the model's intended purpose – resembles the output data that would be expected from the actual system, and is argued to be the most definitive test of a simulation model's validity (Sargent, 2005). Operational validation therefore involves comparing output data from the existing system with the proposed model's output, if the two sets of data compare favorably, then the conceptual model is considered valid (Law and Kelton, 1991). Thus, operational validation is a form of black-box validation that ignores the detailed internal workings of the model but is only interested in whether or not the model's output accurately reflects that of the real system. This type of validation is mainly performed by statistical comparisons of the model's and the real system's outputs and examines the predictive power of the model, i.e. whether it adequately predicts how a given system actually behaves and would behave under certain conditions (Pidd, 1992; Kleijnen, 1995).

An appropriate approach to perform operational validation is the correlated inspection approach (Law and Kelton, 1991; Pidd, 1992) – also called trace-driven validation (Kleijnen and van Groenendaal, 1992): *'In particular, it is recommended that the system and model be compared by driving the model with historical system input data (e.g., actual observed interarrival times and service times), rather than samples from the input probability distributions, and then comparing the model and system outputs. Thus, the system and the model experience exactly the same observations from the input random variables, which should result in a statistically more precise comparison.'* (Law and Kelton, 1991, p.316). When using the correlated inspection approach input data from the real-world system is used to run the simulation program and then the corresponding outputs of both the real-world system's (historical) and the simulation program's outputs are compared as illustrated in Figure 2.3. It is better to use the original input data rather than random input values – from a theoretical distribution fitted to the empirical data – as otherwise the effects of random input variables on the one hand and the conceptual model on the other hand could be mixed. This compounding of effects could result in wrongly rejecting an actually valid model due to output differences induced not by an inappropriate model but by the random input numbers used (Law and Kelton, 1991).

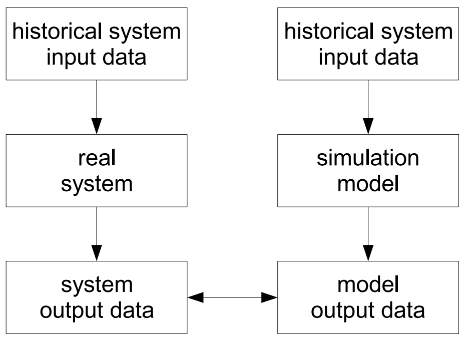


Figure 2.3: Correlated inspection approach (Law and Kelton, 1991, p. 317)

The resulting output data of the system and the model then can be compared either qualitatively by Turing tests – individuals knowledgeable about the operations of the system being modeled are asked if they can discriminate between system and model output and the model is considered valid if they cannot – or by plotting and visual inspection (Kleijnen, 1995; Sargent, 2005), or quantitatively in using statistical tests of the equality distributions for paired samples, like the parametric t-test or the non-parametric Wilcoxon signed-rank test (Law and Kelton, 1991), or by ordinary least squares regression (Kleijnen, 1995). Concerning the statistical testing of correspondence of system output and model output it has to be kept in mind, that due to approximations made in the conceptual model it is likely that the model and the real system do not have identical output and that therefore statistical tests could come to the end that outputs are different, though the model can be useful nevertheless (Bratley et al., 1987; Law and Kelton, 1991). Even if there are differences – which clearly must not be significant enough to affect the conclusions derived from the model – the greater the commonality between the outcomes of the system and the conceptual model the greater the confidence in the model (Law and Kelton, 1991). Further problems of statistical testing arise as the gathered data on real system output is often non-stationary and autocorrelated, which makes it difficult to find appropriate statistical tests (Law and Kelton, 1991).

Assume observations r_i of the output of the system and s_i of the output of the simulation of the conceptual model are available for same inputs i $(i = 1, \ldots, n)$. Then the t-test for paired samples can be used to analyze whether the distributions of the two samples are equal (Law and Kelton, 1991; Kleijnen, 1995), in calculating the paired differences $d_i = r_i - s_i$ and the according t statistic (2.1).

$$t_{n-1} = \frac{\overline{d} - \delta}{s_d/\sqrt{n}} \tag{2.1}$$

In (2.1) $\overline{d}$ is the average and s_d the standard deviation of d_i i.e. the average of the n differences between the system and the model output and its standard deviation. If for the hypothesis $H_0 : \delta = 0$ the t-statistic is significant we reject the model otherwise the means are practically the same so the simulation model is considered valid.

Alternatively, Kleijnen (1995) proposes to use ordinary least squares regression for the regression function (2.2) to test hypotheses about the correspondence of the system and the model, by estimating intercept β_0 and slope β_1 for the system output s and model output r for the same historical input data. This approach takes into account the positive correlation of outputs not just the equality of their means.

$$r = \beta_0 + \beta_1 s \tag{2.2}$$

The most stringent test of operational validity using ordinary least squares regression would be to test hypothesis (2.3), which implies not only identical means of the system and model responses but also their positive correlation. Testing this composite hypothesis involves computing the sum of squared errors for the reduced (without the hypothesis) and full (with the hypothesis) regression model and comparing these two values. A significantly high F statistic indicates that the hypothesis should be rejected and the simulation model is not valid (Kleijnen, 1995). A less stringent validation requirement would be that simulated and real responses do not necessarily have the same mean, but they are positively correlated, which makes sense when the model is used to predict relative response rather than absolute response, the according hypothesis for validation then would be (2.4). To test hypothesis (2.4) a t statistic can be used and the the null-hypothesis – that the responses are not correlated or negatively correlated – is rejected and therefore the model accepted as valid if there is strong evidence that the simulated and the real responses are positively correlated.

$$H_0 : \beta_0 = 0,\ \beta_1 = 1 \tag{2.3}$$

$$H_0 : \beta_1 \leq 0 \tag{2.4}$$

Though operational validation is possible only for existing systems where the necessary system output, that is compared to the model output, can be acquired, a way to achieve valid models for different system configurations is to model the existing system and validate this model and then change the model to reflect proposed changes in the system, assuming that this does not undermine model validity. So for existing systems where the aim of the simulation study is to gain new insights about the systems operations or to compare different policies or configurations for the system to determine and implement the best, both types of validation conceptual and operational validation can be performed as existing data on the system's output can be compared to the output of the simulation of the conceptual model. However, when completely new systems are studied by simulation only the components of the simulation model and their links can be validated, in the hope that the final result still remains satisfactory. For the validation of these components and their relations only conceptual validation is applicable as there exists no real system and therefore no real system's output against which one could compare the output from simulations of the conceptual model (Pidd, 1992). There are, however, some sources of output data against which the model's output can be compared, like for example already validated simulation models or results of analytic models for simpler problems (Sargent, 2005).

2.4 Conceptual model

A model is a purpose oriented, simplified representation of an real-world system and therefore a system too (as can be seen from Figure 2.4). In the modeling phase the system is abstracted to a model in representing the entities and relations – important for the intention and purpose of the simulation study – in a changed way. For conceptual models that are studied by simulation through computer experimentation this means transformation in a form that can be understood by a computer i.e. in form of procedures and mathematical or logical relations. The aim of studying a conceptual model is to gain information and insights about the original system it represents (Page, 1991; Müller, 1998). Conceptual models for simulation can be determined verbally, graphically by flowcharts, or mathematically by equations and logical rules (Law and Kelton, 1991), the documentation of the conceptual model, besides its implementation in a computer program is part of the simulation process.

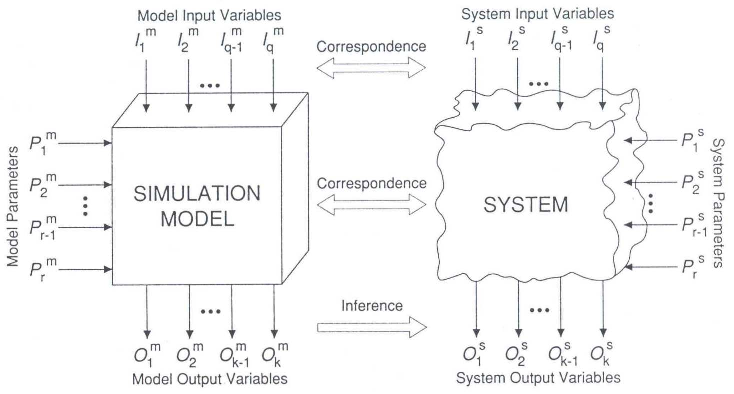

Model Input Variables
I_1^m I_2^m I_{q-1}^m I_q^m
Correspondence
System Input Variables
I_1^s I_2^s I_{q-1}^s I_q^s
Model Parameters
P_1^m P_2^m P_{r-1}^m P_r^m
SIMULATION MODEL
Correspondence
SYSTEM
P_1^s P_2^s P_{r-1}^s P_r^s
System Parameters
Inference
O_1^m O_2^m O_{k-1}^m O_k^m
Model Output Variables
O_1^s O_2^s O_{k-1}^s O_k^s
System Output Variables

Figure 2.4: Model and system characteristics (Balci, 1994, p. 122)

Models can be classified along diverse dimensions: (i) the method used for their investigation – separating analytic models for analytic solutions and simulation models for simulation –, (ii) the medium of representation, which leads to material or immaterial models[8], (iii) if and how their states change over time[9], and (iv) their purpose[10] (Page, 1991).

The most important distinction for models, when applied for simulation, is the way in which the system states change over time, as this builds the basis for two quite distinct simulation approaches i.e. discrete-time simulation and continuous time simulation – also called system dynamics.[11] As automated negotiation belongs more to the realm of systems with discrete state changes over time, which is discussed in more detail in Chapter 4, and due to the major differences between these two simulation approaches we will focus on discrete-time models.[12] Law and Kelton (1991) call models studied by means of simulation 'simulation models' and distinguish such models depending on: (i) the concept of time used, (ii) the use of probabilistic components, and (iii) status transitions, into static versus dynamic, deterministic versus stochastic, and discrete versus continuous simulation models:

[8]Where immaterial models can be further subdivided depending on whether they are formal or informal and whether they are written or drawn into: (a) informal verbal descriptions (e.g. natural language models), (b) informal graphical descriptions (e.g. flow charts), (c) formal mathematical descriptions (e.g. equation systems), and (d) formal graphical descriptions (e.g. petri-nets).

[9]In static and dynamic models, where dynamic models can be further subdivided depending on how they change over time and their ambiguousness into: (a) continuous-time deterministic, (b) continuous-time stochastic, (c) discrete-time deterministic, and (d) discrete-time stochastic models.

[10]Separating descriptive – to describe the behavior of the system –, prognostic – to forecast future system output under alternative assumptions–, decision support – to evaluate alternative policies and system configurations –, and optimizing models – to derive a system configuration that maximizes or minimizes some target function.

[11]Furthermore note, that with respect to the other classification criteria, a simulation program is an immaterial, formal, and written (in computer code) model, while the conceptual model can be any kind of immaterial model. Both the conceptual model and the simulation program can serve any of the mentioned purposes.

[12]In general system dynamics simulations are common in natural science and best modeled using differential equations to model relationships between dependent and independent variables – usually time – so that the state of the system at some point of time in the future can be calculated from a known initial configuration (Bratley et al., 1987). A detailed discussion can be found for example in (Pidd, 1992, part III). A well-known example are ecological 'predator-prey' models, where the population size development of both, the predator and the prey, is calculated from the initial population sizes and birth/dead rates for a particular point in time by numerically integrating the differential equations (Law and Kelton, 1991).

- *(i) Concept of time – static vs. dynamic simulation models:* In static simulation models the system representation is provided for one particular point in time only, while dynamic simulation models represent the system as it evolves over time.

- *(ii) Probabilistic components – deterministic vs. stochastic simulation models:* Deterministic simulation models have no probabilistic components, so that the output of the simulation given a certain input is always the same. So simulation models that experience random inputs and operate them deterministically are classified as deterministic simulation models, while simulation models with inherent stochastic elements that govern the operations belong to the class of stochastic simulation models.[13]

- *(iii) Status transitions – discrete vs. continuous simulation models*: If the simulation model is dynamic – i.e. changes over time – a distinctive feature to further differentiate types of simulation models – which is important to the field of simulation as it discriminates the two major classes of simulation discrete and continuous or system dynamics simulation – is the nature of state transitions in the simulation model, measured by the changes in its state variables – the variables used to describe a system at a particular time – over time. In discrete simulation models the state variables change instantaneously at separate points in time (an example of a discrete simulation model would be the queue in front of a bank counter – where the length of the queue would be a state variable of the simulation model – that changes only if a customer is finished or a new enters the bank). In continuous simulation models state variables change continuously with respect to time (here an example would be an airplane moving through the air where the position, velocity, and height – as state variables – change continuously over time).

Though the distinction between discrete and continuous simulation models is straightforward given the operations of the simulation model, it is not easy at all to decide which type of model to use for studying a given real world system, as systems in general are neither purely discrete nor purely continuous, but in most cases combine both aspects. For discrete systems, such as the production, transportation, and logistics systems that are studied in operations research and management science and where changes in the state of a system mainly are due to certain discrete events, discrete-event simulation is appropriate. On the other hand in physical, biological, and medical systems studied in the natural sciences the continuous change of these systems over time, following physical or biological rules, can be of particular interest and in this case such systems are best studied by system dynamics simulation (Page, 1991). However, discrete events may cause a discrete change in the value of an otherwise continuous state variable or cause the relationship governing a continuous state variable to change at a particular time. Furthermore, continuous state variables reaching a threshold value may cause a discrete event to occur. In such cases it has to be decided which type of simulation model is the more appropriate for the focal system and the purpose of its analysis or if a combination of both classes is a viable approach (Law and Kelton, 1991; Liebl, 1992).

[13]A well known example of a deterministic simulation is Conway's game of 'Life' Gardner (1970), for which a number of implementations can be found on the Internet – e.g. on `www.bitstorm.org/gameoflife/` last accessed 23.03.09. In this simulation simple rules lead to short-lived, constant, or oscillating patterns depending on the start pattern.

2.5 Implementation and verification

Although simulation could be done by manual calculations in principal, the amount of data that must be stored and manipulated, necessary replications, or experiments for different system parameterizations dictate that simulation is performed on a computer (Law and Kelton, 1991). To make a conceptual model run on a computer and thereby applicable for computer experiments it has to be transformed into a computer program. For developing a simulation program one can either use general purpose language like `C++`, `Pascal`, `FORTRAN`, `Java`, etc., for which often simulation specific libraries are provided, or specific simulation languages like `AnyLogic`, `GPSS`, `SIMSCRIPT`, `Simula`, etc. While general purpose languages are more flexible, more programming effort is needed. Simulation specific programming languages on the other hand often adopt a specific world-view or programming perspective (see Section 2.6.2) and provide supportive routines for their specific approach to simulation. While the programming language ideally should be chosen to fit the simulation model, the purpose of the analysis, and provide the necessary performance, the decision which programming language to use for the implementation of the simulation model often is based on the availability of and familiarity with the certain programming languages (Liebl, 1992).

Verification means debugging the computer program and thereby determining that a simulation program performs as intended. Thus, verification checks the translation of the conceptual model into a correctly working simulation program such that the program is free of bugs and consistent with the model (Bratley et al., 1987; Law and Kelton, 1991). Besides sophisticated software engineering – like well structured and organized, modular or object-oriented, and transparent source code with sufficient comments –, which is necessary in any programming project and deals with methods for programming and debugging applications independent of their domain specific verification methods for simulation projects include (Bratley et al., 1987; Law and Kelton, 1991; Kleijnen and van Groenendaal, 1992; Liebl, 1992):

- Tracing: Tracing makes use of the fact that simulations could actually also be done by hand but are done by a computer for convenience. In tracing states and intermediate variables of the simulated system (event lists, state variables, counters, random variables), are printed out and compared with manual calculations or just checked for plausibility to see if the program is operating as intended (tracing facilities such as printout of intermediate results or debuggers are provided by merely all general purpose and simulation languages).
- Structured walk-throughs: A structured walk-through is a discussion of the program code with others, knowledgeable about programming and the domain of the system. Progression to the next part of the program code only takes place when all concerns and questions were addressed.
- Comparison to known results: Comparisons of the results of simplified versions of the simulation for which the correct results are known (e.g. from analytical solutions) or can be calculated easily.
- Sensitivity analysis: Sensitivity analysis involves running the simulation program under a variety of settings of the input parameters and check for sensitivity of the outputs to changed inputs and the reasonability of the changes in the output.

- Check random number generator: i.e. comparing prespecified parameters (e.g. mean and variance) of the input probability distribution with that calculated from the random numbers generated to see if they are correctly generated.

- Visualization and animation: Through visualization of the simulation program behavior one can see whether or not the logic of the conceptual model is correctly implemented (e.g. that a job arrives before it is handled) and if there are bugs (e.g. if jobs miraculously disappear).

- Simulation language: The use of a simulation language or simulation libraries for general purpose languages not only reduces code lines and programming time, but also these languages or libraries provide special features that relieve the programmer of programming these components himself and therefore also avoids possible bugs unless the simulation software contains bugs itself.

2.6 Simulation program

Though discrete-event simulation has many applications for diverse problems typical and basic features can be identified in all of them. The simulation program in any kind of discrete-time simulation not only has to keep track of the states of the system but the essence of a simulation is that the simulation program has to change the system states over time i.e. relates the state of a system (its static structure) to the state changes over time (its dynamic behavior) (Page, 1991; Pidd, 1992). In this section we will therefore first consider approaches how a computer program can handle and represent time in discrete-event simulations (Section 2.6.1) and afterwards the simulation program's routines to relate state changes to time (Section 2.6.2).

2.6.1 Time advance in discrete-event simulation

In discrete-event simulation the computer program must keep track of the current value of simulation time as the simulation proceeds, moreover a mechanism to advance simulation time from one value to the next is necessary. The variable in a simulation model that provides the current value of simulation time is called simulation clock and there is generally no relationship between simulation time and the time needed to run the simulation on the computer (Law and Kelton, 1991). The two principal approaches for the advancement of the simulation clock that are distinguished in literature are the time-slicing approach (or fixed-increment time advance) and the next-event approach (or variable-increment time advance) (Maisel and Gnugnoli, 1972; Law and Kelton, 1991; Pidd, 1992). Note that in both approaches, due to the discrete-event type of the simulation and in contrast to system dynamics simulation, the state changes of the simulation, resulting from discrete events, are only considered at the discrete time points of inspection and updating of the system and that the system state remains 'unchanged'[14] in the meanwhile.

[14]Note that in the time-slice approach the states of the system can actually change within a time slice, but only the changes between the points in simulation time where the system is inspected are perceived.

2.6.1.1 Time-slice approach

Time-slicing is maybe the simplest approach to represent time and the dynamic behavior of a system in a simulation (Pidd, 1992).

> The fixed increment discrete-time system simulates time advance by a method similar to the way motion is represented in a cinema film. A motion picture consists of the reel of film containing a series of still scenes which are projected on a screen at a rate of 24 frames per second. The rapidity of the projection and the correlation between successive frames deceives the eye into interpreting the whole effect as continuous motion. (Evans, 1988, p. 25)

In the time-slicing approach, the system is considered as changing in all of its aspects over time and therefore its status is updated and examined in usually fixed and equal time increments – i.e. in regular intervals – until a prescribed amount of simulation time has elapsed. Thus, for an interval (time slice) of Δt the model is updated at time $t + \Delta t$ for the changes occurring in the interval t to $t + \Delta t$ (Maisel and Gnugnoli, 1972; Pidd, 1992).

In the time-slicing approach for time advance the determination of the length of the time slice – measured in simulation time – is critical. If the time slice is too large then some state changes of the system could be missed or actually non-simultaneous events must be handled simultaneously by employing prioritization rules, otherwise if it is too small then the model could be examined unnecessarily often, which negatively affects computer performance (Bratley et al., 1987; Pidd, 1992). These problems can be circumvented by using the more common next-event time advance approach, however, time-slicing is more appropriate for complex models where many events occur simultaneously (Maisel and Gnugnoli, 1972).

2.6.1.2 Next-event approach

In the next-event approach, in contrast to the time-slicing approach, the system is not updated and examined at predetermined fixed intervals, but the system is considered to proceed from one event to another until a prescribed sequence of events is completed or stopping conditions become true. Thereby the model is only examined and updated when it is known that an event occurs and therefore a state change is due and periods of inactivity are skipped by advancing the simulation clock from event time to event time (Maisel and Gnugnoli, 1972; Law and Kelton, 1991; Pidd, 1992). The slack periods between two events are varying in length in many systems – but need not for the applicability of the next-event time advance approach – and the simulation time is moved from one event time to the closest event time of future events (the next event) at which point the state of the system is updated. Thereby the next-event time approach has two advantages over the time-slicing approach. It avoids unnecessary and wasteful inspection of the state of a system when no changes are possible anyway and it clearly determines when events occur in the simulation – which refers to the problem of simultaneous treatment of actually non-simultaneous events during an interval of the time-slicing approach (Pidd, 1992).

2.6.2 Programming styles

Independent of the approach to advance time the program has to relate state changes of the system to simulation time. This can be done by various approaches each focusing on another building block of the dynamic behavior of a system (see Figure 2.5). The classical modeling styles differ primarily in their view of the dynamics of the system behavior (Page, 1991). Which style is appropriate depends on which of the building blocks of the dynamic behavior is perceived as the most appropriate to study the operations of the system. This interpretation can also be seen as a world-view, as it is not always the problem that determines the choice of a programming approach but also different simulation schools (different styles are prevalent in Northern America and Europe for example) or preferences for particular simulation languages (which in turn often favor particular programming styles), due to availability or prior experience, influence this choice (Page, 1991; Liebl, 1992).

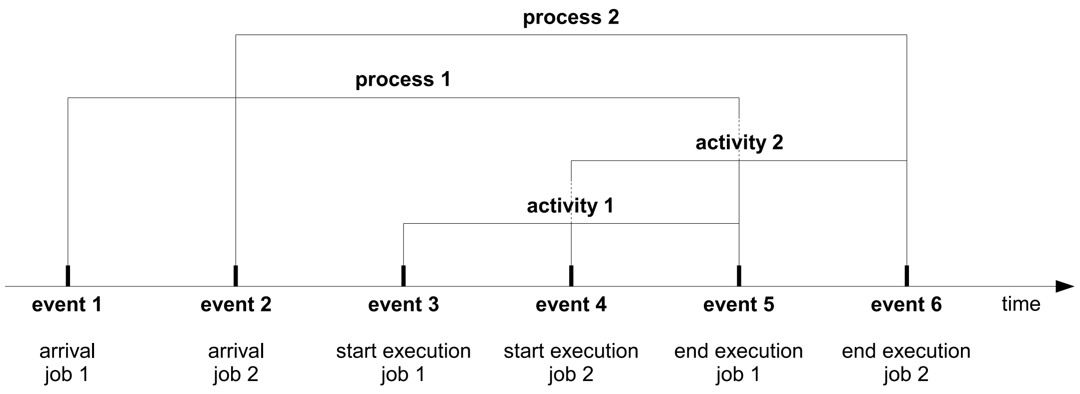


Figure 2.5: Events, activities, and processes (adopted from Page, 1991, p. 22)

As illustrated in Figure 2.5 the dynamic behavior of a system can be divided into (i) events, (ii) activities, and (iii) processes (Page, 1991; Pidd, 1992; Liebl, 1992):

- *Event:* An event is an instant of time at which a significant state change occurs in the system. Such events can be state changes due to temporary entities entering or leaving the system, the beginning or ending of some operations, or state changes of an entity (e.g. from idle to occupied for a machine). Events can be subdivided into environmental or exogenous events that are independent of the model (e.g. a job arrival) and internal or endogenous events that are caused by other events (e.g. an event 'start job execution' causally determines the event 'finished job execution' for some simulation time in the future).

- *Activity:* An activity is a set of operations performed on some temporary entity of the system. These activities cause changes in the state of the system. Thus the operations and procedures on a temporary entity in a period of simulation time initiated by an event and terminated by another event are termed activity (events determine the time when an activity is started and finished).

- *Process:* A process is a group of events and activities for some certain temporary entity in chronological order of their occurrence (i.e. the life-cycle of a temporary entity in the system).

How these basic building blocks of the dynamic behavior of a system are combined and how important they are considered for the system determines which programming approach to use. The three major approaches (Page, 1991; Liebl, 1992; Pidd, 1992) (i) event scheduling, (ii) activity scanning, and (iii) process interaction are discussed in detail subsequently. These approaches embody distinctive programming style, as each requires to divide the operations of the system into different parts that ideally occupy their own program segment: event routines in case of the event scheduling approach, activities in the activity scanning approach, and processes in the process interaction approach, respectively (Page, 1991; Liebl, 1992; Pidd, 1992).

2.6.2.1 Event scheduling approach

In the event scheduling approach the dynamic behavior of the system is implemented by event routines. This programming style for simulation is used for example in the simulation language `SIMSCRIPT` and is more popular in the US than in Europe. The event routines are the dynamic part of the simulation program which change the attributes of the temporary and permanent entities of the system (e.g. jobs and machines). Each event routine is ideally implemented as an separate modular program segment (e.g. a function) that changes the attribute values of entities or schedules other events when called. For example the state of a machine could be set from 'idle' to 'occupied' in the event routine 'start processing' and could schedule the event 'finished processing' for some simulation time in the future. At this simulation time the event 'finished processing' changes the occupation attribute of the machine back to 'idle' (Liebl, 1992; Pidd, 1992).

Though time advance in event scheduling simulation could be done by time-slicing, the approach most often used is the next-event approach, which is more appropriate for this programming style. The next-event approach is implemented by simply using a future event list as simulation controlling device, where all future events are registered with the name of the event and the event time. Starting the simulation clock at time zero, at any point in simulation time the simulation is inspected, this future event list is ordered increasingly in event time by the simulation program's main routine. The event routine of the event scheduled for this time is called, which changes the system state – in changing attributes of the entities – and/or schedules new future events. For the case of multiple events scheduled for the same simulation time prioritization rules have to be provided to determine the processing order of event routine. Processed events are deleted from the future event list, the list is reordered, and the system time is set to the event time of the earliest next event (Liebl, 1992; Pidd, 1992).

Using the event scheduling approach not only allows a clear distinction between the static structure (temporary and permanent entities and their attributes) and dynamic behavior (events changing system states – i.e. entity attributes – and causing other future events) (Page, 1991), but also results in faster simulation programs opposed to the use of the activity-scanning or the process-interaction approach for systems with low complexity of the system components' interactions, like reservation and release of work stations (Liebl, 1992).

2.6.2.2 Activity scanning approach

The activity scanning approach structures the dynamic behavior of a system and therefore the simulation program into activities that occur in the system. This approach was popular in the UK and is implemented in the simulation language `CSL`. Due to the subsequently mentioned deficiencies of this approach it is of minor practical relevance only. Same as for event routines in the event scheduling approach, the activities – the central concept for program structuring in this approach – of the activity scanning approach should be separated in modular program segments. These program segments, one for each activity, have to determine all the necessary conditions that have to be fulfilled for performing the particular activity. If the preconditions for an activity are met then the activity is performed and the state of the system is changed accordingly. For example to process a product on a machine it is necessary that there is a product in the queue in front of the specific machine and that the machine is idle at the simulation time when the conditions for the activity 'process product' are controlled. The program segments representing activities are best divided into a test head, which tests if the conditions for the activity are met, and an operations part which executes the operations that are part of the activity only if the tests of the test head are passed (Liebl, 1992; Pidd, 1992).

At any simulation time (may it be advanced by the time-slicing or next-event approach) the simulation program's main routine has to test all activities and check whether the conditions for these activities are met, and in case these tests are passed execute the activity. This highlights the importance of the correct ordering of the activities in the simulation program, which is more problematic than that of the event routines in the event-scheduling approach, as it is more difficult to correctly implement the logic of the simulated system. The necessity to scan through all possible activities for fulfillment of their conditions at each inspection of the system, which consumes more computer performance than the event-scheduling approach, and the more difficult structuring of the simulation program are the main disadvantages of the activity scanning approach compared to other approaches (Liebl, 1992).

2.6.2.3 Process interaction approach

From the discussion above it can be derived that the event-scheduling and activity scanning approach segment the whole life-cycle of an entity into its fundamental parts and implements these parts as separate program modules – events and event routines or activities and their execution conditions. The process interaction approach on the other hand deviates from this fragmentation and considers the whole life-cycle of entities instead. In the process interaction approach all the events and activities a (normally temporary) entity traverses during its life-cycle in the simulated system are combined into a process. Both, the active phases – during which state changes occur – and passive phases – during which the process waits for its reactivation – of entities are represented in these processes, which again should form separate segments of the simulation program (Page, 1991; Pidd, 1992).

The essence of the process interaction approach is closely related to object-oriented programming and actually the simulation language `Simula`, that follows the process interaction approach, was the first object-oriented programming language. Each class of entities (e.g. different kinds of jobs) has its own process and each entity created as a member of a particular class inherits this process. The 'life' of a focal entity can be traced by checking its progress through and its current position in its process. As entities entering the system during the simulation are initiated as members of their class and inherit their class' process – as a template for their future 'life' in the system – as many processes as entities run concurrently in the system and the number of processes fluctuates with the number of entities entering and leaving the system (Page, 1991; Pidd, 1992).

If entities interact so do their processes, activities of different processes for example can overlap and run parallel, if there are no conflicts, or only sequentially if they block each other. In the later case passive waiting times have to be modeled explicitly and active operation times implicitly through the change from one event and activity to another (Page, 1991). The main routine in a process interaction simulation program – which itself can be seen as a process – has to keep track of all running processes at any point in time, which is difficult as this number might be unknown in advance and fluctuating with the entities in the system. Each process has to be moved forward as far as possible by the main routine, which also needs functionalities to interrupt a process – unconditionally when an delay is determined in advance or conditionally when the movement of an entity through its process is halted until specific conditions in the simulation are satisfied – and to restart it again later. These additional requirements make the process interaction approach more complex and difficult to implement in a simulation program compared to the other programming styles discussed above (Pidd, 1992).

For the purpose of the process interaction approach the main routine has to maintain a record of each process which contains the reactivation time and the reactivation point of the process i.e. when and where the process of an entity has to be continued. If one considers the reactivation of a process as an event then the executive part of the simulation program can be implemented much like the future event list of the event-scheduling approach, with the distinction that it contains the next reactivation times of the processes ordered increasing in time. In contrast to event routines however the processes are not processed from their beginning and do not terminate but can be interrupted before they completed and re-activated later to further proceed from this point. For this reactivation it is necessary to save the process state – all attribute values as well as information on reactivation points of the process – before the deactivation (Page, 1991; Liebl, 1992; Pidd, 1992).

2.7 Experimentation

All these previously mentioned phases and intermediate results – the creation and validation of a conceptual model as well as the implementation of this model in a simulation program and its verification – are 'only' means to the end of being able to perform computer experiments with the simulation program and derive conclusions about the original system from the results of these experiments. For this experimentation, first the experimental design has to be determined, i.e. which factor level constellations are of interest and how the computer experiments can be organized efficiently, second the output of interest and subject to the later analyses of the focal simulation has to be determined, and of course it has to be investigated how sensitive the outputs and conclusions derived from the simulation program are to changes in the input variables.

2.7.1 Simulation output

In contrast to validation, which is concerned with the appropriateness of the conceptual model for questions of the intended study, output analysis of the results of computer experiments is concerned with the outputs of this model and its performance. It therefore is mainly a statistical task involving problems like the determination of the run length of a simulation or the necessary number of replications (Law and Kelton, 1991). Which simulation outputs and performance measures are relevant in the analysis of a simulation and for drawing appropriate conclusions about the real-world system depends on the nature of the system output. Concerning their output and behavior over time systems and their conceptual models for simulation can be divided into (i) steady-state or transient systems (concerning the behavior of output over time) and (ii) terminating or non-terminating systems (concerning the time horizon of the system) (Liebl, 1992; Pidd, 1992). These two factors with each two levels combine to a total of only three reasonable types of systems (Liebl, 1992):[15]

- *non-terminating steady-state systems:* These systems reach a long-term equilibrium state with no trend components so that the system outputs are time-invariate. A system is considered to be in a steady-state if its current behavior is independent of the starting conditions and if the probability of being in one of its states is governed by a fixed probability function, which means that the system may change its state but the probability of doing so can be determined. Therefore a steady-state system embodies a steady stochastic process. Systems that can reach a steady-state clearly should be evaluated when in this state – i.e. when the effects of the initial conditions are no longer noticeable (Liebl, 1992; Pidd, 1992).
- *non-terminating transient systems:* Most systems however do not reach a steady-state but are non-terminating as well, for example due to varying inputs over time ('rush hour' or seasonal trends) as for example an airport which is not closed at night but has lower airplane arrival rates during the night and higher ones during some rush hour (Liebl, 1992; Pidd, 1992).
- *terminating transient systems:* Whereas non-terminating systems are considered as a continuum (like airports without night closure or phone networks that are started only once and then are considered to run infinitely) terminating transient systems have some natural start and end, so that their time horizon is a finite one as they are self terminating by some particular events (Pidd, 1992). Here an example would be a post office opening at 9:00 am and closing at 5:00 pm.

The output or response of a simulation run depends on the input – even in steady-state systems the starting conditions as input to the system will influence if and when the system reaches a steady-state. In simulations where these inputs are random variables a single run of the simulation only yields information about the simulation output for this specific values of the input variable (or combination of values in case of more than one input variable). Therefore we are usually not interested in one input-specific response but rather on the distribution of the

[15]The fourth combination would be a system that reaches a steady-state but is a terminating system, which is a contradiction, as systems with logical start and end do not satisfy the conditions of time invariance of output measures necessary for steady state (Liebl, 1992).

response or summary measures of this output distribution for different input variables like mean, standard deviation, variance, quantiles, and minimum or maximum (Kelton, 1999).

The output measures of interest for non-terminating systems are usually some average measures as it makes no sense to consider aggregate measures due to the infinite time horizon of the system. For non-terminating steady-state systems independence of average measures from initial input variables can be achieved by deleting observations from the so-called run-in phase, where the output measures of the system are still influenced by the initial starting conditions of the simulation and are not in a steady-state yet.[16] However, still observations will not be independent – a major requirement for the applicability of many statistical techniques – but are likely to be autocorrelated (Liebl, 1992; Pidd, 1992). In a simple queuing system for example the waiting time of a customer will depend on the number of people that are already waiting when the customer arrives and the waiting time of the customer will therefore depend on the waiting time of preceding customers (Pidd, 1992). To achieve independence of observations in non-terminating steady-state systems (after removing observations of the run-in phase) the whole output time series can be divided into batches for which average measures are calculated, which are likely to be independent of each other when the batch size is long enough. Batches can also be used for non-terminating transient systems if their output shows some cyclical or periodical behavior where the cycles or periods form the batches for which average measures are calculated. If such cyclical behavior cannot be observed then the only way to account for input dependence of the outcomes is – same as for terminating systems – to do several replications of the simulation run and calculate average or aggregated measures over these replications. In terminating systems one run from the start condition of the simulation until the critical event that terminates the simulation yields a single observation of the response of interest. This observation clearly incorporates start-up and end effects due to the specific input variables. To yield several observations in terminating systems, for which summary measures of the response variables can be calculated – which then are independent of the simulation input –, several replications of the simulation over its time horizon are necessary (Kleijnen, 1987; Pidd, 1992).

An additional possibility to increase the accuracy of outcome measurement of a system is the artificial reduction of the output variance by means of so called variance reduction techniques (Law and Kelton, 1991; Liebl, 1992). Variance reduction is a procedure to increase the precision of the estimates that can be obtained from a number of replications of a simulation. Every output variable of a simulation is in case of the usage of random input variables itself also a random variable with a particular variance that limits the precision of the simulation results. In order to render a simulation statistically more efficient, i.e., to obtain a greater precision for the output variables of interest variance reduction techniques like common random numbers, antithetic variates, control variates, indirect estimation, conditioning, importance sampling, and stratified sampling can be used (Law and Kelton, 1991; Liebl, 1992). We will focus on common random numbers for variance reduction which is – maybe due to its simplicity – the most popular and powerful variance reduction technique. Furthermore the common random numbers technique is the only one of the above mentioned directly applicable for comparisons of two or more alternative

[16]No matter if the simulation is started from an 'empty-and-idle' state or with 'typical' starting conditions – which might be difficult to know exactly and must be equal for all runs or system versions – there will be such a run-in phase, which has to be deleted from the observations, however, it might be shorter in the later approach of starting the simulation (Pidd, 1992).

system configurations, which is the objective of this dissertation, while other variance reduction techniques are applicable only for the investigation of one single configuration (Law and Kelton, 1991).[17]

The basic idea of the use of common random numbers is that alternative configurations should be compared under similar experimental conditions so that observed differences are due to differences in the system configuration rather than the result of fluctuations of the experimental conditions i.e. the different realizations of the random number generator used in the simulation program for interarrival times, demand sizes, etc. The common random number technique for variance reduction requires the synchronization of the random numbers streams i.e. the use of the same realizations of random input variables for the same purposes in system configurations to be compared. Thereby the simulation program experiences the same environmental input for different system configurations and differences can only be due to these differences in system configuration (Law and Kelton, 1991).[18]

2.7.2 Experimental design

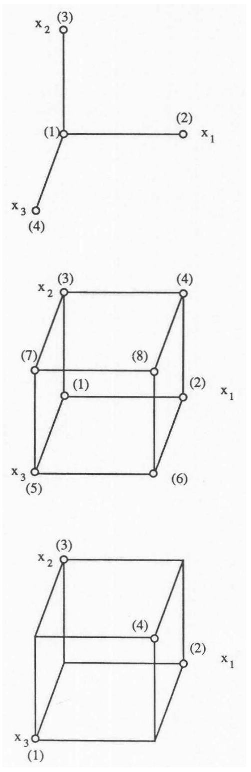
The insights and approaches from experimental design in statistics are applicable to computer experiments in simulations as well. A sophisticated experimental design is necessary in simulation studies whenever alternative system configuration are not externally provided (e.g. different policies or system configurations considered by the decision maker) and only need to be compared, but when the objectives are broader and more ambitious ones, like to find out which of the possibly many parameters and structural assumptions have the greatest effect on the outputs of a system or which of the system configurations leads to the best performance of the system (Law and Kelton, 1991). In the experimental design terminology the system and input parameters as well as structural configurations (under control of the decision maker) that compose the conceptual model are called factors and the output measures are called responses. Factors can be either quantitative, assuming numerical values, or qualitative typically used to represent structural configurations that cannot be quantified naturally. Furthermore, factors can be controllable or uncontrollable depending on whether or not they represent action options to managers of the corresponding real-world system, where it is straightforward to focus on controllable factors in the computer experiments, since they are most relevant to decisions about the implementation of the real-world system (Law and Kelton, 1991; Kleijnen and van Groenendaal, 1992).

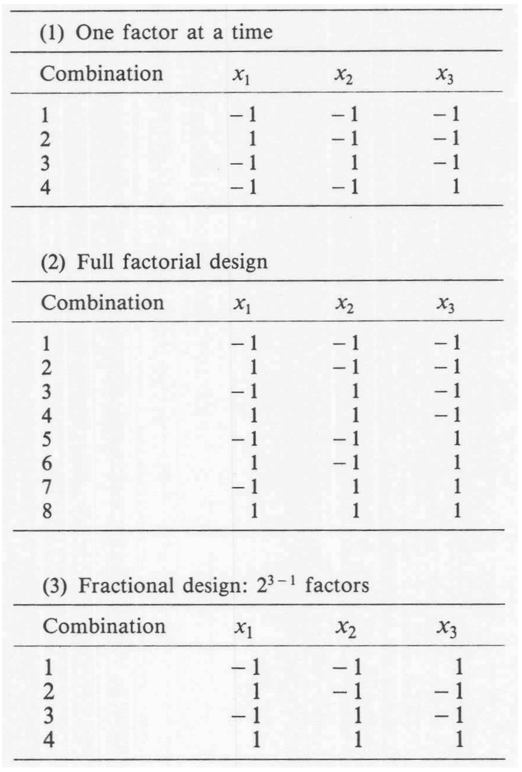
When the total number of possible factor combinations results in a large amount of necessary simulation runs, experimental design provides approaches to decide before the simulation runs, which particular configurations to simulate to obtain the desired information with the least amount of simulating. Therefore, carefully designed experiments are much more efficient than a hit-or-miss sequence of runs in which a number of alternative configurations are explored un-

[17]We refer to the literature (e.g. Law and Kelton, 1991, chapter 11) for a detailed discussion of other variance reduction techniques.

[18]Note that the variance reduction technique of common random numbers is closely related to the validation technique of trace-driven validation (or correlated inspection) we discussed in Section 2.3.3, with the difference that in the later case the same historical input values are used to run the simulation program for operational validation of correspondence of system and simulation output, while the former uses the same random input values to run the simulation program for different system configurations for comparison of output differences (Law and Kelton, 1991). The statistical techniques presented in Section 2.3.3 are applicable for both purposes.

systematically to see the consequences (Law and Kelton, 1991). There exist several classical experimental designs like (i) one-factor-at-a-time design (or *ceteris paribus* design), (2) full factorial design, or (iii) incomplete factorial design (2^{k-p}-design or fractional factorial design) as illustrated for three (binary) factors $\{x_1, x_2, x_3\}$, each with only two levels $\{1, -1\}$ in Figure 2.7.2 (Law and Kelton, 1991; Kleijnen and van Groenendaal, 1992).





(1) One factor at a time

Combination	x_1	x_2	x_3
1	−1	−1	−1
2	1	−1	−1
3	−1	1	−1
4	−1	−1	1

(2) Full factorial design

Combination	x_1	x_2	x_3
1	−1	−1	−1
2	1	−1	−1
3	−1	1	−1
4	1	1	−1
5	−1	−1	1
6	1	−1	1
7	−1	1	1
8	1	1	1

(3) Fractional design: 2^{3-1} factors

Combination	x_1	x_2	x_3
1	−1	−1	1
2	1	−1	−1
3	−1	1	−1
4	1	1	1

Figure 2.6: Classical experimental designs (Kleijnen and van Groenendaal, 1992, p. 168-169)

According to Kleijnen and van Groenendaal (1992) the one-factor-at-a-time design assumes no interactions between the factors. In contrast a full factorial design consists of all possible factor level combinations as treatments. This design results in 2^k combinations while the one-factor-at-a-time design only requires $k + 1$ combinations.[19] The full factorial design not only is more effective if interactions between factors exist – as it enables the analysis of interactions between factors – it also is more efficient than the one-factor-at-a-time design (provided $k > 1$) concerning response variance. Concerning the relation of response variance and the number of observations for a full factorial and for a fractional design, these two experimental designs are equally efficient (Kleijnen and van Groenendaal, 1992). While a full factorial design allows the investigation of interactions between factors and factor levels this is not possible with an one-factor-at-a-time design and only possible with a fractional design for certain interactions. However, in case of no interactions the one-factor-at-a-time design can give the same information as the full factorial and

[19]These numbers hold true only for factors with two levels, where k denotes the number of factors. Generally, if the number of levels in factor i ($i \in 1, \ldots, k$) is l_i then the number of necessary treatments to be considered for a full factorial design is $\prod_{i=1}^{k} l_i$.

the fractional design. In practice the fractional design is often preferred due to the considerably smaller number of considered factor combinations and the therefore lower computer time and performance necessary for the simulation runs (Kleijnen and van Groenendaal, 1992).

In addition to those classical experimental designs the simulation literature often also considers some other relevant techniques from statistical experimental design theory for specific simulation purposes (Law and Kelton, 1991; Kleijnen and van Groenendaal, 1992). For example if the number of factors is very large, so that the number of treatments (factor-level combinations) to be considered for a full factorial design or a sophisticated fractional design grows too large to run the simulation and do the analysis in a reasonable period of time, factor-screening techniques could be applied to identify those factors which appear relevant and irrelevant factors can be fixed – in replacing variables by a constant – at some reasonable value omitting them from further consideration. Other techniques applicable if controllable factors are quantitative and the aim of the simulation is to find some system configuration that achieves optimal output (either maximizing or minimizing some output values) are response surface or meta-modeling techniques, which in such settings are preferable over classical experimental designs as they identify optimal system configurations faster and with fewer simulation runs than undirected approaches.[20]

2.7.3 Sensitivity analysis

If one perceives a system as a function f transforming some uncontrollable input variables X under some controllable system parametrization Y into output E (2.5) (Banks and Carson, 1984; Kelton, 1999) then one studies a system's performance in its environment (contingency). Sensitivity analysis is defined as the investigation of the reaction of model outputs to systematic changes in the uncontrollable environmental inputs and the controllable system parametrization (Kleijnen, 1995) i.e. the sensitivity of E to changes in both X and Y. Accordingly, many scholars argue that sensitivity analysis is a part of the validation of the conceptual model. All inputs to the model controllable and uncontrollable should be altered systematically to be able to observe for which the output changes, and therefore which are sensitive to the output. These inputs then have to be modeled with greatest accuracy and appropriate data for them has to be acquired (Law and Kelton, 1991; Liebl, 1992; Sargent, 2005).

$$E = f(X, Y) \tag{2.5}$$

However, there are two problems with this kind of conceptualization of sensitivity analysis. First, a simulation program already has to be validated and verified to achieve reliable output measures for systematically changed input measures (Liebl, 1992), so the value of sensitivity analysis for validation of a conceptual model is questionable unless it is used for validation of intermediate conceptual models only. Second, in simulation studies with the purpose to evaluate different policies or system configurations (system design) sensitivity analysis overlaps with output analysis if both controllable and uncontrollable inputs are considered in the simulation study. As

[20]We refer to the literature (e.g. Law and Kelton, 1991; Kleijnen and van Groenendaal, 1992) for the detailed discussion of these techniques, which we omit here as they are not applicable to the type of our model and the purpose of our study.

mentioned, experimental design and output analysis is mainly concerned with the effects of controllable system inputs – i.e. the system parametrization or configuration under control of the decision maker. In simulation studies with the purpose of system design sensitivity analysis therefore can be applied to investigate the robustness of the conclusions derived from simulation in investigating the sensitivity of the output on the second major driving force, namely the uncontrollable environmental inputs. Sensitivity analysis can be performed by graphical plotting and visual inspection i.e. plotting the simulation outputs and the values of the input variable on different axes; one plot per input-output combination or – also accounting for interactions – by means of statistical methods much like those applied for testing the sensitivity of the output to controllable system parameters (Kleijnen, 1995).

Chapter 3

Automated Negotiation

For the study of automated negotiation, which though constituting a very specific form still belongs to the realm of negotiation, the framework of negotiation research – with necessary adaptations and inevitable restrictions – is applicable. Negotiation research in general – though some approaches focus only on components and their relations, neglecting others as discussed in Chapter 1 – studies how the preconditions of the negotiation and the context in which it takes place influence aspects of the negotiation process, which in turn shapes the outcome of the negotiation (see Figure 3.1).

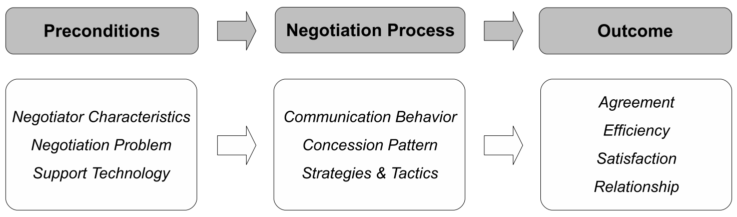


Figure 3.1: Negotiation research framework – adapted from Köszegi (2008, p.11)

The context of negotiations can be categorized along multiple dimensions, like the number of parties and issues, whether the party is a single individual or a group, whether or not an agreement is unconditionally necessary, whether parties can exert power or make threats, whether or not linkages to other concurrent or future negotiations exists, etc. (Raiffa, 1982). Furthermore the context of the negotiation is influenced by the characteristics of the negotiators participating in the negotiation and their attitudes towards negotiating, like their cooperativeness or toughness, their experience in negotiating, impatience, creativity, power, etc. A further ingredient to the negotiation context is the use of support tools (negotiation support systems and the features they provide to the negotiation, like communication support, decision support, etc.) and the assistance of third parties, through for example mediation or arbitration (Wall and Blum, 1991; Carnevale and Pruitt, 1992; Lewicki et al., 1992). The most important constituent of the negotiation context, however, is the subject of the negotiation – the negotiation object –, the preferences of the parties over this negotiation object (Greenhalgh et al., 1985), and the negotiation problem resulting from the joint evaluation of this negotiation object according to these preferences (Mumpower, 1991; Carnevale and Pruitt, 1992; Clyman, 1995). The negotiation problem (Mumpower, 1991) – also called utility scatter plot (Clyman, 1995) or joint utility space (Carnevale and Pruitt, 1992) – is a major concept for the generalization and analysis of negotiations. It is the evaluation of all possible solutions of in the negotiation object by all negotiators. For the case of bilateral negotiations, between two parties, it can be represented by plotting the utility values for all possible settlements in a two dimensional graph, where each axis represents the utilities of solutions for one of the parties.

This context influences the way negotiations proceed over time. Due to cognitive complexity and uncertainty negotiators do not settle for their first offer (Mumpower, 1991), but engage in what Raiffa (1982) calls a negotiation dance, i.e. a sequence of offers and counteroffers. Together with these offers, that constitute tentative proposals for settling the conflict at hand, negotiators also exchange other messages – like promises, threats, or messages providing or requesting information – in an attempt to influence the final outcome (Tutzauer, 1992). Negotiation processes therefore can be studied by means of quantitative analysis – focusing on the offers exchanged –, by means of content analysis – focusing on the content of the messages exchanged and the strategies and tactics employed by the negotiators –, and by combinations of the two former approaches. However, negotiation processes only can be investigated against the background of the context in which they emerge, i.e. the negotiation problem (Zartmann, 2002). Mumpower (1991) for example showed that the same prototypical strategies can lead to different outcomes, which depend on the underlying negotiation problem, and that for some negotiation problems it is more difficult to reach (Pareto-optimal) agreements than for others, independent of the strategy employed (Mumpower, 1991; Mumpower and Rohrbaugh, 1996). So the negotiation outcome is determined not exclusively by the structure of the negotiation problem – as proposed by axiomatic game theoretic approaches – or gross behavioral indicators – as assumed in social psychological research on negotiation – but also by the negotiation process emerging between the parties (Tutzauer, 1986).

In exchanging offers the negotiation parties try to mutually arrive at an agreement for the negotiation object, for which they have conflicting interests and preferences. The outcome of the negotiation is an agreement on one of the possible solutions if they succeed, or non-agreement (or break off of the negotiation) if they fail to do so. Whether or not an agreement is reached is a major outcome measure in negotiations, however there exist several refinements of this rather broad outcome measure that closer investigate the quality of a reached agreement (Tripp and Sondak, 1992). Additional quantitative aspects of the outcome are for example how much time and resources were spent on achieving the outcome (efficiency of the process) and if the solution is good compared to alternative solutions at the dyad level (sum of individual utilities or Pareto-optimality of the outcome) as well as at the individual level. If better solutions exist, then it might be interesting how much better they are, which can be determined by means of distance measures (to the Pareto frontier or to the normative solutions provided by axiomatic bargaining theory). Furthermore one can discuss the fairness of the outcome by means of utility differences of the agreement between parties (the contract balance). However, negotiation processes and outcomes can also be evaluated along more qualitative and subjective measures asking the negotiators in post-negotiation questionnaires if they perceive the process and outcomes to be favorable for the relationship between the negotiators or if they improved trust between the parties. Furthermore questionnaires can be used to evaluate the parties level of satisfaction with both the result of the negotiation (outcome) and the way they achieved it (process).

Automated negotiation is a mechanism by which autonomous software agents, following their users preferences and an interaction protocol, conduct negotiation in automating the negotiation process. Due to this automation of the negotiation process the use of automated negotiation is restricted to a fragment of negotiations only, at least at the current state of the field, namely negotiations over fixed negotiation problems. In a fixed negotiation problem neither the set of possible solutions, nor the preferences of the negotiators change during the negotiation. The reasons for this restriction are twofold: First if automated negotiation is to be used, then the negotiation problem has to be well structured so that it can be communicated to the software agents (Weigand et al., 2003; Chen et al., 2005b). This implies that the issues under discussion and the possible settlement options for these issues are known at the beginning of the negotiation. Second, due to the quick proceeding of automated negotiations it makes no sense to consider the possibility of changing preferences. Therefore automated negotiation – at least at the current state of the field – is limited to fixed negotiation problems, where the negotiation object and the users' preferences over the negotiation object are determined before the negotiation and remain constant throughout its course.[1] Further restrictions of the use of automated negotiations concern the domain of the problems for which automated negotiation are appropriate. Due to the nature of software agents, negotiation problems involving non-economic evaluation criteria – such as fairness, justice, or satisfaction – or of social complexity – like problems concerning the relationship or trust between the negotiators or other social criteria – can only be handled by automated negotiation to a limited extent.

For this specific subset of negotiations, in which automated negotiation is applicable, and when focusing on bilateral negotiations – as the most widely discussed form of negotiation, where neither more than two nor third parties like arbitrators, mediators, etc. are not involved – the negotiation context reduces to the negotiation problem, as the negotiation is conducted by software agents, for which personal characteristics of the 'negotiator' are not applicable.

[1]However, it is also possible to determine the negotiation problem in an automated 'meta-negotiation' before the actual negotiation, where the content of this meta-negotiation is the negotiation object of the subsequent actual negotiation. Mechanisms that alter the negotiation problem during the negotiation are currently under development (Sycara, 1991; Fatima et al., 2004), but were not used in simulation studies yet, and actually are not elaborated sophisticatedly enough to use them in simulations at the moment. At the current state of the art of automated negotiation the issues and settlement options, as well as the preferences over this negotiation object have to be constant during the negotiation and communicated by the user to its software agent. An alternative to meta-negotiations for determining the negotiation object would be an automated determination by the system used for automated negotiation. For this purpose the software agent has to elicit the preferences of the user (about all issues and options regarded to be important by the user) as well as their outside option – their 'best alternative to a negotiated agreement' (BATNA) – before the negotiation. This preference information then has to be communicated by the software agents to a main routine of the negotiation system that is in charge of determining the negotiation object. With preference information for all negotiation parties the system constructs a negotiation object so that only the conflicting issues and only feasible options remain by the following procedure: (i) issues of importance to only one party are settled for the best option for this party in advance, (ii) issues where the parties indicate the same best option are settled for this best option, (iii) the option spaces for the remaining issues, for which conflicting interests exist are set to the upper and lower bound indicated by the parties reservation levels so that only feasible settlements remain in the negotiation object, and finally (iv) in a last step all agreements that do not satisfy the participation constraints of either party (i.e. afford lower utility than the BATNAs of the parties) are deleted from the set of possible agreements. As the preferences used for constructing the negotiation object have to be indicated to the software agents in a first step, which follow them during the automated negotiation, an user only would penalize himself by indicating wrong preferences. E.g. in case of misrepresented reservation levels or BATNAs no zone of possible settlements could be the outcome of the automated negotiation object determination by the system and therefore no agreement can be reached even if according to the true preferences possible settlements would be found, which then would be better than the actual BATNA, in case of misrepresented issue weights or misrepresented partial utility values for options the agreement of the automated negotiation might change to an inferior one as the agent follows these wrong preferences.

Moreover, for the same reason only quantitative outcome measures should be considered for the evaluation of negotiation outcomes. In opposite to the negotiation problem and outcome, which are besides several restrictions actually the same as in traditional negotiations, the negotiation process is quite different as it does not emerge from the interactions of human negotiators, but is automated and emerges from the interactions of software agents that are governed by an interaction protocol. Therefore one major component in automated negotiation studies is the automated negotiation system, which consist of software agents that perform negotiation tasks and make the necessary decisions autonomously, according to the preferences of their users, and an interaction protocol, that governs and coordinates these interaction between the software agents. As Rosenschein and Zlotkin (1994) put it:

> There are two distinct aspects to an interaction. The first aspect is the 'rules of the game' that constrain the public behavior of the participants. For example, if two computers are playing chess, the rules of the game determine how pieces on the board may be moved, the alternating turns of the two players, and when on player has won. The second aspect of an interaction is the private strategy adopted by each participant. The strategy determines which among the possible alternative public actions the agent will choose at each step. In the chess game, the strategy of a player determines how he chooses among the legal moves available to him. (Rosenschein and Zlotkin, 1994, p. xx)

Besides the software agents and the interaction protocol, that constitute the automated negotiation system, the third major component in automated negotiation studies is the negotiation problem (Jennings et al., 2001), as input to this system – in form of the users' preferences over the negotiation object, which have to be communicated to their software agents – as depicted in Figure 3.2.

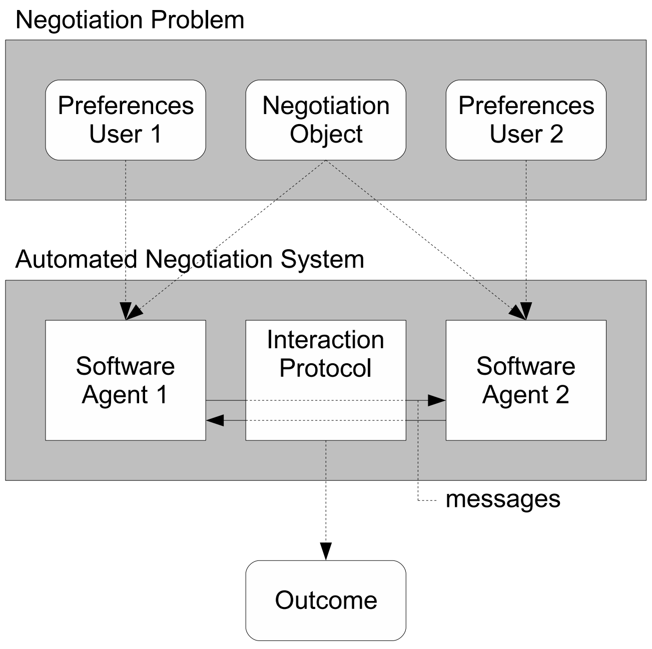


Figure 3.2: Components of automated negotiation

3.1 Current achievements in simulation of automated negotiation

Scholars reviewing the literature on simulation of automated negotiation – like for instance Kraus (1997) or Jennings et al. (2001) – often rely on convenience sampling and tend to focus on their own work and/or work of colleagues in their research area.[2] However, as we are interested in the state of the art of automated negotiation from a more objective perspective and to avoid selection bias, we base the sampling of studies included in this review on a rigorous query of databases and search engines for scientific publications. The search keywords used in the keyword search were 'automated negotiation' and 'negotiation simulation', as well as 'automated bargaining', and 'bargaining simulation' – as in literature 'negotiation' and 'bargaining' often are used synonymously. These keywords were searched in the titles, and where possible also in the abstracts, of publications.[3] Table 3.1 shows the number of results for these keywords in popular databases and search engines for scientific publications.

DB	URL	hits
JSTOR	`http://www.jstor.org`	6
EconLit	`http://www.econlit.org`	6
ABI Informs/ProQuest	`http://proquest.umi.com/login`	55
SSCI	`http://portal.isiknowledge.com/`	126
scholar.google	`http://scholar.google.com`	104

Table 3.1: Results of the keyword search in scientific databases and search engines

We then merged the search results to one single result list by deleting duplicates across and within the result lists.[4] This final result list contains 221 publications, 140 (63%) publications in scientific journals and 81 (37%) contributions to conference proceedings. According to their abstracts, and where necessary their full content, these publications were assigned to one of nine categories (see Table 3.2).

category	journal	proceeding	total
simulation	24	13	37
software framework	8	24	32
theory	13	18	31
analytic model	15	4	19
review	5	2	7
agent communication	3	4	7
experiments	55	2	57
auction	6	3	9
other	11	11	22
all	140	81	221

Table 3.2: Research categories in automated negotiation

As one can see from the lower part of Table 3.2 a large share of the results of our search (88 publications or 40%) are not directly related to automated negotiation. Most of these unrelated

[2]As mentioned in the introduction we focus on simulation studies in this review as analytical models apply restrictive assumptions and operative systems for automated negotiation are not available yet – besides few simplistic experimental systems.

[3]Searching in abstracts is possible in JSTOR, EconLit, and ABI Informs/ProQuest.

[4]For scholar.google in many cases there existed multiple results for the same publication.

publications (57) deal with results of field or laboratory studies on negotiation or present negotiation cases to conduct such studies. A smaller share of unrelated literature (9) deals with different forms of auctions such as one-to-many or many-to-many, single- or multi-attribute auctions – consult Kersten et al. (2000) for a discussion of the differences and similarities between auctions and negotiations. Other unrelated literature (22) covers topics like negotiation support systems, decision support, proposal generation for e-marketplaces, e-mediation, coalition formation, etc.

The remaining 133 publications (60%) actually deal with various topics in automated negotiation. 32 publications propose software frameworks that model software agents and their interactions for software engineering purposes, 31 publications provide theoretical contributions and concepts for the field of automated negotiations, in 19 papers analytical models dealing with automated negotiation are presented, seven studies provide reviews on automated negotiation, and another seven contributions deal with agent communication and argumentation.

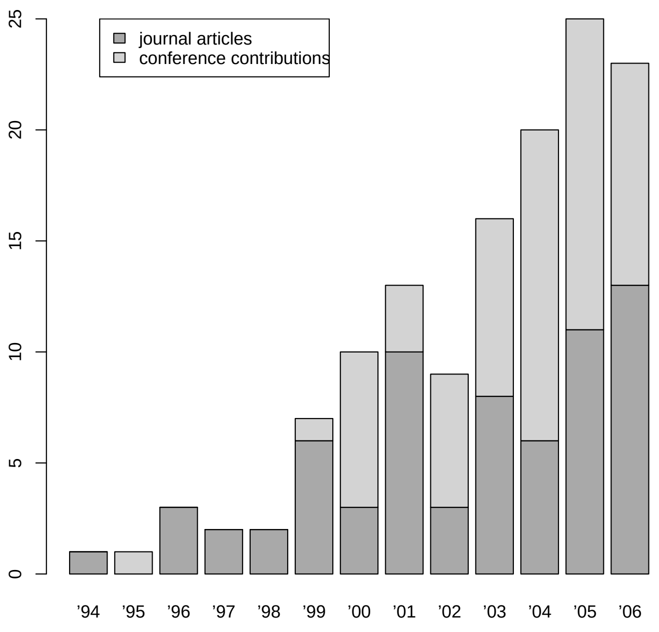


Figure 3.3: Development of publication activity in automated negotiation

The largest share of publications (37), of which 24 are papers published in scientific journals and the remaining 13 are conference contributions, however, deals with studies of interest for this review, i.e. simulation studies of automated negotiation.[5] From the temporal development of the publication activities in automated negotiation (illustrated in Figure 3.3) one can derive the novelty as well as the increasing importance of automated negotiation as a research area.[6]

[5]These 37 publications build the basis for the detailed review in the subsequent section and are marked with an asterisk in the bibliography.

[6]In Figure 3.3 we only illustrate the development until 2006 as the literature search was performed on July 9th, 2007 and we therefore have no complete record of publications in 2007. Furthermore note that given the fast development of research on automated negotiation this review – conducted at the very beginning of this

Severals studies propose criteria for the classification of automated negotiations. Rosenschein and Zlotkin (1994) for example propose criteria desirable for negotiation protocols from a mechanism design perspective, which are computational and communication efficiency, individual rationality of agents, a distribution of computation, and Pareto-optimality of the results. For studying the distinction of cooperation and competition in distributed artificial intelligence and multi agent systems Kraus (1997) uses the level of cooperation among agents, the applied protocols, the number of agents and their types (automated only, human only, or both), as well as communication and computation costs as review criteria. Lomuscio et al. (2003) argue that for designing a broad variety of automated negotiations, including auctions and game theoretic models, parameters for the whole 'negotiation space' have to be considered. The parameters they discuss are the cardinality of the negotiation (i.e. number of buyers and sellers), the agents' characteristics including agents' strategies, characteristics of the environment and goods traded, as well as event, information, and allocation parameters. In this review we focus on a more parsimonious but in return widely applicable set of review criteria adopted from Jennings et al. (2001). The three criteria proposed by them are the main components, discussed above, which each automated negotiation must exhibit: (i) the negotiation problem, (ii) decision making algorithms for the software agents, and (iii) an interaction protocol. These three components are also covered in one or the other way in all of the above-mentioned classifications. In the following sections we first define each of the three components of automated negotiations, as well as their attributes and possible values for these attributes (see Table 3.3), which then are used as criteria for the review of the above identified literature on simulation of automated negotiation.

Attribute	values
Negotiation problem	
Number of issues	one . . . many
Set of possible agreements	finite . . . infinite
Negotiators' preferences	assumed . . . elicited; complexity
Negotiation problem structure	distributive . . . integrative
Interaction protocol	
Chronology of communication	simultaneous . . . sequential
Configuration of offers	single issue offer / agenda . . . package offer
Progression of the negotiation	concession-based . . . improvement-based
Types of actions	offer, reject, exit
Abandonment of negotiations	exogenous . . . endogenous termination
Decision making algorithms of software agents	
Opening offer	extreme . . . moderate; as other offers
Offer generation mechanism	EA-, time-, learning-based, imitation, others
Termination criteria	acceptance, termination

Table 3.3: Review attributes and possible values

3.1.1 Negotiation problem

The negotiation problem is the matter the negotiation is all about, i.e. the object over which participants have different interests and for which they seek to find an agreement through the

dissertation project – could be criticized as already out-dated. However, recently published simulation studies (like e.g. Lai and Sycara, 2009; Lee and Chang, 2009; Ren et al., 2009) in scientific journals relevant for the field like *Group Decision and Negotiation* or *Decision Support Systems* in the few first months of 2009, on the one hand indicated that the field remains highly important, and on the other hand that the major shortcomings detected in the subsequent review still are unresolved.

exchange of offers.[7] As negotiation actually is a form of decision making – i.e. joint decision making with conflicting preferences (Mumpower and Rohrbaugh, 1996) – the constructs in negotiation theory correspond to constructs in decision theory and are only termed differently. The issues and options of a negotiation are the attributes and values of a decision problem. The set of possible agreements, from which one has to be selected mutually as agreement by the negotiators, corresponds to the alternatives a decision maker faces in a decision problem, etc.

3.1.1.1 Number of issues

One attribute of the negotiation problem is the number of its issues, which can be thought of as blanks in a contract that have to be filled with values to reach an agreement. Negotiation problems can consist of only one issue (e.g. the price for an otherwise determined product or service) or it can consist of multiple issues on which agreement has to be reached. 24 of 37 reviewed studies use one or more (but exclusively) multi-issue negotiation problems in their simulation, eleven studies one or more (but exclusively) single issue negotiation problems and two studies use both single- and multi-issue negotiation problems.

3.1.1.2 Set of possible agreements

Within each of the issues there exist different options (at least two discrete ones) which the negotiators could settle for, otherwise there would be no decision problem as the result would be mandatory. The possible values within an issue can either be discrete (e.g. the color of a good) or continuous (e.g. time or money). In the case of a discrete issue there exists a finite – however sometimes large – number of possible options to settle the dispute while in the case of a continuous issue the number of possible options is infinite. All combinations of options for each issue (i.e. their Cartesian product) form the set of possible agreements. Therefore in negotiation problems consisting exclusively of issues with discrete options the number of possible agreements for the negotiation problem is finite, while it is infinite when at least one issue has continuous options. To reduce complexity it often makes sense to discretize the options of an issue that is continuous by nature, as for instance time or money.[8] This can be done without loss of generality when there is a minimal measurement unit or the negotiators in their preferences distinguish only between certain sets of options and are indifferent between options within these sets. Such discretization for example results in days, minutes, or milliseconds (depending on the domain of the negotiation problem and the negotiators' preferences) as unit of measure when coping with the otherwise continuous issue of time.

A restriction on the set of possible solutions is, that its elements have to be, least in principal, acceptable to all participants in the negotiation, i.e. the options have to respect the reservation levels of all parties involved. For example a family with four children, when buying a family car,

[7]In their classification Jennings et al. (2001) term this concept 'negotiation object'. However, they only consider single-issue negotiations in their studies, where objective and subjective valuation of the single issue coincide, when considering multi-issue negotiations also the preferences over these multiple issues have to be considered, and consequently 'negotiation problem' is the more adequate term for this component of automated negotiations.

[8]The reduction of complexity achieved through discretizing issues with continuous options refers only to the number of possible solutions of the negotiation problem – which becomes finite while being infinite otherwise – still the negotiation problem could remain complex in other aspects.

will never negotiate over a car with less than six seats even if the car seller is willing to sell them a two seated sports car. Moreover possible solutions that provide inferior utility than the parties BATNAs will never be chosen as agreement from the perspective of individual rationality of the parties – i.e. in case such a solution is the only to end the negotiation with, the party will prefer its BATNA and non-agreement in the negotiation (Raiffa, 1982; Sebenius, 1992; Raiffa et al., 2002; Kersten, 2007).

Twelve studies use continuous options for all issues, most often these studies consider a single issue (e.g. the price of a good or service) in their simulation of negotiation and rely on time-based strategies, where the offer out of the infinite set of possible solutions is determined as a function of time. Three multi-issue negotiations with both kinds of issues, those with discrete and those with continuous options are simulated in the reviewed literature. Here again the continuous issue is most often the price. Finally 22 studies use negotiation problems that exclusively consist of issues with discrete options.

3.1.1.3 Negotiators' preferences

The participants in the negotiation have certain preferences over these different possible solutions indicating which they prefer over others and between which they are indifferent. Often these preferences are measured and represented in form of an utility function that assigns an utility value to each of the elements of the set of possible solutions. Alternatives with higher utility values are preferred over alternatives with lower utility values, and the negotiator is indifferent between alternatives of same utility. A commonly known and widely applied utility function for multi-issue problems is the additive utility function (Keeney and Raiffa, 1993), which assumes preferential independence of the issues, meaning that preferences for a given option in one issue are not influenced by changes in the options of other issues. Let X be an offer, and therefore a possible agreement, in a multi-issue negotiation, then $X = x_1, \ldots, x_n$, i.e. the offer is a vector of specific options x_i for each of the n issues $i \in (1, \ldots, n)$ of the negotiation. To determine the utility of the offer $u(X)$ the partial utilities $u_i(x_i)$ for the options in each issue are weighted with a measure of importance for the specific issue w_i, $i \in (1, \ldots, n)$ – where these weights have to satisfy the condition $\sum_{i=1}^{n} w_i = 1$. Partial utilities for each issue are then aggregated by addition to receive the overall utility of the offer (3.1)

$$u(X) = \sum_{i=1}^{n} w_i u_i(x_i) \tag{3.1}$$

Often partial utility functions are scaled to unity $(\overline{u_i(x_i)} = 0, \underline{u_i(x_i)} = 1)$ and therefore also the utility for a package can range between $\underline{u(X)} = 0$ – for the package consisting of the worst options in all issues – and one $\overline{u(X)} = 1$ – for the package with the most preferred options in all issues. Different approaches were proposed and experimentally implemented to let negotiation agents elicit the preferences of their users with the objective to later use them with the software agents that negotiate on behalf of the users according to these preferences. Guo et al. (2003) for example propose an evolutionary algorithm to elicit preferences for a multi-issue negotiation in form of a multi-attribute utility function. Another approach was proposed by Luo et al. (2006), who use a default-then-adjust method where users are asked to modify domain dependent default preferences to their trade-off preferences between two issues.

The preferences of humans over the same negotiation object can be various and complex. Partial utility functions might not simply be linear functions of the objective values of the options in the issue, but could take the form of concave, convex, or non-monotonic functions of the options in the issue, as found in some experimental studies (Vetschera, 2006). It is therefore necessary to verify which kind of preferences were used in simulations of automated negotiation and whether the authors accounted for the real diversity of possible preferences in their simulation setting. All but one study – Bosse and Jonker (2005), who elicit the preferences of humans as input for their software agents and in addition performed human versus human, human versus computer, and computer versus computer experiments – assumed one or various preference functions for their agents. Oliver (1996) compares the results of experiments with software agents designed by evolutionary computing to the results of experiments with human agents for equal preferences, which however were predefined for the human agents.

3.1.1.4 The negotiation problem structure

The joint evaluation of the negotiation object, according to the preferences of the negotiators, determines the structure of the negotiation problem. In the case of bilateral negotiations a graphical representation of the negotiation problem structure can be derived easily by plotting the utility of every possible solution for both negotiators on the abscissa and ordinate in a two dimensional graph, respectively (Mumpower, 1991; Mumpower and Rohrbaugh, 1996).

Figures 3.4 to 3.8 illustrate negotiation problem structures for five different combinations of hypothetical negotiators' preferences over a negotiation object with two issues (A and B) and six discrete options (1 to 6) for settlement in each. In these figures the utility for an offer is calculated for both negotiators – using the additive utility function (3.1) – and then plotted as one point in the two dimensional graph, doing this for all $6^2 = 36$ possible offers yields the graphical representation of the negotiation problem structure.

The two extreme cases are on the one hand perfect distributiveness of the negotiation problem structure (see Figure 3.4), where the negotiators have exactly the same weights and directly opposing slopes of the partial utility functions in all issues – i.e. diametrically opposed preferences over the negotiation object. The second extreme is perfect integrativeness, where the negotiators have the same slopes of partial utility functions and the same weighting of issues (see Figure 3.5, where all possible outcomes are Pareto-dominated by one outcome that is the joint optimal solution). Though it is often argued that single-issue problems are distributive and multi-issue problems integrative, this need not be the case. Single issue problems may be integrative – in case of a non-monotonic utility function – and multi issue negotiation distributive – as shown in Figure 3.4 – (Kersten et al., 2000). Some degree of integrativeness can steam from different weights – Figure 3.6 –, different slopes of the partial utility functions (like concave or non-monotonic functions) – Figure 3.7 –, or consistent interests in some issues – Figure 3.8 – such that there exists no conflict about the optimal outcome in this issue, and of course combinations of these causes.

From the negotiation problem structure one can derive the complexity of negotiation problem and the level of conflict between the negotiators, which obviously influence the potential performance of negotiators – both human and artificial ones. Therefore the negotiation problem structure has to be considered when evaluating the performance of human and software agents as well as in comparisons across studies.

The negotiation problem structures resulting from the preferences of the negotiation agents are purely distributive (including all studies with only one issue) such that a loss for one agent is a gain for the other one in twelve studies. 21 studies simulate integrative negotiation problem structures, two studies (Goh et al., 2000; Choi et al., 2001) use both integrative and distributive negotiation problem structures, and in other two cases (Oliver, 1996; Tu et al., 2000) also structures where the preferences are perfectly consistent are used in different settings of the simulation.

	negotiator 1		negotiator 2	
issue	A	B	A	B
weight	0.5	0.5	0.5	0.5
1	0.0	0.0	1.0	1.0
2	0.2	0.2	0.8	0.8
3	0.4	0.4	0.6	0.6
4	0.6	0.6	0.4	0.4
5	0.8	0.8	0.2	0.2
6	1.0	1.0	0.0	0.0

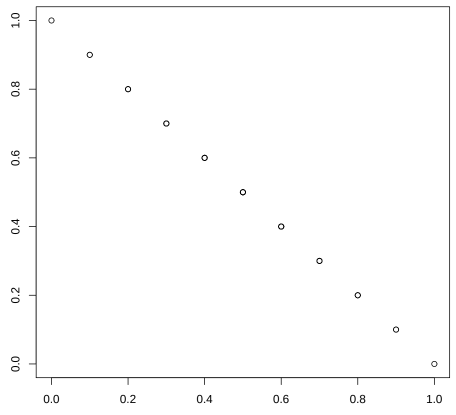


Figure 3.4: Distributive negotiation problem – diametrically opposed preferences

	negotiator 1		negotiator 2	
issue	A	B	A	B
weight	0.5	0.5	0.5	0.5
1	0.0	0.0	0.0	0.0
2	0.2	0.2	0.2	0.2
3	0.4	0.4	0.4	0.4
4	0.6	0.6	0.6	0.6
5	0.8	0.8	0.8	0.8
6	1.0	1.0	1.0	1.0

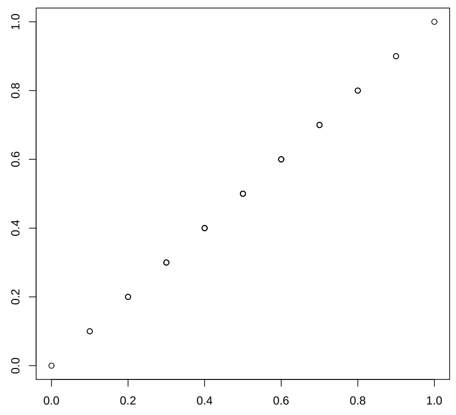


Figure 3.5: Integrative negotiation problem – consistent preferences in all issues

	negotiator 1		negotiator 2	
issue	A	B	A	B
weight	0.3	0.7	0.7	0.3
1	0.0	0.0	1.0	1.0
2	0.2	0.2	0.8	0.8
3	0.4	0.4	0.6	0.6
4	0.6	0.6	0.4	0.4
5	0.8	0.8	0.2	0.2
6	1.0	1.0	0.0	0.0

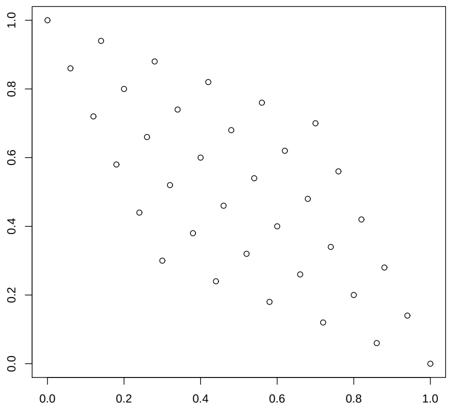


Figure 3.6: Integrative negotiation problem – different weights

	negotiator 1		negotiator 2	
issue	A	B	A	B
weight	0.5	0.5	0.5	0.5
1	0.00	0.00	1.00	1.00
2	0.35	0.35	0.95	0.95
3	0.65	0.65	0.85	0.85
4	0.85	0.85	0.65	0.65
5	0.95	0.95	0.35	0.35
6	1.00	1.00	0.00	0.00

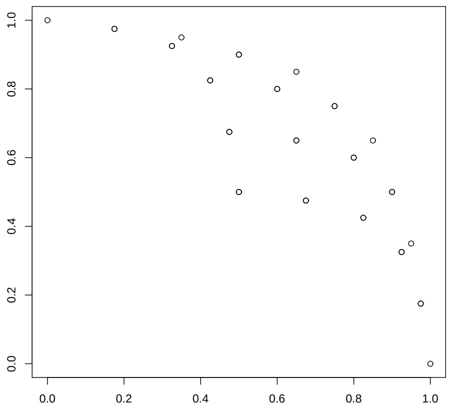


Figure 3.7: Integrative negotiation problem – concave partial utility functions

	negotiator 1		negotiator 2	
issue	A	B	A	B
weight	0.5	0.5	0.5	0.5
1	0.0	0.0	1.0	0.0
2	0.2	0.2	0.8	0.2
3	0.4	0.4	0.6	0.4
4	0.6	0.6	0.4	0.6
5	0.8	0.8	0.2	0.8
6	1.0	1.0	0.0	1.0

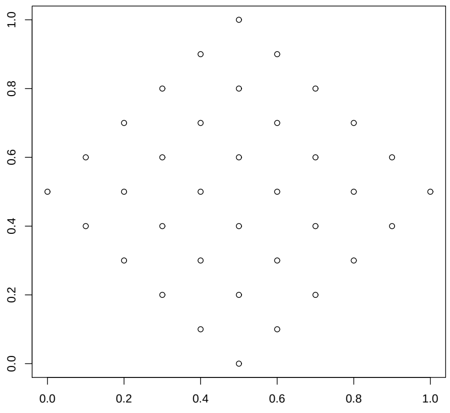


Figure 3.8: Integrative negotiation problem – consistent preferences in one issue

The negotiation problem structure is not necessarily constant over time in real world negotiations. For example the issues and options or the preferences over these issues (weight) and options (partial utilities) could change over time. In this case the negotiation problem structure is variable otherwise it is fixed. We here consider – as mentioned above – only fixed negotiation problems. The reason for doing so is (i) first there are no studies simulating negotiations with variable problems – though there exist some first approaches that deal with the question of how to cope with such problems (Faratin et al., 1999a,b)[9]) and (ii) that it does not make sense to alter preferences or the object during the negotiation due to the fast proceeding of automated negotiation.

Furthermore the payoffs and therefore the utility of the possible agreements need not be certain but could be risky when they depend on some future events the negotiators cannot influence but only know their probability of occurrence. Though all reviewed studies use certain payoffs, one could elicit the negotiators risk attitudes and then apply methods from multi-criteria decision making under risk to determine expected payoffs as basis for the negotiations (Keeney and Raiffa, 1993).

3.1.2 Interaction protocol

Negotiations generally consist of one or more turns in which offers – as tentative settlement proposals – as well as other messages are communicated. While the exchange of offers is a constituting feature of negotiations, the particular rules governing these offer sequences can vary (Cranor and Resnick, 2000). The interaction or negotiation protocol is the set of rules that governs the interaction between the participants in a negotiation, these participants can either be exclusively human, exclusively software agents, or combinations of both. The protocol determines the possible states in a negotiation, the actions particular participants can execute in each of these states, and the events that cause state transitions (Jennings et al., 2001).

In face-to-face negotiations or technology-mediated negotiations between humans – e.g. video conferencing, telephone, e-mail, or fax – such interaction protocols are less strict and rigorous as people easily can deviate from protocols, e.g. through interrupting each other, withdrawing offers, or finding agreements after negotiations were broken off – for example when just after the deadline a party proposes an acceptable offer. In case of human interaction general social codes – e.g. not to interrupt a person when speaking – or norms particularly related to the domain of negotiation – e.g. that offers once put on the 'negotiation table' should not be withdrawn – establish and impose a form of tacit interaction protocol causing some type of sanctions if violated (Bartos, 1977).

However, the more automation is used in negotiation and the more activities are delegated to software agents, the fewer flexibility concerning the negotiation procedure remains. Accordingly in automated negotiation – where the negotiation tasks of a human are completely assumed by a software agent – the flexibility is lowest compared to other alternatives and fully specified interaction protocols (so-called closed protocols) are not only necessary (Kersten and Lai, 2007), but

[9]If the negotiation object is allowed to be manipulated by the software agents the interaction protocol also has to determine the rules for negotiation object manipulation, i.e. the rules that govern the addition or elimination of issues and options during the negotiation process.

also can be enforced by the software implementation of the interaction protocol itself. This software implementation of the interaction protocol can restrict the possible actions of the software agents for example by baring participants from actions in certain states or otherwise completely ignoring actions out of the bounds of the protocol (Cranor and Resnick, 2000).

3.1.2.1 Chronology of communication

One important attribute of the interaction protocol is the timing of offers. Offers can either be proposed simultaneously by the parties such that no party knows the current offer of the opponent in the current round before sending the own offer, or sequentially such that offers can be formulated in response to the opponent's offers. In the later case the protocol has to determine which party has to start the offering sequence. Examples for simultaneous offering protocols can be found in the early game theoretic approaches on strategic bargaining like the Nash demand game (Nash, 1953), the Nash-Zeuthen bargaining game formulated by Harsanyi (1956), or the closely related iterated prisoner's dilemma (Axelrod, 1980a,b). Sequential offering protocols were proposed by Cross (1965) first, and later adopted by Ståhl (1972) and Rubinstein (1982). The alternating turn protocol – where software agents alternate in taking turns where they can perform actions – is dominant in the recent literature on simulation of automated negotiation. Only three of the reviewed studies use other protocols than the alternating turn protocol. Wollkind et al. (2004) use the Nash-Zeuthen bargaining protocol, while in two other studies (Henderson et al., 2005; Nawa, 2006) turn taking is not absolutely sequential, as one turn by one agent is not necessarily followed by a turn of the opponent, but agents can decide to repeat messages or pause in taking their turns in these simulations.

3.1.2.2 Configuration of offers

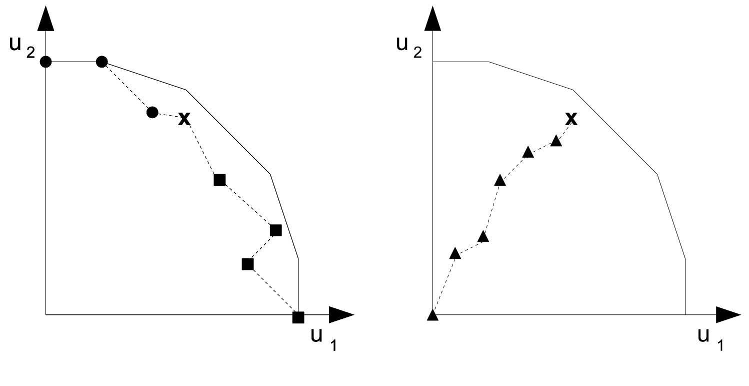
In single-issue negotiations offer configuration poses no problem. An offer by definition – to be a proposal for settlement – can be any feasible option for this single issue. In case of multi-issue negotiation problems, however, the protocol has to determine what constitutes an offer. The two extreme values for this attribute are package offers, where offers have to include proposals for all issues, and agenda setting, where issues are negotiated one after the other according to a specified order called agenda, just like a series of single-issue negotiations. In the later case an agenda has to be determined, which can be either done by the protocol or endogenously agreed on in a meta-negotiation of the negotiation agents (Raiffa, 1982). Clearly any combination in between these extremes is also feasible, for example different offers could cover options for all, some, or only one of the issues of the negotiation problem.

Most of the interaction protocols used in the reviewed simulation studies require package offers in multi-issue negotiations as they allow for trade-offs between issues, which in turn make logrolling procedures possible. In logrolling worse options in lower priority issues are traded for better options in higher priority issues, which finally should lead to mutually beneficial settlements (Pruitt, 1981). An other approach, dominant in game theory, is the aggregation of preferences over multiple issues to one single utility value of a package which reduces the multi-issue to a single-issue problem (Lang and Rosenthal, 2001). In recent years, however, game theorists started analyzing multi-issue negotiations with Rubinstein's (1982) alternating offer protocol without aggregating the packages to utility values but in an issue-by-issue fashion under fixed agenda (Fershtman, 1990). Other studies investigate how such agenda for issue-by-issue negotiations can be determined (Busch and Horstmann, 1999a). Furthermore agenda protocols and package offer protocols are compared for very specific bargaining problems (non-cooperative, zero-sum scenarios with costs of delay) (Lang and Rosenthal, 2001; Inderst, 2000).

Agenda protocols at the moment are used with analytical models in game theory only, in simulation studies most often only one issue is considered. When the negotiation problem consists of more than one issue, package offering is imposed by the protocol in all but one of the 26 studies that simulate multi-issue problems, so that package offers, consisting of options for each of the negotiated issues, are required by the protocol. Wasfy and Honsi (1998) use a slightly different protocol. In their approach, agents exchange offers in only one of the issues of the multi-issue negotiation problem. The current demand of the agents is stored and agreement is reached when the demands of the agents are equal in all issues – so it is a procedure of single-issue offers in a multi-issue negotiation problem, however not following an agenda. In their approach issues for which consistent demands are achieved during the negotiation are considered as solved and the negotiation continues on the remaining issues of the negotiation problem only.

3.1.2.3 Progression of the negotiation

The progression of negotiation determines whether the protocol demands parties to start from inconsistent points and reach an agreement through concession making, or whether they start from a common basis and try to improve this basis through a number of tentative agreements that dominate the previous ones until no Pareto-improvements are left (or can be found in negotiations that settle for inefficient solutions). These two approaches are illustrated in Figure 3.9.



(a) Concession-based progress (b) Improvement-based progress

Figure 3.9: Concession-based and improvement-based progress in negotiations

Teich et al. (1994) call the first approach concession-based models and the second improvement-seeking models. Concession-based progression is the predominant value for this attribute of the interaction protocol in the reviewed simulation studies of automated negotiation. Scholars argue, that when negotiators have conflicting interests they have to make concessions from their initial starting point if any agreement is to be reached. An alternative to this predominant perspective, that one has to start negotiations with an extremely high demand and then continuously lower this demand by concessions, is the improvement-based progression. The improvement-based approach to negotiations is closely related to the 'single negotiation text' procedure proposed by Fisher (1978). The agents first determine a common basis and use this as a reference point from which to move to other solutions preferred by both. Only Klein et al. (2003) use a pure improvement-based negotiation protocol, where a mediator proposes possible Pareto-improvements to the status quo to the agents, which they can either accept or reject.

It is necessary to determine this progression attribute by the protocol as otherwise any tentative agreement which serves as a basis for later improvement in improvement-based approaches would be rendered as a final agreement and terminate negotiations in a concession-based protocol. While it is impossible to use both forms of progress simultaneously in a negotiation, they can be used sequentially. One form of such a combination is implemented in the negotiation support system `Inspire` (Kersten and Noronha, 1999a), which provides the opportunity for Pareto-improvement in a post-settlement phase of negotiations, after the negotiators reached a first agreement typically by concession-based procedures. Such post-settlement phases, however, are not implemented in negotiation protocols for automated simulation yet, but only one form of progression in negotiation is followed, most often the concession-based progress as mentioned above.

3.1.2.4 Types of actions

It is necessary to determine the types of actions the protocol allows the software agents to execute. While various kinds of actions are definitively important in negotiations, the most important action is the communication of offers and counter-offers (Tutzauer, 1992). However, there can also occur actions other than offer-exchange in negotiations. Two actions necessary for the termination of a negotiation are the transmission of acceptance and termination messages, other actions could be rejections of offers (Bartos, 1977) – leading to non-alternating offer sequences – or the submission of messages containing (logical) argumentation.

While in bilateral negotiations the disputing parties – buyer and seller, employee and employer, management and union, etc. – have a symmetric set of possible actions, procedures closely related to negotiation such as the ultimatum game or most forms of auctions limit the option to perform certain actions to only one side. In the ultimatum game only the first player is allowed to formulate take-it-or-leave-it offers which the second player can either accept or reject, and in most auction protocols – here the double auction is an exception – also only one side is allowed to make offers. This side is the buyer side in English, Dutch, or Vickrey auctions – to name only the most commonly known –, but it could also be the seller side as it is the case in a reverse auction for example.[10]

In four of the reviewed studies (Somefun et al., 2004, 2006; Wollkind et al., 2004; Zeng and Sycara, 1998) the set of possible actions of the agents is limited to the exchange of offers. In these studies an agreement is determined by the protocol and reached when the offers of the software agents are consistent. To avoid the necessity of exit messages in three studies it is guaranteed

[10]Consult Klemperer (2004) and Güth et al. (1982) for information on auctions and the ultimatum game, respectively.

that the agents always reach an agreement, and (Wollkind et al., 2004) use the Zeuthen-Nash bargaining protocol where the protocol determines the end of the negotiation.[11] Wasfy and Honsi (1998) allow agents to send, messages about their power, which is determined randomly by the environment and directly affects the opponent's situation and strategy, in addition to their offers. The software agents of Klein et al. (2003) can only accept or reject offers proposed by a mediator as a Pareto-improvement of the status quo, but cannot themselves formulate offers. As this study uses an improvement-based protocol the agents start negotiation from a common starting point, thereby agreement is guaranteed and no exit messages are necessary. If there is no progress in the negotiations the agents stay with the best solution available so far. 13 studies use protocols where the set of possible actions of the agents includes offer-exchange and acceptance of the opponent's offer. In the studies of Chen et al. (2005b) and Chao et al. (2006) agents, with their own offer, provide some information about their satisfaction with the opponent's last offer. In the other 16 studies the possible actions of the agents are offer-exchange, and termination of negotiation by both agreement and break off.

3.1.2.5 Abandonment of negotiations

If a negotiation does not result in an agreement the abandonment of negotiations can be induced by the protocol after a given period of time – the negotiation deadline – or by a break-off probability at the end of each turn. Alternatively the termination of the negotiation can also be the result of an endogenous decision of the software agents. Clearly a combination of these options is also feasible, where the software implementation of the protocol breaks off negotiations at a deadline unless the agents did this before.

In most (22) of the studies a commonly known deadline determined by the protocol and executed by the protocol or the agent is used to terminate negotiations, in two studies break-off probabilities after each round were used for this purpose. In six other studies agreement is guaranteed by the experimental settings and as all negotiations end with an agreement abandonment of negotiations is not discussed. In four studies the protocol terminates negotiations unless agents did so before. As mentioned above (Wollkind et al., 2004) uses the Zeuthen-Nash bargaining procedure where the termination criteria is defined by a protocol rule (no concessions of both parties in one round) and (Klein et al., 2003) use an improvement-based approach where negotiation end if no improvements can be found any more, in these two studies protocol rules exclusively determine when the negotiation ends. Only in the study of (Cheng et al., 2006) the decision to break off negotiations is made endogenously and exclusively by the software agents, here agents decide to break off negotiations when there is no progress for a certain period of time.

3.1.3 Decision making algorithms of software agents

A software agent's decision making algorithm or strategy describes its internal reasoning. It is employed to, acting in line with the interaction protocol, achieve the goals of the human user the software agent represents in the negotiation. Rosenschein and Zlotkin (1994) provide a good definition:

[11] In the Zeuthen-Nash bargaining protocol negotiation ends if both parties refuse to make a concession in one round.

> Given a negotiation protocol, a negotiation strategy is a function from the history of the negotiation to the current offer that is consistent with the protocol. It specifies precisely how an agent will continue (what move it will make) given a specific protocol, and the negotiation up to this point. The strategy is static, in the sense that is is chosen ahead of time (before the negotiation begins). It specifies the agent's reaction for every possible course of events. For an automated agent (what we are interested in), that just means that it has been programmed ahead of time, and the program itself isn't altered; of course, the decisions could be based on dynamics of the negotiation itself. (Rosenschein and Zlotkin, 1994, p.41)

The strategy of the software agent therefore determines – along the course of the negotiation – the opening offer and subsequent counter-offers by some offer generation strategy, and when to accept an offer or break off negotiations. It determines, constricted by the restrictions imposed by protocol, influenced by the actions of the opponent software agent, and directed by its parametrization and the user's preferences, how the negotiation starts (opening offer), how it proceeds (offer generation), and when and how it ends (acceptance or break off conditions). The offer generation (opening offer and subsequent counter-offers) and evaluation (when to agree or break off) decisions have to be made in line with the preferences of the human user for the negotiation problem at hand. These preferences therefore by some way have to be made available to the agent.[12] Besides the preferences of its user the software agent's decision making can be based on deterministic or stochastic rules, actions of and beliefs about the opponent, etc. The determination of the strategy of the software agent therefore is a complex and critical task.

Derived from the above discussion, the attributes of a decision making algorithm for software agents in automated negotiation are the decision about the opening offer, algorithms for the generation of subsequent counter-offers, and finally the criteria for the termination of the negotiation by either acceptance of an offer or break off of negotiation. The methods currently used for starting and ending automated negotiations do not differ much among the reviewed simulation studies, however a variety of approaches are used to generate offers during the negotiation process. We will therefore only briefly mention the options proposed for the former attributes of the strategy and discuss the approaches for offer generation by software agents in automated negotiation in more detail.

An opening offers can either be an extreme – i.e. the highest possible in a concession-based or the lowest acceptable in an improvement-based protocol[13] –, it could be generated as all other offers, or by some other method. The opening offer in the reviewed studies is in most cases an extreme one, more explicit the offer with the highest possible utility as the majority of the studies follow the concession-based progression approach. These extreme offers sometimes are determined by rules (e.g. in learning, time-based, trade-off, or imitating strategies), however they also emerge in strategies developed by evolutionary computing, which come to the same result, i.e. to start the negotiation with extreme offers.

[12]As briefly discussed in the Section 3.1.1.3 there exists some first attempts to make software agents elicit their users' preferences or learn them from previous experience (Guo et al., 2003; Luo et al., 2006), besides just asking the user to input them.

[13]Consult the Section 3.1.2.3 for details on the progression of negotiations determined by the interaction protocol.

Concerning the termination of the negotiation by acceptance only few alternatives are considered in the reviewed simulation studies. One option is the use of thresholds – as a kind of aspiration level – to determine the acceptability of offers. These thresholds can either be constant during the negotiation or vary over time, as in most strategies determined by means of evolutionary computing. Offers that provide higher utility than this threshold are accepted by the agent. Gerding et al. (2003) deviate from this procedure in formulating different functions that determine the probability of acceptance as a function increasing in the utility of the opponent's offer – as a 'social extension' of their agents. An alternative option to determine which offer of the opponent to accept, applied by most approaches other than evolutionary computing, is the acceptance criterion that the opponent's current offer has to provide higher utility than the next own offer to be sent would provide.

Just like for termination by acceptance, for termination by break off of the negotiation without agreement only few options are considered in the reviewed simulation studies on automated negotiation. Moreover, this 'decision' is implemented quite unsatisfactory in many of the reviewed studies, as in most cases – especially for time-based strategies – the software agent's strategy dictates to break off negotiations when a commonly known deadline for negotiation expires, which could also be imposed by the protocol and therefore actually requires no decision from the agent. In some other studies agents decide to terminate the negotiation when reaching a reservation level (Winoto et al., 2005; Lawley et al., 2003b,a; Lin and Chang, 2001; Goh et al., 2000), or the negotiation fails to progress (Cheng et al., 2006). Only in these later cases the decision to break off negotiations actually is a decision endogenously made by the software agents according to their strategies.

The subsequently mentioned approaches for offer generation by software agents in automated negotiation are those extracted from the reviewed literature, which are in turn derived from related fields. Research in negotiation has a long tradition – as can be seen from the short review in the introduction –, therefore scholars studying automated negotiation argue that developers of software agents for automated negotiation do not have to 'reinvent the wheel' (Kraus, 1997; Jennings et al., 2001). But, when determining the software agents' decision making algorithms, one can redeploy the insights of other fields like game theory, evolutionary computing, behavioral social science, or distributed artificial intelligence. When basing agent's decision making algorithms on heuristics derived e.g. from behavioral social science, however, simulations of these agents in various settings are necessary to determine their performance, which is – different from other fields as e.g. game theory – unclear in advance due to the heuristic nature of the resulting strategy (Kraus, 1997; Jennings et al., 2001), as Henderson et al. (2003, p.137) state: *'A significant problem in distributed e-commerce applications is the choice of algorithms used to carry out automated negotiation on behalf of a client. Even very simple algorithms can have behavior which is acceptable in a restricted scenario but which might be unpredictable in a more liberal environment.'*

3.1.3.1 Evolutionary computing-based strategies

Agent strategies developed by evolutionary computing were the very first applied in simulation of automated negotiation (Oliver, 1996), and still hold a large share of software agents in simulations of automated negotiation. The rational behind the use of evolutionary computing to develop agents is its successful previous application for problems closely related to negotiation, like the iterated prisoner's dilemma or double auctions (Oliver, 1996). In their most basic form agent strategies developed by evolutionary computing consist of a sequence of utility thresholds and offers – sometimes also other actions like exit messages are implemented –, so called sequential threshold rules, for the negotiation problem (Oliver, 1996). These combinations of thresholds and offers are encoded in so called 'chromosomes'. On receiving an offer of the opponent the software agent evaluates this offer using an utility function that indicates its user's preferences, if the offer is above the threshold for the round, as determined in the chromosome, the agent accepts it otherwise it submits the counter-offer for this round determined in the chromosome. However, the chromosomes encoding the strategy, need not only consist of offers, i.e. one value for single-issue negotiations or package offers for multi-issue negotiations. Beyond the basic form mentioned above whole finite state machines can be encoded in chromosomes, which then consist of states of the finite state machine, actions for each state, as well as conditions for state transitions (Tu et al., 2000).

At the beginning of the 'evolutionary process', by which the agent strategies are improved, the chromosomes are determined randomly. After a negotiation is conducted the utility of the outcome of negotiation can be used as a measure of fitness of the agent's strategy. Evolutionary computing uses, basing on this fitness measure and in analogy to biological evolution, evolutionary operators to improve the chromosomes. These evolutionary operators are:

- *Selection:* Strategies with highest fitness are selected as parts of the next generation.
- *Cross-over:* Strategies with highest fitness are allowed to produce 'children' which then are also parts of the next generation. These children strategies are combinations of the parent strategies.
- *Mutation:* With a small probability mutation changes some components of the strategy chromosomes of the next generation, so that they are different from their parents or the selected strategies.

The repeated application of these evolutionary operators over many generations of agent strategies was demonstrated to lead to agent strategy generations that reach Pareto-optimal outcomes in simple negotiations and match human performance in more complex settings (Oliver, 1996).

3.1.3.2 Imitating strategies

Another offer generation mechanism, again derived from strategies for the iterated prisoner's dilemma, are imitating – also called responsive or reciprocative – strategies, which base their decision making on the previous behavior of their opponent. The most commonly used imitating strategy for automated negotiation is a strategy analog to the tit-for-tat strategy for the iterated prisoner's dilemma. Tit-for-tat starts with a cooperative move and then mirrors the opponent's move of the previous round in the iterated prisoner's dilemma. Translated to negotiation problems this means reducing the demand when the opponent reduced demand in his previous offer (i.e. conceding) or otherwise demanding more or the same if the opponent did so (i.e. demanding or insisting, respectively) – all in terms of the utility of offers. Scholars argue that tit-for-tat strategies have many conveniences like being responsive in suddenly retaliating and therefore cannot be exploited easily, but fast forgiving. However, simulation studies of the iterated prisoner's dilemma also showed that with small uncertainties about the opponent's true moves (noise) two such imitating strategies can get stuck in an uncooperative vendetta. To circumvent uncooperative phases moving averages or exponential smoothing (Filzmoser, 2007) can be used. Such uncertainties about the opponents true move are always present in negotiations when the opponent's preferences are his private information.

3.1.3.3 Trade-off strategies

Trade-off offer generation mechanisms aim to propose offers of the same utility but of different option configuration in the issues. Some strategies also aim to establish offers with an option configuration as similar as possible to the opponent's last offer – using Euclidean, Hamming, or fuzzy distance measures – to explore opportunities for mutual benefit. Such trade-offs are the individual counterpart of dyadic logrolling in negotiations (Pruitt, 1981), where the parties trade worse options in less important issues for better options in more important issues with the aim to achieve mutual benefit.

3.1.3.4 Learning-based strategies

Learning-based offer generation mechanisms hold a model of the opponent in mind, which consists of parameters that measure the opponent agent's preferences – e.g. its reservation or aspiration level – or negotiation attitudes – e.g. its concession rate or concession function. Initial expectations about these parameters of the opponent's strategy and preferences are updated with the information acquired during the course of negotiation. For this learning mechanism several approaches are applied in simulation studies on automated negotiation, like for instance Q-learning or Bayesian updating. Basing on the information acquired by learning, the software agents calculate the optimal response to the opponent's actions in order to maximize their expected outcome of negotiations – i.e. they economize on learned information.

3.1.3.5 Time-based strategies

Time-based offer generation mechanisms determine (the utility of) the offer as a function of time. Time-based strategies are derived from observations of real and experimental human negotiations – as reported e.g. by Pruitt (1981). The functions used to generate offers can have different forms as illustrated in Figure 3.10.

Typically agents start with their best offer affording highest utility – due to the concession-based approach applied in most simulation studies – and then concede to the reservation level of the negotiator which they offer at the deadline of the negotiation. The negotiation ends when two consistent offers are proposed by the negotiators or the deadline is reached without such an agreement. Typical forms for such offer functions discussed in literature are linear functions, that make concession steps of the same size – i.e. have a constant concession rate – during the whole negotiation as indicated by the middle line in Figure 3.10, convex functions, resulting in larger concessions at the beginning and smaller concessions towards the end of the negotiation – so called conceder strategies represented by the lower line in Figure 3.10 –, or concave functions, making only small concessions at the beginning but considerable concessions towards the end of the negotiation. This last type of concession functions are also called Boulware strategies and are depicted as the upper line in Figure 3.10.

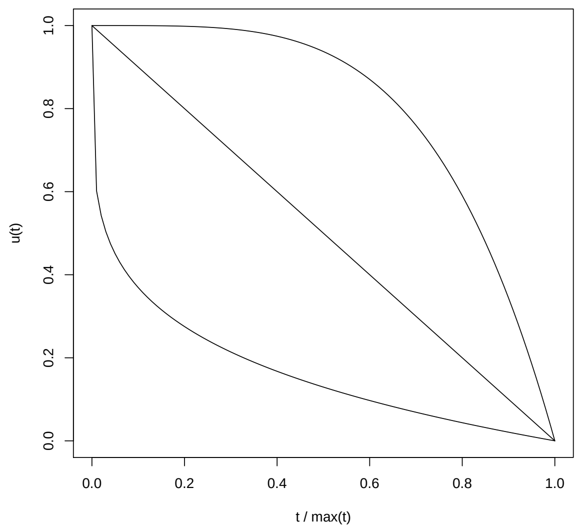


Figure 3.10: Concession functions of time-based strategies

3.1.3.6 Other strategies

Offer generation based on other than the above mentioned approaches are often very simple 'dummy' strategies employed to verify the performance of strategies based on evolutionary computing, imitation, learning, or time. Such simple approaches continuously make concession steps of a predetermined magnitude (Park and Yang, 2004; Zeng and Sycara, 1998), by reducing the gap in the demands by some fixed proportion (Somefun et al., 2004, 2006), or simply take random steps (Winoto et al., 2005; Henderson et al., 2003). In some simulation studies, however, such 'other' strategies are also quite sophisticated ones but build on approaches different from those mentioned above. Such alternative sophisticated strategies are not so frequently used and we therefore only briefly enumerate them: hill-climbing and annealing (Klein et al., 2003), least-cost-issue concession (Wasfy and Honsi, 1998), Zeuthen-risk strategy (Wollkind et al., 2004), case experience-based strategies (Paurobally et al., 2003), or competition-based strategies (An et al., 2006).

In eight studies evolutionary computing are used to determine the offer generation of the software agents. While most of them use simple binary chromosomes with offers and thresholds (sequential threshold rules), Tu et al. (2000) and van Bragt and La Poutre (2003) encode finite state machines in form of chromosomes for evolutionary computing. 14 studies employ imitating strategies, which in most cases acted purely reciprocating. However, some variations of imitating strategies also employed include those proposed by Krovi et al. (1999), which are: (i) reciprocating strategies that fully reciprocate the behavior of the opponent, (ii) cooperative strategies that more than match the opponents concessions, and finally (iii) exploitative strategies that concede less than the opponent. Trade-off mechanisms employing fuzzy-similarity measures, Pareto-search, or other similarity measures to make offers close to those of the opponent were used in ten studies – such trade-off mechanisms for multi-issue negotiation problems find increasing attention in recent years. Trade-off mechanisms typically are combined with some rules about how to determine changes in the demanded utility level, if no trade-off offers (with same utility) are left to propose, to form the final strategy of the agent. In nine studies learning agents with a variety of learning techniques were investigated. The employed learning techniques range from Bayesian network updating (Zeng and Sycara, 1998) over neural networks (Rau et al., 2006; Park and Yang, 2004) and pattern recognition (Lin and Chang, 2001) to Q-learning (Cardoso and Oliveira, 2000). Another 14 studies rely on time-based concession functions to determine offers.

As many studies derive strategies by a combination of approaches or compare different types of strategies in their simulation studies, the overall number of types of strategies used for offer generation in the studies not matches the number of studies reviewed. These comparisons lead to interesting results, which are informative for the design of decision making algorithms for software agents. Zeng and Sycara (1998) show that learning agents can exploit and outperform simple agents with continuous concession strategies. Deveaux et al. (2001) show that agents that hold a model and an expectation about the opponent's strategy in mind and adapt their strategy according to the information they gathered during the negotiation outperform time-based strategies. van Bragt and La Poutre (2003) find that strategies embodying a finite state machine improved by means of evolutionary computing can exploit time-based and imitating strategies, and Tu et al. (2000) state that strategies based on evolutionary computing with chromosomes encoding finite state machines do not perform significantly better than those with chromosomes encoding sequential threshold rules.

In addition to comparisons of different software agent strategies in terms of their performance, few studies also compare the results of software agents in automated negotiation (i.e. simulation results) to the results predicted by game theory (i.e. analytical solutions), to unsupported human negotiation experiments or experiments where humans are supported by negotiation support systems, or even let humans negotiate against software agents in negotiation experiments. Gerding et al. (2003) show that agents based on evolutionary computing reach the results proposed by game theory for rational agents in bargaining games. They propose and accept extreme take-it-or-leave-it offers at the final round of games with a finite number of rounds and reach results near the analytically derived subgame-perfect equilibrium (Rubinstein, 1982) in the first round when the game is characterized by a small probability of negotiation break off after each round. The results for human agent versus software agent comparisons, however, are somewhat disappointing for the field of automated negotiations. Oliver (1996) compares the performance of his software agents ,based on evolutionary computing, to the performance of humans in experiments for negotiations reported by Raiffa (1982) and Rangaswamy and Shell (1997). He finds that his software agents matched the performance of unassisted negotiators but are outperformed by humans that use negotiation support systems in integrative negotiation problems. Similarly Goh et al. (2000) find that in distributive negotiation problems there is no significant difference in performance for humans negotiating via a messaging system, via a negotiation support system that provides analytical support, or software agents that use simple continuous concession functions. In integrative negotiation problems ,however, negotiation support system users outperform messaging system users, which in turn outperform software agents. Bosse and Jonker (2005) use preferences elicited from human users to simulate automated negotiations and also find that the performance of software agents based on simple concession functions matches the performance of human negotiators if the agents face opponents of the same type – i.e. computer versus computer and human versus human. However, in an experiment where humans negotiated with software agents, that use the preferences elicited from humans as input, the human subjects reached significantly better outcomes than their opponent software agents.

3.2 Future challenges

This section discusses challenges for future research that, in our opinion, need to be approached for the advance of the field of automated negotiation, the implementation of its insight in operative systems, and its application in practice. These challenges, derived from deficiencies of existing simulation studies of automated negotiations, are organized along the components of automated negotiation in the next subsections and generally deal with the exploitation of the simulation technique's ability to cope with complex interdependencies and the avoidance of unnecessary assumptions in the context of automated negotiations.

Unlike game theoretic approaches that separate the components of automated negotiation[14], when simulating automated negotiation especially these interdependencies of the components can be studied. And there exist many of such interdependencies between the components of an automated negotiation. As mentioned in the review, the software agents' strategies are not only constricted by the protocol, but their performance also depends on the negotiation problem. The performance of an interaction protocol also might depend on the structure of the negotiation problem for which it is used, and different combinations of software agent strategies and interaction protocols might perform best for different negotiation problems. When comparing different strategies or protocols for automated negotiation these contingencies and interactions have to be considered.

Furthermore simplifying and restricting assumptions concerning the negotiation problem and the software agents' behavior – necessary in game theory to keep strategic interaction or mechanism design problems analytically solvable –, like common knowledge of the agents' preferences or rationality of the agents (Sebenius, 1992), can be avoided in simulation to some extent to make the problem more realistic. In our opinion when simulating automated negotiations, to test various system configurations before their implementation, one should attempt to exploit the computational performance provided by simulation techniques and therefore avoid unrealistic assumptions but model reality as good as possible and necessary.

[14]Branches of game theory investigate for a given negotiation problem and interaction protocol what the optimal (equilibrium) strategy of rational agents should be, given rational behavior of the opponent – strategic approach in bargaining theory –, or how interaction protocols should be configured to achieve a desired outcome for a given strategy and negotiation problem – mechanism design (Rosenschein and Zlotkin, 1994; Binmore and Vulkan, 1999; Cranor and Resnick, 2000).

3.2.1 Realistically complex negotiation problems

The inherent problem solving potential of negotiation lies with multi-issue problems where negotiators can strive for joint gains in integrative negotiation problems. The preconditions that cause integrativeness of the negotiation problem are very weak, as demonstrated different weights for the issues, some issues of shared interest, or concave partial utility functions are sufficient. In distributive negotiation problems – like most single-issue problems – on the other hand, the benefit of negotiations as a joint problem solving mechanism is not that significant and conflicts are more likely to arise.

Moreover the multi-issue negotiation is the more realistic case, for which negotiation is actually used, as real world problems typically consist of more than just one dimension. In multi-issue negotiations combinations of weights for issues and various shapes of the partial utility functions can result in a variety of possible preferences over the negotiation object. Studies revealed that the preferences elicited from humans differ considerably for one and the same negotiation object (e.g. Vetschera, 2006). Therefore it is, in our opinion, necessary to use many different – instead of just one or some few – and as realistic as possible – instead of just assumed – user preferences in simulation of automated negotiation as input for software agents. The combination of various preference settings for the parties in turn forms different negotiation problems, which then can be used to examine the performance of protocols and strategies in various environments.[15]

This idea also coincides with an other criticism of current simulation studies in automated negotiation research. Simulation techniques are actually used for problems too complex to be analyzed analytically by mathematical modeling. Though it is necessary to hold the amount of parameters small to avoid to complex simulation designs, current studies on automated negotiation do not fully exploit the potential complexity simulations can cope with, but unnecessarily make restricting assumptions – e.g. concerning the negotiators' preferences – that make simulation models deviate significantly from the real world problem and therefore undermine their validity and practical relevance.

Validation of the simulation model, a major part of simulation studies, necessary for the usefulness and application of the results of the proposed systems, was neglected in most of the studies. For the automated negotiation system itself – i.e. the interaction protocol and the software agents – no validation is necessary as they are computer code and can be implemented in an operative system for automated negotiation as they are implemented in the simulation system.[16] However, there are real world components that have to be validated if the simulation results are to be valid. These components are the preferences of the users and the characteristics of the negotiation object – i.e. the number of issues and the type of options. The most adequate approach to this validation problem would be the use of real preferences of humans as input for software agents that perform the automated negotiation. Otherwise one should assume preferences as close as possible to human preferences, these assumptions then have to be validated if results of the simulation are to be of any use (Law and Kelton, 1991; Pidd, 1992).

[15]Moreover other than just additive utility models have to be investigated. Additive utility models are easy to handle and could be a good approximation of the user's real preferences (Keeney and Raiffa, 1993), which justifies to use them in simulation systems, however, operative systems should support alternative models that might fit the user's real preferences better.

[16]Furthermore operational validation is not possible as operative systems are not available yet, but current research focuses on system design, and therefore the real system's and simulation's outputs cannot be compared.

Our suggestion for the negotiation problem component of automated negotiations is to study multi-issue problems, where negotiation can be used as a mechanism to effectively search for mutual benefits. For these negotiation problems many different user preferences should be assumed and validated, or even better, real preferences should be elicited from human users and used as input for the software agents. This would not only allow to test the software agents' strategies and the interaction protocols in realistically complex settings, but would also enable the evaluation of the performance of strategy-protocol combinations for various negotiation problems.

3.2.2 Alternative decision making algorithms

Existing approaches for determining the software agents' strategies ignore inherent features of the World Wide Web, as an open medium for automated negotiation, where software agents for automated negotiation can easily be programmed by human users. This openness – for new software agents of user with various preferences and for many different negotiation problems – of media for automated negotiations has to be considered when designing decision making algorithms for software agents.

Evolutionary computing needs many repetitions of negotiations over one negotiation problem in the best possible environment – consisting of opponent software agents as realistic as possible – to achieve competitive strategies for automated negotiation by means of co-evolution (Beam and Segev, 1997). However, when automated negotiations are conducted in the World Wide Web, the number of negotiations with one specific opponent might be quite low and opponents and negotiation problems may change from one negotiation to the other. Moreover, novel software agents might enter the market and new negotiation problems could appear. Therefore it is unlikely that a software agent has many trials against one opponent for one negotiation problem, which causes a major disadvantage for evolutionary computing in developing the agents' decision making algorithms for automated negotiations (Bichler, 2000). Especially strategies based on sequential threshold rules will have problems to cope with different opponents if they are optimized against only some opponent strategies and preferences (Tu et al., 2000).

While learning software agents are designed to cope with opponents with various preferences in learning them, the possibility of novel software agents also causes problems for them. The model of the opponent, learning agents hold in mind to learn the parameters of this model from the course of negotiation with the opponent, will probably be inadequate to model the diversity of possible existing as well as novel opponents. Furthermore the learning capacities are severely limited by the small number of parameters learning agents can effectively process in automated negotiations.

Also time-based strategies for automated negotiation have their drawbacks, though the incorporation of time in strategies for a basically dynamic processes like negotiation is in general appreciable. In traditional negotiations time is likely to have a major impact on the process and outcome of negotiations, as stated by Cross (1965, p.72): *'As any economist knows, time has a cost, both in money and in utility terms; it is our position that it is precisely this cost which motivates the bargaining process. If it did not matter when people agreed, it would not matter whether or not they agreed at all.'* However, it is questionable whether this statement is valid in automated negotiation that reach agreement – or otherwise terminate without an agreement – in some seconds only. The fast and low cost proceeding of automated negotiation is one of the possible sources of comparative advantage over alternative dispute resolution mechanisms, besides the assumed better outcome and novel transactions it enables. Furthermore it is arguable whether designers of software agents should base their decision making algorithms for software agents in automated negotiation on behavioral heuristics – as mentioned above, time-based strategies build on the observation of human negotiation behavior in experiments – given the aim of such agents to improve the outcome of negotiations over that reached by humans through automating the negotiation process. Interestingly simulation studies revealed that time-based and imitating strategies reached more agreements and agreements of higher utility in negotiations with later deadlines (Faratin et al., 1998). This raises the question of why to impose a deadline at all when the decision to terminate negotiations could be delegated to the software agents as well.

Finally, imitating and trade-off strategies do not face the problems mentioned so far, but are for their own insufficient to determine a complete decision making algorithm for a software agent. Imitating strategies can decide the extent of a negotiation step (concession, demand, or insistence) in terms of the difference in utility between two subsequent offers of the software agent. They can easily derive this difference from this difference of their utility between the two previous offers of the opponent. However, this reciprocation of the concession magnitude provides no guidelines for the final configuration of an offer, which is a problem when there exist some offers between which the user is indifferent. In contrast to imitating strategies trade-off strategies have problems to determine when and how much to concede or demand, as they are designed to determine the package configuration for a given utility level. Changes in the utility level of offers, however, will at some point in the negotiation be necessary to avoid getting stuck when no more offers of the same utility level are available.

Given these deficiencies of existing approaches for determining the the decision making algorithms of software agents in automated negotiation, continuous concession strategies proposed in negotiation literature seem to be a viable alternative. Continuous concession strategies follow rule-based deterministic concession algorithms that neither model their opponent nor try to learn something about their preferences or strategy, but are therefore (re)usable for various negotiation problems with different and novel opponents. Furthermore, in not making offers dependent on time, they also respect the low transaction costs associated with automated negotiation in the World Wide Web. The continuous concession strategies proposed in Chapter 4 are not novel ones, but were partly proposed in negotiation literature and not yet implemented in simulation studies on automated negotiation. These strategies are similar to the 'dummy' strategies used for comparison purposes in some of the studies reviewed above, however more sophisticated. Though these mechanisms seem very simplistic, this plainness also has its advantages. First, for acceptance of software agents it must be clear to human users how they decide and act to establish trust and thereby enhance their application in practice (Nwana et al., 1998; Maes et al., 1999). Second, not the complexity of software agents but its performance counts. In a problem closely related to negotiation – the iterated prisoner's dilemma – the tit-for-tat strategy, though being the simplest – measured in number of code lines – repeatedly outperformed opponent strategies in tournaments (Axelrod, 1980a,b).

Obviously it is important to investigate the performance of the agents' strategies when negotiating against each other, for different negotiation problems and in different interaction protocols, to determine which one yields the best outcomes. However, in the decision whether to use automated negotiation instead of traditional negotiation between humans the benchmark for the evaluation of software agents is the outcome of human negotiation. To automate negotiations in substituting their human users, or replace other currently used transaction mechanisms, software agents have to achieve better outcomes (Blecherman, 1999).[17] Therefore it is not only necessary to use preferences as close as possible to human preferences as input for the software agents – as advocated in the previous section –, but it is also necessary to compare the results of automated negotiations between software agents to the results of traditional negotiations between humans (or the currently used transaction mechanisms) – for the same preferences and therefore the same negotiation problems – to evaluate the performance of software compared to human agents. Even if one strategy outperforms all the others it will not be applied in automated negotiations if it fails to outperform human negotiators, unless saved transaction cost compensate for this deficiency.

3.2.3 Flexible interaction protocols

As mentioned in the description of this component of automated negotiation, the interaction protocol determines what actions can be taken by which participants in the negotiation for each of its possible states. In determining these rules of interaction the negotiation protocol not only establishes the basis for the actions of software agents, but also restricts them. Such restrictions are good where necessary, however, in our opinion they reduce the flexibility of agents and thereby maybe negatively impact their performance. Therefore, where possible, decisions should remain with the software agent, which then can decide whether to take actions according to a more restricted protocol but also is free to take other actions when this is considered beneficial. In case of possible gains enabled by more open and flexible interaction protocols the computational complexity should be no counter-argument, as simulation is an adequate technique for complex and analytically intractable problems.

As can be derived from the review the predominant protocol used is an alternating offers protocol or variants of it, where the software agents alternate in making their offers until they reach an agreement. Most often there exists an exogenously determined fixed deadline until which an agreement has to be reached or otherwise negotiations are broken off and in many studies the agents do not have the choice to break off the negotiation. This concern about fixed deadlines actually is not about the nature of deadlines itself, which might be important constitutes of the context in which a negotiation takes place. Actually automated negotiation – due to its fast proceeding – is beneficial for meeting such deadlines – a run in our simulation for example takes on average less than a second as reported in Chapter 5 – so that these deadlines will be no binding constraint in the majority of automated negotiations. Our criticism rather bases on the fact that these deadlines in current simulation studies are imposed in form of a predetermined number of possible rounds or turns for the automated negotiation. Alternating offer protocols with fixed deadlines cause no inconveniences for the agents' decision making algorithms studied in simulations so far. Actually they are beneficial to them, as fixed deadlines introduce the notion of time – in form of a certain number of turns – that is necessary for any time-based strategy to determine the offer for a given point in time or turn. Fixed deadlines also only make it possible to determine sequential threshold rules chromosomes in evolutionary computing, where otherwise there is no possibility to determine their length.

[17] Alternatively they have to compensate inferior outcomes with correspondingly lower costs.

However, as mentioned, these software strategies are not readily applicable for operative systems. Though the continuous concession strategies, proposed as a viable alternative in the previous section, function in such an alternating offers protocol without exit option, the denial of non-alternating offer sequences and the exogenous termination of negotiations could result in inferior performance of these strategies. In case of a fixed deadline in form of a maximal number of turns for negotiations opponents of continuous concession strategies can simply wait until this deadline and accept the last offer before the negotiation expires, without making any offer or concession at all. Thereby they could squeeze out the maximal possible of continuous concession strategies. Even if such exploitative strategies would not exist, fixed deadlines could inhibit agreements when there are only some more concession steps required to reach an agreement. In our opinion, the termination of negotiations should be the outcome of a decision of the software agents. Having this opportunity, they can themselves decide to break off negotiations if the progress is not satisfactory and therefore no satisfactory outcome can be expected for the combination of agents' strategies and the given negotiation problem.[18]

As mentioned continuous concession strategies can be exploited easily if the interaction protocol provides no means to circumvent this. Even if there is no fixed deadline, if agents do not have the possibility to discontinue their strategy they could be exploited by other agents that wait until the continuous concession strategy makes the offer with highest utility to them. The possibility to break off negotiations could prevent this but would also cause many opportunities for a good outcome to be forgone. A way to circumvent both, exploitation and missing agreements due to negotiation abort, is the stipulation of possibilities in the interaction protocol, that enable a software agent to interrupt its strategy.[19] Such an option can easily be implemented in an interaction protocol by allowing an agent to send a reject message instead of an offer when it is its turn. This modification of the interaction protocol not necessarily results in a non-alternating offer sequence, but the decision whether to alternate in the offering process or not is up to the software agents. This possibility to reject offers and thereby interrupt the continuous concession strategy until the opponent catches up can avoid exploitation, however, the possibility to reject offers, is not only beneficial for agents following a continuous concession strategy when negotiating with exploitative agents, but also when such software agents negotiate with each other. As mentioned above human users can have various preferences over one and the same negotiation object. Consider the case where one negotiator assigns different utility values to all possible agreements, while the other one is indifferent between many of them. In a protocol without the possibility to reject offers a continuous concession strategy could cause disadvantages to the later negotiator.

[18]Model validation considerations contribute a further argument in favor of termination by the agents – in real world there also exist no (absolutely) fixed deadlines. However, to avoid infinite negotiations, the negotiation protocol could overrule the agents decisions if they fail to make progress and still not terminate the negotiation. Here one rule adopted from the Zeuthen-Nash bargaining game could be, that the protocol terminates negotiations if the two agents both reject the last offer of the opponent – see Chapter 4.

[19]For example the fair concession making approach proposed by Bartos (1977) in his 'simple model of negotiation' could be applied for this purpose. Basing on the egalitarian norm of reciprocity Bartos (1977) proposes that if the opponent makes an unfairly small concession, an agent should stop to make further concessions and wait until the opponent catches up. A concession is unfairly small if the reduction of utility between the last two offers of the opponent is smaller than the reduction of utility between the last two offers of the focal software agent.

Chapter 4

Model Development and Conceptual Model

This chapter documents the model development process and the resulting conceptual model for the simulation of automated negotiation, covering the steps outlined in Chapter 2. As mentioned in Chapter 3 many studies investigating automated negotiation make use of simulation techniques, however they do not document the model development process (e.g. validation procedures, etc.) though this is an important prerequisite for the transparency and the traceability of a study. We therefore follow the steps outlined in Chapter 2, making and justifying necessary decisions, together with a detailed description of the resulting model.[1]

The conceptual model resulting from the development process aims at addressing the drawbacks of existing simulation studies on automated negotiation, summarized in the conclusions of Chapter 3 which systematically reviewed these studies. These concerns were organized along the components of automated negotiation – (i) the negotiation problem, (ii) the interaction protocol, and (iii) the software agents' decision making algorithms – and generally dealt with the avoidance of unnecessary assumptions and the exploitation of the simulation technique's potential to cope with complex interdependencies. For the negotiation problem we argued that more realistic preferences should be assumed and validated, or even elicited from users, so that the negotiation problems used in simulations better represent the variety and complexity of the environment in which actual automated negotiations will take place. Immanent decisions in negotiations, like rejecting offers or breaking off negotiations, should rather be made be the software agents than being imposed by an overly restrictive negotiation protocol – like the alternating offers protocol (with fixed deadline), which was used in the majority of simulation studies so far. Concerning the methods applied for determining the decision making of software agents, we argue that generic concession strategies are more appropriate than evolutionary computing-, learning-, or time-based approaches for an inherently open medium as the the Internet, over which automated negotiations take place in some seconds only, for various negotiation problems, and with an unknown population of opponent agents.

[1]Considerations concerning experimentation are not discussed in this chapter but in Chapter 5, where the experimental design for the computer experiments, as well as the dependent and independent variables are determined.

Applying simulation techniques for the design of systems for automated negotiation (system design) implies performing computer experiments with various system configurations to evaluate their outcomes and accordingly chose an appropriate one. The objects of an automated negotiation system, relevant for system analysis and modeling, mainly are the components of automated negotiation – discussed in detail in Chapter 3 – i.e. the negotiation problem, the interaction protocol, and the software agents. The interaction protocol and the software agents constitute the entities in an automated negotiation systems. They can be described in more detail through their attributes e.g. whether the protocol allows `reject` or `quit` messages or which offer generation strategy and concession strategy a software agent follows. These entities represent the static structure of the system and thereby form the 'automated negotiation system'.[2] Besides these 'permanent' entities there also exist temporary entities in automated negotiation systems, which are the messages exchanged by the software agents, that only exist in the system as long as it takes to generate, transmit, and evaluate them – or as long as they are stored by the software agents in case they keep a track of the negotiation history. These messages and their succession have to be in accordance with the rules imposed by the interaction protocol and change the state of the system. For example an `offer` message sent by one software agent implies a change of the 'last offer sent'-attribute of the focal software agent and a change of the 'current opponent offer' attribute of its opponent, or an `agree` message of one software agent to an offer of the opponent causes the negotiation protocol to terminate the negotiation and save its outcome. Consequently the flow of these temporary message entities represents the dynamic behavior of the system. For their operations, the software agents need information concerning the negotiation object – which is common to the software agents – and their users' preferences over this negotiation object – which is private information. These components form – as discussed in Section 3.1.1 – the negotiation problem, as the input to the automated negotiation system. Furthermore, as already mentioned in the example above, some types of messages like the acceptance of an offer or messages indicating that the software agent intends to abort negotiations lead to the satisfaction of termination criteria so that the interaction protocol terminates the automated negotiation, in this case the system terminates the specific negotiation which produces some kind of output.

We therefore consider a system for automated negotiation to consist of one software agent for each user and an interaction protocol (as illustrated in Figure 3.2 in Chapter 3). The inputs to this automated negotiation system are the negotiation object and the users' preferences – as private information to the user's software agent – over this negotiation object, which together form the negotiation problem. A conceptual model for simulation of automated negotiation has to represent these entities accordingly. The following Section 4.1 discusses the input component i.e. the negotiation object and the preference functions, provides information on data acquisition, and presents descriptive statistics on this data. Section 4.2 afterwards discusses the conceptual model's entities that represent the automated negotiation system, i.e. the interaction protocol and the software agents.

[2]The system configuration – the static structure of an automated negotiation system – actually only needs to be static for the time it takes to conduct the specific negotiation at hand. For this time, i.e. the duration of the focal negotiation, the interaction protocol and the software agents can be conceived as permanent entities of the system for automated negotiation. Clearly if the automated negotiation system is an open system software agents can freely join or leave the system for different negotiation instances and alternative interaction protocols can be chosen.

4.1 Input

Basing on the review of existing simulation studies on automated negotiation we criticized in Chapter 3, that these studies often fall short in representing the potential complexity of real negotiation problems by assuming simple preferences and in most cases considering only one negotiation problem in their studies, for which different software agent strategies are tested and compared. Actually only one of the reviewed studies (Bosse and Jonker, 2005) elicited preferences from humans as input for their software agents. If preferences, that form the input to the simulation of automated negotiation, are assumed rather than elicited then they should be validated as any input to a simulation model, which however is often neglected in simulation studies on automated negotiation. So either way there is a need to collect data on the users' preferences to use them as an input or for validation of assumed preferences.

A further concern was that the true problem solving potential of negotiations lies with multi-issue problems where joint gains could be achieved through cooperative integration of interests, however, many studies focused on single-issue negotiations (most often the price of an otherwise specified deal), which normally are more competitive. Additionally multi-issue negotiation objects seem to be the more realistic use case, as they are more frequent in real negotiation situations. For such multi-issue objects integrative negotiation problems can emerge easily from the users' utility functions, as the prerequisites for an integrative negotiation problem are not very demanding.[3] The problem of higher complexity should not be an argument against the consideration of many different negotiation problems, as the simulation technique evidenced to be able to cope with such complexity.

Addressing these concerns, we used the preferences elicited from subjects in negotiation experiments as input for the software agents in our study. The multiple-issue negotiation object for which the preferences were elicited, and the utility functions resulting from this elicitation procedure, are the topic of this section.[4]

4.1.1 Negotiation object – the Itex-Cypress case

The negotiation object used in the experiments, and as input to this simulation study, was the Itex-Cypress negotiation case written by Dr. David Cray from Carleton University (Kersten and Noronha, 1999a). The case deals with a bilateral buyer-supplier negotiation about the purchase of bicycle parts. The subjects in the experiments represent either the seller of these parts 'Itex Manufacturing' or the buyer 'Cypress Cycles'. In developing the case an effort has been made to make the case 'culture neutral' in choosing appropriate names and a negotiation object with which users from any country are familiar so that no additional explanations are necessary. Furthermore the language used in the case description is simple and the case is well structured, fitting on a half page, to account for possibly low language proficiencies of the experimental subjects (Kersten and Noronha, 1999a). According to the case description 'Cypress Cycles', an

[3]We showed in Chapter 3.1.1 that e.g. different priorities for the issues, shared interests concerning options for issues, concave partial utility functions, etc. are sufficient.

[4]Furthermore the results of these experiments will act as a benchmark for the evaluation of automated negotiation systems (combinations of software agents and an interaction protocol) in Chapter 6, in which we also discuss these results of human negotiation experiments.

established manufacturer of high quality mountain bikes, plans to launch a new line of bikes and requires a new component its current suppliers cannot provide. A possible supplier of this components is 'Itex Manufacturing', which seeks to increase its share of the component market and is interested in settling a contract for the prestigious supply of 'Cypress Cycles' if this business is profitable (Kersten and Noronha, 1999a). The experiment subjects representing either Itex or Cypress are informed that their companies are interested in achieving a good deal, but that there are also alternative suppliers and buyers so that reaching an agreement is not obligatory if they cannot reach a good deal. The outside options are held vague in order to avoid further specifications as to what a good deal means in this situation (Kersten and Noronha, 1999a). The two parties negotiate about four issues of a supply contract: the price of the parts, delivery time, terms of payment, and the conditions for return of defective parts and refund. The case specifies several discrete options for each issue provided in Table 4.1.

Attribute	Possible Values
Price	\$ 3.47, \$ 3.71, \$ 3.98, \$ 4.12, or \$ 4.37
Delivery time	20, 30, 45, or 60 days
Payment terms	Payment on receipt, 30, or 60 days after delivery
Returns	Items may be returned for refund when ...
	Any part is defective (full return)
	5% are defective
	10% are defective

Table 4.1: Attributes and possible values in the Cypress-Itex case

For each issue the parties have to choose one of these discrete options to compose offers and to finally reach an agreement on such an option configuration eventually. The five possible values for the price, the four options for delivery time, and three for each of the two other attributes, result in a total of $5*4*3*3 = 180$ alternative packages the parties can chose from to individually make an offer and collectively settle the negotiation. There is a time limit of three weeks for the negotiation experiments, but either party is free to terminate the negotiation before this deadline is reached without agreement.

4.1.2 Data acquisition with `Inspire`

The experiments on the above mentioned negotiation case were conducted with the negotiation support system `Inspire` (Kersten and Noronha, 1999b,a) developed by the InterNeg research centre[5] at John Molson School of Business of the Concordia University in Montreal, Canada. `Inspire` – an abbreviation for **I**nter**N**eg **S**upport **P**rogram for **I**ntercultural **Re**search – is a web-based negotiation support system developed for educational and academic purposes, i.e. to facilitate negotiation teaching and study intercultural electronic negotiations.

`Inspire`'s support functions base on a three phase model of the negotiation process, consisting of the phases: pre-negotiation, conduct of negotiation, and post-settlement (Kersten and Noronha, 1999b,a). In the pre-negotiation phase users study the case instructions in order to gain understanding about the open issues and options available for their settlement. Afterwards the preferences of the users are elicited by `Inspire` using a hybrid conjoint analysis approach

[5]`http://www.interneg.org` – last accessed on 17.03.09

(Kersten and Noronha, 1999a). The users first indicate the relative importance of the issues in dividing 100 points over the issues. In a next step the users indicate the relative importance of each issue's options by assigning the maximum of the points, assigned to the issue in the former step, to the most preferred option, 0 points to least preferred option, and values between the issue's maximum and zero to the remaining options. The system uses this information to calculate the overall utility of several packages for holistic evaluation by the user as a controlling device. If the calculated overall utilities for these packages do not reflect the actual preferences of the user he can modify this overall utility value for the package and the system decomposes the information to partial utility values by ordinary least squares regression – consult Kersten and Noronha (1999a) for a detailed description of `Inspire`'s utility elicitation approach. The preferences indicated to the system can be revised anytime during the negotiations if preferences change or initial errors are detected.[6]

$$u(X) = \sum_{i=1}^{n} w_i u_i(x_i) \tag{4.1}$$

Although the hybrid conjoint analysis does not explicitly distinguish between issue weights and partial utility functions, it is equivalent to the additive multi-attribute utility function (4.1) proposed by Keeney and Raiffa (1993). In (4.1) $X = (x_1, \ldots, x_n)$ is the vector of proposed options x_i, for the n issues $i = 1, \ldots, n$ to be negotiated, which constitutes an offer. w_i is the weight as a measure of importance of issue i and $u_i(\cdot)$ is the partial utility function of issue i. Adding the partial utility values of the options in all issues yields the overall utility of the offer $u(X)$. While the case descriptions for both parties clearly indicate the preference direction for all issues, e.g. that the seller favors higher options for the price, no specific trade-off values – i.e. no complete scoring scheme – is provided, so that the subjects in the experiments have to establish their own priorities and trade-offs within and across issues. The partial utility functions therefore could be linear as well as non-linear (Kersten and Noronha, 1999a) and as preferences about the negotiation object were by no means enforced in the experiments there were also cases with non-monotonic partial utility functions and even few cases with monotonic partial utility functions in the wrong direction – i.e. contradicting the case description (Vetschera, 2006).

In the second phase the actual negotiation is conducted. The users can exchange free-text messages and structured offers. Each offer has to consist of options for all issues negotiated, which means that package offers are enforced by the system. The multi-attribute utility function of the user is applied for negotiation support in this phase. `Inspire` automatically calculates and provides utility values during the construction of offers, using the preferences indicated in the pre-negotiation phase, and also evaluates offers of the opponent. The history of the exchanged messages and offers is tracked during the negotiation and can be inspected by the user anytime. Moreover the utilities of all sent and received offers is graphically represented as a function of time. If the negotiators reach an agreement – and the users mutually accept this – negotiations enter its third phase; the post-settlement phase. `Inspire` then calculates Pareto-improvements to the current tentative agreement i.e. packages that offer to at least one negotiator a higher

[6]Therefore it is necessary to determine which preferences of the users to employ as input to the simulation. We decided to use the latest preferences indicated by the subjects in the experiments as this preference information probably is free of errors and therefore representing the actual preferences of the user best.

utility without lowering the utility of the other negotiator. If such dominating packages exist they are presented to the negotiators and negotiations may be continued.

All the data generated during the course of a negotiation experiment with `Inspire` is saved in the `Inspire` database. The results of the utility elicitation procedure are saved in the table `OPTRATE`, with one row per user and utility elicitation, in which the partial utility values for the 15 options are stored. A further table `OFFERS` tracks all the offers exchanged during the negotiation experiment, and two additional tables save the answers to a pre-negotiation questionnaire and a post-negotiation questionnaire. While the pre-negotiation questionnaire deals with questions about the subject's demographic data, negotiation experience, and expectation and reservation levels for the upcoming negotiation after the case description was read and before the negotiation experiment starts, the post-negotiation questionnaire asks the subjects for their satisfaction with the negotiation process and outcome, their perception of their own and their opponent's negotiation behavior, and their attitudes toward the negotiation support system used. These last two tables are due to the focus of this dissertation not considered in this study.

Though the main purposes of the experiments on the Itex-Cypress case are negotiation teaching, as well as testing and further-development of the the negotiation support system `Inspire`, the data that form a by-product of this endeavor was analyzed in a multitude of studies not only in research on negotiation support systems but also in studies on negotiation in general and electronic negotiation in particular.[7] However up to now – at least to the knowledge of the author – this data was not used as input for simulations of automated negotiations.

From October 1996 to September 2004 a total of 2,990 negotiation experiments on the Itex-Cypress case have been set up in `Inspire`. The majority of the subjects in these experiments were students, participating in fulfilling course requirements of their studies. We selected those experiments in which the subjects remained constant during the whole period of the negotiation and where each party sent at least one offer so that negotiations are actually conducted. This reduces the sample of experiments to 2,065 negotiations. The experiment IDs of these negotiations are saved in the vector `experiments` to access them during the simulation. For these relevant experiments the negotiators' (latest) preferences and the negotiation processes were queried from the `Inspire` database and saved to the tables `PREFERENCES` and `PROCESS`. We furthermore created an additional table `OUTCOME` where we stored if an agreement was reached in the experiment and in case an agreement was reached we additionally stored the utilities to the parties and whether the agreement is a Pareto-optimal solution to the negotiation problem or not. Furthermore we saved information on the integrativeness of the negotiation problem in this table as it is the only one that structures the data by the negotiation dyad (see Appendix A).

4.1.3 Preferences

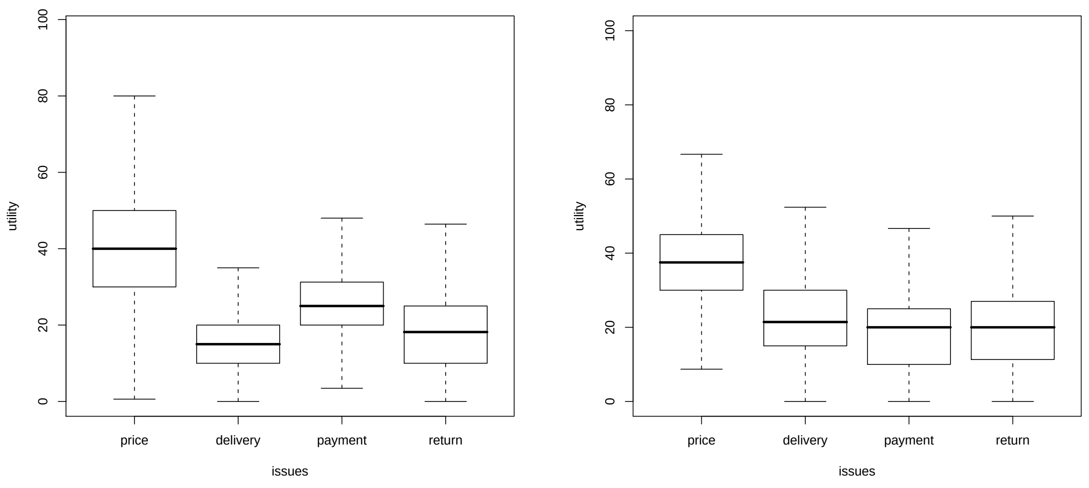
Table 4.2 and Figure 4.1 present descriptive statistics on the importance of the four issues – price, delivery, payment, and return – of the negotiation object to the seller and the buyer

[7]Consult the 'research papers' and 'publications' sections of the InterNeg web page at `http://interneg.concordia.ca/interneg/research/papers/` – last accessed on 17.03.09 – for a sample of these studies

party according to the preferences indicated in the above mentioned first step of the elicitation procedure – and eventually adapted to the holistic evaluation in the procedure's third step if necessary. For both the seller and the buyer party price was the issue of highest importance, achieving nearly 40% of the 100 points to be divided over the issues. However, due to different importance of other issues some integrative potential in the negotiation problems can be assumed. For the seller parties price was followed by the issues payment – as the issue of second highest importance receiving about quarter of the points – return, and delivery, where the two later achieved only lower importance of about 18% and 16%, respectively. For the buyers delivery, the issue of lowest importance to the seller, was in place of second highest importance, followed by return and payment. Buyers considered these remaining three issues about equally important assigning around one fifth of the overall points to each of them. Comparing the overall ordering some potentials for trading off issues for mutual benefit can be identified. Though the price could be a rather competitive issue as it is the one of highest importance to both parties, payment is of second highest importance to the sellers while only of lowest importance to the buyers and for delivery the situation is opposite.

	seller				buyer			
	price	delivery	payment	return	price	delivery	payment	return
min	0.60	0.00	0.97	0.00	0.00	0.00	0.00	0.00
1^{st} Q.	30.00	10.00	20.00	10.00	30.00	15.00	10.00	11.32
median	40.00	15.00	25.00	18.18	37.50	21.43	20.00	20.00
3^{rd} Q.	50.00	20.00	31.25	25.00	45.00	30.00	25.00	27.00
max	97.09	70.00	79.37	96.04	97.00	91.00	100.00	80.00
⌀	38.98	16.40	26.06	18.56	37.72	22.62	18.91	20.76
±	13.49	8.14	11.29	9.68	13.25	9.50	9.79	10.48

Table 4.2: Parties' importance of issues



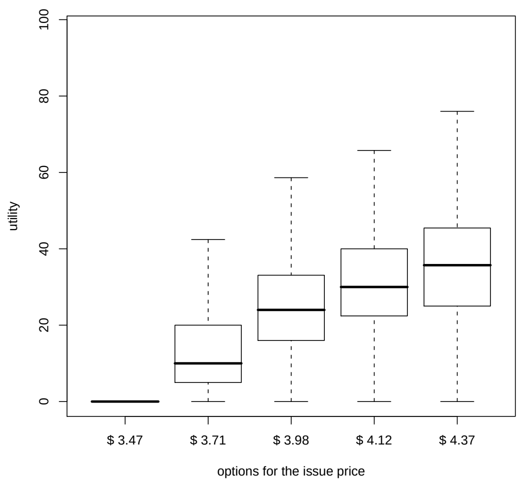
(a) seller (b) buyer

Figure 4.1: Importance of the issues

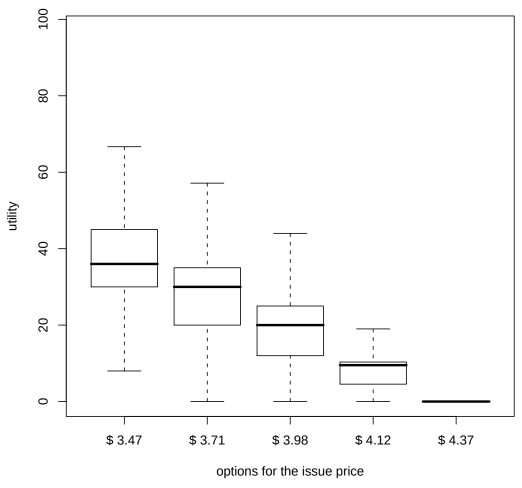
The following four tables and figures, in the same way as for the importance of issues, present descriptive statistics for the importance of the different available options for settling the four issues. Preferences of the subjects in the experiments again are derived from the points they distributed over the options of the issues in the second step of `Inspire`'s preference elicitation procedure – again eventually adapted to the holistic evaluation in the procedure's third step.

	seller					buyer				
	\$ 3.47	\$ 3.71	\$ 3.98	\$ 4.12	\$ 4.37	\$ 3.47	\$ 3.71	\$ 3.98	\$ 4.12	\$ 4.37
min	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1^{st} Q.	0.00	5.00	16.00	22.43	25.00	30.00	20.00	12.00	4.55	0.00
median	0.00	10.00	24.00	30.00	35.71	36.00	30.00	20.00	9.52	0.00
3^{rd} Q.	0.00	20.00	33.08	40.00	45.45	45.00	35.00	25.00	10.34	0.00
max	40.91	67.00	97.09	81.74	82.61	97.00	87.00	83.00	60.00	33.33
$\oslash$	0.74	13.71	25.02	31.14	34.98	36.12	28.01	19.78	9.19	0.21
$\pm$	4.01	11.21	13.59	14.73	17.16	14.91	13.49	11.48	7.89	1.97

Table 4.3: Parties' preferences over the options of the issue price



(a) seller

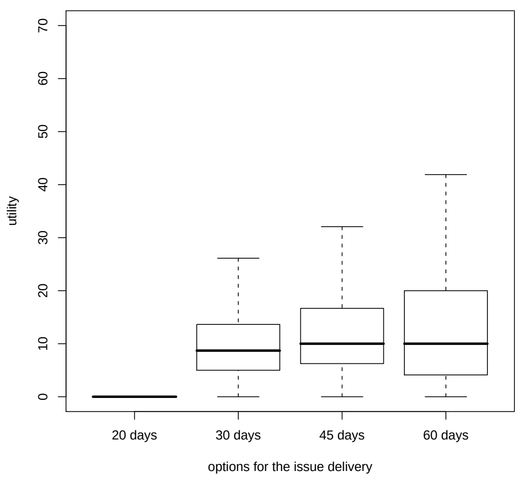


(b) buyer

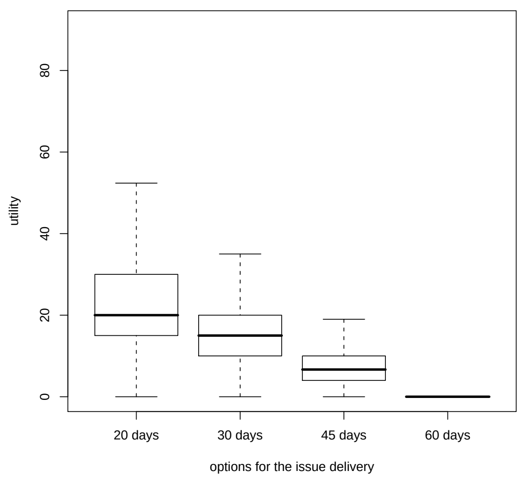
Figure 4.2: Preferences for the issue price

	seller				buyer			
	20 days	30 days	45 days	60 days	20 days	30 days	45 days	60 days
min	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1^{st} Q.	0.00	5.00	6.25	4.11	15.00	10.00	4.00	0.00
median	0.00	8.70	10.00	10.00	20.00	15.00	6.67	0.00
3^{rd} Q.	0.00	13.64	16.67	20.00	30.00	20.00	10.00	0.00
max	39.16	70.00	69.00	70.00	91.00	85.00	80.00	21.43
$\oslash$	2.20	9.59	11.88	12.39	21.86	15.79	7.78	0.22
$\pm$	5.37	7.08	7.77	9.89	10.16	9.02	6.35	1.68

Table 4.4: Parties' preferences over the options of the issue delivery



(a) seller

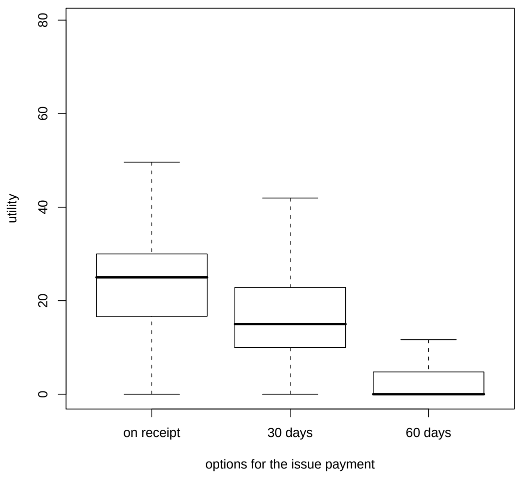


(b) buyer

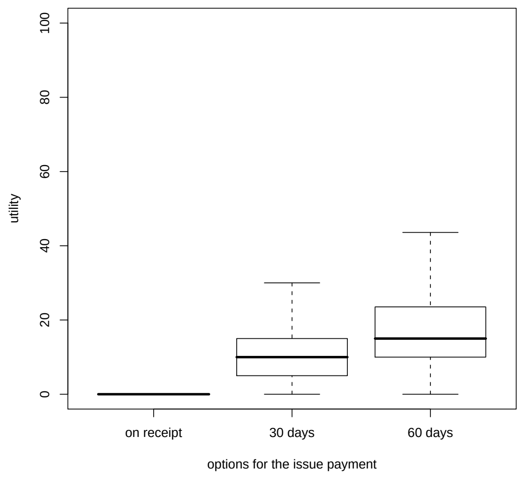
Figure 4.3: Preferences for the issue delivery

	seller			buyer		
	on receipt	30 days	60 days	on receipt	30 days	60 days
min	0.00	0.00	0.00	0.00	0.00	0.00
1^{st} Q.	16.67	10.00	0.00	0.00	5.00	10.00
median	25.00	15.00	0.00	0.00	10.00	15.00
3^{rd} Q.	30.00	22.86	4.76	0.00	15.00	23.53
max	79.37	76.19	70.87	75.00	75.00	100.00
$\oslash$	24.55	17.29	5.01	3.23	12.18	16.44
$\pm$	12.05	11.50	10.37	8.36	9.07	10.71

Table 4.5: Parties' preferences over the options of the issue payment



(a) seller

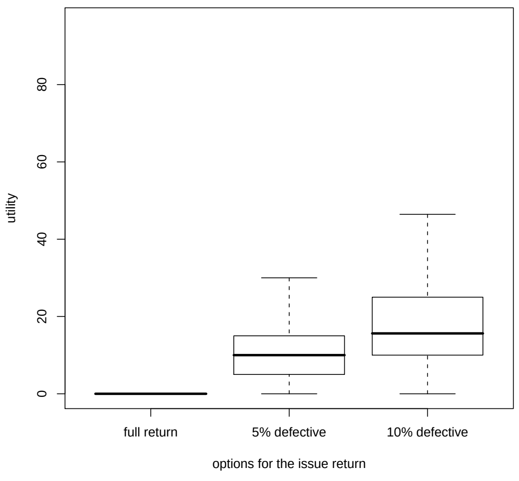


(b) buyer

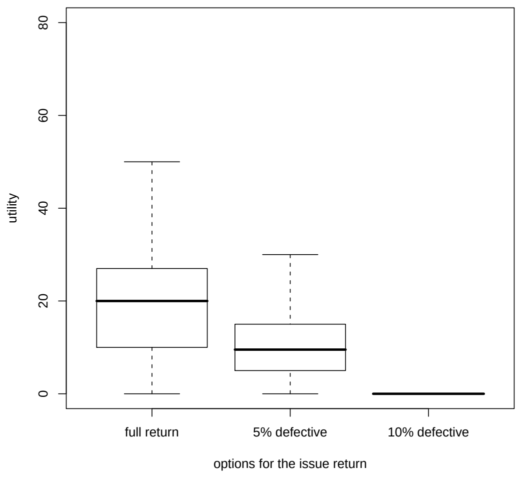
Figure 4.4: Preferences for the issue payment

	seller			buyer		
	full return	5% defective	10% defective	full return	5% defective	10% defective
min	0.00	0.00	0.00	0.00	0.00	0.00
1st Q.	0.00	5.00	10.00	10.00	5.00	0.00
median	0.00	10.00	15.62	20.00	9.52	0.00
3rd Q.	0.00	15.00	25.00	27.00	15.00	0.00
max	28.57	96.04	63.38	80.00	60.00	27.04
∅	0.88	11.73	16.90	20.05	9.83	0.38
±	3.49	8.56	10.65	11.06	7.77	2.18

Table 4.6: Parties' preferences over the options of the issue return



(a) seller



(b) buyer

Figure 4.5: Preferences for the issue return

As can be seen from Table 4.3 and Figure 4.2 both parties distributed the points for the issue price quasi linearly over the five available options for settling this issue, which even exacerbates the competitiveness of this issue, caused by its equal importance to the parties. Concerning the issue delivery Table 4.4 and Figure 4.3 indicate that similar to price the preferences for the options of delivery are quasi linearly distributed over the available options by the buyers. The sellers however evaluate the different options as about equal – with exception of the option '20 days', which is the shortest possible delivery time, – where the range of the partial utility values assigned to the options increases with option values for delivery time. The scores for the options of payment (Table 4.5 and Figure 4.4) and return (Table 4.6 and Figure 4.5) again are distributed quasi linearly from best to worst, where best and worst are oppositional for the two parties.

4.2 Automated negotiation system

As discussed in the introduction of this Chapter in automated negotiation the combination of the software agents, representing the parties in the negotiation, and the negotiation protocol, regulating the interactions between these software agents, form the automated negotiation system. The preferences over the negotiation object and the negotiation object itself are only input to this system and are not object to choices of the system user, as we assume they are exogenously determined by their interests. The configuration of the automated negotiation system i.e. the choice of software agents and interaction protocol to perform automated negotiation, however, can be determined by the system users. So while the negotiation object and the preferences of the users over this negotiation object are taken as given, users can decide about the configuration and parameterization of the system employed to handle this negotiation problem by means of automated negotiation.[8]

In the subsequent sections we propose, addressing the drawbacks of existing simulation studies mentioned in Chapter 3, alternative interaction protocols and different types of continuous concession strategies for software agents in automated negotiation. The three interaction protocols proposed in Section 4.2.1 result from different combinations of types of messages the protocols enable software agents to send. The nine software agents proposed in Section 4.2.2 embody continuous concession strategies that result from different combinations of offer generation and concession strategies. Both, the software agents and the interaction protocols, are not novel but have been proposed in negotiation literature – for software agents – and game theoretic bargaining literature – for interaction protocols – the relevant literature is reviewed and discussed with the model's components in the according sections. At least to the knowledge of the author these concepts have not been applied in simulations of automated negotiation so far. In implementing these components we therefore do not 'reinvent the wheel' but follow the suggestion to build on existing research for designing software agents and interaction protocols for automated negotiation (Kraus, 1997; Jennings et al., 2001).

4.2.1 Interaction protocol

The interaction protocol builds the basis and restriction for the interaction of the software agents in an automated negotiation system. Our concerns about the interaction protocols used in current studies on automated negotiation was above all, that in the majority of the simulation studies variants of only one specific negotiation protocol – namely the alternating offers protocol (Rubinstein, 1982), where the software agents alternate in making offers until they reach an agreement – was considered. Only in some cases the negotiations are terminated earlier – most often by the protocol if a deadline – in terms of a maximal number of turns – is reached before agents reach an agreement – and only in some of these cases the agents had the choice to break off negotiations – again this decision often based on whether or not a prespecified deadline is reached. This alternating offer protocol causes no inconveniences for most of the methods to determine agents' decision making algorithms applied so far, but is beneficial to them as fixed deadlines introduce the notion of time necessary for any time- or evolutionary computing-based strategy as discussed in the previous chapter. Though continuous concession strategies function under such an interaction protocol it causes problems of possible exploitation and inferior outcomes. As mentioned we argue that continuous concession strategies are – in contrast to strategies proposed up to now – applicable for automated negotiation in an open environment as they can easily adapt to novel problems and are independent of the opponent software agent. Therefore

[8]Note that a single party actually can only select its own agent autonomously and cannot determine the software agent choice of the other party. Furthermore the interaction protocol has to be chose jointly by the parties.

we are interested in applying protocols that are suitable for a population of software agents that follow these strategies. Such protocols should exhibit features that enable the software agents to either permanently – by breaking off the negotiation – or temporarily – by stopping to make concessions for some time – interrupt their continuous concession strategy if they think it is in their interest – i.e. to avoid exploitation or unfavorable outcomes.

4.2.1.1 Initiation

At the beginning of the negotiation in our conceptual model the protocol calls the agents in their initiation mode, which implies that it provides the software agents, registered with the system as representing their parties, with the negotiation object indicated by the parties as input – which thereby is identical for both agents –, as well as utility values for possible solutions, and let the agents create private storage variables where they store this information together with information on the negotiation process – e.g. the last own and the opponent's last messages – for later reuse during the negotiation. After this initiation of the software agents the negotiation protocol randomly chooses one agent – either agent is chosen with equal probability – and sends to him a `call for offer`.

4.2.1.2 Messages

The software agent that receives the `call for offer` from the interaction protocol is the first to make an opening offer and the one sending messages at odd turns during the following negotiation process, while the other software agent sends his opening offer in the second turn and thereafter sends messages in the even turns of the negotiation process. As the messages need not be exclusively offers we term this protocol an alternating turn protocol, in which the agent alternate in taking their turns. The possibility to send messages other than offers addresses the above mentioned drawbacks of the alternating offers protocol, which is used in simulation of automated negotiation. In their turn software agents can send one out of a set of messages determined by the protocol; either `offer`, `reject`, `agree`, or `quit` – described in detail subsequently. While a message of the type `offer` proposes one of the possible settlement of the negotiation object as agreement, which the opponent can accept or not, the other three message types are necessary for controlling the negotiation process.

4.2.1.2.1 `offer` An `offer` message in our conceptual model constitutes a proposal for settling the negotiation. In human negotiations there are many different kinds of messages besides offers, like threats, provision of or request for information, etc. all aiming at influencing the final outcome of the negotiation. However, it is argued that, though other messages are clearly of importance, offers are the most important type of messages in negotiations as they promote proposals for the settlement (Tutzauer, 1992). Many fields studying bargaining and negotiation, like game theory or management science, exclusively focus on offers and model negotiation as a process of exchanging offers and counteroffers. Therefore, and as research on argumentation by software agents in automated negotiation is still in an early stage of development, we focus on `offer` messages. To be a proposal for settling the negotiation an `offer` has to provide options for all issues of the negotiation object, i.e. it has to be a full package offer. The interaction protocol not only demands full package offers due to this, there are other protocols that operate in simulation of automated negotiation that work on an issue-by-issue basis or allow partial package offers, but negotiation literature emphasis the opportunities to reach mutually beneficial agreements through package offers, which allow to trade off issues of lower importance against issues of higher importance in a logrolling procedure (Froman and Cohen, 1970; Mumpower, 1991; Milter et al., 1996).

4.2.1.2.2 `reject` If a software agent sends a `reject` message this means that in this turn it does not propose a new offer, or make any other changes to the current state of the negotiation, but insist on its last offer. This message was proposed as a move in non-cooperative bargaining games (Harsanyi, 1956) and as a strategy to avoid unfairly small concessions and exploitation by the opponent (Bartos, 1977). Sending a `reject` message enables the software agent to discontinue the offer generation strategy it normally follows, according to which they would make some predetermined `offer`, for some reasons without necessarily terminating the negotiation.[9] The situations in which such an interruption might be beneficial to the agents are discussed in Section 4.2.2, which deals with the software agents' decision making algorithms.

4.2.1.2.3 `agree` Several bargaining games in non-cooperative game theory do not require the explicit acceptance of an offer, but terminate the negotiation if the offers of the parties are compatible (Nash, 1953; Harsanyi, 1956), i.e. if for both players the demanded utility of their offer is lower or equal to their utility of the offer proposed by the opponent. If the offers do not coincide division rules can be used to divide the surplus or chose one of the compatible offers as agreement. For this purpose equal division of the surplus or some kind of arbitration, where either offer is chosen with equal probability, are usually applied. The problem of choosing one out of two compatible offers by some arbitration rule mainly arises from the fact that in the above mentioned bargaining games players have to make offers simultaneously. An alternating turn protocol, however, is sequential by nature and the problem of deciding between compatible offers is circumvented by the acceptance criterion discussed in Section 4.2.2. A software agent in the situation to send a message will first compare the opponent's last offer to the own offer to be sent next, if the opponent's offer affords higher or equal utility the software agent accepts it rather than proposing its next offer. So the software agents in our sequential interaction protocol detect and utilize compatible offers themselves rather than relying on the interaction protocol to do so. On the other hand splitting procedures are only applicable if the 'zone of possible agreements' is an area rather than a set of points, furthermore splitting does not definitely specify the actual outcome of the negotiation as more than one of these possible settlements may afford the same utility to the parties (Osborne and Rubinstein, 1990). Therefore splitting procedures – without additional refinements – are actually only applicable in a single-issue negotiation with continuous options for this single issue. As this is a very demanding prerequisite for the negotiation object – which is not met for many negotiation objects including the one used in this study – we demand an explicit `agree` message. Sending an `agree` message means that the agent accepts the last offer of the opponent as agreement. This clearly determines what the negotiation was settled for, as the last offer of the opponent is required to be a full package offer with options for all issues

[9]However, termination of the negotiation could be the final result of sending a `reject` message if the opponent does the same as discussed in the section on termination criteria.

of the negotiation object – as discussed above.

4.2.1.2.4 `quit` Like in negotiations between humans, which do not need to end with an agreement (Beam and Segev, 1997) an interaction protocol can allow the software agents to send a `quit` message. Sending this message fulfills one of the termination criteria of the interaction protocol and negotiations will end without agreement. Therefore `quit` messages can be used by the software agents to break off negotiations if they decide this is in their interest in the given context e.g. to avoid exploitation or to stop a negotiation which will not lead to a favorable outcome. For software agents following continuous concession strategies the `quit` message is a means to permanently interrupt the offer generation strategy the software agent normally would follow, in not only denying to propose the next offer – as in case of the `reject` message – but in aborting to negotiate at all.

4.2.1.3 Termination criteria

The interaction protocol terminates the negotiation either if (i) a software agent sends an `agree` message, thereby accepting the last offer of the opponent, or (ii) if a software agent sends a `quit` message, thereby breaking off the negotiation, or finally (iii) if two subsequent messages of the two software agents were `reject` messages. This last termination criterion is applied to avoid infinite negotiations without progress towards an outcome, may this outcome be an agreement or break off of negotiations. If a software agent sends a `reject` message it does not change its internal state, which means that the same message will also be sent in its next turn unless a message of the opponent causes state changes and therefore, due a changed situation, the message of the software agent could be different from that of the previous round. If both agents send `reject` messages subsequently there will be no state changes that lead to either agreement in or break off of the negotiation anymore but only an infinite number of alternating `reject` messages and the protocol terminates negotiations if this would occur. This last termination criterion is equivalent to the mechanism in the Zeuthen-Nash bargaining game (Harsanyi, 1956), where the negotiation ends without agreement if both players reject to make a concession, with the difference that in this game proposals are made simultaneously rather than sequentially and under complete information about the opponents preferences. Note that repeating the same offer would have the same effect as sending a `reject` message, namely no changes in the state of the negotiation. For simplicity the subsequently proposed offer generation strategies are therefore designed in a way that they do not repeat offers but send `reject` messages to interrupt concession making. In principle, however, this repetition of offers could easily be detected by the interaction protocol and treated the same as if a `reject` message was sent, or – the other way around – software agents could be allowed to repeat the previous offer rather than to propose a new offer and the interaction protocol terminates negotiations if the two software agents subsequently repeated their last offers.

4.2.1.4 Interaction protocols and negotiation processes

The combination of enabled messages leads to the three protocols provided in Table 4.7. As can be derived from the above discussion of the messages, `offer` and `agree` messages are a necessary component for any interaction protocol in our conceptual model for simulation of automated negotiation. However, the others are optional and used in different combinations resulting in the three protocols considered in our study. `protocol 1` allows only `offer` and `agree` messages, `protocol 2` additionally enables the software agents to send `quit` messages and `protocol 3` allows besides sending and accepting offers to `reject` them and to thereby elicit a new offer from the opponent. There is a fourth possible combination i.e. an interaction protocol that enables software agents to send – besides the mandatory `offer` and `agree` messages – both `reject` as well as `quit` messages. As can be derived from Figure 4.9, which represents the generic structure and decision making algorithm of the software agents of our conceptual model, software agents will always chose `reject` to keep on negotiating and eventually still reach an agreement rather than instantly breaking off negotiations by sending a `quit` message. Here the `reject` message provides a sufficient means of preventing exploitation or unfavorable agreements, making it not necessary to break off negotiations by a `quit` message for the same purpose, but keep the opportunity of reaching an agreement later in the negotiation, which would be forgone when sending a `quit` message as this terminates the negotiation. Therefore if software agents have the possibility to reject they will always do so rather than breaking off the negotiation, and the negotiation processes and outcomes with an interaction protocol that allows both `reject` and `quit` messages coincide with those achieved in `protocol 3` that only allows `reject` messages. We therefore omit this redundant protocol.

	`offer`	`agree`	`quit`	`reject`
`protocol 1`	x	x		
`protocol 2`	x	x	x	
`protocol 3`	x	x		x

Table 4.7: Protocols resulting from the combination of allowed messages

The combination of these three protocols with the software agents' generic structure presented in Figure 4.9 leads to the negotiation processes depicted in Figures 4.6 to 4.8. `protocol 1` is implemented as a benchmark as it coincides with the alternating offer protocol. In `protocol 1` neither `reject` nor `quit` messages are possible so software agents negotiating in `protocol 1` are able to send only either `offer` or `agree` messages in their turn. As `agree` messages lead to the termination of the negotiation the resulting negotiation process from the application of `protocol 1` is an alternation of offers which always results in an agreement – see Figure 4.6.[10] `protocol 2` in addition to `offer` and `agree` messages allows to send `quit` messages, which like `agree` messages lead to a termination of the negotiation, however, in contrast to an `agree` message the outcome of the negotiation in case a `quit` message is sent is no agreement. As either message, `agree` as well as `quit`, terminates the negotiation and the only alternative to this is to send `offer` messages, `protocol 2` just like `protocol 1` leads to alternating offer negotiation processes, however, the outcome of these processes need not to be agreements.

[10]Though this is a very strong restriction of the interaction between software agents imposed by the protocol, it might be useful in case that reaching an agreement is obligatory. On the other hand, if this is not the case the reached agreements might be inferior to those reached when exploitation can be circumvented by `reject` messages like in `protocol 3`, or negotiations can be broken off by `quit` messages to circumvent unfavorable outcomes like in `protocol 2`. So there are pros and cons of such a protocol and maybe trade-offs between outcome dimensions in negotiations – e.g. the probability of reaching an agreement versus the quality of agreements reached – which have to be evaluated.

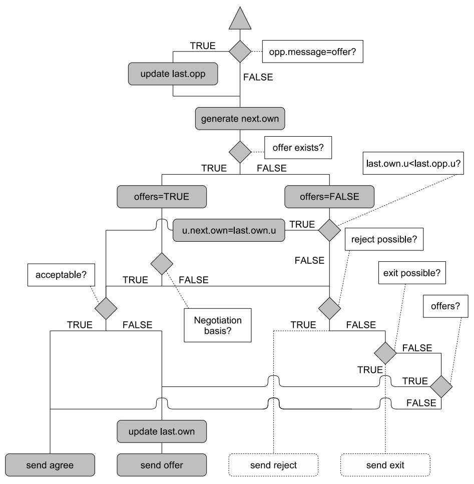


Figure 4.6: Flowchart in `protocol 1`

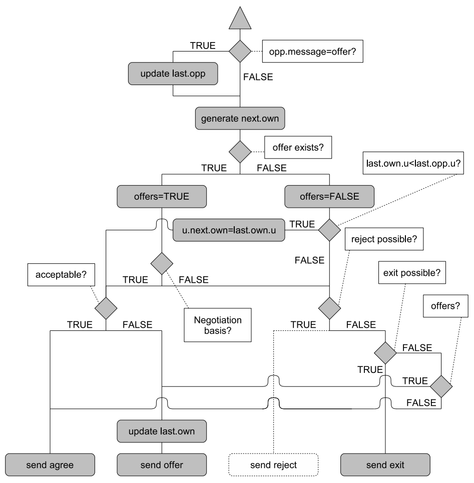


Figure 4.7: Flowchart in `protocol 2`

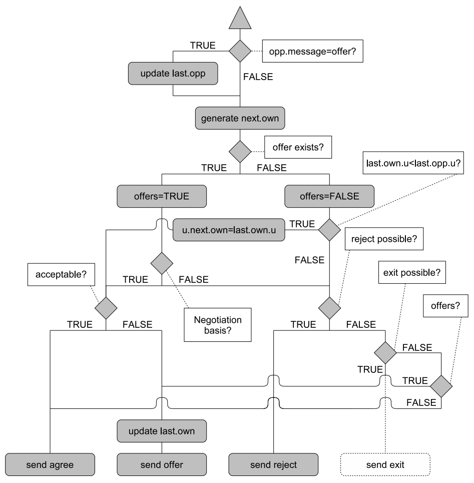


Figure 4.8: Flowchart in `protocol 3`

In `protocol 3`, though the software agents alternate in taking their turns and sending their messages, they can also send `reject` messages instead of `offers`. These `reject` messages not necessarily terminate the negotiation – they only terminate the negotiation if two subsequent messages of the two negotiation parties are of the type `reject`. So `protocol 3` may but need not lead to negotiation processes with non-alternating offers – which is the case when offer sequences are interrupted by single rejections of one party –, which may but need not lead to agreements, as the negotiation could be terminated without agreement if two subsequent `reject` messages are sent by the agents so that the protocol terminates the negotiation due to lack of progress.

4.2.2 Software agents

In the review in Chapter 3 we stated that the strategies of most software agents, used in simulation studies of automated negotiation so far, base on evolutionary computing, learning mechanisms, or time-based concession functions. We further argued that these software agents are not readily applicable to actual automated negotiation – i.e. for an implementation in operative systems – for various reasons. First time-based concession functions are not suitable due to the fast proceeding and therefore time insensitivity of automated negotiation, and the aim to improve outcomes over those reached by the human negotiation behavior these strategies imitate. Though the impatience of the negotiator or the costs associated with the mechanism of negotiation are good arguments for the reduction of the utility level demanded over time – which is the way time-based strategies are actually modeled – the validity of these argumentation is undermined if the procedure only takes several seconds at most like in automated negotiation.

Moreover, the possible variety and complexity of negotiation problems and opponent strategies in automated negotiation causes problems for software agent strategies basing on evolutionary computing or learning mechanisms. In the World Wide Web software agents for automated negotiation can easily be programmed by or for human users. This openness of media for automated negotiations – for new software agents of user with various preferences and for many different negotiation problems – has to be considered when designing decision making algorithms for software agents. We argue that software agent strategies basing on evolutionary computing or learning algorithms are not flexible and generic enough to cope with this variety of possible (novel) opponents, negotiation objects, and therefore negotiation problems – determined by the various preferences over various objects. Evolutionary computing-based strategies – especially those implementing sequential threshold rules – for software agents might not have the chance to have the large number of interactions with the same opponent and for the same negotiation problem these software agents need to reach good agreements by means of co-evolution (Beam and Segev, 1997; Tu et al., 2000), and models of the opponent held by learning strategies might be inadequate for the variety of (novel) opponent strategies, especially given the limited learning capacities and small number of parameters such strategies can effectively process.

Due to these concerns about existing approaches to determine the software agents' decision making algorithms we focus on the class of continuous concession strategies. These rule-based and rather deterministic algorithms neither model their opponent nor try to learn something about the opponent's preferences or strategy, but are therefore (re)usable for various negotiation problems with different opponents. Furthermore, in not making offers dependent on time, but only on the negotiation object, the preferences of their users over this negotiation object, and the opponent's behavior they also respect the low transaction costs and therefore invariance of time associated with the fast proceeding of automated negotiation. The components of the subsequently presented software agents – i.e. their offer generation and concession strategies – are not novel but proposed in negotiation and bargaining literature and only were not implemented in simulations of automated negotiation yet – at least to the knowledge of the author. We therefore implement software agents on an established theoretical basis and test them in our simulation study.

4.2.2.1 Initiation and negotiation mode

The software agents basically consist of two modes in which they can be called by the interaction protocol, the initiation mode and the negotiation mode. When called in the initiation mode the interaction protocol has to provide the software agent with the negotiation object and utility values for all possible settlements according to the preferences of the agent's user. The software agent stores this private information in agent specific data variables for later retrieve and use during the negotiation, these data variables are also used during the negotiation to save the state of the negotiation – i.e. which offers were already proposed by the agent and its opponent. This initiation mode is just a procedural prerequisite for the operations in the negotiation mode, which is more interesting to this study and therefore described in detail in the subsequent sections. The code of the software agent followed, when called in the negotiation mode, determines the opening offer and the procedure to follow when generating subsequent offers, how to react on messages of the opponent, which offer to accept, and when and how to terminate the negotiations, etc., thereby representing the software agent's decision making algorithm or strategy.[11]

4.2.2.2 Opening offers

When the software agent is called by the interaction protocol in the negotiation mode the first time – no matter if it is the first agent to send a message, as it received the `call for offer` from the interaction protocol, or the second agent to send a message – the agent sends a message of the type `offer` i.e. its opening offer. This offer is the most preferred by the software agent according to the utilities indicated by its user. In case there are more offers which afford the same and highest level of utility – as for all other cases of indifference between offers in the subsequent offer generation of the software agents in this conceptual model – the software agent choses one out of these tied offers randomly with equal probability for all tied offers to be chosen.[12] By proposing this opening offer of highest utility the software agent does not risk anything. Nothing is lost if this offer is accepted as no concession is made with this initial offer – as it affords highest utility – which could be exploited by the opponent.

After proposing this opening offer, when a software agent is called in the negotiation mode and

[11]Scholars argue that the game theoretic notion of 'strategy' as a fixed plan of action for all possible situations during a game is more applicable to software than to human agents as their code must explicitly and unchangeably determine all actions to be performed in all possible situations (Rosenschein and Zlotkin, 1994; Binmore and Vulkan, 1999; Vulkan, 1999).

[12]This random choice of one out of the offers between which the user is indifferent, together with the random choice which agent receives the `call for offer` and therefore has to start the negotiation, constitute the only stochastic components of the otherwise deterministic negotiation procedure and causes the conceptual model to be a stochastic discrete event model as discussed in Section 4.3.

with the last message of the opponent as input, all agents follow the generic agent template presented in the flowchart of Figure 4.9. Though there are differences between the specific instances of software agents concerning the offer generation and concession strategy the general structure is equal for all of them - in object oriented programming language one could say they are children of a generic class. They start with updating their private information according to the message they received – additionally they will update their private information after sending an `offer` message –, generate the next offer to be proposed according to their offer generation strategy, determine whether the course of the negotiation builds a basis for further negotiating according to their concession strategy, choses an action – i.e. a specific message to be sent – in accordance with the protocol and the user's preferences, and finally perform this action i.e. send the chosen message. The detailed content of each of these steps is discussed in the subsequent sections with the differences between the specific instances of software agents in these steps.

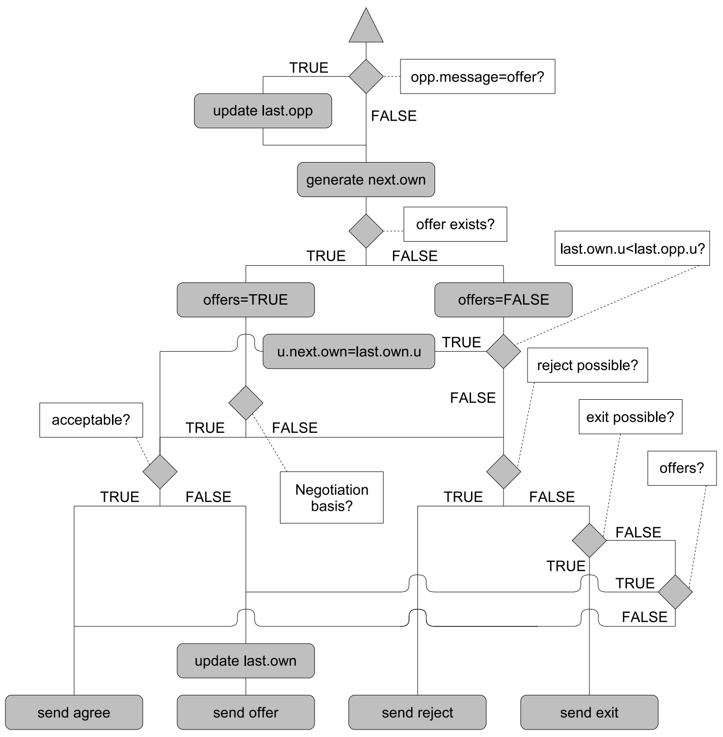


Figure 4.9: Flowchart of the agent template

4.2.2.3 Information updating

In a first step the software agent evaluates the message it received from the opponent in this turn. This message can be either of type `offer` or `reject` – if this message is possible due to the protocol – as an `agree` or `quit` message would have caused the interaction protocol to terminate the negotiation and not to call the software agent with this last message of the opponent as discussed above.[13] If the message is of type `offer` then the software agent evaluates the offer

[13]Clearly for learning agents it is a necessary and valuable information, how the negotiation ended – with a break off or with an agreement – which can be utilized together with information on the course of the focal negotiation for later negotiations, however, in their present form the software agents in this study do not learn

in determining the utility of this last offer of the opponent and updates its private information by saving this utility value in the agent's data variables. This change of information due to the receipt of an `offer` implies a state change in the simulation. Furthermore if the strategy of the software agent implies to submit an offer in the given state of the negotiation, this also constitutes a state change in the simulation, the utility of this offer to be proposed in this turn also is saved and the package is marked as offered in the private information after the message is sent and before the turn is finished by the software agent.

4.2.2.4 Offer generation strategy

Depending on the combinations of the options in the issues – i.e. the offer configuration – and the user's preferences over these options, the difference in the demanded utility between the last and penultimate offer of a negotiator can either be positive – indicating a higher demand –, the same – indicating an insistence– , or negative – indicating a concession – from the point of view of the focal negotiator. When negotiators follow a concession-based approach in negotiations, i.e. start with extreme demands and lower these in the course of the negotiation – as the agents proposed in this conceptual model do and as found to be the dominant approach in human negotiations (e.g. Pruitt, 1981) – then making concessions in continuously lowering the demand is the only viable way to reach an agreement. The software agents in our conceptual model follow this concession making approach in generating their offers. Different forms of concession making to generate offers – depending only on the negotiation object and the user's preferences over this negotiation object – are discussed in detail in the subsequent paragraphs.[14] Current research identified concessions as a common phenomenon in negotiations, as almost all negotiation processes consist of at least some concessions. Frequent concessions increase the probability to reach an agreement and reduce the duration of negotiation. Obviously the reduction of demand implied by concessions leads to agreements that provide lower utility to the conceding negotiator, however, the joint performance – i.e. Pareto optimality – of reached agreements is not negatively influenced by concession making, which underlines the importance of concessions for successful negotiations (Filzmoser and Vetschera, 2008).

When an agent has no information about the preferences of its opponent, which is the case in our study, an offer intended by one party as a concession to the other party could be perceived by the opponent as a reverse concession (higher demand) or no change in the demanded utility level (insistence) (Kersten et al., 2000). To avoid problems of wrong perceptions, in our setting of unavailable information about the opponent's preferences, the software agents act purely myopic in making their decisions. Offers are evaluated according to the only available precise information the agents have access to, namely the preferences of the software agent's user over the negotiation object. Therefore the opponent is perceived to make a concession if the utility difference – from the point of view of focal software agent – between the last and the penultimate offer of the opponent is positive i.e. the last offer affords a higher utility to the focal software agent. Similarly a bargaining step is said to be a concession by the focal software agent if the

for future negotiations but this remains an aspect to be investigated in subsequent studies.

[14]Note that some of the agents do not follow strict concession making, but if possible propose offers of same utility level, like the subsequently discussed monotonic concession strategy, however if there are no such offers of same utility level these agents also reverse to concession making, so their general strategy allows to classify them as concession strategies.

utility difference between its last and penultimate offer is negative i.e. the last offer affords lower utility to the software agent.

4.2.2.4.1 Strictly monotonic concession `SMC` Contini and Zionts (1968) propose a model in which the players in a bilateral negotiation start with an extreme offer and afterwards follow a strictly monotonic concession strategy. Unlike the monotonic concession strategy discussed next, in the strictly monotonic concession strategy the demanded utility is continuously lowered with every offer proposed, so that never two offers of the same utility are submitted. In this game as in our conceptual model (due to the acceptance criterion discussed below) agreement is reached when the demanded utility levels of the negotiators intersect. To avoid exploitation a software agent following this offer generation strategy has to reduce the demanded utility with each subsequent offer it makes by the minimal amount possible. This offer generation strategy can be implemented easily by ordering all offers not sent yet and that afford strictly lower utility than the last offer sent by decreasing utility – offers of same utility between which the agent is indifferent are ordered randomly so that they are chosen with equal probability – and the first offer on this ordered list is selected to be the next offer to propose.

4.2.2.4.2 Monotonic concession `MOC` Kelley (1966) proposes an algorithm for offer sequencing to navigate the progression of offers towards the Pareto frontier of the negotiation problem. He calls this algorithm 'systematic concession making model'. Negotiators start with an extreme opening offer, if their offer is not accepted by the opponent, they propose alternative offers of same utility. The sequence in which these offers of same utility are proposed is arbitrary as the negotiator is indifferent between offers of equal utility. Only if all offers for a given level of utility are proposed the negotiator lowers the demanded utility level to the next lower one and again proposes all offers of this utility level. In the systematic concession making model proposed by Kelley (1966) offers are accepted if the utility levels demanded by the two negotiators intersect – as in our acceptance criterion discussed below. Such an intersection of demands then is likely to present a Pareto-optimal solution as found in experimental studies, where subjects following this systematic concession making strategy succeeded in settling for Pareto-optimal solutions (Kelley, 1966). This offer generation strategy can simply be followed by software agents in ordering all offers not proposed so far decreasingly according to the utility the offers afford to the software agent – as the agent is indifferent between offers of same utility they can be ordered randomly so that there is always the same probability of choosing one out of a set of offer with same utility. The next offer to be proposed then is always the first on this ordered list. Unlike the strict monotonic concession strategy discussed above the software agent not necessarily makes a concession with each offer it proposes, but the utility of the proposed offer could be the same as the utility of the last offer if there exist offers of same utility, so that in this step no concession is made, though the whole process in general constitutes a concession making process.

4.2.2.4.3 Least-cost-issue concession `MUM` The least-cost-issue concession strategy was proposed by Mumpower (1991) and Mumpower and Rohrbaugh (1996) as a simple heuristic negotiators could follow in negotiations. The authors argue that negotiators due to uncertainties and complexities associated with negotiations and their limited cognitive capacities will apply simple 'rules of thumb' when making their decisions about offers and counter-offers. An example for such a heuristic, they propose for multi-issue negotiations, is what we call the least-cost-issue concession strategy (Mumpower and Rohrbaugh, 1996, p. 395): *'One common heuristic, for instance, is simply to offer concessions successively on that issue for which such concessions cost least in terms of the negotiator's utility.'*

Basing on the last offer proposed the negotiator compares the last offer with those where only the option in one of the issues is changed and choses the offer where this change in one issue causes the lowest costs i.e. constitutes the lowest concession. Minimal concessions are chosen to avoid exploitation as the negotiator has limited or no information about the preferences of the opponent, which are private information of the negotiator, and also not knows when the threshold of acceptability of an offer is reached by his concessions. Unlike the two previously mentioned strategies the least-cost-issue strategy no purely bases on utilities of offers but also on the content of the offer. The next offer to propose must not differ from the last one by more than one issue in terms of options. From all these offers that differ in just one issue the one where this change costs least is chosen as next offer, i.e. the one where the difference between the last offer and the next offer is zero or the smallest positive value of all the candidate offers – again in case of ties an offer is chosen randomly with equal probability for all tied offers to be chosen as next offer to be sent. The implementation of this offer generation strategy in a first step needs to determine the similarity of the not yet sent offers and the last offer in comparing in how many issues the options of the last offer and the not yet sent offers are the same and selecting those offers that differ only in one issue as candidates for the next offer to be submitted. In a second step the software agent has to chose that offer out of these candidates for which the utility difference between last and candidate is zero or the minimal positive one.

4.2.2.4.4 Lexicographic concession `LEX` This software agent starts with the highest utility offer and then bases offer generation on a lexicographical ordering of offers. Lexicographical ordering of alternatives in multi-attribute decision making problems as an alternative to additive multi-attribute utility functions was proposed and investigated by Beroggi (2003). The software agent changes the option in the issue of the lowest weight to the next lower level and evaluates whether or not this offer constitutes a concession (or at least no increase of the level of demanded utility) which is the case when the offer has lower utility than (or equal utility as) the last offer of the focal software agent. If no concession is found in changing the options of the issue of lowest importance the software agent continues to change the option in the second lowest weighted issue to find such an offer etc. To implement the lexicographic concession strategy the software agent first has to establish a lexicographic ordering of all offers in ordering the issues and within the issues the options by decreasing utility, i.e. by the issues weights and the options partial utility to the user (and randomly in case of ties). Next the software agent has to chose the subset of offers from this ordered list that afford same or lower utility compared to the last offer made and propose the first package of this resulting subset as the next offer.

4.2.2.4.5 Tit-for-tat concession `TFT` Recently Shakun (2005) proposed a slightly modified version of tit-for-tat – a strategy known for being very successful in the iterated prisoner's dilemma (Axelrod, 1980a,b), which is closely related to negotiation – for software agents in automated negotiation, which fully reciprocates concession – in terms of their own utilities –

received by the opponent's last offer. As mentioned in the review, imitating strategies can simply define the level of utility the next offer should provide, but have no means to determine the actual configuration of the next offer. Therefore Shakun's software agent not only reciprocates concessions but also informs the opponent on which of the issues it would like the opponent to make the next concession. As this is not possible with the messages allowed by the protocols in our conceptual model we decided to combine tit-for-tat concession making with a trade-off mechanism for offer generation. Trade-off mechanisms try to propose offers as similar as possible to the opponents last offer, thereby trading issues to potentially achieve mutually beneficial outcomes. Similarity of offers could be defined in many ways like Hamming distance, for a discrete set of possible solutions, or Euclidean distance, otherwise. The software agent resulting from the combination of tit-for-tat concession and trade-off offer generation operates as follows: In a first step the offers constituting the same (or higher) concession as that provided to the software agent by the opponent with his last offer are selected from the not yet sent offers. In case of ties the offer with the lowest Hamming distance to the last offer of the opponent is chosen i.e. that offer which is most similar to the opponents last offer in terms of the options for the issues. Note that in contrast to the other software agents the `TFT` concession strategy also considers the configuration of the opponents offers in making its decisions about the configuration of the next offer and the magnitude of the concession to be made. Concessions of the opponent are fully reciprocated therefore considerations about following either an active or passive concession strategy – discussed in the next section – are not applicable for the tit-for-tat agent. It always has a basis to further negotiate and reciprocates a concession made by the opponent, however it also reciprocates `reject` messages in `protocol 3`, thereby triggering termination of the negotiation by the interaction protocol.

Note that different weights can be given to the similarity of the concession to the last concession of the opponent, on the one hand, and the similarity in offer configuration to the last offer of the opponent on the other hand, in designing such tit-for-tat trade-off strategies. As can be seen from the above discussion our design first considers reciprocation of concessions – i.e. assigns highest weight on this aspect – and only in a second step considers the similarity of offers. This is justified with the concerns about avoiding exploitation or unfavorable agreements in relation with continuous concession strategies. The offer of highest similarity to the last offer of the opponent would be an offer that coincides with it – i.e. proposing the same package – which would lead to the acceptance of the first offer of the opponent according to the acceptance criterion employed in our study. Accordingly all offers more similar to those of the opponent increase the probability that these offers are accepted, which however could result in inferior outcomes for the software agent that focuses too much on similarity of offers and concedes too much. However, following the logic of trade-off offer generation proposing more similar offers could result in offer configurations acceptable for the opponent without having to give in too much in terms of utility and thereby leading to favorable agreements. So there exist many possible parameterizations for such tit-for-tat trade-off strategies (with different weights assigned to concession reciprocation and offer similarity), however, to evaluate them would exceed the scope of this study and we therefore focus on the design proposed above for the mentioned reasons and postpone a detailed analysis of the possible configurations to later studies.

If a software agents runs out of offers according to the above determined offer generation strategies it checks whether the opponent made an offer of higher or same utility compared to the last own offer sent (see Figure 4.9). If this is the case values are manipulated so that this last offer of the opponent is accepted. Thereby the software agent does not risk to break off negotiations by sending `reject` or `quit` messages but rather accepts the opponent's offer. Otherwise, if the software agents has no offers left to propose according to its offer generation strategy and the last offer is not acceptable it rejects the last offer of the opponent, breaks-off the negotiations, or finally accepts the last offer of the opponent (in this order), whatever is possible according to the restrictions of the interaction protocol.

4.2.2.5 Concession strategy

In this section we discuss the available options for determining whether or not the current course of the negotiation builds for a software agent the basis to keep negotiating in further following the above mentioned offer generation strategies, or to interrupt these strategies to avoid exploitation by the opponent or unfavorable outcomes, if this is enabled by the protocol. According to the approach for fair concession making proposed by Bartos (1977) in his 'simple model of negotiation' one possibility to avoid exploitation is to reject making further offers – and therefore concessions in the case of continuous concession strategies. Basing on the egalitarian norm of reciprocity Bartos (1977) proposes a simple non-mathematical theory of negotiation. Essential to this theory is the rule of distributive justice according to which men view as fair, rewards that are proportional to the recipients' contribution to society – i.e. equal rewards only for equal contribution. For negotiation, as a special form of social interaction, this means that negotiators should receive a payoff proportional to their feasible maximal payoff. The maximal utility is defined in this model as the utility reached by a negotiator when the other negotiator achieves an utility equivalent to his evaluation of the BATNA – as the participation constraint for negotiation –, which are for the two negotiators the extremities of the individually rational zone of the Pareto frontier. These points are also reference points in several normative axiomatic solution approaches for bargaining problems (Rosenthal, 1976; Roth, 1977; Thomson, 1981). This maximal payoff is also the opening offer proposed by Bartos – and used for our software agents as discussed above. After having these first offers on the table the midpoint between these offers achieved by splitting the difference – which is also proposed by Raiffa (1982) as an important strategy in negotiations, and a common wisdom for reaching agreements – acts as an expectation for the final outcome. After having determined the opening offer how large should the subsequent concessions be? For the concession after the opening offer Bartos argues that the first concession should be only small to avoid being exploited, although he states that other aspects could play an important role in determining the first concession like for instance personality, reputation, or institutional constraints. Later concessions should be fair if the opponent's concessions were fair, otherwise the negotiator should not concede until the opponent is up to his level of concession to avoid being exploited. The negotiator should not retract a concession made as there exists a common code in negotiations, that offers once made and therefore on the table should not be withdrawn or otherwise there will be some sanctions like for the violation of any code. Rational negotiators would expect fair concessions if the opponent reciprocates their own last concession, however, due to lack of knowledge of the opponent's payoffs we cannot measure if the opponents concession matches the own last concession. Fair in this context of concession making is operationalized as the demand that the expected final outcome – the midpoint of the two opening offers – should remain the same. Bartos finally shows that the midpoint of the initial offers in his model, for a negotiation problem where the Pareto frontier is a straight continuous line, equals the well-known and empirically supported Nash solution (Nash, 1950), which is due to its symmetry axiom considered as a fair solution.

Adopting Bartos' 'simple model of negotiation' to our conceptual model, where software agents also have no information about the payoffs of the opponent, a concession is considered unfairly small if the total reduction of utility from the starting point of negotiation – assumed to be the worst payoff for the focal software agent as described above – to the current offer of the opponent is smaller than the reduction of utility between the opening offer of the focal software agent – its most preferred offer, as argued in the model and implemented in the opening offer of the software agents – and the current offer of the focal software agent, as this would make it necessary to change the first expectations about the final agreement, which was the midpoint of the opening offers. Denoting j as the focal agent, $-j$ as its opponent and letting u_j be the utility function of software agent j, and $o_{j,t}$ the offer of agent j proposed in round t, then due to the sequential nature of our alternating turn protocol two different reference values against which to compare the opponent's concession made with its last offer $o_{-j,t-1}$ exist, the own concession determined by the last own offer $o_{j,t-2}$ or the next offer to be proposed $o_{j,t}$ as depicted in Figure 4.10.

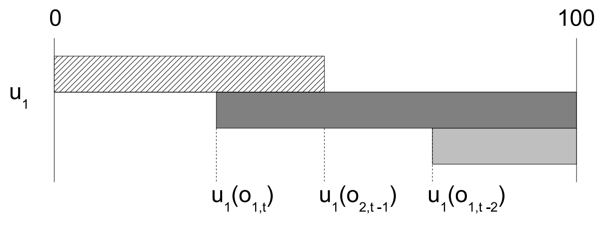


Figure 4.10: Focal negotiator's and opponent's concessions

Using the next own offer to be proposed $o_{j,t}$ according to the software agent's offer generation strategy as reference point leads to a rather passive concession strategy formulated in (4.2). With the passive concession strategy a software agent only further negotiates if it does not exceed the opponent's concession magnitude up to the last offer of the opponent with the own concession determined by their next offer to be sent. So this strategy could be considered rather pessimistic and offers are only made in a way to stay at or below the opponent's concession magnitude to avoid exploitation. Only if this is the case the software agent further negotiates in making offers according to his offer generation strategy or accepting the opponent's offers. This passive strategy can be expected to lead to better agreements for the focal negotiator, however, maybe also to fewer agreements.

$$\texttt{negobasis} = \begin{cases} \texttt{TRUE} & \texttt{if } 100 - u_j(o_{j,t}) \le u_j(o_{-j,t-1}) \\ \texttt{FALSE} & \texttt{else} \end{cases} \tag{4.2}$$

By contrast if the software agent uses the last own offer $o_{j,t-2}$ as the basis for comparing own and opponent's concession magnitudes this leads to the considerations represented in (4.3), which can be considered to be more active and optimistic. If the opponent with his last offer matched the concession magnitude the focal software agent reached with his last offer then the software agent further negotiates in making a new offer and a concession step 'ahead'. By contrast in the passive concession strategy the own concession magnitude will always be lower or at most equal to the concession magnitude of the opponent. Though this going ahead in making concessions possibly could be exploited – actually only the one concession the agent goes ahead – it could be a good means to keep negotiations running and reach agreements where passive concession strategies fail to do so.

$$\texttt{negobasis} = \begin{cases} \texttt{TRUE} & \texttt{if } 100 - u_j(o_{j,t-2}) \leq u_j(o_{-j,t-1}) \\ \texttt{FALSE} & \texttt{else} \end{cases} \tag{4.3}$$

In Figure 4.10 for example the current offer of the opponent constitutes a concession magnitude larger than the concession made by the focal agent with its last offer but smaller than the concession to be made with the next offer. Therefore agent following the active concession strategy – `act` represented in equation (4.3) – would have a negotiation basis – as the opponent's concessions are larger than the own made up to this point in time – while an agent following the passive concession strategy – `pas` represented in equation (4.2) – would have no negotiation basis as the opponent's concessions are smaller than those to be made with the next offer. Furthermore note, that if the next offer is to be proposed – here only in the case of the active agent – the software agent will accept the last offer of the opponent, due to the acceptance criterion discussed subsequently, rather than proposing its own next offer as it affords higher utility.

4.2.2.6 Action execution

If the opponent evaluates the current course of the negotiation to establish no basis for further negotiation according to the their concession strategy, they will try to interrupt the offering procedure, determined by their offer generation strategy if this is enabled by the protocol, in a predefined order. If the protocol allows to send `reject` messages (in `protocol 3`) the agent will do so and elicit a new, maybe better offer, from the opponent which could allow to continue its strategy. This is done at the risk of a termination of the negotiation if the opponent also sends a `reject` message. However, if it is not possible to reject offers and thereby interrupt the offering procedure temporarily the software agent will try to interrupt it permanently to avoid exploitation and unfavorable outcomes by sending a `quit` message and breaking off negotiations (in `protocol 2`). As can be seen from the flowchart in Figure 4.9 the agents will always first try to temporarily interrupt their offering procedure before they break off negotiations, i.e. wait for the opponent to catch up in making sufficiently large concessions to keep on negotiating rather than sacrificing the opportunity of an agreement. If the protocol neither allows to send `reject` nor `quit` messages (in `protocol 1`) the opponent cannot interrupt its offering strategy and has to send an offer or if no more offers are left, accept the last offer of the opponent. In this protocol, as discussed, an agreement is certain but maybe unfavorable for the software agent that not wants to further negotiate but cannot follow its intention due to restrictions of the protocol.

Otherwise, in case the software agent comes to the conclusion – due to the concession strategy it follows – that the current course of the negotiation builds the basis for further negotiating, it has to decide whether to accept the opponent's last offer or make the counteroffer determined according to its offer generation strategy before. The criterion for accepting an offer of the opponent is simple and equal for all agents: Software agents accept the last offer of the opponent if it affords same or higher utility compared to the next own offer to be proposed. If this is the case then the agent sends an `agree` message thereby terminating the negotiation with this agreement, otherwise it sends a message of the type `offer` proposing the generated offer. Using the notation from the previous section acceptance or offer submission is determined by (4.4).

$$\texttt{message}_{j,t} = \begin{cases} \texttt{agree}\ o_{-j,t-1}, & \texttt{if}\ u_j(o_{j,t}) \leq u_j(o_{-j,t-1}) \\ \texttt{offer}\ o_{j,t}, & \texttt{else} \end{cases} \tag{4.4}$$

Note that this acceptance criterion could have drawbacks when the opponent proposes an offer of high utility early in the negotiation process but the software agent at this point in time is not ready to accept it as the level of own demands is still too high. Normally the software agent would come back to this offer first proposed by the opponent later in his own offering sequence and then the opponent would accept the offer. As both agents follow the same acceptance rules and as the agents follow concession strategies a previous offer proposed is always of higher or equal utility to the software agent than later offers and so if a previously proposed offer is back on the negotiation table as it is proposed again by the opponent following the acceptance criterion it will be accepted. However, the negotiations might terminate earlier with an inferior agreement, a break off of negotiations by one party, or termination by the protocol due to lack of progress. In these cases the opportunity of a favorable agreement is forgone due to the acceptance criterion. One alternative to circumvent this drawback would be a fixed acceptance threshold – as implemented in most evolutionary computing-based strategies for software agents (e.g. Oliver, 1996) – for the utility of the opponent's offers, where the offer is accepted if it passes this threshold, however then there exists the problem to define this threshold, which might be not trivial for diverse negotiation problems and without information about the opponent's preferences. If the threshold is to high negotiations often would end without agreement, if it is too low potential value could be left unrealized. For example Chavez and Maes (1996), who implemented a fixed acceptance threshold in their experimental automated negotiation system 'Kasbash', report:

> Users expect their agent not to do clearly stupid things. Even though most participants knew the details of how their agents worked, they were disappointed when agents did 'dumb' things that a human would never do, such as accept an offer when a better one was available. This happened because the agents always accept the first offer which meets their asking price; however, there might be another offer which also meets the asking price but is even better. If Kasbash's agents are to serve as intelligent assistants to the user, then they will have to be made free of such 'bugs'. (Chavez and Maes, 1996, p. 86)

An alternative to this fixed threshold would be a variable threshold like the acceptance functions proposed by Gerding et al. (2003) as a 'social extension' of their evolutionary computing-based software agent strategies. These functions determine a probability of accepting the offer of the opponent which increases with the utility of the proposed offer. However, again the problem to determine the function exists and due to determining only a probability it is not certain that a favorable offer is actually accepted and the problem of our acceptance criterion therefore with such an approach also remains unsolved. Therefore we implemented our acceptance criterion, which also is commonly used in other studies, as other approaches do not solve this problem but rather create additional ones.

4.2.2.7 Resulting software agents

Four of the five proposed offer generation strategies – namely strict monotonic concession `SMC`, monotonic concession `MOC`, least-cost issue concession `MUM`, and lexicographic concession `LEX` – can be used with both concession strategies, the active – `act` represented in equation (4.3) – and the passive one – `pas` represented in equation (4.2). In the tit-for-tat trade-off strategy `TFT` reciprocation of concessions received from the opponent is already implemented such that here no choice between active and passive concession making is necessary. The combination of these distinct offer generation and concession strategies – and `TFT` – with the other decision components equal for all agents results in $4 * 2 + 1 = 9$ distinct software agents for which the general strategies were discussed in this section and for which the source codes are provided in Appendix A.

4.3 Modeling and validation

The conceptual model outlined in the previous sections is a dynamic stochastic discrete-event model. Stochastic results from the random effects in the interaction protocol and the software agents' decision making algorithms – i.e. the random determination of the software agent starting the negotiation by the interaction protocol and the random determination of the offer to be sent in case of ties by the software agents. However, as discussed in Chapter 5 and Appendix B the influence of these stochastic components on the outcome is rather low. The model is dynamic as the model's state changes over time with the exchanged messages at discrete points in time. Between these exchanges of offers the system stays unchanged so that we can speak of a discrete-event model.[15] Finally, automated negotiation can be conceived as terminating as the process ends with some outcome, which could either be an agreement or a break off of the negotiation.

In developing the conceptual model we focus on the basic and higher level interactions of software agents in automated negotiation, which deal with the interaction protocols and software agent strategies in general (Rosenschein and Zlotkin, 1994). In this sense the abstraction from – yet non-existing – operational systems for automated negotiation is that we neglect operative issues of this interaction such as ontologies for negotiation objects, preference elicitation from the users, agent communication languages, or communication security (Rosenschein and Zlotkin, 1994; Nwana et al., 1998). We assume that the software agents can communicate and understand each other – leaving these operational issues, increasingly considered in research and practice and experiencing significant progress in recent years, to software developers – and pose the question how should they interact in automated negotiations, i.e. what are appropriate interaction protocols to implement and successful software agent strategies to follow for a given negotiation

[15]Though the majority of scholars view this exchange of messages as the most important factors that change the state of the negotiation, there also exist few models that consider continuous concession and resistance forces or impatience of negotiators (e.g. Balakrishnan and Eliashberg, 1995), which would call for a system dynamic rather than a discrete-event model.

	Itex (seller)		Cypress (buyer)	
same pref.	frequency	subtotal	frequency	subtotal
6	1	6	1	6
5	–	–	–	–
4	1	4	4	16
3	1	3	5	15
2	19	38	23	46
unique	2,014	2,014	1982	1982
total		2,065		2,065

Table 4.8: Frequency of identical and unique preferences

problem (Rosenschein and Zlotkin, 1994).

As mentioned in Section 2.3 validation can be divided in data validity, conceptual, and operational validation. Concerning data validity of the input data for the simulation, which are the negotiation problems resulting from the negotiation object and the users' preferences over it, we criticized that existing studies often assumed few and simple negotiation problems when it would be better to consider various of these problems or even elicit user preferences as input for automated negotiation systems to render the simulation more realistic. Acquiring valid input data for simulations is a hard and laborious endeavor (Law and Kelton, 1991; Liebl, 1992; Pidd, 1992; Sargent, 2005), however, we are in the fortunate position to have access to the `Inspire` database on the 'Itex-Cypress' negotiation experiments consisting of data on nearly 3,000 negotiation experiments.[16] Though this database forms a good basis for our simulation project we have to admit two shortcomings of this data concerning its validity. First, the data is only about one single negotiation object – the Itex-Cypress case – and second it is only data on negotiation experiments with student subjects not on actual negotiations carried out by professionals. Regarding the first concern it has to be stated that the preferences over this negotiation object elicited from the subjects vary to a large degree as presented in Table 4.8.[17]

Though some of the utility functions of the seller and buyer parties are not unique, in all experiments they combine to unique negotiation problems. So though we have only one negotiation object the data on the experiments provides information on 2,065 distinct negotiation problems. With regard to the second concern – the use of laboratory experiments with student subjects rather than the examination of real negotiation with professionals – the standard answers of experimental negotiation researchers apply. Though one might argue that in understanding and evaluating negotiations one should focus on professional negotiators in research, many others than experienced negotiators are involved in negotiations – as mentioned in the introduction to this dissertation. Furthermore professionals are busy so that it is expensive and time-consuming to get professionals as participants for experiments, which might render many research projects problematic or even impossible to conduct. Finally substantial evidence from experiments (e.g. Bazerman et al., 1985; Neale and Northcraft, 1986; Northcraft and Neale, 1987; Bazerman and

[16]We would like to thank the InterNeg Research Group, especially Rudolf Vetschera, for providing access to this database and InterNeg and the many associated researcher for performing these experiments, which are invaluable for this research project.

[17]In case of the multiple identical preference functions indicated in Table 4.8 the subjects often chose quite apparent distributions of partial utility values over the options of an issue, like linear or nearly linear distribution of the partial values over the options, or giving all issues the same weight and only the most preferred option the full score for the issue and all other options in that issue a score of zero.

Neale, 1992) suggests that negotiation professionals do not differ in fundamental ways from novices to negotiation i.e. they acquire the same skills and make the same types of errors (Moore and Murnighan, 1999; Loewenstein and Thompson, 2006).

As mentioned in Chapter 2 conceptual validation – also called white-box validation (Pidd, 1992) or face validation (Law and Kelton, 1991) – is concerned with determining that the theories and assumptions underlying the conceptual model are correct, that the model is a good representation of the problem entity, and that the model's structure, logic, as well as mathematical and causal relationships are reasonable for the intended purpose of the model (Sargent, 2005). To achieve conceptual validity we adopt findings of other areas applicable to the simulation of automated negotiation, as argued to be more reasonable than 'reinventing the wheel' by researchers in the field (Kraus, 1997; Jennings et al., 2001). We heavily rely on work from game theory – especially concerning the determination of the interaction protocols – and negotiation theory – for determining the software agents' concession and offer generation strategies. Furthermore, the conceptual model was presented in form of a structured walk through in the Ph.D. research seminar at the faculty of business administration and economics of the University of Vienna and at the GDN 2008[18] in Coimbra in front of an expert audience in the domain of negotiation (Filzmoser, 2008). We also validated the basic assumptions of the conceptual model which deal with random moves in starting the negotiation and the choice of an offer in case of ties. Stochastic components should only be used when the underlying behavior cannot – or should not, due to the aggregation level and purpose of the model – be broken down and analyzed any further. In our model we have two such random influences. First, the choice of offers if there are ties in the preferences i.e. if the user is indifferent between two or more offers as they afford the same level of utility. In this case we randomly take one of the alternatives the user is indifferent between with same probability, which can be theoretically justified by the concept of indifference. The second stochastic component, which comes into play in our model, is the choice which software agent starts the negotiation. Clearly one could model the software agents' strategies in a way to determine endogenously which agent starts, however, in our model the opening offer is only one offer and many others follow so the influence is rather low, moreover the opening offer is standardized to the one (or one of those) that afford the maximum utility as the agents subsequently follow concession strategies. This also lowers the importance of who starts the negotiation as the opening offer (of both) is the maximal utility offer and thus it is the same no matter if the agent starts or is the one to respond first. The only influence the decision of who starts has is on the ordering of the turns i.e. who will send his messages at odd or even turns. Moreover the data on the negotiation experiments indicates that the negotiations are started (most often by the party that logs in first) to 49.06% by the seller party and to 50.94% by the buyer party which does not deviate from the equal probabilities of starting the negotiation assumed in the model.[19]

Concerning operational validation we already mentioned that it is impossible due to the non-

[18]The joint conference of the INFORMS section on Group Decision and Negotiation, the EURO Working Group on Decision and Negotiation Support, and the EURO Working Group on Decision Support Systems.

[19]A χ^2 goodness-of-fit test indicates that the observed proportions of starting the negotiations by the two parties do not differ significantly from the expected equal distribution; $\chi^2 = 0.74$, $df = 1$, $p = 0.39$. Furthermore also for the results of the simulation runs no differences in individual utility – the only outcome measure at the individual level, while all other outcome measures are at the dyad level as discussed in Chapter 5 – were found between first mover and second mover in the simulations.

existence of operative systems for automated negotiation. However, as the code and functions used to run the simulation program easily can be applied to implement a real system for automated negotiation we argue that operational validity is given. The higher level interaction will be the same and only some additional features have to be added. Actually if the parties to a real negotiation would agree to use the simulation program and input their preferences directly to the agents the simulation program would be such a real system for automated negotiation and operative validity therefore achieved.

4.4 Implementation and verification

The conceptual model described in his chapter was implemented in `R` and a `Firebird`-database for data handling. The reasons for using `R` were that is allows for flexible data manipulation, advanced functions for statistical analysis and graphical representation of data, and provides good features for database interaction, however, besides these aspects, which are all important to our study, the most immediate reason for using `R` was, as in many cases when it comes to chose a software for simulation, the author's familiarity with the language. Commented `R`-code of the simulation program and the data structure of the `Firebird`-database can be found in Appendix A.

The simulation program follows the process-interaction approach with a next-event time advance approach to handle the simulation clock. This approach seems to be the easiest way to implement the simulation program and in our opinion best reflects the structure an operative system for automated negotiation may adopt. The software agents are implemented as functions which have their own variables to store information about the negotiation object, the user's preferences, and the state of the negotiation. The interaction protocol is also implemented as a function, which calls the agent functions providing them with the last message of the opponent and getting back the message of the focal agent function. Messages are checked for termination criteria and if they pass the message is used to call the opponent agent function.[20] From this brief overview two differences to the standard process-interaction approach and next-event time advance can be derived, which are due to its application in the domain of automated negotiation rather than some scheduling system. First we model the permanent entities rather than the temporary entities as processes i.e. the software agents and not the messages exchanged. However, the software agents are best conceived as processes in an automated negotiation system as they evaluate incoming messages and change the system state accordingly by changing their own information variables, as well as generating and sending new messages. They are stopped after they sent a message and the current status – which offers were sent and received up to that point of simulation time – is saved in their information variables and reactivated with the new message of the opponent. Therefore the simulation program is some kind of degenerated process interaction approach as the processes never run parallel but one software agent is always halted during the other is executed. Furthermore also the time advance is kind of a degenerated approach compared to the next-event time approach as time is handled in turns and at each turn there occurs one

[20]An actual negotiation protocol should, besides informing the agents which messages are possible, also control that the agents follow this advice, however, for simplicity we programmed the negotiation protocol function to just indicate the possible messages and the agents to strictly follow this instruction.

event i.e. a message is sent by one agent function at all odd turns and by the other at all even turns, this allows the negotiation function simply to call the agents with the opponent messages alternately instead of keeping, updating, and inspecting a future event list – actually only a turn index is necessary which is incremented and the according software agent is called at odd and even turns. The simulation program for the automated negotiation system is extended by a main loop which handles the database interaction and agent registration. This part of the simulation program queries the users preferences from the input database, calls the single simulation runs with specified software agents, preferences, and interaction protocols, and saves the outcome of a simulation run to the output database. Through iterating the negotiation experiments to be simulated as well as the software agents and protocols used for the simulation the main loop establishes the full-factorial tournament experiment design discussed in Chapter 5.

To verify that the simulation program works correctly we used a modular programming approach, so that the agents consist of the same modules, same variable names, etc. We not only inspected whether final results are meaningful i.e. the outcome of test runs of the simulation, but for all agents in all protocols and for several randomly selected negotiation problems we collected intermediate results on the random numbers generated, the updating of variables, and the messages sent to ensure that the program works as intended.

Chapter 5

Experimentation

This chapter discusses the experimental design of the simulation with the conceptual model provided in Chapter 4. We will also discuss the dependent and independent variables we use for the evaluation of these experiments in Chapter 6, together with the choice of statistical methods for these analyses.

5.1 Experimental design

To configure a system for automated negotiation for a given negotiation problem in our simulation we have to determine three components, namely the interaction protocol and the software agents representing the two parties in the automated negotiation. These components are the factors in our experimental design and for each factor different factor levels are available. For the interaction protocol there exist three alternatives `protocol 1` – which enables to send `offer` and `agree` messages –, `protocol 2` – which additionally allows to send `quit` messages –, and `protocol 3` – which enables the temporary interruption of the offering sequence by sending `reject` messages in addition to messages for proposing and accepting offers – as discussed in the previous chapter. Concerning the other two factors – the software agents for the parties – there exist nine possible levels for each factor – namely `MOCact`, `MOCpas`, `SMCact`, `SMCpas`, `MUMact`, `MUMpas`, `LEXact`, `LEXpas`, or `TFT` – resulting from combinations of different offer generation and concession strategies.

Similar to Henderson et al. (2003) we use a tournament-based simulation approach – a method already used by Axelrod (1980a; 1980b; 1984a; 1984b) to compare and evaluate strategies for the iterated prisoner's dilemma. This means for a given interaction protocol we let each type of software agent assuming one party's role in the automated negotiation negotiate with each other type of software agent assuming the role of the other party. Repeating this tournament for all three protocols results in a full factorial experimental design as depicted in Table 5.1.

The protocol and the two agents for the two parties together with the negotiation problem – i.e. the preferences of the two parties over the negotiation object – as input to the simulation fully parameterize a simulation run. In this full factorial design one system configuration i.e. one specific combination of a protocol, an agent representing the seller, and an agent representing the buyer is one treatment of the experimental design. The three levels for the factor interaction protocol and the nine levels for each of the two parties' software agents results in a total of $3 * 9 * 9 = 243$ treatments.

treatment	protocol	seller	buyer
1	`protocol 1`	`MOCact`	`MOCact`
2	`protocol 1`	`MOCact`	`MOCpas`
3	`protocol 1`	`MOCact`	`SMCact`
4	`protocol 1`	`MOCact`	`SMCpas`
5	`protocol 1`	`MOCact`	`MUMact`
6	`protocol 1`	`MOCact`	`MUMpas`
7	`protocol 1`	`MOCact`	`LEXact`
8	`protocol 1`	`MOCact`	`LEXpas`
9	`protocol 1`	`MOCact`	`TFT`
⋮	`protocol 1`	`MOCpas`	⋮
⋮	`protocol 1`	`SMCact`	⋮
⋮	`protocol 1`	`SMCpas`	⋮
⋮	`protocol 1`	`MUMact`	⋮
⋮	`protocol 1`	`MUMpas`	⋮
⋮	`protocol 1`	`LEXact`	⋮
⋮	`protocol 1`	`LEXpas`	⋮
73	`protocol 1`	`TFT`	`MOCact`
74	`protocol 1`	`TFT`	`MOCpas`
75	`protocol 1`	`TFT`	`SMCact`
76	`protocol 1`	`TFT`	`SMCpas`
77	`protocol 1`	`TFT`	`MUMact`
78	`protocol 1`	`TFT`	`MUMpas`
79	`protocol 1`	`TFT`	`LEXact`
80	`protocol 1`	`TFT`	`LEXpas`
81	`protocol 1`	`TFT`	`TFT`
82	`protocol 2`	`MOCact`	`MOCact`
⋮	`protocol 2`	⋮	⋮
162	`protocol 2`	`TFT`	`TFT`
163	`protocol 3`	`MOCact`	`MOCact`
⋮	`protocol 3`	⋮	⋮
243	`protocol 3`	`TFT`	`TFT`

Table 5.1: Experimental design

For each treatment we use the preferences elicited from human subjects in the 2,065 experiments on the Cypress-Itex case as input to evaluate the system configuration in different settings, which results in $243*2,065 = 501,795$ unique simulation runs. However, as mentioned in Chapter 4 the negotiation procedures of a system configuration are not purely deterministic, therefore several replications of these simulation runs have to be performed to account for stochastic effects. The number of necessary replications was set to three replications for each simulation run due to the analyzes in the subsequent section and Appendix B, so the total number of simulation runs multiplies to $501,795 * 3 = 1,505,385$ in our experimental design.

Though in general a high number of simulation runs – and the related high computational effort – could be reduced by some experimental design techniques, like using fractional designs, factor aggregation or reduction, factor screening techniques, or optimization techniques like response surface methods (Law and Kelton, 1991; Kleijnen and van Groenendaal, 1992) for our specific simulations, however, these methods are not applicable or appreciated. We have only three factors and all seem to be relevant so factor screening is not appropriate for our study, also we are highly interested in all treatments and the interactions of factors, which could get lost in fractional designs. Furthermore, optimization by response surface methods is not readily applicable in our setting with qualitative factor levels – though we are looking for systems that perform best – as the direction of improvement, necessary for the applicability of response surface methods, cannot be derived from qualitative factors easily. The cons, of more combinations and more computer resources needed in a full factorial design, are in our opinion outweighed by the pros, of being able to reliably study interactions between the factors, so that we decided to use the full factorial design for our study.

5.1.1 Replication

Due to stochastic effects in the interaction protocol and the decision making algorithms of the software agents[1] it does not make sense to look at only one particular simulation run, where stochastic effects could cause or influence the particular outcome, but in analyzing the data one has to focus on aggregated or average results. As mentioned in Chapter 2 in terminating systems – like automated negotiation – one run from the start condition of the simulation until the critical event that terminates the simulation yields a single observation on the response of interest. This observation clearly incorporates effects of the specific input variables or stochastic in the simulation procedure. To yield several observations in terminating systems, for which summary measures of the response variables can be calculated, several replications of the simulation run are necessary (Kleijnen, 1987; Pidd, 1992).

However, for achieving several observations of one simulation run first the necessary number of replications – over which summary measures then are calculated for further analyses – has to be determined. This number could be arbitrary large, but then would cause enormous computational effort – even if one single simulation run only takes some seconds or even a fraction of a second – especially if many treatments are to be considered in the experimental design, as it is the case for our study. To avoid unnecessary computational efforts and to avoid the risk of unreliable results in arbitrarily determining a too high or too low number of replications, respectively, we aim at determining a number of replications such that additional replications do not significantly effect average outcomes.

To determine whether or not there are significant differences in the average outcomes between one number of replications and the next higher number we statistically tested for differences in the means of several outcome measures between different numbers of replications. These outcome measures were the average proportion of agreements, the average duration of negotiations, and the two parties' average utility of reached agreements. In case there exist no significant differences between a given number of replications and the next higher one, this number of replications will

[1]As discussed in Chapter 4 the two aspects where stochastic comes into play are the start of negotiations – which is determined randomly by the interaction protocol with equal probability for each agent to start – and the choice of offers in case of ties – where the software agents choses one out of the alternatives it is indifferent between with equal probability as the next offer.

be sufficient to provide stable average results.

For the parametrization of the simulation runs of these pre-tests we use all agent combinations in all protocols and randomly sampled ten out of the 2, 065 negotiation problems considered for this study. Stable average results are found for a number of three replications as in none of the ten negotiations, none of the three protocols, and for none of the considered outcome measures there existed significant differences between the average outcomes of three and four replications. We therefore set the number of replications of one single simulation run to three.[2] The low number of only three replications to achieve stable average results is actually not very surprising. On the one hand only few stochastic effects play a role in the simulation model, which therefore only have little influence on the final outcome of a simulation run. On the other hand, if these stochastic effects accidentally influence the outcome of one simulation run it is reasonable that already a low number of replications will balance out their effect.

5.1.2 Variance reduction

We use the same set of negotiation problems as input for the automated negotiation systems – which are the treatments of our experimental design resulting from the combination of a specific software agent for each party and an interaction protocol – to reduce variance of the simulation output and therefore increase the precision of the results. Clearly the negotiation processes and outcomes are influenced by the negotiation problem for which automated negotiations are performed, therefore different system configurations should be compared in a similar setting to ensure that differences in the results are due to the system configuration and not due to differences in the input used to run the system. The use of common negotiation problems as input to the automated negotiation systems is closely related to the 'common random numbers' technique for variance reduction, where the same streams of random input numbers are used for different system configurations to ensure that these experience the same environmental input and therefore differences can only be due to differences in the system configuration.

However, distinct from the 'common random numbers' technique we do not use random numbers – sampled from some fitted theoretical or empirical distribution – as input to the systems, but the empirical data from the negotiation experiments. Law and Kelton (1991) argue against using empirical data as input for simulations, which is called trace-driven simulation. On the one hand with historical data as input to the simulation, only conditions that happened in the past can be reproduced, but other potentially interesting conditions – outside the range of the historical input data – cannot be examined as no such input data is available, but would be available if using fitted theoretical distributions. On the other hand seldom enough data is available for the desired amount of simulation runs, while here empirical or theoretical distributions fitted to the data can deliver any number of values necessary. These arguments, though valid in general, do not apply in our specific simulation study as we have many negotiation problems that can be used as input to the automated negotiation systems and as it would be difficult to determine and fit a theoretical or empirical distribution for complex utility functions.

[2]Consult Appendix B for detailed outcomes and statistical tests concerning the determination of the minimal necessary number of replications. Furthermore note, that we tested for up to five replications in a first attempt and also found no significant differences between average outcomes of four and five replications.

Using the empirical data in a trace-driven simulation furthermore enables to use the correlated inspection approach (Law and Kelton, 1991) for our analyses, which was originally proposed for operational validation, i.e. for comparing the correspondence of the real system's and the simulation system's outcomes to determine the validity of the simulation system. By using the negotiation problems from the experiments between humans we can apply the correlated inspection approach for outcome analysis rather than operational validation – which was argued to be impossible for automated negotiation as operative systems do not exist yet – to compare the performance of human negotiation and automated negotiation.

5.1.3 Execution of simulations

Simulations were performed on a HP Compaq 8510p Notebook with Intel Dual Core CPU T8300 @ 2.40 GHz and 2 GB RAM, with the applications `R` 2.7.1 and `Firebird` 2.1, running on the Windows Vista operation system. The simulation time for the software agent tournament under each protocol, the total number of turns, and the average turns per run, as well as the average time per turn are summarized in Table 5.2.

	total time (h)	total turns	avg. turns per run[1]	avg. time per run (sec)[1]
`protocol 1`	104.11	15,166,908	30.23	0.75
`protocol 2`	53.45	4,893,501	9.75	0.38
`protocol 3`	96.61	14,504,694	28.91	0.69
total	254.17	34,565,103	22.96	0.61

[1] On the basis of 501,795 simulation runs per protocol, and 1,505,385 runs in total

Table 5.2: Simulation time and turns

From this table it can be derived, that the major driver of the time needed for simulation are the number of turns in which messages are exchanged, as the simulation time increases with the number of turns necessary to finish the automated negotiation. `protocol 2` needs much fewer turns to come to an outcome in the negotiation than the other two protocols, which need about equal numbers of turns – `protocol 3` slightly fewer than `protocol 1` – and therefore simulation time. These differences mainly result from the possibility to abort the negotiations without agreement and from the proportion of simulation runs where such abruptions occur, as the number of turns and therefore the necessary simulation time actually does not differ between automated negotiations that reached an agreement, but are much lower for aborted negotiations. As `protocol 1` does not enable abruption of the negotiation, for all simulation runs negotiations have to be performed until an agreement is reached. However, agreements are not mandatory in `protocol 3` in which about 30% of the simulation runs were terminated earlier due to a lack of progress in the simulation, indicated by two subsequent `reject` messages by the two parties. These `reject` messages available to the software agents in `protocol 3` prolong the negotiation process and thereby reduce the differences between `protocol 1` and `protocol 3`. The effect of abruption of the automated negotiation becomes most obvious from the results for `protocol 2`, for which agreements were reached only in about 20% of the simulation runs. The average number of turns per simulation run and the time necessary for simulations is therefore significantly lower for `protocol 2` than for the other two protocols – results concerning the proportion of agreements reached and other outcome measures are presented and discussed in the subsequent chapter. Note that though the average duration of one simulation run for neither of the protocols exceeded on second, the overall time to perform the 1,505,385 simulation runs necessary for our experimental design accumulated to about ten and a half days.

5.2 Measurement

This section describes the dependent and independent variables used in the analyses of the output of the simulation.

5.2.1 Independent variables

The independent variables in the output analyses coincide with the components of automated negotiation, the automated negotiation system and the negotiation problem as input to this system. While the evaluation and comparison of different configurations for the automated negotiation system (Section 5.2.1.1) is the main concern of our system design study – and covered in the analyses in Sections 6.1 to 6.4 of Chapter 6, one must not forget to account for the sensitivity of outcomes to the system input – i.e. the negotiation problem, which is done in the sensitivity analyses in Section 6.5 of Chapter 6, by investigation of plots of the relations between the outcome measures and the integrativeness of the negotiation problem for different system configurations (Section 5.2.1.2).

5.2.1.1 System configuration

As mentioned a system configuration is fully determined by a software agent for each of the two parties and an interaction protocol. The three protocols together with nine possible agents for both the seller and the buyer party combine to 243 treatments. However, as discussed in Section 5.3, in case we are interested in the main effects of one component we merge the results of the treatments that are equal in the component of interest – e.g. analyzing the main effects of the protocols we group together all treatments that use the same protocol in the system configuration. For these groups of treatments with different protocols the remaining components are all equal i.e. the same combinations of agents and negotiation problems are represented in every group such that differences can only be due to the protocols.

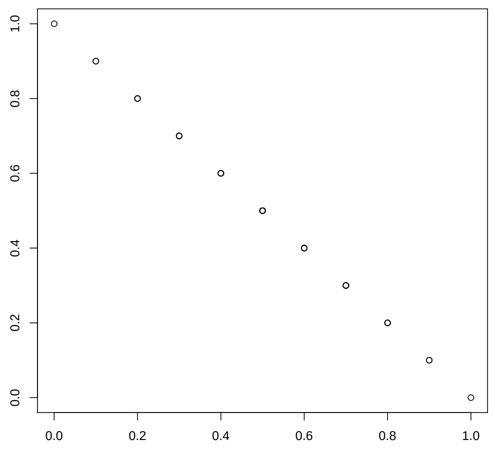
5.2.1.2 Integrativeness of the negotiation problem

While the system configuration is controllable by the users – or the system designer – a further independent variable not at the discretion of the system designer or user is the negotiation problem as an input to the system. In sensitivity analysis – concerning the sensitivity of the output of the simulation to its inputs – we use the integrativeness of the negotiation problem as an aggregate measure for the input to the system – i.e. the negotiation problem resulting from the preferences of the users over the negotiation object as discussed in Section 3.1.1 – an investigate how outcome measures not only depend on the components of the system configuration but also on this uncontrollable input factor.

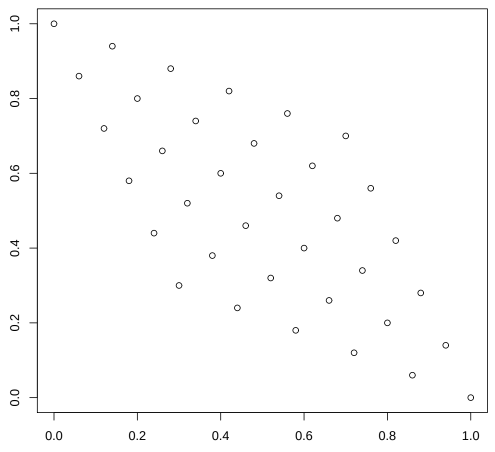
A commonly used classification of negotiation problems is to distinguish between distributive and integrative negotiations, which was first introduced by Walton and McKersie (1965) and is often used as structuring criterion for courses or books about negotiation (e.g. Lewicki et al., 1994; Raiffa et al., 2002; Kersten, 2007). Though used exchangeably in this text the term 'bargaining' often is used to refer to distributive and 'negotiation' to integrative settings (Lewicki et al., 1994). Many models of bargaining and negotiation focus exclusively on one of these two, either on distributive bargaining or on integrative negotiation – while Walton and McKersie (1965) see them as only two of four subprocesses, besides *attitudinal structuring* as efforts to influence the quality and nature of the relationship between the negotiating parties and *intra-organizational bargaining* as the conflict resolution within negotiating teams of one party, of their theory of negotiation (Lewicki et al., 1992). Others limit distributive bargaining to the exchange of more or less specific proposals for the terms of agreement on particular issues (Gulliver, 1979), while integrative negotiation is seen as the broader approach of defining and redefining the terms of the interdependence of parties (Walton and McKersie, 1965). Integrative negotiation then covers more than just offer exchange for specific issues, but also the search for new alternatives, issues, and options for mutual benefit, and is a more cooperative and creative task (Lewicki et al., 1992; Kersten et al., 2000).

The subsequently proposed measure of 'integrativeness of the negotiation problem' is inspired by the work of Tripp and Sondak (1992), who propose and discuss measures for assessing the integrativeness of an agreement found by parties in bilateral negotiations.[3] Assessing the integrativeness of one solution (the agreement) means comparing this focal solution to other possible solutions of the negotiation problem, however, assessing the integrativeness of the whole negotiation problem is more problematic as it involves the discussion of the whole set of possible solutions of the negotiation problem. Basis for our measure of integrativeness of the negotiation problem is a discussion of integrative and distributive negotiations. The distinction between integrative and distributive negotiations, though as mentioned, often used in negotiation literature and having a long tradition reaching back to the seminal contributions of Walton and McKersie (1965), is not clear at all, but definitions of integrative and distributive negotiations are ambiguous (Kersten et al., 2000). We define distributive negotiations as negotiations where the parties (at best – i.e. without leaving potential value at the bargaining table) can divide a fixed pie, such that any gain of one party is made at the expense of the other party. Therefore the best outcomes of distributive negotiations lie on the line connecting the maximum outcome of the two parties. On the other hand in integrative negotiations there exist solutions in the negotiation problem that allow for gains of both parties beyond splitting a fixed pie. Therefore the outcomes of integrative negotiations can exceed the linear combination of the maximal payoffs of the parties – see Figure 5.1.

[3]The measure of integrativeness of the agreement proposed by Tripp and Sondak (1992) bases on the comparison of the agreement with the number of other solutions in the negotiation problem that either dominate it or are dominated by it. Integrativeness of an agreement then is defined as *'1-(the number of possible agreements Pareto superior to the reference agreement/the sum of the number of possible agreements Pareto superior to the agreement and the number of possible agreements Pareto inferior to the agreement)'* (Tripp and Sondak, 1992, p.291).



(a) distributive



(b) integrative

Figure 5.1: Distributive and integrative negotiation problems

A measure of integrativeness therefore needs to determine the amount of such integrative solutions possible due to the structure of the negotiation problem at hand. Having a set $X = \{x_1, \ldots, x_n\}$ of n solutions x_i, and preferences of the negotiators represented by utility functions $u_j(\cdot)$ and $u_{-j}(\cdot)$ we define the set of distributive solutions D as those lying at or below the line connecting the best outcome for the two parties. The set of distributive solutions therefore consists of those solutions where the parties at best split a fixed pie among them (5.1).

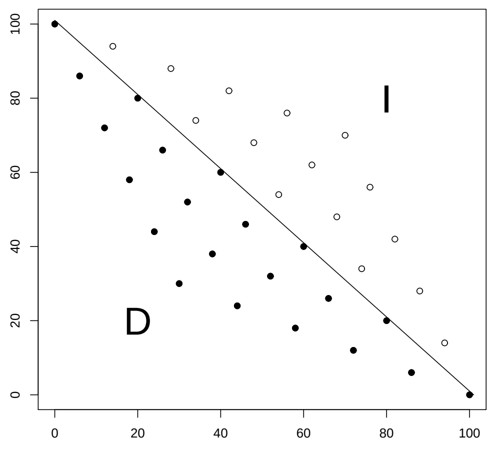


Figure 5.2: Integrativeness of the negotiation problem

$$D = \{x \in X | u_{-j}(x) \leq -\frac{max_{x_i} u_{-j}(x_i)}{max_{x_i} u_j(x_i)} u_j(x) + max_{x_i} u_{-j}(x_i)\} \tag{5.1}$$

In contrast all other solutions of the negotiation problem – i.e. those that lie above the line connecting the extreme outcomes – are assigned to the set I of integrative solutions (5.2). The sets D and I are also presented graphically in Figure 5.2.

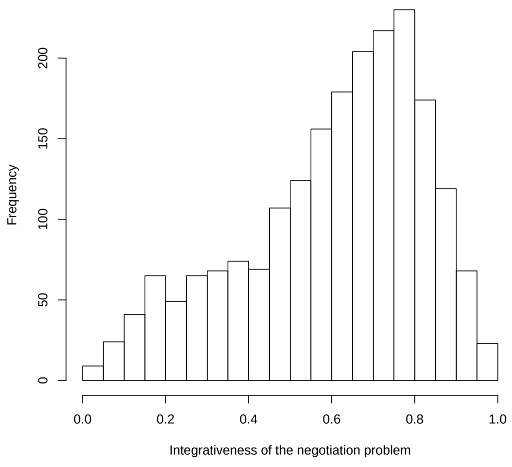
$$I = \{x_i | u_2(x_i) > -\frac{max_{x_i} u_{-j}(x_i)}{max_{x_i} u_j(x_i)} u_j(x) + max_{x_i} u_{-j}(x_i)\} \tag{5.2}$$

Integrativeness of the negotiation problem then can be calculated by (5.3).

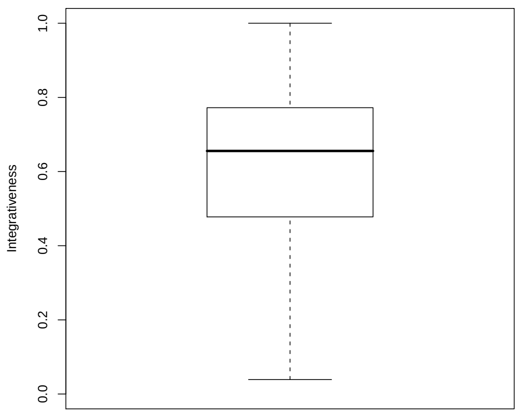
$$Integrativeness = \frac{|I|}{n} \tag{5.3}$$

This measure has the appealing feature that it ranges from 0 – if all solutions of the negotiation problem are distributive – to 1 – in case all solutions of the negotiation problem are integrative – and allows to compare different negotiation problems as it standardizes for the number of possible solutions. The measure depends on the structure of the negotiation problem and therefore covers the effect of all the factors influencing it – weight differences, partial utility curves, etc.[4]

For the negotiation problems of this study, where the best possible outcomes for the parties achieve a maximum utility of 100, this measure of integrativeness implies that all solutions that afford a sum of utilities of both negotiators of at most 100 are conceived as distributive solutions. Descriptive statistics of the integrativeness of the 2,065 negotiation problems of the experiments used for this study are provided in tabular form in Table 5.3, as well as in form of a histogram and a box-plot in Figure 5.3.



(a) Histogram



(b) Box-plot

Figure 5.3: Illustrations on the integrativeness of the negotiation problems

[4]Integrativeness as determined above can be conceived as the proportion of integrative solutions of all possible solutions of the negotiation problem, or the probability of selecting a integrative solution if one solution out of all possible solutions of the negotiation problem is chosen randomly.

min	0.00
1^{st} Q.	0.48
median	0.66
3^{rd} Q.	0.77
max	1.00
$\oslash$	0.61
$\pm$	0.22

Table 5.3: Integrativeness of the negotiation problems

5.2.2 Dependent variables

For the dependent variables of our simulation study we focus on negotiation outcome measures. Several outcome measures are considered as dependent variables of the study as each of them covers an specific aspect of the negotiation outcome. Prior research on negotiation identified fundamental trade-offs between different outcome measures for negotiations, so that these trade-offs can also be expected for different configurations of systems for automated negotiations – i.e. a system configuration achieving good results in one measure need not perform well for others. As the importance attached to different outcome measures by the users is not obvious and can differ between users we opt for a holistic evaluation of the negotiation outcome in using several outcome measures that complementary evaluate different aspects of the outcome of the negotiation.

5.2.2.1 Proportion of agreements

The outcome of negotiations can be defined in various ways (Tripp and Sondak, 1992). One obvious way, commonly used in empirical studies (e.g. in Coursey, 1982; Neale and Bazerman, 1985; Moore et al., 1999), is to consider whether or not an agreement was reached in the negotiation. For a set of comparable negotiations this results in a proportion of agreements or its inverse the impasse rate (Tripp and Sondak, 1992). In their review of dependent variables of negotiation studies Tripp and Sondak (1992) conclude that the impasse rate is rarely considered as a measure for the negotiation outcome, but that more often negotiations ending with an impasse are deleted from the sample and only of the negotiations that reached an agreement joint payoff – the sum of the parties utilities of the agreement – as the most widely used measure of joint performance is reported. Therefore we report the proportion of agreements $\frac{agr}{n}$ in the treatments as one outcome measure, which is simply the number of simulation runs that reached an agreement agr divided by the number of simulation runs in a treatment – or group of treatments – n.

While the outcome if the negotiation ends with the acceptance of an offer is an agreement on a specific set of options for the issues under negotiation, the outcome of termination due to `quit` or two subsequent `reject` messages is no agreement. For simulation runs without agreement we cannot report more than the proportion of agreements reached in the treatment. However, when an agreement is reached in a simulation run, the evaluation – according to the preferences of the parties – of this agreement provides additional information about the quality of the agreement. A high proportion of agreements does not necessarily indicate that one system configuration is better than an other, that achieves a lower proportion of agreements as not all agreements are equally good.

5.2.2.2 Proportion of Pareto-optimal agreements

As not all agreements are equally good, measures evaluating the quality of an agreement are necessary. One such measure evaluating the dyadic performance of the negotiators – or a system for automated negotiation – from an economic perspective is the Pareto-optimality of an agreement – or for a set of comparable negotiation the proportion of Pareto-optimal agreements. An agreement is Pareto-optimal if there exist no other possible solutions to the negotiation problem that dominates this focal agreement – i.e. that provides higher utility to one party without making the other party worse off compared to the focal agreement:[5]

> Pareto optimal agreements are those from which no additional joint gains are possible. When negotiators have achieved Pareto optimal agreements, no agreement is possible that would be preferred by both negotiators or would be preferred by one and to which the other would be indifferent. (Tripp and Sondak, 1992, p.279)

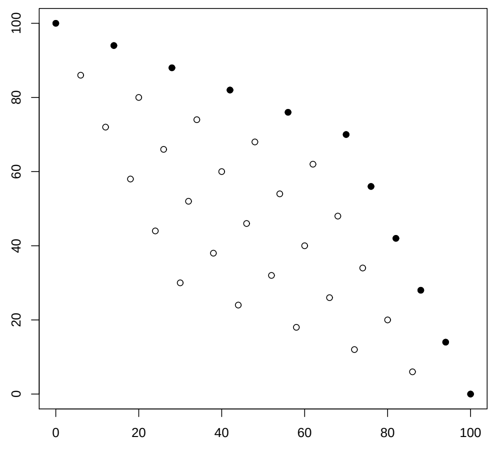


Figure 5.4: Pareto-optimal solutions in a negotiation problem

As Pareto-optimality can only be determined for agreements and not for outcomes of simulation runs other than an agreement, the simulation runs reaching Pareto-optimal solutions can only be a subset of those simulation runs that reach an agreement. Pareto-optimality distinguishes agreements between those that are and those that are not Pareto-optimal. So for several comparable negotiations we again can calculate the proportion of Pareto-optimal agreements for the treatments – or groups of treatments – as the number of simulation runs that reached a Pareto-optimal agreement eff divided by some basis. Candidates for this basis for calculation of the proportion of Pareto-optimal agreements are on the one hand the total number of simulation runs in the treatment n, or on the other hand the number of simulation runs that achieved an agreement agr. Using the total number of simulation runs in a treatment as divisor – which results in $\frac{eff}{n}$ as measure of the proportion of Pareto-optimal agreements – compounds the number

[5]Formally – using the notation from the previous section – the set P of Pareto-optimal solutions – illustrated as the filled points in Figure 5.4 – is the subset of the set X consisting of the solutions x only, for which holds true $\nexists y$ s.t. $u_j(y) > u_j(x) \wedge u_{-j}(y) \geq u_{-j}(x)$ or $u_j(y) \geq u_j(x) \wedge u_{-j}(y) > u_{-j}(x)$, $x, y \in X$, $x \neq y$.

of agreements reached with the Pareto-optimality of agreements. Even if most of the agreements reached in a treatment are Pareto-optimal, a low number of agreements reached would result in a low proportion of Pareto-optimal agreements as Pareto-optimality can only be determined for simulation runs that reached an agreement.

Using the the number of agreements reached in the simulation runs agr as basis for determining the proportion of Pareto-optimal agreements ($\frac{eff}{agr}$) would avoid the above mentioned deficit. However, the consensus of discussions and previous presentations of the results of this study was that one should not consider the Pareto-optimality of agreements independent of the number of agreements, we therefore accept the bias of this measure and determine the proportion of Pareto-optimal agreements on the basis of the total sample size as $\frac{eff}{n}$.[6]

5.2.2.3 Minimal distance to the Pareto frontier

As said Pareto-optimality discriminates between agreements that are or are not Pareto-optimal. However, just like not all agreements are equally good, not all agreements that are not Pareto-optimal are equally bad. Some of the possible solutions can be closer to the Pareto frontier than others, like solution B is closer to the Pareto frontier than solution A in Figure 5.5(a). Therefore we calculate for an agreement $\hat{x}$ the minimal Euclidean distance to the Pareto frontier. Using the notation and definitions provided in the previous sections the minimal euclidean distance ED – the length of the shortest straight line between the agreement and one of the solutions of the set of Pareto optimal solutions P i.e. the Pareto frontier – can be determined by (5.4).

$$ED = min_{x \in P} \sqrt{(u_j(x) - u_j(\hat{x}))^2 + (u_{-j}(x) - u_{-j}(\hat{x}))^2} \tag{5.4}$$

5.2.2.4 Contract imbalance

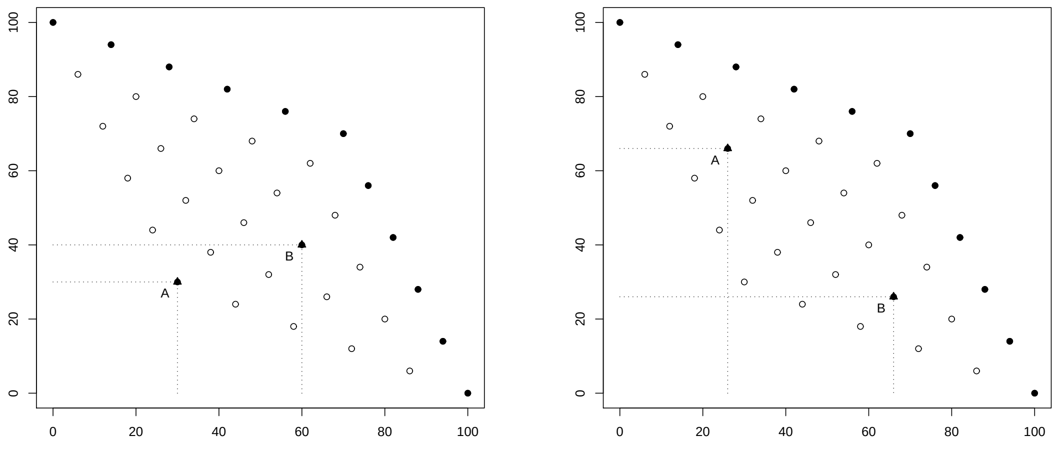
One further measure of joint performance often considered is the fairness of the agreement. Fairness could for example be determined by the distance to axiomatic solutions for the bargaining problem provided by cooperative game theory. The axiomatic approach, first proposed by Nash (1950), defines axioms as properties desirable for the solution and then determines solution functions that select for a given bargaining problem a certain package as solution to the bargaining problem, which features the properties stated in the axioms. One of these axioms used in many axiomatic solutions is symmetry – e.g. used in the well known Nash solution (Nash, 1950) or the Raiffa solution axiomatized by Kalai and Smorodinsky (1975). Symmetry is argued to be an axiom assuring fairness as it requires that for the same preferences the outcome should also be the same. However, another axiom most axiomatic solutions share is Pareto-optimality. When taking distance measures between actual agreement and axiomatic solutions that have to be both fair and Pareto-optimal the two concepts are compound: Distances between the agreement and the Pareto frontier can be large but the utilities of the agreement to the parties quite equal – and the agreement therefore could be considered fair.

[6]Note that if the proportion of Pareto-optimal agreements on basis of the agreements reached is of interest it can easily be derived by multiplying the proportion on total runs $\frac{eff}{n}$ with the proportion of agreements $\frac{agr}{n}$ – both provided as outcome measures in this dissertation – as $\frac{(\frac{eff}{n})}{(\frac{agr}{n})} = \frac{eff}{agr}$

We therefore decided to employ contract imbalance as a measure of unfairness as an additional outcome measure for negotiations in our study – the smaller the contract imbalance of an agreement, defined as the absolute value of the difference in utilities of the agreement $\hat{x}$ for the two parties ($\mid u_j(\hat{x}) - u_{-j}(\hat{x}) \mid$), the fairer is the agreement.

5.2.2.5 Individual utility

Besides the former outcome measures for negotiations which evaluate the quality of agreements at a dyadic level, we also consider the individual utility of an agreement for the two party as an additional outcome measure. Individuals might evaluate agreements quite different even if they are equal at the dyad level according the other outcome measures discussed so far. Consider that either of the solutions A or B in Figure 5.5(b) is chosen as agreement to the negotiation problem by the parties, then both agreements would not be Pareto-optimal, have the same distance to the Pareto frontier, and the same contract imbalance. However, due to the different utilities they provide to the individuals they clearly are not indifferent between these two possible solutions.



(a) Distance to the Pareto frontier

(b) Individual utility

Figure 5.5: Distance to the Pareto frontier and individual utilities of the agreement

5.3 Analysis

The standard statistical method for analysis for the type of data and experimental design – cardinal scaled dependent variables and several ordinal scaled independent variables – of our simulation study would be a three-way – due to the factors interaction protocol, seller software agent, and buyer software agent – analysis of variance (ANOVA) where the factor level combinations determine the different treatments to be investigated. For those treatments found to influence the dependent variables then some post-hoc comparisons can be performed to closer investigate these influences as the ANOVA itself only indicates whether such influences exist but not their direction and strength. Also a three-way analysis of covariance (ANCOVA) can be used when adding the integrativeness of the negotiation problems as additional cardinally scaled independent variable to the model, or when considering simultaneously all dependent variables in one model a multiple analysis of variance (MANOVA) or multiple analysis of covariance (MANCOVA) could be used (Backhaus et al., 2003; Hair et al., 2006).

So we examine whether our data fulfills the requirements for the analysis of variance, which are independence of observations, normal distribution of the dependent variables, and homogeneity of variances (Backhaus et al., 2003; Hair et al., 2006). Independence of observations is guaranteed by the settings of our simulation as one observation has no influences on other observations. To test normal distribution of the independent variables we use the Lilliefors test, a Kolomorov-Smirnov test for the composite hypothesis of normality for empirical distributions with estimated parameters (mean and variance).[7] Furthermore we use the Fligner-Killeen (median) test of homogeneity of variances across the treatments, which is the test most robust against departures from normality of all tests available for this purpose (Conover et al., 1981). The results of the normality tests and homogeneity of variances tests are presented in Table 5.4 together with parameters of the distributions of the independent variables (skewness and kurtosis).

	Lilliefors test		distribution parameters		Fligner-Killeen test		
Variable	D	p-value	skewness	kurtosis	χ^2	DF	p-value
prop. agr.	0.38	0.0000	-0.56	-1.60	127707.27	242	0.0000
prop. eff.	0.34	0.0000	0.73	-1.18	78925.01	242	0.0000
distance	0.24	0.0000	2.41	8.91	69251.01	242	0.0000
u. seller	0.06	0.0000	-0.79	1.04	59763.07	242	0.0000
u. buyer	0.08	0.0000	-0.79	1.04	63051.95	242	0.0000
imbalance	0.11	0.0000	1.35	1.94	69447.38	242	0.0000

Table 5.4: Results of normality and homogeneity of variances tests

As can be derived from Table 5.4 for all dependent variables the Lilliefors test of normality indicates significant non-normal distribution and the Fligner-Killeen test significant heterogeneity of variances across treatments. Note that it is argued that the F-test is robust if either the normality or the variance homogeneity assumption of the analysis of variance is violated, when the number of observations in the treatments is large and about equal in all treatments – so that under these conditions ANOVA still can be applied even if the data does not comply with its requirements. Though our sample sizes are large we do not have equal numbers of observations in the treatments for the dependent variables minimal distance to the Pareto frontier, utility of the agreement to the seller, utility of the agreement to the buyer, and contract imbalance, as these can be calculated only for simulation runs that reached an agreement and the proportion of agreements considerably differs between treatments. Furthermore as the statistical tests indicate highly significant violation of two of the three requirements for ANOVA, we decided not to apply ANOVA for analyses.

An alternative to parametric ANOVA, not relying on the requirements for ANOVA, is the non-parametric Kruskal-Wallis test which performs an one-way variance analysis by ranks – results

[7]Note that the test statistic of the Lilliefors test equals that of the Kolomorov-Smirnov test when comparing the sample against a normal distribution with mean and variance estimated from the sample, but it is not valid to use the p-value from the Kolomorov-Smirnov test for the composite hypothesis of normality (with unknown mean and variance) since the distribution of the test statistic is different when the parameters are estimated. Therefore we use the Lilliefors test which accounts for this difference (Thode, 2002).

presented in Table 5.5.

outcome	χ^2	df	p-value
prop. agr.	288713.04	242	0.0000
prop. eff.	119499.12	242	0.0000
distance	50481.55	242	0.0000
u. seller	135324.22	242	0.0000
u. buyer	140494.10	242	0.0000
imbalance	88011.86	242	0.0000

Table 5.5: Results of Kruskal-Wallis one-way variance analysis by ranks

Unlike m-way ANOVA, from which the main and interaction effects of m factors can be derived, the Kruskal-Wallis test is a one-way test, only testing whether all treatments are equal (null hypothesis) or whether there are differences between at least two treatments (alternative hypothesis), which is the case for all dependent variables in our study – as can be seen from Table 5.5. We therefore do post hoc tests to the Kruskal-Wallis test comparing (groups of) different treatments, to determine the sources of differences between the treatments. As the samples are not normal distributed we use non-parametric tests in multiple pairwise comparisons of the outcomes in different treatments. The statistical tests employed are the parameter free Wilcoxon rank sum test (equivalent to the Mann-Whitney-U test – with continuity correction) to test for differences in the means across treatments for minimal distance to the Pareto frontier, utility of the agreement to the seller (or focal party), utility of the agreement to the buyer (or opponent), and contract imbalance. For testing differences in proportions of agreements and Pareto-optimal agreements, as the two remaining outcome measures, we apply Pearson's non-parametric χ^2 test of independence (χ^2 contingency table test – with Yate's continuity correction) in multiple pairwise comparisons.

In multiple statistical hypothesis tests the problem of alpha-error inflation arises – i.e. an increased probability to incorrectly reject the null hypothesis – as a set of hypothesis is tested simultaneously on the same data in multiple comparison (Miller, 1981). A comparison refers to the comparison of two groups – treatment vs. control or treatment vs. treatment – and multiple comparison then means that several of such comparisons are performed on the same data set. For example with just one hypothesis test performed by comparison of two groups at the standard alpha level of 5% also the chance of incorrectly rejecting the null hypothesis is 5%. However, having multiple comparisons, e.g. 100 hypothesis tests where all null hypotheses are actually true, it is highly likely that at least some null hypothesis will be rejected incorrectly. At an alpha of 5% the expected number of incorrect rejections of the null (for 100 tests) is five i.e. on average in 5% of the comparisons, only due to the number of multiple comparisons not due to the underlying data, the null hypotheses are incorrectly rejected. To prevent this we control for the family wise error rate in adjusting the p-values by the – simple and conservative – Bonferroni-Holm method (Holm, 1979).

Chapter 6

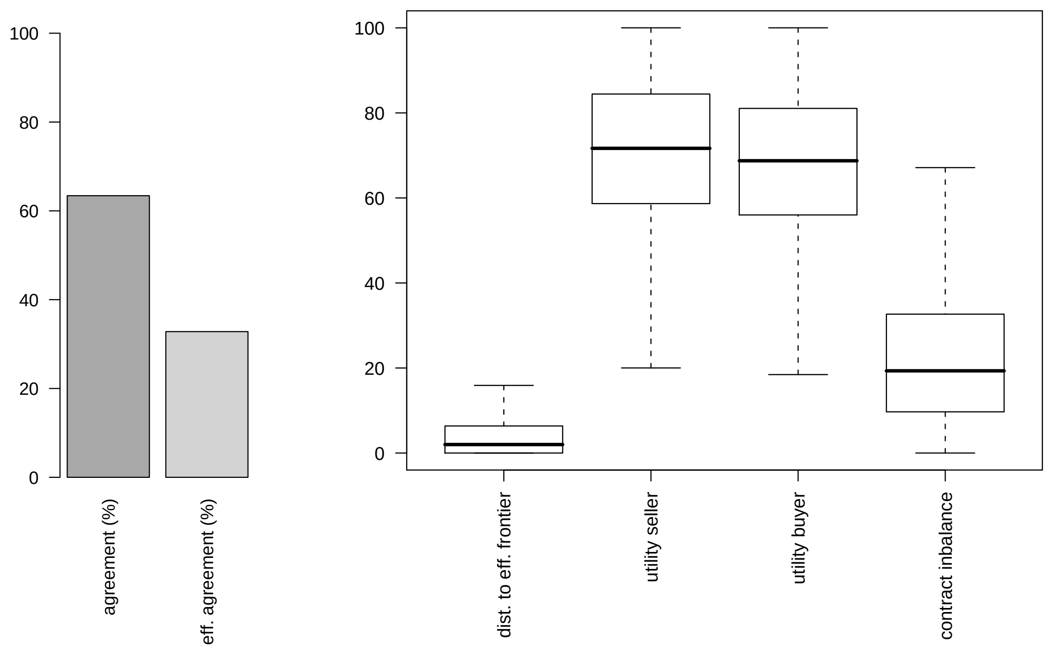
Results

Table 6.1 and Figure 6.1 provide descriptive statistics for the six outcome dimensions discussed in the previous chapter, note that the proportion of agreements for all simulation runs was 63.41% and the proportion of Pareto-optimal agreements was 32.80%. Interestingly automated negotiation achieves higher utility to the seller party than to the buyer party though neither the protocol nor the agent distinguish between parties. This phenomenon is not only found for the overall results but consistently in all treatments and all analyses. The better performance of automated negotiation for the seller party is even more puzzling if one compares the overall results to the negotiation experiments, where no such differences in individual utility of the two parties exist and is discussed in detail in Section 6.6.2.

	distance	u. seller	u. buyer	imbalance
min	0.00	20.01	18.45	0.00
$1^{st}Q.$	0.00	58.67	56.00	9.69
median	2.00	71.67	68.74	19.33
$3^{rd}Q.$	6.36	84.44	81.05	32.67
max	15.91	100.00	100.00	67.13
$\oslash$	4.29	70.31	67.48	23.65
$\pm$	6.02	18.18	18.88	18.87

Table 6.1: Descriptive statistics for dependent variables – automated negotiation

In line with the research questions of this study this chapter presents and discusses the analyses of the performance of automated negotiation versus human negotiation – in negotiation experiments with human subjects – (Section 6.1), analyses of the influence of different the system components – i.e. the different interaction protocols (Section 6.2) and the different software agents (Section 6.3) on the performance of the automated negotiation system for the different outcome measures for negotiation, as well as interactions between these components (Section 6.4). In a sensitivity analysis (Section 6.5) we also investigate the influence of the system input variables – i.e. factors other than the system configuration that cannot be controlled by the system user or designer – which in our case are the negotiation problems – summarized by their integrativeness for sensitivity analyzes – on the performance of different system configurations for automated negotiation for the outcome measures. All of these analyses are performed for all six, above discussed, outcome measures as dependent variables: proportion of agreements, proportion of Pareto-optimal agreements, minimal distance to the Pareto frontier, utility of the agreement to the seller (focal party's utility), utility of the agreement to the buyer (opponent's utility), and contract imbalance, to achieve a holistic evaluation of the system performance for various aspects of the outcome of the negotiation.



(a) Proportions (b) Averages

Figure 6.1: Results for the dependent variables over all simulation runs

For the comparison of automated negotiation with human negotiation we perform all 243 pairwise comparison of the $3*9*9 = 243$ treatments – i.e. the automated negotiation systems – versus the control – i.e. human negotiation in the negotiation experiments – for the six outcome measures. However, for the analyzes of the influence of different system configurations on the outcome measures we do not perform all pairwise comparisons. For these analyzes, rather than performing $\frac{t*(t-1)}{2}$ pairwise comparisons of all the $t = 3*9*9 = 243$ treatments resulting in $29,403$ pairwise comparisons – which would become not only complex but also confusing and therefore hard to interpret – we will perform multiple pairwise comparisons of our dependent outcome measures in grouping together treatments by their levels in the factors – means we merge all the observations for specific factor levels and compare the results among the different levels of the factor, thereby holding equal all the other factors (at all possible values) to get unbiased results.

For example when analyzing the influence of the protocol we group all treatments which use `protocol 1` in one sample, all treatments that use `protocol 2` in one sample, and those that use `protocol 3` in another sample. Therefore, within these samples the same agent combinations and the same negotiation problems are used in the simulation runs, and the samples only differ in the interaction protocol. Outcome measures then are compared across samples in multiple pairwise comparisons. For the comparison of the performance of specific agents we group together all treatments where one side is represented by this focal agent.[1] Combining results from all treatments with same seller agent and pairwise comparing these groups again leaves all other factors equally represented i.e. all three protocols, all nine opponent agents, and all negotiation problems are represented in each of the groups.

These analyses deal with the 'main effects' of choosing different software agents and interaction protocols for an automated negotiation system. For detailed analyses of the sources of differences we also investigate the impact of different offer generation strategies – `LEX`, `MOC`, `MUM`, and `SMC` – and concession strategies – `act` and `pas` – in the three protocols which should provide insights about the interaction of these components in determining the performance of an automated negotiation system.[2] For this purpose we compare the performance of software agents in the different protocols in grouping the treatments that use both the same seller agent and the same protocol to one sample – with all opponent agents for the buyer party and all negotiation problems represented in all samples –, which results in $3 * 9 = 27$ samples and $\frac{27*26}{2} = 351$ pairwise comparisons per outcome measure.

As mentioned in Chapter 5 we do not compare all system configurations with each other for differences in outcome measures for several reasons. First, it would be technically complex and confusing to perform and report all $29,403$ comparisons of the 243 treatments. Second, the additional value of these analyses is questionable, the results of the treatments are presented in the comparison of the treatments versus human negotiation anyway and from studying these results, provided in Table C.1 in the Appendix C, one can see that the combinations of two software agents actually only combines the effects found for these two agents against all opponents. Finally, from the perspective of the context in which the analyses take place the users of automated negotiation systems can only choose their own agent and influence the choice of the protocol – which has to be chosen jointly by the parties – but have no influence on the software agent chosen by the opponent.

The last analyses performed are sensitivity analyses of the performance of system components for the six outcome measures depending on the integrativeness of the negotiation problem, which is used as input to the automated negotiation system. For sensitivity analyzes we graphically analyze for the different interaction protocols and the different software agents if the results in the outcome measures for different values of the integrativeness of the negotiation problem, as well as the direction and size of these effects.

6.1 Automated negotiation vs. human negotiation

As argued in Chapter 3 the benchmark for the evaluation of automated negotiation should be the currently used mechanism, which is – as we use the preferences human subjects indicated in negotiation experiments – for this study the outcome of human negotiation. Table 6.2 and

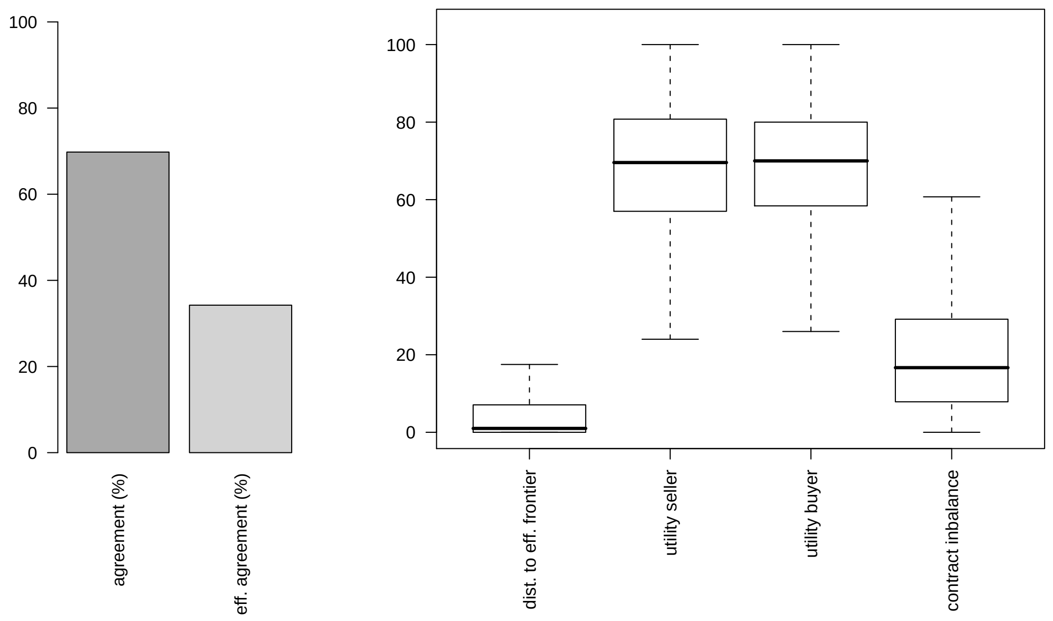
[1]Note that we take the seller side as focal side, and that it actually not matters which side is taken as agents do not distinguish between the party they represent but make decisions solely dependent of the preferences and the opponent's behavior.

[2]Note that the offer generation strategy and concession strategy comparison is not applicable to `TFT`, as `TFT` determines the next offer by reciprocating the last concession of the opponent and making it as similar as possible to the opponent's last offer in case of ties as discussed in Chapter 4. We do, however, compare the performance of `TFT` in the different protocols.

Figure 6.2 provide descriptive statistics of the results of the human negotiation experiments for the outcome measures of our study. 1,441 (69.78%) of the 2,065 experiments reached agreements, of which 707 were Pareto-optimal given the negotiation problem determined by the preferences of the subjects in the experiments (i.e. 34.24% of total experiments and 49.06% of the experiments that reached an agreement).

	distance	u. seller	u. buyer	imbalance
min	0.00	24.00	26.00	0.00
$1^{st}Q.$	0.00	57.00	58.40	7.86
median	1.00	69.57	70.00	16.67
$3^{rd}Q.$	7.07	80.77	80.00	29.17
max	17.49	100.00	100.00	60.73
$\oslash$	5.24	67.93	67.42	20.40
$\pm$	8.80	18.22	17.28	16.58

Table 6.2: Descriptive statistics for dependent variables – negotiation experiments



(a) Proportions (b) Averages

Figure 6.2: Dependent variables in experiments with human subjects

As discussed it is – most often implicitly – assumed in literature that automated negotiation can achieve superior results than humans could. Due to a lack of empirical studies this argument remains an assumption. For the overall data of all treatments the results on our outcome measures are mixed concerning this hypothesis. Comparing Tables 6.1 and 6.2 – which provide the descriptive statistics for the six outcome measures for all simulation runs and the negotiation experiments, respectively – and the proportions of (Pareto-optimal) agreements one can see that automated negotiation seems to be inferior in the proportion of agreements and the proportion of Pareto-optimal agreements reached, and also leads to greater contract imbalance of the utility of the parties if an agreement was reached, indicating unfairer outcomes. On the other hand agreements in automated negotiation are closer to the Pareto frontier and the utility of agreements to the seller party are higher than in human negotiation, while it is fairly the same for the buyer side.

As these effects could result from some treatments performing worse than others we conduct multiple pairwise comparisons of the 243 treatments – which constitute as mentioned above different system configurations for automated negotiation – with human negotiation as control – i.e. 243 comparisons treatment versus control – and test the one-sided hypothesis that automated negotiation achieves better results than human negotiation for the six outcome measures. Better means different things for the different outcome measures. The alternative hypothesis is that the outcome is greater in automated negotiation than in human negotiation for proportion of agreements, proportion of Pareto-optimal agreements, utility of an agreement to the seller and the buyer, but it is that the outcome is less in automated negotiation than in human negotiation for the outcome measures distance to the Pareto frontier and contract imbalance. Table C.1 – due to its length, as covering all 243 pairwise comparisons, shifted to Appendix C – provides the results of these comparisons.

For the same negotiation problems as input to the automated negotiation systems, at $p < 0.05$, 130 of the 243 treatments (53.50% of all treatments) reached a higher proportion of agreements than humans did in experiments, 90 (37.04%) treatments reached a higher proportion of Pareto-optimal agreements, and the distance to the Pareto-efficient frontier was smaller in 79 (32.51%) treatments. Furthermore the utility of agreements to the seller was higher than the utility sellers achieved in human negotiations for 154 (63.37%) system configurations but higher for the buyer only in 127 (52.26%) system configurations. Contract imbalance was smaller in automated negotiation – and therefore fairness larger – in 112 (46.09%) of the 243 treatments. We derive from this that not all treatments outperform human negotiations but some system configurations do worse, while others do better. Furthermore, when comparing the treatments in Table C.1 it becomes evident that trade-offs between the outcome measures exist. In most cases system configurations performing better than humans in experiment for some outcome measures perform worse than humans in others.

However, though these trade-offs are present three systems significantly outperformed human performance in negotiation experiments in all six outcome measures (at $p < 0.05$). These three system configurations are (protocol-selleragent-buyeragent): (i) `3-MOCact-MOCact`, (ii) `3-MOCact-MOCpas`, and (iii) `3-MOCpas-MOCact`. These systems achieve between five and ten percent more agreements than human negotiation and between 25% and 30% more Pareto-optimal agreements, furthermore agreements reached are on average about five utility points better for both the seller and the buyer party, closer to the Pareto frontier by five points and more balanced by twelve points – measured in the utility space of the negotiation problem – than agreements reached in the negotiation experiments for the same negotiation problems – see Table C.1. Note that the system configuration of these automated negotiation systems outperforming human negotiation reveals an interesting pattern. Given the negotiation problems used, for an automated negotiation system to outperform human negotiation in all outcome dimensions it has to operate under `protocol 3` and consist of `MOC` software agents only, that follow the monotonic concession strategy in generating offers, of which at least one has to follow the active concession strategy and therefore make first concession steps if the opponent reciprocated past concessions – i.e. at least either the buyer or the seller agent, or both, has to be of type `MOCact`.

A further interesting result of the comparison of the treatments to the human negotiation control group is the difference between the performance of agents for the different parties – only in 127 treatments agreements with higher utility to the buyer were reached, but 154 treatments achieved agreements of higher utility to seller party. This is interesting because the treatments are symmetric for both the seller and the buyer party – i.e. besides the 27 treatments where the same types of software agents are used for both the seller and the buyer, for each treatment where the seller is represented by a software agent of type A and the buyer by a software agent of type B there exists an other treatment where the seller is represented by B and the buyer by A. So the overall results should not differ due to treatments. Furthermore even for treatments where the seller and the buyer are represented by the same software agent the utility of an agreement to the buyer is always lower than the utility to the seller, which is interesting as the decision making of the software agents by no means depends on the role of the user it represents in the negotiation, but only on the preferences of the user and the behavior of the opponent, as discussed in Chapter 4.[3] Both the trade-offs between different outcome measures and the seemingly 'role-dependent' performance of automated negotiation systems are analyzed in detail in Section 6.6.

6.2 Comparison of interaction protocols

We compare the performance of automated negotiation systems using different protocols in their configuration to analyze the main effects of the protocol on the outcome measures – proportion of agreements, proportion of Pareto-optimal agreements, utility of the agreement to the buyer and seller, distance of the agreement to the Pareto frontier, and contract imbalance.[4] The results of the multiple pairwise comparisons of these samples – within which the same agent combinations and negotiation problems were used in the simulation runs and between which therefore only the protocol differs – are presented in Tables 6.3 to 6.8 for our six outcome measures.

	%	protocol 1	protocol 2
protocol 1	100.00		
protocol 2	19.85	0.0000	
protocol 3	70.39	0.0000	0.0000

Table 6.3: Proportion of agreements in different protocols

	%	protocol 1	protocol 2
protocol 1	46.47		
protocol 2	11.48	0.0000	
protocol 3	40.47	0.0000	0.0000

Table 6.4: Proportion of Pareto-optimal agreements in different protocols

[3]The only exception here are the three treatments that use only `TFT`-agents in the three protocols, which not only achieve about equal utilities for both sides – around 66 utility points – within a treatment – i.e. for the same protocol –, but also across the treatments – i.e. for the different protocols.

[4]Proportions of agreements and Pareto-optimal agreements in the samples are compared using Pearson's χ^2 independence tests, differences in all other outcome measures are analyzed by means of Wilcoxon ranked sum tests. Samples sizes for proportion measures are $81 * 3 * 2065 = 501795$ as each agent combination is used for three replications of all negotiation problems as input to the systems, for all other measures it is 501795 times the proportion of agreement reached in the protocols provided in Table 6.3. Test statistics are omitted in the subsequent tables to save space and to be consistent with the presentation of the results in subsequent tables, where the provision of the test statistics would consume too much space, and can be requested from the author. All p-values reported are adjusted for multiple comparisons by the Bonferroni-Holm method.

	⊘	protocol 1	protocol 2
protocol 1	67.70		
protocol 2	77.04	0.0000	
protocol 3	71.45	0.0000	0.0000

Table 6.5: Utility of the seller in different protocols

	⊘	protocol 1	protocol 2
protocol 1	63.81		
protocol 2	75.89	0.0000	
protocol 3	69.45	0.0000	0.0000

Table 6.6: Utility of the buyer in different protocols

	⊘	protocol 1	protocol 2
protocol 1	5.30		
protocol 2	3.28	0.0000	
protocol 3	3.34	0.0000	0.0000

Table 6.7: Minimal distance to the Pareto frontier in different protocols

	⊘	protocol 1	protocol 2
protocol 1	29.43		
protocol 2	18.38	0.0000	
protocol 3	17.99	0.0000	0.1520

Table 6.8: Contract imbalance in different protocols

The proportion of agreements clearly is highest in `protocol 1`, which by the mechanisms of this protocol reaches agreements in all simulation runs. That the proportion of reached agreements is 100% in `protocol 1`, which neither allows software agents to break off negotiations by `quit` messages nor enables termination due to two subsequent `reject` messages – as these are not permitted under `protocol 1` – is not further surprising but actually adds to the verification of the implementation of the simulation program. The second highest proportion of agreements (70.39%) is reached by systems employing `protocol 3`, while in only 20% of the simulation runs an agreement is reached when using `protocol 2`. Differences between the proportions of reached agreements are highly significant as can be seen from Table 6.3 ($p < 0.001$ for all pairwise comparisons). The sources of these differences are obvious from the protocol descriptions, while `protocol 1` forces the agents to reach an agreement, in `protocol 3` the software agents have the possibility to reject unfavorable offers but keep on negotiating, which could lead to a termination of the negotiation if the opponent also sends a `reject` message as it was the case in the 30% of the simulation runs that achieved no agreement under `protocol 3`. Finally, software agents immediately break off the negotiation in case of such unfavorable offers in `protocol 2`. So the the possibility to reject unfavorable offers of the opponent under `protocol 3` allowed to achieve agreements in additional 50% of the simulation runs where the immediate termination by `quit` messages under `protocol 2` prevented such agreements. However, the possibility of rejecting offers also caused 30% of the simulation runs to fail to reach an agreement compared to `protocol 1`, where this was not possible.

The absolute proportion of Pareto-optimal agreements – as a proportion of the total number of simulation runs in a sample – shows the same tendency (Table 6.4). The proportion of Pareto-optimal agreements is higher for the protocols that reach more agreements and therefore highest in `protocol 1` (46.47 %), followed by `protocol 3` (40.47 %), and systems that use `protocol 2` (11.48 %) – again differences between the proportion of Pareto-optimal agreements are highly significant ($p < 0.001$ for all pairwise comparisons). These numbers, however, indicate that the differences in proportions are smaller for the proportion of Pareto-optimal agreements, than for the overall proportion of agreements. This becomes more evident when looking at the relative proportions of Pareto-optimal agreements – i.e. proportions calculated not on the basis of all simulation runs but on the basis of those that reached an agreement as discussed in the previous chapter – as provided in Table 6.9. The relative proportion of Pareto-optimal agreements is highest for `protocol 3`, followed by `protocol 2`, but lowest – in contrast to the absolute proportion of Pareto-optimal agreements – in simulation runs with systems that use `protocol 1`. This indicates that the possibility to not engage in further negotiations on the basis of unfavorable offers of the opponent through breaking off the negotiation (`protocol 2`) or eliciting a new offer from the opponent (`protocol 3`) moves agreements closer to the Pareto frontier, which consequently match it more often, than not having these options (`protocol 1`), however, this is achieved at the cost of risking a negotiation break-off, which can be seen from the former results on the proportion of agreements.

	⊘	`protocol 1`	`protocol 2`
`protocol 1`	46.47		
`protocol 2`	58.48	0.0000	
`protocol 3`	57.73	0.0000	0.0002

Table 6.9: Relative proportion of Pareto-optimal agreements in different protocols

Results comparable to those for the relative proportion of Pareto-optimal agreements are obtained for the minimal distance to the Pareto frontier (Table 6.7) – the similarity of these measures is not further surprising as the minimal distance to the Pareto frontier is zero when agreements are Pareto-optimal – so this measure covers aspects of the relative proportion of Pareto-optimal agreements well. The distance of reached agreements to the Pareto frontier is smallest when using `protocol 2`, followed by `protocol 3`, and both, `protocol 2` and `3`, achieve agreements closer to the Pareto frontier than `protocol 1` does (all $p < 0.001$).

Similar to the relative proportion of Pareto-optimal agreements and minimal distance to the Pareto frontier, both parties, the seller and the buyer party, achieve – in case of agreement – highest utility of the agreement in `protocol 2` followed by `protocol 3` and `1` (all $p < 0.001$ as can be seen from Tables 6.5 and 6.6, respectively). As already mentioned above from Tables 6.5 and 6.6 one can see that the utility of an agreement is lower for the buyer than for the seller in all protocols. Note that the differences in utilities between the parties slightly differs between the protocols used for the system and is highest for `protocol 1` (difference of 4 utility points), followed by `protocol 3` (2 points) and `protocol 2` (around 1.5 points) – so that the difference between the utility of an agreement to the buyer and the seller increases with the proportion of agreements reached and decreases with the relative proportion of the Pareto-optimal agreements.

Finally if an agreement is reached, the contract imbalance – as the difference between the utilities of the agreement to the parties – is smallest and therefore agreements are fairest in `protocol 3` and `protocol 2` without significant differences between these two protocols, which both achieve more balanced agreements than `protocol 1` ($p < 0.001$ in both cases). This partly is explained by the differences between the utility of an agreement to seller and buyer, discussed above, if these differences are low – as in `protocol 2` and `3` – also the contract imbalance is low, while it is high otherwise – as found for `protocol 1`.

The lower contract imbalance found for `protocol 2` and `protocol 3` also can be explained by the mechanisms of these protocols. While unfavorable offers can be rejected and new offers can be elicited in `protocol 3`, or negotiations can be broken off in `protocol 2`, which either leads to a more favorable and balanced agreement or no agreement at all, such messages are not enabled in `protocol 1`, where strategies have no means to cope with bad offers and interrupt their offering strategies, which could lead to unfavorable agreements for the weaker party – in terms of preferences over the negotiation object – and therefore result in higher contract imbalance and consequently unfairer agreements. This argumentation also provides a hint for the reason behind the seemingly 'role-dependent' performance of automated negotiations that achieves agreements of higher individual utility for the seller than for the buyer, found in this and the previous section, as the system components do not discriminate between the roles the reason must lie in the preferences of the parties and the group of buyers seems to have unfavorable preferences compared to the group of sellers.

6.3 Comparison of software agents

After this analysis of the general influence of the protocol on different outcome measures – for the same software agent combinations and negotiation problems – we shift our attention to the software agents as the second main component in automated negotiation systems. As the software agents do not discriminate between the parties they represent, but base decision making only on the preferences indicated to them, we analyze the performance for a focal party – in our case the seller party – grouping together in one sample all treatments that use the same type of software agent to represent the seller party. Thereby across the samples only the software agent that represents the seller differs and within the samples this seller agent is used in simulation runs with all other agent types representing the buyer party (including a version of itself), in all protocols, and for all negotiation problems.[5]

	%	LEXact	LEXpas	MOCact	MOCpas	MUMact	MUMpas	SMCact	SMCpas
LEXact	66.22								
LEXpas	58.12	0.0000							
MOCact	64.09	0.0000	0.0000						
MOCpas	58.96	0.0000	0.0000	0.0000					
MUMact	71.56	0.0000	0.0000	0.0000	0.0000				
MUMpas	61.40	0.0000	0.0000	0.0000	0.0000	0.0000			
SMCact	65.15	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000		
SMCpas	58.60	0.0000	0.0151	0.0000	0.0363	0.0000	0.0000	0.0000	
TFT	66.62	0.0264	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table 6.10: Proportion of agreements for different software agents

The proportion of agreements reached in automated negotiation with the nine different software agents representing the seller party – negotiating in all three protocols, with all nine software agents representing the buyer, for three replications of all 2,065 negotiation problems – and multiple pairwise comparisons of these proportions are presented in Table 6.10. The proportions of agreements reached and their statistical comparison – with Pearson's χ^2 independence test –

[5]Same as for the comparison of protocols, the proportions of agreements and Pareto-optimal agreements in the samples are compared using Pearson's χ^2 independence tests and differences in all other outcome measures are analyzed by means of Wilcoxon ranked sum tests. In the nine samples (one for each type of software agent) the samples sizes for proportion measures are $9 * 3 * 3 * 2065 = 167265$. This number of simulation runs results from the combination of a specific software agent with all nine other software agents for three replications of the 2065 negotiation problems in the three protocols. As the outcome measures minimal distance to the Pareto frontier, contract imbalance, and utility of the agreement to the focal party and the opponent can be calculated for those simulation runs that resulted in an agreement only, sample sizes for these outcome measures can be derived by multiplying the total number of simulation runs in the sample (167265) with the proportion of agreements reached in this sample – provided in Table 6.10. In Tables 6.10 to 6.15 we omit test statistics due to space restrictions but gladly provide the interested reader with values on request. All p-values reported are adjusted for multiple comparisons by the Bonferroni-Holm method.

reveal, holding all other factors equal at all possible levels, a clear ranking of the software agents (as all $p < 0.05$ at least). In systems where for the seller party `MUMact` is used the proportion of reached agreements is highest (71.56 %), followed by `TFT`, `LEXact`, `SMCact`, `MOCact`, `MUMpas`, `MOCpas`, `SMCpas`, and `LEXpas`, which reached with 58.12% agreements over all simulation runs the lowest performance in this outcome measure. This ranking indicates that software agents with active concession strategies and `TFT` reach a higher proportion of agreements than software agents embodying a passive concession strategy. The higher proportion of agreement with active concession making and `TFT` can be explained by their mechanisms, as these software agents make offers with first concession steps if the opponent reciprocated previous concessions (`act`) or fully reciprocate the opponents previous concessions (`TFT`) – thereby also making concession steps ahead –, which is beneficial to the prospects of reaching an agreement in negotiations that could result in a break off or termination of negotiations – in case the protocols enable this – if such concession steps are not taken. The effect of the offer generation strategy is not clear from this ranking, also the effect of the protocol – as in the compared samples all protocols are covered – and whether this ranking is insensitive of the protocol used in the automated negotiation system or if there exist interactions between the software agent and the protocol – this is analyzed for the proportion of agreements, as well as for the other outcome measures, in the subsequent section.

	%	`LEXact`	`LEXpas`	`MOCact`	`MOCpas`	`MUMact`	`MUMpas`	`SMCact`	`SMCpas`
`LEXact`	28.16								
`LEXpas`	24.15	0.0000							
`MOCact`	42.32	0.0000	0.0000						
`MOCpas`	38.35	0.0000	0.0000	0.0000					
`MUMact`	34.41	0.0000	0.0000	0.0000	0.0000				
`MUMpas`	28.98	0.0000	0.0000	0.0000	0.0000	0.0000			
`SMCact`	32.29	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000		
`SMCpas`	28.66	0.0046	0.0000	0.0000	0.0000	0.0000	0.0442	0.0000	
`TFT`	37.93	0.0000	0.0000	0.0000	0.0223	0.0000	0.0000	0.0000	0.0000

Table 6.11: Proportions of Pareto-optimal agreements for different software agents

Also for the proportion of Pareto-optimal agreements a clear ranking of software agents emerges from the multiple pairwise comparisons summarized in Table 6.11 (all $p < 0.05$). While we found no differences between the ordering of interaction protocols concerning the proportion of agreements and the proportion of Pareto-optimal agreements that are achieved with automated negotiation systems using different interaction protocols in the previous section of this chapter, for the software agents these rankings differ. Automated negotiations in which – for all opponent software agents, negotiation protocols, and negotiation problems – `MOCact` represents the seller party reach a significantly higher proportion of Pareto-optimal agreements than all other agents with 42.32 % of the simulation runs ending with a Pareto-optimal agreement. `MOCact` – which makes offers with monotonically decreasing utility and first concession steps if the opponent reciprocated previous concession – is followed by `MOCpas`, `TFT`, `MUMact`, `SMCact`, `MUMpas`, `SMCpas`, `LEXact`, and finally `LEXpas`, which reaches Pareto-optimal agreements only in 24.15% of the simulation runs where this software agent is used for the seller party.

This ranking is not only different from the previous one concerning the proportion of agreements, but also reveals a different pattern. While for the proportion of agreements reached the concession strategy was of highest importance, for the upper and lower end of the ranking of software agents for the proportion of Pareto-optimal agreements the offer generation strategy matters – as the two

MOC-agents achieved the two highest while the two LEX-agents achieved the two lowest proportions of Pareto-optimal agreements. The concession strategy then determines in a second step the ordering of these agents, as for both MOC and LEX the active version is ranked higher. Higher influence of the concession strategy can be found for the middle of the software agent ranking for the proportion of Pareto-optimal agreements, as for both MUM and SMC, the active versions achieve better values in this outcome measure than the passive concession making versions do.

That the offer generation strategy is of higher importance for the proportion of Pareto-optimal agreements than for the proportion of agreements is plausible as Pareto-optimality depends more on the actual configuration of the offers, while for the acceptance of an offer of the opponent it 'only' has to provide higher utility than the next offer the focal software agent would send – according to the acceptance criterion used with the software agents (see Chapter 4). However, one must not forget, that the proportions of Pareto-optimal agreements are absolute proportions over all simulation runs, so that the proportion of agreements reached influences this outcome measure. This can be an explanation for the higher proportions of Pareto-optimal agreements reached by active concession making strategies and TFT compared to passive concession making for the same offer generation strategy – which reach more agreements – as we would assume the opposite here. Passive concession making strategies resist making first concession steps and thereby should increase the utility of an agreement – however, simultaneously lowering the prospect of reaching an agreement as can be derived from Table 6.10 – to the party using a software agent that follows this strategy and thereby should push agreements towards the Pareto frontier. Given this we will see the actual effect of software agents concerning the efficiency of reached agreements in the discussion of the next outcome measure – minimal distance to the Pareto frontier, which only can be calculated for simulation runs that reached an agreement – which was found to closely resemble the results concerning the relative proportion of Pareto-optimal agreements in Section 6.2.

	⌀	LEXact	LEXpas	MOCact	MOCpas	MUMact	MUMpas	SMCact	SMCpas
LEXact	5.56								
LEXpas	5.78	0.0014							
MOCact	2.52	0.0000	0.0000						
MOCpas	2.61	0.0000	0.0000	0.0127					
MUMact	4.85	0.0000	0.0000	0.0000	0.0000				
MUMpas	5.13	0.0000	0.0000	0.0000	0.0000	0.0092			
SMCact	3.96	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000		
SMCpas	4.07	0.0000	0.0000	0.0000	0.0000	0.0008	0.0000	0.0274	
TFT	4.06	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table 6.12: Minimal distance to the Pareto frontier for different software agents

The results and comparative analyses concerning the minimal distance to the Pareto frontier again form a clear ranking of software agents (all $p < 0.05$), similar to that found for the proportion of Pareto-optimal agreements – see Table 6.12. MOCact agents achieve results closest to the Pareto frontier – with an average minimal distance of 2.52 points in the utility space – followed by MOCpas, SMCact, TFT, SMCpas, MUMact, MUMpas, LEXact, and with an average minimal distance of 5.78 points agreements reached in automated negotiation systems with LEXpas representing the seller were farthest from the Pareto frontier.

Compared to the software agent ranking found for the proportion of Pareto-optimal agreements, this ranking shows the importance of the offer generation strategy more clearly, however the assumed better outcomes of passive concession making are not present as for the same offer generation strategy actively conceding software agents always achieve agreements closer to the Pareto frontier than passively conceding versions.

Tables 6.13 and 6.14 show the average individual utilities of a reached agreement to the focal party (seller – Table 6.13) and the opponent party (buyer – Table 6.14) for specific software agents used to represent the seller side in the automated negotiation systems, as well as the outcomes of multiple pairwise comparisons for these outcome measures.

	∅	`LEXact`	`LEXpas`	`MOCact`	`MOCpas`	`MUMact`	`MUMpas`	`SMCact`	`SMCpas`
`LEXact`	68.24								
`LEXpas`	68.82	0.0000							
`MOCact`	82.31	0.0000	0.0000						
`MOCpas`	82.98	0.0000	0.0000	0.0000					
`MUMact`	63.91	0.0000	0.0000	0.0000	0.0000				
`MUMpas`	64.94	0.0000	0.0000	0.0000	0.0000	0.0000			
`SMCact`	72.42	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000		
`SMCpas`	73.11	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	
`TFT`	58.64	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table 6.13: Focal agent's (seller) utility for different software agents

	∅	`LEXact`	`LEXpas`	`MOCact`	`MOCpas`	`MUMact`	`MUMpas`	`SMCact`	`SMCpas`
`LEXact`	68.13								
`LEXpas`	68.33	0.0326							
`MOCact`	56.83	0.0000	0.0000						
`MOCpas`	56.66	0.0000	0.0000	1.0000					
`MUMact`	72.22	0.0000	0.0000	0.0000	0.0000				
`MUMpas`	72.19	0.0000	0.0000	0.0000	0.0000	1.0000			
`SMCact`	65.99	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000		
`SMCpas`	65.99	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.6008	
`TFT`	78.83	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table 6.14: Opponent's (buyer) utility for different software agents

Utilities of agreements achieved for the party they represent significantly differ between the software agents (all $p < 0.001$) as can be seen from Table 6.13. Here, `MOCpas` achieves, with 82.98 points the highest utility score for its party, followed by `MOCact`, `SMCpas`, `SMCact`, `LEXpas`, `LEXact`, `MUMpas`, `MUMact`, and finally `TFT`, which achieved a relatively low utility score of 58.64 points only. So, as for the proportion of Pareto-optimal agreements and the minimal distance to the Pareto frontier outcome measures, also for the individual utility of an agreement to the focal software agent the offer generation strategy has major impact. Moreover, for individual utility of the agreements – and therefore the individual performance of software agents – we find the positive effects assumed for passive concession making strategies – which, however were not found for joint performance measures as discussed above. We argue that this ordering results from the more systematic offer generation followed by the higher ranked agents. As can be derived from the description of the different offer generation mechanisms in Chapter 4, `MOC` is the most systematic offer generation mechanism, proposing first all offers of a given utility level before this level is decreased to the next lower level, where again all offers are proposed etc. `SMC` only is different from `MOC` in that it never proposes offers of the same utility level but strict monotonically decreases the level of demanded utility. `MUM` proposes offers with least concession in one issue only, and `LEX` proposes offers that constitute concession following a lexicographic ordering of possible solutions. TFT, which is ranked lowest in the average utility of agreements achieved for the party it represents, on the other hand, can be considered as following the least systematic offer generation as it generates offers that reciprocate the opponents' concessions but does not follow any rules in this offer generation that prescribe a specific path through the set of possible solutions.

Concerning the utility of a reached agreement – with different software agents representing the seller side and all types of software agents representing the buyer side in systems using all protocols and negotiating all negotiation problems – to the opponent the highest average utility is achieved when TFT is used by the focal party (78.83). Second best results for this outcome measure are achieved for the opponent if the focal party's software agent is of type MUM, followed by LEX, SMC, and MOC, which results in the worst agreements from the point of view of the opponent's utility. While differences are highly significant between offer strategies, they are insignificant between active and passive concession making strategies for software agents following the same offer generation strategy – with exception of LEX, which follows a lexicographical ordering of offers in its offer generation strategy, where passive concession making is slightly better than active concession making and this difference is significant at $p < 0.05$. That it is better for the opponent if the focal party uses a LEXpas rather than a LEXact software agent is surprising, as passive concession making means fewer and smaller concessions compared to active concession making, but can result from canceling out of 'bad' negotiation problems, which could lead only to mediocre agreements when LEXpas is used, as LEXpas is the software agent that achieved the lowest proportion of agreement. So negotiation problems that would lead to unfavorable agreements, reach no agreement at all in simulation runs that use LEXpas and the remaining negotiation problems cause not only higher utility to the focal party but also to the opponent – compared to the usage of LEXact.

	⌀	LEXact	LEXpas	MOCact	MOCpas	MUMact	MUMpas	SMCact	SMCpas
LEXact	19.65								
LEXpas	20.36	0.0002							
MOCact	27.31	0.0000	0.0000						
MOCpas	28.12	0.0000	0.0000	0.0000					
MUMact	23.87	0.0000	0.0000	0.0000	0.0000				
MUMpas	24.63	0.0000	0.0000	0.0000	0.0000	0.5405			
SMCact	21.95	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000		
SMCpas	22.60	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0008	
TFT	24.58	0.0000	0.0000	0.2350	0.0000	0.0000	0.0000	0.0000	0.0000

Table 6.15: Contract imbalance for different software agents

These results, for the opponent's utility of an agreement if different software agents are used by the focal party, not only underline the higher influence of the offer generation strategy for both, the focal as well as the opponent's individual utility of an agreement, but also show that the ordering of offer generation strategies is opposite for the parties. Moreover, from the above considerations concerning the effects on the utility of the focal and the opponent party when a certain agent is used by the focal party, one already can derive some expectation about the effects of the usage of agents on the contract imbalance – unfairness – of reached agreements. As the agents that perform good for a focal party make the opponent party worse of and vice versa, those agents which lie in the middle of both rankings should be the best in terms of contract imbalance – i.e. achieve more balanced and therefore fairer agreements. Recall that the ranking of offer generation strategies in decreasing order of their average utility score for the focal agent was MOC – SMC – LEX – MUM – TFT, and for the opponent party's average utility score exactly inverse, i.e. TFT – MUM – LEX – SMC – MOC. So we assume LEX agents to lead to fairer agreements followed by SMC and MUM agents and finally TFT and MOC agents.

Table 6.15 supports this assumption, as LEX agents achieve smaller contract imbalance than SMC agents, which again achieve fairer contracts than MUM agents. Furthermore, TFT and MOC software agents used to represent the focal party (seller) in the automated negotiations achieve agreements where the difference between the two parties utilities is highest (all $p < 0.001$, with exception of MOCact versus TFT). The significant differences between active and passive versions of the agents – for same offer generation strategies – indicates a minor but significant effect of the concession strategy on the contract imbalance, namely that active conceding leads to fairer outcomes than passive conceding (all $p < 0.001$, except for MUM offer generation – which makes offers in order to provide concessions in the issue where they cost least – where differences in contract imbalance between MUMact and MUMpas are not significant). So besides the major effects of the offer generation strategy, the positive effect of passive concession making strategies on the utility of the agreement to the party that uses them and the lack of effects on the opponent party's utility shown in Tables 6.13 and 6.14 respectively, leads to imbalance and therefore smaller fairness of the agreements.

6.4 Agent features and agent-protocol interaction

After this analyses of the main effects of the interaction protocols (Section 6.2) and the software agents – used by a focal party – (Section 6.3) of different configurations of the automated negotiation system on the six outcome measures, we will focus on the interaction effects and effect sizes in this section. For this purpose we group all treatments which use the same software agent to represent a focal party (seller) in a specific interaction protocol in one sample. This results in $9 * 3 = 27$ different samples and $\frac{(3*9)*(3*9-1)}{2}$ pairwise comparisons of samples. Within the samples again all other components are equal i.e. all nine software agents are used as opponent for three replications of the 2065 negotiation problems. The tables providing the full results of these multiple pairwise comparisons (Tables C.2 to C.19) are, due to their length, moved to Appendix C and the Tables of this section (Tables 6.16 to 6.21) provide a summary of the relevant comparisons only.[6]

From Table 6.16[7] it can be derived, that the main influence on the proportion of agreements steams from the interaction protocol used in the automated negotiation system. This proportion differs for the same software agents between protocol 2 – that allows quit messages and thereby a break off of negotiations by the software agents – and protocol 3 – that allows reject messages to temporary interrupt the offering strategy and terminates negotiations if two

[6]Note that both, the results of comparisons with the TFT agent and the p-values for the comparisons of different protocols are not presented in Tables 6.16 to 6.21, but only the results of these comparisons are discussed – for the actual values we refer to the Appendix C. This is necessary as on the one hand TFT cannot be divided into concession strategy and offer generation strategy – see Chapter 4 for a discussion –, and therefore does not fit in the structure of the tables. On the other also providing significance values for the comparison between protocols would consume too much space and therefore undermine the idea of these tables, to summarizing the main results.

[7]Note that we omit comparisons of the proportion of agreement for protocol 1 as all simulation runs in this protocol are forced to reach agreements and so the proportions are all equal and all 100%.

such `reject` messages are sent by the software agents subsequently – between around 42% (for `MUMpas`) to 68% (for `MOCact`). Compared to `protocol 1`, which allows neither `quit` nor `reject` messages and therefore results in agreements in all simulation runs, differences in the proportion of agreements to `protocol 3` vary in the range of 42% (for `LEXpas`) to 16% (for MUMact), and difference to `protocol 2` are in a range of 88% (for `MOCpas`) to 70% (for `MUMact`). Same as for the comparison of interaction protocols – for all software agents – in Section 6.2, it can be seen from Tables C.2 to C.4 in Appendix C, that each software agent achieves significantly higher proportions of agreements in `protocol 1` than in `protocol 3`, where the proportion again is higher than in `protocol 2` (all $p < 0.001$).

`protocol 2`			`act`					`pas`		`act-pas`
	%	`LEX`	`MOC`	`MUM`		%	`LEX`	`MOC`	`MUM`	
`LEX`	21.08				`LEX`	16.55				0.0000
`MOC`	12.32	0.0000			`MOC`	11.58	0.0000			0.0005
`MUM`	30.31	0.0000	0.0000		`MUM`	20.99	0.0000	0.0000		0.0000
`SMC`	17.98	0.0000	0.0000	0.0000	`SMC`	15.12	0.0000	0.0000	0.0000	0.0000
`protocol 3`			`act`					`pas`		`act-pas`
	%	`LEX`	`MOC`	`MUM`		%	`LEX`	`MOC`	`MUM`	
`LEX`	77.57				`LEX`	57.82				0.0000
`MOC`	79.94	0.0000			`MOC`	65.30	0.0000			0.0000
`MUM`	84.36	0.0000	0.0000		`MUM`	63.22	0.0000	0.0000		0.0000
`SMC`	77.47	1.0000	0.0000	0.0000	`SMC`	60.69	0.0000	0.0000	0.0000	0.0000

Table 6.16: Summary of system component interactions for the proportion of agreements

When comparing the ranking of the software agents concerning the proportion of agreements they reach in all protocols and separated in the different protocols, one can see that the influence of the offer generation strategy mainly steams from the differences offer generation makes in `protocol 3` – between 15% for `MOC` agents and 30% for `LEX` agents – while it has only minor influence on the proportion of agreements in `protocol 2`. The opposite holds true for the concession strategy. While both concession strategies achieve a proportion of agreements in a range of 7% under `protocol 3` this range is much broader and between 9% (for passive conceding agents) and 12% (for active conceding agents) for `protocol 2` – as mentioned the proportion of agreements for simulation runs with `protocol 1` is equal and 100% for all software agents. The ranking of offer generation strategies varies considerably between the different protocols for some of them – especially `MOCact`, which is ranked second in `protocol 3` but only penultimate in `protocol 2` – and only minor or not at all for others – like `MUMact` is ranked second (behind `TFT`) in `protocol 2` and also first in `protocol 3`, or `LEXact` which holds the third place in the rankings for both protocols.

Comparing the results of different combinations of software agents (representing the seller) and protocols concerning the proportion of Pareto-optimal agreements yields the results summarized in Table 6.17. The results for the protocols for the single software agents coincide with the results presented in Section 6.2 – namely all software agents reach higher proportions of Pareto-optimal agreements under `protocol 1` than under `protocol 3`, which again is better than `protocol 2` (all $p < 0.05$). The only exception is `MUMact` which achieves under `protocol 3` significantly more Pareto-optimal agreements than under `protocol 1` ($p = 0.0001$). That this higher proportion of Pareto-optimal agreements in `protocol 1` than in `protocol 2` and `3` is due to the higher proportion of agreements reached in these protocols is somewhat revealed by this deviation, as

`MUMact` reached a very high proportion of agreements – compared to the other software agents – under `protocol 3` (84.36%).

Besides these differences due to the protocol – which are all significant but larger for `protocol 1` and `3` compared to `protocol 2` and not so large between `protocol 1` and `protocol 3` – the main differences between the performance of software agents in all protocols is caused by the different offer generation strategies of the agents. This strong influence of the offer generation strategy was also found in the comparison of software agents for all protocols in Section 6.3. In `protocol 1` the proportion of Pareto-optimal agreements differs in a range of 20% for the offer generation strategies, but not at all for the concession strategies, when the same offer generation strategy is used by the software agent. This is not surprising, as in `protocol 1` no interruption of offering is permitted and therefore it should not make a difference how the decision about offer interruption – i.e. following either the active or passive concession strategy – is made. So this result 'only' adds to the verification of the correct implementation of the simulation program. However, the proportion of Pareto-optimal agreements varies in a range of 7% for the offer generation strategies in `protocol 2`, where different concession strategies (for otherwise identical software agents) lead only to differences of at most 5% (for `MUM`), and it varies in a range of 24% for the offer generation strategies under `protocol 3`, where different concession strategies account for a maximal difference of 11% only (between `MOCact` and `MOCpas`).

`protocol 1`			`act`					`pas`		`act-pas`
	%	`LEX`	`MOC`	`MUM`		%	`LEX`	`MOC`	`MUM`	
`LEX`	37.23				`LEX`	36.53				0.1947
`MOC`	57.89	0.0000			`MOC`	58.10	0.0000			1.0000
`MUM`	42.74	0.0000	0.0000		`MUM`	42.06	0.0000	0.0000		0.2681
`SMC`	44.26	0.0000	0.0000	0.0000	`SMC`	44.32	0.0000	0.0000	0.0000	1.0000
`protocol 2`			`act`					`pas`		`act-pas`
	%	`LEX`	`MOC`	`MUM`		%	`LEX`	`MOC`	`MUM`	
`LEX`	10.91				`LEX`	8.85				0.0000
`MOC`	8.88	0.0000			`MOC`	8.29	0.0133			0.0089
`MUM`	16.35	0.0000	0.0000		`MUM`	11.31	0.0000	0.0000		0.0000
`SMC`	10.37	0.0517	0.0000	0.0000	`SMC`	9.00	1.0000	0.0005	0.0000	0.0000
`protocol 3`			`act`					`pas`		`act-pas`
	%	`LEX`	`MOC`	`MUM`		%	`LEX`	`MOC`	`MUM`	
`LEX`	36.35				`LEX`	27.07				0.0000
`MOC`	60.20	0.0000			`MOC`	48.67	0.0000			0.0000
`MUM`	44.13	0.0000	0.0000		`MUM`	33.56	0.0000	0.0000		0.0000
`SMC`	42.23	0.0000	0.0000	0.0000	`SMC`	32.66	0.0000	0.0000	0.0234	0.0000

Table 6.17: Summary of system component interactions for the proportion of Pareto-optimal agreements

Same as found for samples grouping together all agent combinations for different protocols – in Section 6.2 – also in samples separating the different software agents, most of them (`SMC`, `MUM`, and `LEX` – as can be seen from Tables C.8 to C.10 in Appendix C) reach agreements significantly closer to the Pareto frontier in automated negotiation systems operating under `protocol 2`, followed by systems using `protocol 3`, and `protocol 1`. Only for `TFT` no significant difference of the minimal distance to the Pareto frontier is found for `protocol 2` and `protocol 3`, and both `MOC` agents (`MOCact` and `MOCpas`) reach agreements significantly closer to the Pareto frontier in `protocol 3` compared to the agreements reached in `protocol 2` with these agents ($p < 0.001$).

`protocol 1`			`act`					`pas`		`act-pas`
	∅	`LEX`	`MOC`	`MUM`		%	`LEX`	`MOC`	`MUM`	
`LEX`	6.74				`LEX`	6.89				1.0000
`MOC`	3.60	0.0000			`MOC`	3.58	0.0000			1.0000
`MUM`	6.17	0.0000	0.0000		`MUM`	6.53	0.0000	0.0000		0.2536
`SMC`	4.89	0.0000	0.0000	0.0000	`SMC`	4.92	0.0000	0.0000	0.0000	1.0000
`protocol 2`			`act`					`pas`		`act-pas`
	∅	`LEX`	`MOC`	`MUM`		%	`LEX`	`MOC`	`MUM`	
`LEX`	3.96				`LEX`	3.77				0.9355
`MOC`	1.81	0.0000			`MOC`	1.82	0.0000			1.0000
`MUM`	3.42	0.0001	0.0000		`MUM`	3.43	0.3344	0.0000		1.0000
`SMC`	2.92	0.0000	0.0000	0.0258	`SMC`	2.79	0.0000	0.0000	0.0001	0.5078
`protocol 3`			`act`					`pas`		`act-pas`
	∅	`LEX`	`MOC`	`MUM`		%	`LEX`	`MOC`	`MUM`	
`LEX`	4.64				`LEX`	4.74				1.0000
`MOC`	1.39	0.0000			`MOC`	1.45	0.0000			1.0000
`MUM`	3.98	0.0000	0.0000		`MUM`	3.86	0.0000	0.0000		0.2336
`SMC`	3.16	0.0000	0.0000	0.0000	`SMC`	3.26	0.0000	0.0000	1.0000	1.0000

Table 6.18: Summary of system component interactions for the minimal distance to the Pareto frontier

Concerning the components of the software agents it can be seen from Table 6.18 that the higher importance of a software agent's offer generation strategy as opposed to its concession strategy, found when comparing different software agents in all protocols in Section 6.3, also can be observed for the protocols individually. While there are only insignificant differences in the minimal distance of the utility of reached agreements to the Pareto frontier with respect to the concession strategy, these distances vary around two to three points in utility space for different offer generation strategies. `MOC`, the offer generation strategies that operates most systematical – as discussed above – is ranked first in all interaction protocols concerning the closeness of agreements reached to the Pareto frontier. The following ordering also decreases with systematical offer generation of the software agents and is `SMC`, `MUM`, and `LEX`. The differences in average minimal distance to the Pareto frontier this ranking bases on are significant at $p < 0.05$ at least, with exception of the differences between the passively conceding versions of `MUM` and `LEX` for `protocol 2`, as well as `SMC` and `MUM` for `protocol 3`.

The results for the individual utility of an agreement to the focal party and its opponent, when a certain interaction protocol and a certain software agent to represent the focal party are used together in the automated negotiation system, can be found in Tables 6.19 (focal parties utility) and 6.20 (opponent's utility). Some of the differences in the individual utilities of the focal party are caused by the interaction protocol used in the automated negotiation and are in general consistent for all software agents with the results found in Section 6.2. Namely that agreements reached in `protocol 2` are of higher utility to the focal party than those reached in `protocol 3` and `protocol 1` (all $p < 0.001$). Furthermore `protocol 3` combined with each software agent achieves higher utility scores than `protocol 1` for six of the nine software agents. The differences in the focal parties utility of the agreement, however, are not significantly different between `protocol 1` and `protocol 3` for the agents `SMCact` and `TFT`, and `MOCpas` achieved significantly higher utility when combined with `protocol 1` than with `protocol 3` ($p < 0.001$).

While the performance of identical software agents combined with different protocols varies maximally around 15 utility points between the protocols, within a protocol offer generation strategies account for differences of up to around 25 utility points – where differences are higher for `protocol 1` than for `protocol 2` and lowest in `protocol 3` – so the offer generation strategy has stronger influence on the individual utility of the focal party than the interaction protocol has. The ordering of these agents in each of the protocols is the same as that found in Section 6.3 - namely in decreasing average utility achieved for the party they represent in the automated negotiations `MOC` – `SMC` – `LEX` – `MUM` – `TFT` (all $p < 0.001$).

protocol 1			act					pas		act-pas
	⊘	LEX	MOC	MUM		%	LEX	MOC	MUM	
LEX	65.15				LEX	64.78				1.0000
MOC	82.96	0.0000			MOC	82.98	0.0000			1.0000
MUM	58.10	0.0000	0.0000		MUM	57.15	0.0000	0.0000		0.0715
SMC	70.09	0.0000	0.0000	0.0000	SMC	69.96	0.0000	0.0000	0.0000	1.0000
protocol 2			act					pas		act-pas
	⊘	LEX	MOC	MUM		%	LEX	MOC	MUM	
LEX	78.26				LEX	80.54				0.0000
MOC	91.30	0.0000			MOC	91.61	0.0000			0.7173
MUM	73.72	0.0000	0.0000		MUM	78.03	0.0000	0.0000		0.0000
SMC	81.78	0.0000	0.0000	0.0000	SMC	83.94	0.0000	0.0000	0.0000	0.0000
protocol 3			act					pas		act-pas
	⊘	LEX	MOC	MUM		%	LEX	MOC	MUM	
LEX	68.80				LEX	71.15				0.0000
MOC	80.00	0.0000			MOC	81.35	0.0000			0.0000
MUM	66.17	0.0000	0.0000		MUM	70.65	0.0000	0.0000		0.0000
SMC	72.44	0.0000	0.0000	0.0000	SMC	74.40	0.0000	0.0000	0.0000	0.0000

Table 6.19: Summary of system component interactions for the utility of the focal agent (seller)

Finally results vary only up to five utility points for different concession strategies, with passive concession making achieving significantly higher individual utility for the focal party than active concession making in `protocol 2` and `protocol 3` – with exception of `MOC`, for which differences between `MOCact` and `MOCpas` in `protocol 2` are not significant. Note that in `protocol 1` there are no significant differences between active and passive concession making as the protocol allows no interruption, and therefore it adds to the verification of the simulation program that different mechanisms to interrupt offering do not influence the results for a given offer generation strategy in this protocol.

The tendencies observed for the utility of the represented party also hold true for the utility of the opponent. The offer generation strategy has the strongest effect on the opponent's utility of an agreement – at least for `protocol 1` and `protocol 2`, followed by the interaction protocol used in the automated negotiation system, and the concession strategy of the software agent. The effects of the interaction protocol are merely the same for the own and the opponent's utility – i.e. automated negotiation systems operating under `protocol 2` achieve higher utility than systems operating under `protocol 3` and `protocol 1` – as can be derived from Tables C.14 to C.16 in Appendix C – where differences are significant at $p < 0.001$ for all comparisons except that `MUM` software agents achieve agreements of similar utility for the opponent in `protocol 1` and `protocol 2` and so differences are insignificant ($p = 1.0000$ for `MOCact` and $p = 0.3969$ for `MOCpas` respectively).

`protocol 1`			`act`					`pas`		`act-pas`
	∅	`LEX`	`MOC`	`MUM`		%	`LEX`	`MOC`	`MUM`	
`LEX`	65.38				`LEX`	65.45				1.0000
`MOC`	49.45	0.0000			`MOC`	49.49	0.0000			1.0000
`MUM`	72.01	0.0000	0.0000		`MUM`	72.17	0.0000	0.0000		1.0000
`SMC`	62.07	0.0000	0.0000	0.0000	`SMC`	62.14	0.0000	0.0000	0.0000	1.0000
`protocol 2`			`act`					`pas`		`act-pas`
	∅	`LEX`	`MOC`	`MUM`		%	`LEX`	`MOC`	`MUM`	
`LEX`	73.82				`LEX`	75.46				0.0000
`MOC`	74.22	1.0000			`MOC`	74.60	0.1135			1.0000
`MUM`	73.38	1.0000	0.2303		`MUM`	73.99	0.0001	1.0000		0.2805
`SMC`	74.79	0.0408	0.9003	0.0000	`SMC`	75.81	1.0000	0.0081	0.0000	0.0204
`protocol 3`			`act`					`pas`		`act-pas`
	∅	`LEX`	`MOC`	`MUM`		%	`LEX`	`MOC`	`MUM`	
`LEX`	69.63				`LEX`	70.38				0.0002
`MOC`	62.36	0.0000			`MOC`	62.98	0.0000			0.0047
`MUM`	71.95	0.0000	0.0000		`MUM`	71.49	0.0000	0.0000		0.0379
`SMC`	68.03	0.0000	0.0000	0.0000	`SMC`	68.55	0.0000	0.0000	0.0000	0.0444

Table 6.20: Summary of system component interactions for the utility of the opponent (buyer)

It is intuitive that software agents achieving better agreements in automated negotiations for the focal negotiator they represent consequently cause lower utility of this agreement to the opponent, which can also be seen from Table 6.20 for `protocol 1` and `protocol 3`. The order of software agents achieving higher utility for the opponent is the inverse of the order of software agents achieving high utility for the focal party – i.e. `TFT` – `MUM` – `LEX` – `SMC` – `MOC`. Note that the same result was found when comparing the performance of software agents from the perspective of the opponent's utility of the agreement for all protocols in Section 6.3. However, for `protocol 2` both the offer generation strategy as well as the concession strategy of the focal party's software agents have only minor influence on the utility of the opponent, so that in this protocol the software agent chosen to represent the opponent seems to be of higher influence.

The concession strategy of the software agent representing the focal party has minor influence on the opponent's utility score of an agreement. Comparisons for `MOC` and `MUM` in `protocol 2` show only insignificant differences between different concession strategies for these offer generation strategies, in all other cases passive concession making of the focal software agents leads to higher utility to the opponent than active concession making ($p < 0.05$ at least). That there are no differences between `act` and `pas` software agents in `protocol 1` is caused by the mechanisms imposed by this protocol, which do not allow offering interruption so that these results are not surprising, but add to the verification of the simulation program. That passive concession making by the focal software agents benefits the opponent is surprising if one considers that this concession strategy causes lower and fewer concessions compared to active concession making, however, it may result from the fact that many negotiation problems that would lead to inferior agreements are already terminated or broken off before an agreement is achieved as passive concession making strategies reach a lower proportion of agreements in the simulation runs.

The contract imbalance – as difference between the parties utilities of the agreement – strongest differs between `protocol 1` and the other two protocols with differences between the same agents of up to 24 utility points – for software agents following the `MOC` strategy in offer generation. Furthermore, though differences are not so large, contract imbalance is significantly larger for many software agents if combined with `protocol 3` than with `protocol 2` ($p < 0.001$ for `MOC`, SMC, TFT, and the active conceding version of MUM as can be seen from Tables C.17 to C.19 in Appendix C).

protocol 1			act					pas		act-pas
	∅	LEX	MOC	MUM		%	LEX	MOC	MUM	
LEX	24.67				LEX	24.99				1.0000
MOC	35.49	0.0000			MOC	35.46	0.0000			1.0000
MUM	31.54	0.0000	0.0000		MUM	32.71	0.0000	0.0000		1.0000
SMC	28.37	0.0000	0.0000	0.0000	SMC	28.57	0.0000	0.0000	0.0000	1.0000
protocol 2			act					pas		act-pas
	∅	LEX	MOC	MUM		%	LEX	MOC	MUM	
LEX	15.33				LEX	15.48				1.0000
MOC	18.84	0.0000			MOC	18.74	0.0000			1.0000
MUM	16.69	0.0000	0.0000		MUM	16.45	0.0002	0.0000		1.0000
SMC	16.04	0.9188	0.0000	0.0040	SMC	15.99	1.0000	0.0000	0.1939	1.0000
protocol 3			act					pas		act-pas
	∅	LEX	MOC	MUM		%	LEX	MOC	MUM	
LEX	14.97				LEX	14.95				1.0000
MOC	19.31	0.0000			MOC	19.94	0.0000			0.0009
MUM	18.38	0.0000	0.0000		MUM	16.54	0.0000	0.0000		0.0000
SMC	16.17	0.0000	0.0000	0.0000	SMC	16.15	0.0000	0.0000	0.0803	1.0000

Table 6.21: Summary of system component interactions for the contract imbalance

Differences within a protocol for different offer generation strategies are in a range of four to ten utility points and are higher in protocol 2 than in the other two protocols. While the ordering of the offer generation strategies is consistent with respect to LEX, SMC, MUM, and MOC, which achieve agreements of increasing contract imbalance – and thereby of decreasing fairness – in this order, TFT is on the first place of this ranking in protocol 1 (achieving agreements of lowest contract imbalance in this protocol) but only at the bottom of the ranking in protocol 2 and protocol 3.

The concession strategy of the software agents, on the other hand, only matters for some offer generations strategies in protocol 3, while having no significant influence on the contract imbalance in other protocols, where MOC agents cause agreements of significantly higher contract imbalance if used with a passive concession strategy and MUM agents cause agreements of significantly higher contract imbalance if used with an active concession strategy ($p < 0.001$ for both software agents).

The lower contract imbalance in protocol 2 and protocol 3 results from the possibility to interrupt offering these protocols provide by enabling the software agents to send quit and reject messages, respectively, which avoids unfavorable and therefore imbalanced agreements. The variance in the ranking of TFT compared to the other software agents can be explained by its quite stable contract imbalance – achieved by reciprocating concession of the opponent – over all protocols. Contract balance is 23.05 points in protocol 1, 27.28 points in protocol 2, and 25.15 points in protocol 3 for TFT and therefore varies much less between protocol 1 and the other protocols than observed for the remaining software agents, as can be derived from Table 6.21 and Tables C.17 to C.19 in Appendix C. So TFT performs best with this stable contract imbalance in protocol 1 concerning this outcome measure, but on the other hand worst in the other protocols where this protocol-stable result is below that observed for LEX, SMC, MUM, and MOC.

6.5 Sensitivity analysis

The factors and factor level combinations discussed in the previous sections of this chapter – i.e. the different interaction protocols and the software agents, consisting of different offer generation and concession strategies – are components of the automated negotiation system, and as such under the control of the user or the system designer – at least partly in case of the interaction protocol, which has to be chosen jointly by both negotiation parties. To perform simulation runs and analyze the output of the automated negotiation systems for our six outcome measures we used the negotiation problems of human negotiation experiments as input to these systems. As the same negotiation problems were used as input for all 243 treatments this factor was held constant in the analysis of the components and therefore the basis for comparison was the same in all comparisons. However, it is of interest how the systems performance depends on different negotiation problems, i.e. on the uncontrollable input (Kleijnen, 1995).

We perform sensitivity analyses to address this question by means of plotting and inspection (Kleijnen, 1995) of the results in the outcome measures achieved by different automated negotiation system components in different negotiation problems – see Figures 6.3 to 6.8. For different system components – as columns in the figures – we plot the integrativeness of the 2065 negotiation problem (see Chapter 5) on the x-axes in all plots and the value achieved for the six outcome measures – averaged over all simulation runs with this negotiation problem and system component – each on one y-axis of the six plots. Moreover an ordinary least squares fitted regression line is included in the smoothed scatter plots to indicate the general direction of the relationship between input and output for the different system components.

As can be seen from Figure 6.3, where the relation between the performance in the six outcome measures and the integrativeness of the negotiation problems is illustrated for the three interaction protocols, simulation output is sensitive to the input to the automated negotiation system, and the directions of these relations differ for some outcome measures and interaction protocols.

While `protocol 1` is totally insensitive to the negotiation problem concerning the proportion of agreements – as this protocol forces agreements in all settings and therefore also for all negotiation problems irrespective of their integrativeness – in `protocol 2` and `protocol 3` the proportion of agreements increases with the integrativeness of the negotiation problems. Furthermore this positive relation is much stronger for `protocol 3` than for `protocol 2`. The same positive relation can be observed for the proportion of Pareto-optimal agreements – which is not further surprising as they are a proportion of the agreements – while the integrativeness of the negotiation problems has merely no influence on the proportion of Pareto-optimal agreements in `protocol 1`, this proportion increases with more integrative negotiation problems in `protocol 2` and `protocol 3`. Again in `protocol 3` the measures are stronger positively correlated than in `protocol 2`. These results indicate that it is easier to find agreements in automated negotiation when there exist more possible agreements in the negotiation problem that allow for mutual benefit over purely distributive outcomes.

Interestingly the minimal distance to the Pareto frontier in simulations with `protocol 1` start at a high level (of about 10 utility points) compared to the other protocols and then decreases with more integrative negotiation problems, while the values for this outcome measure slightly increase with the integrativeness in `protocol 2` and `protocol 3`. This indicates that the latter protocols achieve agreements close the Pareto frontier no matter how integrative the negotiation problem is, while `protocol 1` depends on integrative negotiation problems to achieve results close to the Pareto frontier. This result can be explained by the mechanisms of the protocols. Note that this outcome measure, in contrast to the proportion of Pareto-optimal agreements, bases only on the simulation runs that actually reached an agreement. Given this, `protocol 1` provides no means to circumvent inferior agreements as it does not enable the software agents to send `quit` or `reject` messages and thereby forces automated negotiations to reach agreements even if the negotiation problem only allows for inferior agreements. As this messages are available for software agents in `protocol 2` and `protocol 3`, automated negotiations in negotiation problems that only allow for inferior agreements can be broken of and such agreements can be avoided consequently.

The utility of an agreement to the seller and the buyer increases with the integrativeness of the negotiation problem in all three protocols, which is intuitive as it becomes easier to find agreements of higher utility if more such solutions exist that afford higher utility than distributive solutions. Given this increase in utility of an agreement for both sides with more integrative negotiation problems one might assume contract imbalance to be quite similar regardless of the negotiation problem. However, the interaction protocols differ considerably concerning the fairness of the agreements they can reach for negotiation problems of different integrativeness. While `protocol 1` achieves fairer outcomes with higher integrativeness of the negotiation problem – however on a relatively higher level of contract imbalance compared to the other protocols –, there is no such influence of integrativeness in `protocol 2`, and contract imbalance (unfairness) even increases with the integrativeness of negotiation problems in `protocol 3`. This indicates that the possibility to reject unfavorable offers of the opponent (without immediately breaking off the negotiation) in `protocol 3` leads to more (Pareto-optimal) agreements when integrativeness is higher but these agreements are unbalanced and favor one side. In `protocol 2` on the other hand the immediate break off of negotiations if the offering is to be interrupted rules out many agreements in advance so that only the negotiation problems that allow for balanced agreements remain. Finally in `protocol 1`, where agreements are mandatory, parties handicapped concerning their preferences have no means to avoid unbalanced agreements, but inferior preferences over the negotiation problem compared to the other party are not likely to be present if integrativeness is high and therefore the contract imbalance reduces with more integrative negotiation problems.

As can be seen from Figures 6.4 to 6.8, where the performance of different software agents (in all protocols) in the six outcome measures is illustrated for different values of the integrativeness of the negotiation problem, the integrativeness also influences the performance of the software agents, and strengths and directions of the relationships differ in some cases. Integrativeness of negotiation problems favors the proportion of agreements regardless of the software agent used to represent a focal party, which is the seller for these analyses, in the automated negotiation system. This positive correlation is slightly stronger for the actively conceding versions of the software agents than for the passively conceding ones.[8]

[8]Note that all observations start at 33% reached agreements as lower bound due to `protocol 1` which forces agreements in all simulation runs.

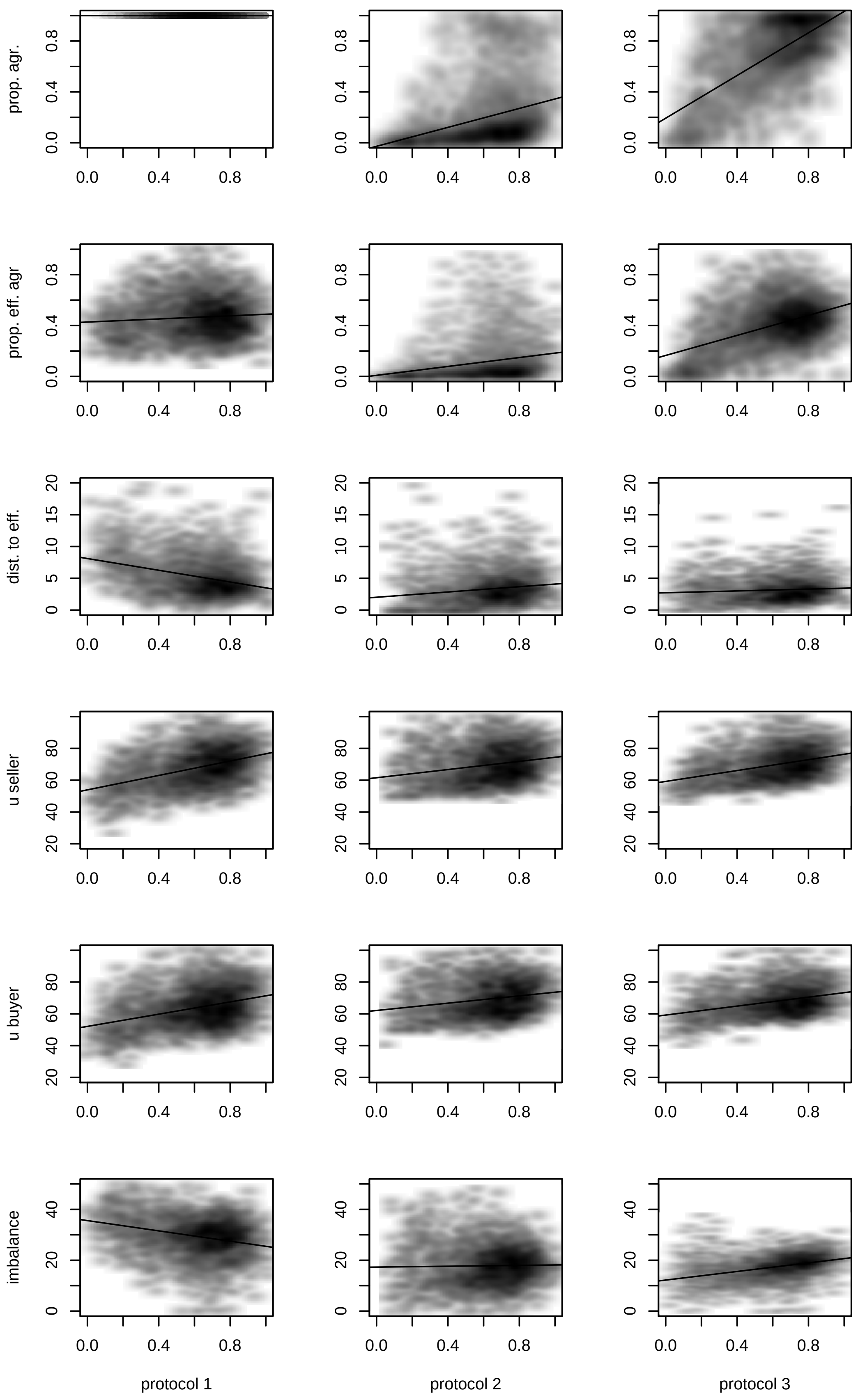


Figure 6.3: Sensitivity analysis – protocols

More integrative negotiation problems also go along with higher proportions of Pareto-optimal agreements and lower distances to the Pareto frontier for all agents. No differences can be identified here between actively and passively conceding versions of software agents following the same offer generation strategy, but the offer generation strategies have an impact on the strength – not the direction – of this correlation. While the relations are quite strong for `MOC`, `SMC`, and `MUM` offer generation – which as mentioned generate offers more systematically – it is weaker for `TFT` and especially `LEX` – which bases offer generation on a lexicographic ordering of possible solutions.

Just like for the interaction protocols, also for the different software agents representing the seller as focal party in the automated negotiations higher integrativeness results in agreements of higher utility to both the focal party (seller) and the opponent (buyer). The strength of these relationships again differs for the software agents in negotiation problems of varying integrativeness, especially due to the offer generation strategy of the software agents, while the concession strategy causes only marginal differences between the performance of software agents following the same offer generation strategy in their decision making. Not surprisingly strategies that reach agreements of higher utility for the party they represent, even in negotiation problems with low levels of integrativeness, only show a weak positive relationship to integrativeness and therefore the utility only slowly increases with more integrative negotiation problems (`MOC` and `LEX`), the agreements achieved by these software agents simultaneously provides lower utility to the opponent, which is stronger positively correlated to the integrativeness of the negotiation problem – especially for `MOC` as can be seen from Figure 6.5. On the other hand strategies that achieve agreements of low utility in less integrative negotiation problems like `MUM` or `SMC` show a stronger positive correlation to integrativeness, achieve better agreements from the perspective of the opponent even for lower levels of integrativeness of the negotiation problems, which then only increases slowly with more integrative problems. The combination of these effects leads to a reduction of contract imbalance with higher integrativeness of the negotiation problems.

Only `TFT` achieves results remarkably different from the tendencies found for the other software agents. Starting with medium level of own utility of agreements and about the same level of opponent utility, the own utility is merely independent of the integrativeness of the negotiation problem, while the opponent's utility increases with it and therefore also the contract imbalance increases with the integrativeness of the negotiation problem, in favor of the opponent i.e. the `TFT`-agent achieves the same utility regardless of the integrativeness of the negotiation problem, while the opponent does better. These results can be explained with the offer generation of `TFT`, which generates offers that fully reciprocate the opponents previous concessions – while all other offer generation mechanisms base offer generation on some systematical ordering of the possible solutions –, which seems to be too generous in integrative negotiation problems. As discussed the source of integrativeness is the compatibility of the preferences of the parties – due to different weights for issues, issues with the same best option, non-monotonic or concave partial utility functions, etc. In such integrative negotiation problems the perceived concession – measured in own utility – is likely to be greater than the actual concession – measured in the opponents utility – note that these two concession magnitudes coincide in purely distributive negotiation problems – and fully reciprocating this perceived concession therefore favors the opponent.

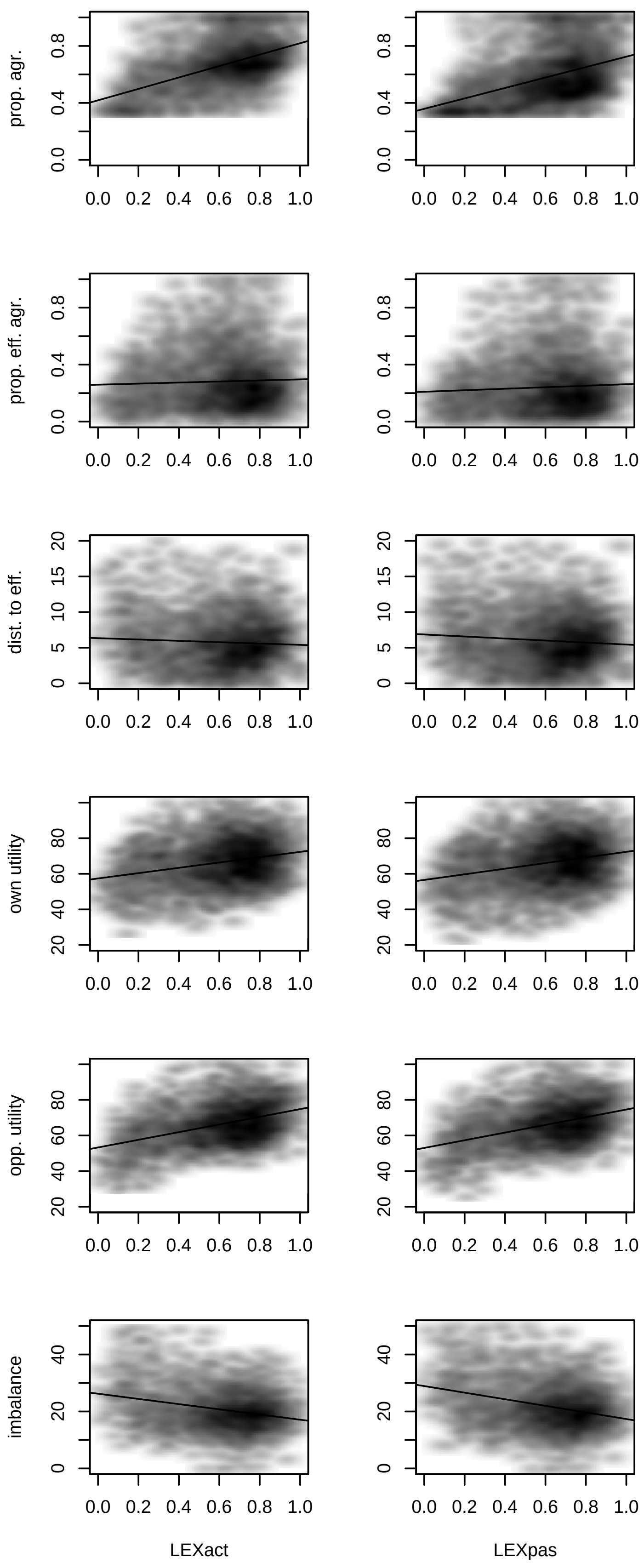


Figure 6.4: Sensitivity analysis – LEX agents

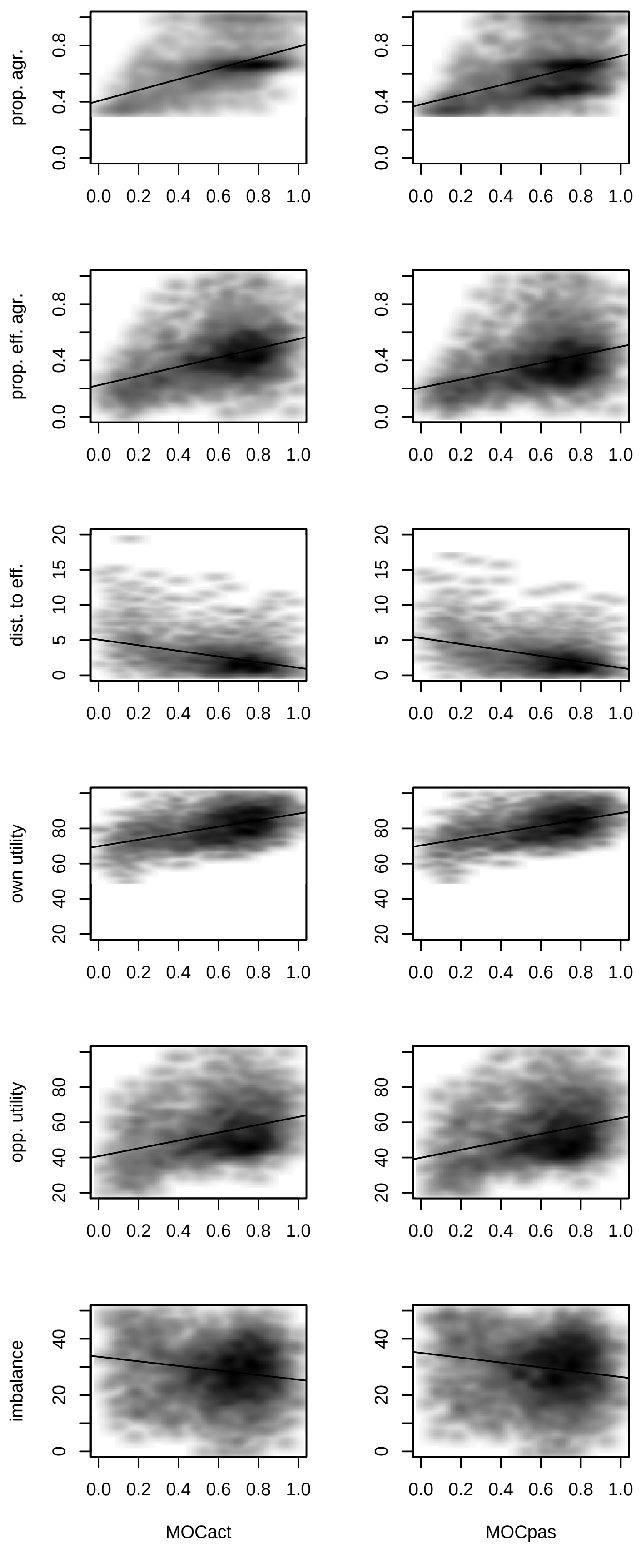


Figure 6.5: Sensitivity analysis – `MOC` agents

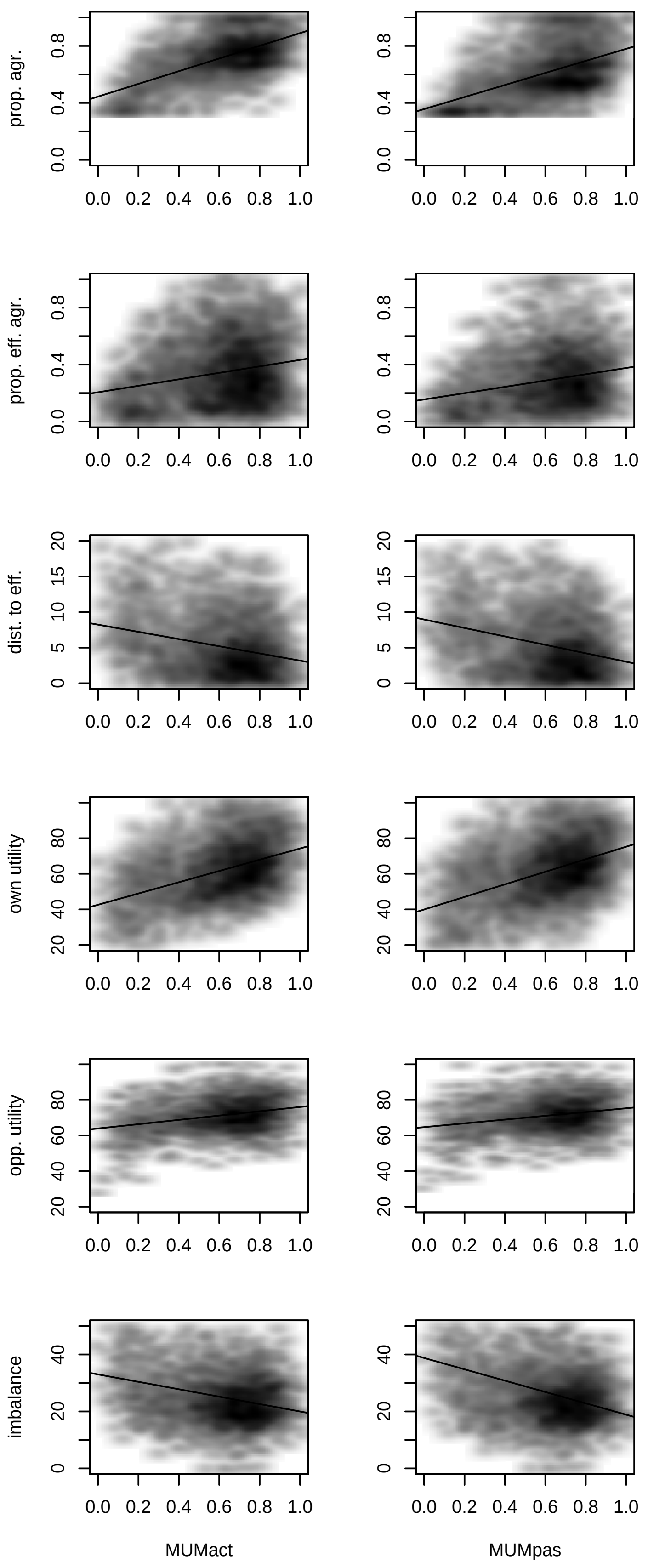


Figure 6.6: Sensitivity analysis – `MUM` agents

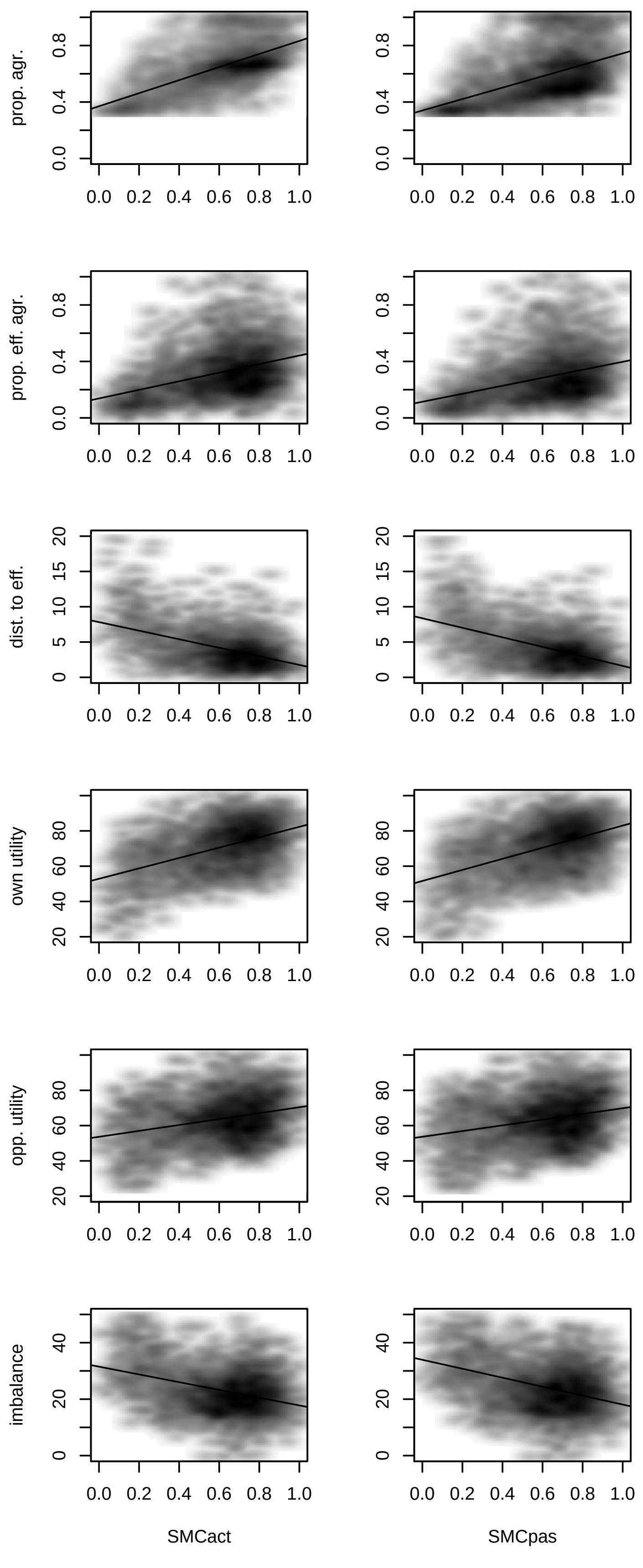


Figure 6.7: Sensitivity analysis – SMC agents

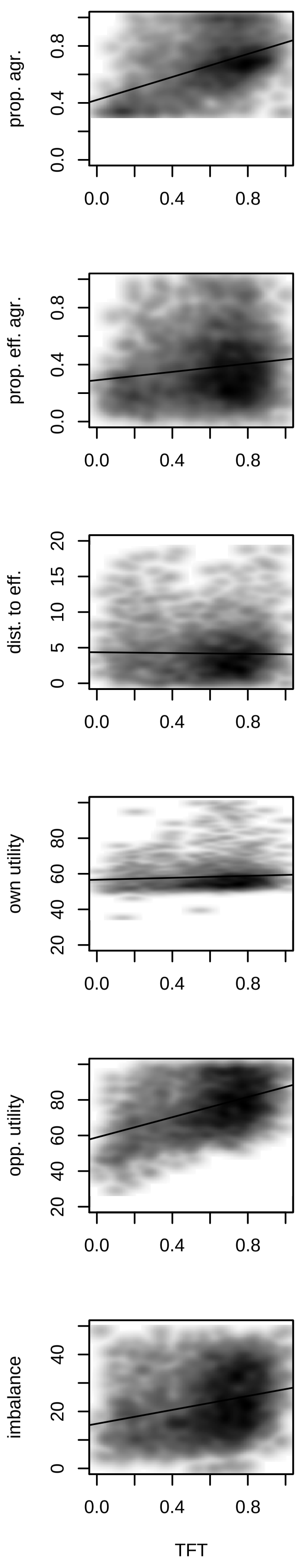


Figure 6.8: Sensitivity analysis – `TFT` agent

6.6 Discussion

This section discusses the outcomes of our simulation study presented in the previous sections of this chapter with respect to our research questions: (i) the comparison of automated negotiation systems to human negotiation, and (ii) the comparison of automated negotiation systems among each other, both for various outcome measures covering different aspects of the negotiation outcome. Furthermore we address interesting results of the simulation study in more detail, especially the 'role-dependent' performance of automated negotiation found in Sections 6.1 and 6.2.

6.6.1 Comparison to the benchmark

As mentioned in the introduction to this dissertation automated negotiation will only be employed if there are benefits of doing so compared to the currently used transaction mechanism (Blecherman, 1999). While for other applications other benchmarks are appropriate, the basis for our simulation study are the negotiation problems – resulting from the preferences of the parties over the negotiation object – elicited from human subjects in negotiation experiments. Therefore, the results of these negotiation experiments build the benchmark for evaluating the automated negotiation systems proposed and simulated in this study. The outcome measures for which we do this comparison cover various aspects of the negotiation outcome, as different of these aspects can be of importance to different negotiators and it is not clear in advance which of these measures is more important. These outcome measures are the proportion of agreements reached, the proportion of Pareto-optimal agreements reached, and in case an agreement is reached: the minimal distance of the agreement to the Pareto frontier, the individual utility of the agreement to the parties, and the contract imbalance of the agreement as a measure of fairness.

We have seen in Section 6.1 that it is not easy at all for automated negotiation – at least with the systems proposed in this study – to outperform human negotiation, though this is a common and often implicit assumption of many researchers in the field of automated negotiation as discussed in the introduction. Of our 243 automated negotiation systems – or treatments – which result from all possible combinations of the three interaction protocols and the nine software agents – embodying different offer generation and concession strategies – for the two parties, and for identical negotiation problems, only 130 (53.50% of all treatments) reached a higher proportion of agreements than humans did in experiments. 90 automated negotiation systems (37.04%) reached a higher proportion of Pareto-optimal agreements, and the minimal distance to the Pareto frontier was smaller in 79 (32.51%) treatments. Furthermore the utility of the agreement to the seller was higher than the utility sellers achieved in human negotiations for 154 (63.37%) automated negotiation systems, but higher for the buyer only with 127 (52.26%) system configurations. Contract imbalance was smaller in automated negotiation – and therefore fairness larger – in 112 (46.09%) of the 243 treatments.

While a good share of the systems achieved better results than humans in negotiation experiments in one or several outcome dimensions, only three automated negotiation systems were better in all outcome dimensions (at $p < 0.05$ – see Table C.1 in Appendix C). These three systems all consisted exclusively of software agents, for representing both parties in the negotiation, that follow monotonic concession making – proposed by Kelley (1966) – in their offer generation (`MOC`) i.e. they lower the demanded utility level to the next lower one only if all offers of this utility level were already proposed and not accepted. Furthermore, at least one of the software agents has to be of type `MOCact`, which means it has to follow an active concession strategy and therefore make first concession steps if the opponent reciprocated past concessions. Finally, the interaction protocol in all three systems was `protocol 3`, which enables the software agents to reject offers of the opponent and thereby elicit a new offer to avoid exploitation or unfavorable agreements, however, simultaneously risking a termination of the negotiation if the opponent also send a `reject` message.

So these are very specific requirements for an automated negotiation system if it is to outperform human negotiation in all six outcome measures considered in this study. However, they are understandable, the negotiation process resulting from such a system configuration features systematic offering of all possible solutions with decreasing utility to the party making the offer and rejecting offers if they are considered to provide a too low concession. This rejection, however, does not cause immediate break off of the negotiation but the software agent following the active concession strategy is likely to propose a new offer that might build the basis for continuing the negotiation, which increases the prospects of reaching an agreement. As utility levels are only reduced marginally – if at all – between offers, reached agreements are likely to be Pareto-optimal or at least very close to the Pareto frontier, be of high utility to both parties and therefore also have low contract imbalance. As mentioned in Section 6.1 the consequence of this negotiation process, induced by the three automated negotiation systems, are up to 10% (30%) more (Pareto-optimal) agreements, that afford about 5 more utility points to both parties, are 5 points closer to the Pareto frontier on average, and have a twelve points lower contract imbalance compared to human negotiation.

While the other systems did not achieve better results than humans in all outcome measures, most of them at least managed to achieve better results in some measures, however at the cost of being inferior in others. Table C.1 in Appendix C reveals some consistent patterns of these trade-offs between different outcome dimensions, which are the topic of the subsequent section.

6.6.2 Design trade-offs for different outcome dimensions

Trade-offs between different outcome dimensions in negotiations are well documented by empirical studies for traditional – face-to-face or electronically mediated – negotiations between humans, e.g. concerning the effectiveness of negotiations – i.e. the probability to reach an agreement – and the efficiency of agreements – i.e. the probability that reached agreements are Pareto-optimal. Aspects increasing the prospects of a favorable agreement – if an agreement is reached –, like for instance a though approach to negotiation, high opening offers, or small concession rates, simultaneously decrease the prospects to reach an agreement at all (e.g. Pruitt, 1981; Zartmann, 2002). We call this fundamental trade-off between outcome dimensions in negotiations – as sometimes done in literature – the negotiation dilemma.

After having presented and discussed outcomes for the different components of automated negotiation systems and the interactions of these components in sections 6.2 to 6.5 this section changes the perspective and primarily focuses on the outcome measures. Given the prevalence of the negotiation dilemma in traditional negotiations, it is obvious to assume it also for automated negotiation, and not much surprising that we also found design trade-offs concerning the configuration of an automated negotiation system with respect to different outcome dimensions in negotiations.

In general – as can be seen from Section 6.4 where we compared the interaction of components and the effect sizes – components of the automated negotiation system influencing the proportion of agreements – and thereby also the proportion of Pareto-optimal agreements as they are a share of the total agreements –, which were above all the interaction protocol and the concession strategy, only have minor or no influence on the quality of the agreement reached, measured on various dimensions – minimal distance to the Pareto frontier, individual utilities of the agreement, and contract imbalance –,, which is mainly influenced by the offer generation strategies the software agents follow.

For a high proportion of agreements the configuration of the automated negotiation system should consist of `protocol 1`, which neither allows `quit` nor `reject` messages of the software agents and therefore results in an agreement in each simulation run irrespective of the software agents applied. If protocols other than `protocol 1` are used, however, it is possible that automated negotiations end without agreement, if a `quit` message is sent by one software agents in `protocol 2`, or if two subsequent messages of the software agents were of the type `reject` in `protocol 3`, which negatively impacts the proportion of agreements reached especially for `protocol 2`. In this case the software agents should follow active concession making strategies, making first concession steps if the opponent reciprocated previous ones, which obviously increases the prospects of reaching an agreement when this is not mandatory. The offer generation strategies were found to be of minor influence on the proportion of agreements only, but last-cost-issue concession `MUM` and full reciprocation of perceived concessions of the opponent `TFT`, both making more generous offers to the opponent, demonstrated to reach more agreements than the other offer generation mechanisms, which explore the set of possible agreements more systematically and resist in making too large concessions.

Given an agreement is reached the quality of this agreement can be studied for various aspects of the outcome. The components of the automated negotiation system influencing the quality of an agreement, however, are quite different from those influencing the existence of an agreement. For the minimal distance to the Pareto frontier, the individual utility of the agreement to the parties, and contract imbalance it was found that the offer generation strategy of the software agents has major influence, followed by the interaction protocol, and only little effects were found for concession strategies of the software agents. Moreover, within these components also the options to chose for achieving high-quality agreements considerably differ from those to be chosen for a high proportion of agreements. These two observations imply that just like for traditional negotiation also for the configuration and design of automated negotiation systems major trade-offs between outcome aspects of the negotiation – i.e. the negotiation dilemma – exist.

`protocol 2`, which accounts for the lowest proportion of agreements, achieves agreements closest to the Pareto frontier, of highest utility to the parties, and lowest contract imbalance. In these outcome measures it is followed by protocol 3, which achieves nearly the same results, which indicates that the possibility to interrupt the offering sequence, which is possible in these protocols by sending `quit` and `reject` messages, increases the quality of agreements reached in these outcome measures. However at the cost of less agreements, especially for `protocol 2`, where the rigid interruption rule causes a break off of negotiations often and leads to agreements only in about 20% of the simulation runs. `protocol 1`, on the other hand, achieves agreements in all simulation runs but of inferior quality in these outcome measures. The concession strategy of the software agents has not so much influence on the quality of the agreement concerning its Pareto optimality, utility to the parties, and contract balance. In most of the cases where it was found to have significant influence – albeit the lower effect sizes – passively conceding software agents achieve outcomes of higher quality than actively conceding software agents, while it is opposite for the proportion of reached agreements.

Finally concerning the offer generation strategies, we already mentioned, that they have the major influence on the quality of the agreement. For the minimal distance to the Pareto frontier the offer generation strategies that propose offers in a more systematic fashion (`MOC` and `SMC`) achieve better performance than those found to reach a high proportion of agreements (`MUM` and `TFT`). Furthermore, concerning the other quality measures of an agreement – individual utility to the parties and contract imbalance – additional trade-offs emerge. In case an agreement is reached it is higher for the focal party (lower for the opponent party) if the focal party is represented by – in this order – `MOC`, `SMC`, `LEX`, `MUM`, and finally `TFT`. As the ordering is exactly inverse for the opponent's utility of an agreement a major trade-off can be identified for the individual utilities of the parties. Moreover, as those offer generation strategy lying in the middle of these two rankings (`LEX` and `SMC`) of agents for agreements of high utility to the own party and the opponent, achieve agreements of lowest contract imbalance the optimal software agent to use if fairness is of importance also differs from those to use if the party's utility of an agreement is important (`MOC`) or if the opponent's utility of an agreement is important (`TFT`).

Summarizing, no system configuration is superior in all aspects of the outcome of negotiations, but the optimal system configuration has to be determined according to the intentions of the users and the purpose of the system. However, if we are to suggest a system configuration in general, we opt for systems consisting of `protocol 3` and `MUM` agents – `MUMact` if reaching an agreement is critical or `MUMpas` if the quality of agreements reached is of higher importance. Though such a system not achieves best performance in all outcome measures, in our opinion the lower proportion of agreements is outweighed by the higher quality of the agreements, and at least in comparison to human negotiation these systems achieve higher performance in all outcome dimensions as discussed in the previous section.

6.6.3 Role 'dependence'

In Section 6.1 and 6.2 we found that the individual utility of an agreement is higher for the seller party than for the buyer party. In the comparison of the performance of different automated negotiation system configurations (treatments) to the results of the negotiation experiments between human subjects (Section 6.1) 154 treatments (63.37% of all 243 treatments) achieved higher individual utility for the seller side, while only 127 treatments (52.26%) did so for the buyer side. This is interesting as the set of treatments is symmetric for the two parties – i.e. besides the 27 systems where the same type of software agent is used for both parties, for each system where the seller is represented by a software agent of type A and the buyer by a software agent of type B, there exists another system in the set of treatments where the seller uses agent

B and the buyer agent A. Furthermore, even in the setting where the same software agents are used to represent the buyer and the seller side – with exception of combinations of `TFT` agents – the same software agent achieves higher individual utility, in case of an agreement, for the seller than for the buyer on average. Which is surprising as the decision making of the software agents is by no means influenced by the party they represent, but only by the preferences of the party over the negotiation object and the actions of the opponent – as can be seen from the discussion in Chapter 4 and the source code provided Appendix A. As the components of the automated negotiation system do not distinguish between roles, the only source of the differences in the individual utility to the parties can be the negotiation problem, i.e. the seller side in general seems to have preferences over the negotiation object that allow for more beneficial agreements to them than the buyer side has, which leads in the combination of these preferences to negotiation problems that favor sellers.

The seemingly 'role dependent' performance of the software agents with respect to the individual utility of agreements for the party they represent is even more disturbing when looking at the performance of the parties in the negotiation experiments between human subjects, where one cannot observe such a difference in individual utilities of the agreement. In the negotiation experiments the average utility of agreements to the sellers is 67.93 and that to the buyers 67.42, without significant difference between the parties according to a Wilcoxon rank sum test ($W = 1051174$ and $p = 0.5624$ for $n = 1441$). However, note that these results are in contradiction to previous results of negotiation experiments with students as subjects, where normally the seller side performs worse than the buyer side (e.g. Bazerman et al., 1985). It is argued that the comparative inferior performance of sellers in such negotiation experiments results for the familiarity of students of being in the position of a buyer, while students often have no negotiation experience in the role of a seller. Though the majority of the subjects in the negotiation experiments on the 'Itex-Cypress' case conducted with the negotiation support system `Inspire` were students, in these experiments the seller side did not perform worse than the buyer side, but no significant differences are found for the average utility of agreements to the two parties. An explanation of the source of this difference of the 'Itex-Cypress' experiments, which build the basis for the simulations in this dissertation, to other negotiation experiments is the way preferences are determined in the experiments, which differs between negotiation experiments with `Inspire` and the majority of the experimental designs of other studies.

In most negotiation experiments the preferences of the parties are imposed to the subjects in form of point schemes, indicating the partial utilities of the available options in the issues under negotiation. These point schemes typically are symmetrical for the parties, unless differences in the preferences of the parties are under investigation, to reduce outcome variance, establish the negotiation problem of interest, and make it easier to compare the individual performance of parties in one negotiation experiment and across different negotiation experiments (Croson, 2005). For such experiments it is found that student subjects perform worse as sellers than as buyers. However, in negotiation experiments conducted with `Inspire` the preferences are not imposed to the subjects but elicited from the subjects by the negotiation support system, so they need not be symmetric for the parties. And actually they are not symmetric as the average utility over all 180 possible solutions for the 'Itex-Cypress' case was significantly higher for the seller side (55.58) than for the buyer side (50.78) – according to a Wilcoxon rank sum test $W = 287633$ and $p < 0.0001$ for $n = 2065$. Also the standard deviation of the utility of possible solutions was significantly smaller according to the preferences of the sellers (21.51) than it was the case for the buyers (21.96), which indicates that the utility values were more concentrated around this higher average utility for the seller, while they were more spread for the buyers – according to a Wilcoxon rank sum test with $W = 1971819$ and $p < 0.0001$ for $n = 2065$. The worse performance of student subjects representing the seller side in the negotiation experiments on the 'Itex-Cypress' case is mediated by their systematically better preferences.

This discussion also explains the seemingly 'role-dependent' performance of the software agents concerning the individual utility of an agreement to the parties. Actually performance does not depend on the party represented in the negotiation, but only depends on the preferences of the represented party over the negotiation object – and the behavior of the opponent software agents –, which favored better outcomes for the seller side than for the buyer side in the negotiation problems used as input to the simulation of automated negotiation systems, and therefore also better individual performance for the seller side was reached in automated negotiation.

Summarizing this discussion, we find, in line with Blecherman (1999), that a more powerful, experienced, or creative negotiator – in our setting the buyer side in general, where the subjects have more experience in negotiations in this position – is worse off in automated negotiation compared to traditional negotiation between humans, while the less powerful, experienced, or creative negotiator is better off. Furthermore software agents and the outcomes of automated negotiations are sensitive to the underlying preferences of the parties and the resulting negotiation problem – as also can derived from the sensitivity analysis in Section 6.5 –, which is actually desirable. It also indicates that there will be resistance to the use of automated negotiation instead of traditional negotiations between humans if there exist differences in the parties power, negotiation experience, creativity, etc. – which is likely to be the case – as the party in the better position cannot exploit its beneficial situation, unless automated negotiation can compensate for this loss by lower transaction costs or even better outcomes – which is possible as discussed in Section 6.6.1. This, together with the possibility to determine preferences directly and objectively from the context and domain of the negotiation – and not through time-consuming elicitation from the human user – maybe also renders coordination of agents in autonomous systems the more appealing area of application of automated negotiation than electronic business.

Chapter 7

Conclusion

The motivation of this dissertation are the prospects of the use of automated negotiation for electronic business or coordination of software agents in autonomous systems. Automated negotiation is argued to achieve better outcomes than negotiation between humans, at lower transaction cost, and enabling higher volumes and new sorts of transactions in electronic business. Through its automation the mechanism of negotiation becomes available to autonomous systems, thereby improving the performance of these systems when negotiation is used for agent coordination and cooperation instead of existing often simplistic and rigid interaction mechanisms.

However, currently no operative systems for automated negotiation are implemented and system designers rely – besides the use of analytical models for very specific problems – on simulation studies to propose and evaluate configurations for automated negotiation systems. We systematically reviewed the state of the art in simulation of automated negotiation – along its main components: (i) negotiation problem, (ii) interaction protocol, and (iii) software agent strategies – based on studies identified by a keyword search in scientific databases. This review revealed deficiencies of existing approaches concerning their implementation in operative systems and practical applicability for all components of automated negotiation. The central research question of this dissertation therefore was to propose, simulate, and evaluate different system configurations for automated negotiation systems in a first attempt to address the identified deficiencies of existing approaches.

Concerning the negotiation problems used in the reviewed studies we argued that they often fell short in representing the potential complexity of real negotiation problems by assuming simplistic preference functions and in most cases considering only one negotiation problem to evaluate different software agent strategies. Addressing these concerns, we used the preferences elicited from subjects in negotiation experiments as input for the software agents in our study. The multiple-issue negotiation object for which the preferences were elicited and the utility functions resulting from this elicitation procedure combined to a variety of realistically complex negotiation problems as basis for our study. The use of the negotiation problems from negotiation experiment between human subjects also enabled the comparison of the results of the experiments to the results of the simulation and thereby to test the implicit assumption that automated negotiation can outperform human negotiation – a *sine qua non* for the actual application of automated negotiation in practice.

As the interaction protocol of an automated negotiation system not only builds the basis for but also restricts the interactions of software agents, our concern was that current studies merely only use different versions of one single interaction protocol, that restrictively forces the software agents to alternately propose offers, often without a possibility to endogenously decide to break off the negotiation, but the termination of the negotiation is determined exogenously in form of deadlines or reservation levels the software agents have to respect. We therefore focused on protocols – derived from game theoretic literature on mechanism design – that enable the temporary or permanent interruption of the strategy the software agent otherwise would follow, which allows software agents to circumvent exploitation or unfavorable agreements.

Concerning the software agents currently used in simulations of automated negotiations, we argued that these are not readily applicable to actual automated negotiations – i.e. for an implementation in operative systems – as they ignore the openness of the media over which automated negotiations are conducted as well as the low importance of time in automated negotiation due to its fast proceeding. Time-based concession functions are not suitable due to the time insensitivity of automated negotiation. Furthermore, the possible variety and complexity of negotiation problems and opponent strategies in automated negotiation causes problems for software agent strategies basing on evolutionary computing or learning mechanisms. On the one hand the many trials against the same opponents for the same negotiation problems, necessary for a good performance of software agents developed by means of evolutionary computing, are likely not possible. On the other hand the models of the opponent, learning agents hold in mind and update according to the course of the negotiation to economize on learned parameters of these models, likely are not able to satisfactory cover the variety of (novel) opponent agents. Consequently we focused on the class of continuous concession strategies and implemented rule-based rather deterministic algorithms proposed in negotiation literature but not used in simulations yet. These strategies neither make offers dependent on time, nor model their opponent or try to learn something about the opponent's preferences or strategy, but base decision making in negotiation solely on the preferences of the party they represent and the behavior of the opponent and are therefore (re)usable for various negotiation problems with changing opponents. The proposed software agents in general consist of two parts the offer generation strategy, which determines the next offer to propose, and the concession strategy, which given the history of the negotiation, determines whether to propose the generated offer or to interrupt the offering strategy if this is enabled by the protocol.

We used a tournament-based simulation approach and let all combinations of software agents in all protocols – which constituted the 243 system configurations or treatments of our study – negotiate in three replications – to account for stochastic influences on the simulation outcome – of all negotiation problems elicited from the subjects of the negotiation experiments, which resulted in a full factorial design. The protocol and the two software agents representing the two parties together with the negotiation problem as input to the system fully parameterize a simulation run. The output of the simulation then was analyzed for several outcome measures to cover various aspects of the negotiation outcome, which were the proportion of (Pareto-optimal) agreements, and for those simulation runs that reached an agreement, additionally as measures of the quality of the outcome, the minimal distance to the Pareto frontier, the individual utility of the agreement to the parties, and the contract imbalance as a measure of fairness of the agreement.

The evaluation of the results of the simulations indicated that the negotiation dilemma found in traditional negotiation – i.e that aspects favoring an agreement negatively effect the quality of the agreement – also could be observed for the components of automated negotiation systems. The interaction protocol and the concession strategies of the agents which had been found to have high influence on the proportion of agreements reached, only had low influence on various quality measures of the outcome, but here the offer generation strategy of the software agents mattered more. Additionally the different options available in the components in general oppositely influenced the proportion of agreements and the quality of reached agreements. Given these fundamental trade-offs between the different outcome dimensions, it is not surprising that most system configurations only managed to outperform the benchmark – the result of the negotiation experiments with human subjects – only for some outcome measures but were inferior in others. Only a set of systems, consisting of an interaction protocol that enables to reject unfavorable offers and software agents that systematically propose offers of monotonically decreasing utility representing the parties in the negotiation – at least one of them making first concession steps if the opponent reciprocated previous concessions –, though they were not the best in all outcome measures, performed comparably well and managed to significantly outperform human negotiations in all outcome measures.

After this summary of the dissertation's results the remainder of this section deals with future research on (simulation of) automated negotiation, which necessarily has to be address before the insights from simulation studies on automated negotiation can find their way in implementations of operative systems. In our opinions such future research could cover the limitations of this study, further develop the components proposed, or proceed to the next steps on a research agenda for automated negotiations.

A first limitation to the generalizability of the results of this dissertation results from the input used for the simulations. Though the elicited preferences combined to a variety of different negotiation problems all negotiation problems deal with the identical negotiation object of the 'Itex-Cypress' negotiation case. The structure of the negotiation object – i.e. the number of issues under negotiation and possible options for settlement within the issues –, however, might influence the performance of the software agents. Therefore it is necessary to investigate the existence, strength, and direction of effects of different negotiation objects on the performance of automated negotiation systems. A further limitation of the validity of the results emerges from the fact that the majority of the subjects in the negotiation experiments were students. Though students are often used in negotiation experiments and such experiments are argued to be a good proxy to actual negotiations, it would be interesting to compare the results of automated negotiations to outcomes of experiments with negotiation practitioners or even the outcomes of actual negotiations for given negotiation problems. Finally we only could compare the software agents proposed in this study among each other, but not to the software agents proposed in former simulation studies, due to the different design philosophies followed. As mentioned the majority of software agents used in previous simulation studies require a predetermined number of rounds or turns for the negotiation, which we avoided in this study. However, the number of packages of the negotiation object on the one hand, and the interaction protocol on the other hand, determine also in our design a maximal number of rounds, which could be used in future research for comparing the software agents proposed in this study to those proposed in literature.

There are many ways to further develop and refine the components of the proposed automated negotiation systems, which however are beyond the scope and purpose of this dissertation. For example, the effect of different configurations of the `TFT` agent deserves attention, where different weights for the similarity and the reciprocation criteria could be evaluated, as we only considered one configuration in the simulations. Furthermore, the decision making of the software agent could be made more sensitive to the stage of the negotiation, especially in the protocol that allows to break off negotiations, and the behavior of the opponent, especially in the protocol that allows to reject offers of the opponent. It does not make much sense to break off negotiations in an early stage of the negotiation, due to one unfavorable offer of the opponent, where the level of demanded utility is still high and therefore the risk of exploitation low. Moreover, if necessary the same result – i.e. no agreement – can be achieved in later rounds too. In the protocol allowing reject messages software agents should be cautious about the behavior of their opponents as the rejection of offers – at least in the current implementation of the software agents – reveals information about the preferences of the opponent, which can be used for own or mutual benefit in the automated negotiation. Furthermore protocols and software agents similar to those proposed in this study could be used – with only minor adaptations – for the implementation of improvement-based progress in automated negotiation, rather than the concession-based progress prevalent in current simulation studies and also used in this dissertation. The software agents in improvement-based negotiations would start at the lowest acceptable level of utility, rather than with the best offer, and then continuously demand more instead of less, but can use similar offer generation strategies. Agents accept an offer of the opponent unless it is of lower utility than the last tentative agreement, and otherwise reject the offer. The protocol terminates the negotiation if there is no progress in the negotiation, i.e. if both software agents rejected their opponent's last offer. In this case the last accepted offer is the final agreement. Such automated negotiation systems – if they reach comparable outcomes – might be easier to accept for users for psychological reasons as they do not give in but ask for more.

Especially this acceptance of automated negotiation systems by the possible users is critical for the actual application of automated negotiation in practice. We actually only considered benefits in terms of superior outcome and lower transaction costs – from an optimistic point of view, which is also present in the literature. Other sources of costs, like the loss of power – due to better information, patience, or creativity –, maybe the lost pleasure of negotiating and competitive interaction with other people, the difficulty of preference elicitation, and efforts spent on the configuration or even programming of a software agent, must not be neglected. Humans even could be more satisfied with worse outcomes, compared to those possible with automated negotiation, they reached themselves, due to the effort spent on negotiating and the feeling to have done the best they could, or if social aspects like trust or relationship between the parties are important. These aspects need to be addressed in experiments – that go beyond questions of performance but tackle questions of acceptance, satisfaction, and usage – in future research.

Though the simulation system used in this dissertation could be easily transformed into an operative system – or even used directly for automated negotiation if the negotiation object and the parties' preferences are fed into our database – as discussed there remain several issues to be addressed, in the novel but fast developing field of automated negotiation, to make the insights from simulations studies applicable for the development and implementation of operative systems.

This, not the mere simulation of automated negotiation, has to be the final purpose of our endeavor as only the implementation of operative systems will realize to the prospected benefits of automated negotiation. However, we are confident to have made valuable contributions to the field of automated negotiation, in identifying deficiencies of current studies on the simulation of automated negotiations, proposing, simulating, and evaluating a conceptual model for automated negotiation in a first step to address them, and pointing out the next steps on a research agenda for automated negotiation research – and its ultimate goal, the implementations of operative systems and their actual application for electronic business and coordination of autonomous systems in practice – with this dissertation.

Bibliography

An, B., K. M. Sim, L. G. Tang, S. Q. Li, and D. J. Cheng (2006): "Continuous-time negotiation mechanism for software agents", *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 36, pp. 1261–1272*.

Axelrod, R. (1980a): "Effective choice in the prisoner's dilemma", *Journal of Conflict Resolution*, 24(2), pp. 3–25.

Axelrod, R. (1980b): "More effective choice in the prisoner's dilemma", *Journal of Conflict Resolution*, 24(3), pp. 379–403.

Axelrod, R. (1984a): *The evolution of co-operation*, Basic Books, New York.

Axelrod, R. (1984b): "The evolution of strategies in the iterated prisoner's dilemma", in: L. Davis (ed.), *Genetic algorithms and simulated annealing*, Pitman, London, pp. 32–41.

Axelrod, R. (1997): "Advancing the art of simulation in the social science", in: R. Conte, R. Hegselmann, and P. Terno (eds.), *Simulating social phenomena*, Springer, Berlin, pp. 21–40.

Bac, M. and H. Raff (1996): "Issue-by-issue negotiations: the role of information and time preference", *Games and Economic Behavior*, 13, pp. 125–134.

Backhaus, K., B. Erichson, W. Plinke, and R. Weiber (2003): *Multivariate Analysemethoden*, 10th edition, Springer, Berlin.

Balakrishnan, P. V. and J. Eliashberg (1995): "An analytical process model of two-party negotiations", *Management Science*, 41(2), pp. 226–243.

Balci, O. (1994): "Validation, verification, and testing techniques throughout the life cycle of a simulation study", *Annals of Perations Research*, 53, pp. 121–173.

Banks, J. and J. S. Carson (1984): *Discrete-event system simulation*, Prentice-Hall, Englewood Cliffs.

Bartos, O. J. (1977): "Simple model of negotiation", *Journal of Conflict Resolution*, 21(4), pp. 565–579.

Bazerman, M. H., J. R. Curhan, D. A. Moore, and K. L. Valley (2000): "Negotiation", *Annual Review of Psychology*, 51, pp. 279–314.

Bazerman, M. H., T. Magliozzi, and M. A. Neale (1985): "Integrative bargaining in a competitive market", *Organizational Behavior and Human Decision Processes*, 35, pp. 294–313.

Bazerman, M. H. and M. A. Neale (1991): "Negotiator rationality and negotiator cognition: The interactive roles of prescriptive and descriptive research", in: H. P. Young (ed.), *Negotiation analysis*, The University of Michigan Press, Ann Arbor, pp. 109–129.

Bazerman, M. H. and M. A. Neale (1992): *Negotiating rationally*, Free Press, New York.

Beam, C. and A. Segev (1997): "Automated negotiations: A survey of the state of the art", *Wirtschaftsinformatik*, 39(3), pp. 263–267.

Beroggi, G. E. G. (2003): "Internet multiattribute group decision support in electronic commerce", *Group Decision and Negotiation*, 12, pp. 481–499.

Bichler, M. (2000): *Protocols for multi-dimensional negotiation support*, Department of Information Systems, Vienna University of Economics and Business Administration.

Bichler, M. and A. Segev (2001): "Methodologies for the design of negotiation protocols on e-markets", *Computer Networks*, 37(2), pp. 137–152.

Binmore, K. and N. Vulkan (1999): "Applying game theory to automated negotiation", *Netnomics*, 1(1), pp. 1–9.

Bishop, R. L. (1963): "Game-theoretic analyses of bargaining", *The Quarterly Journal of Economics*, 77(4), pp. 559–602.

Blecherman, B. (1999): "Adopting automated negotiation", *Technology in Society*, 21(2), pp. 167–174.

Bosse, T. and C. M. Jonker (2005): "Human vs. computer behavior in multi-issue negotiation", in: *Proceedings of the First International Workshop on Rational, Robust, and Secure Negotiation in Multi-Agent-Systems (RRS2005)**.

Bratley, P., B. L. Fox, and L. E. Scharge (1987): *A guide to simulation*, 2nd edition, Springer, New York.

Buffett, S., L. Comeau, B. Spencer, and M. W. Fleming (2006): "Detecting opponent concessions in multi-issue automated negotiation", in: *Proceedings of the ICEC06*, pp. 11–18*.

Buffett, S. and B. Spencer (2005): "Learning opponents' preferences in multi-object automated negotiation", in: *Proceedings of the ICEC05**.

Busch, L.-A. and I. J. Horstmann (1999a): "Endogenous incomplete contracts: A Bargaining Approach", *The Canadian Journal of Economics*, 32(4), pp. 956–975.

Busch, L.-A. and I. J. Horstmann (1999b): "Signalling via an agenda in multi-issue bargaining with incomplete information", *Economic Theory*, 13(3), pp. 561–575.

Byrnes, J. F. (1987): "Negotiating: Master the ethics", *Personnel Journal*, 66, pp. 97–101.

Cardoso, H. L. and E. Oliveira (2000): "Using and evaluating adaptive agents for electronic commerce negotiations", *Lecture Notes in Computer Science*, 1952/2000, pp. 96–105*.

Carnevale, P. J. and D. G. Pruitt (1992): "Negotiation and mediation", *Annual Review of Psychology*, 43, pp. 531–582.

Chao, K.-M., M. Younas, N. Godwin, and P.-C. Sun (2006): "Using automated negotiation for grid services", *International Journal of Wireless Information Networks*, 13, pp. 141–150*.

Chatterjee, K. and G. L. Lilien (1984): "Efficiency of alternative bargaining procedures", *Journal of Conflict Resolution*, 28(2), pp. 270–295.

Chavez, A. and P. Maes (1996): "Kasbah: An agent marketplace for buying and selling goods", in: *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, London, pp. 75–90.

Chen, E., R. Vahidov, and G. E. Kersten (2005a): "Agent-supported negotiations in the e-marketplace", *International Journal of Electronic Business*, 3(1), pp. 28–49.

Chen, J.-H., K.-M. Ghao, and N. Godwin (2005b): "Combining cooperative and non-cooperative automated negotiations", *Information Systems Frontiers*, 7(4/5), pp. 391–404*.

Cheng, C.-B., C.-C. H. Chan, and K.-C. Lin (2006): "Intelligent agents for e-marketplace: Negotiation with issue trade-offs by fuzzy inference systems", *Decision Support Systems*, 42, pp. 626–638*.

Choi, S. P. M., J. Liu, and S.-P. Chan (2001): "A genetic agent-based negotiation system", *Computer Networks*, 37, pp. 195–204*.

Clyman, D. R. (1995): "Measures of joint performance in dyadic mixed-motive negotiations", *Organizational Behavior and Human Decision Processes*, 64(1), pp. 38–48.

Conover, W. J., M. E. Johnson, and M. M. Johnson (1981): "A comparative study of tests for homogeneity of variances, with applications to the outer continental self bidding data", *Technometrics*, 23, pp. 351–361.

Contini, B. and S. Zionts (1968): "Restricted bargaining for organizations with multiple objectives", *Econometrica*, 36(2), pp. 397–414.

Coursey, D. (1982): "Bilateral bargaining, Pareto optimality, and the empirical frequency of impasse", *Journal of Economic Behavior and Organization*, 3(2-3), pp. 243–259.

Cranor, L. F. and P. Resnick (2000): "Protocols for automated negotiations with buyer anonymity and seller reputations", *Netnomics*, 2, pp. 1–23.

Crawford, V. P. (1982): "A theory of disagreement in bargaining", *Econometrica*, 50(3), pp. 607–637.

Croson, R. (2005): "The method of experimental economics", *International Negotiation*, 10, pp. 131–148.

Cross, C. R. (1965): "A theory of the bargaining process", *American Economic Review*, 55(1/2), pp. 67–94.

Cross, C. R. (1969): *The economics of bargaining*, Basic Books, New York.

Cross, C. R. (1977): "Negotiation as a learning process", *Journal of Conflict Resolution*, 21(4), pp. 581–606.

Cyert, R. and J. G. March (1963): *A behavioral theory of the firm*, Prentice-Hall, Englewood Cliffs.

Davis, J. P., K. M. Eisenhardt, and C. B. Bingham (2007): “Developing theory through simulation methods”, *Academy of Management Review*, 32(2), pp. 480–499.

Delaney, M. M., A. Foroughi, and W. C. Perkins (1997): “An empirical study of the efficacy of a computerized negotiation support system (NSS)”, *Decision Support Systems*, 20, pp. 185–197.

DeSanctis, G. and M. S. Poole (1994): “Capturing the complexity in advanced technology use: Adaptive structuration theory”, *Organization Science*, 5(2), pp. 121–147.

Deveaux, L., C. Paraschiv, and M. Latourrette (2001): “Bargaining on an internet agent-based market: Behavioral vs. optimizing agents”, *Electronic Commere Research*, 1, pp. 371–401*.

Druckman, D., J. N. Druckman, and T. Arai (2004): “E-mediation: Evaluating the impacts of an electronic mediator on negotiation behavior”, *Group Decision and Negotiation*, 13(6), pp. 481–511.

Druckman, D., B. Ramberg, and R. Harris (2002): “Computer-assisted international negotiation: A tool for research and practice”, *Group Decision and Negotiation*, 11(3), pp. 231–256.

Dupont, C. and G.-O. Faure (2002): “The negotiation process”, in: V. A. Kremenyuk (ed.), *International negotiation: Analysis, approaches, issues*, 2nd edition, Jossey-Bass, San Francisco, pp. 39–63.

Edgeworth, F. Y. (1881): *Mathematical psychics - An essay on the application of mathematics to the moral sciences*, C. Kegan Paul & Co, London.

Evans, J. B. (1988): *Structures of discrete event simulation*, Ellis Horwood, Chichester.

Faratin, P., C. Sierra, and N. R. Jennings (1998): “Negotiation decision functions for autonomous agents”, *Robotics and Autonomous Systems*, 24, pp. 159–182*.

Faratin, P., C. Sierra, and N. R. Jennings (2002): “Using similarity criteria to make issue trade-offs in automated negotiations”, *Artificial Intelligence*, 142(2), pp. 205–237*.

Faratin, P., C. Sierra, N. R. Jennings, and P. Buckle (1999a): “Designing flexible automated negotiators: Concessions, trade-offs and issue changes”, Technical Report RR 99-03, Institut d’Investigacio en Intelligencia Artificial Technical.

Faratin, P., C. Sierra, N. R. Jennings, and P. Buckle (1999b): “Designing responsive and deliberative automated negotiators”, in: *Proceedings of the AAAI Workshop on Negotiation: Settling Conflicts and Identifying Opportunities*, Orlando, Florida, pp. 12–18.

Fatima, S. S., M. Wooldridge, and N. R. Jennings (2004): “An agenda-based framework for multi-issue negotiation”, *Artificial Intelligence*, 152(1), pp. 1–45.

Fershtman, C. (1990): “The importance of the agenda in bargaining”, *Games and Economic Behavior*, 2, pp. 224–238.

Filzmoser, M. (2007): “Exponential smoothed Tit-for-Tat”, in: G. Kendall, X. Yao, and S. Y. Chong (eds.), *The iterated prisoner’s dilemma: 20 years on*, World Scientific, pp. 127–138.

Filzmoser, M. (2008): “Non-alternating offer protocols and concession strategies for automated negotiation”, in: J. Climaco, G. E. Kersten, and J. P. Costa (eds.), *Proceedings of the International Conference on Group Decision and Negotiation (GDN), Coimbra, Portugal*, INESC Coimbra, pp. 267–268.

Filzmoser, M. and R. Vetschera (2008): “A classification of bargaining steps and their impact on negotiation outcomes”, *Group Decsion and Negotiation*, 17, pp. 421–443.

Fink, A. (2004): “Supply chain coordination by means of automated negotiations”, in: *Proceedings of the Hawai'i International Conference on System Sciences.*

Fisher, R. (1978): *International mediation: A working guide*, International Peace Academy, New York.

Fisher, R. and W. Ury (1981): *Getting to yes*, Houghton-Mifflin, Boston.

Foroughi, A. (1998): “Minimizing negotiation process losses with computerized negotiation support systems”, *Journal of Applied Business Research*, 14(4), pp. 15–26.

Foroughi, A., W. C. Perkins, and M. T. Jelassi (1995): “An empirical study of an interactive, session-oriented computerized negotiation support system (NSS)”, *Group Decision and Negotiation*, 4(6), pp. 485–512.

Froman, L. A. and M. D. Cohen (1970): “Compromise and logroll: comparing the efficiency of two bargaining processes”, *Behavioral Science*, 15, pp. 180–183.

Gardner, M. (1970): “The fantastic combinations of John Conway's game 'Life”', *Scientific American*, 223(4), pp. 120–123.

Gerding, E. H. and H. La Poutre (2006): “Bilateral bargaining with multiple opportunities: Knowing your opponent's bargaining position”, *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Review*, 36, pp. 45–55*.

Gerding, E. H., D. D. B. van Bragt, and J. A. La Poutre (2003): “Multi-issue negotiation processes by evolutionary simulation, validation and social extensions”, *Computational Economics*, 22, pp. 39–63*.

Goh, K.-Y., H.-H. Teo, H. Wu, and K.-K. Wei (2000): “Computer-supported negotiations: An experimental study of bargaining in electronic commerce”, in: W. Orlikowski, S. Ang, P. Weill, H. Krcmar, and J. I. DeGross (eds.), *Proceedings of the 21st International Conference of Information Systems (ICIS 2000)*, Brisbane, Australia, pp. 104–116*.

Gordon, G. (1978): *System simulation*, 2nd edition, Prentice-Hall, Englewood Cliffs.

Greenhalgh, L., S. A. Neslin, and R. W. Gilkey (1985): “The effects of negotiator preferences, situational power, and negotiator personality on outcomes of business negotiations”, *Academy of Management Journal*, 28(1), pp. 9–33.

Gulliver, P. H. (1979): *Disputes and negotiations: A cross cultural perspective*, Academic Press, New York.

Guo, Y., J. P. Müller, and C. Weinhardt (2003): "Learning user preferences for multi-attribute negotiation: An evolutionary approach", in: J. P. Müller, V. Marik, and M. Pechoucek (eds.), *Multi Agent Systems and Applications III, volume 2691 of Lecture Notes in Artificial Intelligence*, Springer, Berlin, pp. 303–313.

Gupta, S. (1989): "Modeling integrative, multiple issue bargaining", *Management Science*, 35(7), pp. 788–806.

Gupta, S. and Z. A. Livne (1988): "Resolving a conflict situation with a reference outcome: An axiomatic model", *Management Science*, 34(11), pp. 1303–1314.

Güth, W., R. Schmittberger, and B. Schwarze (1982): "An experimental analysis of ultimatum bargaining", *Journal of Economic Behavior and Organization*, 3(4), pp. 367–388.

Guttman, R. H., A. G. Moukas, and P. Maes (1998): "Agent-mediated electronic commerce: A Survey", *Knowledge Engineering Review*, 13, pp. 147–159.

Hair, J. F., W. C. Black, B. J. Babin, R. E. Anderson, and R. L. Tatham (2006): *Multivariate Data Analysis*, 6th edition, Pearson Prentice Hall, New Jersey.

Harsanyi, J. C. (1956): "Approaches to the bargainig problem before and after the theory of games: A critical discussion of Zeuthen's, Hick's, and Nash's theories", *Econometrica*, 24(2), pp. 144–157.

Hausken, K. (1997): "Game-theoretic and behavioral negotiation theory", *Group Decision and Negotiation*, 6(6), pp. 511–528.

Henderson, P., S. Crouch, R. J. Walters, and Q. Ni (2003): "Comparison of some negotiation algorithms using a tournament-based approach", *Lecture Notes in Computer Science*, 2592, pp. 137–150*.

Henderson, P., S. Crouch, R. J. Walters, and Q. Ni (2005): "Effects of introducing survival behaviours into automated negotiators specified in an environmental and behavioural framework", *Journal of Systems and Software*, 76, pp. 65–76*.

Hicks, J. (1932): *The theory of wages*, Macmillan, London.

Holm, S. (1979): "A simple sequentially rejective multiple test procedure", *Scandinavian Journal of Statistics*, 6, pp. 65–70.

Inderst, R. (2000): "Multi-issue bargaining with endogenous agenda", *Games and Economic Behavior*, 30, pp. 64–82.

Jennings, N. R., P. Faratin, A. R. Lomuscio, S. Parsons, M. Wooldridge, and C. Sierra (2001): "Automated negotiation: Prospects, methods and challenges", *Group Decision and Negotiation*, 10(2), pp. 199–215.

Jonker, C. and V. Robu (2004): "Automated multi-attribute negotiation with efficient use of incomplete preference information", in: *Proceedings of the AAMAS04*, pp. 1054–1061*.

Kalai, E. and M. Smorodinsky (1975): "Other solutions to Nash's bargaining problem", *Econometrica*, 43(3), pp. 513–518.

Keeney, R. L. and H. Raiffa (1993): *Decision with multiple objectives: Preferences and value tradeoffs*, Cambridge University Press.

Kelley, H. H. (1966): “A classroom study of the dilemmas in interpersonal negotiations”, in: K. Archibald (ed.), *Strategic interaction and conflict: Original papers and discussion*, Institute of International Studies, Berkeley, pp. 49–73.

Kelton, W. D. (1999): “Designing simulation experiments”, in: P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans (eds.), *Proceedings of the 1999 Winter Simulation Conference.*

Kersten, G. E. (1997): “Support for group decision and negotiations – An overview”, in: J. Climaco (ed.), *Multicriteria analysis*, Springer, Heidelberg, pp. 332–346.

Kersten, G. E. (2007): *Negotiations and e-negotiations –Analysis, management and support*, Springer.

Kersten, G. E. and H. Lai (2007): “Satisfiability and completeness of protocols for electronic negotiations”, *European Journal of Operational Research*, 180, pp. 922–937.

Kersten, G. E. and S. Noronha (1999a): “Negotiation via the World Wide Web: A cross-cultural study of decision making”, *Group Decision and Negotiation*, 8(3), pp. 251–279.

Kersten, G. E. and S. J. Noronha (1999b): “WWW-based negotiation support: Design, implementation, and use”, *Decision Support Systems*, 25(2), pp. 135–154.

Kersten, G. E., S. J. Noronha, and J. Teich (2000): “Are all e-commerce negotiations auctions?”, in: *Fourth International Conference on the Design of Cooperative Systems, Sophia-Antipolis, France (COOP’2000).*

King, D. (1981): “Three cheers for conflict”, *Personnel*, 58, p. 21.

Kleijnen, J. P. C. (1987): *Statistical tools for simulation practitioners*, Marcel Dekker Inc., New York.

Kleijnen, J. P. C. (1995): “Verification and validation of simulation models”, *European Journal of Operational Research*, 82(1), pp. 145–162.

Kleijnen, J. P. C. and W. van Groenendaal (1992): *Simulation – A statistical perspective*, John Wiley & Sons.

Klein, M., P. Faratin, H. Syayama, and Y. Bar-Yam (2003): “Negotiating complex contracts”, *Group Decision and Negotiation*, 12(2), pp. 111–125*.

Klemperer, P. (2004): *Auctions: Theory and practice*, Princeton University Press, Princeton.

Kontolemakis, G., P. Kanellis, and D. Martakos (2004): “Software agents for electronic marketplaces: Current and future research directions”, *Journal of Information Technology Theory and Application*, 6(1), pp. 43–60.

Köszegi, S. T. (2008): “e-Nego-motion – Foundations of effective electronic negotiation support: socio-emotional, prescriptive, and technological perspectives”, FWF research project proposal.

Köszegi, S. T., K. J. Srnka, and E.-M. Pesendorfer (2006): “Electronic negotiations – A comparison of different support systems”, *Die Betriebswirtschaft*, 66(4), pp. 441–463.

Kraus, S. (1997): “Negotiation and cooperation in multi-agent environments”, *Artificial Intelligence*, 94, pp. 79–98.

Krovi, R., A. C. Graesser, and W. E. Pracht (1999): “Agent behaviors in virtual negotiation environments”, *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Review*, 29, pp. 15–25*.

Lai, G. and K. Sycara (2009): “A generic framework for automated multi-attribute negotiation”, *Group Decision and Negotiation*, 18(2), pp. 169–187.

Lang, K. and R. W. Rosenthal (2001): “Bargaining piecemeal or all at once?”, *The Economic Journal*, 111(July), pp. 526–540.

Law, A. M. (2005): “How to build valid and credible simulation models”, in: M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joins (eds.), *Proceedings of the 2005 Winter Simulation Conference*, pp. 24–32.

Law, A. M. and W. D. Kelton (1991): *Simulation modeling and analysis*, 2nd edition, McGraw-Hill, New York.

Lawley, R., K. Decker, M. Luck, T. Payne, and L. Moreau (2003a): “Automated negotiation for grid notification services”, *Lecture Notes in Computer Science*, 2790/2003, pp. 384–393*.

Lawley, R., M. Luck, K. Decker, T. Payne, and L. Moreau (2003b): “Automated negotiation between publishers and consumers of grid notifications”, *Parallel Processing Letters*, 13, pp. 537–548*.

Lee, C.-F. and P.-L. Chang (2009): “Evaluations of tactics for automated negotiations”, *Group Decision and Negotiation*, in print (DOI: 10.1007/s10726-008-9109-y).

Lee, W.-P. (2004): “Towards agent-based decision making in th electronic marketplace: interactive recommendation and automated negotiation”, *Expert Systems with Applications*, 27, pp. 665–679*.

Lewicki, R. J., J. A. Litterer, J. W. Minton, and D. M. Saunders (1994): *Negotiation*, 2nd edition, Irwin, Burr Ridge.

Lewicki, R. J., S. E. Weiss, and D. Lewin (1992): “Models of conflict, negotiation and third party intervention: A review and synthesis”, *Journal of Organizational Behavior*, 13(3), pp. 209–252.

Liebl, F. (1992): *Simulation*, Oldenbourg, München.

Lim, L. and I. Benbasat (1992): “A theoretical perspective of negotiation support systems”, *Journal of Management Information Systems*, 9(3), pp. 27–44.

Lin, F.-R. and K.-Y. Chang (2001): “A multiagent framework for automated online bargaining”, *Intelligent Systems*, 16, pp. 41–47*.

Loewenstein, J. and L. L. Thompson (2006): “Learning to negotiate: Novice and experienced negotiators”, in: L. L. Thompson (ed.), *Negotiation theory and research*, Psychology Press, New York, pp. 77–97.

Lomuscio, A. R., M. Wooldridge, and N. R. Jennings (2003): “A classification scheme for negotiation in electronic commerce”, *Group Decision and Negotiation*, 12, pp. 31–56.

Luo, X., N. R. Jennings, and N. Shadbolt (2006): “Aquiring user tradeoff strategies and preferences for negotiating agents: A default-then-adjust method”, *International Journal of Human Computer Studies*, 64(4), pp. 304–321.

Maes, P., R. H. Guttman, and A. G. Moukas (1999): “Agents that buy and sell”, *Communications of the ACM*, 42(3), pp. 81–91.

Maisel, H. and G. Gnugnoli (1972): *Simulation of discrete stochastic systems*, Science Research Assoc., Chicago.

Miller, R. G. (1981): *Simultaneous Statistical Inference*, 2nd edition, Springer series in statistics, Springer, New York.

Milter, R. G., T. A. Darling, and J. L. Mumpower (1996): “The effects of substantive task characteristics on negotiators’ ability to reach efficient agreements”, *Acta Psychologica*, 93, pp. 207–228.

Moore, D. A., T. R. Kurtzberg, L. L. Thompson, and M. W. Morris (1999): “Long and short routes to success in electronically mediated negotiations: group affiliations and good vibrations”, *Organizational Behavior and Human Decision Processes*, 77(1), pp. 22–43.

Moore, D. A. and J. K. Murnighan (1999): “Alternative models of the future of negotiation research”, *Negotiation Journal*, 15(4), pp. 347–353.

Müller, J.-A. (1998): *Simulation ökonomischer Prozesse*, Manz, Wien.

Mumpower, J. L. (1991): “The judgement policies of negotiators and the structure of negotiation problems”, *Management Science*, 37(10), pp. 1304–1324.

Mumpower, J. L. and J. Rohrbaugh (1996): “Negotiation and design: Supporting resource allocation decisions through analytical mediation”, *Group Decision and Negotiation*, 5(4-6), pp. 385–409.

Napel, S. (2002): *Bilateral bargaining: Theory and applications*, Springer, Berlin.

Nash, J. F. (1950): “The bargaining problem”, *Econometrica*, 18(2), pp. 155–162.

Nash, J. F. (1953): “Two-person cooperative games”, *Econometrica*, 21(1), pp. 128–140.

Nawa, N. E. (2006): “Agents that acquire negotiation strategies using a game theoretic learning theory”, *International Journal of Intelligent Systems*, 21, pp. 5–39*.

Naylor, T. H. and J. M. Finger (1967): “Verification of computer simulation models”, *Management Science*, 14(2), pp. 92–106.

Neale, M. A. and M. H. Bazerman (1985): “The effects of framing and negotiator overconfidence on bargaining behaviors and outcomes”, *Academy of Management Journal*, 28(1), pp. 34–49.

Neale, M. A. and G. Northcraft (1986): “Experts, amateurs, and refrigerators: A comparison of expert and amateur negotiators in a novel task”, *Organizational Behavior and Human Decision Processes*, 38, pp. 305–317.

Neelin, J., H. Sonnenschein, and M. Spiegel (1988): “A further test of noncooperative bargaining theory: Comment”, *American Economic Review*, 78(4), pp. 824–836.

Northcraft, G. B. and M. A. Neale (1987): “Experts, amateurs, and real estate: An anchoring-and-adjustment perspective on property pricing decisions”, *Organizational Behavior and Human Decision Processes*, 39, pp. 228–241.

Nwana, H. S., J. Rosenschein, T. Sandholm, C. Sierra, P. Maes, and R. Guttmann (1998): “Agent-mediated electronic commerce: Issues, challenges and some viewpoints”, in: *Proceedings of the second International Conference on Autonomous Agents*, Minneapolis, US, pp. 189–196.

Ochs, J. and A. E. Roth (1989): “An experimental study of sequential bargaining”, *American Economic Review*, 79(3), pp. 355–384.

Oliver, J. R. (1996): “A machine-learning approach to automated negotiation and prospects for electronic commerce”, *Journal of Management Information Systems*, 13(3), pp. 83–112*.

Osborne, M. J. and A. Rubinstein (1990): *Bargaining and markets*, Academic Press, San Diego.

Page, B. (1991): *Diskrete Simulation*, Springer, Berlin.

Park, S. and S.-B. Yang (2004): “An efficient automated negotiation system using multi-attributes in the online environment”, *Lecture Notes in Computer Science*, 3140/2004, pp. 544–557*.

Paurobally, S., P. J. Turner, and N. R. Jennings (2003): “Automating negotiation for m-services”, *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 33, pp. 709–724*.

Pen, J. (1952): “A general theory of bargaining”, *American Economic Review*, 42(1), pp. 24–42.

Petzoldt, T. (2003): “R as a simulation platform in ecological modelling”, *R News*, 3(3), pp. 8–16.

Pidd, M. (1992): *Computer simulatin in management science*, 3rd edition, Wiley, Chichester.

Pruitt, D. G. (1981): *Negotiation behavior*, Academic Press, New York.

Raiffa, H. (1953): “Arbitration schemes for generalized two-person games”, in: H. W. Kuhn and A. W. Tucker (eds.), *Contributions to the theory of games II, (Annals of Mathematic Studies 28)*, Princeton University Press, pp. 361–387.

Raiffa, H. (1982): *The art and science of negotiation*, Belknap, Cambridge.

Raiffa, H., J. Richardson, and D. Metcalfe (2002): *Negotiation analysis: The science and art of collaborative decision making*, Belknap Press, Cambridge.

Rangaswamy, A. and G. R. Shell (1997): “Using computers to realize joint gains in negotiations: Toward an ”electronic bargaining table””, *Management Science*, 43(8), pp. 1147–1163.

Rau, H., M.-H. Tsai, C.-W. Chen, and W.-J. Shiang (2006): “Learning-based automated negotiation between shiper and forwarder”, *Computers and Industrial Engineering*, 51, pp. 464–481*.

Ren, F., M. Zhang, and K. M. Sim (2009): “Adaptive conceding strategies for automated trading agents in dynamic, open markets”, *Decision Support Systems*, 46, pp. 704–716.

Rosenschein, J. S. and G. Zlotkin (1994): *Rules of encounter*, MIT Press, Cambridge.

Rosenthal, R. W. (1976): “An arbitration model for normal-form games”, *Mathematics of Operations Research*, 1, pp. 82–88.

Roth, A. E. (1977): “Independence of irrelevant alternatives and solutions to Nash’s bargaining problem”, *Journal of Economic Theory*, 16, pp. 247–251.

Rubin, J. Z. and B. R. Brown (1975): *The social psychology of bargaining and negotiation*, Academic Press, New York.

Rubinstein, A. (1982): “Perfect equilibrium in a bargaining model”, *Econometrica*, 50(1), pp. 97–109.

Sandholm, T. (1999): “Automated negotiation”, *Communications of the ACM*, 42(3), pp. 84–85.

Sargent, R. G. (2005): “Verification and validation of simulation models”, in: M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines (eds.), *Proceedings of the 2005 Winter Simulation Conference*, pp. 130–143.

Schoop, M., A. Jertila, and T. List (2003): “Negoisst: A negotiation support system for electronic business-to-business negotiations in e-commerce”, *Data and Knowledge Engineering*, 47(3), pp. 371–401.

Sebenius, J. K. (1992): “Negotiation analysis: A characterization and review”, *Management Science*, 38(1), pp. 18–38.

Shakun, M. F. (2005): “Multi-bilateral multi-issue e-negotiation in e-commerce with a tit-for-tat computer agent”, *Group Decision and Negotiation*, 14(5), pp. 383–392.

Shea, G. F. (1983): *Creative Negotiating*, CNI Publishing, Boston.

Simon, H. A. (1955): “A behavioral model of rational choice”, *Quarterly Journal of Economics*, 69, pp. 99–118.

Somefun, D., E. H. Gerding, S. Bohte, and J. La Poutre (2004): “Automated negotiation and bundling of information goods”, *Lecture Notes in Artificial Intelligence*, 3048/2004, pp. 1–17*.

Somefun, D. J. A., E. H. Gerding, and J. A. L. Poutre (2006): “Efficient methods for automated multi-issue negotiation: Negotiating over a two-part tariff”, *International Journal of Intelligent Systems*, 21, pp. 99–119.

Ståhl, I. (1972): *Bargaining theory*, Stockholm School of Economics, Stockholm.

Sycara, K. P. (1991): “Problem restructuring in negotiation”, *Management Science*, 37(10), pp. 1248–1268.

Teich, J. E., H. Wallenius, and J. Wallenius (1994): "Advances in negotiation science", *Transactions on Operational Research*, 6(1), pp. 55–94.

Thode, H. C. (2002): *Testing for Normality*, Marcel Dekker, New York.

Thompson, L. (1998): *The mind and the heart of the negotiator*, Prentice Hall, Upper Saddle River.

Thomson, W. (1981): "A class of solutions to bargaining problems", *Journal of Economic Theory*, 25, pp. 431–441.

Tripp, T. M. and H. Sondak (1992): "An evaluation of dependent variables in experimental negotiation studies: Impasse rates and Pareto efficiency", *Organizational Behavior and Human Decision Processes*, 51(2), pp. 273–295.

Tu, M. T., E. Wolff, and W. Lamersdorf (2000): "Genetic algorithms for automated negotiations: A FSM-based application approach", in: *Proceedings of the 11th International Workshop on Database and Expert Systems Applications (DEXA00)**.

Tutzauer, F. (1986): "Bargaining as a dynamical system", *Behavioral Science*, 31, pp. 65–81.

Tutzauer, F. (1992): "The communication of offers in dyadic bargaining", in: L. Putnam and M. E. Roloff (eds.), *Communication and Negotiation*, Sage, Newbury Park, pp. 67–82.

van Bragt, D. D. B. and J. A. La Poutre (2003): "Why agents for automated negotiations should be adaptive", *Netnomics*, 5, pp. 101–118*.

Varian, H. R. (1995): "Economic mechanism design for computerized agents", in: *Proceedings of the 1st USENIX Workshop on Electronic Commerce*, New York.

Vetschera, R. (2006): "Preference structures of negotiators and negotiation outcomes", *Group Decision and Negotiation*, 15(2), pp. 111–125.

von Neumann, J. and O. Morgenstern (1944): *Theory of games and economic behavior*, Princeton University Press, Princeton.

Vulkan, N. (1999): "Economic implications of agent technology and e-commerce", *The Economic Journal*, 109, pp. F67–F90.

Wall, J. A. and M. W. Blum (1991): "Negotiations", *Journal of Management*, 17(2), pp. 273–303.

Walton, R. E. and R. B. McKersie (1965): *A Behavioral Theory of Labor Negotiations*, McGraw-Hill, New York.

Wasfy, A. M. and Y. A. Honsi (1998): "Two-party negotiation modeling: An integrated fuzzy logic approach", *Group Decision and Negotiation*, 7, pp. 491–518*.

Weigand, H., A. de Moor, M. Schoop, and F. Dignum (2003): "B2B negotiation support: The need for a communication perspective", *Group Decision and Negotiation*, 12, pp. 3–29.

Winoto, P., G. I. McCalla, and J. Vassileva (2005): "Non-monotonic-offers bargaining protocol", *Autonomous Agents and Multi-Agent Systems*, 11, pp. 45–67*.

Wollkind, S., J. Valasek, and T. R. Ioerger (2004): "Automated conflict resolution for air traffic management using cooperative multiagent negotiation", in: *Proceedings of the AIAA guidance, Navigation, and Control Conference and Exhibit**.

Young, H. P. (ed.) (1991): *Negotiation Analysis*, The University of Michigan Press.

Zartmann, I. W. (2002): "The structure of negotiation", in: V. A. Kremenyuk (ed.), *International negotiation: Analysis, approaches, issues*, 2nd edition edition, Jossey-Bass, San Francisco, pp. 71–84.

Zeng, D. and K. Sycara (1998): "Bayesian learning in negotiation", *International Journal of Human-Computer Studies*, 48, pp. 125–141*.

Zeuthen, F. (1930): *Problems of monopoly and economic warfare*, Routledge and Sons, London.

Publications marked with an asterisk are part of the literature review in Chapter 3.1.

Appendix A

Data and Source Code

This appendix provides information on the structure of the data used in and generated by the simulation study, furthermore it provides the code of the simulation program.

A.1 Input and output data

All data is queried from and saved to a `Firebird` 2.1 database. The input data for the simulation, which is also used for the comparison of the performance of automated negotiation and human negotiation, is stored in the three tables `PREFERENCES`, `PROCESS`, and `OUTCOME` – Figure A.1. –, which were generated from the data provided by the InterNeg Research group on the Itex-Cypress negotiation experiments with the negotiation support system `Inspire` (see Figure A.1). The table `PREFERENCES` contains the result of `Inspire`'s utility elicitation procedure – i.e. partial utility values for all options of the negotiation object in the cells `PRI347` to `RET10`[1] – as results of the hybrid conjoint method used for utility elicitation. This data is combined with information about the user from whom these preferences were elicited (an unique ID for the user as primary key, the ID of the negotiation and the role of the user in the experiment). The table `PROCESS` contains a documentation of the offers exchanged in all negotiation experiments again combined with the ID of the negotiation, the ID and role of the user that sent the message, the cell `ACTION` that determines whether it was an offer or an acceptance message, the unique offer ID, the utility rating of this offer (the sum of the partial utility values of table `PREFERENCES` for this combination of options), and negotiation object specific columns that indicate the options the offer consists of. The last input table `OUTCOME` consists of a cell for the unique negotiation ID, the number of messages exchanged by both parties during the negotiation, a binary variable `AGR` indicating whether an agreement was reached (1) or not (0), and a binary variable indicating whether an agreement was Pareto-optimal (1) or not (0). Furthermore the cell `DIST` contains the minimal distance to the Pareto frontier if an agreement was reached. The utility of the final agreement – if one was reached – for both parties according to their preferences is stored in in the cells `UITEX` for the seller party and `UCYPRESS` for the buyer party and the integrativeness of the negotiation

[1]Note the name convention `ISSUEOPTION` for the table `PREFERENCES` and throughout the simulation program for determining the partial utilities of subjects in experiments and software agents in simulations for an option in an issue.

problem is saved in the cell `INT`.[2]

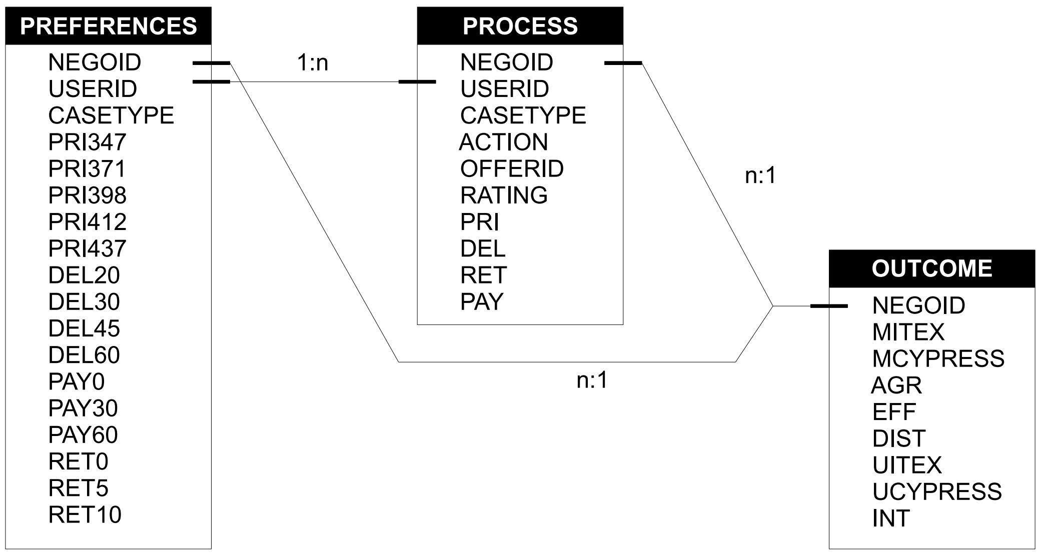


Figure A.1: Input tables of the database

The output of the simulation is stored by the simulation program in the database table `SIMOUTCOME` (see Figure A.2). This table consists of the ID of the negotiation simulated, the protocol used for simulation, the replication number, the number of turns the simulation took, the name of the seller and the buyer agent, the type of the last message – which can be either of `agree` in all protocols, `quit` in `protocol 2` only, or `terminated` in `protocol 3` only – and the option values for the issues of the negotiation object if an agreement was reached. Furthermore the cells `USELLER` and `UBUYER` contain the utility values to the parties if an agreement was reached. In `EFF` a binary variable is stored which indicates whether an agreement was Pareto-optimal (1) or not (0), and in `DIST` the minimal distance to the Pareto frontier is stored if an agreement was reached. Finally the number of messages sent by either party in total and divided into `offer`, `reject`, and `quit` messages is stored in the respective cells of table `SIMOUTCOME`.

[2]We save the integrativeness calculated according to the formula provided in Chapter 5 in the table `OUTCOME`, though it is actually no outcome of the negotiations but a result from the joint evaluation of the negotiation object, as this table is the only one that consists of unique tuples for each negotiation experiment.

SIMOUTCOME
NEGOID
PROTOCOL
IT
TURN
SELLER
BUYER
MSG
PRI
DEL
PAY
RET
USELLER
UBUYER
EFF
DIST
MSELLER
MBUYER
OSELLER
OBUYER
RSELLER
RBUYER
QSELLER
QBUYER

Figure A.2: Output table of the database

A.2 Source code

For the implementation of the conceptual model in a simulation program we used `R`[3]. `R` is a language and environment for statistical computing and graphics. It runs on a variety of platforms like UNIX/Linux, Windows, and MacOS, and is available as free software under the terms of the Free Software Foundation's GNU General Public License. It is similar to the `S` language, which was developed at Bell Laboratories by John Chambers and colleagues, and now is further developed as a different implementation of `S` by a core team and many people contributing software packages to the project. `R` not only provides a wide variety of statistical techniques – linear and nonlinear models, classical statistical tests, time-series analysis, clustering, etc. – but also functions for graphical illustration of data – including mathematical symbols and formulas – and is highly extensible. The strengths of the `R` language are the facility of data manipulation and database interaction, its simple and effective programming language – including conditionals, loops, user-defined function, etc. –, its sophisticated array and matrices operations, and its large collection of tools for statistical data analysis and graphical display.

Besides its functionality as general statistical computing program, `R` recently is also used in simulation studies. Many types of simulations can easily be implemented in `R`, like differential equation models (e.g. predator-prey-simulations), individual-based models (e.g. population dynamics or particle diffusion), or cellular automates (e.g. Conway's game of life) Petzoldt (2003). The strength of `R` in simulation, compared to standard simulation tools for these purposes, is its

[3]`www.r-project.org`

high flexibility and customizability concerning the users' special requirements.

The database interaction facilities and the possibility to systematically perform statistical test in combination with simulation are a great benefit to this study. Though, for the implementation of an operative system for automated negotiation other programing languages are more adequate as the resulting system has to be platform independent, operate over the Internet, be capable of handling agent registration and online message exchange, perform preference elicitation, ensure communication security, provide ontologies, etc., for the purpose of our study – i.e. the evaluation and comparison of different automated negotiation systems by means of simulation – `R` is sufficient.

A.2.1 Sourcing and parametrization

Before running the simulation the necessary functions have to be sourced and parameters for the simulation have to be determined, as shown in Listing A.1. First the current version and therefore the path from where to source the functions for the simulation is determined in the variables `version` and `path`. Then the database connection is established and the negotiation IDs of the experiments to be simulated is queried from the input database and saved in the vector `experiments`.

```
version<-15
path<-paste("W:/data/AN/V",version,"/",sep="")
source(paste(path,"DB_connection.r",sep=""))
experiments<-as.vector(t(sqlQuery(db, "SELECT NEGOID FROM OUTCOME")))

source(paste(path,"preferences.r",sep=""))
negoobject<-list(PRI=c(347,371,398,412,437),
                 DEL=c(20,30,45,60),
                 PAY=c(0,30,60),
                 RET=c(0,5,10))

source(paste(path,"negotiation.r",sep=""))
pnum<-3
protocol<-switch(pnum,
                 "1"=c(REJECT=FALSE,EXIT=FALSE),
                 "2"=c(REJECT=FALSE,EXIT=TRUE),
                 "3"=c(REJECT=TRUE,EXIT=FALSE))

source(paste(path,"MCact.R",sep=""))
source(paste(path,"MCpas.R",sep=""))
source(paste(path,"SMCact.R",sep=""))
source(paste(path,"SMCpas.R",sep=""))
source(paste(path,"MUMact.R",sep=""))
source(paste(path,"MUMpas.R",sep=""))
source(paste(path,"LEXact.R",sep=""))
source(paste(path,"LEXpas.R",sep=""))
source(paste(path,"TFT.R",sep=""))
agents<-c("MCact",
          "MCpas",
          "SMCact",
          "SMCpas",
          "MUMact",
          "MUMpas",
          "LEXact",
          "LEXpas",
          "TFT")
```

Listing A.1: Initiation of the simulation

After these initial steps the actual components of the simulation of automated negotiation are determined. The function `preferences`, for querying the users' preferences over the negotiation object from the input database, is sourced and the negotiation object is determined in the variable `negoobject` – for our study the negotiation object is the Itex-Cypress negotiation case (see Chapter 4). Afterwards the function with the generic interaction mechanism `negotiation` is sourced, which is closer specified in determining the protocol to be used in the simulation as being one out of the three protocols discussed in Chapter 4. After the protocol the nine agents are sourced as functions with their respective names and all agents are registered in the vector `agents`.

A.2.2 Database connection

To enable the simulation program to query data from the input tables and save simulation results to the output table of the `Firebird` database a link to the database has to be established in the variable `db` for reused during the simulation by the code provided in Listing A.2.

```
library(RODBC)
fblink<-function()
{
 odbcDriverConnect("DRIVER=Firebird/InterBase(r) driver;
                    UID=sysdba;
                    PWD=masterkey;
                    DBNAME=C:/Users/fedaykin/Documents/switch/an.fdb",
                   case="toupper")
}
db<-fblink()
```

Listing A.2: Database connection

A.2.3 Preference determination

To bring the simulation in accordance to the existing data on the Itex-Cypress experiments we first have to re-frame the the buyer and seller role as 'Cypress' and 'Itex' which were the names of the buyer and seller party in the experiment. In a next step the partial utilities for all options in all issues of the negotiation are queried from the table `PREFERENCES` of the database. Note that we here apply a name convention: The columns in the database containing the partial utility values for an option in an issue have to have the form `ISSUEOPTION` according to the negotiation object variable `negoobject` – e.g. the column where the utility value for delivery time of 60 days is saved in the database has to be named `DEL60`. The function `expand.grid` creates a matrix of all combinations of options for the issues of the negotiation object. For all rows of this matrix – i.e. all possible offers – the utility values are calculated and attached to the matrix. The function `preferences` returns a list containing the partial utility values queried from the input database, the negotiation object provided when calling the function, and the matrix with all possible offers together with their utility values. Note that the agents work with any form of utility elicitation and representation as long as they receive an input in form of a matrix that contains all possible settlement alternatives and utility values for them. Accordingly, if other elicitation methods or utility representations are used only this function `preferences` has to be adapted.

```
preferences<-function(negoid,negoobject,role)
{
 if(role=="seller"){role<-"ITEX"}
 if(role=="buyer"){role<-"CYPRESS"}
 utilrows<-NULL

 for(i in 1:length(names(negoobject)))
 {
  for(j in 1:length(negoobject[[i]]))
  {
   utilrows<-c(utilrows,paste(names(negoobject)[i],negoobject[[i]][j],sep=""))
  }
 }

 utilquery<-paste(utilrows,"",sep="",collapse=", ")
 util<-sqlQuery(db,paste("SELECT ",utilquery," FROM PREFERENCES
                          WHERE NEGOID=",negoid," AND CASETYPE='",role,"'",sep=""))

 all<-as.matrix(expand.grid(negoobject))
 U<-NULL
 for(k in 1:nrow(all))
 {
  dum<-sum(util[paste(names(negoobject),all[k,],sep="")])
  U<-c(U,dum)
 }
 all<-cbind(all,U)
 colnames(all)<-c(names(negoobject),"U")
 erg<-list(UTIL=util,PACKAGES=all,OBJ=negoobject)
 erg
}
```

Listing A.3: Preference determination

A.2.4 General interaction mechanism

The general negotiation mechanism is implemented in the function `negotiation`. This function starts the negotiation by initiating the agents, performs negotiations in alternately calling the software agents, and concludes the negotiation if the termination criteria described in Chapter 4 are fulfilled. The initiation of software agents is performed by the function in calling both sides' software agent functions with the preferences indicated in the database in the mode `initiate`. The return of a software agent function called in the mode `initiate` is a list containing its private information, which is saved in a variable for later reuse by the software agent during the negotiation.

After the initiation of agents the general interaction mechanism randomly determines a party to start the negotiation – with equal probability for each party to be chosen – in calling this agent in the mode `negotiate` with a message of type 'call for offer' `CFO`. Though the opening offer of both agents is the offer that affords highest utility to them the procedure of sending a 'call for offer' determines the sequence of message exchange. In subsequent turns the software agent that received the 'call for offer' will send its messages at odd turns, the other software agent at even turns. The function `negotiation` saves all messages sent by the software agents in the negotiation track variable `course` and changes in the private information of the software agents in their private storage variables, then it switches the roles and data variables and calls the other agent with the last message stored in the negotiation track. These operations result in an alternating turn procedure, where the software agents are called alternately with the last message of their opponent.

During the whole negotiation process the function scans the last messages saved for termination criteria i.e. if these last messages are either of the type `agree` or `quit`, or if the last two messages are of the type `reject`. Should this be the case the function terminates and returns the variable `course` which contains the whole negotiation track. The general interaction mechanism can be seen as a primitive negotiation protocol as it ensures that agents alternate in sending their messages, the other components of the protocol, especially the possible messages the protocol enables the software agents to send, are implemented in the software agents themselves. For determining which messages the software agents are allowed to send they refer to the `protocol` variable determined when parameterizing the simulation. That agents follow this protocol is ensured by their design for simplicity and not controlled by the general interaction mechanism of the function `negotiation`.

```
negotiation<-function(selleragent,buyeragent,sellerpref,buyerpref,protocol)
{
 sellerdata<<-do.call(selleragent,
                      list(mode="initiate",position="seller",preferences=sellerpref))
 buyerdata<<-do.call(buyeragent,
                      list(mode="initiate",position="buyer",preferences=buyerpref))

 if(runif(1)<=0.5)
 {current<<-buyeragent;position<<-"buyer"}else
 {current<<-selleragent;position<<-"seller"}

 if(position=="seller"){negodata<<-sellerdata}else{negodata<<-buyerdata}

 course<<-do.call(current,
                   list(mode="negotiate",position=position,message.type="CFO"))

 if(position=="seller"){sellerdata<<-negodata}else{buyerdata<<-negodata}
 if(position=="seller"){current<<-buyeragent}else{current<<-selleragent}
 if(position=="seller"){negodata<<-buyerdata}else{negodata<<-sellerdata}
 if(position=="seller"){position<<-"buyer"}else{position<<-"seller"}

 ende<-FALSE;
 while(!ende)
 {
  msg<-do.call(current,list(mode="negotiate",
                             position=position,
                             message.type=course[nrow(course),"MSG"],
                             content=course[nrow(course),
                             names(negoobject)],
                             protocol=protocol))
  course<<-rbind(course, msg)

  if(position=="seller"){sellerdata<-negodata}else{buyerdata<-negodata}

  if(course[nrow(course),"MSG"]=="agree"){ende<-TRUE}
  if(course[nrow(course),"MSG"]=="quit"){ende<-TRUE}
  if(course[nrow(course),"MSG"]=="reject"&course[nrow(course)-1,"MSG"]=="reject")
  {
   ende<-TRUE;
   course[,"MSG"]<-as.vector(course[,"MSG"]);
   course[nrow(course),"MSG"]<-"terminated";
  }

  if(position=="seller"){current<<-buyeragent}else{current<<-selleragent}
  if(position=="seller"){negodata<<-buyerdata}else{negodata<<-sellerdata}
  if(position=="seller"){position<<-"buyer"}else{position<<-"seller"}
 }
 course
}
```

Listing A.4: Interaction mechanism

A.2.5 Main loop

The main loop ensures the tournament-style full factorial – when applied for each interaction protocol – experimental design for the simulation discussed in Chapter 5, in letting for a given interaction protocol all possible combinations of software agents negotiate for all negotiation problems of the vector `experiments` (see Listing A.5).

```
selleragents<-agents
buyeragents<-agents

for(i in experiments)
{
 for(j in selleragents)
 {
  for(k in buyeragents)
  {
   sellerpref<-preferences(i,negoobject,role="seller")
   buyerpref<-preferences(i,negoobject, role="buyer")
   for(l in 1:3)
   {
    erg<-negotiation(j,k,sellerpref,buyerpref,protocol)
    outcome<-as.data.frame(c(
     NEGOID=i,
     PROTOCOL=pnum,
     IT=l,
     TURN=nrow(erg),
     SELLER=j,
     BUYER=k,
     erg[nrow(erg),3:(3+length(negoobject))],
     USELLER=0,
     UBUYER=0,
     EFF=1,
     MSELLER=nrow(subset(erg,CASETYPE=="seller")),
     MBUYER=nrow(subset(erg,CASETYPE=="buyer")),
     OSELLER=nrow(subset(erg,CASETYPE=="seller"&MSG=="offer")),
     OBUYER=nrow(subset(erg,CASETYPE=="buyer"&MSG=="offer")),
     RSELLER=nrow(subset(erg,CASETYPE=="seller"&MSG %in% c("reject","terminated"))),
     RBUYER=nrow(subset(erg,CASETYPE=="buyer"&MSG %in% c("reject","terminated"))),
     QSELLER=nrow(subset(erg,CASETYPE=="seller"&MSG=="quit")),
     QBUYER=nrow(subset(erg,CASETYPE=="buyer"&MSG=="quit"))))
    if(outcome[1,"MSG"]!="agree"){outcome[1,"EFF"]<-0;}
    if(outcome[1,"MSG"]=="agree")
    {
     outcome[1,"USELLER"]<-sum(sellerpref$UTIL
            [paste(names(negoobject),outcome[1,8:(7+length(negoobject))],sep="")]);
     outcome[1,"UBUYER"]<-sum(buyerpref$UTIL
            [paste(names(negoobject),outcome[1,8:(7+length(negoobject))],sep="")]);
     for(m in 1:nrow(sellerpref$PACKAGES))
     {
      if(
         sellerpref$PACKAGES[m,"U"]>outcome[1,"USELLER"]&
         buyerpref$PACKAGES[m,"U"]>=outcome[1,"UBUYER"]|
         sellerpref$PACKAGES[m,"U"]>=outcome[1,"USELLER"]&
         buyerpref$PACKAGES[m,"U"]>outcome[1,"UBUYER"])
      {
       outcomerow[1,"EFF"]<-0;
      }
     }
    }
    sqlSave(db,outcome,tablename="SIMOUTCOME",append=TRUE,rownames=FALSE,)
   }
  }
 }
}
```

Listing A.5: Main loop

The main loop for the computer experiments first defines the software agents to be used in the run of the simulation – in our case all nine agents are used for the buyer and the seller side, therefore the whole registration vector `agent` is assigned to the vectors representing the set of seller and buyer agents, `selleragents` and `buyeragents` respectively. For all combinations of one seller and one buyer agent the preferences over the negotiation object for the focal experiment are queried from the input table of the database using the `preferences` function and saved in the variables `sellerpref` and `buyerpref` for reuse in all replications.[4] For each replication of a specific simulation the function `negotiation` is called with the agents to be used for the seller and buyer party, their preferences, and the protocol to be used as parameters.

As mentioned above the return of this function is the negotiation track. From this variable the data necessary for analyzing the computer experiments are aggregated, extracted, and saved in the `SIMOUTCOME` table of the `Firebird` database. This data covers the negotiation ID of the experiment, the ID of the protocol used, the ID of the replication, the number of turns the negotiation took, the name of the seller and the buyer agent, the final message of the negotiation, the number of messages sent by the buyer and seller agents in total and divided into `offer`, `reject`, and `quit` messages. Furthermore in case the software agents reached an agreement the utility of this agreement to both parties is calculated and it is analyzed whether this agreement is Pareto-optimal, i.e. if there exist no other possible alternatives dominating the reached agreement.

A.2.6 Software agents

The structure of the software agents is equal for all agents, as are parts of their decisions algorithms, we therefore – to emphasize the equalities, save space, and increase transparency – provide the code parts that are equal together in a software agent 'template' and only discuss the parts in which the agents differ in specific sections. The general structure of the software agents is represented in Table A.1.[5] After a formal definition of the function that implements the software agent the initiation procedure followed if the function is called in the mode `initiate` is discussed. In this mode the agent receives the preferences over the negotiation object queried from the input table of the database and creates its private data variable. The next section deals with the actions to be performed if called in mode `negotiate`. The general reaction to the `CFO`, or if no offer was sent yet, is to send the best possible offer as initial or opening offer. In a next step the opponent's last message is evaluated and if necessary the private information is updated – i.e. if the last message of the opponent was an offer, the 'last opponent offer' variable is changed accordingly. Then the next offer to propose is generated and it is decided whether

[4] As described in Chapter 5 and the subsequent chapter of the appendix the number of replications is set to three.

[5] Function determination and nomenclature (Listing A.6), reaction to a `CFO` (Listing A.8), execution of actions – i.e. sending the messages – in line with the protocol (Listing A.10), and updating of information if an offer was sent (Listing A.11) are equal for all agents and therefore covered in the software agent template in Section A.2.6.1. Agent initiation is different for `LEX` and `TFT` agents and therefore discussed in Sections A.2.6.5 and A.2.6.6, respectively. As message evaluation is more complex for the `TFT` agent it is also handled in the `TFT` section. Offer generation and the decision whether or not the strategy should be interrupted (step 5) is the distinguishing feature of the software agents and therefore discussed for each agent separately and in detail in Sections A.2.6.2 to A.2.6.6. As not all of the components of the code are equal in length for all software agents, to be consistent with the code lines indicated in Table A.1, we report the maximal length and, to keep a good overview, added blank lines to the components of the agents which are shorter than this maximal length.

Template part	Code lines		
1. Function determination and nomenclature	1	–	8
2. Agent initiation procedure	9	–	66
3. Reaction to a `CFO`	67	–	79
4. Message evaluation and standard message generation	80	–	105
5. Offer generation and `negobasis` decision	106	–	131
6. Execution in line with protocol	132	–	148
7. Information updating	149	–	171

Table A.1: Structure of the software agents

or not to follow the offering procedure on the basis of the current course of the negotiation, which means determining and evaluating the internal variable `negobasis`. The decisions made are executed in form of specific actions according to the restrictions imposed by the interaction protocol. If this execution means that an offer will be sent the agent's private information is updated accordingly and the function terminates returning the message for its current turn to the function `negotiation`.

A.2.6.1 Agent template

The function determination and nomenclature is equal for all agent, the functions are only named differently – `AGENT` in the code stands for the name of the agent i.e. one of `MOCact`, `MOCpas`, `SMCact`, `SMCpas`, `MUMact`, `MUMpas`, `LEXact`, `LEXpas`, or `TFT`, respectively.

```
1   AGENT<-function(mode="",
2                   message.type="",
3                   position="",
4                   content="",
5                   preferences="",
6                   protocol)
7   {
8    agentname<-"AGENT"
```

Listing A.6: Function determination and nomenclature

```
9    if(mode=="initiate")
10   {
11    RANK<-rank(preferences$PACKAGES[,"U"],ties.method="random")
12    SENT<-rep(0,times=nrow(preferences$PACKAGES))
13    preferences$PACKAGES<-cbind(SENT,RANK,preferences$PACKAGES)
14    preferences[[length(preferences)+1]]<-0
15    preferences[[length(preferences)+1]]<-100
16    preferences[[length(preferences)+1]]<-0
17    preferences[[length(preferences)+1]]<-0
18    names(preferences)<-c(names(preferences)[1:(length(preferences)-4)],
19                             "lastown",
20                             "lastownu",
21                             "lastopp",
22                             "lastoppu")
23    preferences
24   }
```

Listing A.7: Agent initiation procedure

The initiation is quite equal for the `MOC`-, `SMC`-, and `MUM`-agents too, but differs slightly from `TFT` – as an additional variable and a different setting of `lastoppu` is necessary to guarantee a concession by TFT in its first move and to calculate concessions of the opponent. It furthermore differs significantly from the initiation of the LEX-agent as the lexicographical ordering is more complex. The two exceptions therefore will be discussed in their specific sections. When called in the mode `initiate` the agents use the preference information provided to them with the call to rank the offers according to the utility they afford (and randomly in case of ties), include a field that indicates whether the offer already was sent during the negotiation (a sent-flag), and create some additional variables that are necessary for decision making and updating of information during the course of negotiation. These additional variables are `lastown` and `lastownu` where the last offer and its utility are stored, as well as `lastopp` and `lastoppu`, which contain the last offer of the opponent and its utility, respectively. The return of the software agent functions in mode `initiate` is a changed and enlarged list of private information for negotiation.

```
else if(mode=="negotiate")
{
 if(message.type=="CFO"|length(negodata$lastown)==1)
 {
  x<-order(negodata$PACKAGES[,"SENT"],-negodata$PACKAGES[,"RANK"])
  negodata$PACKAGES<<-negodata$PACKAGES[x,]
  msg<-data.frame(CASETYPE=position,
                  AGENT=agentname,
                  MSG=as.character("offer"),
                  t(negodata$PACKAGES[1,names(negoobject)]))
  negodata$PACKAGES[1,"SENT"]<<-1;
  negodata$lastown<<-as.data.frame(t(negodata$PACKAGES[1,names(negoobject)]))
 }
```

Listing A.8: Reaction to a `CFO`

```
 else
 {
  if(message.type=="offer")
  {
   negodata$lastopp<<-content
   names(negodata$lastopp)<<-names(negoobject)
   negodata$lastoppu<<-sum(negodata$UTIL[paste(names(negoobject),content,sep="")])
  }

  nas<-rep(NA,times=length(names(negoobject)))
  names(nas)<-names(negoobject)
  reject<-data.frame(CASETYPE=position,
                     AGENT=agentname,
                     MSG=as.character("reject"),
                     t(nas))
  quit<-data.frame(CASETYPE=position,
                   AGENT=agentname,
                   MSG=as.character("quit"),
                   t(nas))
  agree<-data.frame(CASETYPE=position,
                    AGENT=agentname,
                    MSG=as.character("agree"),
                    negodata$lastopp)
```

Listing A.9: Message evaluation and standard message generation

If the agent is called not in mode `initiate` but in mode `negotiate` it first checks if it is its first message to send – i.e. whether it received the `CFO` from the protocol or has not sent any message yet. In either of these cases the software agent makes the highest ranked – most preferable – offer and sets the sent flag for this offer to 1 in its private information. This initial offer is forced as no comparisons of concessions are possible for the first offer, i.e. utility differences between two offers cannot be calculated for this first offer but only for subsequent offers. Furthermore as the offer is the most preferred one with the highest utility to the software agent nothing is lost by proposing this initial offer. If the message is not the first but any other during the negotiation, the software agent checks whether the last message of the opponent was an offer and updates its private information accordingly. After this information updating the software agent creates different variables indicating different possible messages. Here only the standard messages `agree`, `reject`, and `quit` are covered as the determination of an `offer` message is agent specific and therefore discussed in later sections.

```
    if(negobasis)
    {
     if(negodata$lastoppu>=unextown){msg<-agree;update<-FALSE}
     else{msg<-offer;update<-TRUE}
    }
    if(!negobasis)
    {
     if(protocol["REJECT"]){msg<-reject;update<-FALSE}
     if(!protocol["REJECT"])
     {
      if(protocol["EXIT"]){msg<-quit;update<-FALSE}
      if(!protocol["EXIT"])
      {
       if(nooffers){msg<-agree;update<-FALSE}else{msg<-offer;update<-TRUE}
      }
     }
    }
```

Listing A.10: Execution in line with protocol

```
    if(update)
    {
     negodata$lastown<<-possible[1,names(negoobject)]
     negodata$lastownu<<-unextown
     sentref<-matrix(data=rep(negodata$lastown,times=nrow(expand.grid(negoobject))),
                     nrow=nrow(expand.grid(negoobject)),
                     ncol=length(names(negoobject)),
                     byrow=TRUE)
     origin<-negodata$PACKAGES[,names(negoobject)]
     SENTSIM<-apply(sentref==origin,1,sum)
     negodata$PACKAGES<<-cbind(negodata$PACKAGES,SENTSIM)
     for(z in 1:nrow(negodata$PACKAGES))
     {
      if(negodata$PACKAGES[z,"SENTSIM"]==max(SENTSIM))
      {
       negodata$PACKAGES[z,"SENT"]<<-1
      }
     }
     negodata$PACKAGES<<-negodata$PACKAGES[,1:ncol(negodata$PACKAGES)-1]
    }
   }
   msg
  }
 }
```

Listing A.11: Information updating

The decision which of the created messages to send, in accordance with the protocol, is made according to the agent flowchart presented in Figure 4.9 in Chapter 4. If there exists a basis for further negotiation (which is an agent specific decision and discussed subsequently in Section A.2.6.7) the agent checks whether to make the offer just generated or to accept the opponents last offer. This decision is made in favor of accepting the last offer of the opponent if it affords at least the same utility as the next offer of the agent would provide, otherwise the offer generated will be sent. If there exists no negotiation basis the agent sends a `reject` message if allowed by the protocol (`protocol 3`) – to elicit a new offer from the opponent but also risking termination by the protocol if the opponent also rejects to send an offer. If rejection is not possible the agent will break off the negotiation by sending a `quit` message – to avoid exploitation in permanently interrupting the concession strategy – if the protocol allows this (`protocol 2`). If neither rejection nor exit are possible (`protocol 1`) the agent sends the generated offer if he has still offers or accepts the last offer if there are no offers left to send. In line with the determination of the message the variable `update` is assigned with a Boolean term indicating whether the agent has to update its private information after sending a message, which is only the case if something changes i.e. if the next message of the software agent is an offer. This updating of the private information includes changing the variables `lastown`, `lastownu` as well as setting the sent-flag in the package matrix from 0 to 1 for the offer made. The final part of the agent function's code returns the generated message `msg` to the calling function `negotiation` which saves it in the negotiation track.

A.2.6.2 `SMC`

```
   x<-order(negodata$PACKAGES[,"SENT"],-negodata$PACKAGES[,"RANK"])
   negodata$PACKAGES<<-negodata$PACKAGES[x,]
   dummy<-as.data.frame(negodata$PACKAGES)
   possible<-subset(dummy, U<negodata$lastownu&SENT!=1)

   nooffers<-nrow(possible)==0
   if(!nooffers)
   {
    offer<-data.frame(CASETYPE=position,
                      AGENT=agentname,
                      MSG=as.character("offer"),
                      possible[1,names(negoobject)])
    unextown<-sum(negodata$UTIL
                  [paste(names(negoobject),possible[1,names(negoobject)],sep="")])
    negobasis<-(negodata$lastoppu>=100-negodata$lastownu)
   }
   if(nooffers&negodata$lastoppu<negodata$lastownu){negobasis<-FALSE}
   if(nooffers&negodata$lastoppu>=negodata$lastownu)
   {unextown<-negodata$lastownu;negobasis<-TRUE}
```

Listing A.12: Offer generation and `negobasis` decision `SMC`

`SMC` orders all packages decreasing in the rank determined in the `initiation` mode i.e. decreasing in the packages' utility and randomly in case of ties. The subset of the packages that were not sent yet and constitute a real concession i.e. provide strictly lower utility than the last offer sent is generated and the first row of this matrix is selected as the next offer.

If there exist no such offers the `nooffers` variable is set to `TRUE`. In case no further offers to propose exist and the last offer of the opponent is not acceptable there exists no basis for further negotiation, otherwise `unextown` is set to `lastownu` and `negobasis` to `TRUE` so that the agent accepts the last offer of the opponent. If there exists an offer to be sent, however, the agent determines whether or not this offer should be sent or rather the concession strategy should be interrupted – if this is enabled by the protocol. The result of this decision is saved in the variable `negobasis`. Note that here and in all following code parts only the active concession strategy (`act`) is presented – i.e. `SMCact` for this agent – the difference to the passive version is discussed in Section A.2.6.7.

A.2.6.3 `MOC`

```
  x<-order(negodata$PACKAGES[,"SENT"],-negodata$PACKAGES[,"RANK"])
  negodata$PACKAGES<<-negodata$PACKAGES[x,]
  dummy<-as.data.frame(negodata$PACKAGES)
  possible<-subset(dummy, U<=negodata$lastownu&SENT!=1)

  nooffers<-nrow(possible)==0
  if(!nooffers)
  {
   offer<-data.frame(CASETYPE=position,
                     AGENT=agentname,
                     MSG=as.character("offer"),
                     possible[1,names(negoobject)])
   unextown<-sum(negodata$UTIL
                 [paste(names(negoobject),possible[1,names(negoobject)],sep="")])
   negobasis<-(negodata$lastoppu>=100-negodata$lastownu)
  }
  if(nooffers&negodata$lastoppu<negodata$lastownu){negobasis<-FALSE}
  if(nooffers&negodata$lastoppu>=negodata$lastownu)
  {unextown<-negodata$lastownu;negobasis<-TRUE}
```

Listing A.13: Offer generation and `negobasis` decision `MOC`

The only difference between `MOC` and `SMC` is – as discussed in Chapter 4 – that `MOC` also sends offers affording the same level of utility and not only offers with strictly lower utility than the last offer. This is achieved in changing the criterion for subsetting the possible next offers in selecting those that afford smaller or equal utility compared to the last offer sent, the remainder of the code is equal to the code of the `SMC`-agent.

A.2.6.4 `MUM`

The `MUM`-agent also orders the offers in decreasing utility and then queries the subset of packages where the package configuration differs only in one issue form that of the last offer sent and affords smaller or equal utility. The first row of this subset of possible offers is taken as the next offer to be sent. Thereby being the offer that constitutes the smallest concession by a change in one issue only which implements the least-cost-issue approach discussed in Chapter 4.

A.2.6.5 `LEX`

The offering procedure and decision whether or not to follow the concession procedure of the `LEX`-agent are very similar to those of the other agents discussed above as can be seen from Listing A.15. The only difference is that the offers are ordered by increasing rank – due to the form of the lexicographical ordering in the initiation phase. As for the other software agents also the `LEX`-agent subsets all not yet send offers that afford lower or equal utility compared to the last offer sent, but for the `LEX`-agent this subset is ordered lexicographically. The real difference and complexity of this software agent results from establishing this lexicographic ordering when called first in mode `initiate` as described in Listing A.16.

```
    ref<-matrix(data=rep(negodata$lastown,times=nrow(expand.grid(negoobject))),
                nrow=nrow(expand.grid(negoobject)),
                ncol=length(names(negoobject)),
                byrow=TRUE)
    x<-order(negodata$PACKAGES[,"SENT"],-negodata$PACKAGES[,"RANK"])
    negodata$PACKAGES<<-negodata$PACKAGES[x,]
    origin<-negodata$PACKAGES[,names(negoobject)]
    SIM<-apply(ref==origin,1,sum)
    similarity<-as.data.frame(cbind(negodata$PACKAGES,SIM))
    possible<-subset(similarity,
                     U<=negodata$lastownu&SIM==length(names(negoobject))-1&SENT!=1)

    nooffers<-nrow(possible)==0
    if(!nooffers)
    {
     offer<-data.frame(CASETYPE=position,
                       AGENT=agentname,
                       MSG=as.character("offer"),
                       possible[1,names(negoobject)])
     unextown<-sum(negodata$UTIL
                   [paste(names(negoobject),possible[1,names(negoobject)],sep="")])
     negobasis<-(negodata$lastoppu>=100-negodata$lastownu)
    }
    if(nooffers&negodata$lastoppu<negodata$lastownu){negobasis<-FALSE}
    if(nooffers&negodata$lastoppu>=negodata$lastownu)
    {unextown<-negodata$lastownu;negobasis<-TRUE}
```

Listing A.14: Offer generation and `negobasis` decision `MUM`

```
    x<-order(negodata$PACKAGES[,"SENT"],negodata$PACKAGES[,"RANK"])
    negodata$PACKAGES<<-negodata$PACKAGES[x,]
    dummy<-as.data.frame(negodata$PACKAGES)
    possible<-subset(dummy,U<=negodata$lastownu&SENT!=1)

    nooffers<-nrow(possible)==0
    if(!nooffers)
    {
     offer<-data.frame(CASETYPE=position,
                       AGENT=agentname,
                       MSG=as.character("offer"),
                       possible[1,names(negoobject)])
     unextown<-sum(negodata$UTIL
                   [paste(names(negoobject),possible[1,names(negoobject)],sep="")])
     negobasis<-(negodata$lastoppu>=100-negodata$lastownu)
    }
    if(nooffers&negodata$lastoppu<negodata$lastownu){negobasis<-FALSE}
    if(nooffers&negodata$lastoppu>=negodata$lastownu)
    {unextown<-negodata$lastownu;negobasis<-TRUE}
```

Listing A.15: Offer generation and `negobasis` decision `LEX`

To achieve the lexicographic ordering of alternatives when the software agent is first called in mode `initiate` the rather complex procedure provided in Listing A.16 has to be followed. Its complexity mainly results from the aim of keeping the software agents generic to be applicable for many different negotiation objects, so that numbers and names of issues and options of the negotiation object are assumed to be unknown in advance, and the lack of standard functions for lexicographic ordering in `R`. In a first step the options are ranked by comparing the average utility of alternatives that have different options in one issue. The option affording the highest average utility is ranked highest and the option leading to lowest average utility lowest. Furthermore the weights of the issues are calculated as the difference between their highest average utility and lowest average utility option. In a last step a lexicographic ordering is established by ordering the packages decreasing from the highest (second highest, third highest, . . .) ranked option in the highest (second highest, third highest, ...) ranked issue to the (..., third lowest, second lowest) lowest option in the (..., third lowest, second lowest) lowest ranked issue.

```
if(mode=="initiate")
{
 prefdf<-as.data.frame(preferences$PACKAGES)
 rankobj<-preferences$OBJ
 weights<-NULL
 for(x in names(negoobject))
 {
  dummy<-aggregate(prefdf[,"U"],list(prefdf[,x]),mean)
  value<-max(dummy[,2])-min(dummy[,2])
  valuerank<-rank(dummy[,2],ties.method="random")
  weights<-c(weights,value)
  rankobj[[x]]<-rbind(rankobj[[x]],valuerank)
 }
 names(weights)<-names(negoobject)
 weights<-sort(weights,decreasing=TRUE)

 RANKUTIL<-NULL
 NAMES<-NULL
 for(y in names(negoobject))
 {
  RANKUTIL<-c(RANKUTIL,rankobj[[y]][2,])
  NAMES<-c(NAMES,paste(y,rankobj[[y]][1,],sep=""))
 }
 names(RANKUTIL)<-NAMES
 for(i in 1:nrow(prefdf))
 {
  for(j in names(negoobject))
  {
   prefdf[i,j]<-RANKUTIL[paste(j,prefdf[i,j],sep="")]
  }
 }

 beginning<-paste("order(-prefdf[,\"",names(weights)[1],"\"]",sep="")
 middle<-NULL
 for(i in 2:(length(names(weights))-1))
 {
  middle<-paste(middle,",-prefdf[,\"",names(weights)[i],"\"]",sep="")
 }
 end<-paste(",-prefdf[,\"",names(weights)[length(names(weights))],"\"])",sep="")
 dummy<-paste(beginning,middle,end,sep="")
 x<-eval(parse(text=dummy))
 preferences$PACKAGES<-preferences$PACKAGES[x,]

 RANK<-seq(nrow(preferences$PACKAGES):1)
 SENT<-rep(0,times=nrow(preferences$PACKAGES))
 preferences$PACKAGES<-cbind(SENT,RANK,preferences$PACKAGES)

 preferences[[length(preferences)+1]]<-0
 preferences[[length(preferences)+1]]<-100
 preferences[[length(preferences)+1]]<-0
 preferences[[length(preferences)+1]]<-0
 names(preferences)<-c(names(preferences)[1:(length(preferences)-4)],
                                          "lastown",
                                          "lastownu",
                                          "lastopp",
                                          "lastoppu")
 preferences
}
```

Listing A.16: Agent initiation procedure LEX

A.2.6.6 TFT

The two differences in the initiation phase between TFT and the other agents are first, that TFT requires an additional variable to save the utility of the penultimate opponent's offer `prelastoppu` – to calculate the opponent's last concession from the difference between the utility of the penultimate and last offer. Second, to guarantee that the agent starts with a concession – and not makes an other offer of same utility or sends `reject` or `quit` messages – the value of `lastoppu` is set to -1, which leads to the smallest possible concession made by the TFT agent with his second offer.

```
if(mode=="initiate")
{
 RANK<-rank(preferences$PACKAGES[,"U"],ties.method="random")
 SENT<-rep(0,times=nrow(preferences$PACKAGES))
 preferences$PACKAGES<-cbind(SENT,RANK,preferences$PACKAGES)
 preferences[[length(preferences)+1]]<-0
 preferences[[length(preferences)+1]]<-100
 preferences[[length(preferences)+1]]<-0
 preferences[[length(preferences)+1]]<-(-1)
 preferences[[length(preferences)+1]]<-0
 names(preferences)<-c(names(preferences)[1:(length(preferences)-5)],
                             "lastown",
                             "lastownu",
                             "lastopp",
                             "lastoppu"
                             "prelastoppu")
 preferences
}
```

Listing A.17: Agent initiation procedure TFT

```
 else
 {
  if(message.type=="offer")
  {
   negodata$prelastoppu<<-negodata$lastoppu
   negodata$lastopp<<-content
   names(negodata$lastopp)<<-names(negoobject)
   negodata$lastoppu<<-sum(negodata$UTIL[paste(names(negoobject),content,sep="")])
   opp.concession<-negodata$lastoppu>negodata$prelastoppu
   if(opp.concession){negobasis<-TRUE}else{negobasis<-FALSE}
  }

  nas<-rep(NA,times=length(names(negoobject)))
  names(nas)<-names(negoobject)
  reject<-data.frame(CASETYPE=position,
                     AGENT=agentname,
                     MSG=as.character("reject"),
                     t(nas))
  quit<-data.frame(CASETYPE=position,
                   AGENT=agentname,
                   MSG=as.character("quit"),
                   t(nas))
  agree<-data.frame(CASETYPE=position,
                    AGENT=agentname,
                    MSG=as.character("agree"),
                    negodata$lastopp)
```

Listing A.18: Message evaluation and standard message generation TFT

TFT is also specific concerning the fourth step, where messages are evaluated, as this additional variable `prelastoppu` has to be updated. The remaining opponent's message evaluation and standard message generation procedure is the same as for the other agents. It updates its additional variable `prelastoppu` before updating `lastoppu` and then calculates whether or not a concession was made with the offer in a first step. Only in case of a concession `TFT` has a basis for further negotiation – and reciprocates this concession – otherwise not. Also `reject` messages are reciprocated which indirectly results in a termination of the negotiation.

```
    ref<-matrix(data=rep(negodata$lastopp,times=nrow(expand.grid(negoobject))),
                nrow=nrow(expand.grid(negoobject)),
                ncol=length(names(negoobject)),
                byrow=TRUE)
    origin<-negodata$PACKAGES[,names(negoobject)]
    SIM<-apply(ref==origin,1,sum)
    similarity<-as.data.frame(cbind(negodata$PACKAGES,SIM))
    possible<-subset(similarity,SENT==0&U<=(100-negodata$lastoppu))

    nooffers<-nrow(possible)==0
    if(!nooffers)
    {
     y<-order(-possible[,"U"],-possible[,"SIM"])
     possible<-possible[y,]
     offer<-data.frame(CASETYPE=position,
                       AGENT=agentname,
                       MSG=as.character("offer"),
                       possible[1,names(negoobject)])
     unextown<-sum(negodata$UTIL
                   [paste(names(negoobject),possible[1,names(negoobject)],sep="")])
    }
    if(nooffers&negodata$lastoppu<negodata$lastownu){negobasis<-FALSE}
    if(nooffers&negodata$lastoppu>=negodata$lastownu)
    {unextown<-negodata$lastownu;negobasis<-TRUE}
    if(message.type=="reject"){negobasis<-FALSE}
```

Listing A.19: Offer generation and `negobasis` decision `TFT`

In the offer generation step `TFT` subsets all possible offers to find only those that were not sent already and provide the same or larger overall concession compared to the concessions made by the opponent up to this point in time. Furthermore the similarity of the offers to the offer of the opponent is calculated by counting in how many issues the last opponent's offer and the possible packages have equal options. This subset of possible packages is then ordered first decreasing by utility (so that the highest utility offers still being a concession of equal size are ranked first) and second decreasing by similarity (so that in case of ties in utility the offers more similar to the last of the opponent are ranked first) and then the first of these offers is taken as next offer to be sent. Note that the `TFT`-agent not checks whether the opponents concession magnitude is up to the own for determining the basis for further negotiation as all the other agents do. This is implemented in the `TFT`-agent anyways, as concessions of equal size of the opponent's concession are made only if the opponent itself makes a concession.

A.2.6.7 `act` vs. `pas` concession strategies

The only difference between software agents following an active `act` or passive `pas` concession strategy for `SMC`, `MOC`, `MUM`, and `LEX` is the determination of the variable `negobasis`. This is set for the active agents to

```
negobasis <-(negodata$lastoppu >=100-negodata$lastownu)
```

Listing A.20: `negobasis` determination for `act` concession strategies

as it is done in the agent specific sections A.2.6.2 to A.2.6.5.[6] A software agent following a passive concession strategy can easily be derived in replacing this line by

```
negobasis <-(negodata$lastoppu >=100-nextownu)
```

Listing A.21: `negobasis` determination for `pas` concession strategies

This code indicates that agents following passive concession strategies will make a concession, of the magnitude indicated by their next offer to be sent, only if the opponent agent already made a total concession of this magnitude – measured in terms of the focal negotiators utility. By contrast the software agents following an active concession strategy make a concession by its offer already if the opponent with his current offer reciprocated concessions the focal agent made up to his last offer – ignoring the next offer to be sent.

[6]Note that, therefore the code provided in the previous listings represents the code of software agents that follow the active concession strategy: `SMCact`, `MOCact`, `MUMact`, and `LEXact`.

Appendix B

Replications

As argued in Chapter 5 to obtain reliable results replications of terminating simulations are necessary if the results are influenced by stochastic effects. However, an arbitrarily determined number of replications can either lead to high computational efforts, if it is set too high, or unreliable results, if it is set too low. For the purpose of determining the minimal necessary number of replication to achieve stable average results we randomly sampled ten out of the $2,065$ negotiation problems and used them in simulations of all system configuration (i.e. all software agent combinations with all protocols) for several numbers of replications in a pre-test.[1]

The process and outcome variables considered for determining the minimal necessary number of replications are the average duration of the negotiation (measured in turns), the proportion of agreements reached, and the average utility of the outcome to both the buyer and the seller party (measured on an utility scale ranging from 0 to 100). We determined the number of replications with x replications of a simulation run – in the first trial reported in this appendix x ranged from 2 to 5 . For the ten randomly sampled negotiation problems and all system configurations the two to five replications resulted in to $9*9*3*x$ or 486, 729, 972, and 1215, simulation runs, respectively.

Results were aggregated by calculating proportions – in case of agreement – and averages – in case of the remaining outcome variables duration, utility to the seller and the buyer – over the number of replications for each of the 81 agent combinations. These aggregated results over different numbers of replications then are compared for each of the ten experiments selected and the three protocols to determine the minimal necessary number of replications to achieve stable results in our simulation.[2] For statistical tests of significant differences in the average outcomes of x replications and $x+1$ replications of a simulation run we could use a parametric t-test for two paired samples, as the settings (agent combination, protocol, experiment) are identical in both samples except the number of replications, that leads to the average outcomes. However,

[1]We employed the `R` function `sample` to randomly sample – without replacement – ten negotiation IDs out of the $2,065$ from vector `experiments` that contains all IDs of the negotiation experiments used for this study. The resulting resulting sample was 2381, 3044, 519, 348, 2771, 848, 1941, 1541, 617, and 304.

[2]Note that we separated analyses between protocols as the protocol was assumed to have major influence on the final outcome. Given the tournament experimental design is repeated for each protocol – as discussed in the previous chapter of the appendix – it would be easy to determine different numbers of replications for the different protocols if the pre-test indicates that for the protocols the minimal numbers of necessary replications differ – however this was not the case.

as Shapiro-Wilk normality tests indicate non-normal distribution of the outcome variables (with $p < 0.001$ for all samples) we decided to use the non-parametric Wilcoxon signed-rank exact test for this statistical analysis.[3] Only if there exist differences between the samples this test will result in significant p-values. Aggregated results for the outcome variables for the ten negotiation problems as well as statistical test are depicted in tables B.1 to B.11. As can be derived from these tables in none of the outcome dimensions, none of the protocols, and for none of the ten negotiation problems of our pre-test average outcomes differ significantly after three replications, this number of replications therefore is used in the simulation study.

B.1 `protocol 1`

Note that we do not calculate the proportion of agreements for the simulation runs with `protocol 1` as by the definition of this protocol all negotiations have to end with an agreement. We therefore omit a statistical test of differences in the proportions of agreement as they are equal for all agent combinations.

[3]We employ the `R` function `wilcox.exact` from the package `exactRankTests`. The 'exact' test is used to be able to calculate exact p-values for samples with ties.

ID	2381	3044	519	348	2771	848	1941	1541	617	304
Rep.	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)
2	21.79	38.02	34.05	14.17	22.26	37.95	51.81	44.73	33.91	31.58
	(14.94)	(33.27)	(19.64)	(6.94)	(10.78)	(27.34)	(27.12)	(35.38)	(14.52)	(25.65)
3	21.52	38.33	33.31	14.14	21.93	37.79	51.17	45.17	34.17	32.17
	(14.74)	(33.54)	(19.16)	(7.02)	(10.51)	(27.39)	(26.73)	(35.63)	(14.12)	(27.24)
4	21.85	38.14	33.20	14.16	21.99	37.51	51.35	44.75	34.58	31.77
	(15.03)	(33.59)	(19.37)	(7.00)	(10.70)	(27.10)	(26.66)	(35.21)	(14.62)	(26.05)
5	21.88	38.51	32.95	14.01	21.99	37.68	51.10	45.00	33.77	32.26
	(14.79)	(33.84)	(18.85)	(6.93)	(10.72)	(27.35)	(26.58)	(35.28)	(14.33)	(26.97)
	V	V	V	V	V	V	V	V	V	V
2 vs 3	1091.5	431.5	1013.0*	758.5	496.5**	456.5	663.5*	339.0**	795.0	372.0
3 vs 4	491.0	345.5	963.5	542.0	138.0	263.5	419.5	197.0	486.0	484.5
4 vs 5	600.0	185.5	737.5	537.5	90.0	92.0	274.5	80.0	725.5	204.5

*** $p < .001$, ** $p < .01$, * $p < .05$

Table B.1: Comparison of the average duration in `protocol 1` for different numbers of rep.

ID	2381	3044	519	348	2771	848	1941	1541	617	304
Rep.	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)
2	73.40	46.25	65.52	90.07	70.27	54.71	56.97	68.33	57.22	73.35
	(21.35)	(23.27)	(22.72)	(7.10)	(16.42)	(18.92)	(28.41)	(16.08)	(31.59)	(14.32)
3	73.74	46.05	65.35	90.02	69.80	54.51	58.31	68.17	57.98	73.19
	(22.16)	(23.50)	(22.91)	(6.77)	(17.01)	(18.73)	(28.10)	(16.12)	(31.43)	(14.57)
4	73.41	46.20	66.03	90.42	70.46	54.80	57.94	68.20	57.53	73.29
	(20.49)	(23.69)	(22.63)	(5.88)	(16.84)	(18.71)	(27.80)	(16.12)	(30.94)	(14.55)
5	73.26	46.23	65.52	90.02	70.08	54.69	57.98	68.14	56.94	72.86
	(21.32)	(23.92)	(22.75)	(5.74)	(16.21)	(18.73)	(27.79)	(16.10)	(30.79)	(14.30)
	V	V	V	V	V	V	V	V	V	V
2 vs 3	119.5	336.5	602.5	1117.5	64.5*	258.0	89.5	143.5	159.5	1030.0
3 vs 4	161.0	286.5	530.5	1037.0	153.0	189.0	146.5	130.5	314.5	822.5
4 vs 5	115.0	313.5	683.5	1304.5	68.0	360.0	148.0	146.5	304.0	1081.0

*** $p < .001$, ** $p < .01$, * $p < .05$

Table B.2: Comparison of the average utility to the seller in `protocol 1` for different numbers of replications

ID	2381	3044	519	348	2771	848	1941	1541	617	304
Rep.	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)
2	70.81	58.42	57.55	73.95	78.91	63.52	42.08	41.33	47.82	50.59
	(18.47)	(19.34)	(20.59)	(13.09)	(11.61)	(17.36)	(28.97)	(24.49)	(25.89)	(26.34)
3	70.39	58.32	57.48	74.28	78.53	63.63	41.25	41.37	46.99	49.81
	(18.47)	(19.56)	(20.29)	(13.42)	(11.63)	(17.19)	(29.09)	(24.15)	(25.14)	(27.03)
4	69.53	58.15	57.96	74.94	78.93	63.54	41.23	41.35	47.74	50.00
	(19.21)	(19.75)	(20.47)	(13.39)	(11.69)	(17.11)	(28.71)	(24.31)	(25.05)	(26.75)
5	70.13	58.14	57.99	74.64	78.56	63.55	41.32	41.40	47.87	49.68
	(18.55)	(19.98)	(19.98)	(13.13)	(12.03)	(17.13)	(28.37)	(24.20)	(24.73)	(26.80)
	V	V	V	V	V	V	V	V	V	V
2 vs 3	1047.5*	492.5	552.0	935.0	242.5	220.5	127.5	141.0	641.0	267.0
3 vs 4	878.0	500.5	471.0	1186.0	126.5	338.0	181.0	104.0	638.5	278.0
4 vs 5	666.0	485.0	683.0	1458.0	289.0	291.0	290.5	178.0	668.0	340.0

*** $p < .001$, ** $p < .01$, * $p < .05$

Table B.3: Comparison of the average utility to the buyer in `protocol 1` for different numbers of replications

B.2 `protocol 2`

Note that in `protocol 2` for the negotiation problems 1941 and 617 none of the agent combinations in none of the replications reached an agreement. Therefore the resulting utility to both the seller and the buyer party are zero. In negotiation problem 1541 only the combination of `MUMact` as buyer agent and `TFT` as seller agent resulted in exactly the same agreement in all replications. The equality of both, the proportion of agreements and the parties' utilities, renders the average outcomes for these three negotiation experiments and in these three outcome dimensions exactly equal so we omit the statistical test of differences.

ID	2381	3044	519	348	2771	848	1941	1541	617	304
Rep.	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)
2	11.11 (6.72)	6.85 (3.84)	6.30 (2.47)	13.89 (6.90)	14.57 (8.29)	6.73 (3.09)	5.44 (1.32)	5.38 (3.28)	4.11 (1.17)	5.77 (1.73)
3	10.48 (6.13)	6.53 (3.60)	6.01 (2.26)	13.91 (7.04)	14.31 (8.40)	7.17 (3.71)	5.38 (1.19)	5.25 (3.34)	4.02 (1.23)	5.31 (1.45)
4	10.42 (5.59)	6.58 (3.42)	6.22 (2.16)	14.07 (7.18)	14.79 (7.99)	7.04 (3.62)	5.25 (1.26)	5.37 (3.80)	3.90 (1.03)	5.52 (1.45)
5	10.98 (5.83)	6.31 (3.26)	6.14 (2.32)	13.83 (7.11)	14.14 (7.98)	7.04 (4.07)	5.12 (1.26)	5.27 (3.50)	3.83 (1.07)	5.38 (1.46)
	V	V	V	V	V	V	V	V	V	V
2 vs 3	1004.5	794.5	722.0	791.0	662.0	462.0*	384.0	179.5	434.0	898.0**
3 vs 4	855.5	428.0	364.0	490.0	639.0	479.5	327.0	38.5	177.0	485.0
4 vs 5	395.0	752.0	429.0	726.0	594.0	600.5	314.0	41.0	114.0	670.5

*** $p < .001$, ** $p < .01$, * $p < .05$

Table B.4: Comparison of the average duration in `protocol 2` for different numbers of replications

NegoID	2381	3044	519	348	2771	848	1941	1541	617	304
Replications	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)
2	0.17 (0.35)	0.09 (0.27)	0.04 (0.18)	0.93 (0.24)	0.44 (0.46)	0.09 (0.29)	0.00 (0.00)	0.01 (0.11)	0.00 (0.00)	0.02 (0.16)
3	0.20 (0.34)	0.07 (0.24)	0.03 (0.15)	0.96 (0.13)	0.40 (0.45)	0.08 (0.25)	0.00 (0.00)	0.01 (0.11)	0.00 (0.00)	0.02 (0.12)
4	0.20 (0.33)	0.07 (0.23)	0.02 (0.13)	0.95 (0.12)	0.42 (0.44)	0.10 (0.28)	0.00 (0.00)	0.01 (0.11)	0.00 (0.00)	0.03 (0.13)
5	0.20 (0.34)	0.07 (0.24)	0.04 (0.13)	0.95 (0.12)	0.42 (0.43)	0.10 (0.27)	0.00 (0.00)	0.01 (0.11)	0.00 (0.00)	0.02 (0.13)
	V	V	V	V	V	V	V	V	V	V
2 vs 3	57.5	9.0	14.0	24.0	165.5	14.0	*n.a.*	*n.a.*	*n.a.*	3.0
3 vs 4	128.0	8.0	16.0	72.0	113.0	4.0	*n.a.*	*n.a.*	*n.a.*	5.0
4 vs 5	177.0	6.0	8.0	140.0	137.0	11.0	*n.a.*	*n.a.*	*n.a.*	10.0

*** $p < .001$, ** $p < .01$, * $p < .05$

Table B.5: Comparison of the proportion of agreements in `protocol 2` for different numbers of replications

ID	2381	3044	519	348	2771	848	1941	1541	617	304
Rep.	⌀	⌀	⌀	⌀	⌀	⌀	⌀	⌀	⌀	⌀
	(±)	(±)	(±)	(±)	(±)	(±)	(±)	(±)	(±)	(±)
2	12.72	4.72	3.33	82.98	32.33	5.50	0.00	0.85	0.00	1.46
	(26.32)	(14.88)	(14.33)	(22.23)	(34.26)	(16.99)	(0.00)	(7.67)	(0.00)	(9.25)
3	15.00	4.01	2.15	86.88	29.54	5.00	0.00	0.85	0.00	1.23
	(26.77)	(13.11)	(9.83)	(12.27)	(34.01)	(15.21)	(0.00)	(7.67)	(0.00)	(7.29)
4	14.95	3.84	1.80	86.02	30.66	6.01	0.00	0.85	0.00	1.66
	(26.61)	(12.73)	(9.66)	(11.28)	(32.85)	(16.62)	(0.00)	(7.67)	(0.00)	(7.83)
5	15.32	3.89	2.61	85.05	30.96	6.00	0.00	0.85	0.00	1.47
	(24.97)	(13.09)	(9.79)	(11.19)	(32.45)	(16.25)	(0.00)	(7.67)	(0.00)	(7.87)
	V	V	V	V	V	V	V	V	V	V
2 vs 3	62.0	12.0	14.0	812.5	261.0	23.0	*n.a.*	*n.a.*	*n.a.*	3.0
3 vs 4	180.0	9.0	18.0	1222.0	254.0	13.0	*n.a.*	*n.a.*	*n.a.*	8.0
4 vs 5	229.5	6.0	10.0	1548.0	246.0	23.5	*n.a.*	*n.a.*	*n.a.*	13.0

*** $p < .001$, ** $p < .01$, * $p < .05$

Table B.6: Comparison of the average utility to the seller in `protocol 2` for different numbers of replications

ID	2381	3044	519	348	2771	848	1941	1541	617	304
Rep.	⌀	⌀	⌀	⌀	⌀	⌀	⌀	⌀	⌀	⌀
	(±)	(±)	(±)	(±)	(±)	(±)	(±)	(±)	(±)	(±)
2	12.22	5.10	2.66	69.54	34.85	6.01	0.00	0.62	0.00	1.85
	(24.54)	(15.99)	(11.09)	(22.38)	(36.93)	(18.90)	(0.00)	(5.56)	(0.00)	(11.71)
3	14.34	4.41	1.98	72.00	30.82	5.23	0.00	0.62	0.00	1.54
	(25.44)	(14.32)	(8.65)	(16.88)	(35.62)	(16.20)	(0.00)	(5.56)	(0.00)	(9.14)
4	14.00	4.22	1.46	71.30	32.01	6.66	0.00	0.62	0.00	2.08
	(23.88)	(13.90)	(7.67)	(16.80)	(34.40)	(18.68)	(0.00)	(5.56)	(0.00)	(9.83)
5	14.14	4.26	2.20	71.25	32.57	6.64	0.00	0.62	0.00	1.85
	(24.33)	(14.35)	(8.01)	(17.11)	(34.01)	(18.08)	(0.00)	(5.56)	(0.00)	(9.89)
	V	V	V	V	V	V	V	V	V	V
2 vs 3	73.0	12.0	13.0	826.0	300.0	21.0	*n.a.*	*n.a.*	*n.a.*	3.0
3 vs 4	220.0	8.5	21.0	1370.0	246.5	9.0	*n.a.*	*n.a.*	*n.a.*	5.0
4 vs 5	270.0	6.0	9.0	1427.0	247.0	20.0	*n.a.*	*n.a.*	*n.a.*	10.0

*** $p < .001$, ** $p < .01$, * $p < .05$

Table B.7: Comparison of the average utility to the buyer in `protocol 2` for different numbers of replications

B.3 `protocol 3`

In `protocol 3` all agent combinations in all replications failed to reach an agreement for negotiation problem 1941. Therefore statistical tests of the proportion of agreements and the parties' utilities are omitted for this negotiation problem. Moreover virtually all agent combinations reached an agreement in negotiation 348, only in one of the replications in the four replications setting were `MUMact` represented the buyer agent and `SMCact` was the seller agent no agreement could be reached. In the settings with two, three, and five replications the proportion of agreements therefore was exactly the same and the statistical test of differences was omitted.

ID	2381	3044	519	348	2771	848	1941	1541	617	304
Rep.	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)
2	24.98 (17.23)	36.09 (35.94)	42.48 (28.53)	14.02 (6.97)	21.01 (11.48)	34.80 (30.39)	18.33 (15.22)	29.21 (43.06)	10.20 (9.64)	37.69 (37.84)
3	25.30 (16.73)	38.48 (39.62)	41.93 (27.15)	14.30 (7.03)	20.46 (11.66)	36.32 (30.56)	19.48 (15.04)	29.85 (44.40)	10.19 (7.72)	38.23 (36.90)
4	25.11 (16.90)	35.57 (36.40)	40.64 (26.21)	14.07 (6.93)	20.53 (12.73)	35.86 (29.67)	18.32 (14.97)	28.43 (41.96)	11.10 (10.27)	38.58 (36.50)
5	25.02 (16.94)	37.60 (39.69)	40.79 (25.63)	14.21 (6.92)	20.57 (11.59)	34.91 (29.37)	18.33 (14.33)	29.27 (44.11)	11.40 (9.16)	36.70 (35.90)
	V	V	V	V	V	V	V	V	V	V
2 vs 3	872.5	619.0*	1422.5*	608.0	607.0	512.5	819.5	527.5	574.5	772.0
3 vs 4	778.0	451.5	1186.0	961.0	438.5	439.0	883.5	607.0	475.0	1088.5
4 vs 5	692.0	574.0	1019.5	616.5	241.0	434.0	570.0	387.5	365.0	1357.5

*** $p < .001$, ** $p < .01$, * $p < .05$

Table B.8: Comparison of the average duration in `protocol 3` for different numbers of replications

ID	2381	3044	519	348	2771	848	1941	1541	617	304
Rep.	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)	⌀ (±)
2	0.95 (0.19)	0.43 (0.48)	0.78 (0.36)	1.00 (0.00)	0.91 (0.23)	0.59 (0.48)	0.00 (0.00)	0.19 (0.37)	0.01 (0.08)	0.67 (0.34)
3	0.95 (0.16)	0.44 (0.47)	0.77 (0.36)	1.00 (0.00)	0.87 (0.26)	0.60 (0.45)	0.00 (0.00)	0.19 (0.38)	0.02 (0.07)	0.74 (0.31)
4	0.96 (0.14)	0.44 (0.47)	0.76 (0.37)	1.00 (0.03)	0.86 (0.30)	0.60 (0.44)	0.00 (0.00)	0.18 (0.36)	0.02 (0.08)	0.71 (0.27)
5	0.97 (0.12)	0.43 (0.46)	0.76 (0.36)	1.00 (0.00)	0.89 (0.23)	0.59 (0.44)	0.00 (0.00)	0.19 (0.38)	0.03 (0.09)	0.68 (0.30)
	V	V	V	V	V	V	V	V	V	V
2 vs 3	18.0	47.0	204.5	*n.a.*	143.0	84.0	*n.a.*	11.5	6.0	547.0
3 vs 4	22.0	51.0	183.0	1.0	109.5	103.5	*n.a.*	21.0	22.0	976.5
4 vs 5	19.5	112.5	182.5	0.0	62.5	197.5	*n.a.*	10.5	49.0	1094.5

*** $p < .001$, ** $p < .01$, * $p < .05$

Table B.9: Comparison of the proportion of agreements in `protocol 3` for different numbers of replications

ID	2381	3044	519	348	2771	848	1941	1541	617	304
Rep.	∅ (±)	∅ (±)	∅ (±)	∅ (±)	∅ (±)	∅ (±)	∅ (±)	∅ (±)	∅ (±)	∅ (±)
2	70.37 (21.99)	23.12 (25.90)	56.48 (27.96)	90.54 (6.34)	66.52 (19.69)	34.99 (28.75)	0.00 (0.00)	12.12 (24.57)	0.74 (4.68)	43.54 (23.71)
3	71.26 (21.45)	23.72 (25.35)	54.78 (27.62)	90.17 (6.39)	63.28 (20.89)	35.84 (27.03)	0.00 (0.00)	12.65 (24.83)	0.99 (4.36)	48.93 (22.20)
4	71.23 (20.19)	23.98 (25.34)	53.47 (28.77)	89.85 (6.37)	62.82 (23.70)	35.14 (26.27)	0.00 (0.00)	11.50 (23.65)	1.36 (5.24)	46.85 (19.46)
5	71.27 (19.67)	23.30 (24.73)	54.50 (27.28)	90.37 (5.74)	65.10 (19.29)	34.64 (26.73)	0.00 (0.00)	12.28 (24.87)	2.07 (5.30)	44.55 (21.25)
	V	V	V	V	V	V	V	V	V	V
2 vs 3	167.0	73.0	847.5	957.5	311.0	141.0	*n.a.*	11.5	6.0	775.0
3 vs 4	267.5	100.0	823.5	1384.5	251.5	174.5	*n.a.*	21.0	22.0	1420.0
4 vs 5	285.0	180.0	617.0	1082.5	228.0	255.0	*n.a.*	9.5	49.0	1495.0

*** $p < .001$, ** $p < .01$, * $p < .05$

Table B.10: Comparison of the average utility to the seller in `protocol 3` for different numbers of replications

ID	2381	3044	519	348	2771	848	1941	1541	617	304
Rep.	∅ (±)	∅ (±)	∅ (±)	∅ (±)	∅ (±)	∅ (±)	∅ (±)	∅ (±)	∅ (±)	∅ (±)
2	69.47 (18.08)	24.94 (28.17)	50.40 (24.55)	75.12 (13.28)	71.87 (21.49)	38.11 (31.09)	0.00 (0.00)	10.14 (20.68)	0.76 (4.83)	44.57 (24.50)
3	69.83 (15.42)	26.05 (28.03)	49.93 (24.34)	75.10 (13.11)	68.68 (22.96)	38.71 (29.49)	0.00 (0.00)	10.56 (21.03)	1.02 (4.50)	49.38 (21.82)
4	70.54 (15.65)	26.13 (27.89)	48.37 (25.01)	74.18 (13.32)	67.68 (25.47)	39.08 (29.56)	0.00 (0.00)	9.66 (20.02)	1.23 (4.61)	47.21 (19.29)
5	71.42 (13.89)	25.52 (27.17)	49.29 (23.92)	74.41 (12.90)	69.74 (21.02)	37.75 (29.57)	0.00 (0.00)	10.24 (20.82)	1.98 (5.24)	44.74 (21.61)
	V	V	V	V	V	V	V	V	V	V
2 vs 3	752.0	62.5	733.0	1258.5	329.5	205.0	*n.a.*	17.5	6.0	549.5
3 vs 4	731.0	108.0	821.5	1841.0	309.0	198.5	*n.a.*	31.0	22.0	1155.5
4 vs 5	752.5	177.5	760.0	1520.5	278.0	394.0	*n.a.*	21.0	49.0	1287.0

*** $p < .001$, ** $p < .01$, * $p < .05$

Table B.11: Comparison of the average utility to the buyer in `protocol 3` for different numbers of replications

Appendix C

Summary Tables

This appendix provides result tables with additional results for Chapter 6 that do not fit there. Table C.1 provides results for the outcome variables proportion of agreements (prop. agr.), proportion of Pareto-optimal agreements (prop. eff.), minimal distance to the Pareto frontier (distance), the utility of the agreement to the seller (u. seller) and buyer (u. buyer) and the contract imbalance as a measure of fairness (imbalance) for the system configurations indicated in the first three columns which specify the interaction protocol (protocol) as well as the software agents used for representing the seller (seller) and buyer (buyer) party. While for the first two outcome variables proportions of reached agreements and reached Pareto-optimal agreements proportions of the total number of simulation runs with this system configuration are reported, for the remaining four outcome variables average values for the subset of simulation runs that reached an agreement are provided as central tendency. The sample sizes therefore are $2,065 * 3 = 6,195$ for the proportions and $6,195 * \textit{prop.agr.}$ for all other outcome variables.

At the bottom of each part of the table we also provide the results of the negotiation experiments between humans, used as benchmark for the evaluation of the different systems for automated negotiation – here the sample size is 2,065. Indicators for the significance values for one-sided tests that automated negotiation systems reach better results than human negotiators in the experiments – i.e. that they reach more agreements and more Pareto-efficient agreements, agreements of lower distance to the Pareto frontier, lower contract imbalance and higher utility to the parties – are reported with the results. However, we omit the test statistics of the tests – χ^2 for Pearson's χ^2 test of independence to test hypotheses concerning proportions and W for the Wilcoxon rank sum test employed to test the remaining of the above mentioned hypotheses – due to space restrictions, the interested reader is provided with this information upon request. Furthermore note that * indicates significance at $p < 0.05$, ** at $p < 0.01$, and $***$ at $p < 0.001$, all referring to p-values adjusted by the Bonferroni-Holm method to control for the family wise error rate in the multiple comparisons.

protocol	seller	buyer	prop. agr.	prop. eff.	distance	u. seller	u. buyer	imbalance
1	MOCact	MOCact	100.00***	81.97***	0.92***	73.12***	68.40	11.61***
1	MOCact	MOCpas	100.00***	82.28***	0.94***	73.06***	68.40	11.57***
1	MOCact	SMCact	100.00***	52.36***	3.41	83.47***	49.56	35.07
1	MOCact	SMCpas	100.00***	52.03***	3.51	83.59***	49.16	35.46
1	MOCact	MUMact	100.00***	42.31***	5.80	90.03***	34.06	56.36
1	MOCact	MUMpas	100.00***	39.79***	6.31	90.04***	32.89	57.51
1	MOCact	LEXact	100.00***	43.42***	5.03	84.95***	44.44	40.97
1	MOCact	LEXpas	100.00***	42.44***	5.38	84.94***	43.74	41.63
1	MOCact	TFT	100.00***	84.37***	1.07***	83.46***	54.42	29.22
1	MOCpas	MOCact	100.00***	81.82***	0.98***	73.06***	68.37	11.56***
1	MOCpas	MOCpas	100.00***	82.18***	0.94***	73.10***	68.38	11.63***
1	MOCpas	SMCact	100.00***	52.62***	3.42	83.56***	49.51	35.13
1	MOCpas	SMCpas	100.00***	52.96***	3.44	83.58***	49.30	35.34
1	MOCpas	MUMact	100.00***	42.08***	5.74	90.02***	34.16	56.22
1	MOCpas	MUMpas	100.00***	40.02***	6.31	90.00***	33.00	57.40
1	MOCpas	LEXact	100.00***	43.91***	5.03	84.99***	44.43	40.98
1	MOCpas	LEXpas	100.00***	42.65***	5.29	84.99***	43.85	41.60
1	MOCpas	TFT	100.00***	84.63***	1.05***	83.48***	54.38	29.30
1	SMCact	MOCact	100.00***	57.40***	3.05	55.62	79.75***	26.81
1	SMCact	MOCpas	100.00***	56.97***	3.03	55.53	79.82***	26.95
1	SMCact	SMCact	100.00***	43.42***	4.61	69.33***	63.97	25.59
1	SMCact	SMCpas	100.00***	43.41***	4.53	69.56***	63.78	25.97
1	SMCact	MUMact	100.00***	37.11	6.31	79.74***	48.19	37.65
1	SMCact	MUMpas	100.00***	36.50	6.75	79.99***	46.83	39.09
1	SMCact	LEXact	100.00***	35.04	6.09	71.23***	59.87	24.43
1	SMCact	LEXpas	100.00***	33.72	6.39	71.17***	59.30	24.94
1	SMCact	TFT	100.00***	54.74***	3.30	78.68***	57.14	23.92
1	SMCpas	MOCact	100.00***	57.14***	3.05	55.33	79.86***	27.28
1	SMCpas	MOCpas	100.00***	57.40***	3.11	55.39	79.80***	27.12
1	SMCpas	SMCact	100.00***	43.26***	4.58	69.19***	64.12	25.65
1	SMCpas	SMCpas	100.00***	43.87***	4.60	69.35***	63.85	26.06
1	SMCpas	MUMact	100.00***	36.87	6.33	79.71***	48.15	37.89
1	SMCpas	MUMpas	100.00***	35.72	6.78	79.83***	46.98	39.22
1	SMCpas	LEXact	100.00***	35.74	6.08	71.12***	59.99	24.63
1	SMCpas	LEXpas	100.00***	34.16	6.41	71.02***	59.42	25.27
1	SMCpas	TFT	100.00***	54.74***	3.31	78.68***	57.08	24.03
1	MUMact	MOCact	100.00***	52.45***	4.55	40.50	87.58***	48.01
1	MUMact	MOCpas	100.00***	51.91***	4.57	40.48	87.61***	48.06
1	MUMact	SMCact	100.00***	43.52***	5.40	55.55	75.39***	31.99
1	MUMact	SMCpas	100.00***	42.84***	5.52	55.46	75.30***	32.13
1	MUMact	MUMact	100.00***	38.24	7.55	69.35	61.11	24.07
Control (human negotiation exp.)			69.78	34.24	5.24	67.93	67.42	20.40

continued on next page

protocol	seller	buyer	prop. agr.	prop. eff.	distance	u. seller	u. buyer	imbalance
1	MUMact	MUMpas	100.00***	37.05	7.93	69.35	60.18	25.25
1	MUMact	LEXact	100.00***	36.34	7.37	58.55	71.43***	25.18
1	MUMact	LEXpas	100.00***	35.72	7.47	58.48	71.30***	25.27
1	MUMact	TFT	100.00***	46.57***	5.17	75.16***	58.17	23.88
1	MUMpas	MOCact	100.00***	51.38***	5.06	39.22	87.64***	49.35
1	MUMpas	MOCpas	100.00***	51.36***	4.93	39.50	87.64***	49.13
1	MUMpas	SMCact	100.00***	42.23***	5.90	54.24	75.76***	33.32
1	MUMpas	SMCpas	100.00***	42.03***	5.98	54.26	75.54***	33.55
1	MUMpas	MUMact	100.00***	37.47	7.85	68.73	61.09	24.99
1	MUMpas	MUMpas	100.00***	35.90	8.25	68.62	60.30	26.67
1	MUMpas	LEXact	100.00***	35.79	7.67	57.49	71.75***	26.54
1	MUMpas	LEXpas	100.00***	35.87	7.80	57.34	71.62***	26.78
1	MUMpas	TFT	100.00***	46.52***	5.30	74.93***	58.22	24.08
1	LEXact	MOCact	100.00***	48.05***	4.54	50.04	82.13***	33.03
1	LEXact	MOCpas	100.00***	49.22***	4.47	50.14	82.14***	32.97
1	LEXact	SMCact	100.00***	38.89*	5.47	64.77	68.26	21.16
1	LEXact	SMCpas	100.00***	37.84	5.67	64.67	67.99	21.43
1	LEXact	MUMact	100.00***	33.85	7.91	75.38***	53.20	28.24
1	LEXact	MUMpas	100.00***	33.32	8.41	75.49***	52.04	29.62
1	LEXact	LEXact	100.00***	27.09	9.00	65.49	62.81	16.94***
1	LEXact	LEXpas	100.00***	27.15	9.16	65.65	62.33	17.49***
1	LEXact	TFT	100.00***	39.69***	6.06	74.72***	57.57	21.18
1	LEXpas	MOCact	100.00***	46.47***	4.89	49.34	82.12***	33.73
1	LEXpas	MOCpas	100.00***	46.41***	4.87	49.34	82.15***	33.77
1	LEXpas	SMCact	100.00***	38.29	5.67	64.48	68.18	21.53
1	LEXpas	SMCpas	100.00***	37.61	5.76	64.44	68.04	21.83
1	LEXpas	MUMact	100.00***	33.82	7.78	75.31***	53.37	28.22
1	LEXpas	MUMpas	100.00***	32.64	8.32	75.38***	52.31	29.37
1	LEXpas	LEXact	100.00***	26.99	9.10	65.15	62.95	17.26***
1	LEXpas	LEXpas	100.00***	26.96	9.34	65.13	62.44	17.84***
1	LEXpas	TFT	100.00***	39.56***	6.32	74.47***	57.48	21.36
1	TFT	MOCact	100.00***	85.88***	0.96***	54.96	83.05***	28.37
1	TFT	MOCpas	100.00***	85.49***	0.97***	54.96	83.02***	28.34
1	TFT	SMCact	100.00***	57.56***	3.25	57.38	78.28***	23.75
1	TFT	SMCpas	100.00***	58.13***	3.19	57.40	78.39***	23.83
1	TFT	MUMact	100.00***	47.46***	5.36	58.28	74.58***	23.24
1	TFT	MUMpas	100.00***	47.34***	5.61	58.32	74.10***	23.67
1	TFT	LEXact	100.00***	38.76*	6.33	58.07	73.75***	20.19
1	TFT	LEXpas	100.00***	38.26	6.53	57.96	73.61***	20.30
1	TFT	TFT	100.00***	37.21	6.97	66.04	66.04	15.80***
2	MOCact	MOCact	10.33	9.14	0.56***	87.97***	86.21***	7.49***
2	MOCact	MOCpas	10.12	8.81	0.64***	88.15***	86.29***	7.36***
Control (human negotiation exp.)			69.78	34.24	5.24	67.93	67.42	20.40

continued on next page

protocol	seller	buyer	prop. agr.	prop. eff.	distance	u. seller	u. buyer	imbalance
2	MOCact	SMCact	12.40	9.06	1.54***	90.36***	77.00***	15.18***
2	MOCact	SMCpas	11.25	7.99	1.59***	90.74***	78.18***	14.30***
2	MOCact	MUMact	16.50	9.69	2.60*	91.10***	67.44	24.31
2	MOCact	MUMpas	12.09	7.52	2.24***	91.37***	72.98***	19.61
2	MOCact	LEXact	11.72	8.22	1.90***	91.21***	75.29***	16.79*
2	MOCact	LEXpas	10.48	7.44	1.85***	91.39***	76.46***	15.98**
2	MOCact	TFT	15.98	12.03	2.27***	96.47***	61.26	35.22
2	MOCpas	MOCact	10.04	8.59	0.69***	88.15***	86.26***	7.30***
2	MOCpas	MOCpas	9.88	8.52	0.70***	88.28***	86.36***	7.18***
2	MOCpas	SMCact	12.19	8.93	1.37***	90.83***	77.01***	15.56***
2	MOCpas	SMCpas	10.83	7.81	1.56***	90.73***	78.68***	13.91***
2	MOCpas	MUMact	14.19	8.17	2.78	91.59***	68.58	23.70
2	MOCpas	MUMpas	11.20	6.65	2.49	91.83***	73.17***	19.65
2	MOCpas	LEXact	10.90	7.64	1.75***	91.40***	75.50***	16.89*
2	MOCpas	LEXpas	9.91	6.86	1.81***	91.60***	77.24***	15.39***
2	MOCpas	TFT	15.08	11.44	2.26***	96.78***	60.90	35.90
2	SMCact	MOCact	13.67	10.07	1.35***	78.50***	87.87***	12.12***
2	SMCact	MOCpas	12.64	9.27	1.35***	78.94***	88.12***	11.89***
2	SMCact	SMCact	19.61	11.25	2.64	79.58***	77.44***	12.86***
2	SMCact	SMCpas	16.08	9.10	2.50	80.86***	79.08***	11.69***
2	SMCact	MUMact	26.30	12.72	3.56	80.43***	69.35	16.76**
2	SMCact	MUMpas	18.51	9.35	3.33	81.12***	74.27***	14.34***
2	SMCact	LEXact	18.39	9.64	3.71	79.89***	74.93***	13.05***
2	SMCact	LEXpas	14.77	7.65	3.71	80.99***	76.94***	12.40***
2	SMCact	TFT	21.84	14.32	2.88***	92.21***	60.09	32.29
2	SMCpas	MOCact	12.19	8.86	1.37***	79.75***	87.95***	11.03***
2	SMCpas	MOCpas	11.69	8.43	1.35***	79.94***	88.14***	10.93***
2	SMCpas	SMCact	16.80	9.70	2.64	82.11***	77.94***	12.85***
2	SMCpas	SMCpas	14.29	8.80	2.22*	83.55***	80.40***	11.26***
2	SMCpas	MUMact	20.44	10.27	3.53	82.53***	70.43	17.10**
2	SMCpas	MUMpas	14.74	7.52	3.43	84.19***	75.64***	14.56***
2	SMCpas	LEXact	14.95	8.09	3.61	82.27***	76.19***	12.75***
2	SMCpas	LEXpas	12.51	6.76	3.22	83.60***	78.38***	12.26***
2	SMCpas	TFT	18.48	12.57	2.74***	93.66***	60.44	33.27
2	MUMact	MOCact	18.19	10.80	2.54*	69.87	88.73***	20.51
2	MUMact	MOCpas	16.55	9.91	2.78	70.46	89.33***	20.27
2	MUMact	SMCact	29.65	15.11	3.26	71.65**	78.16***	15.38***
2	MUMact	SMCpas	22.28	11.27	3.49	73.53***	80.22***	14.95***
2	MUMact	MUMact	50.96	27.31	3.62	72.35***	68.80	14.06***
2	MUMact	MUMpas	32.72	17.40	3.53	72.66***	72.28***	12.95***
2	MUMact	LEXact	33.03	15.69	4.28	71.37	73.76***	13.40***
2	MUMact	LEXpas	22.57	10.83	4.26	73.23***	76.30***	13.42***
Control (human negotiation exp.)			69.78	34.24	5.24	67.93	67.42	20.40

continued on next page

protocol	seller	buyer	prop. agr.	prop. eff.	distance	u. seller	u. buyer	imbalance
2	MUMact	TFT	46.83	28.86	2.67***	82.82***	57.34	26.14
2	MUMpas	MOCact	14.29	8.94	2.44*	73.66***	88.88***	16.97*
2	MUMpas	MOCpas	13.54	8.35	2.35*	74.39***	89.25***	16.72*
2	MUMpas	SMCact	21.42	10.67	3.17	76.06***	78.57***	13.57***
2	MUMpas	SMCpas	16.63	8.23	3.40	78.18***	80.80***	13.68***
2	MUMpas	MUMact	33.51	17.34	3.69	75.65***	69.10	14.01***
2	MUMpas	MUMpas	19.82	9.18	4.31	78.46***	75.03***	13.29***
2	MUMpas	LEXact	22.50	10.49	4.34	76.18***	74.49***	12.71***
2	MUMpas	LEXpas	15.85	7.36	4.70	78.51***	77.99***	12.73***
2	MUMpas	TFT	31.36	21.23	2.44***	85.91***	57.69	28.42
2	LEXact	MOCact	14.24	9.36	2.04***	75.51***	88.95***	14.98***
2	LEXact	MOCpas	13.16	9.01	1.92***	75.40***	89.39***	15.45***
2	LEXact	SMCact	21.08	10.90	3.70	76.21***	77.19***	11.97***
2	LEXact	SMCpas	17.16	8.88	3.72	77.81***	79.59***	11.92***
2	LEXact	MUMact	33.03	15.09	4.44	76.75***	69.18	14.45***
2	LEXact	MUMpas	22.39	10.72	4.06	77.44***	73.22***	12.60***
2	LEXact	LEXact	21.99	9.28	5.50	74.80***	72.72***	10.77***
2	LEXact	LEXpas	16.38	7.38	5.13	77.01***	75.63***	10.60***
2	LEXact	TFT	30.33	17.59	3.62	88.73***	58.70	30.20
2	LEXpas	MOCact	12.38	8.35	2.03***	76.36***	89.11***	14.54***
2	LEXpas	MOCpas	11.83	8.17	1.95***	76.54***	89.35***	14.35***
2	LEXpas	SMCact	17.01	9.06	3.34	78.81***	78.78***	11.84***
2	LEXpas	SMCpas	14.50	7.94	3.32	79.41***	81.86***	11.68***
2	LEXpas	MUMact	23.36	10.72	4.48	78.74***	70.66	14.33***
2	LEXpas	MUMpas	16.30	7.89	4.17	80.63***	76.17***	12.39***
2	LEXpas	LEXact	16.51	7.23	5.10	78.14***	74.80***	10.62***
2	LEXpas	LEXpas	13.20	6.20	5.05	79.68***	77.16***	10.14***
2	LEXpas	TFT	23.83	14.14	3.47	90.70***	59.23	31.62
2	TFT	MOCact	17.27	14.19	1.65***	58.93	95.81***	36.92
2	TFT	MOCpas	15.61	12.74	1.84***	58.81	96.11***	37.32
2	TFT	SMCact	23.91	16.71	2.47***	58.13	90.73***	32.90
2	TFT	SMCpas	20.50	14.83	2.33***	58.57	92.68***	34.31
2	TFT	MUMact	52.66	33.69	2.64***	56.89	81.93***	25.82
2	TFT	MUMpas	36.35	25.38	2.28***	57.22	85.38***	28.44
2	TFT	LEXact	29.57	16.61	3.94	58.01	87.66***	30.05
2	TFT	LEXpas	22.97	13.58	3.76	58.35	89.13***	30.94
2	TFT	TFT	75.93***	26.02	7.55	66.90	66.54	16.41***
3	MOCact	MOCact	79.82***	66.44***	0.86***	73.71***	70.86*	8.12***
3	MOCact	MOCpas	74.04*	61.50***	0.86***	74.17***	71.61***	8.07***
3	MOCact	SMCact	82.57***	60.69***	1.45***	79.63***	62.26	18.26
3	MOCact	SMCpas	75.32***	53.45***	1.51***	79.87***	63.46	17.43
3	MOCact	MUMact	91.85***	64.47***	1.81***	82.65***	55.68	27.43
Control (human negotiation exp.)			69.78	34.24	5.24	67.93	67.42	20.40

continued on next page

protocol	seller	buyer	prop. agr.	prop. eff.	distance	u. seller	u. buyer	imbalance
3	MOCact	MUMpas	81.24***	60.19***	1.51***	81.39***	60.47	21.71
3	MOCact	LEXact	86.36***	60.27***	1.68***	80.74***	60.11	21.26
3	MOCact	LEXpas	75.06***	52.49***	1.69***	81.41***	61.70	20.20
3	MOCact	TFT	73.20	62.31***	1.06***	86.75***	55.34	31.47
3	MOCpas	MOCact	75.24***	62.79***	0.89***	74.22***	71.14***	8.06***
3	MOCpas	MOCpas	50.93	42.44***	0.93***	76.89***	73.73***	8.48***
3	MOCpas	SMCact	79.24***	59.00***	1.34***	80.02***	62.35	18.46
3	MOCpas	SMCpas	49.28	34.33	1.59***	82.70***	65.72	18.03
3	MOCpas	MUMact	88.59***	62.62***	1.79***	83.02***	55.70	27.72
3	MOCpas	MUMpas	52.69	37.09	1.77***	84.23***	62.40	22.67
3	MOCpas	LEXact	83.47***	56.98***	1.79***	81.16***	59.91	21.75
3	MOCpas	LEXpas	49.86	33.74	1.83***	83.89***	63.79	20.66
3	MOCpas	TFT	58.42	49.06***	1.14***	87.79***	55.57	32.25
3	SMCact	MOCact	80.27***	58.29***	1.46***	65.19	77.63***	14.73***
3	SMCact	MOCpas	76.00***	54.96***	1.46***	65.22	78.10***	14.92***
3	SMCact	SMCact	79.55***	40.65***	3.19	71.51***	68.92	13.94***
3	SMCact	SMCpas	71.69	36.17	3.32	71.49***	70.16	13.32***
3	SMCact	MUMact	89.02***	41.28***	3.95	75.85***	61.38	19.00
3	SMCact	MUMpas	76.51***	37.69	3.62	75.08***	65.44	15.65***
3	SMCact	LEXact	80.32***	36.24	4.26	72.56***	65.95	14.14***
3	SMCact	LEXpas	68.25	30.80	4.30	72.87***	67.65	13.54***
3	SMCact	TFT	75.58***	44.00***	2.99	82.52***	57.01	26.12
3	SMCpas	MOCact	76.21***	53.77***	1.52***	66.09	77.57***	13.97***
3	SMCpas	MOCpas	47.25	32.83	1.62***	68.85	80.34***	14.13***
3	SMCpas	SMCact	74.25***	38.43	3.15	72.69***	68.86	13.32***
3	SMCpas	SMCpas	44.10	22.07	3.35	75.08***	72.24***	12.90***
3	SMCpas	MUMact	83.42***	39.48***	3.92	76.86***	61.26	19.19
3	SMCpas	MUMpas	46.94	21.73	3.99	77.77***	67.77	15.86***
3	SMCpas	LEXact	75.17***	32.87	4.40	73.62***	65.55	14.05***
3	SMCpas	LEXpas	42.44	19.24	4.57	76.18***	69.37	13.47***
3	SMCpas	TFT	56.43	33.48	2.97	84.17***	57.45	27.24
3	MUMact	MOCact	91.35***	64.07***	1.89***	57.84	81.22***	24.36
3	MUMact	MOCpas	85.71***	60.21***	1.93***	57.97	81.80***	24.73
3	MUMact	SMCact	88.05***	44.47***	3.88	64.17	73.86***	16.88**
3	MUMact	SMCpas	80.69***	40.69***	3.71	64.29	75.11***	17.15**
3	MUMact	MUMact	89.48***	40.32***	4.98	70.41	66.30	15.16***
3	MUMact	MUMpas	78.85***	37.09	4.69	69.97	69.34	13.91***
3	MUMact	LEXact	87.12***	36.46	5.59	65.96	69.89	14.31***
3	MUMact	LEXpas	76.56***	31.11	5.61	66.06	71.36***	14.47***
3	MUMact	TFT	81.44***	42.73***	3.76	79.81***	57.81	23.88
3	MUMpas	MOCact	82.47***	61.92***	1.53***	62.34	79.89***	18.99
3	MUMpas	MOCpas	52.32	36.19	1.81***	64.90	82.48***	19.25
Control (human negotiation exp.)			69.78	34.24	5.24	67.93	67.42	20.40

continued on next page

protocol	seller	buyer	prop. agr.	prop. eff.	distance	u. seller	u. buyer	imbalance
3	MUMpas	SMCact	78.47***	40.71***	3.43	67.87	72.90***	14.24***
3	MUMpas	SMCpas	47.39	23.23	3.88	70.94	75.48***	14.46***
3	MUMpas	MUMact	81.50***	38.82*	4.62	72.97***	65.66	15.37***
3	MUMpas	MUMpas	47.30	19.73	5.29	74.52***	70.56	14.58***
3	MUMpas	LEXact	76.01***	32.66	5.25	69.58	68.93	12.42***
3	MUMpas	LEXpas	45.00	16.82	6.20	72.13***	71.67***	13.03***
3	MUMpas	TFT	58.55	31.93	3.55	82.75***	57.90	25.73
3	LEXact	MOCact	85.15***	58.00***	1.72***	62.17	79.15***	17.96
3	LEXact	MOCpas	80.26***	54.79***	1.71***	62.40	79.80***	18.31
3	LEXact	SMCact	81.39***	36.84	4.32	68.05	70.58*	12.60***
3	LEXact	SMCpas	73.25	33.93	4.22	68.02	71.95***	12.38***
3	LEXact	MUMact	87.57***	34.83	5.73	72.15***	63.22	15.09***
3	LEXact	MUMpas	74.71***	31.28	5.25	71.30***	67.73	12.51***
3	LEXact	LEXact	78.63***	24.94	7.16	67.90	66.65	11.27***
3	LEXact	LEXpas	65.68	20.76	7.19	68.46	68.40	10.84***
3	LEXact	TFT	71.49	31.75	5.13	80.12***	57.52	23.53
3	LEXpas	MOCact	75.58***	52.15***	1.66***	63.88	79.61***	16.95*
3	LEXpas	MOCpas	48.14	31.95	1.86***	66.02	82.05***	17.31
3	LEXpas	SMCact	70.99	32.15	4.27	69.75	70.82***	12.42***
3	LEXpas	SMCpas	43.23	20.39	4.52	71.77***	73.95***	12.15***
3	LEXpas	MUMact	79.27***	31.91	5.73	73.37***	63.41	15.30***
3	LEXpas	MUMpas	43.87	16.13	6.18	73.95***	69.86	12.50***
3	LEXpas	LEXact	67.99	22.13	7.08	69.69	66.81	11.16***
3	LEXpas	LEXpas	39.73	13.46	7.14	71.87***	70.57	10.66***
3	LEXpas	TFT	51.62	23.37	5.13	82.24***	57.90	25.08
3	TFT	MOCact	75.27***	64.58***	1.17***	55.33	85.69***	30.92
3	TFT	MOCpas	60.13	51.32***	1.07***	55.93	86.82***	31.62
3	TFT	SMCact	77.37***	47.68***	2.94	57.16	81.41***	25.62
3	TFT	SMCpas	56.22	35.80	2.85***	57.57	83.45***	27.02
3	TFT	MUMact	80.82***	43.10***	4.01	58.06	78.86***	23.35
3	TFT	MUMpas	57.26	32.20	3.79	58.04	81.94***	25.56
3	TFT	LEXact	70.25	30.15	5.54	58.29	78.47***	22.61
3	TFT	LEXpas	49.85	22.50	5.43	58.80	81.16***	24.59
3	TFT	TFT	76.90***	26.84	7.34	66.83	66.63	16.46***
Control (human nego.)			69.78	34.24	5.24	67.93	67.42	20.40

Table C.1: Comparison of automated negotiation systems vs. human negotiation experiments

Tables C.2 to C.19 also provide the results of multiple comparisons, but not for system configurations versus control but, between different system configurations for the outcome dimensions considered in this study. Note that each of the tables with the lower triangles of the multiple comparisons for the outcome dimensions spreads over three tables on three pages: proportion of agreements in Tables C.2 to C.4, proportion of Pareto-optimal agreements in Tables C.5 to C.7, minimal distance to the Pareto frontier in Tables C.8 to C.10, the utility of the agreement to the seller in Tables C.11 to C.13 and buyer in Tables C.14 to C.16 and the contract imbalance in Tables C.17 to C.19.

In the comparisons we grouped all simulations where a specific software agents representing the seller party negotiated in a specific interaction protocol with all other software agents representing the buyer party. Therefore we have 27 groups resulting from the combination of the nine software agents for the seller party and the three possible interaction protocols. For the outcome dimensions that are proportions the sample size for each group therefore is $2,065 * 9 * 3 = 55,755$ resulting from the three replications of negotiations with each software agent as opponent for all negotiation problems. For all other dimensions again sample sizes can be derived by multiplying these $55,755$ simulation runs with the proportions of negotiations that reached an agreement – provided in Tables C.2 to C.4 – as they can be calculated only for agreements.

The provided significance values for two-sided tests comparing the performance of different system configurations are p-values adjusted by the Bonferroni-Holm method to control for the family wise error rate in the multiple comparisons. Due to the limited space we again omit the test statistics of the tests – χ^2 for Pearson's χ^2 independence test for proportions and W for the Wilcoxon rank sum test for the remaining outcome dimensions – but gladly provide the interested reader with this information upon request.

	%	LEXact | 1	LEXact | 2	LEXact | 3	LEXpas | 1	LEXpas | 2	LEXpas | 3	MOCact | 1	MOCact | 2	MOCact | 3
LEXact | 1	100.00									
LEXact | 2	21.08	0.0000								
LEXact | 3	77.57	0.0000	0.0000							
LEXpas | 1	100.00		0.0000	0.0000						
LEXpas | 2	16.55	0.0000	0.0000	0.0000	0.0000					
LEXpas | 3	57.82	0.0000	0.0000	0.0000	0.0000	0.0000				
MOCact | 1	100.00		0.0000	0.0000		0.0000	0.0000			
MOCact | 2	12.32	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000		
MOCact | 3	79.94	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	
MOCpas | 1	100.00		0.0000	0.0000		0.0000	0.0000		0.0000	0.0000
MOCpas | 2	11.58	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0005	0.0000
MOCpas | 3	65.30	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
MUMact | 1	100.00		0.0000	0.0000		0.0000	0.0000		0.0000	0.0000
MUMact | 2	30.31	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
MUMact | 3	84.36	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
MUMpas | 1	100.00		0.0000	0.0000		0.0000	0.0000		0.0000	0.0000
MUMpas | 2	20.99	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
MUMpas | 3	63.22	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCact | 1	100.00		0.0000	0.0000		0.0000	0.0000		0.0000	0.0000
SMCact | 2	17.98	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCact | 3	77.47	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCpas | 1	100.00		0.0000	0.0000		0.0000	0.0000		0.0000	0.0000
SMCpas | 2	15.12	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCpas | 3	60.69	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
TFT | 1	100.00		0.0000	0.0000		0.0000	0.0000		0.0000	0.0000
TFT | 2	32.75	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
TFT | 3	67.12	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table C.2: System component interactions for the proportion of agreements 1/3

	%	MOCpas | 1	MOCpas | 2	MOCpas | 3	MUMact | 1	MUMact | 2	MUMact | 3	MUMpas | 1	MUMpas | 2	MUMpas | 3
LEXact | 1	100.00									
LEXact | 2	21.08									
LEXact | 3	77.57									
LEXpas | 1	100.00									
LEXpas | 2	16.55									
LEXpas | 3	57.82									
MOCact | 1	100.00									
MOCact | 2	12.32									
MOCact | 3	79.94									
MOCpas | 1	100.00									
MOCpas | 2	11.58	0.0000								
MOCpas | 3	65.30	0.0000	0.0000							
MUMact | 1	100.00		0.0000	0.0000						
MUMact | 2	30.31	0.0000	0.0000	0.0000	0.0000					
MUMact | 3	84.36	0.0000	0.0000	0.0000	0.0000	0.0000				
MUMpas | 1	100.00		0.0000	0.0000		0.0000	0.0000			
MUMpas | 2	20.99	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000		
MUMpas | 3	63.22	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	
SMCact | 1	100.00		0.0000	0.0000		0.0000	0.0000		0.0000	0.0000
SMCact | 2	17.98	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCact | 3	77.47	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCpas | 1	100.00		0.0000	0.0000		0.0000	0.0000		0.0000	0.0000
SMCpas | 2	15.12	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCpas | 3	60.69	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
TFT | 1	100.00		0.0000	0.0000		0.0000	0.0000		0.0000	0.0000
TFT | 2	32.75	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
TFT | 3	67.12	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table C.3: System component interactions for the proportion of agreements 2/3

	%	SMCact | 1	SMCact | 2	SMCact | 3	SMCpas | 1	SMCpas | 2	SMCpas | 3	TFT | 1	TFT | 2
LEXact | 1	100.00								
LEXact | 2	21.08								
LEXact | 3	77.57								
LEXpas | 1	100.00								
LEXpas | 2	16.55								
LEXpas | 3	57.82								
MOCact | 1	100.00								
MOCact | 2	12.32								
MOCact | 3	79.94								
MOCpas | 1	100.00								
MOCpas | 2	11.58								
MOCpas | 3	65.30								
MUMact | 1	100.00								
MUMact | 2	30.31								
MUMact | 3	84.36								
MUMpas | 1	100.00								
MUMpas | 2	20.99								
MUMpas | 3	63.22								
SMCact | 1	100.00								
SMCact | 2	17.98	0.0000							
SMCact | 3	77.47	0.0000	0.0000						
SMCpas | 1	100.00		0.0000	0.0000					
SMCpas | 2	15.12	0.0000	0.0000	0.0000	0.0000				
SMCpas | 3	60.69	0.0000	0.0000	0.0000	0.0000	0.0000			
TFT | 1	100.00		0.0000	0.0000		0.0000	0.0000		
TFT | 2	32.75	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	
TFT | 3	67.12	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table C.4: System component interactions for the proportion of agreements 3/3

	%	LEXact | 1	LEXact | 2	LEXact | 3	LEXpas | 1	LEXpas | 2	LEXpas | 3	MOCact | 1	MOCact | 2	MOCact | 3
LEXact | 1	37.23									
LEXact | 2	10.91	0.0000								
LEXact | 3	36.35	0.0324	0.0000							
LEXpas | 1	36.53	0.1947	0.0000	1.0000						
LEXpas | 2	8.85	0.0000	0.0000	0.0000	0.0000					
LEXpas | 3	27.07	0.0000	0.0000	0.0000	0.0000	0.0000				
MOCact | 1	57.89	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000			
MOCact | 2	8.88	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000		
MOCact | 3	60.20	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	
MOCpas | 1	58.10	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000
MOCpas | 2	8.29	0.0000	0.0000	0.0000	0.0000	0.0133	0.0000	0.0000	0.0089	0.0000
MOCpas | 3	48.67	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
MUMact | 1	42.74	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
MUMact | 2	16.35	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
MUMact | 3	44.13	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
MUMpas | 1	42.06	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
MUMpas | 2	11.31	0.0000	0.3874	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
MUMpas | 3	33.56	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCact | 1	44.26	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCact | 2	10.37	0.0000	0.0517	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCact | 3	42.23	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCpas | 1	44.32	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCpas | 2	9.00	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	1.0000	0.0000
SMCpas | 3	32.66	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
TFT | 1	55.12	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
TFT | 2	19.30	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
TFT | 3	39.35	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table C.5: System component interactions for the proportion of Pareto-optimal agreements 1/3

	%	MOCpas | 1	MOCpas | 2	MOCpas | 3	MUMact | 1	MUMact | 2	MUMact | 3	MUMpas | 1	MUMpas | 2	MUMpas | 3
LEXact | 1	37.23									
LEXact | 2	10.91									
LEXact | 3	36.35									
LEXpas | 1	36.53									
LEXpas | 2	8.85									
LEXpas | 3	27.07									
MOCact | 1	57.89									
MOCact | 2	8.88									
MOCact | 3	60.20									
MOCpas | 1	58.10									
MOCpas | 2	8.29	0.0000								
MOCpas | 3	48.67	0.0000	0.0000							
MUMact | 1	42.74	0.0000	0.0000	0.0000						
MUMact | 2	16.35	0.0000	0.0000	0.0000	0.0000					
MUMact | 3	44.13	0.0000	0.0000	0.0000	0.0001	0.0000				
MUMpas | 1	42.06	0.0000	0.0000	0.0000	0.2681	0.0000	0.0000			
MUMpas | 2	11.31	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000		
MUMpas | 3	33.56	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	
SMCact | 1	44.26	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000
SMCact | 2	10.37	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCact | 3	42.23	0.0000	0.0000	0.0000	0.8870	0.0000	0.0000	1.0000	0.0000	0.0000
SMCpas | 1	44.32	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000
SMCpas | 2	9.00	0.0000	0.0005	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCpas | 3	32.66	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0234
TFT | 1	55.12	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
TFT | 2	19.30	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
TFT | 3	39.35	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table C.6: System component interactions for the proportion of Pareto-optimal agreements 2/3

	%	SMCact | 1	SMCact | 2	SMCact | 3	SMCpas | 1	SMCpas | 2	SMCpas | 3	TFT | 1	TFT | 2
LEXact | 1	37.23								
LEXact | 2	10.91								
LEXact | 3	36.35								
LEXpas | 1	36.53								
LEXpas | 2	8.85								
LEXpas | 3	27.07								
MOCact | 1	57.89								
MOCact | 2	8.88								
MOCact | 3	60.20								
MOCpas | 1	58.10								
MOCpas | 2	8.29								
MOCpas | 3	48.67								
MUMact | 1	42.74								
MUMact | 2	16.35								
MUMact | 3	44.13								
MUMpas | 1	42.06								
MUMpas | 2	11.31								
MUMpas | 3	33.56								
SMCact | 1	44.26								
SMCact | 2	10.37	0.0000							
SMCact | 3	42.23	0.0000	0.0000						
SMCpas | 1	44.32	1.0000	0.0000	0.0000					
SMCpas | 2	9.00	0.0000	0.0000	0.0000	0.0000				
SMCpas | 3	32.66	0.0000	0.0000	0.0000	0.0000	0.0000			
TFT | 1	55.12	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000		
TFT | 2	19.30	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	
TFT | 3	39.35	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table C.7: System component interactions for the proportion of Pareto-optimal agreements 3/3

	∅	LEXact | 1	LEXact | 2	LEXact | 3	LEXpas | 1	LEXpas | 2	LEXpas | 3	MOCact | 1	MOCact | 2	MOCact | 3
LEXact | 1	6.74									
LEXact | 2	3.96	0.0000								
LEXact | 3	4.64	0.0000	0.0000							
LEXpas | 1	6.89	1.0000	0.0000	0.0000						
LEXpas | 2	3.77	0.0000	0.9355	0.0000	0.0000					
LEXpas | 3	4.74	0.0000	0.0000	1.0000	0.0000	0.0000				
MOCact | 1	3.60	0.0000	0.0000	0.0000	0.0000	0.0610	0.0000			
MOCact | 2	1.81	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000		
MOCact | 3	1.39	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	
MOCpas | 1	3.58	0.0000	0.0000	0.0000	0.0000	0.0324	0.0000	1.0000	0.0000	0.0000
MOCpas | 2	1.82	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000
MOCpas | 3	1.45	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0009	1.0000
MUMact | 1	6.17	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
MUMact | 2	3.42	0.0000	0.0001	0.0000	0.0000	1.0000	0.0000	1.0000	0.0000	0.0000
MUMact | 3	3.98	0.0000	1.0000	0.0000	0.0000	0.0361	0.0000	0.0000	0.0000	0.0000
MUMpas | 1	6.53	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
MUMpas | 2	3.43	0.0000	0.0000	0.0000	0.0000	0.3344	0.0000	1.0000	0.0000	0.0000
MUMpas | 3	3.86	0.0000	1.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000
SMCact | 1	4.89	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCact | 2	2.92	0.0000	0.0000	0.0000	0.0000	0.0001	0.0000	0.2718	0.0000	0.0000
SMCact | 3	3.16	0.0000	0.1012	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000
SMCpas | 1	4.92	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCpas | 2	2.79	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCpas | 3	3.26	0.0000	0.3538	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000
TFT | 1	4.35	0.0000	1.0000	0.0000	0.0000	0.0274	0.0000	0.0000	0.0000	0.0000
TFT | 2	3.85	0.0000	0.0000	0.0000	0.0000	0.1569	0.0000	1.0000	0.0000	0.0000
TFT | 3	3.81	0.0000	0.0000	0.0000	0.0000	0.5464	0.0000	1.0000	0.0000	0.0000

Table C.8: System component interactions for the minimal distance to the Pareto frontier 1/3

	∅	MOCpas | 1	MOCpas | 2	MOCpas | 3	MUMact | 1	MUMact | 2	MUMact | 3	MUMpas | 1	MUMpas | 2	MUMpas | 3
LEXact | 1	6.74									
LEXact | 2	3.96									
LEXact | 3	4.64									
LEXpas | 1	6.89									
LEXpas | 2	3.77									
LEXpas | 3	4.74									
MOCact | 1	3.60									
MOCact | 2	1.81									
MOCact | 3	1.39									
MOCpas | 1	3.58									
MOCpas | 2	1.82	0.0000								
MOCpas | 3	1.45	0.0000	0.0000							
MUMact | 1	6.17	0.0000	0.0000	0.0000						
MUMact | 2	3.42	1.0000	0.0000	0.0000	0.0000					
MUMact | 3	3.98	0.0000	0.0000	0.0000	0.0000	0.0000				
MUMpas | 1	6.53	0.0000	0.0000	0.0000	0.2536	0.0000	0.0000			
MUMpas | 2	3.43	1.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000		
MUMpas | 3	3.86	0.0000	0.0000	0.0000	0.0000	0.0001	0.2336	0.0000	0.0000	
SMCact | 1	4.89	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCact | 2	2.92	0.4843	0.0000	0.0000	0.0000	0.0258	0.0000	0.0000	0.4321	0.0000
SMCact | 3	3.16	0.0000	0.0000	0.0000	0.0000	0.0589	0.0000	0.0000	0.0088	0.6158
SMCpas | 1	4.92	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCpas | 2	2.79	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0000
SMCpas | 3	3.26	0.0000	0.0000	0.0000	0.0000	0.0209	0.0001	0.0000	0.0029	1.0000
TFT | 1	4.35	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.1351
TFT | 2	3.85	1.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	1.0000	0.0000
TFT | 3	3.81	1.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	1.0000	0.0000

Table C.9: System component interactions for the minimal distance to the Pareto frontier 2/3

	∅	SMCact | 1	SMCact | 2	SMCact | 3	SMCpas | 1	SMCpas | 2	SMCpas | 3	TFT | 1	TFT | 2
LEXact | 1	6.74								
LEXact | 2	3.96								
LEXact | 3	4.64								
LEXpas | 1	6.89								
LEXpas | 2	3.77								
LEXpas | 3	4.74								
MOCact | 1	3.60								
MOCact | 2	1.81								
MOCact | 3	1.39								
MOCpas | 1	3.58								
MOCpas | 2	1.82								
MOCpas | 3	1.45								
MUMact | 1	6.17								
MUMact | 2	3.42								
MUMact | 3	3.98								
MUMpas | 1	6.53								
MUMpas | 2	3.43								
MUMpas | 3	3.86								
SMCact | 1	4.89								
SMCact | 2	2.92	0.0000							
SMCact | 3	3.16	0.0000	0.0000						
SMCpas | 1	4.92	1.0000	0.0000	0.0000					
SMCpas | 2	2.79	0.0000	0.5078	0.0000	0.0000				
SMCpas | 3	3.26	0.0000	0.0000	1.0000	0.0000	0.0000			
TFT | 1	4.35	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001		
TFT | 2	3.85	0.0000	1.0000	0.0000	0.0000	0.0009	0.0000	0.0000	
TFT | 3	3.81	0.0000	0.0388	0.0002	0.0000	0.0000	0.0001	0.0000	1.0000

Table C.10: System component interactions for the minimal distance to the Pareto frontier 3/3

	∅	LEXact | 1	LEXact | 2	LEXact | 3	LEXpas | 1	LEXpas | 2	LEXpas | 3	MOCact | 1	MOCact | 2	MOCact | 3
LEXact | 1	65.15									
LEXact | 2	78.26	0.0000								
LEXact | 3	68.80	0.0000	0.0000							
LEXpas | 1	64.78	1.0000	0.0000	0.0000						
LEXpas | 2	80.54	0.0000	0.0000	0.0000	0.0000					
LEXpas | 3	71.15	0.0000	0.0000	0.0000	0.0000	0.0000				
MOCact | 1	82.96	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000			
MOCact | 2	91.30	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000		
MOCact | 3	80.00	0.0000	0.0000	0.0000	0.0000	0.0700	0.0000	0.0000	0.0000	
MOCpas | 1	82.98	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000
MOCpas | 2	91.61	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.7173	0.0000
MOCpas | 3	81.35	0.0000	0.0000	0.0000	0.0000	0.0010	0.0000	0.0000	0.0000	0.0000
MUMact | 1	58.10	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
MUMact | 2	73.72	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
MUMact | 3	66.17	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
MUMpas | 1	57.15	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
MUMpas | 2	78.03	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
MUMpas | 3	70.65	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCact | 1	70.09	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCact | 2	81.78	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCact | 3	72.44	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCpas | 1	69.96	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCpas | 2	83.94	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0000	0.0000
SMCpas | 3	74.40	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
TFT | 1	58.15	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
TFT | 2	60.04	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
TFT | 3	58.55	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table C.11: System component interactions for the focal party's (seller) utility 1/3

	∅	MOCpas | 1	MOCpas | 2	MOCpas | 3	MUMact | 1	MUMact | 2	MUMact | 3	MUMpas | 1	MUMpas | 2	MUMpas | 3
LEXact | 1	65.15									
LEXact | 2	78.26									
LEXact | 3	68.80									
LEXpas | 1	64.78									
LEXpas | 2	80.54									
LEXpas | 3	71.15									
MOCact | 1	82.96									
MOCact | 2	91.30									
MOCact | 3	80.00									
MOCpas | 1	82.98									
MOCpas | 2	91.61	0.0000								
MOCpas | 3	81.35	0.0000	0.0000							
MUMact | 1	58.10	0.0000	0.0000	0.0000						
MUMact | 2	73.72	0.0000	0.0000	0.0000	0.0000					
MUMact | 3	66.17	0.0000	0.0000	0.0000	0.0000	0.0000				
MUMpas | 1	57.15	0.0000	0.0000	0.0000	0.0715	0.0000	0.0000			
MUMpas | 2	78.03	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000		
MUMpas | 3	70.65	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	
SMCact | 1	70.09	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCact | 2	81.78	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCact | 3	72.44	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCpas | 1	69.96	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCpas | 2	83.94	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCpas | 3	74.40	0.0000	0.0000	0.0000	0.0000	0.0004	0.0000	0.0000	0.0000	0.0000
TFT | 1	58.15	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
TFT | 2	60.04	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0018	0.0000	0.0000
TFT | 3	58.55	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table C.12: System component interactions for the focal party's (seller) utility 2/3

	∅	SMCact | 1	SMCact | 2	SMCact | 3	SMCpas | 1	SMCpas | 2	SMCpas | 3	TFT | 1	TFT | 2
LEXact | 1	65.15								
LEXact | 2	78.26								
LEXact | 3	68.80								
LEXpas | 1	64.78								
LEXpas | 2	80.54								
LEXpas | 3	71.15								
MOCact | 1	82.96								
MOCact | 2	91.30								
MOCact | 3	80.00								
MOCpas | 1	82.98								
MOCpas | 2	91.61								
MOCpas | 3	81.35								
MUMact | 1	58.10								
MUMact | 2	73.72								
MUMact | 3	66.17								
MUMpas | 1	57.15								
MUMpas | 2	78.03								
MUMpas | 3	70.65								
SMCact | 1	70.09								
SMCact | 2	81.78	0.0000							
SMCact | 3	72.44	1.0000	0.0000						
SMCpas | 1	69.96	1.0000	0.0000	1.0000					
SMCpas | 2	83.94	0.0000	0.0000	0.0000	0.0000				
SMCpas | 3	74.40	0.0000	0.0000	0.0000	0.0000	0.0000			
TFT | 1	58.15	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000		
TFT | 2	60.04	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	
TFT | 3	58.55	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0616	0.0000

Table C.13: System component interactions for the focal party's (seller) utility 3/3

	∅	LEXact | 1	LEXact | 2	LEXact | 3	LEXpas | 1	LEXpas | 2	LEXpas | 3	MOCact | 1	MOCact | 2	MOCact | 3
LEXact | 1	65.38									
LEXact | 2	73.82	0.0000								
LEXact | 3	69.63	0.0000	0.0000							
LEXpas | 1	65.45	1.0000	0.0000	0.0000						
LEXpas | 2	75.46	0.0000	0.0000	0.0000	0.0000					
LEXpas | 3	70.38	0.0000	0.0000	0.0002	0.0000	0.0000				
MOCact | 1	49.45	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000			
MOCact | 2	74.22	0.0000	1.0000	0.0000	0.0000	0.0028	0.0000	0.0000		
MOCact | 3	62.36	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	
MOCpas | 1	49.49	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000
MOCpas | 2	74.60	0.0000	0.7152	0.0000	0.0000	0.1135	0.0000	0.0000	1.0000	0.0000
MOCpas | 3	62.98	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0047
MUMact | 1	72.01	0.0000	0.4872	0.0000	0.0000	0.0000	0.0000	0.0000	0.2439	0.0000
MUMact | 2	73.38	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2303	0.0000
MUMact | 3	71.95	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
MUMpas | 1	72.17	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.6682	0.0000
MUMpas | 2	73.99	0.0000	1.0000	0.0000	0.0000	0.0001	0.0000	0.0000	1.0000	0.0000
MUMpas | 3	71.49	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCact | 1	62.07	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCact | 2	74.79	0.0000	0.0408	0.0000	0.0000	0.2763	0.0000	0.0000	0.9003	0.0000
SMCact | 3	68.03	0.0016	0.0000	0.0000	0.0015	0.0000	0.0000	0.0000	0.0000	0.0000
SMCpas | 1	62.14	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCpas | 2	75.81	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000
SMCpas | 3	68.55	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
TFT | 1	76.09	0.0000	0.0000	0.0000	0.0000	0.0003	0.0000	0.0000	0.0000	0.0000
TFT | 2	83.00	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
TFT | 3	80.22	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table C.14: System component interactions for the opponent's (buyer) utility 1/3

	∅	MOCpas | 1	MOCpas | 2	MOCpas | 3	MUMact | 1	MUMact | 2	MUMact | 3	MUMpas | 1	MUMpas | 2	MUMpas | 3
LEXact | 1	65.38									
LEXact | 2	73.82									
LEXact | 3	69.63									
LEXpas | 1	65.45									
LEXpas | 2	75.46									
LEXpas | 3	70.38									
MOCact | 1	49.45									
MOCact | 2	74.22									
MOCact | 3	62.36									
MOCpas | 1	49.49									
MOCpas | 2	74.60	0.0000								
MOCpas | 3	62.98	0.0000	0.0000							
MUMact | 1	72.01	0.0000	0.0081	0.0000						
MUMact | 2	73.38	0.0000	0.0072	0.0000	1.0000					
MUMact | 3	71.95	0.0000	0.0000	0.0000	0.0000	0.0000				
MUMpas | 1	72.17	0.0000	0.0384	0.0000	1.0000	1.0000	0.0000			
MUMpas | 2	73.99	0.0000	1.0000	0.0000	0.0715	0.2805	0.0000	0.3969		
MUMpas | 3	71.49	0.0000	0.0000	0.0000	0.0000	0.0000	0.0379	0.0000	0.0000	
SMCact | 1	62.07	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCact | 2	74.79	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2303	0.0000
SMCact | 3	68.03	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCpas | 1	62.14	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCpas | 2	75.81	0.0000	0.0081	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCpas | 3	68.55	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
TFT | 1	76.09	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
TFT | 2	83.00	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
TFT | 3	80.22	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table C.15: System component interactions for the opponent's (buyer) utility 2/3

	∅	SMCact | 1	SMCact | 2	SMCact | 3	SMCpas | 1	SMCpas | 2	SMCpas | 3	TFT | 1	TFT | 2
LEXact | 1	65.38								
LEXact | 2	73.82								
LEXact | 3	69.63								
LEXpas | 1	65.45								
LEXpas | 2	75.46								
LEXpas | 3	70.38								
MOCact | 1	49.45								
MOCact | 2	74.22								
MOCact | 3	62.36								
MOCpas | 1	49.49								
MOCpas | 2	74.60								
MOCpas | 3	62.98								
MUMact | 1	72.01								
MUMact | 2	73.38								
MUMact | 3	71.95								
MUMpas | 1	72.17								
MUMpas | 2	73.99								
MUMpas | 3	71.49								
SMCact | 1	62.07								
SMCact | 2	74.79	0.0000							
SMCact | 3	68.03	0.0000	0.0000						
SMCpas | 1	62.14	1.0000	0.0000	0.0000					
SMCpas | 2	75.81	0.0000	0.0204	0.0000	0.0000				
SMCpas | 3	68.55	0.0000	0.0000	0.0444	0.0000	0.0000			
TFT | 1	76.09	0.0000	0.0000	0.0000	0.0000	0.0220	0.0000		
TFT | 2	83.00	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	
TFT | 3	80.22	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table C.16: System component interactions for the opponent's (buyer) utility 3/3

	∅	LEXact | 1	LEXact | 2	LEXact | 3	LEXpas | 1	LEXpas | 2	LEXpas | 3	MOCact | 1	MOCact | 2	MOCact | 3
LEXact | 1	24.67									
LEXact | 2	15.33	0.0000								
LEXact | 3	14.97	0.0000	1.0000							
LEXpas | 1	24.99	1.0000	0.0000	0.0000						
LEXpas | 2	15.48	0.0000	1.0000	1.0000	0.0000					
LEXpas | 3	14.95	0.0000	1.0000	1.0000	0.0000	1.0000				
MOCact | 1	35.49	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000			
MOCact | 2	18.84	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000		
MOCact | 3	19.31	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0120	
MOCpas | 1	35.46	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000
MOCpas | 2	18.74	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0017
MOCpas | 3	19.95	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0009
MUMact | 1	31.54	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
MUMact | 2	16.69	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
MUMact | 3	18.38	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000
MUMpas | 1	32.71	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
MUMpas | 2	16.45	0.0000	0.0001	0.0000	0.0000	0.0002	0.0000	0.0000	0.0000	0.0000
MUMpas | 3	16.54	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCact | 1	28.37	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCact | 2	16.04	0.0000	0.9188	1.0000	0.0000	1.0000	0.7803	0.0000	0.0000	0.0000
SMCact | 3	16.17	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCpas | 1	28.57	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCpas | 2	15.99	0.0000	1.0000	1.0000	0.0000	1.0000	1.0000	0.0000	0.0000	0.0000
SMCpas | 3	16.15	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
TFT | 1	23.05	0.0000	0.0000	0.0000	0.0010	0.0000	0.0000	0.0000	0.0000	0.0000
TFT | 2	27.28	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
TFT | 3	25.15	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table C.17: System component interactions for the contract imbalance 1/3

	∅	MOCpas | 1	MOCpas | 2	MOCpas | 3	MUMact | 1	MUMact | 2	MUMact | 3	MUMpas | 1	MUMpas | 2	MUMpas | 3
LEXact | 1	24.67									
LEXact | 2	15.33									
LEXact | 3	14.97									
LEXpas | 1	24.99									
LEXpas | 2	15.48									
LEXpas | 3	14.95									
MOCact | 1	35.49									
MOCact | 2	18.84									
MOCact | 3	19.31									
MOCpas | 1	35.46									
MOCpas | 2	18.74	0.0000								
MOCpas | 3	19.95	0.0000	0.0000							
MUMact | 1	31.54	0.0000	0.0000	0.0000						
MUMact | 2	16.69	0.0000	0.0000	0.0000	0.0000					
MUMact | 3	18.38	0.0000	1.0000	0.0000	0.0000	0.0000				
MUMpas | 1	32.71	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000			
MUMpas | 2	16.45	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000		
MUMpas | 3	16.54	0.0000	1.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.2672	
SMCact | 1	28.37	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCact | 2	16.04	0.0000	0.0000	0.0000	0.0000	0.0040	0.0000	0.0000	0.4403	0.0000
SMCact | 3	16.17	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	1.0000	0.3963
SMCpas | 1	28.57	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SMCpas | 2	15.99	0.0000	0.0000	0.0000	0.0000	0.0015	0.0000	0.0000	0.1939	0.0000
SMCpas | 3	16.15	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	1.0000	0.0803
TFT | 1	23.05	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
TFT | 2	27.28	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.0012	0.0000	0.0000
TFT | 3	25.15	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table C.18: System component interactions for the contract imbalance 2/3

	∅	SMCact | 1	SMCact | 2	SMCact | 3	SMCpas | 1	SMCpas | 2	SMCpas | 3	TFT | 1	TFT | 2
LEXact | 1	24.67								
LEXact | 2	15.33								
LEXact | 3	14.97								
LEXpas | 1	24.99								
LEXpas | 2	15.48								
LEXpas | 3	14.95								
MOCact | 1	35.49								
MOCact | 2	18.84								
MOCact | 3	19.31								
MOCpas | 1	35.46								
MOCpas | 2	18.74								
MOCpas | 3	19.95								
MUMact | 1	31.54								
MUMact | 2	16.69								
MUMact | 3	18.38								
MUMpas | 1	32.71								
MUMpas | 2	16.45								
MUMpas | 3	16.54								
SMCact | 1	28.37								
SMCact | 2	16.04	0.0000							
SMCact | 3	16.17	0.0000	0.0016						
SMCpas | 1	28.57	1.0000	0.0000	0.0000					
SMCpas | 2	15.99	0.0000	1.0000	0.0005	0.0000				
SMCpas | 3	16.15	0.0000	0.0178	1.0000	0.0000	0.0053			
TFT | 1	23.05	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000		
TFT | 2	27.28	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	
TFT | 3	25.15	1.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000

Table C.19: System component interactions for the contract imbalance 3/3

Simulation of Automated Negotiation

Mag. Michael Filzmoser

Abstract

Automated negotiation is argued to improve negotiation outcomes by replacing humans and to enable coordination in autonomous systems. As operative systems do not yet exist scholars rely on simulations to evaluate potential systems for automated negotiation. This dissertation reviews the state of the art literature on simulation of automated negotiation along its main components – negotiation problem, interaction protocol, and software agents. Deficiencies of existing approaches concerning the practical application in an open environment as the Internet – where automated negotiation proceeds fast, with changing opponents, and for various negotiation problems – are identified.

To address these deficiencies we develop and simulate automated negotiation systems, consisting of software agents that follow generic offer generation and concession strategies and protocols that allow these agents to interrupt their strategy to avoid exploitation and unfavorable agreements. Outcomes of simulation runs are compared across systems and to human negotiation along various outcome dimensions – proportion of agreements, dyadic and individual performance, and fairness – for various negotiation problems derived from negotiation experiments with human subjects.

Though there exist trade-offs between the different outcome dimensions, systems consisting of software agents, that systematically propose offers of monotonically decreasing utility and make first concession steps if the opponent reciprocated previous concessions, and an interaction protocol that enables to reject unfavorable offers – without immediately aborting negotiations – in order to elicit new offers from the opponent, performed best. These systems performed very well in all outcome dimensions when compared with other systems and were the only that outperformed negotiation between humans in all dimensions.

Simulation automatisierter Verhandlungen

Mag. Michael Filzmoser

Abstract

Durch die Automatisierung von Verhandlungen sollen bessere Verhandlungsergebnisse erzielt werden können als bei Verhandlungen zwischen Menschen und neue Koordinationsformen für autonome Agentensysteme ermöglicht werden. Diese Arbeit beschäftigt sich mit der Simulation solcher Systeme füur automatisierte Verhandlungen, da operative Systeme zur Zeit noch nicht verfügbar sind. Die Arbeit basiert auf einer Erhebung und Diskussion der aktuellen Literatur im Bereich der Simulation automatisierter Verhandlungen. Existierende Ansätze weisen einige Unzulänglichkeiten bezüglich deren praktischer Umsetzbarkeit in einer offenen Umgebung wie dem Internet auf, wo automatisierte Verhandlungen nicht nur sehr schnell durchgeführt werden sondern sich auch Software-Agenten und Verhandlungsprobleme ändern können.

Diese Defizite thematisierend werden Verhandlungssysteme für automatisierte Verhandlungen vorgeschlagen. Diese bestehen zum einen aus Software-Agenten, die generische Angebots- und Konzessionsstratgien verfolgen, zum anderen aus Interaktionsprotokollen, die es Agenten erlauben ihre Strategien vorübergehend oder permanent auszusetzen. Ergebnisse der Simulation dieser Systeme, mit Verhandlungsproblemen aus Verhandlungsexperimenten mit menschlichen Probanden als Input, werden für unterschiedliche Ergebnisdimensionen – Übereinkunftshäufigkeit, Fairness, individuelle und kollektive Effizienz – zwischen Systemen und auch mit den Ergebnissen der Experimente verglichen.

Trotz fundamentaler Zielkonflikte zwischen den einzelnen Ergebnisdimensionen erzielen einige Systeme konsistent bessere Ergebnisse sowohl im Systemvergleich als auch verglichen mit den Ergebnissen der Experimente. Diese Systeme bestehen aus Software-Agenten die systematisch Angebote mit monoton abnehmendem Nutzen unterbreiten und erste Konzessionensschritte tätigen solange der Opponent bisherige Konzessionen erwidert hat. Das verwendete Interaktionsprotokoll zeichnet sich dadurch aus, dass es den Agenten erlaubt ungünstige Angebote zurückzuweisen und damit neue Angebote des Opponenten einzufordern, durch diese Unterbrechung der eigenen Angebotsstrategie können ungünstige Verhandlungsergebnisse vermieden werden.

Curriculum Vitae

Personal Information

Name	Michael Filzmoser
Date of birth	29. 09. 1980
Citizenship	Austria
Marital status	married, four children
Address	Höllweg 19
	A-2011 Sierndorf
E-mail	`michael.filzmoser@univie.ac.at`
Phone	0043 676 9612475

Education

09/1987–06/1991	Elementary school, Volksschule Sierndorf (passed with distinction)
09/1991–06/1995	Comprehensive secondary school, Bundessportrealgymnasium Hollabrunn (passed with distinction)
09/1995–06/2000	Commerical academy, Bundeshandelsakademie Hollabrunn (passed with distinction)
07/2000–02/2001	Military service (obligatory in Austria)
03/2001–03/2005	Studies in international business administration, University of Vienna, School of Business Administration, Economics, and Statistics (passed with distinction in minimum duration)
03/2005–today	Ph.D. Management, University of Vienna, School of Business Administration, Economics, and Statistics (title of the dissertation: 'Simulation of Automated Negotiation')

Professional Experience

	GEKO Großhandelsgmbh – accounting department (internship: inventory accounting, bookkeeping and general office occupations) *07/1996, 08/1997, and 07/1999*
08/1999	**Haas Waffelmaschinen AG** – accounting department (internship: general office occupations)
09/2001	**Creditanstalt AG** – export funding department (volontariat: preparation of SWIFT-transactions, process documentation and automatization)
08/2003	**Saint Gobain Isover AG** – marketing department (internship: database maintenance, participation in a market research project)

11/2003–03/2004 **Saint Gobain Isover AG** – marketing department (part-time internship: database maintenance, input and evaluation of market research data)

07/2004–02/2005 **Austrian Airlines Group** – process-management department (part-time participation in the 'Key-Performance-Indicator'-Workshop: evaluation and improvement of AAG's business processes)

10/2004–02/2005 **University of Vienna** – Chair of organization and planning (part-time undergraduate assistant: participation in research and teaching, student support)

03/2005–today **University of Vienna** – Chair of organization and planning (university assistant: research and teaching, Ph.D. thesis)

Teaching Experience

courses

	term							
course	winter 05	summer 06	winter 06	summer 07	winter 07	summer 08	winter 08	summer 09
Bachelor level								
Organization (dt)	2	3	2	2		1	1	
Organization and Human Resources (dt)					1	1		
Master level								
Organizational Design (en)					1		1	
Indiv. and org. Decision Making (en)								1
Organizational Culture and Development (dt)					x*		x*	

* 3h lecture in the postgraduate program 'Master of Public Health' of the University of Vienna

supervised master theses

author	title	finished
Stradner, Isabella	Expatriates and the encountered barriers to transfer their knowledge	2006
Weinzinger, Hanna	Business Reengineering in der Evangelischen Kirche Österreich	2007
Simson, Philip	Kriterien und Methoden zur Lieferantenauswahl	2008
Steidl, Christian	Motivation der Mitarbeiter in Massen- oder Fließproduktionsbetrieben	2008
Hutzinger, Clemens	Wirtschaftlichkeit des Prozessmanagements	2008
Fitzke, Rene	Six Sigma im tertiären Sektor - Am Beispiel der LIWEST Kabelmedien GmbH	in process
Baluha, Elisabeth	Strategisches Geschäftsprozessmanagement	in process

Scholarships, Grants, and Awards

2003	'Siegried Ludwig Stipendium' – scholarship for excellent performance of the Siegfried Ludwig Fonds
2003	'Top Stipendium' – scholarship of the provincial government of Lower Austria
2002-2004	'Leistungsstipendium der Universität Wien' scholarship for excellent performance of the University of Vienna
03/2001–02/2005	'Studienbeihilfe' general grant for students of the Austrian Federal Ministry of Science and Research
2006	'Young Scientists Best Paper Award' at the International Conference of Group Decision and Negotiation (GDN), Karlsruhe
2006	'Best of the Best Award' of the career center of the University of Vienna, BCG, and MLP, for being the best graduate from the school for business administration, economics and statistics of the University of Vienna in 2004/05

Publications

refereed journals

Filzmoser M and R Vetschera (2008): A classification of bargaining steps and their impact on negotiation outcomes, *Group Decision and Negotiation*, 17, p 421-443

book chapters

Filzmoser M (2007): Exponential smoothed Tit-for-Tat, in Kendall G, Yao X, and S Yew Chong (eds): *The iterated prisoner's dilemma: 20 years on*, World Scientific, p 127-138

conference proceedings

Filzmoser M (2008): Non-alternating offer protocols and concession strategies for automated negotiations, in: J Climaco, G E Kersten and J P Costa (eds): *Proceedings of the international conferenc on group decision and negotiation (GDN) 2008*, INESC Coimbra, Coimbra, p 267-268

Filzmoser M, J Rios and R Vetschera (2008): The impact of support and preference elicitation on consistency in e-negotiations, in: J Climaco, G E Kersten and J P Costa (eds): *Proceedings of the international conferenc on group decision and negotiation (GDN) 2008*, INESC Coimbra, Coimbra, p 241-242

Filzmoser M and R Vetschera (2007): The relation of messages and offers in negotiations, in: G E Kersten, J Rios and E Chen (eds): *Proceedings of the international conference on group decision and negotiation (GDN) 2007*, InterNeg Research Centre, Montreal, p 135-137

Filzmoser M (2006): Asessing the predicitive accuracy of axiomatic solutions to bargaining, in: Seifert S and C Weinhardt (eds): *Proceedings of the internatnional conference on group decision and negotiation (GDN) 2006*, universitätsverlag karlsruhe, Karlsruhe, p 250-252

Filzmoser M and R Vetschera (2006): The influence of bargaining steps on the process and outcome of online negotiations, in: Seifert S and C Weinhardt (eds): *Proceedings of the internatnional conference on group decision and negotiation (GDN) 2006*, universitätsverlag karlsruhe, Karlsruhe, p 224-227

Filzmoser M (2005): The firm as a bundle of resources - A synergy- and value-based view of economic organization, *Proceedings of the international conference on economics and management of networks (EMNet) 2005*, Budapest (CD)

master thesis

Filzmoser M (2005): Prozessführung am Beispiel der Austrian Airlines Group, Master Thesis, University of Vienna

Misc

languages	german – mother tongue english – fluent french – basics
software	MS Windows, MS Office, Open Office, Latex, java, PHP, MySQL, html, xml, javascript, R, SPSS
interests	auxiliary fire brigade, chess, reading, music, philosophy, hiking and climbing