COOPERATIVE TEAM FORMATION USING DISTRIBUTED DECOMPOSITION KNOWLEDGE

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Humanities

2010

Martin Carpenter
Manchester Business School

Contents

Word count 60673

List of Figures

Abstract

Cooperative team formation using distributed decomposition knowledge

A thesis submitted in August 2010 by Martin Carpenter to the University of Manchester for the degree of Doctor of Philosophy.

In recent years, the problem of automating the formation of Virtual Organisations (VO) has risen to prominence. Work in this area has typically considered the process of VO formation to be a centralised process driven by a company with responsibility for the business opportunity.

Such systems use two main stages: first they decompose the business opportunity into a set of roles and then select suppliers for each role by matching their advertised capability against criteria supplied by the user. Both stages require that the company driving the VO formation process has access to considerable amounts of centralised knowledge.

In contrast, this thesis considers virtual organisations as forming by combining the cooperative contributions within a group of organisations. It is shown that, within this context, both the knowledge required to facilitate and the control within the virtual organisation formation process are naturally distributed. In particular companies are free to vary their level of commitment to given projects and so only they have detailed knowledge of their capabilities. Supporting VO formation within this context requires a novel approach capable of utilising this distributed information.

The primary contribution of this thesis is to provide such a novel approach to supporting virtual organisation formation. This approach builds on the traditions of blackboard and multi-agent systems. It allows virtual organisation formation to be driven by the accumulation of voluntary contributions from the prospective members of the virtual organisation. The principle focus of the system is on identifying candidate virtual organisations, and it does not offer automated support for such aspects as the creation of contracts. Crucially this system works with the distributed knowledge encountered in the chosen problem domain.

The following technical contributions shape the general approach into a detailed system: (a) the representation of company's capabilities, (b) an algorithm for combining those capabilities and (c) mechanisms enabling intelligent agents representing the companies to produce candidate virtual organisations. The proposed system is evaluated in three ways - its technical feasibility is demonstrated through the implementation of a testbed prototype, a theoretical discussion of the systems performance is given and finally its potential benefits are shown in a reasoned case study.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgements

The work within this thesis was initially developed as part of the Crosswork European research project (2004-06) and further developed during the SUDDEN European research project (2006-09). I would like to thank all the participants in the projects for their valuable contributions and, in particular, for the discussions about the automotive industry which provided both the initial inspiration for this work and the basis of the case study later presented.

I would also like to thank my supervisor Dr Nikolay Mehandjiev and my second supervisor Dr Iain Stalker for their invaluable support and input during the development of this work. Without their help, the experience of reading this document would be considerably less pleasant.

In addition I would like to thank the members of my family for providing valuable psychological support, and in particular my mother for help in proofreading the thesis.

Chapter 1

Introduction

Many tasks can only be effectively addressed through the combined efforts of a team. The investigation of methods offering support for the formation of teams has proved to be a highly fertile field of research. Many differing aspects of this problem are studied, one of which concerns the formation of teams to fulfil a task for an external entity - for example to supply a component to an automotive company. The temporary groupings formed by a set of companies to fulfil such tasks are known as virtual organisations, henceforth VOs, and the support of their formation forms the main area of study within this thesis.

Traditionally the problem of supporting VO formation has been investigated from the viewpoint of an 'owner' controlling the task. In such approaches, this owner first uses its domain knowledge to break down the task into subtasks and then uses some form of advertising service to locate the best candidate(s) for each of these subtasks. Sometimes, however, the owner of the task might lack appropriate domain knowledge and choose to advertise the opportunity before selecting a VO from among the bids they receive.

The latter situation presents an opportunity for companies to form temporary VOs in order to bid for the advertised contract. The desirability of participating in VOs can lead groups of companies to form looser but durable groupings which aim to facilitate the formation of VOs to bid for such tasks amongst their members. The main focus of this thesis. is the formation of VOs within such groupings, and, in particular, automated techniques offering support for it.

The cooperative nature of VO formation within such long term groupings makes techniques developed for supporting owner driven VO formation unsuitable. This thesis presents a technique which supports a cooperative style of VO formation through focusing on aggregating voluntary contributions from the member companies. This approach uses the domain knowledge distributed throughout the group of companies. Additionally, by placing each company in charge of its own contributions, it allows the member companies to intelligently tailor their contributions to the project at hand.

The original motivation for the work described within this thesis came from the study of virtual organisation formation within the automotive industry. This area provides valuable motivation and is discussed briefly within the following section and in greater detail in the subsequent background chapter.

1.1 Motivation

The primary motivation for the work presented within this thesis derives from considering how automatic support for the process of virtual organisation formation within a group of collaborative entities might best be offered.

The Austrian Automotive Cluster(AC) represents a classic example of such a group and was studied as a case study within the Crosswork European project [3]. An overview of the findings of this case study is presented in [1]. The AC is an organisation whose membership comprises a loosely affiliated set of automotive manufacturers. The major purpose of the AC is to help these companies to form virtual organisations to bid to supply substantial components to original equipment manufacturers (OEMs). OEMs are those companies, such as BMW, responsible for producing complete cars to sell.

A notable feature of the team formation process within this context is that the knowledge required to perform reliable virtual organisation formation is distributed throughout its member companies. It is thus natural to consider using distributed techniques to support team formation.

This thesis describes a novel approach to supporting team formation in which the control is fully distributed. The design of this approach utilises ideas from the areas of multi-agent and blackboard systems to produce a system in which team formation is driven by the voluntary contributions of companies. This proposed approach reflects the natural distribution of both knowledge and control and allows companies to intelligently vary which processes they offer in response to their personal evaluation of the value of participation in the overall task.

1.2 Aim

The primary aim of the work described within this thesis is the design of an approach for supporting virtual organisation formation within a group of cooperative companies.

The first objective is to present a clear discussion of the problem domain of interest. Of particular importance is the way in which the information needed to drive VO formation is naturally distributed within this domain. In addition a detailed examination of why existing techniques are unsuitable for application within this domain is given.

This study provides the insight required to achieve the second objective - the development of a novel system for supporting VO formation in the chosen domain. In developing such a system there is a natural tension between presenting an abstract approach with the widest possible overall scope and ensuring that the chosen approach is concrete enough to implement and test. This thesis aims to resolve this conflict by first presenting an abstract approach and then adding detail to it, ultimately developing a more grounded system which can be both implemented and tested.

This facilitates the final objective of this thesis - the evaluation of both the proposed approach and the system derived from it. A combination of actual software testing and theoretical arguments is used to test both the validity and utility of the proposed approach and the system derived from it.

1.3 Contributions

The primary contribution of this thesis is:

> An approach for providing automatic support for the cooperative formation of virtual organisations in the absence of centralised decomposition information through the identification of candidate virtual organisations.

A candidate virtual organisation is a set of companies who have identified that, in principle, they can and are willing to work together to bid for a business opportunity but who have yet to conduct the detailed negotiations required to do so. The system does not attempt to automatically support these detailed negotiations.

The development of this approach is motivated by an investigation of why existing techniques fail to provide adequate support for this kind of team formation. The functionality of this approach is tested in three ways: its technical validity is shown through the development of a proof of concept demonstrator, its theoretical utility is shown in a reasoned case study and its performance and the associated practical implications of this are discussed in detail. The process of developing a concrete demonstrator from the initial abstract approach gives rise to several technical contributions:

- a data structure capable of allowing companies to specify which service they propose to contribute to partial teams without committing to a concrete method of doing so;

- a method for combining these statements of capability into partial and ultimately complete teams;
- the detailed design of a multi agent system implementing the approach of the primary contribution.

While each of these contributions is interesting in their own right, their major importance is that they combine to produce a method for implementing a concrete system based on the general approach from the primary contribution.

1.4 Related Publications

The abstract approach presented within this thesis has been presented in the following two papers:

Carpenter, M; Mehandjiev, N; Stalker, I D, *Emergent Process Interoperability within Virtual Organisations*, AAMAS 05 Agent-based Technologies and applications for enterprise interOperability workshop (ATOP) 2005

In addition an extended version of this paper for publication within a volume of Lecture notes in Business Information Processing is currently under preparation and is due to be published in the first half of 2009.

M. Carpenter, N. Mehandjiev, I.D. Stalker, *Flexible Behaviours for Emergent Process Interoperability*, Workshop on Process Integration of Collaborative Enterprises (PINCET) held at IEEE WETICE'06, Manchester, UK, 26th-28th June 2006

1.5 Outline

The next two chapters contain the main literature review of the thesis. Chapter 2 presents a detailed investigation of the application domain for this thesis - cooperative VO formation within virtual breeding environments, VBEs, - and derives from this certain criteria for evaluating the suitability of existing approaches within this domain. The chapter starts by introducing the abstract application domain for this thesis followed by a detailed discussion of the automotive domain and related manufacturing domains, demonstrating how the features in the abstract domain can be mapped onto ones within the more concrete domain. Finally, a set of criteria for evaluating team formation techniques is derived.

Chapter 3 then presents an examination of existing theoretical work related to supporting VO formation. This includes an evaluation of which approaches are best suited to application within the domain of this thesis using the criteria derived within Chapter 2. As well as techniques directly associated with supporting VO formation, this chapter also considers contributions from related fields such as planning and distributed AI. Additionally the review briefly surveys certain other topics including software agents.

Chapter 4 introduces the primary contribution of the project - the abstract approach developed in order to supporting VO formation within the problem domain of this thesis. It starts with a discussion of which stages of VO formation are most suited to automated support, concluding by closely defining the focus of the proposed approach. The approach is then introduced and certain aspects of interest are discussed.

The next three chapters - 5, 6 and 7 - add more detail to this abstract approach, cumulating in an approach which is implemented within a concrete system. Chapter 5 presents a detailed discussion of the basic data types used to represent the individual contributions of the agents and the data structure and algorithms used to combine these. This chapter thus presents the first two technical contributions of the thesis.

The third technical contribution is described in Chapter 6 which discusses how the approach presented within Chapter 5 can be turned into a system with distributed control through the use of intelligent agents to represent the member companies of the VBE; these agents intelligently control the addition of the contributions of their parent companies. This is done by presenting a set of agent types together with their basic roles. Detailed consideration is given to the types of reasoning required from these agents.

Chapter 7 presents a validation of the approach proposed within the thesis. There are two major elements to this. Firstly, the technical validity is demonstrated by the implementation of a proof of concept technical demonstrator. Secondly the ability of the approach to facilitate the intelligent location of novel solution types is shown by a reasoned case study which takes inspiration from a scenario examined within the Crosswork European project, [3]. In addition the chapter presents a discussion of the theoretical performance of the system.

Chapter 8 discusses how the proposed system compares with existing work, how future research into it might best be focused, the obstacles likely to be faced by a real world implementation and to which domains it might best be applied. Finally the chapter concludes the thesis with a discussion of how well the proposed system met its original goals.

Chapter 2

Cooperative Virtual Organisation Formation

The purpose of the current chapter is to define the problem area addressed within this thesis. The chapter starts by defining the domain within which the virtual organisation formation in this thesis is assumed to happen, and then considers how this affects the process of supporting VO formation.

Certain characteristics of the problem domain are stated, the relevance of which is then illustrated by using the concrete example of the automotive manufacturing domain. The chapter then discusses how the problem domains characteristics affect the process of VO formation. This provides the basis for Chapter 3 which presents a discussion of how well suited existing techniques for VO formation are to application in the chosen domain.

2.1 Virtual Organisations and their Breeding Environments

2.1.1 Basic Definitions

The basic background for the work within this thesis derives from the ideas of Virtual Organisation and Virtual Breeding Environments. There have been many, slightly differing, definitions of virtual organisations. The following definition was given by Oliveira and Rocha in [61]:

> a virtual organisation is an aggregation of autonomous and independent organisations connected through a network and brought together in response to a customer need.

.

This definition clearly states the characteristic features of a virtual organisation, and is adopted as the basic definition of a virtual organisation within this thesis. This definition however mentions neither the context in which the virtual organisation formed nor any structures relevant to its formation. All of these features are crucial to the creation of systems supporting VO formation and are examined in the following discussion.

An additionally useful categorisation is found in Faisst in [28] where the author identifies three distinct types of virtual organisation, A, B and C. These types are distinguished by the level of previous interaction between the agents forming the virtual organisation. Within a type C organisation the companies forming the virtual organisation are purely independent of each other and have had no previous contact. Within a type A organisation the companies forming the virtual organisation all come from within a pool of companies who have been cooperating in a stable network. Type B is a hybrid between types A and C in that some companies come from a stable network whereas some come from outside. This thesis deals mainly with type A virtual organisations.

While Faisst's categorisation of virtual organisations is useful, he does not attempt to provide a description of the pool of companies from which type A VOs form. One candidate idea for this is that of congregations, defined by Brooks and Durfee in [10] as a set of self interested agents grouped in order to "allow a member to find suitable partners for interaction more easily". In [10] a congregation is categorised as a groups of agents with the following features:

1. the members are individually rational agents but have no specific sense of group rationality;
2. agents are free to join and leave the congregation at will;
3. the satisfaction of an agent with a congregation is defined by how well the other agents meet its needs;
4. agents will have repeated interactions and long term existence;
5. the agents must pay some cost in order to advertise for potential partners.

The first criteria defining a congregation captures the idea that the members of the congregation have joined it out of self interest, thus differentiating it from a purely cooperative grouping of companies. A clear example of this form of motivation can be seen in a farmers' marketplace - given as an example of congregation in [10] - here there is a fixed cost to setting up such a system, but it makes locating appropriate providers much easier in the future. Such an organisation clearly contrasts with a farmers cooperative.

An alternative theoretical model is the idea of a virtual organisation breeding environment (VBE) originally introduced by Hamideh Afsarmanesh and Luis M. Camarinha-Matos in [12] and later developed into the core application context behind the ECOLEAD European project. A definition of a VBE was given by the same authors in a later paper,[2] as:

> A VBE is an association of organisations and their related supporting institutions, adhering to a base long term cooperation agreement, and adoption of common operating principles and infrastructures, with the main goal of increasing both their chances and preparedness towards collaboration in potential virtual organisations

A fuller development of this idea with numerous real world examples can be found in [17]. The concept of a VBE is analogous to that of an automotive cluster and henceforth all VO formation in this thesis is assumed to take place within a VBE.

In order to fully reflect the problem domain of this thesis certain changes and clarification to the basic definition of a VBE are required. The first concerns the requirement in the above definition that member companies should share a base long term cooperation agreement and common operating principles. This requirement does not fit well with organisations such as automotive clusters which purely exist to serve their membership.

Within the system proposed in this thesis the requirement that each member make an explicit commitment to long term cooperation is replaced by the basic assumption that the companies within the VBE joined it in order to constructively participate in VO formation. As argued by Sen within [73] the social control exhibited as a result of repeated interactions between the same group of entities often suffices to ensure cooperative behaviour without the need for explicit, formal penalties.

In order to fit in well with the chosen concrete application domain for this thesis, the tasks motivating VO formation are henceforth known as business opportunities, or BOs. This decision should not be taken as indicating that the system can only be used for domains containing concrete businesses. Further clarification of both the nature of business opportunities and the additional properties assumed of VBEs are covered in the following section.

2.1.2 Assumed Domain Features

As is demonstrated in [17] the idea of VBE encompasses many differing structures. In order to develop an approach for supporting VO formation certain additional assumptions about the nature of the VBEs treated were made.

The assumptions presented within this section do not define the most general conditions in which the abstract approach itself might prove useful. Section 8.2.2 presents a detailed discussion of precisely which domain features the proposed approach relies on and why it does so.

The basic application domain for the work within this thesis is a VBE of companies within which the following additional conditions hold:

- team formation happens over a long time period;
- the individual business opportunities have sufficient worth to motivate companies to offer customised processes;
- the output of the team formation process does not need to be directly executable;
- a typical business opportunity consists of a request to manufacture a certain amount of some concrete product.

The first of these conditions means that there is a reasonable length of time between the announcement of an business opportunity and the awarding of the contract - certainly days and often weeks or months. This length of time allows companies time to form consortia to bid, create accurate estimates of how much participating in the project will cost them etc. This time period allows for the use of software which supports companies in allowing virtual organisations to slowly emerge from a process involving considerable negotiation between the companies involved.

The second characteristics means that the business opportunities addressed possess a certain 'scale'. This is important for several reasons. One is that it allows companies to decide that they can invest in new facilities to meet a specific business opportunity. If, for example, it will take several years for a particular machine to repay its initial cost a company is much more likely to invest if the machine comes with a guaranteed market for its products. Conversely if the business opportunities each only involved supplying a small batch of standardised components then there would be little chance of companies offering customised solutions for that opportunity.

The third requirement enforces a clear separation between the need to form a virtual organisation and the need to produce a detailed plan which this organisation will follow. In consequence there is no need for either the virtual organisation as a whole or its individual members to commit to detailed processes before the virtual organisation has formed. This provides a direct contrast with domains such as web service composition where the output must be a directly executable combination of services.

The fourth requirement provides a restriction on the nature of the business opportunities for which VOs formed. The restriction is intended to support the reasoning required for the overall system later proposed to successfully operate. The requirement chosen both reflects the systems original inspiration from the automotive domain and is straightforward to understand without prior knowledge of the system itself. More general statements of this requirement are possible and are discussed at length in Section 8.2.2.

A further implicit assumption is that the domain must support the existence of VBEs. The basic requirement for this to happen is that some of the people originating business opportunities are happy assigning large opportunities to a single team rather than insisting on individually selecting every member of that team. Finally, while the abstract system itself is designed for the application domain outlined, the more concrete reference model of the automotive sector is later used when developing a concrete system. This domain is examined within the next section.

2.2 Vehicle Manufacturing Industry

The following section illustrates how the domain features in the previous section hold within certain concrete application domains. The work within this thesis was developed during the course of two 6th Framework IST European projects, CrossWork, [3] and SUDDEN, [60] which both focused on certain aspects of supporting virtual organisation formation within the European Automotive industry. Additional case studies are drawn from the closely related area of aircraft manufacturing.

The section contains three subsections, the first is a general introduction to the automotive industry and the second illustrates how the characteristic features of the problem domain of this thesis hold within this domain. The final subsection contains a discussion of the processes driving the introduction of VOs and VBEs within the automotive domain.

2.2.1 Introduction and Basic Structure

To understand the nature of automotive development projects it is important to understand how the suppliers relate to each other. As presented in [1] the suppliers within the automotive industry are typically viewed as forming a hierarchical structure. The top tier of this hierarchy comprises the original equipment manufacturers (OEMs). These companies are the ones responsible for delivering complete cars to customers and are the public face of the industry - BMW for instance. The next stage in the hierarchy are tier 1 suppliers who are responsible for the delivery of substantial parts of the car, such as motors, as single units.

The hierarchy then continues down with companies in lower tiers delivering smaller components and the bottom tier being companies providing raw material or standardised parts. The exact number of tiers within this model is not universal, for instance [1] goes down to tier 3, whereas the equivalent model given in [63] goes down to tier 4. The essential nature of the two models is however the same with OEMs and Tier 1 suppliers on top and hierarchical levels of suppliers beneath them. As well as representing the size of components supplied by each tier the hierarchy also reflects the power structures within the industry - every business opportunity ultimately derives from the need of an OEM to produce a car.

Traditionally OEMs have taken responsibility for the selection and coordination of every supplier within their development projects and designed and/or manufactured many of the parts in house. Recently, as mentioned in [1], OEMs have been steadily reducing the amount of actual manufacturing they perform in house, preferring to focus on design, brand management and the final assembly. Indeed, as noted in [1] the OEMs are also moving a substantial amount of design work onto the suppliers.

A further extension of this trend, known as modularisation, is that certain OEMs are giving tier one suppliers responsibility for the selection and coordination of the sub suppliers used to construct a specific submodule within a car. From the OEMs point of view this has the benefit that they need only manage the suppliers directly responsible for each module rather than coordinating every sub supplier within the project.

The extensive case study presented in [63] illustrates how modularisation works in practice. This paper presents a case study from the initial phases of a project at AeroCo to manufacture a new jet aircraft. This domain has many parallels with automotive manufacturing, although each plane has many more components than a typical car - the paper mentions three million for a large jet as against thirty thousand for a car.

AeroCo has been 'progressively upgrading' its suppliers to tier one and gives each tier one supplier full responsibility for a specific module, including the management of the lower tier suppliers working underneath it. The basic structure of the project studied has six phases. The first of these involves making an initial supplier selection, initially through invitations from AeroCo. Conceptual designs for each of the modules were then independently produced by the tier 1 supplier responsible for that module in conjunction with its sub suppliers. Finally, detailed negotiations took place about the commercial terms.

Phase 2 then involved integrating the conceptual designs of the modules into an overall design and fixing the interfaces between them. Phase 3 featured each partner expanding the conceptual design of their module to a state supporting prototyping. By the end of phase 2, the main bulk of the codesign had been performed and an estimated 80% of the production costs determined. Once a prototype had been produced, the final phases involved integrated product testing before finally moving to full scale production.

This case study might seem to represent an idealised example of modular design in action and was viewed as a successful project by AeroCo. None the less the process of coordinating the codesign of modularised components was highly involved. The paper examines these issues in detail and concludes that such problems are inherent to such processes. A particular problem came from the companies responsible for each module focusing purely on reducing the individual cost of their module.

Since the costs of each module in the aeroplane are dependent on the specification of the modules with which they interact this can lead to the overall design becoming suboptimal. For instance it might be theoretically possible to dramatically reduce the cost of one module at the expense of increasing the cost of another. However the individual ownership of each module ensures this will not be considered.

One feature of the approach outlined in the AeroCo case study is that each individual module is first designed at a conceptual level and then integrated together. As noted by Sobek, Ward and Liker in [44], this is typical of automotive development projects - single best designs for each component are individually sought and then modified until they fit together. Within the paper they contrast this with Toyota's highly successful approach to development which they characterise as 'set based concurrent engineering'.

One example they give to illustrate the difference is in developing the styling for a vehicle. In many companies the desired styling for the car is fixed using a single clay model before detailed design of the components starts. Toyota, in contrast, develop several full scale clay models simultaneously and additionally develop structural plans defining a range of potential styles. This explicit specification of the set of possible styles allows their manufacturability to be considered in the final choice rather than making an initial choice of styling and changing it when it proves not to be easily manufacturable.

The paper contains a detailed exposition of set based design but in summary it involves the following stages:

1. initially the design space is mapped;

2. every party involved in the design independently defines the set of designs they consider feasible within the overall space;

3. these sets are communicated to other departments - in contrast to discussing a

single design;

4. the intersections of these sets are sought, with an emphasis on *minimal* commitment at each stage.

This idea of set based design is of particular interest as the basic idea of exploring sets of possible designs will be reflected in several systems later examined within this literature review and in the system for supporting virtual organisation formation proposed within this thesis.

An examination of the process of modularisation within the automotive industry, with a particular focus on contrasting the approaches of Japanese and European companies, can be found within [45]. The major distinction between the modularisation in the two areas is given as Europe concentrating on 'assembly' but Japan on 'development' - European modularisation typically does not devolve the actual development process. For instance, the paper mentions a case where Volkswagen mandated both the use of certain suppliers and the arrangement and cost of the parts that a module supplier must use while still requiring annual cost reductions from them. The AeroCo example above is much more closely related to the Japanese model than the European one.

As the paper mentions it is not clear what overall benefits can be expected from moving the subassembly of the same components from the OEM to a sub supplier. The paper lists six advantages such as the ability to request annual cost reductions from the module suppliers, the ability of the OEMs to coordinate far less suppliers etc. The main direct beneficiary of these advantages is the OEM.

Intriguingly the paper also mentions an effect of modularisation which is potentially damaging for the automakers - if the OEMs reduce their role to merely assembling modules supplied by other companies, those companies can gain considerable leverage over them. This could potentially cause particular problems if the OEMs later wish to control the cost that the module suppliers charge them. Japanese automotive OEMs treat this risk very seriously and are consequently reluctant to consider total modularisation whereas the European OEMs do not as yet worry greatly about this. The ability of a VO to gain such leverage over the OEM to which they provide a module provides one of the strongest motivations for VO formation within the automotive domain.

2.2.2 Identification of Assumed Abstract Domain Features

To recap, the problem domain for this thesis is defined by the following items:

- team formation typically takes a real period of time;

- the output of the team formation process does not need to be directly executable;
- the individual business opportunities have sufficient worth to motivate companies to offer custom processes potentially requiring investment;
- a typical business opportunity consists of a request to manufacture a certain amount of some concrete product.

The first of the requirements is designed to distinguish the problem domain of this thesis from situations such as web service composition where the result of the 'team formation' process must be produced both instantaneously and on demand. This contrasts with the automotive domain where more time is allocated to the creation of the correct consortium.

In particular there is typically a very considerable length of time between the formation of a team to produce the designs of a prototype and that design moving into full scale production - the design project studied within [63] ran to time and took three and a half years. For comparison [1] mentions that projects can take up to eight years from the start of initial conceptual design to the start of mass production.

Indeed this information, and in particular the extensive gap between the award of the contract and the commencement of mass production, makes it clear that second requirement is also met. With so long for a formed team to operate there is certainly no need to produce a complete, concrete process for manufacture before bidding for a contract. In fact to do so on the time scales required for making bids would not be possible. This is in direct contrast to the situation when - for example - combining webservices where the result must be an immediately executable program.

The case study given in [69] presents an illustration of the third domain attribute. This case study mentions bidding for contracts and receiving contracts to start production in between three and five years time with substantial development costs coming in the years immediately following the awarding of the contract. Indeed, since one must have a fair idea of how much the component will cost to make before submitting a bid, or risk being tied into an underbid contract, there are substantial development costs involved in even making a bid.

The study demonstrates how such development projects require considerable investment of time, effort and money by the companies involved. This provides more than sufficient motivation for companies to offer customised processes for lucrative opportunities. For example a company might purchase of new machinery in order to participate in a VO.

The fourth requirement was specifically designed to reflect the nature of development projects within the automotive domain. In fact many companies within the automotive domain focus on supplying services such as logistics rather than directly supplying concrete parts.

When considering which of these services have a real impact on the process of forming individual virtual organisations, a clear distinction can be made between services that a company will require as part of their normal operation - such as machine maintenance - and services which are specifically needed for the virtual organisation - such as the logistical services to run the manufacturing chain. This second type of service forms a major part of the overall cost of the virtual organisation and should be taken into account when potential virtual organisations.

In this thesis ancillary services such as finance and administration are assumed to be provided internally by the companies requiring them and to not directly impact choices between potential VO members. Since the VO itself forms in direct response to a request for quotes there is no need for it to advertise beyond putting in a bid for the contract: ultimately it is the responsibility of the OEMs to sell the cars produced.

2.2.3 Automotive VOs and VBEs

An important factor in the recent rise to prominence of automotive VBEs has been the adoption of the process of modularisation by many OEMs. The OEMs initially adopted modularisation as a means to save themselves money and it indeed offers few direct benefits to tier 1 suppliers. Modularisation does however offer a potential opportunity to groups of lower tier suppliers: it allows them to form virtual organisations and then to bid to supply entire modules.

The ability of virtual organisations to bid for large scale projects provides several benefits for their members. Of particular note is the additional security of contract gained - it is much harder to replace the supplier of an entire module, or to force them to lower costs by threatening to do so, than it is to replace the supplier of a small, standardised subcomponent. Further the high costs and risks in developing novel parts can be shared between the companies within the virtual organisation.

These potential advantages have led to the formation of VBEs in the automotive sector. These are typically referred to as automotive clusters, one example of which, the Austrian Automotive Cluster (AC), is described in [1]. The ACs membership is formed from a group of more than 280 individual suppliers, who mainly come from tiers 2 and 3. Each of these members funds the automotive cluster which in return provides several benefits.

The primary benefit derived by these members is greater ease in cooperating and in particular in forming virtual organisations. In addition the cluster can itself can offer certain services to its members. For example, the AC offers a system of 'quick checks' in which the AC provides an objective evaluation of the state of its member's plants. While this is currently quite limited there are plans to develop this into a third party certification scheme, whereby the AC guarantees the quality of its individual members. This would allow the individual members to benefit from a form of shared brand identity.

The potential gains from such a shared brand identity within a different domain can be seen in the case study presented in [5] which reports on a study of a UK company Cavendish Management Resources(CMR). At the time of the study this company had 150 members who each pay a fee to be a member of the organisation in addition to passing membership tests. The companies actual offices consist solely of one office containing a few secretarial staff.

Individual members then derive several benefits from membership:

1. They may get work passed onto them through the network. A percentage of the income from such work is returned to the company.

2. They can smooth out their otherwise erratic workload.

3. They can seek work in CMR's name enabling them to capitalise on its reputation and 'present a scale of operations which they could not achieve in isolation'. A percentage from such income is returned to the business. This is achieved though such aspects as using unified stationery.

The individual members of CMR remain free to seek work in their own capacity. The following summary of what is offered to CMR members is presented

> Effectively for many individuals, membership of CMR therefore becomes a "club" for the group promotion of one's own expertise within a marketplace which trusts scale, and within which a single individual will always remain "contact-challenged" in comparison to a larger collective with more ears to put to the ground.

.

That people are prepared to invest not only a membership fee but a percentage of the income generated in this way illustrates how important such considerations can be. While the settings that the AC and CMR work within differ greatly, the fundamental motivations are very similar. In particular, if a VBE possessed a valued brand name, any virtual organisation forming within it can present themselves as coming from that brand, and benefit from the consequent impression of scale.

2.3 Stages of VO formation in practice

There are many aspects to the process of virtual organisation formation, ranging from locating the opportunities to the final negotiations as to how the companies plan to share out the profits gained by the virtual organisation. The following list of stages is taken from deliverable 23.1 on the ECOLEAD project, [18], a four year European integrated research project looking specifically at the formation of virtual organisations within virtual organisation breeding environments.

1. Collaboration Opportunity Identification;
2. Draft VO Planning;
3. Partner search and selection;
4. VO Composition and Negotiation;
5. Detailed VO Planning;
6. Contracting;
7. VO Set up.

These stages provide a useful means of structuring discussion of the existing work and are used for this purpose here. The boundaries between the different stages are not as clear cut as they can appear from this list. In particular certain systems combine the two phases of draft VO planning and partner search and selection.

The remainder of this chapter discusses how the specific nature of the application domain of this thesis affects each of these phases, using the automotive industry as a reference. This then provides the basis for the next chapter which uses this information to consider how well suited certain existing approaches are to supporting VO formation within the target domain.

2.3.1 Opportunity identification

Within the context of the automotive industry, collaboration opportunities occur when a company, typically an OEM, requires a consortium to supply it with some major component of a car. Companies can discover such opportunities in several ways - they might be advertised on electronic markets, the OEM might directly contact appropriate companies or a company may notice that a particular model is likely to need replacing and from this identify the possibility of supplying a major module.[1].

[1]An example of this can be seen in the case study within Section 7.3

The common element to all of these methods of discovery for business opportunities is that there is no pressing need to actively look for or to attempt to proactively create business opportunities. Thus the main interest lies in the information contained within a typical business opportunity. A typical automotive business opportunity might contain the following elements:[2]

1. the expected lifespan of the project;
2. the frequency with which this product must be delivered;
3. the average volume of product which must be delivered;
4. to whom it must be delivered;
5. CAD drawings giving a detailed description of the part required.

Additionally OEMs sometimes have long term contracts with suppliers of a particular component and mandate that these suppliers must be part of any consortium supplying them with a larger component. This thesis does not concern myself with the exact format in which business opportunities are published, and instead assumes that any business opportunities discovered can be mapped into the standardised format discussed in Section 6.2.1.

2.4 Draft VO Planning and partner search and selection

Once a business opportunity has been identified the process of VO formation can commence. The ECOLEAD deliverables split the initial stages of team formation into two distinct phases - draft VO planning and partner search and selection. Of these draft VO planning is taken to consist of:

- determining the required competences and capacities;
- determining the rough organisational form of the VO and the required roles;

Partner search and selection is taken to consist of:

- identifying potential partners for each required role;
- evaluating these suppliers and selecting the preferred ones.

[2]The following section is based on confidential information/discussions within Crosswork and Sudden

It is interesting to consider if these two phases within team formation can be combined, thus attempting to cooptimise the choice of team structure and the selection of suppliers. This possibility is raised within ECOLEAD deliverable, [18], but only in the context of a system which informs the VBE members of the existence of a new business opportunity and subsequently expects virtual organisations to form spontaneously. The deliverable rejects such systems on the grounds that since

> Acquiring new collaboration opportunities and forming an appropriate consortium requires strong motivation and commitment,

it is not reasonable to expect such consortia to form spontaneously.

This argument is weak in any domain within which the value of the potential contract is sufficient motivation to drive VO formation. The strong motivations driving VO formation within the automotive sector have already been presented and are shared by the problem domain of this thesis.

While the motivation for VO formation might be strong, the actual VO formation process is far from trivial - the individual VBE members must discover which processes the other members of the VBE can offer, which processes are required to manufacture a given component and combine these to produce a team. While it is certainly reasonable to expect individual companies to possess knowledge about when their processes are typically useful it is much less so to expect them to possess detailed knowledge of the other members of the VBE. The above argument suggests examining methods for *supporting* VO formation which do not rely on the use of an initial, centrally mandated, decomposition.

An additional factor pertaining to partner selection within the current domain relates to the final choice of team. Traditional approaches to VO formation typically aim to produce a small set of candidate VOs which are each judged to have high performance against external criteria. In situations where the person driving the VO formation is also responsible for the selection of the final team - an OEM selecting its favoured suppliers for instance - this is reasonable.

Within the current context the assumption is that teams form within the VBE before competitively tendering - in competition with teams from outside the VBE - for the contract from the OEM. In this context there is no obvious candidate for an centralised evaluation procedure - the entity responsible for selecting the final team is not directed involved in the VO formation process and the central element of the VBE lacks the required authority to impose one.

In addition there is a natural tendency for the individual VBE members to disagree on the relative merits of teams: each member is likely to show a particular preference for VOs of which they are members. Thus that such a system should focus on facilitating the formation of whichever candidate VOs the VBE members consider it reasonable to form. This change of emphasis suggests that careful consideration should be given to distributed techniques for supporting VO formation.

2.4.1 Partner Search and Selection

The most notable feature of partner search and selection within the automotive domain is its considerable difficulty, something which is principally caused by the following factors:

1. most automotive components are specifically designed for each new module and so cannot be replaced off the shelf;

2. the fact that they are provided by companies.

The first of these is the principal reason for the length of time that such development projects take - every component must be carefully designed to work both individually and in combination. The second factor is a problem because it means that it is very difficult to use external adverts to represent what a company can do. For example if a company could only participate in a contract by buying a new machine they might well do so if the contract was lucrative enough. Alternatively they might be technically capable of participating in a project but either be simply too busy with existing orders or unwilling to work with certain other companies.

The net effect of these two factors is that it is very difficult to use fixed advertising to represent the capabilities of companies. Instead one must represent the company's capabilities at an abstract level and rely on the companies themselves to provide more detailed information relating to their precise capabilities. The effects of this on the process of partner search and selection are examined in the next chapter.

2.4.2 VO Composition and Negotiation, Detailed VO Planning and Contracting

These three phases are interlinked and will be discussed together. As presented in [18] the VO Composition and Negotiation and Detailed VO Planning phases involve a loose set of companies who expect to be be able to work together making explicit agreements as to how they plan to work together within a VBE. Detailed contracts governing this are then constructed within the contracting stage.

The primary distinction between the VO composition and negotiation phase and detailed VO planning is when they happen - VO composition and negotiation happens immediately after, and to some extent during the selection of team members, and involves checking if the initially selected companies are happy to work together. Indeed the deliverable mentions that the originally selected set of team members might have to be reconsidered as a result of this. Detailed VO planning commences once there is a strong expectation that a team will be able to form and examines the relevant issues in much more detail.

While similar problems are met when forming VOs within any domain, certain additional domain specific problems are met within the current problem domain. For example automotive VOs form before every company has decided exactly which process they plan to provide to produce the overall product. Indeed the companies will not be able to do this in general without first performing some collaborative design. Moreover the various processes that each company could use will frequently have differing costs while the coalition must bid to gain the contract and so must aim for the lowest possible combined cost. Thus ideas such as fair profit sharing become important and the negotiation of contracts becomes even more involved than usual.

Another notable domain feature is that the size and complexity of the contracts under consideration, especially in comparison to the size of the companies involved, is so large that there is very little chance of automated negotiation ever being acceptable. This suggests that systems which aim to support such negotiations might be more useful than systems focusing purely on automatic negotiation.

2.5 Summary

The concept of supporting VO formation within a virtual organisation breeding environment has been chosen as the basic theoretical basis for this thesis. In addition to this, a set of abstract domain features have been proposed, which tightly categorise the problem domain under consideration within this thesis. These features have been illustrated using the example of the automotive domain. Finally the effect of these features on the process of VO formation has been discussed and will form the background to the next chapter's examination of existing systems for supporting VO formation.

Chapter 3

Existing support for VO formation

The problem of supporting virtual organisation formation has a broad scope and there are many active areas of research of relevance to it. This chapter reviews these areas with a particular focus on their applicability to the application domain introduced in chapter 2. This discussion is structured using the stages of VO formation, taken from [18] and listed within the previous chapter.

3.1 Techniques for Opportunity Identification

As discussed within Chapter 2, the automatic identification of opportunities is of little importance within the current domain. Within other domains, a potentially interesting approach investigated within the context of the ECOLEAD project, and detailed in [19], is to automatically trawl the world wide web using data mining techniques.

A more relevant problem to this thesis is that of ensuring that the data contained within a business opportunity can be mapped into a form usable by the system. Since the details of such mappings are very context dependent, it is assumed that any business opportunities discovered can be mapped into a standardised format presented later in Section 6.2.1. This assumption parallels that made within the ECOLEAD business opportunity discovery software.

Indeed as seen within, for instance [65], this approach of translating from a potentially wide variety of input types to a single type used internally is also used by webservices. However within this domain the basic idiom is that of a user inputting structured natural language requests to a computer system. Some examples given within [65] include requests for five day forecasts and booking a flight between Pittsburgh and New York. Such requests are both much less detailed and much more varied in structure than requests to manufacture a given automotive component. Thus the difficulties encountered in translating such requests into usable form, while interesting, are both different and greater than those addressed within this thesis.

3.2 Techniques for Draft VO Creation

3.2.1 Centralised Planning

The problem of draft VO planning is strongly analogous to a standard planning task. This section provides a brief introduction to planning in general. Particular emphasis is placed on drawing a comparison between the efficiency of hierarchical and standard planning. The natural analogies between these types of planning and methods for supporting VO formation - hierarchical planning corresponds to systems which to first decompose the business opportunity and then search for team members to fill it while more standard planning corresponds to generating the team structure from scratch - gives this discussion relevance to the main topic of this thesis.

As noted by Weld in [85] a traditional planning problem is typically concerned with the problem of a single agent deciding on a course of actions to achieve some goal, and is defined by three elements:

- a description of the initial state of the world in some formal language,
- a description of the agents goal in some formal language,
- a description of the possible actions in some formal language.

The basic problem is then to work out which set of actions the agent should perform in order to achieve its goal. The precise details of the formal languages used have been found to be of vital importance for the overall efficiency of planners and many alternatives have been proposed, several of which Weld describes in [85]. This level of technical detail is not required for the purposes of the current discussion, where it is sufficient to consider the potential actions as defined by a set of preconditions, things which must hold for the action to be used, and postconditions, the actual effects of applying the action.

The simple approach to developing plans is then to search the space of all legal combinations of operators to find a combination which produces a world state in which the desired goal holds. However the size of the search space typically grows too quickly for this to be viable. In order to work around this problem, many different algorithms have been proposed and explored in depth. An example of a fairly modern and successful planner, Graphplan, is described in [8] and several examples can be found in [85].

One major distinction within planning is between forwards and backwards chaining. These terms refer to the direction in which the operators are combined:

- Forwards chaining starts from the initial state and builds towards the goal state by applying operators in a forwards direction.
- Backwards chaining starts from the goal state and applies operators backwards to reach the initial state.

As an example, consider the problem of finding a route travelling from Manchester to London. A forwards chaining search would start in Manchester, look at the set of destinations reachable in one step from there, and work onwards until London was reached. A backwards chaining search would start at London, look at the range of places from which London was reachable in one step and work backwards until it reached Manchester. While the complexity of the search in the above example is largely unaffected by the choice of planning direction, in other domains there can be a significant difference.

This example additionally highlights the usefulness of intelligent search control - a brute force search would consider options such as first travelling across the country to York before going back down to London whereas a human would only consider options taking them closer to the actual destination. It is desirable to design a planner to use this information and indeed the A* algorithm, introduced by Hart, Nilsson and Raphael in 1968 in [41] does this - it uses the difference remaining in a journey to guide a route search in fruitful directions.

This kind of search space control does not prevent the location of optimal results and can provide considerable performance benefits. The major difficulty is working out a reliable method to compare two potential directions of 'travel'. A frivolous example based on the previous one would be if there were an instant travel portal connecting York with London - then going from Manchester to London via York would potentially make sense. More seriously, in many domains there is no natural choice of metric for determining the usefulness of individual contributions and one must instead rely on ad hoc heuristics.

An alternative way to speed up the planning process is to provide a loose initial structuring to the plan. These ideas have been developed within the context of hierarchical planning. The idea behind this form of planning, originally introduced by Sacerdoti in [71] and since developed in numerous papers, is to start the planning process by selecting an appropriate complete but highly abstract plan. The individual abstract steps in this plan are then progressively refined using more concrete processes and finally a complete, detailed plan is produced.

While hierarchical planning is sometimes more efficient than the planning techniques previously mentioned, it can also be less effective under some conditions. The following section examines which factors affect the comparative performance of hierarchical and standard planning.

Comparative Performance of Standard and Hierarchical Planning

Several papers offer comparisons between the performance of normal and hierarchical planning, for instance [47, 4]. The conclusion from these papers is that the use of hierarchical decomposition can offer considerable performance benefits but that these are not universally seen. Indeed hierarchical planning is found to be less efficient than normal planning under certain circumstances. The following properties are offered in an attempt to categorise those domains in which hierarchical planning is most useful:

The ordered monotonicity property , defined as follows: *"For all abstract plans, all refinements of those plans leave the literals established at the abstract level unchanged". [47]*

The downwards refinement property [4], which holds if any abstract plan can be successfully refined into a plan at the next level of abstraction down.

In essence the ordered monotonicity property requires the distinct elements of the abstract plans to have independent effects on the real world. When this property holds it permits the choice of how to expand each element of the abstract plan to be made independently. The downwards refinement property requires a good match between the set of subplans available for refining the abstract plan and the abstract plan itself. When it holds it allows the abstract plan to be expanded without fear of being left with subtasks which cannot be fulfilled, something which would require undesirable replanning.

The construction of hierarchies with these properties can be a difficult task requiring considerable amounts of domain knowledge. Knoblock notes in [47] that the manual production of efficient decomposition hierarchies is far from simple and in [47] he proposes a technique for their automatic generation. This technique takes the set of all possible low level operators as an input and automatically derives an abstraction hierarchy which meets the ordered monotonicity property. This suggests that a minimal requirement to form an efficient hierarchy is knowledge of the *types* of concrete processes available.

If we consider the fourth domain feature of this thesis, we are interested in developing a team to produce a concrete product - a car door for instance. Consider that this task has been decomposed into needing to produce the car window, the door frame and the door covering. In this case, for ordered monotonicity to hold perfectly the value attached to making a choice of supplier for one submodule should be independent of the choice of suppliers for the other submodules.

This property does not hold in the automotive domain - the overall efficiency of a group of automotive companies is strongly dependent on both the logistics of that team and the degree of cooperation within the group of companies. In addition the lack of fully standardised components means that two suppliers identified as capable of producing the same class of component might in fact have quite different capabilities. It is thus possible to select a team where every individual supplier can provide the capabilities required but the combined team cannot. This problem is especially likely if the initial decomposition operates at a high level of abstraction.

An additional obstacle is that the knowledge required to produce a decomposition hierarchy is often lacking. For well understood parts, the set of subcomponents might be known from experience and be quite stable over time. For less understood parts, as are often encountered in the automotive industry, the existence of such knowledge is not guaranteed. Additionally, since both the set of processes offered by the suppliers within the VBE and the overall VBE membership vary dynamically, there is no solid basis for the development of such a hierarchy.

Finally the use of 'a priori' decompositions can limit the set of potential solutions considered. Even when the product to be manufactured is well understood and the set of subcomponents required easily decided upon this can be restrictive, especially when the initial abstract plan itself contains submodules. Such cases offer extensive opportunities for variety in the assignment of subsubmodules to the individual submodules.[1]

Thus hierarchical approaches to VO formation are not ideally suited to the problem domain of this thesis. More precisely they correspond to a model where a central supplier must find specific suppliers to fulfil a specific design rather than to supporting VO formation within a VBE. This strongly suggests investigating techniques for supporting cooperative VO formation within a VBE without using a decomposition. On this note, it is interesting to see it reported in [53] that both hierarchical and standard planning have been used to address the related problem of automated web service composition.

3.2.2 Distributed Planning

The planning techniques examined within the previous section made the assumption that there was a single entity in overall charge of the planning process. An alternative approach is to distribute the control of the planning process between multiple independent entities which produce individual subplans which are then combined to

[1]Hypothetical examples of this can be seen within the case study presented later in this thesis

produce an overall plan. Such approaches have proven especially fruitful in two circumstances: some problems are simply too big for a single, centralised model to be used and sometimes it is not possible to gather the data needed to form a centralised model.

The problem domain of this thesis is an example of the second type. The knowledge required to break a business opportunity into subparts - namely which processes the individual companies are able and willing to contribute to the given business opportunity - varies according to the specifics of the business opportunity and company workload and so is only known by the companies themselves.

Research in distributed systems can be distinguished according to whether the controllers have individual agendas, as is typically considered the case in multiagent systems research, or form part of a single coherent system - this is termed distributed problem solving by Durfee in Chapter Three of [84]. The formation of virtual organisations is a classic example of the first case, in that it features multiple self interested companies each with their own agenda. The stages of distributed problem solving nonetheless provide a useful basis for discussion. Durfee distinguishes four stages within distributed problem solving:

1. Task Decomposition;
2. Task Allocation;
3. Task Accomplishment;
4. Result Synthesis.

The decomposition and allocation steps here refer to deciding on a set of appropriate subtasks and assigning them to individual sub suppliers, accomplishment to the suppliers performing their subtask and synthesis to the problem of recombining the individual results of the subtasks.

In the context of supporting cooperative virtual organisation formation there is a natural solution to task decomposition and allocation - every company is represented by an entity with responsibility for their interests. The task accomplishment phase then corresponds to each of these agents volunteering to contribute a certain process to a partially formed virtual organisation.

The final stage, result synthesis, then involves combining these contributions into a set of virtual organisations and identifying when complete virtual organisations have formed. Two principal issues must be addressed in this stage - firstly the individual contributions from every agent must be brought into one place and secondly they must be combined into integrated solution(s). The following sections examine each of these stages in turn.

3.2.3 Architectures Supporting Plan Sharing

Multi Agent Systems

One way to support the sharing of partial solutions is to allow the individual planning elements to directly communicate with each other. The field of multi agent systems combines this idea with that of making each planning element an independent, self interested, reasoning entity.

Before discussing elements of multi agent platforms in detail it is useful to precisely define the term agent. As made clear by Wooldridge within chapter 1 of [84], there is no universally accepted definition for the concept of a software agent. The term agent was in use long before the idea of software, let alone software agents, existed and was used in several subtly different contexts. Of these contexts the following dictionary definition taken from [59] is perhaps the meaning from which the idea of software agents was built:

> [Agent] 5. Of things: The material cause or instrumentality whereby effects are produced; but implying a rational employer or contriver.

Although software agents are typically discussed in anthropomorphic terms this definition fits them quite well. It is however too general, admitting as it does the typical example of a thermostat. The approach taken by Wooldridge and Jennings in [46] is to list a minimal set of additional characteristics that a software agent is required to exhibit. In this definition a software agent is any piece of software is posessing the following characteristics: [2]

1. Autonomy;
2. Social Ability;
3. Reactivity;
4. Pro-activeness.

While this was explicitly designed by the authors to be a minimal definition and has been widely adopted as such, many people use the term agent in a somewhat looser fashion. This thesis exemplifies this in that, while some of the agents within the system do in fact meet this definition of agency, several of the things later termed agents fail to meet this test. This split between the agents within the system comes from the way the system focuses on supporting companies in forming teams. As will be seen later this leads to a natural split between those agents representing

[2]each of the characteristics is defined precisely in the paper but basically retain their usual meanings

the companies, which are responsible for intelligently driving the system, and those 'agents' responsible for facilitating the systems operation.

It is this second category of agents which fails to meet the definition above. They feature a social ability in that they communicate with the agents representing the companies and are reactive to the ongoing process of team formation. Where they fall down is arguably the most distinctive element of pro activity - by default these agents have no 'drive' and merely respond mechanistically to messages from the more intelligent agents within the system.[3]

Such elements within a multi agent system which exist purely to facilitate the intelligent operation of the remainder of the system are often called agents. An example is the directory facilitator service that FIPA requires to be present within any agent platform, [29]. This service exists to store information about what each agent in the system can do and answer queries about this and the specification mentions that it may be implemented as an agent - as it is in JADE, [36] for instance. Such an agent exhibits little proactive behaviour thus raising the question of how essential proactivity should be to the definition of an agent.

However proactive behaviour is the main distinguishing mark between agents and active objects and so must be retained. A suitable compromise position can perhaps be sought by focusing on multi agent systems instead of *individual* agents. This would allow a definition to require that the systems overall operation is driven by proactive, intelligent agents while allowing any element of such a system meeting the other three criteria to also be regarded as an agent. This approach permits a unified exposition and is adopted for the remainder of this thesis.

Agent Communication and Ontology

Fundamental to any agent system is the method used to communicate between the agents within it. The system proposed in this thesis will use the multi agent system JADE, [36], whose message envelopes are derived from the standard proposed by the main agent standards body FIPA, [34]. This provides a standard structure along with a set of performatives to indicate the basic goal of the message. For example a REQUEST message indicates that the agent wishes the receiver to perform a certain action. A detailed description of these performatives can be found in [32] and [30].

For such messages to be meaningful they must be populated with concepts and predicates having a natural meaning for the agents sending and receiving the messages. The difficulty in doing this is ensuring that all of the agents within the system share a meaning for these concepts. While it is never possible to be *certain* that an

[3]Later extensions making fuller use of their status as agents will in fact be discussed but do not affect this discussion

autonomous agent will not misinterpret message content the risk can be minimised.

This is typically done by explicitly defining the set of terms that the agents can be used to communicate - such a set of terms is typically referred to as an ontology. The study of ontologies within the context of philosophy has a very long and rich history. Indeed as indicated in [74], which presents a historical development of the theory of ontology as it applies to computer science, it dates back at least as far as Aristotle.

One interesting aspect of this is that it shows how the idea of ontology, as applied within computer science, was developed independently of that from within philosophy. Indeed this is not so surprising due to the differing focuses - within philosophy the main focus is on the problem of what sorts of concept exist in principle. In computer science the main focus is on the more pragmatic problem of providing a description of enough aspects of a domain to facilitate either reuse or communication about the objects in that domain.

These different motivations account for the considerable additional rigour seen within the philosophical branch of ontology. This paper however is purely interested in ontologies for supporting communication and this thesis adopts the following definition from [40]:

An ontology is a formal, explicit specification of a shared conceptualisation

Within computer science there are different levels of rigour within the ontology definitions. This can be seen by the contrast between ontology languages designed to enable rich reasoning such as OWL, [54], and the ontology language used within the multi agent system JADE,[36]. OWL, a World Wide Web Consortium(W3C) standard, is based on a subset of description logics and provides a rich set of ways of defining classes in terms of other classes and properties while at the same time restricting their use so that reasoning can be efficiently supported. This ability to reason efficiently is why OWL uses description logic rather than first order predicate logic as its basis.

On the other hand JADE's ontology language is simply designed to facilitate communication between agents. Thus it contains a set of classes of concepts, predicates etc each of which is a frame with certain attached attributes. Each of these attributes can have attached facets constraining their legal values and the classes can be sub/supertypes of each other. While JADE's ontology language is technically more expressive than OWL certain elements of it such as the use of arbitrary JAVA statements to constrain attribute values make effective reasoning over such ontologies (beyond simple sub/superclass comparisons) impossible.

However to worry about this is to miss the point of the ontology support within JADE: the classes are simply used to provide a structure that the concepts within messages should follow and to verify that they do in fact do this. For these purposes the JADE ontology language is highly suitable. Since it also sufficed to support the reasoning required it is adopted as the main ontology language of the system developed within this thesis.

Blackboard Systems

The direct communication used within multi agent systems can prove problematic in scenarios where there are many agents, all of whom must be informed of the results generated by every other agent. These concerns motivate techniques which control the quantity of broadcast messages required by providing a centralised place where agents can 'post' their partial results. One important class of such techniques is known as blackboard systems.

The blackboard paradigm is an architecture for constructing AI systems which was first introduced by Hayes Roth in several papers such as, [42]. The field has a rich history, a discussion of which can be found in [14]. The original inspiration for blackboard systems derived from how groups of human experts work concurrently on a problem in a room with a blackboard - based on the information on the blackboard every expert decides if they can contribute and if they can, does so by writing a contribution on the blackboard.

This conceptual model translates into a system which contains a central software blackboard and a set of knowledge sources. The central blackboard holds both the initial solution and all partial solution contributed towards it while the knowledge sources contribute to partial solutions when possible.

A major motivation behind the development of blackboard systems was the need to integrate disparate types reasoning of program into one overall system. In addition they provide the opportunity for the opportunistic reasoning - a knowledge source can be inspired by a partial solution and thus suggest a contribution which they otherwise would not have considered.

An in depth case study examining the reasons to support opportunistic planning was presented by Hayes Roth in [43]. In this paper, which provided the initial conceptual basis for the later development of blackboard systems, a detailed case study of how people plan a day's shopping trip in which they needed to visit certain shops is given. The authors found that the people planned their trips in neither a purely top down or bottom up manner but instead followed the opportunistic model later seen in blackboard systems.

This support for opportunistic reasoning is especially valuable for supporting knowledge sources which specialise in providing customised support for specific classes of partial solution. Within the automotive domain these might correspond to manufacturers of bespoke items. Such companies cannot a priori precisely specify the types of project or parts which they can contribute, but instead must be allowed to identify opportunities they consider promising and to contribute as they consider appropriate.

In the original interpretation of the blackboard metaphor there was a single fixed scheduler responsible for activating each of the knowledge sources when appropriate. This was typically done with rules saying that when content of a particular type was placed on the blackboard given knowledge source(s) should be activated. This is however quite inflexible and in [42] the idea of having a separate section of the blackboard containing control decisions was presented. This allowed the scheduler and hence the blackboard system itself to adapt to changes in the problem structure by altering its own scheduling rules. Another important feature of traditional blackboard systems is that *all* the communication is required to go through the blackboard - there is no direct negotiation between the knowledge sources.

Combining Blackboards and Agents

It is interesting to consider how accurate an interpretation of the original metaphor the original design of blackboard system actually is. In particular a large group of people working around a blackboard will typically not restrict themselves to solely communicating on that blackboard - where appropriate they will split off to conduct face to face group discussions. Also they do not necessarily wait politely to be told when they are allowed to write on the blackboard.

When blackboards were originally developed the prevailing AI technology was expert systems which concentrated on solving a given problem. In particular there was no specific notion of how such systems should communicate with each other except through the communication of their results, or their partial results. The traditional design of blackboard system provided a natural and effective way to combine such systems into a single coherent whole.

The recent development of multiagent systems provides an opportunity to develop the original metaphor in a different way. A paper which proposes this is [21], which advocates the idea of replacing the knowledge sources within blackboard systems with agents. The blackboard is then used for public broadcast communication while agents desiring more privacy can instead use direct communication. An example of a system that combines both blackboards and agents into a system for combining disparate pieces of software supporting collaborative design into a single coherent system is given by Lander et al in [50].

Without a centralised scheduler the performance of such a system is greatly dependent on the behaviour of the individual knowledge sources. In particular a badly behaved knowledge source can greatly affect the overall performance of the system. While this unpredictability is undesirable, it is an inevitable consequence of fully distributing control.

The distribution of control has several compensating benefits, including they way in which it allows the set of knowledge sources to dynamically change without affecting the system. In addition the ability of such systems to support private communication is beneficial when a small number of knowledge sources wish to communicate amongst themselves in a private manner.

Hybrid agent/blackboard systems represent an equally valid, and arguably truer, interpretation of the original blackboard metaphor. They are however best applied to slightly different types of problems to traditional blackboard systems. For instance the traditional interpretation is better suited to the control of mechanical systems where predictability is necessary, communication easily limited to intended actions and effects and every knowledge source/agent will be wholly focused on optimally controlling the system.

This is confirmed by Corkill in [20]. This paper concentrates on comparing multi agent and blackboard systems within the context of controlling a single system and finds the historical style of blackboard system to have certain advantages. However the agent based interpretation is better suited to situations where every agent/knowledge source has self interest, perhaps due to representing a company. In such circumstances it is vital to allow the set of knowledge sources/agents to change and to act as they wish. Further such agents may well wish to personally negotiate. The problem domain of this thesis requires both personal negotiations and flexibility in the set of knowledge sources present in the system and so the system presented within this thesis will use a hybrid blackboard/agent approach.

Tuple Spaces

The idea of Tuple spaces was introduced by Gelernter who proposed the Linda system in [39]. The purpose of Linda was to permit the programming of concurrent programs by permitting communication between concurrent processes which are distributed in both time and space. This was done by creating a central communication space - a tuple space - into which the processes could insert typed tuples. Other processes could then subscribe to the tuple space with a template defining types of tuple in which they are interested. These processes are then informed when such a tuple appears and can act on it.

While the basic model underlying such systems is closely related to that in the hybrid multi agent/blackboard systems from the previous section, they have typically been applied to different types of problem. One major application has been supporting multi agent communication and in particular supporting communication between mobile agents.

The essential difficulty met by such systems is that, since mobile agents have no fixed location, direct communication is often impossible. This problem is then avoided by placing a set of tuple spaces on the nodes of the network. The mobile agents then use these to 'post' messages to each other. Two such systems are TuCSoN presented by Omicini and Zambonelli in [62] and LIME presented by Picco and Roman in [68].

Both papers extend this basic model - TuCSoN proposes the use of reactive tuple spaces which in addition to passively sending/receiving tuples contain a set of reactive rules which fire at appropriate moments to either edit or produce new tuples from existing tuples and LIME proposes something similar.

3.2.4 Plan Merging

Closely related to the problem of combining processes proposed by companies is the problem of combining plans, and especially the problem of merging hierarchical plans comprising a set of plan steps linked by and/or connections. Each step in such a plan is be atomic or a combination of smaller plan steps and contains a set of pre, post and Inconditions. Inconditions are conditions which must hold while the plan is being enacted, pre and post those that must hold before it starts/which hold after it completes.

There are two basic approaches to merging such plans: either the individual plans generated are broken down into their atomic components which are then merged, or the individual high level plans are merged. The first of these approaches ensures the location of an optimally efficient final plan but is much more computationally demanding than the second. The second of these approaches is much simpler to enact but fails to account for any potential efficiency gains from the merger. Neither of these approaches is entirely satisfactory in all circumstances.

A very interesting approach to this dilemma is presented by Clement and Durfee in [16]. Essentially the system proposed within the paper attempts to indicate when it is profitable to consider merging the plans in detail, and importantly when it is not. It does this by calculating in/post/pre conditions for the overall plan from those of the individual steps comprising the plan.

To see why the naive approach of making every pre/post/incondition of an individual plan step a pre/post/incondition of the overall plan does not work, consider the case of a plan containing a step which can be expanded into either of two distinct subplans. Clearly to consider the overall plan's preconditions to be the *combination* of those of either of these potential choices is wrong - the combined preconditions of any choice of subtasks to execute the overall plan are either the preconditions of the first process or the second but never both.

A further problem with this approach is that it restricts the situations in which you might consider interleaving sequential steps. For instance, in a manufacturing context, suppose that two plans are composed of multiple sequential steps and the first step in the first plan is to use some machine, whereas this is the last step in the second plan. While these plans could probably be run in parallel, simply comparing the aggregated inconditions of each plan at a black box level would lead you to conclude that they both need the machine and must run in sequence.

Brooks and Durfee present a system which avoids these problems by introducing extra quantifiers on the pre/post/inconditions of each individual top level plan. There are two major types of these - one relates to *existence* and to *timing*. The existential quantifiers apply to all forms of conditions and are must and maybe indicating that the corresponding condition must/might apply to the overall plan.

The timing quantifiers are first, last, always and sometimes. First/Last are applied to pre/post conditions respectively and indicate respectively that the condition must be fulfilled before the overall plan can start or that will be true when it completes. Always/Sometimes indicate that the given condition must hold either throughout the entire enactment of the plan or that it simply must hold at some stage during it.

Each condition is then quantified using both a timing and an existential constraint. The paper provides full rules for recursively generating the overall conditions for a plan from those of its subplans. These generated conditions then allow the comparison of whether two plans can be merged or not. However their nature does not allow a definite answer - instead they can be used to discover if it *might* be possible to merge the plans or not, and sometimes how this might be possible.

If is indicated that it may be possible to run two plans concurrently, one can consider how the plans might be merged at a more detailed level by looking at the generated conditions of their subtasks at the next level down. This process can potentially continue down to examining the atomic tasks within the processes. The additional must/maybe quantifiers indicate where it is useful to consider merging plans, thus allowing the approach to create combined plans as efficient as those created by atomic task level comparison while being much faster than this process. It would be interesting to consider adapting this approach to the related problem of merging workflows discussed in, for instance, [26].

An alternative approach to merging plans was suggested by de Weerdt et al in [22]. Instead of focusing on the details of how the plans to be merged are executed this paper focuses on exploiting the effects produced by their operation, whether intention or unintentional. To do this, they consider the world to consist of a set of agents with skills which can be used to manipulate a set of resources.

The major focus of the paper is using this representation to allow different agents to merge plans and detailed algorithms for using it to do this. An example given in the paper is of two taxi companies who wish to cooperate. Instead of representing each taxi ride as a detailed itinerary for going to the next customer and moving on from there, they purely focus on illustrating the *effects* of the plan on the companies resource - namely that it places an empty taxi in the spot that it delivers its current fare. By focusing on this element they allow companies to efficiently make use of these effects by, for instance, selling a fare to another company who has a taxi near the required pick up point thus reducing the overall journey time.

This focus on the resources produced by the plans of agents rather than the details of how they plan to do so fits very well into the domain context of this thesis. Indeed, the main system presented within this thesis also focuses exclusively on the resources manipulated by the manufacturing process and how they are changed by the companies processes. The way in which the process representations are used - team formation as opposed to opportunistic plan merging are however quite different and beyond the resource representations used the systems share few features.

The problem of plan merging is of crucial importance in domains which require the production of directly executable teams. For example the result of web service composition must comprise an instantly executable composition of software. Within the automotive domain these techniques might be applied in two situations - one is during the formation of a potential team and the other in facilitating a potential team in making a detailed operational plan.

In the case of potential team formation, it is too restrictive to insist that each potential team member should decide on the process they plan to use to produce a component they offer to supply. Indeed, since they will not yet know the detailed design of the component it will in general be impossible for them do so. Further, even if there are minor conflicts between the proposed method of organising their processes the scale of the project in terms of time (years) and monetary value means that this is only a minor concern, rather than a primary criteria for partner selection.

In the second stage, the major concern of the partners is ensuring that sufficient quantities of correctly specified parts are delivered to the correct partners in time. The details of how their manufacturing processes might merge or not are of little importance. The problem of plan merging is thus of lesser importance when forming manufacturing virtual organisations.

3.2.5 Distributed Continual Planning

Not every application of distributed planning applications concentrates on producing a single plan. An alternative problem area is distributed, continual planning. A useful survey of work in this area is given by by desJardins et al in [23].

This problem domain differs markedly from that met when trying to form a static plan and then finding agents to enact it. The following factors are of particular importance: (1) *every* agent is involved in the formation of the distributed plan so there is no form of partner selection and (2) plan formation must be interwoven with plan enactment - the plan must take into account the effects of the actions of the other agents. These differences make such planning a different, arguably harder, problem than that addressed in this thesis. Only a brief survey is offered here.

One particular problem in such domains stems from the inability to create a single plan governing the entire lifespan of a team. This can lead to several problems. One - addressed in [81] - is the potential need to achieve rapid, potentially suboptimal coordination if surprised by events. In the paper the authors attempt to facilitate this by allowing agents to consider how long it will take the team to achieve a given level of coordination and how long the team has to coordinate before running into trouble. The reported results in the paper suggest that this could prove a useful technique.

A further potential problem in such domains comes from the limited look ahead possible when selecting a plan. This can lead to plans which look promising due to their utility gains in operation being undesirable due to leaving the team in a state from which they cannot avoid subsequent negative consequences. An interesting attempt to avoid this is given in [78] which uses learning techniques to derive estimates of the future utility that the team will derive from a given state. This allows the planner to consider this when forming an initial plan.

Such problems have been well studied within the domain of playing board games, where the complexity of the search in selecting the best move frequently forces truncation. In this context, in addition to static estimators such as variable search depth, techniques are used so that search do not terminate in a volatile situation[4] but are instead extended until a stable position is reached. It might be interesting to see if such techniques can be usefully applied within this domain.

In addition to such classical approaches some approaches have attempted to produces systems in which no plan is ever created but instead the desired behaviour is allowed to emerge from regulated interactions between the subcomponents of the system. A very popular technique for doing this is inspired by ant based colonies. The use of such techniques was originally proposed by Dorigo et al in [24] to solve

[4]where the static evaluation is unreliable

combinatorial search problems. However they have since been applied to many areas relevant to distributed, continuous planning such as continuous control of network traffic routing. This form of adaptive control system can be very useful in highly dynamic environments.

3.3 Techniques for Partner Search and Selection

3.3.1 Matchmaking

The basic assumption made in viewing partner search and selection as a stand alone exercise is that it is possible to decide on the set of roles to fill independently of assigning suppliers to them. Many approaches to the problem of finding suppliers for roles have been proposed.

One commonly used approach is known as matchmaking. The basic principle behind matchmaking is that the capabilities of the candidates are defined using formally structured external adverts which can then be matched against the structures used to define who can fill the role. The language used to represent both the capabilities and the roles which must be filled form a crucial aspect of any matchmaking approach.

Within web service composition the combination of services produced must be automatically executable. This problem is addressed through the use of specific service description languages which often contain several levels - the top level indicates what the service does while the lower levels indicate how they do it. For example the OWL-S service description language, described in [25], has the following models: the service profile which indicates 'what the service does' in terms of input/output transformations and also addresses such matters as quality of service information, a service model which details how to use the service in terms of message based access on a conceptual level and finally a grounding which translates the service model into concrete details of how to access the service.

As might be imagined the task of matching a service description to a requirement is quite complex, especially since as noted in [52], the match is rarely exact, so that you must also decide what level of match is acceptable. In OWL-S, and related service description languages, the task of assessing the level of fit is assisted by the usage of a ontology language to describe the types of inputs and outputs. This allows for accurate sub/superclass comparisons between the datatypes used within the input and output. In the case of OWL-S the W3C standard ontology language OWL is used. An overview of this language is given in [54].

An additional service description language is the WSMO project, whose home page can be found at [87]. This project is developing a web service description language together with the necessary software infrastructure to allow it to be used. WSMO takes a very similar overall approach OWL-S, with only certain technical details differentiating the two languages.

Once a service description language is chosen the next problem is to define an algorithm for comparing the service descriptions with the requirements generated. This algorithm must be able to compare data types which are not exactly equal, and to combine these measures into a single measure of whether the descriptions match. This remains a very active and unclear area of research. A brief overview of some of the higher level technical challenges faced within this area is given in [79]. Since the comparison of the lower levels of the service descriptions is designed solely to enable the subsequent automatic enactment of the combined service produced, something which is not important for this thesis, these aspects - also very active areas of research - are not further considered here.

3.3.2 Contract Net

The matchmaking approach discussed in the previous section is dependent on representing the capabilities provided by individual companies in terms of adverts. Section 2.4.1 however demonstrated several problems with applying such an approach to the chosen problem domain of this thesis.

An alternative approach is offered by the Contract Net protocol. Contract net, originally introduced in [75], is typically used when a buyer agent must select an agent to provide a given item. The buyer initiates the protocol by sending a call for quotes to other agents who it thinks might be willing to provide the item. If interested each of these agents then replies to this message stating at what price it is prepared to offer the item. Finally the buyer agent chooses its preferred offer.

When this process is adapted to use several rounds of calls for quotes it is termed the iterated contract net protocol. As well as locating suppliers for individual products, the same algorithm can be used to find agents willing to provide a service - in such circumstances the bids returned do not need to solely focus on price but can also include quality of service aspects.

Contract net differs from matchmaking approaches in that it transfers the main burden of reasoning from the centralised system to the candidates - it forces the candidates to decide whether they are able to fill a role, whether they wish to fill that role and finally how they propose to fill that role. This approach corresponds well with the domain features discussed in the previous chapter. In particular it allows the system to reflect the way that companies intelligently vary the levels of cooperation they offer to projects. Contract net however remains dependent on using centralised knowledge to transform the initial problem specification into a set of tasks described in sufficient detail to allow candidate suppliers to be directly located for each task.

3.4 Techniques for VO Composition and Negotiation, Detailed VO Planning and Contracting

The discussion in the previous chapter showed that, within the chosen domain, the provision of full automatic support for these stages of VO formation is a task of considerable complexity. This section focuses on two specific areas of particular interest - equitable sharing of the utility generated by fulfilling an awarded contract and computer based support for the process of collaborative design.

3.4.1 Utility Sharing

When forming a VO it is important to decide how the profit generated by the VO gaining the contract should be shared out between its member companies. When a central company has driven the process of team formation it will normally take responsibility for assigning a cost to each part and the companies assigned to the part must bid low enough to ensure that they remain in profit. However, when a set of companies have cooperatively formed a team, no company has the authority required to dictate the level of profits which each team member should enjoy and more cooperative techniques are required. This problem of fair division has been studied in several places including [9] in which Brams and Talyor study numerous algorithms to support it.

The primary focus of this work is ensuring the fair division of sets of concrete products when the people between whom the products are to be shared value them differently. In this case the usual focus is on envy free divisions - divisions in which no person would prefer to have anyone else's collection over their own. The basic problem studied is taken to be that of dividing a cake between two or more people - in the case of two people the solution is to have one person split the cake and the other person choose which slice they prefer.[5]

While the two person case is quite simple the problem of ensuring an envy division between n people is much more complicated. An algorithm which does this using a minimal number of cuts is presented in [48]. Much of the complexity involved in this domain is caused by the way that distinct people have different valuations of different objects. In the current context the main focus is on the rather simpler problem of sharing out money. Never the less other factors than straight profit such

[5]A basic assumption behind such algorithms is that personal evaluations of worth remain consistent over time.

as importance within the project, security of contract etc should be considered and these cake cutting algorithms may provide useful input for possible solutions.

3.4.2 Collaborative Design

The process of collaborative design, discussed in Section 2.2, is both exceedingly complex and time consuming. It is thus natural to examine a few systems supporting this process. One approach is suggested in [70], which uses a case study from the train manufacturing domain. The main focus of the engineers within the project studied is reported as being on the values of certain parameters and in particular on those parameters which have to be kept within a certain range to avoid a major redesign of the system. These are known as killer parameters, and there are about 300 within the typical project discussed in the paper.

The collaboration between the companies in the case study took place mainly through the communication of parameter values. This suggests investigating how such parameter exchange could be used to support collaborative design. Indeed parameter exchange forms the basis of the approach proposed by Lander in [49]. Lander's main scenario concerns the configuration of multiple components of a steam pump system to produce steam pumps to a given specification.

Every component within the system is modelled as a concepts with a set of valued attributes and given an agent to represent them within the system. Interestingly a distinction is made between those attributes of the pump which are only relevant to the pump agent and those which are relevant to the other agents in the system - each agent only sees the attributes which affect its operations.

The system then operates by negotiation - one of the component agents initiates a solution by proposing a set of values for their parameters and the other agents can then extend the solution if it is compatible with their requirements or critique the solution by mentioning which of their constraints the proposed solution violated. Constraints are given levels of importance from 0 to 4 with 0 representing no possibility of relaxation and 4 meaning the agent is quite happy to relax the constraint if required.

If a hard constraint is violated then an agent can mark the proposed solution as infeasible indicating that it cannot be usefully extended. If a relaxable constraint is reported as violated then the system stores the solution. The system works by extending solutions until either a complete solution is formed or all the currently ongoing solutions are blocked due to constraint violations. The agents also learn when they violate other agents constraints and can later avoid suggesting such solutions.

If no completely acceptable solution can be found then the agents start relaxing their constraints rendering some of the previously unacceptable solutions acceptable and then start extending those solutions. A fully worked through example can be found within the paper. While the problem studied within this paper is simpler than many real world collaborative design problems the approach seems promising, especially for providing high level support for later detailed negotiation between designers. Indeed this paper provided inspiration for the design of certain elements of the main system proposed within this thesis.

A related approach is proposed by Tan et al in [82]. Tan's approach aims to ensure that the initial designs produced are both feasible and manufacturable. This is facilitated by the use of a blackboard which stores design information and a set of agents representing the different phases of the development and production process. This includes agents who evaluate the manufacturability of a design.

The system then proposes an initial design which is critiqued for, in turn, feasibility in a design sense, the feasibility of the design in production terms and finally the overall cost of producing the product (including manufacturing overheads). These suggestions are made linearly but the blackboard architecture allows for their interactions to be considered. The size of the search space involved means that creating a fully detailed design using this technique is likely to prove highly problematic. The system however has promise for initial exploration of the set of potential designs - perhaps in conjunction with the idea of set based design discussed in Section 2.2 above.

It is also possible to view collaborative design as a simple search problem, albeit one over an enormously large search space. This approach is taken in [11], which focuses on car body design. This involves searching over a space defined by three dimensional shapes with a very large number of parameters which the paper attempts to tame through the use of heuristics generated from design knowledge from the engineers. This system potentially offers the possibility to optimise results. However the difficulties encountered by human designers indicate that the real test will lie in whether the heuristic based search can usefully cope with the size of the search space - at the time of writing the paper no tests had been run. A final interesting paper which supports collaborative design, but also includes agents which perform some reasoning aiding the design process - for instance contacting the collaborators affected when a given parameter changes its value - is given in [51]. Such collaborative design support systems, while both highly useful and well studied, will not form a major focus for this thesis and will not be further discussed here.

Finally the most common method proposed to support collaborative design is the use of groupware systems which offer support for a group of engineers working in a collaborative manner. A substantial survey of the problems within this domain is given in [13].

3.5 Summary

This chapter has examined the theoretical background to the problem of supporting VO formation within the chosen application domain of this thesis. In combination with the discussion contained within Chapter 2, it concludes that the particular demands of supporting VO formation within the chosen domain might be best addressed by a system which:

- supports the formation of multiple teams rather than focusing on producing a single 'best' team;
- allows the VBE members to evaluate the attractiveness of an opportunity and then to decide which processes they wish to offer;
- allows the VBE members to control which companies they might end up working with in a VO;
- distributes the process of decomposing the original opportunity, thus permitting innovative solution paths to be investigated.

No existing system supports these features and so a novel system must be developed. A major theme identified is that this system should be directly controlled by the VBE members. To do this requires that the components representing the companies in the system can reason, negotiate amongst themselves and behave proactively. These factors suggest that the companies should be represented within the VO formation support system by software agents with knowledge of both 'their' company's processes and the conditions under which the company would prefer to propose them.

Indeed a system within which such agents were allowed to negotiate with each other would in principle suffice for VO formation, and meet the requirements above. Such a system however fails on the crucial point of *supporting* team formation rather than just 'hoping' that it will happen spontaneously. Where VO formation happens in a VBE small enough for each company to have a good idea of the capabilities of the other members this approach could be effective. However within the larger groups typically found in the current context there are several problems:

- there is no way for individual agents to proactively detect when some other agents have formed a team capable of partially meeting the required goals;
- there is no systematic way for the agents to share decomposition information between themselves.

Thus we are looking for a way to support negotiation driven VO formation amongst independent agents by addressing these problems. Since there is a one to one correspondence between the companies within the VBE and their agent representatives, the agents themselves are henceforth referred to as VBE members.

Chapter 4

Virtual Organisation Formation: An Approach

The previous chapters have highlighted that current approaches supporting VO formation have limited applicability within the problem domain of this thesis. This motivates the development of a novel approach designed to address the domain's specific requirements. The remainder of this thesis is devoted to the development and evaluation of such an approach.

The current chapter introduces this approach. The first section precisely defines which phases of VO formation are supported. The second section provides a summary of the motivations behind the chosen approach. Finally, the remainder of the chapter provides a conceptual introduction to the approach itself.

4.1 Scope of Proposed Approach

The following stages of VO formation, discussed earlier, are taken as a starting point:

1. Collaboration Opportunity Identification;
2. Draft VO Planning;
3. Partner search and selection;
4. VO Composition and Negotiation;
5. Detailed VO Planning;
6. Contracting;
7. VO Set up.

The proposed approach fully supports draft VO planning, partner search and selection and VO composition and negotiation. For reasons discussed in section 2.3.1 no specific support is offered for collaboration opportunity identification.

The discussion within Chapter 2 showed that the final three stages of detailed VO planning, contracting and VO set up to be particularly involved in the current domain. Indeed, as discussed in Section 2.2, the single element of concurrent engineering is enormously complex and typically takes many months or even years to complete.

This complexity makes it practically impossible to fully cooptimise these three stages during the production of an initial virtual organisation. These matters do however directly affect the worth of a candidate VO. The proposed approach considers the likely outcome of these phases during team formation but offers no direct support for them.

The scope of the approach is summarised by the following:

> To take a business opportunity and create certain selections of companies from within a VBE which can *probably* be expanded into VOs addressing the opportunity.

4.2 Outline of Proposed Approach

The basic principle behind the proposed approach is that VO formation should be driven by supported negotiation between intelligent agents. In order to construct such a system it is helpful to examine where a system based on unsupported negotiation between intelligent agents fails. Such a system encounters two major problems, the net effect of which is to make it impossible for agents in the system to know when they can usefully contribute:

- there is no systematic way for agents to discover groups of other agents which have formed;
- there is no systematic way for the agents to share decomposition information amongst themselves.

The first of these problems can be addressed by providing a centralised information repository to which partially formed groups of agents can register, allowing other agents to discover their existence. Such a system, whereby planning is driven by a set of independent agents using a centralised information repository brings to mind the idea of blackboard systems previously discussed in Section 3.2.3. In this case, when compared to traditional blackboard systems, the knowledge sources have been replaced by independent agents who have taken over responsibility for scheduling from the blackboard itself. As previously discussed, it is perhaps clearer if such systems, several examples of which were discussed in Section 3.2.3, are not called blackboard systems and instead the term *noticeboard* system will be used.

As well as differing in their possession of a distributed control mechanism, noticeboard systems differ from blackboard systems in that the agents within the system do not have to use the blackboard to communicate with each other and can instead negotiate directly. Indeed the noticeboard is only used for communicating information of potential global interest. In the case of VO formation within a VBE there are two primary contexts requiring such local negotiations - one is when a company wishes to join an existing, partially formed team and another is when a set of companies potentially capable of meeting the opportunity has been located and must decide exactly how they plan to work together.

The most important use of negotiation in the system is a protocol by which the existing members of a partially formed VO can refuse the request of another company to extend it. This ensures that partially formed teams represent a set of companies happy to work with each other. In addition it provides a form of distributed solution space control. This protocol, and the potential decision processes used by the agents within it, form a crucial part of the proposed system and will be discussed in some detail.

The use of agents within the system offers potential support for other forms of negotiation. The most important of these are the negotiations triggered when a potentially complete set of companies. As outlined in Section 4.1 these are considerd to fall outside the main scope of the proposed system. Additionally the system offers potential scope for 'unscripted' negotiations between agents. For instance one company may wish to contact another to gain more information regarding an extension it has proposed. While such negotiations could be useful and are facilitated by the system they are not intrisic to the efficient operation of the system. In addition they are context specific and would require complex reasoning.

Since the time scale in which VO formation occurs permits time for these negotiations to be conducted by human representatives, it is therefore anticipated that these additional forms of negotiation will typically be conducted in this manner. These negotiations are therefore not treated in detail in the remainder of this thesis. [1]

The second requirement remains - a systematic way for the agents to share decomposition information must be provided. Here the proposed approach adopts the same approach as blackboard systems, discussed within Section 3.2.3. Such approaches facilitate the sharing of knowledge by storing the partial solutions generated in a

[1]Where such 'unscripted' negotiations are implicitly mentioned in the Case Study of section 7.3 they should be considered as being conducted by human representatives of the companies.

centralised data repository, here a noticeboard. This allows the knowledge sources, within this approach agents, to examine this information and deduce where they might most usefully contribute.

Using independent agents instead of knowledge sources means that, in contrast to blackboard systems which have a single main blackboard, the system can naturally contain multiple noticeboards. The approach proposed within this thesis uses a system whereby each individual noticeboard is responsible for holding both a detailed definition of, and the details of, the partially formed teams contributed towards a single business opportunity.

In summary the proposed approach to VO formation within a VBE uses the following elements:

- one agent for every company present within the VBE;
- noticeboards, each representing a given business opportunity.

On an abstract level the process of VO formation then typically follows the following sequence of steps:

1. a business opportunity is identified;
2. a noticeboard is created to catalyse VOs to form to meet that opportunity;
3. the member agents of the VBE observe the partially formed VOs on the noticeboard and offer to extend any interesting opportunities they find;
4. each proposed extension triggers a negotiation between the existing members in the partially formed VO, as to whether or not they accept it;
5. any potentially complete VOs are notified and decide amongst themselves both if they actually wish to place a bid for the opportunity, and the details of any such bid.

This process then continues, with complete VOs being generated and allowed to bid, until some natural end point such as the deadline for bidding for the contract is reached. The implementation of such a system would meet the goal of supporting decentralised VO formation within a VBE.

4.3 Example of solution development

Figures 4.1 to 4.5 show an example of how the system might generate a complete combined capability. The first figure depicts a newly created noticeboard, and later figures depict the stages of solution development. For clarity only the creation of one, simple, complete solution is shown and a simplified, linear representation of the combined capabilities used. In the general case multiple solutions of differing forms will be explored.

As an aid when considering these pictures consider the opportunity to require the production of a water tank. Figure 4.1 shows the new noticeboard together with a set of associated capability provider agents. In this diagram I and G respectively indicate the resources in the initial and goal states. The utility information on the noticeboard is not given in full for clarity. The goal state is assumed to contain only one resource, a water tank.

The first step in the scenario then occurs when the capability provider agent A observes the noticeboard and recognises that it involves producing a product type in which its company has some expertise. On closer examination they decide that they can contribute a declared capability saying that they could produce the water tank if they were provided with a specifically configured reservoir.

Since there are no agents already present in the combined capability this extension is automatically approved, and the additional combined capability produced can be seen in Figure 4.2. In this figure the new resource type M represents the reservoir type that agent A requires in order to produce the water tank.

The next step in the scenario is that another capability agent, agent B who represents a company who specialises in producing reservoirs, notices the opportunity represented by the new combined capability and offers to further extend it. This is seen in Figure 4.3.

Since agent A is already present within the combined capability to be extended the noticeboard triggers a protocol whereby agent A can choose to veto the proposed extension. They might do this if, for instance, they do not wish to work with agent B. Agent A does not know much about agent B and so a period of negotiation between the agents ensues. These conclusions complete to agent A satisfaction and the extension request is granted. This process can be seen in Figure 4.4.

The result of agent B's extension being accepted, shown in Figure 4.5, is a complete solution. The noticeboard agent then detects this solution and informs the two companies concerned about it. These companies then know that they form part of a team which might be able to bid for the contract. Before doing so they must however negotiate further details, and this negotiation is also shown in Figure 4.5. If these negotiations were to complete successfully then the companies represented by agents A and B would then form a virtual organisation and tender for the business opportunity.

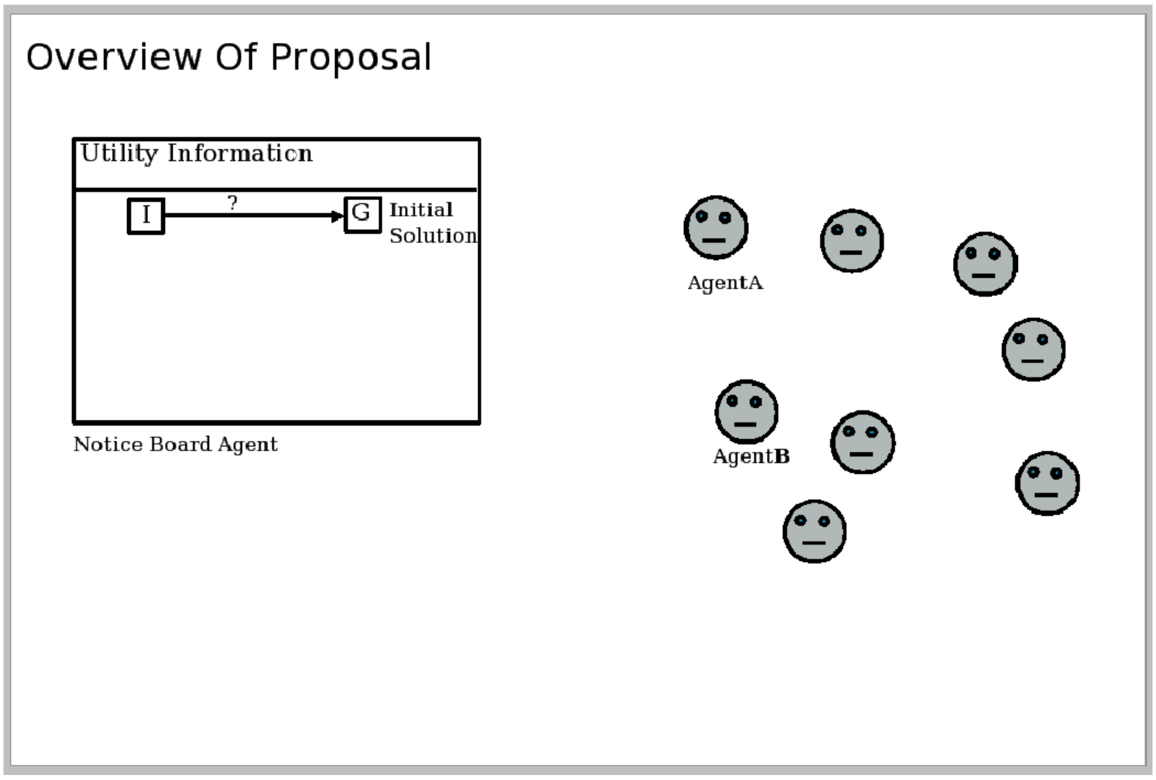


Figure 4.1: The opportunity is noticed and a noticeboard is created

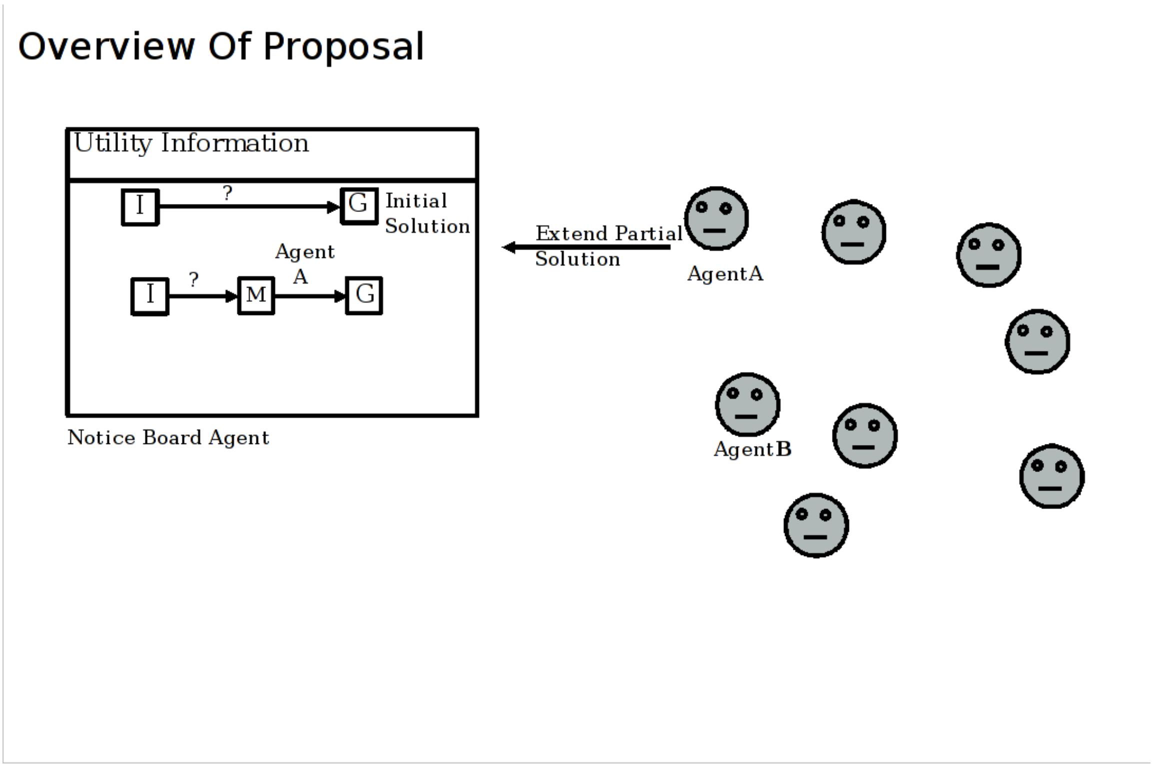
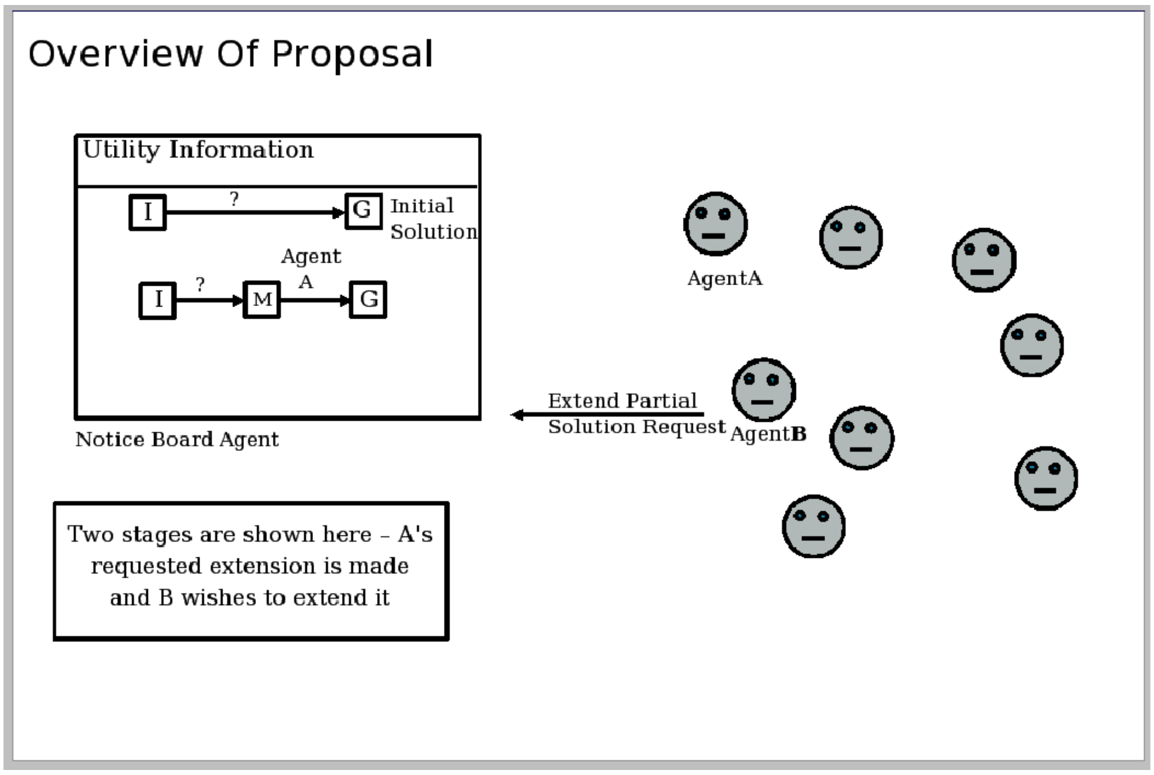


Figure 4.2: Agent A extends the initial solution



Two stages are shown here – A's requested extension is made and B wishes to extend it

Figure 4.3: Agent B wishes to further extend the extended solution

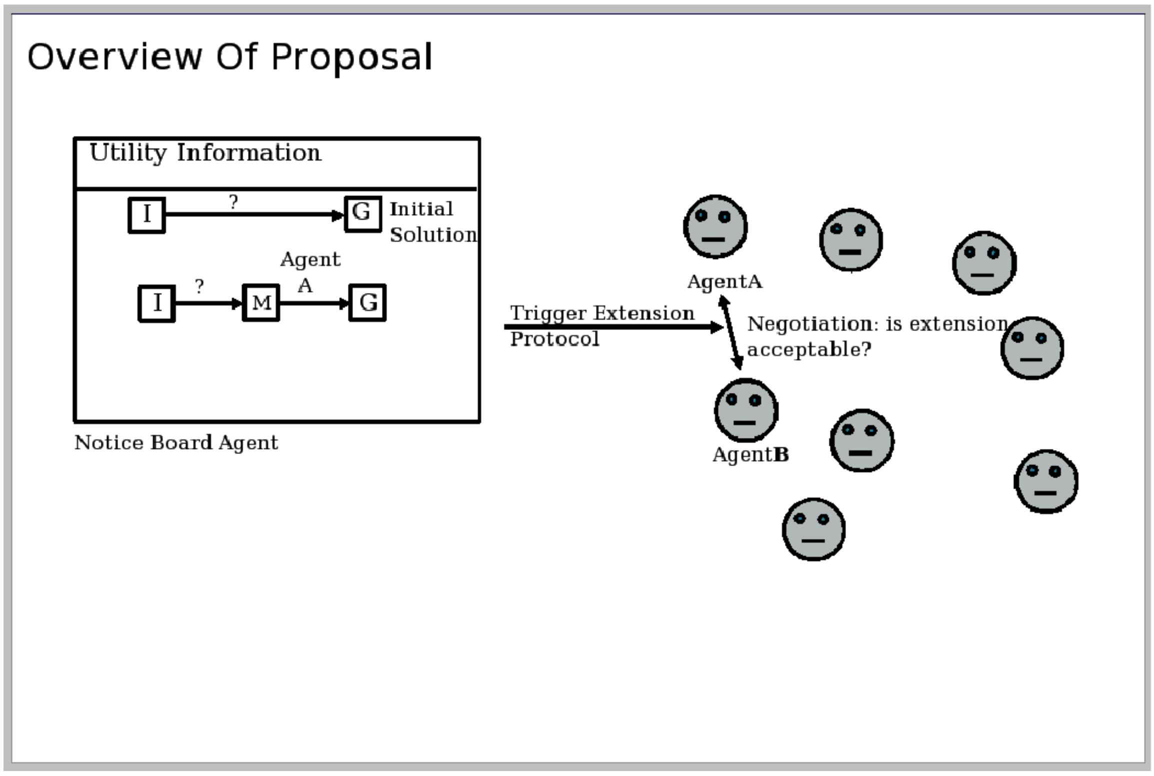


Figure 4.4: The proposed extension is verified

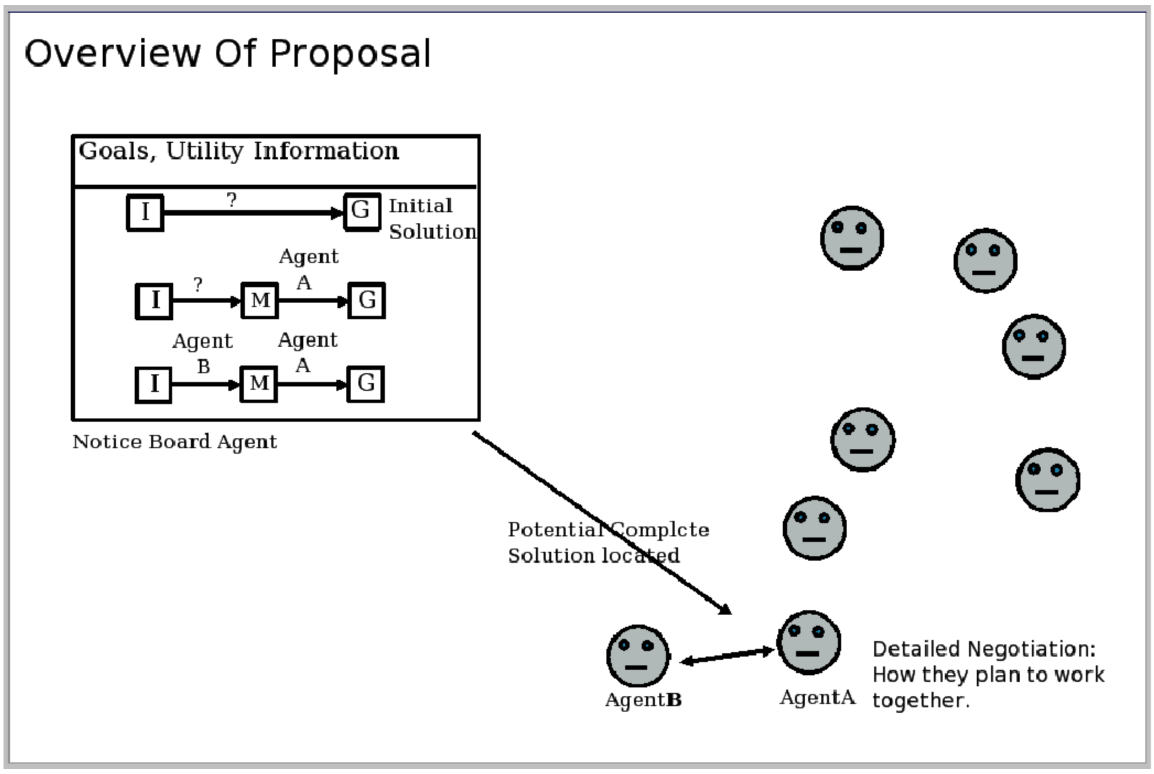

Overview Of Proposal
Goals, Utility Information
I ? G Initial Solution
I ? M Agent A G
I Agent B M Agent A G
Notice Board Agent
Potential Complcte Solution located
AgentB
AgentA
Detailed Negotiation: How they plan to work together.

Figure 4.5: A complete solution has been located

4.4 Process Representation

Any concrete system developed from the, fully general, abstract approach so far presented will be tied more closely to a specific application domain. The concrete details developed in this section and the next few chapters develop the approach towards the chosen problem domain. In particular, the formalism used to represent the processes of the companies is closely tailored to manufacturing processes.

An especially important decision is which forms of data representation should be used. The following elements are of particular importance:

- the individual contributions of each company;
- the partially formed teams.

As with all blackboard systems, these elements provide direct support for the reasoning required within the system. In the current approach the following types of reasoning must be supported:

- the agents within the VBE must be able to examine a partially formed team and deduce what processes might help to complete it - the 'gaps' must be visible;

- given a partially formed team, an individual contribution and a suggestion of where this contribution will fit it must be possible to generate a new partially formed team reflecting the additional contribution;
- it must be possible to detect when a partially formed team is 'complete'.

The second of these requirements permits the information within partially formed teams to accumulate, starting from an (typically) 'empty' initial set of companies and finishing with a complete team. In the current context a complete partially formed team is one containing a set of companies who in principle could form a VO to bid for the business opportunity without using any additional companies.

4.4.1 Individual Processes

Before presenting any concrete design of the data types it is vital to consider exactly what information they should contain. The minimal requirement on the representation used for individual companies processes is that it should express the intended effects of the process. As seen in Section 3.2.4, systems often also consider *how* the company intends to achieve this. In particular web service representation such as OWL-S, [25], contain considerable detail relating to the execution of webservices. Such workflow based formalisms are also often encountered when considering how to merge the 'real world' processes of companies.

If one intends the combined process generated by the system to be automatically executable, as for instance web services do, then such matters must be considered. In contrast, in the current context the VOs generated cannot immediately start work - they must first tender for the contract. Even if they are awarded the contract the initial design phase takes a considerable length of time. Thus there is no need to generate immediately executable teams and the processes can be kept abstract thus allowing flexible reasoning on the parts of the agents.

In fact the process representation used within the system contains only the following information:

- which company is offering the process;
- the observable effects of that process.

Companies are represented by name and perhaps contact details. The actual effects of the process are represented as 'black box' machines - a set of resources goes in and a different set of resources comes out. Such a representation is well suited to the representation of manufacturing processes. While this representation differs from the commonly used planning paradigm of process altering the state of the world, it is close to that found within [22] and previously discussed in Section 3.2.4.

The proposed representation uses an innovative approach in its detailed treatment of the resource representation. In order to reflect the way that machines can deal with resources within a certain tolerance levels - for instance a woodcutting machine might be able to deal with planks of wood whose thickness, length and height fell within certain ranges - each resource is represented by a combination of a type and an indication of the acceptable range for values of that type's attributes.

This form of process representation has several useful side effects. One is that 'processes' defined in this manner can in fact represent a combination of multiple processes within the actual company. There is no requirement that the company has *any* specific concrete process in mind - merely that they consider it likely that they could consume/produce resources with the indicated dimensions. For example a company who could only process certain types of metal sheeting with their current machines might volunteer a 'process' claiming that they could process any type of metal sheet. In this case their intention would be to consider purchasing new equipment if they were required to process a type of metal sheeting beyond their current capabilities.

An additional effect of the abstract nature of the external process representation used is that companies do not need to reveal any of the internal details of their processes. This is useful since, even when such detail exists, the level of trust between the members of the VBE can vary considerably making the companies reluctant to share such information.

In fact calling the things represented in this way 'processes' is rather misleading and they are instead termed *declared capabilities*. Since the partially formed VOs represent collections of these capabilities they are consequently called *combined capabilities*.

4.4.2 Combined Processes

The decision to use sets of input and output resources to describe declared capabilities dictates that a similar technique should be used to describe combined capabilities. Indeed the idea of sets of resources being consumed/produced is central to the chosen representation. The term *state* is used for such a collection of resources.[2]. A combined capability contains the following elements:

- an initial state - representing the set of resources provided for 'free' to any consortium wishing to meet the opportunity;

[2]Chapter 5 formally defines the terms introduced in this section

- a goal state - this specifies which resources must ultimately be produced;
- a set of intermediate states;
- a set of linkages between these states - each of which states that a given company will use certain specified resources from the chosen state to produce certain other specified resources.

The above presentation does not mention the 'gaps' within the combined capability, *ie* those places where agents could usefully contribute new declared capabilities. These are generated from a combination of those resources which a declared capability has provided but no agent has yet consumed and those resources which a declared capability needs as input but have yet to be provided.

Since the states and linkages represent things produced/consumed by processes the states must be ordered to reflect this - resources cannot be consumed before they are produced. A partial ordering between the states most closely represents reality and a data structure implementing this is presented in the next chapter.

The final important matter here relates to the extension of combined capabilities with new declared capabilities. Solution development within a noticeboard system can occur in several ways:

- backwards chaining - here a company realises that it could produce a large component if supplied with certain subcomponents;
- forwards chaining - here a company realises that it can supply a subcomponent which will typically be required to produce the final system and suggests it;
- combinations of the two approaches.

Examples of both of these forms of solution development are presented within the case study of Section 7.3.

4.5 Areas of Interest

The current chapter has presented an approach to supporting VO formation within the chosen domain. In order to develop this approach into an actual system certain elements must be further defined:

- The datatypes used within the system, and the algorithm used for combining them must be formally defined.

- The noticeboards must store the non process based information defining the business opportunity - for instance which company created the business opportunity. This information is vital in enabling the agents in the VBE to decide the level of cooperation they wish to give towards the opportunity.
- The consequences of distributing control in the system to agents must be discussed.
- The process providing agents must be defined, including those aspects of their behaviour relating to representing real world companies.

An example of this is the ability of agents to veto proposed extensions to combined capabilities of which they are a member. When will companies do this? What consequences does this have for the overall efficiency of the system? Could it potentially prevent the system working under some circumstances?

The following two chapters develop these aspects. Chapter 5 focuses on presenting the theoretical groundwork and data structures required by the system. It presents a detailed definition for declared capabilities, combined capabilities and finishes by specifying an algorithm for developing initial capabilities into complete capabilities. This algorithm is presented in the manner of a centralised planner - the effects of the distributed control of the system by agents representing companies are considered in Chapter 6.[3]

Chapter 6 then considers the effects of the system being controlled by agents representing companies. It thus discusses the consequences of matters such as the ability of agents to veto proposed extensions. The reasoning requirement placed on the agents representing the companies is also examined, and an abstract framework for how such agents might best be designed presented. The chapter concludes by considering several matters occurring as side effects of the distributed control. When reading these two chapters it is not necessary to understand the full technical detail of the first chapter before reading the second.[4] Indeed on a first reading it may prove useful to initially understand the basic datatypes and algorithm flow introduced in Chapter 6 before moving onto Chapter 5.

The abstract design for the system developed in the current chapter will be taken as the basis for the development of a concrete system within chapter 6. Chapter 6 is thus mainly concerned with deciding the exact responsibilities of the agents within the system and the details of how the algorithms can be implemented.

[3]They were of course considered during the design of the algorithm

[4]The technical details of chapter 5 correspond to a single step in the overall system

Chapter 5

Technical Details

The current chapter presents a detailed, formal definition of the data types and algorithms introduced within chapter 4. Thus it presents two of the technical contributions of this thesis - the knowledge structures representing the capabilities of individual companies and a method for combining them. The final technical contribution, an agent system implementing the approach of the main contribution, is presented in detail in chapter 6. In detail chapter 5 addresses the following aspects:

- how declared capabilities are represented;
- how combined capabilities are represented;
- how to generate the potential extensions of a combined capability;
- how potential extensions and declared capacities can be compared;
- how a combined capability can be extended by a given declared capability which has been matched to a given potential extension;
- how complete combined capabilities can be identified.

The chapter starts by defining the basic building block of the resource representation - that of a resource. The definition of declared capabilities as consuming certain resources to produce other resources is then developed, followed by the combined capability representation and algorithms for adding a declared capability to a combined capability. The chapter finishes by presenting an example illustrating how a set of declared capabilities can be combined to generate certain combined capabilities.

5.1 Representing Companies Contributions

5.1.1 The Resource Representation

The basis for the process representation is the idea of processes consuming/producing certain classes of resource - for instance a process producing wood planks might produces planks with lengths between 10 and 12 metres, widths between 1 and 2 metres and depth 10 centimetres. The following section precisely defines the method used to represent these resources within the system.

Crucial to this is ensuring that the agents within the system can effectively communicate about the basic types of resource involved. The use of an ontology, previously discussed in Section 3.2.3, represents a proven, effective solution to this and is used within the system.

The question of how to represent the way that the processes can consume/produce a 'range' from within these types must still be addressed. One possible solution is to represent every possible range of parameters and material type as a separate class within the ontology. For instance the wooden plank above might be represented as a class named 'large-wooden-plank'. This representation has significant drawbacks. The first is that every time a company changes the parameters on its machines, a new class is required within the main ontology. The ontology will thus become very large, increasing the effort that a company must expend on joining the VBE. Finally such a representation means that comparing two such classes would require the full ontology.

An alternative approach, adopted here, instead defines only the basic material types, their relationships and their attributes within the ontology. The constraints on the attributes of the basic material types are explicitly communicated at the same as the basic material type. This representation avoids the need to expand the new ontology when new configurations are detected and permits easy comparisons between resources.

The following example compares how these two techniques would represent the input resource type of a process which can roll steel girders with lengths between 2 and 3 metres. If everything is representing in the centralised ontology then that ontology would contain the following classes (defined informally):

Class SteelGirder, Attributes = (Length,Width)

Class LimitedLengthSteelGirder subclass of steel girder, Restrictions = (ValueOf(Length) between 2 and 3)

Within messages individuals of the type LimitedLengthSteelGirder would then be used. If another agents process could produce steel girders of length between 1.5 and 2.5 metres then a class of the form:

> Class AnotherLimitedLengthSteelGirder subclass of steel girder, Restrictions = ValueOf(Length) between 1.5 and 2.5

would have to be present in the ontology.

In comparison when using explicit constraints the ontology would only contain the base class:

> Class SteelGirder, Attributes = Length,Width

and within messages the following would be sent:

> SteelGirder (Length: ValueFromRange(2,3); Width: anyValue)
>
> SteelGirder (Length: ValueFromRange(1.5,2.5); Width: anyValue)

Measured purely in terms of their expressive power there is no difference between these two approaches to representing the data. They differ in where the information is stored - the second one makes more information explicit within the messages sent while the first stores all of its information within the centralised ontology.

5.1.2 Constrained Resource Types

The following section presents a formal treatment of the approach introduced above. The resource representation comprises of two basic, largely independent elements:

- data structures representing the basic material/component types present within the system;
- structures to represent the constraints within the attributes of the structures.

The representation chosen is largely independent of the precise language used to represent the material/component types. Informally it requires that the type system should contain connected types which have attributes attached to them. More formally the type system is required to have the following attributes:

1. it must contain a set of types, $T = (t_i)$;
2. for each $t_i, t_j \in T, t_i \neq t_j$ it must be possible to decide if t_i is a subtype, a supertype or a type incompatible with t_j;

3. every $t_i \in T$ must have an associated set of attributes, Att_i where each Att_i is an injective function, $Att_i : T \ni t_i \rightarrow \wp(T \cup BasicTypes)$. Here BasicTypes represents a set containing a few primitive data types forming the basis of the type system. At a minimum BasicTypes is required to contain both integers and strings. It can however optionally contain certain other primitive types.

Within the following discussion $Att_i(C)$ will be used to denote the value that the attribute Att_i takes on the individual element C. Any frame based approach to ontology construction will meet these requirements as would any of the ontology languages discussed in Section 3.2.3. Of these OWL, [54], is a commonly used standard. The remaining element of the representation is the constraints which go within the ontology:

Definition 1 *An* explicit constraint *(henceforth simply constraint) within a type system T is an identified member of the type system, C, such that $\forall t_i \in T, \forall Att_i$ such that Att_i is an attribute of t_i, every individual of type C is a member of the domain of Att_i*

The above definition does not attempt to define what a constraint is or how one might compare them. The next idea is that of a constrained resource type:

Definition 2 *A* constrained resource type*, $\mathcal{T}$, consists of an individual*[1]*, t_i of type, T, where $\forall Att_i$ of T, $Att_i(t_i) = c$ where c is a constraint*

It may initially seem odd to call an individual a type, but this is the logical consequence of the chosen representation. Within this thesis two main constraint types are used: FromRange and ValueFrom. Conceptually these constraints indicate that an attribute's values should, respectively, either fall within some numerical range or come from a specifically enumerated set. Formally, denoting by $Att(C)$ the value that the attribute, Att takes on C:

Definition 3 *A FromRange constraint, FR, has two attributes - R_{low} and R_{high} - each of which takes a single real number with,$R_{low}(FR) \leq R_{high}(FR)$. A ValueFrom constraint, VF, has a single attribute - R_{values} - which contains a set of resources of any type.*

The requirement that every attribute must contain a unique constraint causes an interesting problem when you do not wish to constrain an attribute's value. To permit this a special constraint termed the *unconstraint* is used. This constraint is defined as allowing any value.

[1]An instance in object orientated terminology

For convenience when representing constrained resources in the future, those attributes containing the unconstraint will be left blank and those restricted to have a unique value will simply have that value - rather than a ValueFrom constraint containing a single member - inserted. Additionally the names of the attributes of the constraints are not shown. An example follows:

$Box(: HeightFromRange(5, 10) : WidthFromRange(6, 9) : DepthFromRange(4, 8) : MaterialSteel)$.

5.1.3 Constrained Resource Sets and States

Since a declared capability might consume and/or produce multiple types of resources, groupings of constrained resource types must be considered.

Definition 4 *A* constrained resource set*,$\mathcal{RS}$, consists of a natural number, Q, and a constrained resource type $\mathcal{T}$.*

In this definition Q denotes the quantity of the constrained resource type present within the set. This is not used to represent *absolute* quantities of resources but rather to specify the *relative* amount of resources required within a process. The use of this is illustrated in the example below.

Definition 5 *A* state *is a collection $\mathcal{C}$of constrained resource sets*

These states represent the basic building blocks for a process representation - essentially each state represents either the input or output of a manufacturing process. For instance a process which takes four metal bars and a window pane as input and produces a window might be represented as:

[Input =

:ResourceSet

(:Quantity (4) :Type (metal bar

(:length (FromRange (1,2)) :height (FromRange (8,9)) :depth (1)))),

:ResourceSet

(:Quantity (1) :Type (window pane (:area FromRange(60,80)))) ,

Output =

:ResourceSet (:Quantity (1) :Type (Window (:area FromRange(60,80))))]

Within the above representation the brackets split the various classes while the attributes and type names are preceded by :'s and the attribute values enclosed in brackets. For instance :Chair (:Legs (4) :Material (Oak)) would represent a oak chair with 4 legs.

The above exemplifies the use of quantities - the quantity within the final product is kept as small as possible (preferably at one) while the other quantities indicate the amount of that resource required to produce that amount of final product. Since the quantities should ideally remain integral the output quantity can end up larger than one when several processes are combined. Intuitively the quantities give an idea of how much of each subcomponent is required to make a single finished product, thus aiding later consideration of logistical matters.

5.1.4 Comparing Constrained Resources and States

Intuitively to compare resource sets you simply compare the resources within their states. In a formal sense it is more involved. The following definition will be useful in the following discussion.

Definition 6 *The abstract resource type corresponding to a given constrained resource type, t_i, is its base type, T.*

In order to compare states one must be able to compare their component constrained resource types. For the sake of clarity only the following relationships between two abstract resource types, T and T', are considered possible:[2]

1. T is the same type as T';
2. T is a sub/supertype of T';
3. there is no relationship between T and T' within the type hierarchy.

The other element within a comparison of constrained resource types is the constraints within their attributes. Two constraints C,C' whose permitted values are written P(C) and P(C') respectively are related in the following ways:

1. if P(C) = P(C') then C is equivalent to C';
2. if $P(C) \subset P(C')$ then C is less general than C' and C' more general than C;
3. if all of $P(C) \cap P(C'), P(C) \setminus P(C')$ and $P(C) \setminus P(C')$[3] are non empty then C and C' are comparable;

[2]Some ontology languages such as OWL permit partial overlaps between classes

[3]Here $\setminus$ represents the set difference operator, ie the set of all objects which are members of the first set which are not also members of the second

4. if $P(C) \cap P(C')$ is empty then they are incomparable.

The range of permissible values for the types of constraint currently under consideration are:

- for a FromRange constraint, the (inclusive) interval of real numbers, (R_{low}, R_{high});
- for a ValueFrom constraint it is the members of the countable set R_{values};
- for the unconstraint it is every individual allowed by the type system.

An immediate corollary of the above is that the unconstraint is more general than any other constraint except for another unconstraint to which it is equal. Assuming that two constraints concern the same attribute the following rules for comparison can be deduced:

1. The comparison between two FromRange constraints or two ValueFrom constraints is governed by a comparison between the elements within their sets of permissible values.
2. A ValueFrom constraint is compatible with/less general than a FromRange constraint if some/every value within its set of permissible values lies within the range of values allowed by the FromRange constraint. Note that it is impossible for a ValueFrom constraint to be more general than a FromRange constraint since any non trivial FromRange constraint allows an uncountable number of values.

The current implementation in fact only considers that a ValueFrom constraint might be compatible with a FromRange constraint when the ValueFrom constraint specifies that an attribute should have a single value. In principle this is too restrictive - consider comparing a declared capability capable of consuming wood planks of lengths 10,11 or 12 metres in length but of no length in between with one which could produce wooden planks of any length between 10.5 and 11.5 metres. Since the tolerance intervals of a typical manufacturing process typically takes values from a continuous interval and not a few discrete values, the practical importance of such scenarios is expected to be limited.

The ability to compare both abstract resource types and their attached sets of constraints allows comparisons to be made between actual constrained resource types. When reading the following definition it is useful to remember that, for any constrained resource type R, $Att_i(R)$ is automatically a constraint. Thus when comparing $Att_i(R)$ and $Att_i(R')$ one is comparing the constraints which the constrained resource types contain within the same attribute.

Definition 7 *Given two constrained resource types, R, R' with respective types and attribute sets T, T' and Att_i, Att_j then:*

- *$R = R'$ if and only if $T = T'$ and for each attribute, Att, $Att_i(T) = Att_i(T')$,* ie *the constraints on every attribute are equivalent;*[4]
- *if T is more general or equal to T' and for every attribute, Att_i, shared by both T and T', $Att_i(R)$ is equal to or more general than $Att_i(R')$ then R is more general than R' and R' less general than R;*
- *if either T is incompatible with T' or there is an attribute, Att_i such that $Att_i(R)$ is incompatible with $Att_i(R')$ then R and R' are incompatible;*
- *if none of the above hold then R and R' are said to be compatible. This relationship occurs when every type and constraint of R and R' are compatible, but certain elements of R are more general than their equivalent in R' while other elements are less general.*

This definition naturally extends to cover constrained resource sets and states:

Definition 8 *Two resource sets, R and R' are said to be of equal/more general/less general/(in)compatible type according to the relationship between their constrained resource types*

Definition 9 *Given two states, $S = (R_i), i = 1..n$ and $S' = (R'_j), j = 1..m$ then $S = S'$ if and only if $n = m$ and for each $R_i \exists$ a unique R'_j such that R_i and R'_j have equal type. If instead $\exists S_{sub} \subset S$ such that $S_{sub} = S'$ then S' is said to be a subset of S.*

This definition could in fact be further extended to cover ideas such as the intersection of two states. However the exact matches on which such notions would be based are in fact of lesser importance within the algorithm. Much more frequent is the case of an inexact match such as occurs when *e.g.* a company contributing a declared capability can in fact produce a more general type of resource than is strictly required. The following definition covers such situations:

Definition 10 *Given two states, $S = R_i, S' = R'_j$ then S is of respectively less general/more general type than S' if and only if it is possible to reorder the resource sets within S' such that $\forall R_i \in S, R_i$ is either of equal or (respectively) of less/more general type than R'_i. If a reordering of the R'_j exists such that no R_i is of incompatible type with its corresponding R'_i then S and S' are compatible. Otherwise they are said to be incompatible.*

[4]since the types are equal, the set of attributes are automatically also equal

Implicit within the above definition is that the two states compared must contain the same number of resource sets. In the case where one state has more resource sets than the other, one can talk of a subset of the resources in the larger state having a relationship with the smaller state, with the unmatched resources being a remainder. This idea will later prove to form a crucial stepping stone within the overall development of complete combined capabilities.

A basic example of two compatible types is a wooden table with area between 20 and 25 square units and an oak table with area between 15 and 25 units. While there is an overlap between the concepts defined by these two constrained resource types (oak tables with an area between 20 and 25 square units) it is smaller than either of the sets of types defined by the individual constrained resource types. If both table types were oak, the first would be a less general type (and vice versa).

5.1.5 Declared Capabilities and Potential Extensions

This section builds on the previous sections to present a representation for declared capabilities, potential extensions and how the these can be compared. The following is the main definition within this section:

Definition 11 *A* declared capability*, T, consists of a pair of states $S0_T$ and $S1_T$ together with a label,L_T. L_T is normally a string. However L_T can also be a special character, ?. A declared capability whose label is ? is a called a* potential extension*. $S0_T$ and $S1_T$ are respectively referred to as the input/output states.*

For the remainder of this thesis a diagrammatic representation is adopted whereby a declared capability is represented as a pair of states connected by an arrow containing its label. For example $[ElectricMotor] \xrightarrow{PumpMaker} [Pump]$ would denote a declared capability in which the PumpMaker agent could supply pumps if supplied with electric motors. This representation extends to cover potential extensions through using ? as the label on the arrow.

The two states $S0_T$ and $S1_T$ respectively indicate which resources are taken as input/produced by the declared capability. The input state is of special interest in that it will frequently be empty, seeming to suggest that the output is produced from thin air. In fact this simply indicates that the company needs no *external* resources to produce the output - typically they will either have stockpiles of, or existing suppliers for, the subcomponents they need. The label within a declared capability indicates which company is prepared to provide that declared capability.

A potential extension indicates where companies might profitably introduce new declared capabilities. The matching of declared capabilities into potential extensions is the basic step in the process of forming complete combined capabilities. While potential extensions suggest the kinds of processes that might be added to a combined capability, the matching of declared capabilities to declared capabilities is not based on a mechanistic comparison of their respective resources.

The most common situation is that a proposed declared capability can fulfil *part* of a potential extension. The parts of the original potential extension not met then cause the generation of new potential extensions, thus driving the team formation process. For instance a company might see a potential extension of the form $(\emptyset) \xrightarrow{?} (CarWindow)$ and realise they could produce a car window given a window pane. This would generate a new combined capability with $(\emptyset) \xrightarrow{?} (CarWindowPane)$ as its sole potential extension. In fact declared capabilities matching *none* of the resources within a potential extension can also be matched into potential extensions. Here the company simply considers that its contribution is likely to prove a useful intermediate step to meeting the requirements of the potential extension.

The comparison of declared capabilities with potential extensions uses the ideas of state comparison from the previous section but must also deal with cases where the corresponding states in the declared capability or the potential extension have different amounts of resource sets. This produces the following definition:

Definition 12 *An* identification *of a declared capability, $\mathcal{DC} = (S0 = (S0_i), S1 = (S1_i), AgentA)$, with a potential extension, $\mathcal{PE} = (S'0 = (S'0_k), S'1 = (S'1_m), ?)$ is a pair of functions $F_{input} : S0 \to S'0 \cup (\phi)$,$F_{output} : S1 \to S'1 \cup (\phi)$ where $\forall s \in S0$ either $F_{input}(s) = (\phi)$ or $F_{input}(s)$ is type compatible with s and $\forall s \in S1$ either $F_{output}(s) = (\phi)$ or $F_{output}(s)$ is type compatible with s. Here type compatibility is in the sense introduced in definition 10.*

Essentially this says that all of the resources within the declared capability are either identified with a specific compatible resource within the corresponding state of the potential extension or aren't identified with anything. Given this definition the following auxiliary definition is useful:

Definition 13 *Given such an identification the following sets are named:*

1. *$F_{input}(S0)$ is called the* consumed resources *and $F_{output}(S1)$ the* produced resources*;*

2. *$F^{-1}_{input}((\phi))$ is the* input remainder *and $F^{-1}_{output}((\phi))$ the* output remainder*;*

3. *$S'0 \setminus F_{input}(S0)$ is the* forwards unmatched resources *and $S'1 \setminus F_{output}(S1)$ the* backwards unmatched resources*.*

The following list, in which those resources within the potential extension are referred to as free to be consumed/needed to be produced, clarifies the intent of the above definition:

- the consumed resources indicate which of the resources free to be consumed are consumed by the declared capability;
- the produced resources indicate which of the resources requiring production have been produced by the declared capability;
- the input remainder represents the additional resources which the declared capability requires to operate;
- the output remainder represents those resources that the declared capability produces but which are not directly 'needed' in the potential extension;
- the unmatched resources represent those resources left untouched by the declared capability.

5.2 Combining the Contributions of Companies

The following section completes the presentation of the data types by presenting a full definition of a combined capability. A combined capability must combine two forms of information - the declared capabilities already contributed to it and the potential extensions to represent what must be done. Since these two structures both use states containing constrained resource sets as the core of their definition, such states are the natural building block for combined capabilities. The main difficulty lies in deciding how to relate the states within the combined capability itself with those within the combined capabilities contributed to it.

The most obvious solution is to directly use the states within these declared capabilities and augment them with extra states to represent those resources which are free for consumption/need production. An example of how such a representation might look can be seen in Figure 5.1. This figure illustrates a case where two companies manufacture A and B in house and a third makes C from this combination.

The difficulty with such a representation is that it does not represent which of the resources in the middle state is contributed by which declared capability. This causes a problem in generating the potential extensions - if for instance AMakers declared capability were to be removed it would not be clear if it was A or B which needed to be contributed. This problem is solved by introducing a set of partially ordered resource states to the combined capability. The relationships between these

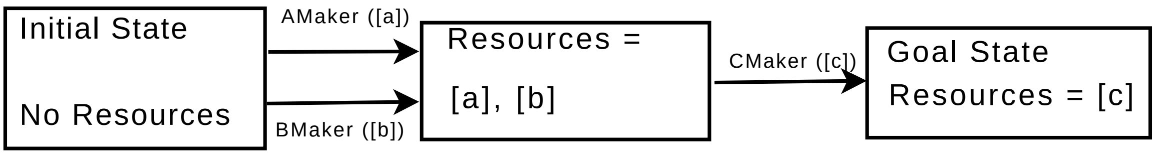


Figure 5.1: A simple example

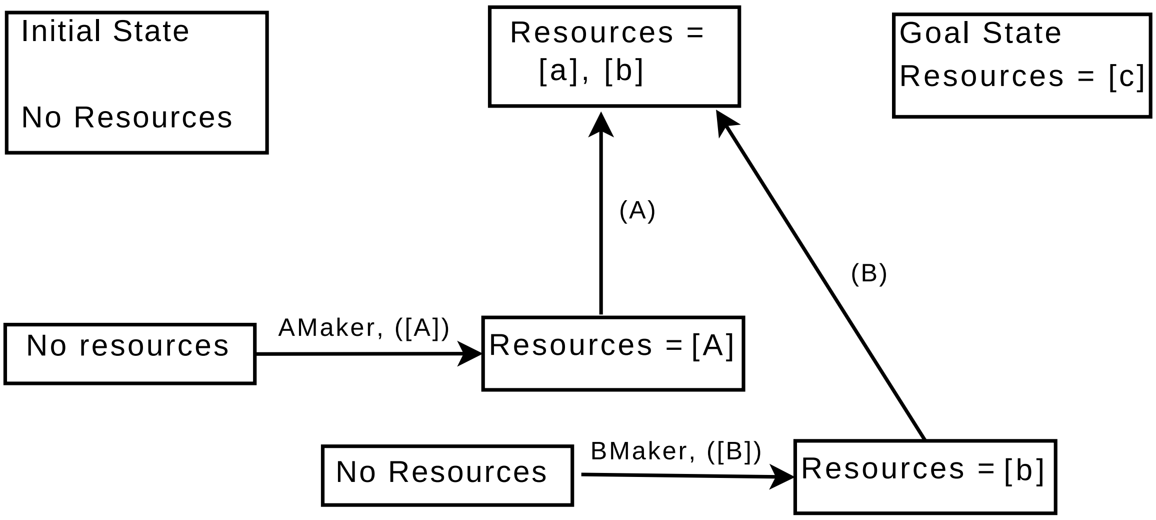


Figure 5.2: The actual representation

states and the resources within the declared capabilities are then represented by a set of projection functions. A picture of this representation can be seen in Figure 5.2, where the arrows from the declared capabilities to the state indicate where the contributed resources originate.

In fact the separation between the state and declared capability information within the following definition is only needed while still adding new declared capabilities. Once a complete set of declared capabilities has been located, the state structure can - and will - be removed to leave a partially ordered set of declared capabilities. The potential extensions can then be calculated using the information. In formal terms a combined capability is defined as follows.

Definition 14 *A Combined Capability, $\mathcal{PS}$, is a tuple $(\mathcal{S}, \mathcal{DC}')$ where $\mathcal{S}$ is a partially ordered set of states containing both a greatest and a smallest element, called the goal state,S_G, and the initial state,S_I respectively.*

Every member of $\mathcal{DC}'$ is a tuple, $(d, \mathcal{IS}_d, \mathcal{OS}_d)$ where d is a declared capability and $\mathcal{IS}_d : S0_d \rightarrow \prod_{r \in s \in \mathcal{S}} r$ and $\mathcal{OS}_d : S1_d \rightarrow \prod_{r \in s \in \mathcal{S}} r$ are projection functions. For future notional convenience the tuple $(d, \mathcal{IS}_d, \mathcal{OS}_d)$ will be simply denoted by the capability d with the projection functions implicit.

Given a tuple $(d, \mathcal{IS}_d, \mathcal{OS}_d)$ the i-th input projection function*, $\mathcal{IS}_{di} : S0_d \rightarrow$*

$(\prod_{r \in s_i \in \mathcal{S}} r) \cup \emptyset$ is defined by $\mathcal{IS}_{di}(r) = \mathcal{IS}_d(r)$ if $\mathcal{IS}_d(r) \in S_i$ or $\emptyset$ otherwise. The output projection functions, $\mathcal{OS}_{di}$, are directly analogous. Thus $\mathcal{IS}_d \approx \times_{(i|s_i \in \mathcal{S})} \mathcal{IS}_{di}$[5]*.*

The image of these functions will be denoted using the same notation as the functions themselves, ie $\mathcal{OS}_{di}(S1_d) \in s_i$ will be denoted by just $\mathcal{OS}_{di}$, and denotes those resources in the i-th state produced (here) or consumed (for the input projection functions) by the declared capability.

These projection functions must obey the following constraints [6]*:*

1. *$\forall d \in \mathcal{DC}', \forall s_i, s_j \in \mathcal{S}$ if $\mathcal{IS}_{di} \neq \emptyset$ and $\mathcal{OS}_{dj} \neq \emptyset$ then $s_j > s_i$;*

2. *$\forall s_i \in \mathcal{S}, \cup_{d \in \mathcal{DC}'} \mathcal{IS}_{di} \subseteq s_i$;*

3. *$\forall s_i \in \mathcal{S}, \cup_{d \in \mathcal{DC}'} \mathcal{OS}_{di} \subseteq s_i$.*

The constraints on the projection functions within this definition enforce causality - the first ensures that no declared capability consumes a resource before it has been contributed to the algorithm and the final two that no resource can be consumed (respectively produced) twice. All combined capabilities within this thesis will be considered to have been iteratively generated by starting from a specific form of initial combined capability and consequently extending it through the addition of declared capabilities. Combined capabilities generated in this way will be seen to meet the constraints in the above definition by induction: the admissible forms that the initial combined capability can take, discussed later in Section 6.2.1, meet them, while the algorithm for extending combined capabilities with declared capabilities preserves them.

5.2.1 Generating Potential Extensions

The following discussion will show that it is possible to mechanically derive the set of potential extensions within a combined capability. Henceforth they will therefore be considered to form part of the combined capability.

Definition 15 *Given a combined capability $\mathcal{P} = (\mathcal{S}, \mathcal{DC}' = (d, \mathcal{IS}_d, \mathcal{OS}_d))$ and a state $s_i \in \mathcal{S}$ then the set of resources free at s_i is given by $s_i \backslash \sum_{d \in \mathcal{DC}'} \mathcal{IS}_{di}$. Analogously the set of resources required at s_i is given by $s_i \backslash \sum_{d \in \mathcal{DC}'} \mathcal{OS}_{di}$. These are denoted $free_{s_i}$ and $requireds_i$ respectively.*

[5]This is not direct equality, and in fact not even a function, due to the bits of each projection function mapping to $\emptyset$. If these are ignored then it is equality.

[6]There is a potential confusion here between the labelling on the states ie S_j, S_i etc and the partial ordering on those states. The labelling is in principle completely independent of the partial order - in fact in practice it reflects the order the states were created. Hence $S_i < S_j$ simply means that S_i is smaller than S_j with respect to the partial order between the states and i might be numerically larger than j

Here $free_{s_i}$ corresponds to the resources which are available for consumption and $required_{s_i}$ to those which need to be produced. As with the potential extensions themselves these will be treated as part of the combined capability. This following definition defines how potential extensions are generated from a combined capability:

Definition 16 *Given a combined capability* $\mathcal{P} = (\mathcal{S}, \mathcal{DC}' = (d, \mathcal{IS}_d, \mathcal{OS}_d))$*, then the* cumulative *set of required resources at a state s,* $CuReq_s$ *is given by* $CuReq_s = \sum_{s' \in \mathcal{S}, s' \not< s} required_{s'}$*. Note that incomparable states are allowed within this sum. Dually* $CuFree_s = \sum_{s' \in \mathcal{S}, s' \not> s} free_{s'}$*. A state s is said to be in CuReq/CuFree if and only if one of its free/required resources are within it. As a special case the initial state in a combined capability never has any resources required, and the goal state never has any resources free.*

Given $s, s' \in \mathcal{S}$ *then* $CuReq_s$ *and* $CuFree_{s'}$ *are said to be compatible if* $\forall s_i \in CuReq_s, \forall s_j \in CuFree_{s'}, s' < s$ *or s' and s are incomparable. The set of all compatible pairs of cumulative resources is denoted by* $\mathcal{CCR}_{\mathcal{P}}$ *and its individual members will be simply referred to as a pair of states,* (s, s') *where* (s, s') *refers to the (compatible) pair* $(CuFree_s, CuReq_{s'})$*.*

The set of potential extensions, $\mathcal{CP}$*, for a combined capability,* $\mathcal{C}$*, is then given by* $\mathcal{CP} = (PE = (CUFree_{OS_{PE}}, CUReq_{IS_{PE}}) | (OS_{PE}, IS_{PE}) \in \mathcal{CCR}_{\mathcal{P}})$*.*

The technique used to add a new declared capability to a combined capability is not affected if the declared capability is matched to a potential extension which contains more resources than strictly required, whereas using a potential extension with too few resources can lead to difficulties. This motivates the concept of maximal potential extensions:

Definition 17 *Given a combined capability* $\mathcal{P}$*, then a potential extension, PE, in* $\mathcal{P}$ *is said to be maximal if* $\exists$ *no* $PE' \in \mathcal{P}$ *such that* $PE'_{IS} \supseteq PE_{IS}$ *and* $PE'_{OS} \supseteq PE_{OS}$*.*

The set of maximal potential extensions of a declared capability is henceforth denoted by $\mathcal{MCP}$, and, unless specifically stated to the contrary, all potential extensions within this document are considered to be maximal.

5.2.2 Adding a New Declared Capability to a Combined Capability

The next step is to develop an algorithm allowing for the addition of a specific declared capability to a given combined capability. In essence this algorithm is driven by identifications between declared capabilities and appropriate potential extensions. In brief the sets of resources within the identifications are dealt with in the following manner:

- the consumed/produced resources are marked as having been consumed/produced by the added declared capability;
- the input remainder resources are added to the combined capability as resources needing production;
- the output remainder resources are added to the combined capability as, potentially unnecessary, resources available for consumption;
- the unmatched resources are unaffected.

This gives rise to the following algorithm for extending a combined capability with a given declared capability. It is important to be clear about the scope of this algorithm - all it does is allow a single declared capability to be added to an existing combined capability. In particular, no consideration is given to how either the declared capability was chosen or the declared capability to which it is matched was chosen - such matters are the responsibility of intelligent agents representing the companies within the system.

1. The inputs provided are a combined capability, $\mathcal{CP} = (\mathcal{S}, \mathcal{DC}' = (d, \mathcal{IS}_d, \mathcal{OS}_d))$ and a declared capability $\mathcal{DC} = (SO, S1, L)$.
2. An identification, $\mathcal{I} = F_{input}, F_{output}$, between one of the maximal potential extensions of $\mathcal{CP}$ and $\mathcal{DC}$ is set up.
3. If the input remainder of $\mathcal{I}$ is non empty then a new state, S_i, is added to $\mathcal{S} \in \mathcal{CP}$, where the resources of S_i consist of those resources in the remainder. The partial order is updated thus: S_i is incomparable to any state, S_j, such that $\mathcal{IS}_{dj} \neq [\phi]$ and less than any state where $\mathcal{OS}_{dj} \neq [\phi]$. Finally S_i is greater than the initial state and smaller than the goal state. The relation of S_i to the remainder of the states if then worked out by calculating the transitive closure of the partial order.[7],[8]
4. If the output remainder of $\mathcal{I}$ is non empty then a new state, S_i is added to $\mathcal{S} \in \mathcal{CP}$, where the resources of S_i consist of those resources in the remainder. The partial order is updated thus: S_i is incomparable to any state, S_j, such that $\mathcal{OS}_{dj} \neq [\phi]$ and greater than any state where $\mathcal{IS}_{dj} \neq [\phi]$. Further S_i is greater than any new state created from the input remainder. Finally S_i is

[7]unmatched resources are simply ignored here and in the output section, explaining why using 'over large' potential extensions causes no damage

[8]Here calculating the transitive closure is intended simply to mean using the transitivity of the partial order - ie $a < b, b < c \Rightarrow a < c$ to ensure that every state is related to every other state in some manner.

greater than the initial state and smaller than the goal state. The relation of S_i to the remainder of the states is then worked out by calculating the transitive closure of the partial order.

5. The two functions F_{input} and F_{output} from the identification are augmented to also map any resources they originally mapped into $[\phi]$ (the input/output remainders respectively) into the new states created in steps three and four. These new functions are called F'_{input} and F'_{output} and represent valid projection functions for $\mathcal{DC}$.

6. The tuple $(\mathcal{DC}, F'_{input}, F'_{output})$ is added to $\mathcal{DC}' \in \mathcal{CP}$ and the resultant new combined capability is the output of the extension.

The following example helps to clarify the above definition. It covers a situation in which a combined capability, $\mathcal{C}$, which contains only an empty initial state and a goal state containing $[c, d]$, is extended by a declared capability given by $([a], [c], ACMaker)$. The first thing to work out are the potential extensions of $\mathcal{C}$. Since there are only two states, and the initial state comes before the goal state in the partial order, there is only one pair of compatible states - $(Initial, Goal)$. Since there are no declared capabilities all of the resources are free in the direction they can be within the initial/goal states and the only potential extension is $[\phi] \xrightarrow{?} [[c], [d]]$.

The identification of this with the declared capability,$\mathcal{DC}$, $[a] \xrightarrow{ACMaker} [c]$ has:

- no matched resources forwards, [[c]] as the matched resources backwards;
- input remainder as [[a]] and no output remainder;
- no unmatched resources forwards and [[d]] as unmatched resources backwards.

As the input remainder is non empty a new state, S1, containing $[[a]]$ is added to $\mathcal{C}$, leaving $\mathcal{C} = [S_I = [\phi], S_1 = [[a]], S_G = [[c], [d]]$. The partial order is then updated from $S_I < S_G$ to $S_I < S_1 < S_G$. Finally the function $F_{input} : [a] \in \mathcal{DC} \rightarrow [[\phi]]$ is changed to $F'_{input} : [a] \in \mathcal{DC} \rightarrow [a] \in S_1$. The corresponding output function already maps the resource from the declared capability into that in the potential extension and hence, through that, to one in the combined capability.

These functions produce the projection functions required and the tuple $[\mathcal{DC}, F'_{input}, F_{output}]$ is added to the corresponding set within $\mathcal{C}$. This now has:

- states $= [S_I = [\phi], S_1 = [[a]], S_G = [[c], [d]]$;
- declared capabilities $= [[([a], [c], ACMaker), \mathcal{IS}_{d1}([a]) = [a], \mathcal{OS}_{dG}([c]) = [c]]]$.

In the declared capabilities section above, only the component projection functions which are non zero are shown - all the other functions such as $\mathcal{IS}_{dG}$ have empty images. Here the cumulative sets of resources forwards/backwards are:

- S_I has no resources in either direction;
- S_G has no resources free forwards and just $[d]$ free backwards (the projection functions take out the $[c]$ resource.);
- S_1 has no resource free forwards (the projection functions take out the $[a]$ resource) and $[a]$ free backwards.

Since there are no resources forwards, any potential extension clearly also contains no resources free forwards. However, if we look at the partial order with $S_I < S_1 < S_G$ we see two distinct potential extensions - $[\phi] \xrightarrow{?} [a]$ and $[\phi] \xrightarrow{?} [[a],[d]]$), the second coming from accruing the resources from S_1 and S_G across the partial order. Of these only $[\phi] \xrightarrow{?} [[a],[d]]$ is maximal and this is thus the potential extension exposed.

To see why only maximal potential extensions are used, consider the non maximal extensions of the initial combined capability. These are $[\phi] \xrightarrow{?} [c]$ and $[\phi] \xrightarrow{?} [d]$. It can be verified that the effect of identifying the declared capability with $[\phi] \xrightarrow{?} [[c],[d]]$. and $[\phi] \xrightarrow{?} [c]$ would be identical - the 'extra' resource d is simply ignored.

Moreover if a declared capability producing both [c] and [d] were to be identified with one of the smaller potential extensions, say $[\phi] \xrightarrow{?} [c]$ then the d resource would be identified as part of the output remainder and inserted into the combined capability in an extra state not clearly related to the already present state containing d. This could potentially cause highly undesirable problems and it is desirable to avoid this possibility completely by considering only maximal potential extensions.

Verification of the Correctness of Extensions

In order for the above algorithm to be useful it must map combined capabilities into combined capabilities - *ie* given a combined capability, a declared capability, $\mathcal{D}$, and a potential extension, $\mathcal{PE}$, with which it has been identified, the output from enacting the extension must remain a valid combined capability. In order to verify this, we consider how the constraints on the structure of a combined capability apply to the output of the algorithm, C'.

Clearly any new states added to C' will be valid states, the declared capabilities valid declared capabilities and the projection functions adequate. Less obvious is whether the partial order based constraints on the declared capabilities still hold within C'. The second of these, that $\forall s_i \in \mathcal{S}, \cup_{d \in \mathcal{DC}'} \mathcal{IS}_{di} \subseteq s_i$, clearly holds - the projection functions generated above do not map to any resources not either originally within the states of the combined capability or added as part of one of the remainders.

The first constraint is however more complex to demonstrate, namely that: $\forall d \in \mathcal{DC}', \forall s_i, s_j \in \mathcal{S}$ then if $\mathcal{OS}_{di} \neq \emptyset$ and $\mathcal{IS}_{dj} \neq \emptyset$ then $s_j > s_i$. The first thing to note is that it is only the newly added declared capability which could have broken this constraint.[9].

Now the projection functions alongside $\mathcal{D}$ were generated such that $\mathcal{IS}_{\mathcal{D}}$ maps onto resources from either one of the input states of $\mathcal{PE}$ from which it took resources or from the newly added state reflecting the input remainder. Conversely $\mathcal{OS}_{\mathcal{D}}$ maps onto the output states/output remainder. It is therefore sufficient to verify that:

- for every potential extension, $p = (s1, s2)$ within a combined capability, every resource within $s1$ comes from states either less than or incomparable to those states from which the resources within $s2$ come from.
- the states newly added from the input remainder are less than (or incomparable) to those from which the output resources came, and conversely those added from the output remainder larger than those from which the input resources came.

The first of these requirements was directly used to derive the requirement for two sets of cumulative free resources to be compatible within Definition 16 where potential extensions are defined. The second condition reflects exactly the conditions which the algorithm in Section 5.2.2 specifies must be placed upon the partial ordering of the remainder sets.[10] These constraints are therefore preserved and the algorithm above will always produce a valid combined capability from a valid combined capability.

5.3 Algorithm for Developing Complete Solutions

The following section concerns itself with how this is extended to produce complete combined capabilities - in particular how do we know when to start and stop adding declared capabilities to a combined capability?

5.3.1 The Input and Output

The following section addresses the two ends of the development of combined capabilities: how it starts - the valid initial forms for a combined capability - and when it ends - how complete combined capabilities can be detected.

[9]the other declared capabilities and the partial ordering between existing states are not changed when adding a new declared capability

[10]In fact the condition there is more restrictive - it forces strict $<$ or $>$ respectively.

Definition 18 *A combined capability $\mathcal{P}$ is said to be* complete *if* $\forall s \in \mathcal{S}, required_s = \emptyset$.

Essentially a complete combined capability is one in which the declared capabilities could be combined to fulfil the required goal of producing the product(s) originally requested - every resource within the combined capability has a company to produce it. There is not in fact a precise relationship between this formal output and an actual virtual organisation - the details of this relationship are discussed in the next chapter.

Not every combined capability is allowed as an initial combined capability - a minimal form of combined capability is proscribed along with specific rules for expanding it:

Definition 19 *a minimal initial combined capability contains just two states - the initial and goal states - and no declared capabilities.*

Thus minimal initial combined capabilities are of the form: $[InitialResources] \xrightarrow{?} [GoalResources]$. The general form of initial combined capability allowed is generated from this by adding additional states, with an appropriate partial order between them to reflect the order in which their resources should be produced/consumed. Combined capabilities produced in this way will be referred to as *initial* combined capabilities.

It can easily be seen that general initial combined capabilities are no more complex to solve than minimal ones - for instance solving one with initial states: $S_I = [[a]], S_1 = [[b]], S_G = [[c]]$ with $S_I < S_1 < S_G$ is directly equivalent to solving two distinct combined capabilities, one with S_I as its initial state and S_1 as its goal state and the other with S_1 as its initial state and S_G as its goal state. Due to the potential for recombining states it is less obviously straightforward what happens when multiple extra states are added - however, as long as the states are added one at a time and the partial order between them is kept consistent no problems will be encountered.

Later, when agents representing companies are introduced, a distinction will be drawn between declared capabilities coming from those agents - which create new combined capabilities but leave the original one on the noticeboard - and declared capabilities which *must* be included in any possible solution.[11] Such mandated declared capabilities are dealt with by extending the initial combined capability with them before placing it on the noticeboard. However while the non technical distinction here is significant, there is no technical distinction between these two cases and initial combined capabilities will be considered to not contain declared capabilities.[12]

[11] For instance those occurring when an OEM mandates a specific supplier

[12] In the full system shown later the 'initial' combined capabilities *will* sometimes contain declared capabilities - this is equivalent to using the extension algorithm to add a declared capability to a

5.3.2 The algorithm

The following subsection describes an algorithmic approach to generating complete combined capabilities from initial combined capabilities. This algorithm does not implement all of the approach discussed in the previous chapter - in particular it lacks the intelligent contributions of agents. Instead it forms the technical basis of a complete system which does implement that approach and is presented in the next chapter.

The following description uses three supporting sets: $\mathcal{CC}$ the set of combined capabilities, *complete* which contains the set of complete combined capabilities located, and $\mathcal{PE}$ which contains a set of tuples of the form (PE, CC) where PE is a potential extension of the combined capability CC. In the following $\mathcal{PE}$will be regarded as containing just potential extensions which 'remember' the combined capability they were taken from. In the initial state $\mathcal{CC}$ has one member - a combined capability generated as described in Section 6.2.1, and $\mathcal{PE}$contains the potential extensions of this combined capability. The algorithm then consists of iterating the following steps:

1. an identification between a declared capability,DC, and one of the potential extensions, $PE \in \mathcal{PE}$ is created;
2. the combination of DC, PE, the combined capability, CC, which PE came from and the identification is used to create a new combined capability, CC', in the manner described in Section 5.2.2;
3. if CC' is a complete combined capability and CC' is not equivalent (in the sense of definition 20) to any complete combined capability within *complete* it is added to *complete*;
4. if CC' is not complete and not equivalent to any other combined capability within $\mathcal{CC}$ then it is added to $\mathcal{CC}$ and its set of potential extensions added to $\mathcal{PE}$;
5. the algorithm reverts to step 1.

5.3.3 Multiple Complete Combined Capabilities

One notable aspect of the above algorithm is that it does not have any natural stopping point: it will continue to produce combined capabilities until there are no new declared capabilities to add. Initially, it might seem as if it should stop once the first complete combined capability had been located. However, when dealing

combined capability.

with cooperative VO formation within a virtual organisation breeding environment every complete combined capability represents a different potential team, each of which might potentially go forwards to bid for the contract. This immediately creates several problems with putting forward the first complete combined capability located:

- the first complete combined capability located is unlikely to be the most efficient solution possible using the members of the VBE - finding efficient solutions is particularly important when the team generated will later have to tender competitively for the contract;
- in general the members of the VBE will not agree as to which is the most efficient solution. In particular there will be a natural tendency for them to prefer solutions of which they are a member to ones in which they aren't.

These elements combine to mean that the algorithm should ideally attempt to generate every possible complete combined capability. To do this every combined capability generated is kept even once it has been extended. This approach means that a 'pool' of combined capabilities must be kept and made visible for extension and permits the natural exploration of multiple possible solution paths. This decision means that in principle the algorithm can run, piling up new combined capabilities and complete combined capabilities, until people cease wanting to extend them. Since there is no known set of processes which might be used to extend the combined capabilities there is in fact no 'algorithmic' end point identifiable. Within the context of the fuller system, this is not a problem - once the deadline for bidding for the business opportunity has passed there is little purpose served in generating more combined capabilities.

5.3.4 Symmetrical Solutions

One consequence of keeping every combined capability generated is that the performance of the overall algorithm becomes greatly dependent on minimising the number of combined capabilities considered. This problem has both technical and non technical aspects. The most important technical aspect is detecting symmetrically equivalent combined capabilities.

This problem can be illustrated by considering a case with three declared capabilities: $[\phi] \xrightarrow{am}][a]]$, $[\phi] \xrightarrow{bm} [[b]]$ and $[[a],[b]] \xrightarrow{cm} [[c]]$. If these are matched into an combined capability with goal state, $[[c]]$, and initial state $[\emptyset]$ then ideally the order in which the declared capabilities of am, bm and cm are applied should not affect the final complete combined capability generated.

Within the initial basic approach above this does not happen - in this case two distinct combinations are possible. The first is given by cm then am then bm and produces a solution with the following states: $S_I = [\phi], S_1 = [[a],[b]], S_G = [[c]]$. The second is given by am then bm then cm and produces the following sets of states: $S_I = [\phi], S_1 = [[a]], S_2 = [[b]], S_G = [[c]]$.

While they are not directly equivalent, intuitively the two complete combined capabilities mentioned above represent the same team and should be equal. The difficulty lies in producing a criteria for identifying non equivalent combined capabilities. One candidate solution is to identify two combined capabilities who contain the same set of declared capabilities. Indeed this criteria is sufficient to characterise distinct *complete* combined capabilities.[13]

For incomplete combined capabilities the state information must be taken into account. For example, if the goal in the above example was to create $[[d]]$ then one of am or bm might have been contributed to the $[[c]] \xrightarrow{?} [[d]]$ potential extension rather than the $[\phi] \xrightarrow{?} [[a],[b]]$ potential extension. Thus a more complex condition is required which also considers the structure of the states within the combined capabilities:

Definition 20 *Two combined capabilities $\mathcal{C}$, and $\mathcal{C}'$ are said to be equivalent if and only if they share the same labels on their declared capabilities*[14] *and have identical sets of maximal potential extensions.*

Since two combined capabilities ruled equivalent by this definition share all of the information used to extend them, they are 'functionally' equivalent with respect to future extensions and this definition is used in the system to reduce the overall number of combined capabilities considered.

5.3.5 Diagrams of Combined Capabilities

Representing combined capabilities within a diagram is rendered complex by the complexity and amount of data which must be represented - the states, their resources, the declared capabilities and the projection functions linking the declared capabilities into the state structure.

One approach is to represent the partial order between the states by drawing them in a graph (incomparable states are parallel with each other), positioning the declared capabilities beside this graph and drawing the projection functions between

[13]While it is pedantically possible that the same set of declared capabilities can be fitted together in a truly different order this possibility is considered to be unlikely enough to disregard, especially as the final output - teams - is identical.

[14]ie the same people have contributed declared capabilities to them

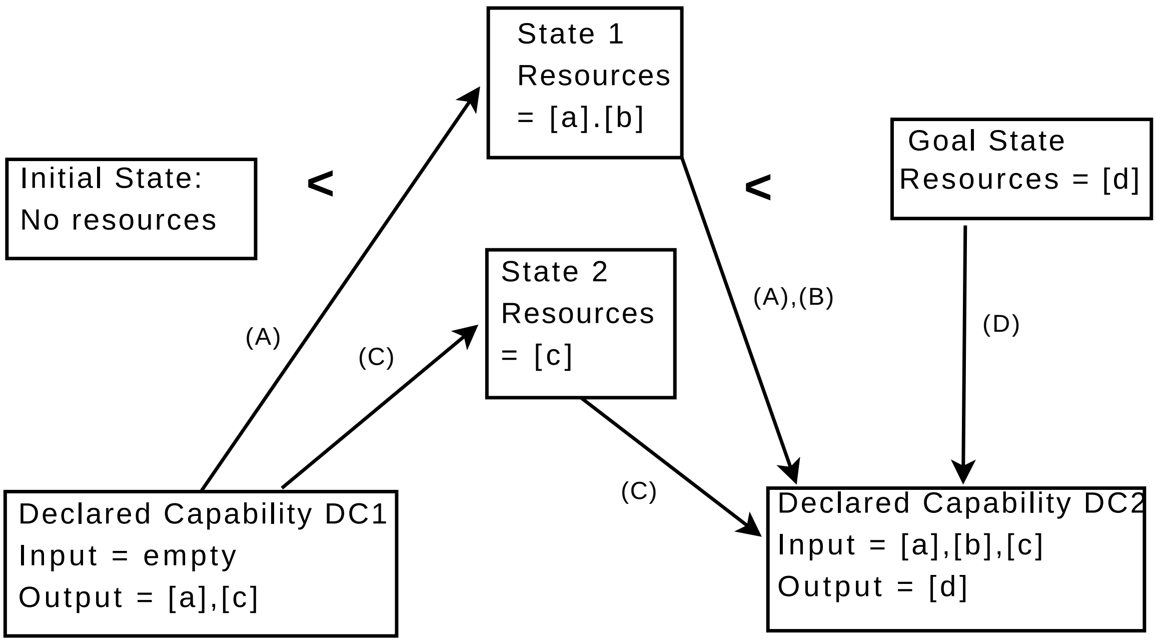


Figure 5.3: The most thorough representation

the resources. An example of this representation can be found in Figure 5.3. Within this diagram the partial order is drawn on the diagram using < symbols (no explicit indication of when two states are incomparable is given) and the projection functions are drawn as going from the declared capabilities into the states within the resource they are producing/consuming marked. The example shown has a pair of declared capabilities and four states but is not complete - there is no declared capability producing the b resource within state1.

While the above representation is very accurate, it is also rather cumbersome - ideally a representation might combine all of the information into one, graph like, structure. This could be achieved by having the declared capabilities as the edges of a graph which contains the states as nodes. Whether a resource is mapped into by a declared capability would then be indicated on the resource itself.

The basic structure of this representation is good and is adopted within the remainder of this thesis. There is however a natural tension between the wish to minimise the number of arrows and that to properly present the parallel elements within the combined capabilities. To see this consider the two diagrams shown in Figures 5.4 and 5.5.

In the case of Figure 5.4 the diagram would need to have two 'gm' arrows - something which isn't ideal - were a,b, and c,d in two distinct states. Merging all the resources into one state however risks obscuring the natural parallel relationships within certain diagrams. An example is Figure 5.5 where using two states allows the

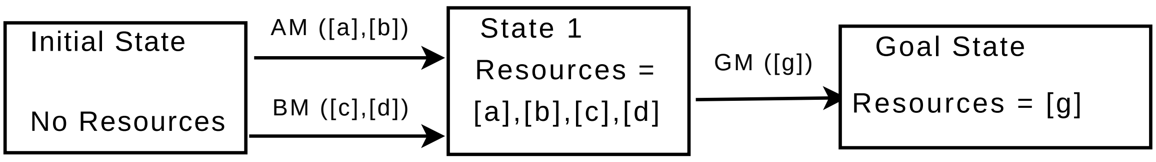


Figure 5.4: A combined capability in the chosen form

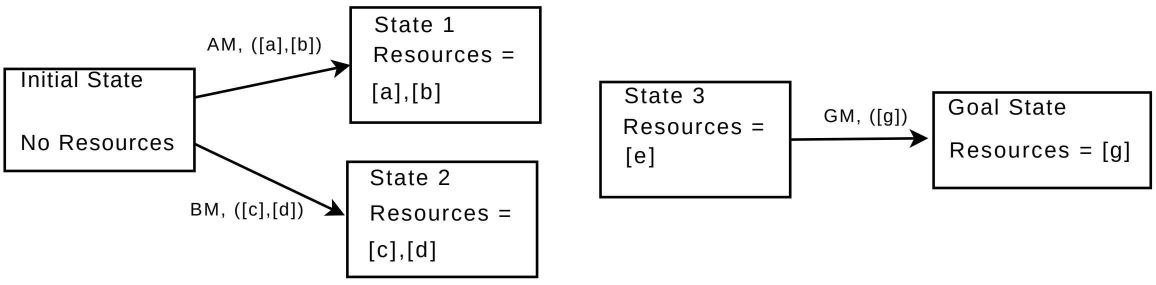


Figure 5.5: A combined capability more conducive to drawing

relationship between am and bm to be clearly depicted.[15]

At first sight this problem is not important - you have to commit to one form of representation and then cope with situations in which it produces non ideal diagrams. This problem can in fact be addressed: the rules for deciding when two combined capabilities are equivalent allow their detailed state structure to be altered. The combined capability produced by either combining the resources from a set of incomparable states into a single state or splitting the resources of a single state into a set of incomparable states is equivalent to the original.

To see this note that combined capabilities are considered equivalent if their potential extensions and contributors of declared capabilities are equivalent. Altering the state structure will not affect who has contributed to the combined capability and thus only the set of potential extensions might be affected. However if you take any state with multiple resources and split it into multiple states containing single resources, each inheriting the partial order of the original state but incomparable with each other, then the set of maximal potential extensions is left unchanged.[16]

Thus it is safe to manipulate the state structure of a combined capability into a form suitable for diagrammatic representation and this will be done in future. For additional convenience in all subsequent figures the partial ordering of the states/declared capabilities will be left implicit in their ordering from left to right while declared capabilities will be labelled with both their label and the resources they produce.

[15]The position of state 3 in this diagram has been slightly altered for cosmetic reasons - at this stage it is in fact only incomparable to states 1 and 2 and so should in principle should be positioned underneath those states rather than after them.

[16]The non maximal potential extensions are in fact slightly changed but this is not important

When implementing combined capability visualisation techniques for the demonstration software a more formal algorithmic approach was needed. The basis of this approach is to allow multiple arrows for each declared capability while maintaining coherency by 'splitting' each declared capability into three sections - a central section holding the label of the declared capability, a set of arrows from the states from which it takes resources to this central section and a set of arrows from this section to the states it contributes resources too. All of the arrows are labelled with the resources they concern. Finally, those resources which need to be produced are coloured red and every resource which can be consumed labelled orange. Examples of this approach can be found in later chapters relating to case studies.

Complete combined capabilities admit a much simpler representation than incomplete ones - they can be represented by a totally, partially ordered set of declared capabilities. This structure can easily be extracted from the structure within a complete combined capability using techniques relating to the algorithms used above. The graph of declared capabilities thus produced represents the basic information content of a complete combined capability - a group of people together with processes which can combine to produce the required product together with certain minimal ordering constraints.[17]

5.3.6 Worked Example

The purpose of the following example is to illustrate the technical steps involves in developing a set of combined capabilities by extending an initial combined capability with various declared capabilities. While reading this section it should be remembered that this process will be driven by intelligent contributions made by agents within the final system. Indeed this example will enable the importance of intelligent agents to be highlighted. Since it is designed to clearly illustrate purely technical aspects of the operation of the algorithm it focuses on a purposefully simple scenario with trivial resource types. Further it omits details of how new combined capabilities are generated from the old ones - this process was illustrated earlier in Section 5.3.2. The following set of declared capabilities is used within the example:

- $[[a]] \xrightarrow{BM} [[b]]$;
- $[\phi] \xrightarrow{CM} [[c]]$;
- $[[b],[c]] \xrightarrow{DM} [[d]]$;
- $[[a]] \xrightarrow{EM} [[e]]$;

[17]The partial order here only reflects fundamental input/output constraints and should not be confused with a workflow

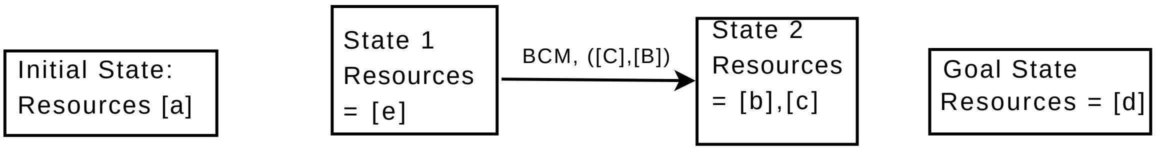


Figure 5.6: The first combined capability generated

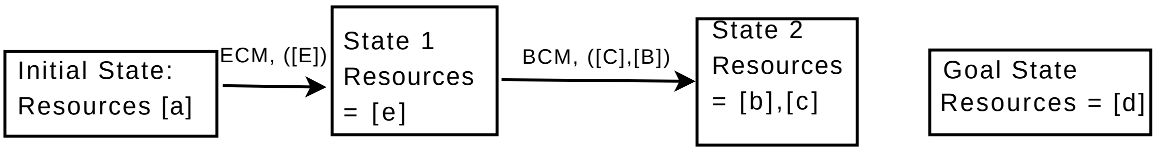


Figure 5.7: The second combined capability generated

- $[[e]] \xrightarrow{BCM} [[b],[c]]$.

The initial combined capability has just two states and can be represented by its only potential extension - $[[a]] \xrightarrow{?} [[d]]$. Throughout the remainder of this example the set of potential extensions is kept up to date but new combined capabilities are only shown when they are first added. Every combined capability is stored instead of being thrown away.

The first declared capability to be matched to this potential extension will be taken as that of BCM. This has been partly done to illustrate that, in principle, potential extensions and declared capability don't have to match at *either* end. The combined capability thus produced is shown in Figure 5.6, and the overall set of potential extensions is now: $[[a]] \xrightarrow{?} [[d]]$, $[[a]] \xrightarrow{?} [[e]]$ and $[[b],[c]] \xrightarrow{?} [[d]]$.

The following two figures, 5.7,5.8, then show how first EM then DM could naturally complete this combined capability. The only additional potential extension added when these two combined capabilities are added is a second copy of $[[b],[c]] \xrightarrow{?} [[d]]$ but referring to the combined capability which contains am rather than the one which does not. Both of these potential extensions are kept, and this nicely illustrates the importance of the overall solution being linked to potential extensions.[18]

An additional combined capability can be generated by applying DM - while the results of applying EM are detected as being equivalent to the complete capability already seen in Figure 5.8; the combined capability generated 'on the way' is not

[18]In fact as will be seen later when agents are discussed it really only the set of contributing agents which has to be attached to potential extensions

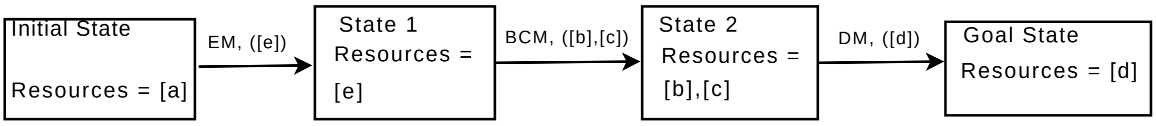


Figure 5.8: The first complete combined capability generated

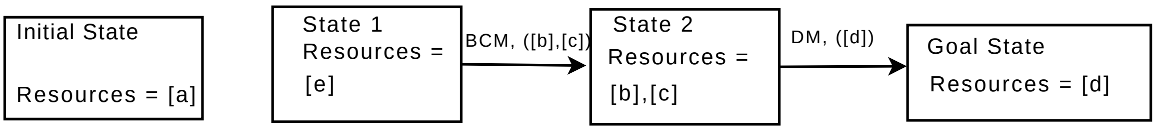


Figure 5.9: An 'excess' combined capability

equivalent to anything yet seen. In general this is one of the more intractable problems relating to controlling the number of combined capabilities considered - there is no way to avoid considering all of the partial ways to make a complete combined capability while retaining the flexibility of reasoning forwards, backwards and opportunistically. This 'excess' combined capability can be seen in figure 5.9. Finally it should be noted that this combined capability creates an extra $[[a]] \xrightarrow{?} [[e]]$ potential extension in the same way that the on in Figure 5.7 created an extra $[[b],[c]] \xrightarrow{?} [d]$.

While the above combined capability is ultimately superfluous in this scenario it is possible to think of scenarios where it is useful - perhaps $[[a]] \xrightarrow{?} [[e]]$ could be done in several ways. Several combined capabilities which are less likely to prove useful can be generated by matching BM and CM into various of these combined capabilities. Of these CM, having no prerequisites, could cause the worse havoc - potentially offering an endlessly increasing pile of c resources for consumption. For instance it could chain into the initial combined capability any number of times producing combined capabilities like the one seen in Figure 5.10. This behaviour provides a graphic illustration of where the approach needs the intelligent contributions of agents to work efficiently. The potential extensions generated by this set of combined capabilities would have the form $[[a],[c],[c],....] \xrightarrow{?} [d]$.

Finally it should be noted that there is an alternative solution path involving BM, CM and DM. The following three combined capabilities show this evolving in one order (BM,DM then CM). As with the discussion above this in fact produces several extra combined capabilities such as that produced by adding BM then CM which are not given here. For the purposes of showing the development of the solution clearly the [b] and [c] resources are not combined into one state in Figure 5.12. This requires the use of multiple arrows to represent DM.

5.4 Conclusion

The current chapter has formally developed the technical background required of a system implementing the abstract approach from chapter 4. Of special note are the detailed definition of declared capabilities, combined capabilities, potential extensions and an algorithm for extending a combined capability with a given declared capability - this is described in Section 5.3.2.

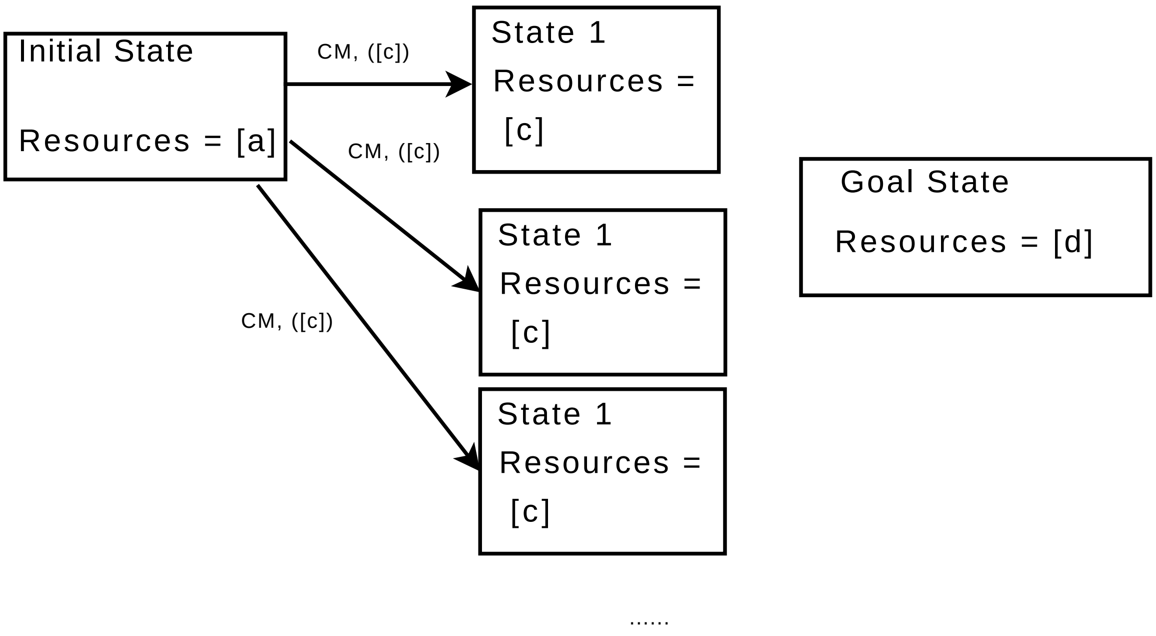


Figure 5.10: Too many c resources

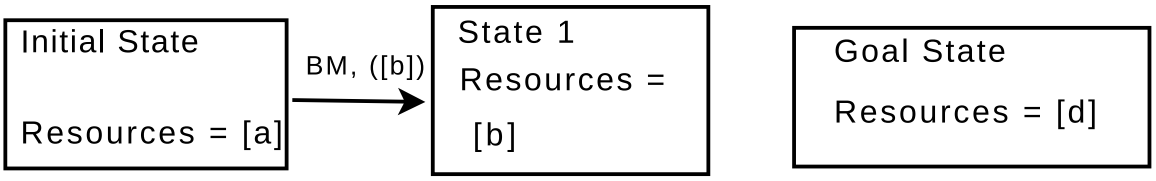


Figure 5.11: An alternative solution path, step 1

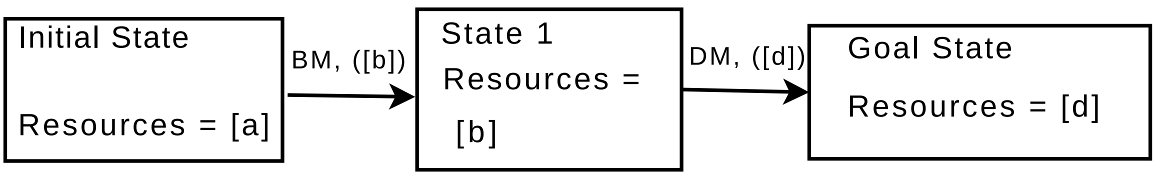


Figure 5.12: An alternative solution path, step 2

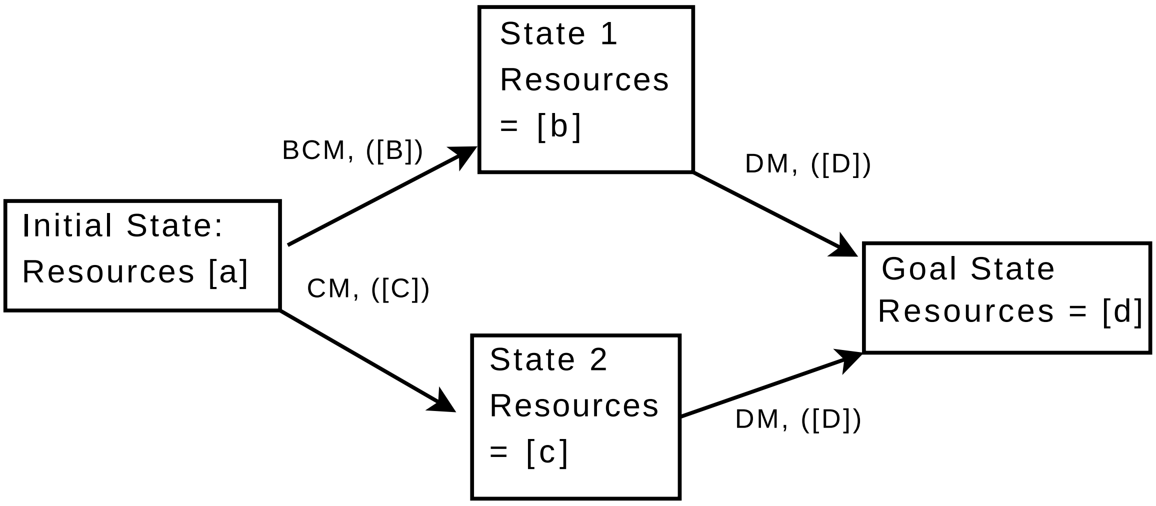


Figure 5.13: an alternative complete combined capability

What this chapter has not contained is any discussion of the motivation behind these extensions. Several important questions remain, including: how is a particular declared capability chosen to match the given potential extension? What does a complete combined capability actually mean?

The answers to these questions will become clear in the next chapter which considers how the ideas in this chapter can be combined with intelligent agents to produce a system implementing the abstract approach introduced within chapter 4. Indeed the contents of the current chapter should not be viewed as an end in themselves but rather as a necessary step in the development of such a system.

Chapter 6

Representing Companies: the role of agents

The primary purpose of the current chapter is to present a complete, distributed multi-agent system which reflects the approach presented within Chapter 4. A central aspect of this approach was that agents representing the companies within the VBE were responsible for the decision of which declared capability to propose and which combined capability to extend. In detail this chapter will consider:

- Exactly what types of agents are required to construct such a system?
- What reasoning capabilities does the system require these agents to have?
- What other features are needed to make such a system work?

The first section of this chapter focuses on identifying the types of agents required within the system, and the tasks assigned to each agent type. An outline of the overall system in which these agents operate is then presented. The chapter concludes by examining how the agents might be implemented within the system, and the reasoning requirements placed upon them when doing so.

6.1 The types of agents

The first step in the construction of a multi agent system is to characterise the types of agents within the system. The following section considers the agents required by a concrete system implementing the abstract approach from Section 4.2, with a particular focus on the tasks that each agent type must play. The core agent types required - capability provider agents and noticeboard agents - have already been identified in Section 4.2.

6.1.1 Capability Provider Agents

For the remainder of this thesis the agents representing companies within the VBE are termed *capability provider agents*. As identified in Section 4.2 each capability provider agent is responsible for holding a set of declared capabilities and contributing them where relevant. In detail each capability provider must perform the following tasks:

- storing declared capabilities;
- creating identifications between declared capabilities and potential extensions;
- controlling the growth of combined capabilities;
- dealing with complete combined capabilities.

The first two of these can be immediately grounded within the setting of the system discussed in Chapter 4:

- having knowledge of the processes that their parent company might wish to contribute;
- knowing which, if any, processes their parent company might typically wish to contribute to a given project and which potential extensions they should extend.

Capability provider agents should not be thought of as representing a small, fixed set of processes. Rather a typical capability provider agent has knowledge of a large set of potential processes and intelligently varies the ones suggested according to the attractiveness of the contract. For instance they might not bid at all on an unattractive project or offer custom designed processes for attractive ones.

The third role - controlling the growth of combined capabilities - contains several aspects. One aspect is that the agent should only contribute a declared capability when it is likely to prove useful. Other aspects will be further discussed in Section 6.4.4.

The final role of dealing with complete combined capabilities requires the agent to decide if the complete combined capability can be expanded into a team sufficient to make a bid for the business opportunity and in making the preparations needed for this. As previously discussed, this stage is viewed as lying largely outside the scope of the current system. Indeed it is considered that such negotiations will be conducted by people rather than their representative agents.

In addition the capability provider agents are responsible for representing their parent company's interests within certain protocols used within the noticeboard team formation system. A detailed discussion of these protocols and the role of capability provider agents within them can be found in the later sections discussing the detailed reasoning required of capability provider agents.

6.1.2 Noticeboard agents

The basic purpose of noticeboard agents, as introduced in Section 4.2, is to store the information needed to coordinate the formation of virtual organisations relating to a specific business opportunity. In detail this means they must be able to fulfil the following tasks:

- storing the information needed to fully define the business opportunity;
- storing combined capabilities;
- generating potential extensions from combined capabilities;
- creating new combined capabilities from identifications between potential extensions and declared capabilities;
- identifying complete combined capabilities.

Worthy of note with relation to the first two of these tasks is that not only should the agent be capable of storing the data concerned but also must be capable of answering queries about the data which it stores. The system used to perform such queries within the actual implemented test bed is discussed in Section 7.1.1. The final three tasks simply involve correctly applying elements from the technical algorithm introduced in Chapter 5 and are not further elaborated here.

6.1.3 The Noticeboard factory agent

One area not covered by the core agent types discussed so far is that relating to the creation of noticeboards. In particular the areas of identifying new business opportunities and using these opportunities to create new noticeboards have not as yet been assigned to any agent type.

The companies within the VBE will have existing methods of locating business opportunities. However many VBEs, including the Austrian Automotive Cluster, offer support for this process through services such as collective subscriptions to electronic marketplaces. In addition a VBE might wish to offer a single, centralised contact point for original equipment manufacturers wishing to present an opportunity to the members of the VBE.

The discussion above suggests the idea of create an agent type which provides a centralised service for locating BOs and creating noticeboards from them. This led to the introduction of the noticeboard factory agent. There is typically only one noticeboard factory agent for every system[1] and it is principally responsible for creating new noticeboards from business opportunities and offering services facilitating the location of new business opportunities. Since the noticeboard factory agent automatically knows of every noticeboard created within the system it is natural for them to store this information. In detail, the noticeboard factory agent performs the following set of tasks:

- locating appropriate business opportunities in electronic marketplaces;
- providing a central contact point for the proposal of new business opportunities to the VBE;
- creating new noticeboard agents from business opportunities;
- storing a list of the set of noticeboard agents created together with the initial information used to create them;
- answering queries as to which noticeboards they have created.

6.2 The consequences of intelligence

In order to presents a full multi agent system containing the agent types above, several additional aspects must be discussed. The unifying theme of this section is dealing with the consequences of giving control of the system to intelligent, self interested capability provider agents.

6.2.1 The information within business opportunities

So far a business opportunity has been considered to be wholly defined by an initial combined capability, which typically contains a combination of the products that must be produced to meet it and those products supplied to help this. This treatment of business opportunities allowed the presentation of a formal algorithm yet is inadequate for real world situations where the following additional elements should be considered:

- the information used by companies to decide how lucrative a business opportunity seems to them;

[1]Although there is no a priori barrier to having more

- the constraints placed upon the business opportunity by the customer;
- when does the initial combined capability contain more than just the basic structure.

Utility Information

Companies will often vary their level of interest in a business opportunity based on their perception of the potential rewards from participating in it. In particular they might offer custom processes for projects they perceive as lucrative, potentially buying in new machines or even new factories. In order for the capability provider agents to reflect the behaviour of their parent companies they must therefore be provided with the information required to perform such reasoning. This information is henceforth known as the utility information and considered to form part of the business opportunity.

The utility information is, by its very nature, highly domain dependent. The following list is based on the types of information typically found within automotive request for quotes and is also representative of related manufacturing domains:

- the name and location of the company to which the finished product must be delivered;
- the expected cost per item;
- the expected overall quantity of items to produce;
- the number of items which must be produced every year;
- the duration of the contract;
- the frequency of delivery required;
- whether the finished product must be delivered just in time or just in sequence;
- the date by which production of the product must start.

Constraints on the initial solution

A further type of information contained within a typical business opportunity, is the constraints that the customer places on the order. For example automotive sector OEMs often mandate that certain suppliers should be used to provide certain products. In this thesis the following, non exhaustive, list of constraint types is considered:

1. constraints stating that a given company should not be considered as a valid team member;
2. constraints stating that a given company should produce *all* resources of a given type within the eventual team;
3. constraints that all companies within teams fulfilling the opportunity should meet certain quality criteria.

The first two types of constraint relate to information that the noticeboard agent can itself check and can be enforced centrally. The third must be left to the companies and hence their capability provider agents to verify. Some typical requirements here might be that the team members have at least a certain score on some scale used by the OEM to measure suppliers, or that they all should possess certain qualifications.[2]

The Initial Combined Capability

The final area of interest is in the initial combined capability. The formal structure of this is unchanged from the previous chapter. The simplest form of initial combined capability from this has just two states, an initial and a goal state. This corresponds to a basic opportunity where the resources in the goal state must be constructed from those in the initial state.

This basic initial combined capability is then extended whenever the business opportunity specifies that a given supplier should be used to supply a given resource. This differs subtly from the constraint considered above that *only* a given supplier should be used to supply a resource in that it permits other copies of the resources to be supplied by other organisations.

Many automotive business opportunities come with an attached CAD drawing providing a detailed specification of the part required. While some of this detail can be captured within appropriate constrained resource sets, much of the detailed information relates to the detailed design of the components, something which is purposefully not directly addressed by the proposed system. This detailed information might still influence how interested companies are in the project, and so is stored in the appropriate noticeboard agent and made available to interested agents.

[2]Ultimately the OEM who originated the business opportunity will verify if such constraints have been met

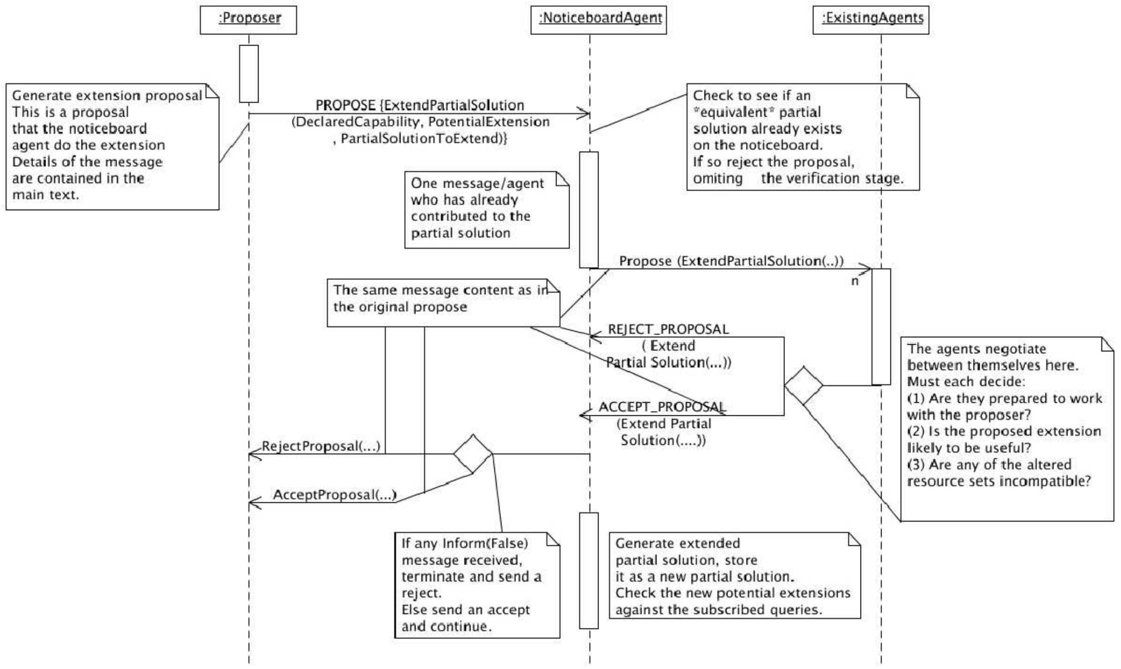


Figure 6.1: The protocol for verifying the acceptability of extensions

6.2.2 Contributing declared capabilities

Within the context of Chapter 5, extending combined capabilities with a declared capability involved creating an identification between a potential extension and a declared capability followed by the subsequent generation of a new combined capability.

When companies are considered, such an extension is effectively a request that the company proposing the extension be allowed to join the potential team represented by the combined capability. The existing team members should thus be given the opportunity to refuse the proposal. Within the system this is done using a simple protocol, detailed in Figure 6.1, in which each existing team member is given the chance to veto the proposed extension.

Within this diagram the existing agents class represents those agents who already 'members' of the combined capability - *i.e.* those who have already contributed declared capabilities to it. In fact - as discussed below in Section 6.4.4 - this protocol offers the additional functionality of allowing the capability provider agents to exercise a form of distributed search space control.

Approximate Type Matching

In all of the solution expansions shown so far, the resources in the combined capability and those being added from the declared capability have matched exactly. Formally the two sets of resources are only required to be *type compatible.* This allows companies to express their ability to produce resources less/more general than those currently required.

For example a potential extension might require steel girders of length between 1 and 2 metres while a company could only produce them with lengths between 1.2 and 1.5 metres. In such cases the resources in the combined capability must be replaced with the less(more) general ones produced by the company to ensure that the combined capability remains a true reflection of the declared capabilities within it.

While such resource narrowing/expanding is vital to the systems correct operation, it can potentially cause problems. In particular, if some other company is already linked to the resource whose type is being changed problems can occur. The main source of such problems is when the type of a resource being consumed by a declared capability is narrowed - the process corresponding to the declared capability might no longer be able to operate with the reduced range of inputs. The consequences of such effects can be more subtle than stopping a declared capability directly consuming the resource from operating - narrowing the range of resources entering a manufacturing process will often narrow the range of resources produced by it. Declared capabilities consuming resources from the originally affected declared capability might be affected in turn.

For instance consider a company which paints metal sheets of area between 1 and 2 metres. When originally proposed its declared capability consumes unpainted metal sheets of area between 1 and 2 metres and produces painted sheets of area between 1 and 2 metres. If some company can then produce metal sheets of area between 1.5 and 1.7 metres they might match into the input state here.

If no safeguards were in place this would produce a combined capability in which the painter agent claims to be able to take sheets with an area between 1.5 and 1.7 and produce painted sheets of between 1 and 2 metres. If an agent wishes to consume painted sheets of areas between 1.1 and 1.2 metres - the combined capability would then show this as possible, but the overall combination of declared capabilities would not permit it.

Indeed it is possible for the functional dependencies to become more complex than this. There are three potential ways to resolve this problem:

- only permit agents to make 'safe' type alterations - more general types for inputs, less general types for outputs and never merely compatible types;

- every time the type of a resource changes calculate the 'knock on' effects of this change for the other resources within the system - in the example above this would entail the painter agent updating its output to [PaintedSheet(Area: 1.5 - 1.7)];
- simply allow such potentially incorrect combined capabilities to form and catch the problems once complete combined capabilities have formed.

The first suggestion would avoid any problems but would also restrict the set of complete combined capabilities generated by the system. The second suggestion could be implemented by inserting a new protocol after an extension has been accepted whereby the resources in the system are fed forwards along the chain of declared capabilities and the corresponding output resources updated. This would be time consuming. The third suggestion retains the potential for generating 'useless' complete combined capabilities, however several factors suggest that this might not be a serious problem: complete combined capabilities are never *guaranteed* to correspond to 'real life' teams and those companies whose declared capabilities are *directly* adversely affected by a proposed type narrowing/expansion can use the extension vetoing protocol presented in Figure 6.1 to reject the extension.

Only empirical testing could conclusively tell whether the time lost in generating 'useless' combined capabilities was compensated by the time not spent fully validating all extensions for function dependencies, and so which of the second or third solution is better. The currently implemented test beds uses the third solution.

6.2.3 Complete Combined Capabilities

The algorithm described within chapter 5 outputs complete combined capabilities. Since the system's principal aim is to support virtual organisation formation it is important to be clear of the exact relationship between complete combined capabilities and potential virtual organisations. There are two aspects to this discussion; how an individual complete combined capability can be expanded into a team capable of tendering for the contract and how the system treats multiple complete combined capabilities.

When considering how a complete combined capability might be extended into a team capable of bidding for a contract, both the information within it and that missing must be considered. A complete combined capability can be viewed as a partially ordered set of declared capabilities and provides:

- a set of companies willing to work together;

- a set of declared capabilities which in combination might be capable of producing the required goods;
- a minimal set of resource dependencies between these capabilities.

Essentially then it represents a group of companies who have good reason to believe that they might be able to combine to form a VO for the opportunity. However numerous elements must be discussed before these companies can bid for a contract. These include:

- exactly which process each company plans to use;
- how much each company wishes to be paid for their contribution to the project;
- the overall price they wish to charge within their bid;
- the logistics of combining their processes.

Not only are these elements individually non trivial, with the logistical elements being of particular complexity, but there is also a considerable degree of interaction between them. Finally these negotiations are of such a level of importance to the companies involved that full automation is unlikely to prove acceptable.

While it would be desirable to only generate complete combined capabilities which can eventually operate as virtual organisations, to do so the system would have to take into account *every* factor which might cause a complete combined capability to fail to correspond to a VO. Both the number of factors to be addressed and the fact that many of them cannot be truly taken into account before a full team is assembled mean that this would be impractical. The current system instead attempts to produce complete combined capabilities which are likely to be able to develop into virtual organisations. For instance the veto protocol ensures that the companies in a complete combined capability are willing to work together in principle, while the constrained resource types make it likely that the processes will actually fit together.

As well as the treatment of individual complete combined capabilities, the way in which the VBE treats the overall set of complete combined capabilities is of interest. This might vary from only the single 'best' team generated within the VBE being allowed to bid to permitting any team generated to bid. The first of these might occur within a VBE attempting to build a brand name in its own right. This would both give the impression of a monolithic company and protect the VBE's reputation from low quality teams. Since the agents within a VBE naturally differ in their choices of the best team, with an obvious bias towards those teams in which they are members, such a system requires an agreed, centralised method for comparing different teams.

Self interested companies not contained in the chosen bid(s) might still be prepared to sacrifice the chance to tender for the contract in exchange for the benefits from the VBEs overall reputation being strong.

Where the VBE has less collective purpose every team which wishes to bid will typically be allowed to do so. Even in such situations it seems unlikely that every possible team generated will wish to tender for the contract. The limiting factor here is that individual companies might wish to focus on the most promising teams in which they are involved to both protect their reputations and avoid wasting time preparing doomed bids. As every company will have an individual view of which teams are most promising, some negotiations within the companies will normally be required.

6.2.4 Terminating noticeboards

Once a noticeboard has been created then typically, as discussed in Section 5.3.3, the process of virtual organisation formation is allowed to run, generating multiple complete combined capabilities, until the deadline for bidding on the original deal has been met. There is no requirement for every noticeboard be based on a concrete business opportunity - noticeboards can also be used in order to explore how well the VBE could meet potential future business opportunities. This kind of exploration might enable the VBE to react faster to future business opportunities and to identify where they might be able to improve through developing new processes or attracting new members. Such gaps could be identified by examining 'nearly' complete combined capabilities which are a process short of being complete, but would be highly efficient if completed.

6.3 A Complete System for Supporting VO Formation

The following section summarises the previous discussions in order to present a system capable of supporting VO formation. Examples of how this system might operate in practice are given within later case studies.

To set up the system, the members of the VBE first agree on a shared set of basic resource types which is encapsulated within an ontology. A shared network is then set up between the members of the VBE and a noticeboard factory agent set up on this. Finally every VBE member creates a capability providing agent encapsulating both the processes they can offer and knowledge of when they wish to do this. This set up process is purposefully lightweight, thus facilitating the dynamic formation of

VBEs. Once this setup is complete, team formation is facilitated by the following steps:

1. A business opportunity appropriate to the members of the VBE is identified and rendered into a format that the noticeboard factory agent can use.

2. This is sent to the noticeboard factory agent which creates a noticeboard agent to coordinate team formation towards this opportunity.

3. Capability provider agents within the system examine the opportunity represented by this noticeboard. If it is of interest to them they then examine its potential extensions. For each potential extension that they wish to extend they identify an appropriate declared capability, set up an identification between it and the relevant potential extension and finally send this to the noticeboard agent.

4. The noticeboard agent requests the existing members of the original combined capability to validate the proposed extension and, if they agree, uses it to create a new combined capability.[3]

5. If this new combined capability is not complete, the noticeboard agent stores it and updates its set of potential extensions appropriately.

6. If the new combined capability is complete, then the agents who have contributed towards it are informed that they have participated in a complete combined capability, provided with the details of it, and can consider how to act on this information.

7. Steps 3-6 are repeated until, most often, the period to submit bids for the original opportunity has been reached.

6.4 Detailed Agent Design

The following section examines how the agent types identified might be implemented with a particular focus on the types of reasoning they must support, and how this might be done.

6.4.1 The Noticeboard Factory Agent

The most obvious requirement placed upon the noticeboard factory agent is that it should be capable of creating noticeboard agents when provided with the correct types

[3]The creation of the new combined capability uses the algorithm discussed in Section 5.3.2.

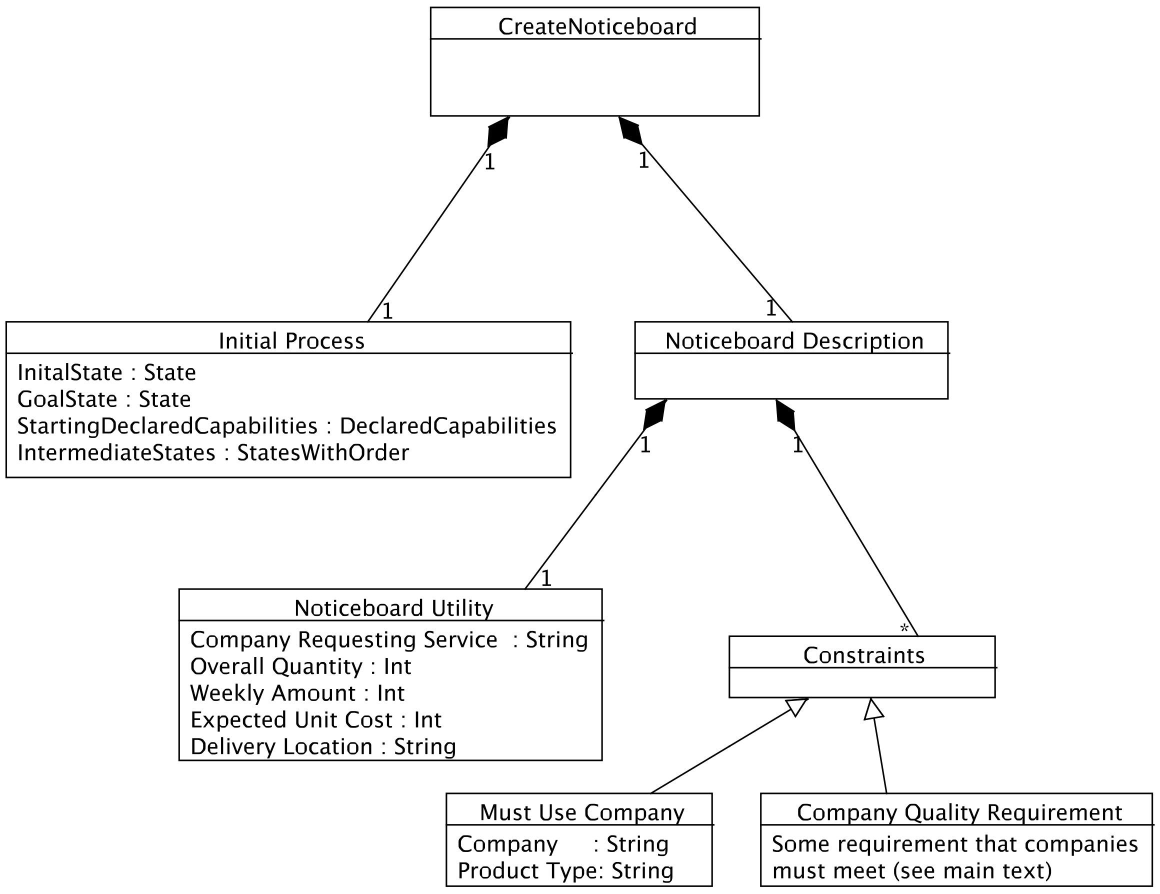


Figure 6.2: The structure used to create noticeboards

of information. The information provided to permit the creation of a noticeboard can be seen in Figure 6.2. As discussed in section Section 6.2.1 it is not possible to fully proscribe the forms this can take, and the exact form of the constraints/utility information in the figure should be taken as indicative rather than prescriptive.

Once such a request is received the noticeboard factory agent builds a noticeboard in the following manner:

1. First the initial process information is used to generate a initial combined capability. This uses the information in the initial process and the algorithm given in section 6.2.1. This starts with a simple combined capability containing just the initial and goal states. After this the 'states with order' are each added in - these are a combination of states with a partial ordering between them. Finally the starting declared capabilities are added in one at a time.

2. The noticeboard description information concept is enhanced with concepts such as the name of the (to be) created noticeboard and the time it was created. The utility information and constraints are retained.

3. The initial combined capability, description, constraints and utility information

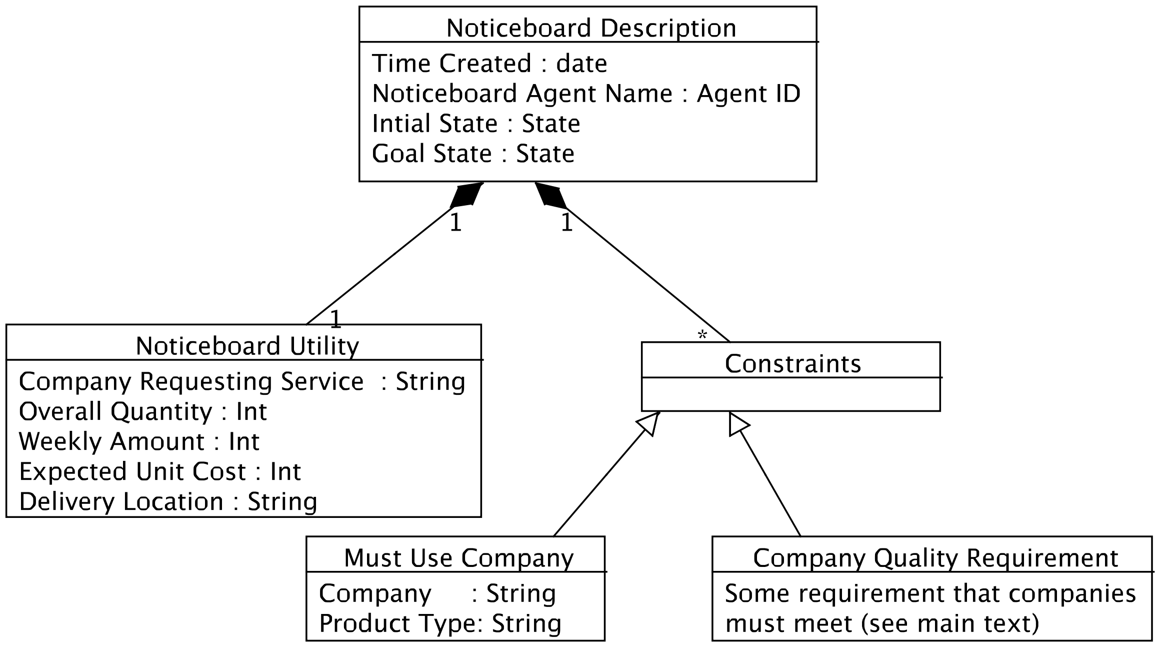


Figure 6.3: The structure used to represent created noticeboards

are used to create a new noticeboard.

4. Finally the information regarding this new noticeboard is stored within the noticeboard factory agent.

A summary of the information stored by the noticeboard factory agent when a noticeboard is created can be seen in Figure 6.3. This information is stored by both the noticeboard agent themselves and the noticeboard factory agent.

Capability provider agents must be made aware when new noticeboards are created. This could be achieved by directly broadcasting the information that a new noticeboard has been created to the capability provider agents. However this is an inefficient solution and instead the noticeboard factory agent acts much more like a directory service[4], in that it permits capability provider agents to query for all recently created noticeboard agents, and to subscribe to be informed of any new noticeboard agents created meeting such a query. In principle a query for a noticeboard could use any of the information the noticeboard factory agent has stored about that noticeboard. However it is anticipated that the following information will suffice for most queries:

- Time created - this allows capability agents to avoid being informed of the same noticeboards more than once when making repeated queries.
- The abstract resource types in the goal state - this allows an capability provider agent to be informed of, for instance, every project which involves producing

[4]In agent terms a specialised directory facilitator agent

a car wheel. Such information might be useful not only for companies which produce wheels but for companies who produce components often used within wheels.

- Overall Quantity - some companies may only be interested in large projects and some might be interested in *any* project of sufficient size.

While queries for resource types can use the full detail contained within the constrained resource types it is not anticipated that this will often prove necessary. This is because we are considering the classes of project that a company might be interested in rather than an opportunity to contribute a specific declared capability. Thus the exact details of the company's processes, and those of the constrained resource types, are of reduced importance and it is anticipated that a combination of the abstract resource types and the ontology comparing these will suffice.

When such a query is received the noticeboard factory agent notes which of its noticeboards match it and sends out the full information defining each one. In the case of a subscription the query is lodged with the noticeboard factory agent and checked against every new noticeboard as it is created, and the capability provider agent informed if it matches. Once an initial match is located the capability provider agents can examine the detailed information returned to them and decide if they are potentially interested in this noticeboard. If they are then they can query the noticeboards themselves in a manner examined in the next section. A detailed discussion of the implementation of the queries for both the noticeboard factory agent and the noticeboard agents can be found in Section 7.1.1.

6.4.2 Noticeboard Agents

This section focuses on the problems of storing combined capabilities, making them visible to the capability provider agents and reacting when a complete combined capability is detected.

Combined Capability Storage and Expansion

This role entails storing both the information defining the business opportunity and the potential extensions available within it. This information is then used to allow capability provider agents to query the noticeboard for detailed information of each of its combined capabilities and the potential extensions within them and thus decide where to extend them. Here a potential extension is defined by a set of resources as inputs, a set of resources as outputs and a set of agents already present within its parent combined capability. The following types of query are thus supported:

- the time that the potential extension was created;
- the resource types free for consumption in the potential extension;
- the resource types requiring production within the potential extension;
- the set of companies already within the parent combined capability of the potential extension.

Of these the set of companies already present within a combined capability can be used to ensure that a company does not propose to extend the solution of a company with whom they do not wish to work. The most important information is the set of resources which are free for consumption and those which require production. The combination of these two sets provides a 'potential' process into which the capability provider agents can match their sets of declared capabilities. The details of which potential extensions the capability provider agents might look for given the ownership of a set of declared capabilities are discussed in the following section.

Capability provider agents might make queries based on either the details of the constrained resources within the potential extensions or the abstract types of those resources. This decision is based on how general the capability providers processes are - if it can only consume/produce a specific subset of a type then it will query based on the constrained resources representing that subset. If it can produce a wide range of different configurations of the basic resource type then it might use abstract resource types. Even when the abstract resource type is used for the initial query, the capability provider agent can still use the detailed information provided in the matched potential extensions when deciding whether or not to propose an extension.

Finally, as well as direct queries to the noticeboards, capability provider agents are permitted to subscribe to noticeboards to be informed whenever a new combined capability contains a potential extension matching their query. Once either a subscription or a direct query has matched a potential extension, the noticeboard agent returns both the details of the resources and the set of agents present in that potential extension to the corresponding agent.

Complete Combined Capabilities

The consequences of the overall approach of the VBE to complete combined capabilities have already been discussed within section 6.2.3 above. In contrast, this section focuses on how noticeboard agents deal with individual complete combined capabilities and in particular on the information sent to those agents who have contributed declared capabilities to the complete combined capability.

Sending these agents the full complete combined capabilities would require them to understand the combined capability representation. Conversely sending the agents solely the set of declared capabilities within the complete combined capability would lose the ordering information within it - this provides a useful starting point for the design of a full process.

The solution adopted is to translate the complete combined capability into a partially ordered set of declared capabilities and to send this set to the process provider agents. The following, recursive, algorithm can be used to extract an appropriately partially ordered set of declared capabilities from a given complete combined capability:[5]

1. Take the set of declared capabilities, $\mathcal{DC}$, and form an unordered list of them.
2. Take the set of resources in the goal state of the complete combined capability, and identify the set of declared capabilities producing one or more of these resources. These declared capabilities represent the final declared capabilities in the partial ordering.
3. For each of these declared capabilities, $\mathcal{D}$, repeat the following steps:
4. Take the set of resources which $\mathcal{D}$ consumes.
5. Identify the set of all declared capabilities which produce one of these resources.
6. Mark each of these declared capabilities as less than $\mathcal{D}$. Calculate the transitive closure of the partial order.[6]
7. Recursively apply steps 4 onwards to each of these declared capabilities.
8. When all such resource based orderings have been created, all declared capabilities not already related in some way are marked as incomparable to each other.

That this algorithm will be able to operate can be seen by considering the condition for a solution to be complete. Its original phrasing is : "A combined capability $\mathcal{P}$ is said to be *complete* if $\forall s \in \mathcal{S}, required_s = \emptyset$". This can be rephrased as saying that every resource within a state of a complete combined capability is the output of some declared capability. Its termination is guaranteed by the existence of the initial state - the recursion will always bottom out with combined capabilities taking their resources from this state. The algorithm presented above ommits any declared

[5]This does not work for incomplete combined capabilities

[6]Here this will simply amount to noting that $a < b, b < c => a < c$.

capabilities not used in the production of the chain of resources leading to the goal state from the final solution. Such declared capabilities would not contribute to the performance of the VO ultimately produced.

6.4.3 Capability provider agents

The intelligence and ultimate control within the system resides solely within the capability provider agents. In reasoning terms these agents have the exacting task of representing their parent company within the system, and in particular of extending combined capabilities with appropriate declared capabilities. They thus must not only have knowledge of the processes their company might represent, something which is typically not a fixed set, but also the types of projects the company is interested in and when their processes would be useful. The following section will consider the requirements this places on the agents in more detail and suggest how such agents might be constructed.

Abstract Framework

Certain requirements can be placed on the external behaviour of capability provider agents within the system, such as outputting their processes as declared capabilities when proposing extensions. Their *internal* behaviour or reasoning cannot be controlled - capability provider agents belong to the company they represent and not the system. None the less it is useful to investigate approaches which might allow such agents to operate and the following section presents a potential skeletal theoretical framework for supporting the reasoning required by capability provider agents.

When and which process to propose

The first problem which must be addressed by a capability provider agent is the identification of which noticeboards represent promising business opportunities, and how promising each one is. This is likely to use a combination of the overall worth of the project, the types of resources which it aims to produce and other factors.

A basic requirement for a company to be interested in a project is the expectation that they could contribute to it. A simple method of recognising such situations would be for the capability provider agent to have some condition action rules trying to identify projects relating to the company's areas of expertise,*ie* Interested if and only if ProjectBuilds (CarWheels). These rules could then be built into queries/subscriptions registered with the noticeboard factory agent.

Such resource based rules would match against the type of resource to be ultimately created while the company only provides a certain subcomponent of the final product. Thus they are not based on specific subprocesses of the company but rather on identifying those types of projects to which they might be able to contribute - for instance the above rule about CarWheels might be used by a company producing tyres. A company which provided a standard, widely applicable, process might be interested in almost every project whereas a more specialised company might focus on certain classes of projects.

Once a capability provider agent has identified a potentially interesting noticeboard it must evaluate precisely how interested they are in it. Within this framework this is done with a utility function which takes all of the information defining the business opportunity on the noticeboard and produces a numerical evaluation of the opportunities attractiveness to the agent. Depending on this value multiple decisions are possible:

- noticeboards with utility beneath a certain threshold are ignored;
- noticeboards with utility above a certain threshold are referred 'upwards' to people within the agents parent company;
- noticeboards with values between these thresholds are considered by the agent.

The agent then uses this utility value to calculate a set of potential processes which they might be prepared to contribute towards combined capabilities on the noticeboard. This set of processes is typically allowed to vary, and will contain a larger number of processes for those noticeboards with higher utilities. This variation in the number of processes considered is designed to reflect the companies differing level of interest in the corresponding processes.

For noticeboards in which the company has only a marginal interest it may only be prepared to offer the 'routine' processes it can provide with little effort - for instance ones for which it already has the required machines. In contrast, the company might be willing to go to greater lengths to contribute to projects with a higher utility, including potentially purchasing new equipment - for instance a company specialising in heat treatment techniques might be prepared to purchase equipment enabling its machines to treat a wider range of materials if doing so permitted it to participate in a major project. This kind of reasoning could be represented by having the capability provider agent select from wider ranges of declared capabilities when considering contributions to projects with high utility values. Since declared capabilities involve no formal commitment by the company, the final decision of whether or not to buy new machinery remains in human hands and the capability provider agents can be programmed to suggest speculative processes without the risk of committing their parent company.

Some companies might be prepared to go great lengths for especially lucrative projects - for instance buying in new machines, or even constructing a new factory. While it is reasonable to expect a capability provider agent to detect such lucrative projects, the action taken by the company will typically be highly project specific. Thus the process provider agents might be programmed to refer such project 'upwards' to people within their parent company. These people could then examine the project and either hand create an appropriate set of possible declared capabilities or have the agent refer all potential extensions meeting certain criteria back to them. This hybridisation of automatic and human reasoning expands the range of projects in which the system is useful.

While the external representation of the declared capabilities within the system is fixed the only requirement on the internal representation is that the agents are able to extract declared capabilities from them. Hence an agent could potentially extract declared capabilities from a set of internal workflows. This would be possible for companies who supply a fixed set of routine processes but is not expected to be normal practice. Indeed it is unreasonable to expect a company to have a detailed plan of how they plan to enact a process before they know the details of that process - and such details are often far from fixed in the stages of team formation addressed within this thesis. Most agents are thus expected to have their sets of possible declared capabilities defined by a set of rules representing a set of processes which the company thinks it will be able to supply.

The following discussion assumes that a capability provider agent has identified a noticeboard, calculated its utility value and decided on a set of declared capabilities it is prepared to contribute towards the combined capabilities on this noticeboard. Its next task is then to identify the types of potential extensions in which it might be interested. This information is calculated from the set of declared capabilities and typically contains:

- those potential extensions containing resources free for consumption which are type compatible with a resource consumed in one of the declared capabilities;
- those potential extensions containing resources which need to be produced which are type compatible with a resource produced by one of the declared capabilities;
- potential extensions where no resources specifically match but the agent expects one of its declared capabilities may prove useful.

The third category reflects the fact that a company can often tell that a process is likely to be useful even when there is no direct need for it. For instance a glass manufacturing company can tell that any project involving the construction of windows will need a company to supply it with glass. Once the agent has identified the types of potential extension in which they are interested, they form a set of queries enabling them to identify them and send them to the relevant noticeboard agent, either through a periodic direct query or via a subscription.

Once a potential extension matching these queries is detected the agent evaluates the potential extensions and, in particular, the companies already within it. This might lead them to change their evaluation of the potential extension, potentially affecting the set of declared capabilities they will offer towards it. A likely scenario may involve the agent having a black list of suppliers with whom they do not wish to work and a list of people they are prepared to invest in forming partnerships with and use this to filter every potential extension that they locate.

Once the overall utility of a potential extension has been calculated the remaining issue is which declared capability to propose. The first stage here is that the agent uses heuristics to decide which of its processes are most likely to be contextually useful. If this leaves only one candidate then it is proposed. When more than one declared capability is judged to be likely to be useful the agent might react in several different ways:

- If the set of feasible declared capabilities are very similar - for instance different configurations of the same process - then the agent might propose a single declared capability covering all of them.
- If the declared capabilities are notably distinct in some way then the agent might propose multiple different extensions including one for each declared capability.

The creation of multiple, distinct combined capabilities in the second case allows other manufacturers to consider the distinct effects of the declared capabilities when deciding whether or not to join the team. For instance an agent might be able to manufacture a given component from either steel or copper. In the first case agents providing steel would be interested, while in the other it would be only those providing copper.

6.4.4 Composite Capability Control

Intelligently governing the extensions of declared capabilities is one of the most important tasks faced by the capability provider agents - unless intelligence is shown in when, and which, extensions are proposed the number of combined capabilities within the system can rapidly reach unmanageable levels. An ideal solution to this problem would be for every capability provider agent to possess a metric on the sets of constrained resource sets, enabling them to tell when contributing their declared capability made a combined capability 'closer' to completion. The use of such metrics has been well explored in normal planning domains and a classic example of their use is the A^* algorithm for pathfinding originally presented in [41]. This algorithm works by building up a set of partial routes before using the combination of the cost for each route so far combined with an estimate of the cost to complete each route to guide which solution to explore.

While such metrics are fairly easy to devise for pathfinding algorithms, with straight line distance being a feasible first guess for many types of maps, to create a reliable, *generally applicable* metric is much harder within manufacturing domains. In particular the 'distance' from one product type to another is wholly dependent on the set of processes available within the domain - something which, even were it static across projects, it is unreasonable to expect a given capability provider agent to know. It does however seem feasible that a process provider agent might know when their specific process is likely to be useful - for instance a company might know that the metal used to produce particular parts often needs to be heat treated in a specific way.

In addition certain simple heuristics are useful. The simplest of these is for the capability provider agents to have a memory of those composite capabilities to which they have already proposed extensions, hence allowing them to avoid repeatedly proposing the same extension if their original proposed extension is rejected. In fact, without this, since the original composite capability is retained an agent might end up repeatedly proposing an extension which has already been *accepted*.[7]. A similarly useful heuristic is for agents to be reluctant to contribute the same declared capability more than once to a given combined capability.

A more sophisticated form of composite capability control is provided by the ability of agents already within a combined capability to refuse an extension request. This can be done not only when the agents dislike the people proposing the extension but also when they already know of an alternative declared capability they consider more efficient. For instance some other company may already have proposed to perform the same task for less money. Such techniques which use the evaluation of previously identified results to prune the search tree are common in algorithms to play board games.

Whereas in standard applications there is a single body responsible for deciding when a more promising result has already been identified, here *every* agent already within the combined capability is free to veto a proposed extension if it thinks it

[7]While the additional combined capabilities would be detected as identical to existing ones and not stored, this would still waste considerable effort

knows of a better alternative. Since there is no a priori reason that these agents should share such evaluations, interesting 'social' effects can occur. The situation is highly reminiscent of some of those studied within game, and especially voting, theory. Of particular interest is the famous Condorcet's paradox, attributed to Concordet in 1785, of which a thorough study can be found within [38].

As described in [38] this occurs in a situation with three voters, V_1, V_2 and V_3, together with three alternatives A,B and C. Then, using $>$ to denote voting preference, if V_1 has preference $A > B > C$, V_2 has $C > A > B$ and finally V_3 has $B > C > A$ then the following pairwise comparisons of combined preferences all hold simultaneously: $A > B$, $B > C$ and $C > A$. Hence no option beats each of the others in pairwise comparisons. This means either that no result is produced or that, if the results are compared piecewise, the choice of the order of comparison determines the final result.

While this paradox cannot itself directly occur within the noticeboard system, a closely related problem can. To see this consider a case where a composite capability, P, contains declared capabilities from two agents, A and B. Next consider two agents, C and D, who can each offer the same declared capability extending P. Now assume that B prefers C to D but A prefers D to C. However *both* A and B would prefer either extension to no extension at all. In this case A might block proposed extensions from C while B blocked those from D. In this situation no extension is made - something which is not desirable for *any* of the agents involved.

As multi agent systems frequently feature negotiation it is no surprise that such problems have been studied within this context. A useful overview is offered in [72]. This covers considerably more ground than is necessary here, including a discussion of how to design protocols so that they force agents to vote according to their true preferences. The main focus of this work is on appropriate protocol design, and the protocols used in the current system are restricted to simple, one at a time, verification of the acceptability of extensions.

A solution must therefore be sought through altering the behaviour of the agents. The simplest way to do this is would be to make them less intelligent by stopping them from rejecting proposed extensions based on their knowledge of better alternatives. A better solution would be for the agents to act more intelligently, identify when these problems occur, and negotiate their way out of them. This requires a considerable level of intelligence on the part of the capability provider agents. Ultimately this serves as an illustration that highly distributed systems frequently exhibit unexpected behaviours, the results of which can be both undesirable and difficult to anticipate in advance.

6.4.5 Social and Privacy issues

Having discovered the inadvertent problems caused by the social behaviour of the agents described above, it is natural to worry that other problems may occur. Two potential areas stand out as potentially most serious:

- Companies may be worried about making certain type of detail public;
- A company may wish to manipulate the VO formation process in a manner which benefits them but acts to the overall detriment of the VBE.

The first of these concerns is principally mitigated by the nature of the information placed on noticeboards. The basic unit of information placed on the noticeboard is a declared capability. These only describe the visible effects that the company is committed to achieving. It thus most closely resembles an advert and does not give away and detailed process information. Indeed it does not have to link to any specific process. A combined capability then consists of a set of companies, with declared capabilities, who are prepared to work together. Finally the noticeboard system only concerns itself with the earlier stages of team formation, thus avoiding such potentially sensitive areas as detailed contract negotiations.

When considering how a company might manipulate the VO formation process, it is important to realise that a company can only affect the VO formation process in two ways. These are:

- Proposing to join the partially formed VO represented by a given combined capability
- Refusing to accept the proposal of another agent to join a combined capability of which they are a member

While the second of these does have a direct impact on the team formation process, it is entirely transparent and only affects the formation of virtual organisations of which the company concerned would ultimately be a member. It is thus essentially a mechanism allowing companies to choose who they wish to/not to work with and cannot be considered undesirably manipulative.

Because the original combined capability is also retained on the noticeboard even when an extension request is granted, the proposal to join combined capabilities can only affect virtual organisations of which the company ultimately becomes a member. Importantly it cannot be used to restrict the options for other companies to form virtual organisations.

While the system does not offer scope for constructive manipulation it is not directly protected against destructive manipulation. For instance a company may be prepared to lie about their declared capabilities, or decide to try and bring down the systems storage by endlessly proposing extensions. The protection in such cases comes from the fact that the system is being used in an environment containing companies who may well wish to work together in the future. When the agents represent real companies from within a VBE this motivation should be sufficient to ensure that such purely destructive behaviour is not encountered. In a looser environment where the agents might not represent real companies, additional safeguards might be required.

6.4.6 Simple Capability Provider Agents

The following section describes the simplest possible form of capability provider agent - a purely cooperative agent with a single fixed declared capability. This agent is purely cooperative in that it is happy to provide its declared capability whenever a need for it can be identified, irrespective of the nature of the business opportunity or its potential collaborators.

This form of agent will behave in the following manner within the system:

- it first subscribes to the noticeboard factory agent to be informed of all new noticeboards;
- for each new noticeboard it then subscribes with a query for potential extensions containing any types free for consumption/requiring production compatible with those its declared capability respectively consumes/produces;
- whenever these queries indicate the existence of a matched potential extension the agent proposes its declared capability as an extension for it;
- it accepts all proposals to extend combined capabilities of which it is currently a member.

As an example of how declared capabilities can be turned into queries, if the agent possessed the declared capability $WindowPane, WindowFrame \xrightarrow{Agent} Window$ then its query would look for all potential extensions involving any one of: window frames or window panes as consumable resources or Windows as resources which need to be produced. This form of agent is technically important since it represents the simplest possible type of agent which can operate within the system. Further the behaviour of a community of such agents is *externally* identical from that of a set of more intelligent agents who, after reasoning, decided to propose the same processes, agreed with all extensions *etc.* This, combined with their predictable and simple behaviour, makes these agents a perfect choice for testing concrete implementations of the system. If an implementation can work with these agents then it can also do so with more complex agents. Indeed such simple agents were used for testing purposes within the proof of concept demonstrator discussed in Section 7.1.

The system should not be judged on its performance in facilitating the behaviour of such simple agents, and in particular should not be directly compared to that of centralised systems under such circumstances. While centralised systems can deal with such simple cases, a primary motivation for the design of the system was to allow agents to reason when and which processes they wish to provide in a manner that a centralised system cannot. In addressing such complex cases it is unsurprising that the system loses efficiency in simpler ones.

This form of agent might represent a company capable of supplying a single type of standardised part - for instance a specialist car tyre manufacturer. Since a range of possible concrete examples can be encapsulated within a single constrained resource type the processes of such a company might be adequately represented by a single declared capability. In practice it is likely that a company would require more than one declared capability - this can be done by making one query for each of the declared capabilities required.

It is anticipated that most companies would want to use more complex agents than this, with more flexible process representations and more intelligent suggestions of processes being the norm. The previous two sections have suggested how this might be done. Indeed even the simple agents used within the demonstrators are restricted to only contributing their processes once to every combined capability.[8]

6.5 Conclusion

The current chapter has focused on combining the abstract approach introduced within Chapter 4 with the technical background developed in Chapter 5 to produce a distributed multi agent system suitable for supporting the cooperative formation of virtual organisations within a virtual organisation breeding environment. In doing this the chapter has focused on which types of agents are required within such a system and, in particular, on the reasoning required of these agents. Finally consideration has been given how the number of combined capabilities considered can be controlled within such a distributed system.

No specific claims are made regarding the details of the implementation of the system. Instead its primary value lies in its validation that it is in fact possible to develop a usable system which implements the abstract approach from Chapter 4. The development of this system also provided the required background for further

[8]This represents an unsubtle solution to the problems seen in Figure 5.10. More subtle solutions are desirable but require more intelligent agents.

validation of the approach through the development of a test bed demonstrator. The results of this development form a major part of the evaluation presented in chapter 7.

Chapter 7

Evaluation

The following chapter evaluates the effectiveness of both the proposed approach to supporting VO formation and the system developed from this. In turn its sections address the following questions:

- Is it possible to implement the system in a software system?
- How well might it perform?
- What benefits might it have?

7.1 Feasibility Evaluation

The following section addresses the first question; is it possible to turn the system proposed in chapter 6 into an actual software system? This section answers this question by describing the implementation of a proof of concept demonstrator and then discussing the major obstacles to turning this demonstrator into a fully fledged 'real world' implementation.

7.1.1 Implementation Technologies

Throughout this thesis it has been implicitly assumed that the overall system would be agent based: multi-agent based systems are ideally suited to the implementation of systems with distributed, intelligent control. The multi agent system used to implement the demonstrator was JADE, the Java Agent DEvelopment Framework, whose main website can be found at [36]. JADE is a widely used, open source framework for implementing multi agent systems which is fully compliant with the main agent standards outlined by FIPA, [34].

The system uses JADE's communication system and, in particular, uses its type system to represent the basic material types. Since JADE's type system fully supports the reasoning required of it and is flexible enough for the constraints to be introduced this did not restrict the implementation of the constrained resource types, originally discussed in Section 5.1.2. Another result from adopting JADE was the adoption of FIPA SL and ACL as, respectively, the content and communication languages within the system. These languages are both FIPA mandated standards and more information can be found both within the specifications for SL, [32], and ACL,[30] and within JADE documentation such as [6].

Queries - discovering noticeboards and potential extensions

The demonstrator supports two major forms of queries - capability provider agents can query for promising noticeboards and interesting potential extensions. The basic forms of information involved in these queries have been discussed in Sections 6.4.1 and 6.4.3 respectively. These queries were implemented using the standard query structure within SL: identifying referential expressions or IREs in combination with the FIPA Query,[31] and FIPA Subscribe, [33] protocols.

An IRE contains the following elements:

1. A reference operator which defines how many answers should be returned, and the appropriate error conditions: Any (return some answer or an error if there are none), All (return all answers, potentially an empty set) and Iota (return the *unique* answer, returning an error if either none or more than one answer is located.

2. A set of variables for which concrete values should be found.

3. A term containing these variables, and potentially further variables, contained within a set of predicates linked with And/Or/Not connections.

An example would be the following $(\mathit{All}?x)((WheelHubSpecification(?x))$ $\wedge(ManufacturedBy(?x,?y)))$. When answering this query the variable x is first bound to the results of the WheelHubSpecification predicate and would thus represent any form of wheel hub known to the system. The additional $ManufacturedBy(?x,?y)$ ensures that only wheel hub for which suppliers are known are returned. In order to additionally discover the suppliers of the wheel hubs you would use the query $(\mathit{All}?x?y)((WheelHubSpecification(?x)) \wedge (ManufacturedBy(?x,?y)))$.

The expressive power of an IRE is dependent on the predicates available to use within it. The following list contains those predicates supported within the implemented system for querying about new noticeboards and thus defines the sorts of queries that capability provider agents can use to discover new noticeboards of interest.

1. NoticeBoard(x). This is true if and only if x represents a noticeboard.

2. Produces/RequiresAbstractTypes(x,y/Set of Strings). This is true if and only if the noticeboard x contains an abstract type compatible with every type within y in, respectively, the goal/initial states of its initial combined capability. Here each member of y is a string corresponding to the type name of some concept from a product ontology.

3. Produces/RequiresConstrainedTypes (x, y/Set of Constrained Resource Types) this is true if and only if the noticeboard x contains an constrained resource type more general and compatible with every type within y in the goal/initial state of its initial combined capability.

4. RequiredByCompany(x, Company Names/Set of strings)). This is true if and only if the process required is being requested by a company within the set of companies within the predicate.

5. CreatedSince(x, t). This is true if and only if x was created at a time later than t.

The following example query uses these predicates to request the return of all noticeboards who were created after time t and whose business opportunity involves the creation of some form of wheel:

$(All?x)(Noticeboard(x) \wedge ProducesAbstractType(x, Wheel) \wedge CreatedSince(x, t))$

A similar set of predicates is used to support the identification of promising potential extension on noticeboards by capability provider agents. There are two changes, the first of which is that all references to noticeboards are replaced with those to potential extensions. In addition the *RequiredByCompany* predicate is replaced by a *ContainsDeclaredCapabilitiesFrom* predicate which allows an agent to identify if a specific agent(s) have already contributed to the combined capability. An example query designed to return all potential extensions involving the production of bolts or nuts, but not including company A might be:

$(All?x)(ProducesAbstractType(x, Bolt, Nut) \wedge$
$(\neg(ContainDeclaredCapabilitiesFrom(x, CompanyA)))$

The above query does not contain the PotentialExtension(x) predicate. In fact, since JADE uses typed variables within IREs, this predicate's main use is to express queries which should match every potential extension.[1]. IREs are the standard form used to express queries within SL and support for their expression is provided within JADE. However JADE provides no standard method for answering queries based on

[1]An identical note also holds for the Noticeboard(x) predicate

IREs. Thus in order to facilitate the construction of the demonstrator a basic generic framework for answering IREs was created.

7.1.2 The agents within the testbed

The implemented testbed focuses purely on demonstrating the technical feasibility of the system. It does not address any issues specific to a 'real world' implementation of the system. In particular the capability provider agents used within the system are of the simple form described in the previous chapter in Section 6.4.6, and lack detailed internal reasoning capabilities.

As argued within 6.4.6 the use of such 'simple' agents does not restrict the test beds ability to test the technical aspects of the system. Indeed the simplicity of the behavior of these agents made it much easier to identify the 'desired' result from any test run and thus to test that the algorithm was working as intended. The principal limitation of a demonstrator using these types of agents is that it cannot demonstrate the potential 'real world' advantages gained from allowing intelligent contributions by capability provider agents. The extent of these advantages could only be fully measured in by a full scale real world test, something which lies outside the scope of this thesis. The analysis within Section 7.3 illustrates the potential benefits of the system.

In addition the testbed contains both fully featured noticeboard factory and noticeboard agents. It therefore implements, and permits tests of, those aspects of the system not wholly reliant on the intelligence of the capability provider agents. Indeed many valuable insights into the best design of the algorithm were gained during the implementation of early versions. Finally the demonstrator served as a useful demo for introducing people to the system within the context of the CrossWork project, and to this end some extra visualisation software was implemented. Pictures from this software are used within the case studies below.

7.1.3 Proof of Concept Testbed Evaluation

The proof of concept demonstrator supports the creation of multiple distinct concrete test beds, distinguished by differing initial processes and different declared capabilities within the capability provider agents[2]. This allowed several tests. These tests mostly served to demonstrate the technical feasibility of the algorithm but also provided some insight into how it performed in varying conditions. For illustration one of these is shown here with pictures to illustrate the team formation process. Since

[2]Who remember in this case only have one each.

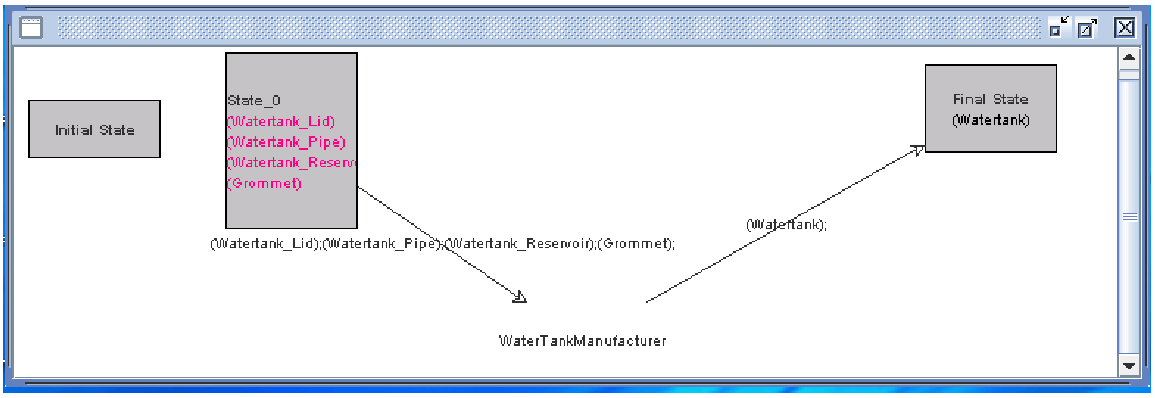


Figure 7.1: The initial solution

this is purely a technical demonstration no attempt is made to consider *why* each extensions is made.

The following example is loosely inspired by one considered within the CrossWork project and involves the formation of a team to construct a water tank. This case study also shows how additional declared capabilities can be incorporated within the initial combined capability - the initial combined capability is formed from a combination of

$$[\phi] \xrightarrow{?} [Watertank]$$

and

$$[Grommet, WatertankReservoir, WatertankLid, WatertankPump] \xrightarrow{WatertankMaker} [Watertank].$$

These are then combined to produce the initial combined capability seen in Figure 7.1.

The system contains four simple capability provider agents each of which provides one of the four elements required to complete the water tank. Each of these is added in turn to produce the final complete solution seen in Figure 7.2.

The combined capabilities generation within this test bed only requires that each of the declared capabilities should be matched with its corresponding potential extension - no declared capabilities beyond the first consume or produce multiple resources or indeed produce new states. Other test beds were used to test the algorithms ability to correctly generate more complex combined capabilities.

7.1.4 Feasibility of Technology Uptake

The test bed demonstrator presented within the previous subsection establishes the inherent technical viability of the noticeboard approach to team formation. Many

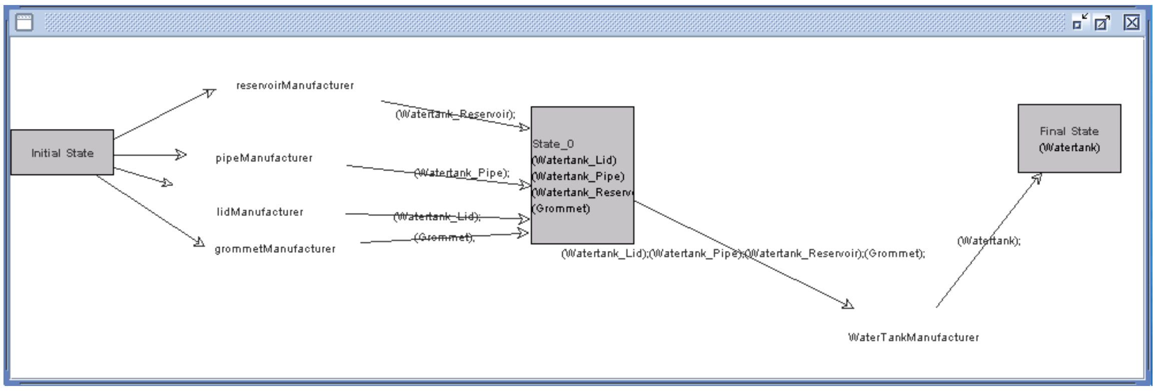


Figure 7.2: The final solution

obstacles remain between this work and an eventual deployment within an industrial scale system.

One major question is the systems ability to deal with the large numbers of combined capabilities produced when many companies are contributing declared capabilities. Since rapid growth of the numbers of combined capabilities considered is an inherent feature of the system, this is a major concern and is discussed in detail in Section 7.2.

Another obstacle to a practical implementation of the system stems from the entry requirements to the noticeboard system. While every effort has been made to reduce the infrastructure requirements that the system imposes on companies wishing to join it, there is no avoiding the need that every company should be represented by an intelligently programmed process provider agent. Thus the system will require many small, often not technically competent, companies to create potentially quite sophisticated capability provider agents. Indeed only these companies have the information needed to create and maintain these agents. This represents a substantial technical burden on these companies. One way to lessen this would be if the centralised part of the cluster organisation to provide the necessary technical support.

Ideally the companies would be able to produce these agents independently. This might be facilitated by using the skeletal abstract framework discussed in Section 6.4.3 as the generic basis for how such an agent should behave and providing a user friendly software interface which permitted this framework to be tailored to the specific requirements of each company. The specific details of how such software might be constructed are not discussed here. A general idea can be gained by examining the research within the area of end user computing. Research within this area focuses on the creation of domain tailored programming languages simple enough for end users to use.

An important early contribution to the field was made by Nardi in [56] where she discussed the need to enable users to create applications and some of techniques that might be used to support this. One of the major techniques proposed to do this is the construction of appropriate, domain specific programming languages. In addition to supporting normal end users such languages have often been constructed to support technically able users. The authors of [83] provide a useful annotated bibliography of this research.

While perhaps less directly relevant than work explicitly supporting end users this remains relevant to the task of creating a domain language for process provider agent creation. This very brief survey concludes by noting that [55] presents a thorough discussion of the challenges involved in creating new domain specific languages.

If software suitable for supporting the end user creation of capability provider agents could be provided and the system's performances proved adequate to deal with large numbers of companies then there would be no major technical obstacles hindering a real world implementation of the system. In particular the way that the system focuses on supporting the existing processes of team formation should make its usage acceptable to the companies concerned.

7.2 Performance Evaluation

7.2.1 Theoretical Discussion

The following section discusses the performance of noticeboard team formation and in particular the ability of its performance to scale in the presence of large numbers of companies and declared capabilities. This consideration is crucial to the systems potential application within real world scenarios. An estimate of this performance is presented.

The technical test bed above shows this to be a potentially significant problem. Even in such a simple scenario, with only four agents involved, up to 16 distinct combined capabilities are generated. Indeed were the identification of symmetrical solutions previously discussed within Section 5.3.4 not included within the demonstrator then up to 24 *complete* combined capabilities, along with many more combined capabilities, would be generated.

When viewed purely as a centralised planner it is clear that the performance of the system, being equivalent to blind brute force planning, does not scale at all well. In principle there is no upper limit on the number of partial solutions generated before a candidate complete combined solution is found. If one makes the simplifying assumption that every combined capability can only be contributed to each combined capability once, then the upper limit becomes 2^{DC} combined capabilities, where DC represents the number of distinct combined capabilities within the system. Indeed were symmetrically equivalent solutions not identified as equal, as discussed in Section 5.3.4, then a further factor growing in factorial fashion would be introduced.

Were this the whole story, the system would not scale well enough to deal with real world scenarios. However the suggestion of combined capabilities within the system is driven by intelligent agents. Even the minimally intelligent agents used in the technical demonstrators - who only suggested extending combined capabilities when they possessed a matching input or output resource and had not already contributed to that combined capability - make a large difference.

The theoretical performance of the system when using such agents is dependent on the precise declared capabilities of the agents within the VBE and cannot be precisely quantified in a general sense. It is however certainly large, as can be seen from the quantified example in section 7.2.2 below. Indeed when using such agents the search space examined by the noticeboard system is broadly equivalent in size to that searched by the input/output chaining algorithms frequently used within systems supporting web service composition.

In practice the use of more intelligent agents should further improve the systems performance. The major additional performance gain from the use of intelligent agents is that only those declared capabilities likely to be relevant to a specific problem are considered. This is crucial when considering the scaling of the system's performance when large numbers of actual companies are represented within the system - only a small subset of these companies will be interested in any given business opportunity thus considerably reducing the number of declared capabilities and hence combined capabilities considered.

The extent of this performance gain is dependent on the ability of the intelligent agents to use heuristics to identify when the contribution of a declared capability to a combined capability might be useful. Since this is dependent on the detailed nature of the problem and the construction of the agents within the system it cannot be fully quantified.

In addition the use of intelligent agents offers further potential performance gains such as those coming from the ability of the agents to reject extension requests if they have seen a declared capability which they personally prefer to the proposed one[3]. These benefits however seem likely to only have a minor impact on the overall performance of the system.

An appropriate estimation of the system performance is thus that it will generate about $2^{DC'}$ combined capabilities in order to solve a given business opportunity. Here DC' represents the set of declared capabilities considered *relevant* to the business opportunity by the agents within the overall system.

[3]See section 6.4.4 for a discussion of the potential effects of such subjective evaluations

7.2.2 Real World Estimation

Fully validating the real world performance of the system would require a full scale real world scenario in which various companies individually provided appropriate reasoning agents and teams for various business opportunities were found. Such a full performance test lies outside the scope of this thesis but remains an important area for future investigation.

It however remains possible to get some idea of the likely scale involved. To do this consider the case of the Austrian Automotive Cluster. As described in [1] this has about 220 members. Then take a typical automotive opportunity to construct a component which needs ten subcomponents and assume that roughly 5 per cent of the ACs membership are interested in providing each potential component. If we moreover simplify by assuming that there is only one decomposition and each supplier only provides one component then there are ten components with eleven options for each one.

Assuming a basic level of intelligence for the agents, so that they respect the 'structure' of the decomposition and do not suggest their declared capabilities twice this produces 11^{10} complete combined capabilities, and roughly 10^{32} combined capabilities. These are obviously substantial numbers. In practice this number would often be reduced, as some components would only have a few suppliers. The additional effects which might substantially reduce this number concern the ability of the various declared to link together. This concerns both the basic compatibility of the declared capabilities in terms of machines and processes [4] and the ability of the agents to trim the solution space by refusing solutions. For instance if there was only supplier for one component who will only work with 2 choices of people for 3 of the other components then the number of complete combined capabilities reduces to $8 * 11^6$. While no general formula can be given for the volume of the reduction of core components produced here, the potential for substantial reductions is clear. The number of combined capabilities will reduce accordingly, although the precise numerical effect there is less clear cut.

While this quantity of combined capabilities is very large, it does not rule out the systems application within the real world. In particular there is a considerable time scale available for locating potential virtual organisations. In addition it should be noted that the total number of declared capabilities generated does not directly greatly affect the speed that the system takes to find a solution. The system is not forced to search the entire space before stopping, and indeed the speed of producing a given chain of declared capability contributions is largely independent of the number of unrelated combined capabilities within the system.

[4]as represented in the system through the constraints in the attributes of the types

The basic problem produced by large quantities of combined capabilities are the storage requirements and the effects on the speed of querying for appropriate potential extensions. A technique which might greatly reduce the storage requirements induced by multiple combined capabilities is discussed in Section 8.2.1 below. This might clearly be of considerable practical importance in a potential real world application of the system. Finally it should be remembered that the theoretical examination of the performance of hierarchical planning and normal planning in Section 3.2.1 suggests that the performance of the proposed system is likely to be comparable to that of alternative approaches to supporting VO formation.

7.2.3 The effects of intelligence

This subsection is devoted to demonstrating how the use of intelligence can impact the total number of declared capabilities considered in the generation of combined capabilities. To do this a purposefully abstracted case study is used. Assume that the system is trying to generate solutions for the construction of metal boxes from a group of 20 companies, each with just one declared capability.

The first crucial use of agent intelligence is to ensure that no company proposes their declared capabilities more than once without good reason. This reduces the number of combined capabilities potentially considered from an effectively infinite value to the 2^{DC} figure from the theoretical discussion above, so 2^{20} in this case. The second is that only those agents with 'useful' declared capabilities contribute. Here the purpose is to illustrate how the 'raw' 2^{DC} figure from the theoretical discussion above reduces under realistic conditions.

For the further discussion I treat an idealised scenario in which the heuristics used to detect the applicability of declared capabilities to opportunities works perfectly, and where there is a single, known way in which to construct a box. This process will be assumed to contain four stages - first a metal sheet is produced, then a pattern is cut from it, then this pattern is heat treated and finally the box is constructed.

Further assume that within this group of 20 companies, there is only one agent who can make metal sheets, 3 who can cut them, 3 who can heat treat them and two who can construct them. This then produces a direct reduction of the number of companies interested in contributing to the business opportunity to 9 and thus the set of combined capabilities to 2^9. This roughly corresponds to the $2^{DC'}$ figure from the theoretical discussion above.

This figure is however still much higher than you would expect to find in practice with plausibly intelligent agents. For example, it seems reasonable that the agents might be knowledgeable enough to respect the basic 'pattern' of the process, ie that they generate combined capabilities containing at most one agent producing the metal sheets, one cutting them, one heat treating them and one doing the final manufacture. [5]. In this case the number of combined capabilities considered reduces dramatically to 56. While the precise number of combined capabilities is dependent on the number of agents capable of filling each role, the size of the reduction is typical. [6]

Importantly this reduction does not require that every agent should have a precise idea of the structure of these solutions. Indeed simple agents, such as those in the implemented test bed, which only proposed their declared capabilities once on the resources they consume/produce is free for consumption/production would achieve this level of reduction.

In a practical situation the system would be very likely to explore somewhat more than one solution path, and so produce more solutions overall. However even if we assume three solution paths with roughly similar numbers of combined capabilities to the one above for each the search spaces contains about 150 solutions rather than the 512 which would be produced by the unintelligent contributions of 9 agents, or the roughly one million produced by combining the unstructured contributions from all 20 agents within the VBE.

This makes clear that the ability of agents to only propose useful declared capabilities is the principal factor determining the practical performance of the proposed system. This applies to both allowing agents to identify interesting business opportunities at the top level and, once such opportunities have been discovered, to locate appropriate places for them to propose declared capabilities within them.

The system then contains additional features which build on top of this. The precise value of these cannot be generally stated but they will generally be much less dramatic than the reductions above. For example if the constraints in the attributes of the metal sheeting - for instance the type of metal used - produced by the first company are only compatible with 2 of each of the agents responsible for cutting and treating it, then only 35 combined capabilities are considered. Alternatively consider if the company producing steel sheeting was only happy to work with a specific company which cut it. In this case they will veto any proposal from the other two cutting companies to join a combined capability of which they are a member. In this case 38 combined capabilities will be generated. [7]

This case study clearly demonstrates the impact which the use of the intelligence by the agents in the system has on the quantity of declared capabilities considered in the generation of a solution, and especially the crucial role played by the identification of appropriate opportunities for agents to contribute declared capabilities.

[5]But with some containing no agents for some roles.

[6]All of the above assumes that symmetrically equal combined capabilities are identified and removed, as happens within the test bed

[7]The majority of them featuring no company producing metal sheeting at all

7.3 Analytical Case Study Evaluation

The remaining question to be addressed is how useful the system might be in real world scenarios. The following section contains a reasoned case study which aims to both illustrate typical modes of operation for noticeboard based team formation and to highlight situations where its support for intelligent team formation is especially useful. In order to do this it presents a case study, based upon one studied within CrossWork, considering what might have happened if the system discussed within this thesis had been used. While the set of modules, components and company names are retained from the original case study, the actual details of team formation are purely fictional.

The major benefits claimed for noticeboard based team formation are that it allows the process of virtual organisation formation to proceed in situations where the domain knowledge required to decompose the business opportunity into subtasks is distributed throughout the agents within the system and allows for the opportunistic formation of teams in response to the specific nature of business opportunities. The case study focuses on demonstrating these two aspects. It does this by presenting a series of hypothetical scenarios each of which shows how noticeboard based team formation supports the formation of a VO while catering for the specific circumstances of the business opportunity concerned. In several cases it will be seen to support the formation of teams which traditional approaches would not have considered.

7.3.1 Magna Intier

The main company within the case study is Magna Intier. The structure of Magna is discussed within CrossWork deliverable D1.1/2 [1] from which the basis of the following short discussion was taken. Magna Intier is a very large tier 1 automotive supplier with global reach focused on the engineering, manufacture and assembly of automotive components and systems of any scale up to the potential delivery of complete cars to original equipment manufacturers such as BMW.

While Magna operates as a single company, the main part of its structure consists of a set of companies who each operate largely autonomously. The central coordinating body within Magna is responsible for providing certain services such as advanced development and information technology and for ensuring that the companies have a greater level of security than they would have as individual companies. In particular there is no requirement or expectation that the companies within Magna should preferentially choose to work with other companies within Magna when forming teams to supply large systems. The process that a company within Magna goes through to form a VO is thus closely related to that of forming a VO from within a virtual organisation breeding environment, making this a suitable case study for the system proposed within this paper.

The case study focuses on Intier Automotive, one of six automotive system groups within Magna. Intier Automotive consists of a a group of companies who specialise in car interiors, within which they collectively have a wide range of expertise. A major aim for Intier Automotive is to contract with OEMs to develop and manufacture complete interiors as opposed to simply providing components within them. The advantage of this from the OEMs point of view is that they only have to deal with a single point of contact for the interior as opposed to coordinating a large number of sub suppliers.

7.3.2 Case Study Background

The real world scenario on which this case study is based started when Intier Automotive Eybl Ebergassing, a company within Intier Automotive henceforth referred to as Intier Ebergassing or Ebergassing, noticed that BMW were planning a new series of car. Based on this they identified the possibility for them to provide the entire interior for this series of car and approached BMW with this aim in mind.

Subsequently they received a contract from BMW to supply entire interiors for this series of car. Interestingly they received this contract before knowing which suppliers they intended to use for every subcomponent - clearly BMW were confident that Intier Ebergassing, with potential support from Intier Automotive, were a big enough company to supply the interior. In fact Magna found the team formation to be a time consuming process and have decided to form teams in advance in the future.

The fact that Intier Ebergassing had the contract with BMW before forming the team placed Intier Ebergassing in a privileged position in that, if they wished, they could be part of any team formed and were broadly free to choose which parts of the manufacturing process they wished to undertake personally. Finally Intier Ebergassing were responsible for interacting with BMW on behalf of the team formed. The existence of such a privileged supplier introduces a new element into the process of team formation, and its effects can be seen in the initial combined capabilities considered below.

Intier Ebergassing had two main partners on the project, Intier Eybl Straubing and the Magna Engineering center from Munich. Of these companies, Intier Ebergassing and Intier Straubing developed and produced 'most' of the components while Magna Engineering developed the overall design. The remaining suppliers were then chosen based on their past performance. However for certain components BMW mandated that a particular supplier should be used. Within the case study it is assumed that Ebergassing's potential external suppliers are drawn from within a VBE containing the system proposed in this thesis. The scenarios within the remainder of the case study are 'what if' scenarios intended to illustrate the potential benefits of using noticeboard team formation and bear no resemblance to the actual process of virtual organisation used.

7.3.3 General Approach

The chosen scenario commences when Intier Ebergassing receive a contract to supply an entire car interior from BMW. This is too big a task for them to deal with themselves and they must assemble a team to help them. The first step in this process is the formation of a high level decomposition of the interior into three main submodules - the cockpit, door covering and overhead systems. Separate teams must then be generated for each of these submodules, while Intier Ebergassing will assemble the completed modules into a complete interior.

For each module Intier Ebergassing must initially decide if they wish to perform the final assembly of the subcomponents. This decision might be based on their expertise in the area, and in particular on the number of subcomponents they could supply 'in house' - if they could produce most of the submodule themselves or through known suppliers, it makes sense for them to perform assembly. If they do decide to assemble a module the first step is for them to identify the subcomponents they need in order to perform the assembly.

Depending on their level of knowledge there are then several possible scenarios. If Ebergassing can fully specify the subcomponents required for the submodule then they first select those subcomponents which they either wish to produce themselves or possess a known subsuppliers for. Finally they produce a noticeboard requesting suppliers of the remaining subcomponents. In fact it is entirely possible that Ebergassing might be capable of identifying suppliers for all components without using a noticeboard.

It is also possible that Intier Ebergassing can only generate a partial decomposition,*ie* they can identify broadly which types of object are required but cannot specify them to a level of detail permitting direct team formation to occur. In such cases they create a noticeboard specifying:

1. that they will do the final assembly of the submodule;

2. those components they have identified as required from the decomposition but cannot directly identify a supplier for, or supply themselves.

Suppliers can then either propose to provide either entire components or part of them indicating those parts they require to complete the component. While this exact behaviour is not illustrated below, the behaviour in the case of the overhead systems module is closely related.[8] Finally it is possible that Ebergassing are interested in performing the final assembly but either lack the knowledge to provide a direct decomposition, or are not confident enough in their choice of decomposition to *impose* themselves as assemblers. In such cases they create a noticeboard with a basic initial combined capability but keep a close watch for chances to provide final assembly. This can be seen in the cockpit module scenarios.

Finally Intier Ebergassing might simply not wish to perform the final assembly of the component. In this case they do not impose a decomposition on the solution, and instead create a noticeboard with a blank initial solution. Following on from this empty partial solution there are several ways in which a solution might develop. One is that a company decides to provide assembly and a partial decomposition - this is illustrated in the Overhead systems module case study. However it is also possible for solutions to grow 'forwards' and part of the cockpit module case study shows this.

For the sake of clarity each submodule case study focuses on the formation of one particular type of team. In practice it is expected that multiple differing types of team would form for each module. Since this case study focuses much more on *why* each declared capability is proposed and added than on the technical details of how these extensions occur a simplified, linear representation is used for the combined capabilities generated. This builds on the declared capability representation to represent the a combined capability as a linear chain of states connected by arrows which are either labelled with the company contributing the required declared capability or ? in the case of potential extensions.

7.3.4 Detailed Case Study

Overhead Systems

The following case study illustrates a case where Intier Ebergassing lack sufficient knowledge to decompose a module into submodules. This means that they do not place themselves as the final assembler for the module and further it is assumed that they had no overall interest in later assembling the module. Thus the initial combined capability states simply that given no initial starting resources an overhead systems module should be produced, ie:

$$[\emptyset] \xrightarrow{?} [OverheadSystemsModule(: WeightvalueFrom(AcceptableRange) : CostvalueFrom(AcceptableRange) : Geometry....]$$

[8] It differs only in the identity of the company providing the decomposition

The constraints here are intended to give an illustration of the purpose of the constraints within constrained attribute sets. At this stage they will be couched in broad terms, thus allowing multiple potential solutions to be considered. In order to keep the presentation efficient in the remainder of this case study and in all subsequent case studies only the abstract resource types will be shown.[9]

In addition to this initial combined capability some utility information is provided which might say that:

- the contract is likely to be for a substantial number of components;
- the contract is with BMW;
- the assembled component must be delivered to Intier Ebergassing.

The final element here is most interesting in that it is not only relevant for suppliers considering if they wish to work with Intier Ebergassing or not, but also relates to logistical considerations. As with the constrained resource types, this utility information will not be explicitly mentioned in later scenarios. The creation of this noticeboard then stimulates the creation of teams to produce the overhead systems. While only the formation of one team is shown here, in general multiple potentially complete teams will be created, along with numerous failed partial teams.

For the purposes of this case study it is assumed that Intier Straubing have expertise in the area of overhead systems but prefer not to assemble them in typical cases. Their capability provider agent is thus programmed to be prepared to supply various components relating to overhead systems to such noticeboards but not, typically, to provide assembly. However when it calculates the utility of the noticeboard it notices that it represents an extremely lucrative opportunity and informs people at Straubing of the existence of this opportunity.

On evaluating the opportunity they decide that in addition to providing several components of the overhead systems module they would also like to assemble it. This is not something they normally offer since they must purchase some new equipment to do this. However they consider the opportunity lucrative enough to support this cost. They thus create a declared capability expressing that they would do the assembly and provide many components - the latter of these by only requiring a few components to be provided. This is:

$$[\emptyset] \xrightarrow{?} [SunRoof, Shelf] \overset{IntierStraubing}{\longrightarrow} [OverheadSystemsModule]$$

Suppliers for the Sunroof and Shelf then propose themselves, potentially each creating a new partial solution. However Intier Straubing limits the number of partial

[9]These should still be considered to represent constrained resource types.

solutions by refusing proposed extensions from suppliers when it has already seen an identical supplier which it prefers. This decision might be based on how close the suppliers are to Straubing or simply on their previous experience of the suppliers. The synergy between the sunroof and shelf suppliers is taken into account by allowing all of the existing suppliers to veto proposed extensions.

As a result of this several complete VOs consisting of Intier Straubing together with Sun Roof and Shelf suppliers form, the best of which is identified through negotiations between the potential teams and Intier Ebergassing. Since Intier Ebergassing and Intier Straubing have worked together successfully before the VO produced from these negotiations is chosen to make the overhead systems.

This scenario provides a useful illustration of a typical mode of operation for noticeboard based VO formation - a single company states that it can supply the desired product if supplied with other modules and thus effectively provides a partial decomposition. The primary advantage of using noticeboard based team formation within this scenario is that it allows team formation to be driven by Intier Straubbings knowledge of how to best decompose an overhead systems module into subcomponents. Not only will this often be more accurate than a centrally stored decomposition but it also can be customised. In this case, Intier Straubing were free to decide that they themselves should supply certain of the subcomponents to be assembled and could intelligently customise the decomposition in response to the specific needs of the business opportunity.

The Door Covering Module

The door covering module case study considers the situation when the company responsible for identifying the business opportunity, Ebergassing here, strongly wishes to assemble the component and can provide a full list of the subcomponents required. This is strongly related to the overhead systems case study above differing mainly in that the initial combined capability already contains the decomposition.

Here it is assumed that Ebergassing can supply the window trimmings, storage cupboards and packing foils internally and know which external supplier they wish to use for the door handles. They therefore need to locate suppliers for the door loudspeaker grill and the injection moulded clasps. To do this they create a noticeboard containing the following initial combined capability:

$$[\emptyset] \xrightarrow{?} [Doorloudspeakergrill, Injectionmoldedclasps]$$
$$\xrightarrow{Ebergassing} [DoorCoveringModule]$$

This noticeboard then inspires suppliers for both the door loudspeaker grill and clasps to propose appropriate declared capabilities. Intier Ebergassing, being already present within the declared capability, controls the number of suppliers considered for each component by only accepting declared capabilities they consider better, or at least nearly as good, as those declared capabilities it has already seen. Once various teams have formed the team considered as best overall is chosen and taken forward.

The usage of the initial combined capability in the case study above is in fact not typical - it would be more typical for Ebergassing to create an basic initial combined capability such as $[\phi] \xrightarrow{?} [DoorCoveringModule]$ and then to extend this combined capability by a declared capability. This would create the same combined capability as that given as initial here, permitting the same teams to form, but the simpler initial combined capability would additionally permit alternative team structures to be investigated.

Indeed the initial capability is typically only elaborated with essential declared capabilities, for instance those that the OEM has mandated must be used within any team considered. It can however also be elaborated in situations where, as here, the agent 'owning' the business opportunity is sure that it wishes any team formed to follow a specific template.

This scenario has been included to illustrate that, while it does not rely on them, noticeboard base team formation can use preprovided decomposition information when it is provided. More traditional techniques would also cope well with this scenario.

The Cockpit Module

In this scenario Intier Ebergassing decide that they are potentially interested in assembling the cockpit module. However they lack the knowledge to decompose the module into individual components and so can not, initially, create an appropriate declared capability. They therefore put an empty initial solution on the noticeboard, but watch the combined capabilities on the noticeboard in case an opportunity for them to act as assemblers occurs. The initial combined capability is:

$$[\emptyset] \xrightarrow{?} [CockpitModule]$$

and the utility information contained:

- that the contract is likely to be for a substantial number of components;
- that it is with BMW;
- that the assembled component must be delivered to Intier Ebergassing.

In addition to this information, in the case of this module BMW had mandated that certain suppliers should be used for some components. These are captured in the constraints given below.

- MandatedSupplier(SoundSystems, MandatedSoundSupplier);
- MandatedSupplier(Switches, MandatedSwitchSupplier);
- MandatedSupplier(Lamps, MandatedLampSupplier).

This noticeboard then catalyses the formation of potential teams to produce the cockpit module. In this case two different ways in which this could happen are illustrated, both of which illustrate how bottom up team formation can support complex, opportunistic team formation. These two case studies thus illustrate the major benefit of using noticeboard based team formation under such circumstances - it allows for the formation of teams which more traditional, decomposition based systems fail to consider as candidates.

Intier Ebergassing as opportunistic assembler

This scenario starts when several distinct suppliers within a geographical cluster examine the initial solution, notice that they could supply components often useful for the construction of cockpits and contribute appropriate declared capabilities. Since these companies are located close together they are happy to combine their individual declared capabilities, and prefer not to work with people outside their group, producing the following set of partial solutions:

$[\emptyset] \xrightarrow{?} [CockpitModule][\emptyset] \overset{AirbagMaker}{\longrightarrow} [Airbag] \xrightarrow{?} [CockpitModule][\emptyset] \overset{AirbagMaker}{\longrightarrow} [Airbag] \overset{Decorator}{\longrightarrow} [Airbag, Decorations] \xrightarrow{?} [CockpitModule]$

and so on up to:

$[\emptyset] \overset{SupplierGrouping}{\longrightarrow} [Airbag, Decorations, Pedals, GearStickCasing, armrestashtray] \xrightarrow{?} [CockpitModule]$

The names of the individual suppliers within the final partial solution are omitted for clarity. At this stage Intier Ebergassing notices that, in combination with the components from the mandated suppliers, they can assemble a complete cockpit and offer to do so. When this proposal is examined in detail, however, the logistical distance between Intier Ebergassing and the individual component suppliers is found to be prohibitive. The proposal from Ebergassing is thus rejected.

However the companies remember Ebergassing's offer to perform the assembly and try to find a way to reduce the overall logistics cost. In doing this they find a company close to the grouping which could assemble their components into 'partial' cockpit. This results in the extension of the combined capability with the following declared capability:

$$[\emptyset] \overset{SupplierGrouping}{\longrightarrow} [Airbag, Decorations, Pedals, GearStickCasing, armrestashtray] \overset{PartialAssembler}{\longrightarrow} [PartialCockpit] \overset{?}{\longrightarrow} [CockpitModule]$$

Since it is unlikely that a precise, pre-agreed term for the specific mixture of components represented by the partial cockpit exists it is interesting to speculate on how it should be represented within the partial solution. Here the term partial cockpit is considered to be a generic term for such submodules and other agents can query the Partial Assembler agents to receive precise details of its composition. However this is not an ideal solution to the problem and some form of flexible ontology support such as that discussed in [77] would probably be ideal, allowing the agents to interactively generate suitable terms to represent such concepts. Identical issues are met in the sub assemblers scenario following this one.

Since the mandated suppliers are not geographically close to the supplier cluster the partial assembly does not include the components from the mandated suppliers. Intier Ebergassing then notice this new combined capability and again propose to perform the final assembly, this time combining the partially assembled cockpit with the components from the mandated suppliers. This produces a complete solution which on closer examination is found to have much better logistical performance than the previous example and is accepted as a final solution by all participants.

$$[\emptyset] \overset{SupplierGrouping}{\longrightarrow} [Airbag, Decorations, Pedals, GearStickCasing, armrestashtray] \overset{PartialAssembler}{\longrightarrow} [PartialCockpit] \overset{MandatedSupplier}{\longrightarrow} [AllComponents] \overset{IntierEbergassing}{\longrightarrow} [CockpitModule]$$

This scenario illustrates how noticeboard team formation allows the choice of which subcomponents to produce to reflect the precise nature of the suppliers of the components. In this case the partial assembling of the cockpit only makes sense due to the geographical location of the supplier for the components being assembled, and would not have been considered by a system using an initial, centralised decomposition. In this scenario noticeboard team formation thus locates a more efficient team than traditional approaches.

Subassemblers

Within this scenario the first extension to the initial combined capability is proposed by a company who produces combined arm rest modules - these include the Arm rest ash tray and the gear stick together with appropriate decoration. In order to do this they need some decoration and moulded parts so the initial partial solution is:

$$[\emptyset] \xrightarrow{?} [Decoration, Mouldedparts] \xrightarrow{ArmRestMaunfacturer} [ArmRestModule] \xrightarrow{?} [CockpitModule]$$

Now several suppliers for decoration and moulded parts propose themselves, creating several combined capabilities. However the arm rest manufacturer controls these partial solutions by refusing extensions from suppliers where it has already seen a supplier it prefers. Finally there is some kind of final choice of decoration and moulded part suppliers made extending this partial solution to:

$$[\emptyset] \xrightarrow{DecorationSupplier} [Decoration] \xrightarrow{PartsSupplier} [Decoration, Mouldedparts] \xrightarrow{ArmRestMaunfacturer} [ArmRestModule] \xrightarrow{?} [CockpitModule]$$

Henceforth this is abbreviated to:

$$[\emptyset] \xrightarrow{ArmRestManufacturer} [ArmRestModule] \xrightarrow{?} [CockpitModule]$$

Following on from this a second company realises that it could manufacture the remainder of a cockpit module apart from the sound system. The submodule representing the remaining components is here termed a front panel module. In order to complete this module the company requires both some decoration and moulded parts. The partial solution is thus extended to:

$$[\emptyset] \xrightarrow{?} [MouldedParts, Decoration] \xrightarrow{FrontPanelMaker} [FrontPanelModule] \xrightarrow{ArmRestMaunfacturer} [FrontPanelModule, ArmRestModule] \xrightarrow{?} [CockpitModule]$$

Once again a choice of moulded part and decoration manufacturer is made through appropriate vetoing of extensions. As with the previous choice the main driver behind this decision is the front panel maker, however in this case there are also the Arm Rest manufacturer and the suppliers of decorations and moulded parts to it to consider.

Each of these suppliers can choose to veto a given proposed supplier. While the front panel maker is very likely to be most involved since they're directly affected by this choice, it is possible that the preferred decoration suppliers might insist on being the exclusive supplier of decoration within the team. However the front panel maker may well prefer some suppliers who they know better or who are closer to them. In this case the agents need to negotiate between themselves to find a mutually agreeable set of suppliers, perhaps involving the arm rest supplier using a less preferable choice of supplier. After a mutually agreed set of decoration and moulded part suppliers are located the partial solution (again abbreviating) is effectively:

$$[\emptyset] \xrightarrow{ArmRestManufacturer} [ArmRestModule] \xrightarrow{FrontPanelManufacturer} [ArmRest, FrontPanel] \xrightarrow{?} [CockpitModule]$$

On examination of this, Intier Ebergassing realise that they can assemble the Cockpit module from the two subcomponents and a sound system (provided by the mandated supplier). This produces another potentially complete solution of:

$$[\emptyset] \xrightarrow{ArmRestManufacturer} [ArmRestModule] \xrightarrow{FrontPanelManufacturer} [ArmRest, FrontPanel] \xrightarrow{SoundSystemMaker} [ArmRest, FrontPanel, SoundSystem] \xrightarrow{IntierEbergassing} [CockpitModule]$$

This solution demonstrates both how the noticeboard system allows for flexible decompositions, and how it allows the choices of subsuppliers, here the decoration and moulded part suppliers, to be made in a manner reflecting the specific requirements of the scenario. It differs from the previous scenario mainly in its motivation - that scenario starts with the collection of local suppliers forming a partial solution which then motivates a local supplier to act as a partial assembler. In this scenario, the solution development is instead directly driven by companies who can assemble major submodules of the module.

7.4 Conclusion

This chapter has presented an evaluation of the noticeboard system proposed within this thesis. In summary it contains:

- a demonstration that the system works at a technical level through the implementation of a proof of concept demonstrator;
- a theoretical discussion of the system's performance, concluding that it is likely to be adequate within the target domain;
- a case study containing a series of scenarios which illustrate situations in which the system supports the intelligent, opportunistic formation of virtual organisations which more traditional decomposition approaches fail to consider.

In summary, this chapter has shown that noticeboard based team formation represents a fully valid approach to supporting VO formation within VBEs and has highlighted situations in which it has substantial benefits over more traditional approaches. The next, and final, chapter presents both a comparison of this system to related techniques and a discussion of which domains best support its use.

Chapter 8

Discussion

The previous chapters of this thesis have presented an approach suitable for supporting VO formation using distributed information, developed this approach into a concrete system and evaluated several aspects of the approach. The current chapter starts with a discussion of how the proposed system compares to certain existing systems. This is followed by a discussion of how the work within this thesis might be profitably extended. Particular attention is given to how case based reasoning might be integrated into the system and the extension of the systems domain of application.

8.1 Related Systems

8.1.1 Systems combining Blackboards and Agents

In recent times many systems aiming to support virtual organisation have been proposed. The current section examines several of these with emphasis on those systems of particular relevance to noticeboard VO formation. In particular it focuses on where these systems share ideas with and differ from the noticeboard based approach proposed within this thesis.

One characteristic feature of noticeboard team formation is its combination of blackboard systems and agents. This combination has previously been used in several systems. For example [86] uses such a system to support large scale planning while also allowing the incorporation of agents who are in themselves quite complex planners.

To do this the paper introduces the idea of a planning cell. Each planning cell concerns a single planning task and contains three principle types of agent: plan servers, a planning cell manager and a set of planning agents. Of these the plan servers serve as passive repositories for plans and partial plans, allowing the agents performing the planning to share information; the planning cell managers are responsible for taking in requests for plans, breaking them into subtasks assigning agents to those subtasks and finally reintegrating their partial plans. Finally the planning agents each provide a form of planning capability. The overall system contains multiple individual planning cells arranged into a hierarchy.

The structure within an individual planning cell is strongly reminiscent of a blackboard system but is fundamentally different from noticeboard team formation for two reasons. The first is that it focuses on generating an abstract plan for a single entity, and does not concern itself with finding actors to fulfil each of its task steps. The second is that the decomposition performed by the planning cell manager in this system refers to decomposing the task of actually generating the plan. So the system uses its blackboard elements to assign certain functional components to tasks such such as constraining the space searched, contributing certain types of process etc. These components then go ahead and generate the potential plan. Noticeboard team formation in contrast uses its blackboard elements to drive the actual planning process. In essence the planning cell system is a meta system for combining existing planners, while noticeboard team formation is itself a form of planner.

Another system which combines agents and blackboard systems is described in [37]. In this paper the authors use their own blackboard/agent hybrid architecture to compare three different approaches to organising teamwork within agents - contract net, cooperative problem solving and shifting matrix management. For the current discussion however, the principal interest of this paper lies in the architecture used. This system represent a slightly different approach to hybridising blackboards and agents in that there is a single, central blackboard and the agents are implemented as private areas on the blackboard together with independent threads of control.

In comparison to a straight mix of blackboards and agents, this architecture keeps the overall system complexity low, and is especially appropriate for running such a system on one computer. Its principle drawback is that it limits the freedom of the agents within the system to engage in personal negotiation. This makes it unsuitable for noticeboard team formation. Perhaps the main contribution of this paper is to highlight that the formal distinction between blackboard and multi agent based systems is not as clear as sometimes thought.

8.1.2 Contract Net Based Approaches

An alternative approach to supporting team formation is the contract net protocol discussed in section 3.3.2. Such systems typically operate by first splitting the overall task into a set of subtasks and then sending requests for quotes out to find suppliers for each of these subtasks.

The distinctive feature of such systems is that they distribute the process of matching the advertised capabilities of suppliers to tasks. In contrast to the noticeboard based approach proposed within this thesis they remain dependent on an initial, centralised decomposition.

The drawbacks of relying on such decompositions have been discussed at length within section 3.2 and contract net in particular has been discussed in section 3.3.2. Notably none of the systems metioned within this section can offer as much support for novel or customised solutions as noticeboard team formation.

One interesting development of the basic contract net process can be found within [58]. The system proposed within this paper goes beyond a basic contract net based approach by allowing each agent, on receipt of a call for proposals, to either directly propose a process to meet it, propose a process on behalf of an existing virtual organisation or to send a call for proposals of its own for some subpart of the proposal which it cannot meet itself.

The ability of agents to form their own virtual organisations and to search for people to cooperate with is implicit within any contract net based system. However it is not often mentioned explicitly and this paper is worthy of note. The paper also mentions techniques for dealing with quality of service provisions, a language for making and replying to calls for proposals and finally a constraint based method for agents to reason about whether they should be proposing to meet a call for proposals.

In contrast to noticeboard based team formation this approach lacks any systematic method of information sharing or dispersal within the system beyond broadcast request for quotes. The drawbacks of this are discussed in section 3.5. In addition the scope for proactivity by agents is limited - they can only react to the tasks offered by offering to fill them themselves, or by decomposing that task than using that decomposition to form a team to fill them. There is no scope for an agent to suggest a way they could perform part of a role, while leaving the remainder of that role open for other contributions. This limits the extent to which the system can utilise the information distributed within the members of the VBE.

Another system with similar principles at heart but more of a webservices focus is the Adlets system proposed by Chrysanthis et al in [15]. The basic input to the system in this paper is a workflow which defines both a set of tasks which a user wishes to be performed and their preferences for how these might be performed. The set of services and information required to enact this workflow are distributed throughout a set of companies.

Each of the tasks within the workflow is then represented by a mobile agent, termed an 'adlet', which is responsible for either directly enacting that task or, if required, locating an appropriate external service to do so. The team formation aspect of this system is quite traditional and works through adlets negotiating with servers to find the service most closely matching its preprovided specification. The primary focus of this paper is on how best to support distributed, dynamic workflow adaptation, something which it does well - when a service provider fails the adlet in control of the task can seek out a new one.

A final contract net based approach is proposed by Oliveira and Rocha in [61]. This system proposes the use of an electronic market system populated by agents providing processes. When a business opportunity is located a market agent is created which has two modules - a goal descriptor module and a virtual organisation selection module. The goal descriptor module contains the original request together with a required set of subtasks to complete it, each of which has an attached set of criteria for evaluating any later bids later located to supply it.

Virtual organisation formation within this system is driven by the market agent sending out a set of requests, company agents sending a set of replies and these being evaluated. Some degree of constraint based negotiation is also used within the system and the company agents can learn from experience to guide their future behaviour within this. This system, like the one presented in this thesis, features agents representing companies and agents taking care of individual business opportunities. In contrast, however, the main areas of decision making are centralised: both the decomposition of the task into a set of subtasks for which suppliers are required and the detailed criteria used to select the suppliers for each task are specified when the market agent is created.

This centralisation contrasts markedly with the approach taken within noticeboard team formation. These differences reflect the different underlying models underpinning the two approaches - the approach taken by Oliveria and Rocha aims to support a scenario in which a central company is locating a set of subsuppliers for a set of subtasks whereas the noticeboard based approach proposed in this thesis tries to support cooperative team formation within a VBE.

8.1.3 Other Approaches

One approach which firmly focuses on supporting the creation of virtual organisations from within virtual breeding environments is proposed within the ECOLEAD project and is detailed in Deliverable 23.2, [19]. ECOLEAD's approach is notable for the extent and detail of its coverage - it covers the entire life cycle from the location of business opportunities to the creation of detailed contracts of how the virtual organisations members will work together. Also notable is its focus on the specific problems of team formation occurring within VBEs. In particular this is reflected by the importance that ECOLEAD places on supporting negotiation between companies within the VBE during team formation.

Despite the emphasis placed on the more collaborative aspects of VO formation, the proposed system within ECOLEAD remains dependent on an initial decomposition followed by a centralised choice of the best suppliers for each role using adverts. Thus, while sharing a basic application domain and similar goals with noticeboard based team formation, it adopts a markedly different approach to providing support for team formation.

Indeed all of the approaches so far studied remain dependent on using a centralised decomposition to generate a set of subtasks from a business opportunity. One approach which attempts to reduce the need for such a centralised decomposition is that given by Ozturk and Gundersen in [64]. This paper attempts to use agents to yield a more flexible mix of decomposition and allocation of tasks within team formation. In addition to proposing a method for support VO formation they also propose a technique for coordinating the operation of VOs using a form of blackboard system known as a task space. Since this thesis does not deal with coordinating the operation of VOs this aspect is not further examined here.

In the paper they assume the existence of multiple different possible decompositions of the task and argue that the best one of these cannot be chosen on an a priori, centralised basis. To avoid this they attempt to "interleave top-down planning with a bottom up process where the possible candidate agents contribute (though implicitly) to the planning". Their proposed system facilitates this by advertising the atomic tasks using a form of contract net based system. When no agent can be located to perform a specific task, that task is noted as unusable. If at least one agent capable of performing the task is located then the system makes an internal choice of the best agent for that task.

Once the set of atomic tasks available within the system has been determined, a decomposition is chosen, which uses only tasks for which suppliers have been located. By delaying the choice of decomposition to this stage the system avoids the expense of choosing an alternative decomposition after the initially chosen decomposition contains tasks for which no supplier exists. This system represents an interesting attempt to meld top down team formation with information gained in a bottom up manner. However the efficiency of doing this by querying every agent within the system to see if they can provide every possible atomic process is questionable - when there are multiple potential decompositions and atomic tasks this could involve a huge amount of wasted communication.

Additionally the system, while more flexible than one which presupposes a specific decomposition, is still tied to only considering those decomposition and task types contained within a centralised repository. Both this problem, and the problem of excess initial communication, could potentially be avoided if the agents joining the team were allowed to build up the decomposition themselves by suggesting processes. In summary, while it represents an interesting attempt to avoid reliance on a single decomposition, this system remains quite different from the system presented within this thesis.

8.2 Further Work

The work developed in this thesis could be extended in many ways. One might be to develop the system into an overall solution applicable within the real world. A brief discussion of the likely problems faced when doing this can be found in Section 7.1.4.

The following section focuses on in depth discussion of two principal areas for further work. The first concerns the idea of reusing the information that accumulates during the operation of a noticeboard. The second concerns a detailed consideration of precisely which features a domain should have in order to profitably support the application of the technique proposed within this thesis. This facilitates a discussion of how effective it might be for supporting some areas, including webservice composition.

8.2.1 Learning from previous solutions

During the process of forming virtual organisations for a specific business opportunity notice agents have many solutions, both partial and complete, added to them. These represent a potentially valuable resource for guiding the formation of future virtual organisations and it is wasteful to discard them once the noticeboards business opportunity is no longer open for bid.

The idea of reusing previously created planning information is studied within the field of case-based planning. Case-based planning is an approach to planning problems[1] which focuses on the reuse of previously designed plans where possible. Where no previous plan fits exactly into the current situation the best fitting plan is adapted. This is an attractive idea and has seen much interest, a survey of which can be found in, [76].

Despite initial impressions, case-based planning is far from guaranteed to be more efficient than replanning from scratch. In [57] the authors examine the theoretical efficiency of cased-based planning and show that it is, in general, not possible to prove it more efficient than traditional planning. This paper also contains a practical survey of the efficiency of case-based planners in two domains, showing that the two main obstacles to efficient case-based planning are identifying candidate plans for

[1]The basic idea of reusing previous solutions can be applied within other domains

reuse and adapting plans to fit the current situation when they don't fit well.

Indeed a basic approach to case-based reasoning would not fit well with the distributed control of noticeboard systems. However a system whereby notice board agents use promising combined capabilities they have seen to generate large scale declared capabilities and then suggest these derived declared capabilities to other noticeboards would do so. Such an approach would differ somewhat from traditional case-based reasoning in that the plan reuse would focus on complementing rather than *replacing* the planning within the main noticeboard system. The main obstacles to making plan reuse work within a noticeboard system are:

- identifying which combined capabilities on a noticeboard represent the most useful information sources;
- representing this information in a form facilitating its reuse.

A balance must be struck when identifying which combined capabilities to reuse. If a noticeboard agent were to attempt to reuse every declared capability on it then it would simply duplicate the existing behaviour of the agents owning those declared capabilities. Indeed it would do so in a less intelligent manner - it does not know when that declared capability should best be contributed. This would be of little overall benefit to the system. On the other hand if too little information is reused there is a risk of potentially useful information being lost.

This need for balance suggests that the noticeboard agents should only remember combined capabilities formed from the declared capabilities of several companies. In this way, companies could be spared the effort of working out how their declared capabilities fit together from scratch every time. Further it could reduce the overall number of combined capabilities created - if the combination is A-B-C you won't also get A-?-C, ?-B-C etc.

As well as extracting only those elements containing substantial amounts of information, ideally the information extracted should be capable of likely reuse. When predicting the reuse probability of such combined declared capabilities a guide might be to take information from those combined capabilities which later developed into complete combined capabilities. Indeed it would be reasonable to only reuse complete combined capabilities - *ie* if a noticeboard were to form to build X from Y it would examine future noticeboards to see if they also needed to make X and if they did would suggest the reuse of some of its complete combined capabilities. It does however seem possible to do better than this by additionally reusing appropriately large 'chunks'. Only practical testing would fully resolve this issue.

This leaves the second issue - the form in which this information should be represented. The easiest option within the current system structure would be to extract the information as declared capabilities with multiple agents owning them. This is straightforward to do: for a complete combined capability it simply involves aggregating the agents, setting the input as the input, the output as the output and removing the intermediate states. For example:

$[\emptyset] \overset{ArmRestManufacturer}{\longrightarrow} [ArmRestModule] \overset{FrontPanelManufacturer}{\longrightarrow}$
$[ArmRest, FrontPanel] \overset{SoundSystemMaker}{\longrightarrow}$
$[ArmRest, FrontPanel, SoundSystem] \overset{IntierEbergassing}{\longrightarrow} [CockpitModule]$

would become the declared capability:

$[\emptyset] \overset{SoundSystemMaker,ArmRestManufacturer,FrontPanelManufacturer,IntierEbergassing}{\longrightarrow} [CockpitModule]$

In a similar way large parts within the combined capabilities could be extracted. For instance above one might also extract :

$[\emptyset] \overset{SoundSystemMaker,ArmRestManufacturer,FrontPanelManufacturer}{\longrightarrow}$
$[ArmRest, FrontPanel, SoundSystem]$

as a declared capability. Doing this might entail combining certain states unrelated by the partial order but, in line with the discussion in Section 5.3.5, this can be done safely. While this undoubtedly represents a viable approach to solution reuse in a noticeboard system it has some potential drawbacks:

- since all of the information in the constrained resources is retained, matches will only be flagged when *exactly* the same resources are needed;
- the declared capability extracted is tied to the set of agents who originally contributed to it.

The first of these will clearly reduce the chance of applying the extracted declared capability to new situations. While this might be mitigated by using abstract resource types rather than concrete ones this is not really possible when using unmodified declared capabilities. The problem with the second item is that agents are allowed to tailor their level of cooperation in response to the details of the business opportunity. This can have several effects:

- some of the agents in the group may view the new BO as less attractive and not wish to contribute in the same way;
- certain agents who were not willing to contribute to the original opportunity might be willing to contribute to the new one, leaving the old team suboptimal;
- applying the composite declared capability to larger projects may entail working with agents with whom people in the group do not wish to collaborate;

- the basic transformation represented by the composite declared capability - making X from Y - might be represented by multiple different sets of agents on the noticeboard.

The first three issues here principally cause problems by reducing the probability of reusing the composite declared capability. In contrast the fourth might cause problems by requiring multiple composite capabilities with identical structures to be remembered.

An alternative approach to storing the information would be to extract the abstract structure of the composite declared capability but forget the details of both the constrained resource types and the agents involved. This would produce abstract decompositions which could be given to other noticeboards as suggestive guides. For example the complete combined capability above might then become:[2] $[\emptyset] \xrightarrow{?} [ArmRestModule] \xrightarrow{?} [ArmRest, FrontPanel] \xrightarrow{?}$

$[ArmRest, FrontPanel, SoundSystem] \xrightarrow{?} [CockpitModule]$

This form of abstract combined capability can not be directly used as a declared capability. Instead it would have to be placed on the noticeboard in a position where it suggests a possible decomposition to any process provider agents wishing to contribute to the noticeboard. Perhaps the best way to facilitate this would be for noticeboards to contain an additional layer in which potential abstract structures for solutions are examined, with the actual assignment of agents to these happening on a lower layer. This, together with the whole area of solution reuse, would provide an interesting area for further research.

8.2.2 Application to Alternative Concrete Domains

It is natural to consider where else the system proposed in this thesis might be profitably applied. This is done by considering the domain characteristics originally introduced in Section 2.1.2. Particular consideration is given to how the fourth domain requirement, namely that a typical business opportunity consists of a request to manufacture a certain amount of some concrete product, can be rephrased in a more general way. The implications of these more general domain requirements for the applicability of noticeboard based team formation within other domains are then considered.

Assumed Domain attributes

The problem domain within this thesis is defined by a set of four properties, introduced in Section 2.1.2. The fourth of these requirements was that "a typical business

[2]The resources in this case were already being shown as abstract for simplicities sake

opportunity consists of a request to manufacture a certain amount of some concrete product". The purpose of this requirement placed on the domain of this thesis was to give the process provider agents a 'hook' on which they could base their reasoning about which of their processes are likely to represent useful contributions. The discussion in Section 6.4.4 highlighted how this provides a highly desirable boost to the systems overall efficiency. This is similar to the benefits derived from the distance metric used in the A* algorithm for travel planning, discussed in Section 3.2.

The statement of this condition given in Section 2.1.2 does not represent the most general statement possible. This section considers exactly what the minimal set of domain attributes to support noticeboard based team formation are. The following statement rephrases the fourth requirement from Section 2.1.2 in a fully general fashion:

- given a task T, a collection of subtasks (ST_i) and another subtask ST_j it must be possible to decide if it is likely that $(T \setminus (\bigcup_i ST_i \cup ST_j)) < (T \setminus \bigcup_i ST_i)$ without reference to an externally provided decomposition of T into subtasks.

The word 'might' here is important - since the agents do not work with an explicit break down of T into subtasks they can never be sure that the process they are contributing is useful. They should however only propose declared capabilities which they consider have a good chance of proving to be useful.

As discussed in Section 6.4.4, within the context of a noticeboard system, this knowledge takes the form of process provider agents knowing the types of projects in which their capabilities are typically used. The ability of the agents to use such knowledge is domain dependent, and only detailed examination would show how well a specific domain supports it. In general terms it is expected that capabilities grounded in real world services will provide the basis that this reasoning requires, while capabilities based on abstract software components or mathematical functions may prove more problematic.

So far the discussion has focused on the problem of which domains noticeboard team formation can operate efficiently. As well as identifying those domains in which it is designed to operate efficiently, it is interesting to compare its performance in classes domain where more traditional approaches might be expected to work better.

In order to do this consider a task T which admits only a single decomposition into three subtasks T_1,T_2 and T_3. Here it is assumed that the central element of the Contract Net system has knowledge of this decomposition and that the noticeboard system is provided with an initial solution specifying that these three specific tasks should be used.

In order to form a team a Contract Net system would first break down T into T_1,T_2 and T_3 and then contact every relevant agent within the VBE to see if they wish to supply T_1,T_2 or T_3, before choosing the individually best suppliers. Here $C(T_i)$ represents a choice of supplier to supply T_i and $Value(C(T_i))$ represent the value of making that choice. Noticeboard based team formation would operate through every agent within the VBE examining the initial solution containing the three subtasks T_1,T_2 and T_3, deciding if they wished to contribute to them and proposing to join any partial formed teams (ie ones where agents have already been assigned to other subtasks) they would like to extend. The ability of agents within a partial solution to refuse to accept proposals would ensure that the teams forming contain agents who are both willing to work together and, given intelligent capability provider agents, think they can work together effectively.

The overall performance of the teams assigned by the two approaches is then: $Value(C(T_1), C(T_2), C(T_3)) =$

$Value(C(T_1)) + Value(C(T_2)) + Value(C(T_3)) + F(C(T_1), C(T_2), C(T_3))$, ie the sum of the individual values of the suppliers and a correction factor based on their interaction. The basic assumption made within approaches such as contract net which assign the individually optimal suppliers to each task is that $Value(C(T_1)) + Value(C(T_2)) + Value(C(T_3)) \gg F(C(T_1), C(T_2), C(T_3))$. In such a situation the effect of the $F(C(T_1), C(T_2), C(T_3))$ term can be safely ignored.

In scenarios where a company directly procures suppliers to deliver certain components this assumptions holds, choosing the best value supplier for each raw material works well and the approach taken within Contract Net systems is more computationally efficient than the Noticeboard based approach.

In contrast within other types projects the assumption breaks down. For example within an automotive development project the suppliers must cooperate over a sustained period to develop prototype products and much of the overall cost of the project comes from the cost of the logistics, something which is very much dependent on the interactions between the individual choices of suppliers.

The basic Contract Net approach can be adapted to work well in such domains. Instead of locating the single best supplier for each task you find a set of the most promising candidate team members for every task and pick the best combination from amongst the candidate teams thus produced. Several papers have suggested mathematical techniques which cater for the combination value term, notable examples of which include [80] and [27].

Any such approach remains reliant on its ability to centrally calculate the value of the $F(C(T_1), C(T_2), C(T_3))$ term. However it seems unlikely that the companies will wish to make public the information needed to fully cater for the interactions between them. If such matters are highly important then noticeboard based approaches are likely to prove as, or perhaps more, efficient.

The final point is that any move away from the pure assignment of the best supplier to each role reduces the performance benefits that Contract Net enjoys over noticeboard based systems. This effect becomes more noticeable as the importance of the $F(C(T_1), C(T_2), C(T_3))$ term increases, forcing more candidate suppliers to be considered for every role. Indeed if $Value(C(T_1))+Value(C(T_2))+Value(C(T_3)) << F(C(T_1), C(T_2), C(T_3))$ then the approach within noticeboard team formation is probably more efficient than that used within Contract Net.

The above discussion suggests the following conditions for noticeboard team formation to be useful. Either of these conditions holding is sufficient to motivate considering its use, while both holding is a strong indication that it might prove effective: [3]

- $\sum_i Value(C(T_i)) \not\gg F(C(T_1), C(T_2), C(T_3)....)$,
- for any given high level task, T there are multiple valid sets of subtasks (T_i) such that $\sum_i T_i = T$.

In order to judge the suitability of noticeboard based team formation in a domain these criteria should be considered in conjunction with the requirements from Section 2.1.2. To recap these, with the modification discussed above, are that:

- team formation happens over a long time period;
- the output of the team formation process does not need to be directly executable;
- the individual business opportunities have sufficient worth to motivate companies to offer customised processes,
- given a task T, a collection of subtasks (ST_i) and another subtask ST_j it must be possible to decide if it is likely that $(T \setminus (\bigcup_i ST_i \cup ST_j)) < (T \setminus \bigcup_i ST_i)$ without reference to an externally provided decomposition of T into subtasks.

Alternative Domains

One problem domain with a distinctively different character to supporting the formation of virtual organisations within an automotive VBE is that of composing web services. This area of research has been very active and both [66] and [67] provide surveys of the current state of the art and of the prominent challenges expected to be faced in the future.

[3]The statements here are concern the general case of a sum over n different roles rather than specifically three as above

The basic problem addressed in this field is that of immediately selecting and combining a set of software services to produce an executable software service which meets the needs of the customer. A commonly quoted example, seen in for instance [25], is that of arranging a holiday to a given location. This might involve selecting and coordinating individual services which book flights, hotels and events at the destination.

The first criterion regarding the application of noticeboard team formation has an interesting status within this domain - it is crucial that the individual services chosen within each task can combine into a single executable service but the cost of the combined execution of a working set of services is equivalent to the sum of their individual costs.

Thus bottom up team formation is only useful with respect to this criteria if the decision as to whether a set of services can run in combination is better taken on a local level rather than by comparing published process descriptions. This suggests that the Noticeboard team formation will be most useful in situations where each service can be enacted in multiple ways rather than when supporting the direct compositions of fixed, software components.

The second criteria for considering noticeboard team formation holds - there is rarely just one specific way to combine webservices to produce a specific objective. For instance in the example above there may be a service which will book both hotels and flights at a cheaper combined rate than that offered by two combined services.

It is thus worth considering how well Noticeboard team formation might operate in this domain. This critically depends on the ability of the agents to determine when their individual declared capabilities might be useful contributions to a specific combined capability. The importance of this is highlighted within web service formation by both the very large numbers of declared capabilities potentially considered and the very short time span allocated for service formation.

A distinction must be drawn between the two, quite distinct, problem areas to which web service composition has been applied. One is seen in [25], where the composition of web services is used in order to enact a real world action such as booking a combination of the journey/hotel/entertainment required by someone for a given trip.

The other main problem area is automatically combining web services in order to create large pieces of software. An extreme example of this approach, which envisages dynamically creating pieces of software from a composition of services every time the user wishes to use it, is reported in [7]. It is hoped that this approach will make maintenance and upgrading easier by enabling each component to be individually upgraded as and when required.

A related area of research is that of grid based computing. The goal of this field, to which an introduction can be found in [35], is to enable large computing tasks to run on automatically assembled groups of computers each of which contributes certain amounts of computing resource. A related area is that of dynamically composing software components.

Within such areas the compositions must be generated instantly and on demand. Furthermore there is no obvious metric to use for judging the utility of contributions. The performance of noticeboard based composition is therefore unlikely to prove adequate.

In contrast the system has promise for supporting the composition of web services which are grounded in real world services. The time available for team formation in such domains, while typically lower than that seen within automotive VBEs, is none the less relatively relaxed. In addition it is easy to see how the ability of noticeboard based team formation to support the proposal of customised services might prove of genuine use - service providers might wish to offer special services for either lucrative requests and regular customers.

In addition the grounding of the services in the real world offers a substantial data source which the companies can use to determine their likely utility for a specific task. Consider two companies, one who offers a parcel transport service and one who offers a small fixed set of web services. The parcel transport company could easily determine the utility of its services for specific requests and potentially customise their service for larger requests by purchasing new containers or even trucks whereas the company providing a fixed set of webservices has no such options.

That there is potential for the development of new approaches for the composition of web services based on concrete services is highlighted on page 13 of [66] where, under the heading 'Adaptive and Emergent service compositions', the authors discuss the formation of web service compositions between organisations who have not or cannot make their services explicitly available as fixed services on the web. Indeed it seems that an adaptation of the techniques proposed within this thesis might be ideally suited for solving such problems.

In summary there is limited scope to apply noticeboard based techniques to the automatic composition of pieces of software. However the basic approach shows promise for supporting the composition of web services grounded in concrete services. In order to do this certain aspects of the detailed system presented in this thesis such as the process representation used would require customisation to better represent the specific details of the domain addressed.

8.3 Summary

This thesis started by investigating the feasibility of using the information distributed throughout the member companies of a VBE to support VO formation without using centralised information. The approach is shown to be viable and an abstract approach to doing so developed. This approach is further developed within the context of certain classes of VBEs - typified by automotive clusters - to produce a detailed system. The technical validity of this system is demonstrated through the implementation of a proof of concept demonstrator.

The potential utility of the approach is initially shown by an examination of existing team formation systems which demonstrate that they were ill suited to the specific demands of the domain. This is highlighted by contrasting their performance with that of the proposed approach within specific scenarios. This can be seen in Section 7.3.

Having presented an approach and validated both its technical validity and worth, the ability of this system to be applied within other domains is discussed in Section 8.2.2. The conclusion is that, while the system could technically work within a broad spectrum of domains, the approach is best suited to domains in which companies contribute substantial, 'real world' processes to large scale projects. In particular the approach is much better suited to composing web services if they represent the provision of a concrete service by a real world company rather than simple software components.

While it would be undoubtedly interesting to investigate and implement the features outlined in Section 8.2, the system developed within this thesis would perhaps most benefit from being developed towards potential usage within real world scenarios. In addition to those obstacles foreseen and discussed in Section 7.1.4, this process would entail the further development of such features of the system as the ontology used to represent the products produced and the constraints placed on initial orders. More important than this would be to answer the major remaining question about the approach - will real world companies actually find the approach useful?

8.4 Conclusion

The primary intended impact of this thesis is the demonstration that it is possible to supporting virtual organisation using the knowledge distributed throughout the member companies of a VBE. Indeed this thesis has demonstrated that this approach can be superior to approaches relying on centralised information under certain conditions.

A further major motivation behind the work in this thesis was the realisation that virtual organisation formation can be seen a cooperative process. This contrasts with much of the work in the area which often seems to be implicitly assume that VO formation is driven by a company selecting subsuppliers to provide it with certain tasks. While this basic insight has been acknowledged by projects such as ECOLEAD, as yet it has had little impact on the approaches used to support virtual organisation formation and the approach within this thesis highlights how it could be incorporated.

A final major impact of this work is to challenge the idea that supporting the formation of virtual organisations is broadly equivalent to supporting the formation of compositions of members of a fixed set of services or workflows. While the formal validity offered by such approaches is attractive and indeed crucial for the automated composition of software, it should be recognised that they deal imperfectly with the problems met when trying to support the formation of teams composed of companies offering services representing real processes. Indeed the single most notable feature of the noticeboard based approach is its ability to support companies in intelligently varying the set of processes that they offer in response to the specific nature of the business opportunity.

Bibliography

[1] Crosswork European Project. FP6 Strep Research Project 2004-2006. Public deliverable to work packages 1.1 and 1.2. details available on request., 2004.

[2] H. Afsarmanesh and L.M. Camarinha-Matos. *Collaborative Networks and Their Breeding Environments*, chapter A framework for management of virtual organizations breeding environments, pages 35–48. Springer, 2005.

[3] Crosswork European Project. FP6 Strep Research Project 2004-2006. Details available on request.

[4] F Bacchus and Q Yang. The expected value of hierarchical problem-solving. In *AAAI-92 proceedings : tenth national conference on artificial intelligence*, pages 369–374. American association of artificial intelligence, 1992. 10th national conf on artificial intelligence (aaai-92), san jose, ca, jul 12-16, 1992.

[5] Christopher Barnatt. Virtual organization in the small business sector: The case of cavendish management resources. *International Small Business Journal*, 15(4):36–47, 1997.

[6] Fabio Luigi Bellifemine, Giovanni Caire, and Dominic Greenwood. *Developing Multi-Agent Systems with JADE.* Wiley, 2007.

[7] K. Bennett, M. Munro, J. Xu, N. Gold, P. Layzell, N. Mehandjiev, D. Budgen, and P. Brereton. Prototype implementations of an architectural model for service-based flexible software. In *HICSS '02: Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)-Volume 3*, page 76.2, Washington, DC, USA, 2002. IEEE Computer Society.

[8] Avrim Blum and Merrick Furst. Fast planning through planning graph analysis. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence(IJCAI 95)*, pages 1636–1642, 1995.

[9] Stephen J Brams and Alan D. Talyor. *Fair Division: From Cake-Cutting to Dispute Resolution.* Cambridge University Press, 1996.

[10] Christopher H. Brooks and Edmund H. Durfee. Congregation formation in multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 7(1-2):145–170, 2003.

[11] Sven Brueckner and Richard Gerth. Applying distributed adaptive optimization to digital car body development. In Sven Brueckner, Giovanna Di Marzo Serugendo, Anthony Karageorgos, and Radhika Nagpal, editors, *Engineering Self-Organising Systems*, volume 3464 of *Lecture Notes in Computer Science*, pages 267–279. Springer, 2004.

[12] Luis M. Camarinha-Matos and Hamideh Afsarmanesh. Elements of a base ve infrastructure. *Computers in Industry*, 51:139–163, 2003.

[13] Peter H. Carstensen and Kjeld Schmidt. Computer supported cooperative work: New challenges to systems design. In *In K. Itoh (Ed.), Handbook of Human Factors*, pages 619–636, 1999.

[14] Norman Carver and Victor Lesser. The evolution of blackboard control architectures. *Expert systems with applications*, 7(1):1–30, Jan-Mar 1994.

[15] Panos K. Chrysanthis, Sujata Banerjee, and Shi-Kuo Chang. Establishing virtual enterprises by means of mobile agents. In *RIDE*, pages 116–123, 1999.

[16] Bradley J. Clement and Edmund H. Durfee. Theory for coordinating concurrent hierarchical planning agents using summary information. In *Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, pages 495–502. American Association for Artificial Intelligence, 1999.

[17] Ecolead Consortium. D22.1 key components, features and operating principles of the virtual organisation breeding environment. http://www.ve-forum.org/projects/284/Deliverables/d21_1_vbe.pdf, 2005.

[18] Ecolead Consortium. D23.1 requirements and mechanisms for vo planning and launching. http://www.ve-forum.org/Projects/284/Deliverables/D231VOcreation.pdf, 2005.

[19] Ecolead Consortium. D23.2 specification of vo formation support tools. http://www.ve-forum.org/Projects/284/Deliverables/D23.2_Final.pdf, 2006.

[20] Daniel D. Corkill. Collaborating software: Blackboard and multi - agent systems and the furture. In *In Proceedings of the International Lisp Conference*, 2003.

[21] Iain D. Craig. From blackboards to agents. In *In Online Proceedings of the VIM Project Spring Workshop on Collaboration Between Human and Artificial Societies (Lanjar*, pages 2000–08, 1998.

[22] Mathijs de Weerdt; Andre Bos; Hans Tonino; and Cees Witteveen. A resource logic for multi-agent plan merging. *Annals of Mathematics and Artificial Intelligence; volume 37; Issue 1-2*, pages 93 – 130, 2003.

[23] Marie E. desJardins, Edmund H. Durfee, Charles L Ortiz, Jr., and Michael Wolverton. A survey of research in distributed, continual planning. *The AI Magazine*, 20(4):13–22, 1999.

[24] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, 26(1):29–41, 1996.

[25] David Martin (editor) et al. Owl-s: Semantic markup for web services. w3c member submission 22 november 2004. http://www.w3.org/submission/owl-s/.

[26] Rik Eshuis, Paul Grefen, and Sven Till. *Business Process Management*, volume 4102/2006, chapter Structured Service Composition, pages 97–112. Springer Berlin/Heidelberg, 2006.

[27] Toni Jarimo et all. *Hierarchical Multi-Attribute Decision Support Approach to Virtual Organization Creation.* 2005.

[28] Wolfgang Faisst. Information technology as an enabler of virtual enterprises: A life-cycle-oriented description. *Proceedings of the European Conference on Virtual Enterprises and Networked Solutions*, 1997.

[29] FIPA. Fipa abstract agent specfication. http://www.fipa.org/specs/fipa00001/index.html.

[30] FIPA. Fipa acl message structure specification. http://www.fipa.org/specs/fipa00061/SC00061G.html.

[31] FIPA. Fipa query interaction protocol specification. http://www.fipa.org/specs/fipa00027/SC00027H.html.

[32] FIPA. Fipa sl content language specification. http://www.fipa.org/specs/fipa00008/SC00008I.html.

[33] FIPA. Fipa subscribe interaction protocol specification. http://www.fipa.org/specs/fipa00035/SC00035H.html.

[34] FIPA. The foundation of physical agents. http://www.fipa.org/.

[35] Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *Int. J. High Perform. Comput. Appl.*, 15(3):200–222, 2001.

[36] Java Agent DEvelopment framework. http://jade.cselt.it/.

[37] Li G. and Hopgood A.A.and Weller M.J. Shifting matrix management: a model for multi-agent cooperation. *Engineering Applications of Artificial Intelligence*, 16(3):191–201, 2003.

[38] William V. Gehrlien. Condorcet's paradox. *Theory and Decision*, 15:2:161–198, 1983.

[39] David Gelernter. Generative communication in linda. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 7(1):80–112, 1985.

[40] Thomas R. Gruber and Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5:199–220, 1993.

[41] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4:100–107, 1968.

[42] Barbara Hayes-Roth. A blackboard architecture for control. *Artif. Intell.*, 26(3):251–321, 1985.

[43] Barbara Hayes-Roth and Frederick Hayes-Roth Perrault. A cognitive model of planning. *Cognitive Science: A Multidisciplinary Journal*, 3(4):275–310, 1979.

[44] Durward K Sobek II, Allen C. Ward, and Jeffrey K. Liker. Toyota's principles of set-based concurrent engineering. *Sloan Management Review*, pages 67–83, 1999.

[45] M. Ikeda and Y. Nakagawa. Two ways of modularization strategy in japan:toyota - honda vs. nissan - mazda. In *Proceedings of Ninth GERPISA International Colloquium*, 2001.

[46] N. R. Jennings and M. Wooldridge. Applying agent technology. *Applied Artificial Intelligence special issue on Intelligent Agents and Multi-Agent Systems*, 9:357–369, 1995.

[47] Craig A. Knoblock. Automatically generating abstractions for planning. *Artificial Intelligence*, 68(2):243–302, 1994.

[48] Sven O. Krumke, Maarten Lipmann, Willem E. de Paepe, Diana Poensgen, Jorg Rambau, Leen Stougie, and Gerhard J. Woeginger. How to cut a cake almost fairly. In *SODA '02: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 263–264, Philadelphia, PA, USA, 2002. Society for Industrial and Applied Mathematics.

[49] S.E. Lander. Distributed Search and Conflict Management Among Heterogeneous Reusable Agents. *Ph.D. thesis, University of Massachusetts, Amherst*, May 1994.

[50] Susan E. Lander, Daniel D. Corkill, and Scott M. Staley. Designing Integrated Engineering Environments: Blackboard-Based Integration of Design and Analysis Tools. *Concurrent Engineering*, 4(1):59–71, 1996.

[51] Yuliang Li, Xinyu Shao, Peigen Li, and Qiong Liu. Design and implementation of a process-oriented intelligent collaborative product design system. *Comput. Ind.*, 53(2):205–229, 2004.

[52] David Martin, Mark Burstein, Drew McDermott, Sheila McIlraith, Massimo Paolucci, Katia Sycara, Deborah L. McGuinness, Evren Sirin, and Naveen Srinivasan. Bringing semantics to web services with OWL-S. *World wide web-internet and web information systems*, 10(3):243–277, SEP 2007.

[53] David Martin, Massimo Paolucci, Sheila McIlraith, Mark Burstein, Drew McDermott, Deborah McGuinness, Bijan Parsia, Terry Payne, Marta Sabou, Monika Solanki, Naveen Srinivasan, and Katia Sycara. *Semantic Web Services and Web Process Composition*, chapter Bringing Semantics to Web Services: The OWL-S Approach, pages 26–42. Springer Berlin / Heidelberg, 2005.

[54] Deborah L. McGuiness and Frank van Harmelen. Owl web ontology language overview. http://www.w3org/TR/owl-features.

[55] Marjan Mernik, Jan Heering, and Anthony M. Sloane. When and how to develop domain-specific languages. *ACM Comput. Surv.*, 37(4):316–344, 2005.

[56] Bonnie A. Nardi. *A small matter of programming: perspectives on end user computing*. MIT Press, 1993.

[57] Bernhard Nebel and Jana Koehler. Plan reuse versus plan generation: A theoretical and empirical analysis. Technical Report RR-93-33, 1993.

[58] TJ Norman, A Preece, S Chalmers, NR Jennings, M Luck, VD Dang, TD Nguyen, V Deora, JJ Shao, WA Gray, and NJ Fiddian. CONOISE: Agent-based formation of virtual organisations. In Coenen, F and Preece, A and Macintosh, A, editor, *Research and development in intelligent systems xx*, BCS conference series, pages 353–366. SGAI, Springer-Verlag London ltd, 2004. 23rd International Conference on Innovative Techniques and Applications of Artificial Intelligence, Cambridge, ENGLAND, DEC, 2003.

[59] The oxford english dictionary. 2nd ed. 1989, oed online, 2000.

[60] SMEs Undertaking Design of Dynamic Ecosystem Networks (SUDDEN). FP6 European Research Project. 2006-2009. http://www.sudden.biz.

[61] Eugenio Oliveira and Ana Paula Rocha. Agents advanced features for negotiation in electronic commerce and virtual organisations formation processes. In *Agent Mediated Electronic Commerce, The European AgentLink Perspective.*, pages 78–97, London, UK, 2001. Springer-Verlag.

[62] Andrea Omicini and Franco Zambonelli. Coordination for internet application devlopment. *Autonomous Agents and Multi-Agent systems*, 2(3):251–269, 1999.

[63] Alan O'Sullivan. Why tense, unstable, and diverse relations are inherent in co-designing with suppliers: an aerospace case study. *Industrial and Corporate change*, 15(2):221–250, 2006.

[64] P. Ozturk and O.E. Gundersen. A combined top-down and bottom-up approach to integrated task-decomposition and allocation. In *Proceedings of 2004 International Conference on Machine Learning and Cybernetics*, volume 1, pages 163–168, 2004.

[65] Massimo Paolucci, Julien Soudry, Naveen Srinivasan, and Katia Sycara. *Extending Web Services Technologies*, volume 13, pages 79–98. Springer US, 2004.

[66] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-oriented computing research roadmap. In Francisco Curbera, Bernd J. Krämer, and Mike P. Papazoglou, editors, *Service Oriented Computing(SOC), 15.-18. November 2005*, volume 05462 of *Dagstuhl Seminar Proceedings*. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2006.

[67] Michael Papazoglou. *Web Services: Principles and Technology.* Prentice Hall, 2007.

[68] Gian Pietro Picco, Amy L. Murphy, and Gruia Catalin Roman. LIME:linda meets mobility. In *Internation Conference on Software Engineering*, pages 368–377, 1999.

[69] Fred Philips Roger D. Martin and Michael D. Shields. Cases in strategic systems auditing, reiter automotive north america inc., 2000.

[70] Kamel Rouibah. Managing concurrent engineering across company borders: A case study. In *HICSS '03: Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track1*, page 9.2, Washington, DC, USA, 2003. IEEE Computer Society.

[71] Earl D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5:115–135, 1974.

[72] Tuomas W. Sandholm. *Distributed Rational Decision Making*, chapter 5. MIT Press, 1999.

[73] S Sen. Believing others: Pros and cons. *Artificial Intelligence*, 142(2):179–203, DEC 2002. 4th International Conference on MultiAgent Systems, Boston, Massachusetts, JUL 10-12, 2000.

[74] Conference Chairs Barry Smith and Christopher Welty. Fois introduction: Ontology—towards a new synthesis. In *FOIS '01: Proceedings of the international conference on Formal Ontology in Information Systems*, New York, NY, USA, 2001. ACM Press.

[75] RG Smith. The Contract Net protocol - high-level communication and control in a distributed problem solver. *IEEE transactions on computers*, 29(12):1104–1113, 1980.

[76] Luca Spalzzi. A survey on case-based planning. *Artif. Intell. Rev.*, 16(1):3–36, 2001.

[77] Iain Duncan Stalker, Nikolay Mehandjiev, and Martin Carpenter. *Devolved Ontology for Smart Applications*, volume 4604/2007, pages 360–373. Springer Berlin / Heidelberg, 2007.

[78] Malcolm J. A. Strens and Neil Windelinckx. Combining planning with reinforcement learning for multi-robot task allocation. In *Adaptive Agents and Multi-Agent Systems*, pages 260–274, 2005.

[79] K. Sycara, M. Paolucci, J. Soudry, and Naveen Srinivasan. Dynamic discovery and coordination of agent-based semantic web services. *IEEE Internet Computing*, 8(3):66–73, 2004.

[80] Srinivas Talluri, R. C. Baker, and Joseph Sarkis. A framework for designing efficient value chain networks. *Internation journal of production economics*, 62:133–144, 1999.

[81] Milind Tambe and Weixiong Zhang. Towards flexible teamwork in persistent teams: Extended report. *Autonomous Agents and Multi-Agent Systems*, 3(2):159–183, 2000.

[82] Gek Woo Tan, C.C. Hayes, and M. Shaw. An intelligent-agent framework for concurrent product design and planning. *IEEE Transactions on Engineering Management*, 43:297–306, 1996.

[83] Arie van Deursen, Paul Klint, and Joost Visser. Domain-specific languages: an annotated bibliography. *SIGPLAN Not.*, 35(6):26–36, 2000.

[84] Gerhard Weiss, editor. *Multiagent systems - a modern approach to distributed artifical intelligence.* MIT Press, 1999.

[85] Daniel S. Weld. Recent advances in ai planning. *AI Magazine*, 20(2):93–123, 1999.

[86] David E. Wilkins and Karen L. Myers. A multiagent planning architecture. In *Artificial Intelligence Planning Systems*, pages 154–163, 1998.

[87] WSMO. Ongoing research project, main web page with details:. http://www.wsmo.org/.