DOCTOR OF PHILOSOPHY

Customisable transaction support for web services

Neugebauer, Rose T.

Award date:
2012

Awarding institution:
Coventry University

Link to publication

Customisable Transaction Support for Web Services

Rose T. Neugebauer

A thesis submitted in partial fulfilment of University's requirements for Degree of Doctor of Philosophy

2012

Acknowledgement

This work would not have been realised if not for the support I got from my Director of Studies, Professor Anne James. I should like to sincerely thank her for the direction, guidance and encouragement she unlimitedly gave me. I highly appreciate the opportunities she gave me to present and discuss the ideas relating to this work through publications.

Additionally I would like to express my gratitude and admiration to Dr R. Iqabal, Rafal, Fahmida who gave me technical comments, assistance and support during the work.

I am incalculably indebted to the love and support my husband Frank Neugebauer gave me over the years. Most of all I am grateful to my beloved Mum, sister Tebogo Levita and brother Okeditse Khachana; this could not have been possible without their support and encouragement.

Lists of Publications

- Khachana, T., James, A. and Iqbal, R. (2009) ‘Adaptive user-defined transaction relaxing approach for CSCW’ in Borges, M.R.S., Shen, W. Pino, J.A., Barthès, J-P. A., Luo, J., Ochoa, S.F. and Yong, J (eds.) *Proceedings of 13th International Conference on Computer Supported Cooperative Work in Design* held on 22–24 April 2009 at Santiago, IEEE, 362–367.

- Khachana, R. T., James, A. and Iqbal, R. (2011) ‘Relaxation of ACID properties in AuTrA, the adaptive user-defined transaction relaxing approach.’ *Future Generation Computer Systems* 27 (1), 58–66.

- Neugebauer. R. T., James, A. and Iqbal, R. (2011) ‘A General User-Defined Negotiation Application based AuTrA System for Computer Supported Collaboration Work’ in Shen, W., Barthès, J-P.A., Luo, J., Kropf, P.G., Pouly, M., Yong, J., Xue, Y. and Ramos, M.P. (eds.) *Proceedings of 15th International Conference on Computer Supported Cooperative Work in Design* held on 8–10 June 2011 at Lausanne. IEEE, 131–138.

Table of Contents

List of Figures

List of Tables

Abstract

Web services transactions have some unique characteristics. A Web transaction may be composed of a number of individual Web services, executed across multiple loosely coupled autonomous systems. Each Web service may be executed on an independent system belonging to an independent provider. There raises the question whether Web transactions can and should be maintained as a single business unit in a similar way to how transactions are maintained in classical database systems. In classical database systems, the transaction management protocol and mechanism are constrained by the primary properties of *atomicity, consistency, isolation* and *durability* (ACID). These ACID properties are the cornerstone of maintaining data integrity in transaction management. However, ACID properties were meant for centralised systems and are better suited for short transactions. Unlike short transactions, Web services transactions may be long-running; they can take hours or even days depending on the application. Composing certain actions from loosely coupled distributed business processes across multiple distributed applications is one of the essentials of Web services transactions. The classic ACID model, which is tightly coupled, is therefore seen as too rigid to support all the requirements of the new Web transactions model.

The research proposes a system that increases throughput while maintaining the consistency and correctness required by the particular applications that are using the model; the system is known as AuTrA (Adaptable user-defined Transaction relaxed Approach). AuTrA allows relaxation of each ACID property. The model is adaptable to meet different situations with different characteristics. For instance, in some cases it will be appropriate to relax atomicity, whereas in others it may be appropriate to relax isolation and atomicity while maintaining consistency. The research explores how transaction support for Web services can be customised to suit the needs of varying applications and result in improved service.

The AuTrA prototype has been implemented. The experimental results show that the AuTrA application is able to support the basic features of Web services transaction management, allowing users to specify their correctness requirements, and it can increase throughput of transactions in models in a flexible and reliable manner. Additional facilities allow users to specify application-specific, non-ACID criteria that can increase throughput. Safeguards have also been implemented to prevent execution of inappropriate user specifications, such as relaxation of properties that may damage data integrity. AuTrA can be used as a tool by software developers who need to compose applications from independent Web services and who wish to build applications which result in improved performance while maintaining application-required consistency.

Chapter 1: Introduction

1.1 Introduction

This chapter discusses the motivation that led to the research, outlines the problem, provides the path for the research and states the contributions of the research. The structure of the thesis is also outlined in this chapter.

1.2 Motivation

Web services are programs that are self-contained, modular business applications that can be dynamically discovered and invoked across the Web. Web services are based on industry principal technologies such as SOAP, UDDI, WSDL and XML. They make available a variety of ways to incorporate several applications in business-to-business (B2B) communication through SOAP and WSDL standards. Different organisations are able to connect their applications with those of other organisations in order to do business across networks; this is made possible through the composition of component Web services.

A transaction is a unit of work that may be made up of individual operations. For instance, a transaction which transfers an amount of money from one bank account to another may be made up of a number of operations. Traditionally in classical database systems, transactions should be completed in their entirety or not at all. For instance, in the transfer of money from one account to another it would not be good if the money was taken from one account but not added to the other. To achieve reliable transactions, strict ACID (atomicity, consistency, isolation, durability) properties have to be applied in traditional database systems. ACID properties are traditional properties for a transaction. *Atomicity* is based on an all-or-nothing policy; that is to say, the transaction must execute completely or not at all. *Consistency* implies that the transaction must be left in a state in which data or information is reliable and consistent. *Isolation* stresses that a transaction cannot show its uncommitted data to another transaction, meaning the transaction should finish before it can show its results to other transactions. Data that is committed needs to be saved; that is what *durability* implies.

However, in the Web environment a transaction may be composed of a variety of Web services, each of which may be seen as a mini-transaction in its own right. Maintenance of strict ACID properties becomes more difficult because of the fact that the Web transaction can be long-running (and it is not good to lock out resources for hours or days) and furthermore, Web services are owned by different providers, each of which may have their own policies which might conflict with a centralised transaction management system. In addition, many Web transactions may not need strict application of ACID properties, although in some circumstances these may be necessary. Thus the question arises of the applicability of ACID maintenance in a Web services environment.

1.3 Transaction issues

Web services transaction management has been an interesting topic to the research world and has constituted an important branch of data and systems research over the two last decades; first in the context of database systems and more latterly in the context of internet systems and e-commerce technologies. The major problem with traditional transaction management is the rigidness of the ACID properties, especially for long-running transactions. This can be a drawback when it comes to business applications. A number of models have been proposed by researchers with respect to providing support for advanced transactions (e.g. distributed transactions, nested transactions and chained transactions) to meet some of the requirements of long-running transactions. The relaxing of the traditional ACID properties, for example, allows parts of the transaction to commit even if other parts fail, or allows intermediate results to be shown to other transactions. This can support collaborative work which is becoming key to today's business environment and is characterised by long-running transactions. An example could be collaborative design. To lock a user out completely while another user runs a transaction may not be ideal. It may be preferable for a user to see uncommitted results rather than to wait for a long time to do their part of the work. Models introduced in the early eighties serve as building blocks for the current proposed models.

In 1982, Moss introduced the nested transaction model to extend on flat transactions by dividing transactions into sub-transactions, i.e. parent and child transactions. A child can start after the parent has started and can commit locally. The parent also terminates only when the child transaction has completed. When the child transaction commits the results, they cannot be seen immediately; they can be seen only after the parent transaction has completed. The idea of sub-transactions is good but could fail the modern business environment, which in some circumstances needs intermediate results to be shown due to the fact that the transactions can be long-running. This would require the relaxation of isolation, which this nested transaction model does not support. However, since this model allows increased modularity, concurrency and finer recovery than traditional flat transactions, it has some advantages and it can increase performance. But with relaxed isolation it might have been even better. The nested transaction model and its extensions are more powerful than the traditional flat transactions but they are still only suitable for specific environments and are still a long way off supporting environments requiring long-running transactions. This model was aimed at federated database systems.

In 1983 Lynch introduced a model that relaxed atomicity. The model allowed the modulation specification between operations for transaction execution. The modulation specification defines how each module can infuse with others, allowing each module to commit, even if some of its siblings have not committed.

Garcia-Molina and Salem introduced the Saga model in 1987. Its main goal was to deal with long-running transactions. The model uses the concept of compensation and, similarly to the Moss model, it uses the concept of sub-transactions but it caters for long-running transactions, not specific federated databases. Saga is based on the idea of chained transactions, which decompose long-running transactions into small, sequentially-executing sub-transactions. According to Gray and Reuter in 1993, the idea originated from IBM's Information Management System (IMS) and HP's Allbase database products. In the Saga model, each transaction is allowed to commit individually, allowing partial results to be seen by other transactions. Compensation is used to undo all the effects if the whole transaction has to abort. The benefit of this model is that it allows the sub-transaction to commit, resulting in its intermediate results being shown to other transactions and thus relaxing isolation, which may be useful for the business environment. However, this model has never been implemented, but many models introduced after Saga used the foundation of Saga in terms of compensation implementation.

In 1990, Chrysanthis and Ramamritham proposed a novel approach, which moved towards customisable transactions. The motivation for this was that some transactions might need customisation according to user requirements. They developed a framework named ACTA (A Comprehensive TransAction Framework for Extended Transactions). The model unifies the existing models to capture the semantics and rationale for the concurrency and recovery properties of composite transactions. The relations between the transactions are articulated in the form of effects, i.e. the effects of transactions on other transactions and effects of transactions on objects they access. The effect on the transactions used *commit-dependency* and *abort-dependency*. Dependency exists between two transactions, T_1 and T_2. For example, in commit-dependency, a hotel service cannot book a hotel until the flight service has booked the flight. The abort-dependency states that if the flight service aborts the booking of the flight, the hotel service must also abort the booking of the hotel.

ACTA motivated the later ASSET (A System Supporting Extended Transactions) model introduced by Biliris et al. in 1994, which uses primitives at a programming language level based on ACTA building blocks, like *history*, *delegation*, *dependency* and *conflict set*.

Since 1994 further models have been introduced which address transaction management in a Web environment and these are discussed in the literature review in Chapter 3.

1.4 Research question

Based on the above motivation, this research is interested in finding suitable user-defined and customised ways to provide transaction support for the Web services environment.

Hence, the main question that this research aims to answer is:

> ***Can transaction support for Web services be customised to suit the needs of varying applications and result in improved service?***

1.5 Research aim

Building on from the research question, this research assumes an improved service to be an increase in throughput of transactions while maintaining correctness according to application requirements. Throughput is the number of tasks successfully completed over a given period of time. Increase in throughput therefore means a better service to consumers, since tasks will be successfully completed with less delay. However, it is important that this improvement is not at the expense of correctness of the database and correctness should be maintained in accordance with user requirements.

Thus the research aim is to develop a system that increases throughput while maintaining the consistency and correctness required by particular applications.

It is conjectured that the above aim is achievable by relaxing some of the ACID requirements that are used in traditional transaction processing. The characteristics of Web-based transactions indicate that this is a plausible direction (Ramampiaro and Nydard 2004; Younas et al. 2006; Zhou, Wang and Jia 2004). Implementing a system which relaxes ACID properties in a manner that does not compromise correctness will be a matter for this research.

1.6 Research design

The research began with the literature review so that a good understanding of current work in the field of transaction management could be gained. Following the literature review, a model for Web-based transactions support was designed which is adaptable to different situations and the requirements of consumer and provider. The model was implemented and evaluated through simulation. The simulation consisted of a number of experiments in which batches of transactions were entered into a new transaction support system. The results were analysed, presented and discussed. Finally an evaluation was made of the project and scope for future work was outlined. Thus the research methodology was as follows:

- Formulation of the research question
- Literature review
- Model design
- Model implementation
- Design of evaluation method
- Model evaluation
- Presentation of results
- Discussion of results

The problem was approached by first analysing the current situation. This was done by a literature review of the existing Web services' transaction protocols or models. The researcher conducted a literature review through reading different materials like journals, conference proceedings, magazines and books related to the research topic. The material which has been read is summarised in Chapter 2. Theoretical analysis and problem analysis was performed on the gathered information and summarised. Relevant themes from publications or papers were drawn upon during this phase and critically analysed. Analysis gave rise to the problem statement on which the research was based. From the summary gathered from the literature review, the models were compared and contrasted with each other. This was done by studying the scenarios of usage given in each paper and identifying the most important features of each model. Limitations of each model were identified and a gap was found which could be filled by the development of a more flexible and extensive model. Then a usage scenario was developed that included typical characteristics of the usage scenarios in the literature studied. The scenario was used (with subsequent extensions) to evaluate the new model and system developed in this research.

The new model AuTrA (Adaptable user-defined Transaction relaxed Approach) was developed (see Chapter 3). To validate the new model, a prototype system was then designed and implemented (see Chapters 4 and 5). The system was also called AuTrA. A number of experiments were carried out to further validate the model. The experiments were based around three realistic business scenarios.

1.7 Contribution

Many of the contributions centre around the development of the Adaptable User-defined Transaction relaxing Approach (AuTrA).

The main contributions of the research are as follows:

- A transaction support system, AuTrA, has been developed which allows Web services consumers to vary specific transaction support characteristics to suit their needs. Examples of this are: to allow partial completion of a transaction if this is appropriate to the application but to specify full completion when required; or to allow intermediate results to be seen when this will increase throughput but not damage integrity.

- The AuTrA system allows Web services providers to vary specific transaction support characteristics to suit their needs; in particular to ensure consistency is maintained when required or to increase throughput when required.

- The AuTrA system allows the relaxation of additional application-specific criteria to increase the throughput and success rate. As the name suggests, these are criteria that are specific to each application rather than generic like ACID properties.

- The AuTrA system supports negotiation between the system and the consumer to allow reconsideration of requirements if there is a conflict between the provider and the consumer.

- The work has yielded increased understanding of properties of Web-based transactions and the responsibilities of consumers and providers in the Web services environment.

1.8 Organisation of the thesis

The thesis is organised as follows:

Chapter 1 sets the scene and the problem statement, and states the contributions and organisation of the thesis.

Chapter 2 introduces transaction models and Web services. It then concentrates on related work on relaxation of ACID properties in Web services transaction processing, providing a critical assessment of the literature, and it also discusses protocols for long-running transactions. The main purpose of this chapter is to show how this work is based on existing research. Additionally the chapter aims to delineate the current state-of-the-art in commercial and open source protocols. Also this chapter provides a succinct overview of the area in order to more clearly show the gap that is addressed in the thesis.

Chapter 3 introduces AuTrA, the model and system developed as a major part of this research work. The chapter covers its underlying principles and also discusses the concept of correctness.

Chapter 4 presents the AuTrA system in more detail, as a proof of concept prototype. It provides the framework of the proposed system and the mechanisms of how each component works to attain the research aim.

Chapter 5 presents the simulation model that was developed to test the AuTrA system. This chapter also presents the strategy developed to evaluate AuTrA. The main idea of the chapter is to clarify how the research carried out the simulation, collected the raw data, and also to give the reason behind why each test case was chosen, emphasising what each case intended to show against the research aim.

Chapter 6 analyses and evaluates the value and effectiveness of the proposed system, AuTrA, through a set of experiments.

Chapter 7 critically assesses the research, states its contribution and compares it with its closest rivals. The chapter also includes a discussion, conclusion and recommendation for further work.

Chapter 2: Literature Review

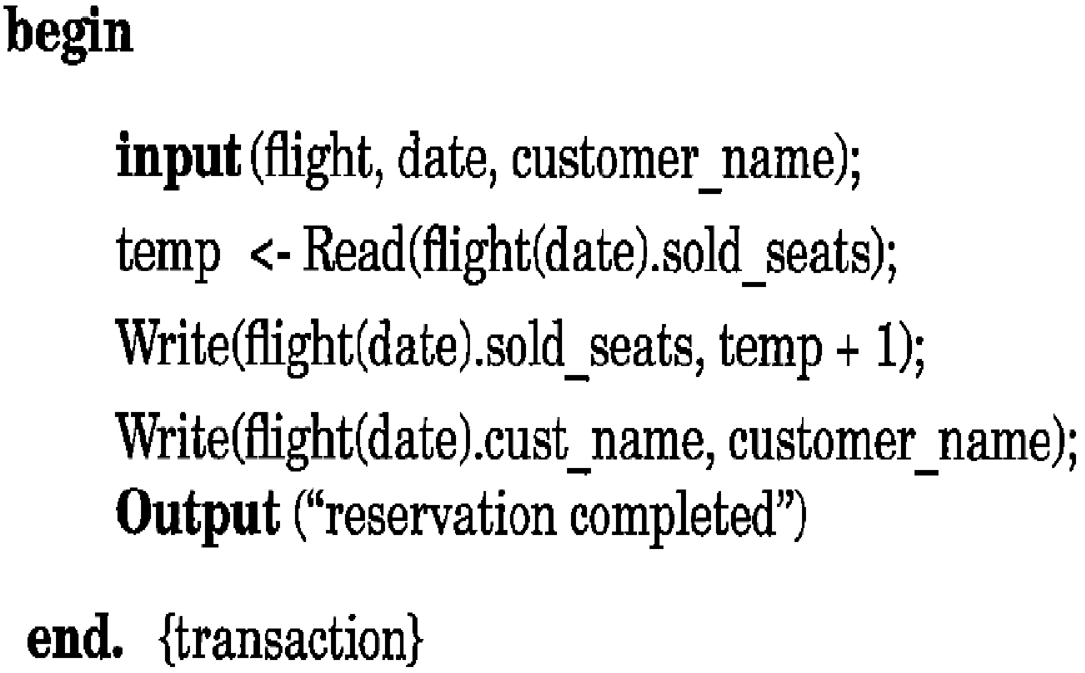
2.1 Introduction

This chapter discusses work related to this research. It sets the scene by presenting the concepts of transaction-processing, early transaction models, and characteristics of Web services. It then leads into work on relaxation of ACID properties for Web services transaction processing. Finally the state of the art in *commercial and open source transaction protocols* is discussed.

ACID properties are rigid and can lead to resources being held for a long period, even days, for long-running business transactions. This situation is undesirable in a business world where good throughput is necessary. Due to the problems of the classic ACID properties for long-running transactions, a lot of research has proposed models that extend the traditional transaction model for the Web services environment. Since the eighties, different researchers have introduced different models that relax mainly atomicity and isolation; some work has also been done in relaxing consistency. Interestingly, until recently durability has not been relaxed but some significant commercial companies, for example IBM, have now introduced models that relax durability. In this chapter the research will discuss the following: related work on relaxation of atomicity, related work on relaxation of isolation, related work on a combination of relaxation of atomicity and isolation, related work on relaxation of consistency and a combination of any atomicity or isolation or both, and related work on relaxation of durability.

2.2 Transactions

A transaction is a collection of operations that performs a single logical function. Transactions are often used in business processing environments and often involve reading from and writing to a database. The following primitives are usually used to describe such transactions in programming environments: *begin*, *read*, *write*, *commit* and *abort. Begin* starts a new transaction, *commit* ends a transaction, stores changes made during a transaction and makes changes accessible to other transactions. *Abort* ends a transaction and undoes all changes made during the transaction. An example of a transaction for booking a flight is shown in Figure 1.



```
begin

        input (flight, date, customer_name);
        temp  <- Read(flight(date).sold_seats);
        Write(flight(date).sold_seats, temp + 1);
        Write(flight(date).cust_name, customer_name);
        Output ("reservation completed")

  end.  {transaction}
```

Figure 1 Transaction example for airline reservation (Elmagarmid 1992)

In transaction processing, a DBMS preserves the database integrity and constraints using ACID properties. ACID stands for atomicity, consistency, isolation and durability. Let us consider the ACID properties.

- *Atomicity* implies that the transaction is all-or-nothing. That is to say, the transaction operation is either completely performed or is not performed at all. When the transaction fails, the incomplete results must be undone.

- *Consistency* concentrates on ensuring the data or information is reliable and consistent. For example, concerning the transaction in Figure 1, there could be a consistency requirement that the number of seats reserved is equal to the number of passengers that have reserved seats. This requirement would be checked as a post-condition of the transaction. At the start of a transaction the database should be in a consistent state, during the transaction processing the database may be in an inconsistent state, but at the end of the transaction the database should be in a consistent state again.

- *Isolation* stresses that a transaction cannot show its uncommitted data to another transaction. This means the transaction should finish before it can show its results to other transactions. Isolation does not allow concurrent access to data in a distributed database system.

- *Durability* implies that at the end of the transaction if no failure has occurred, the updates of the data should be made persistent, on either the disk or any other suitable mode of backup. For instance, with regard to the transaction of Figure 1, at the end of the transaction the updates must be persistent and they can then safely be read by other transactions.

In 1993, Gray and Reuter pointed out that the intention of transaction management in the context of databases is to guarantee the consistency of data in the incidence of failures and concurrent access. Often ACID properties are maintained through ensuring that transactions are executed in a serial manner whenever there is a potential conflict. This is achieved through locking and time-stamping.

2.3 Some seminal transaction models

Transaction models have developed as information processing has advanced. Business applications have over the last decade or so moved from centralised environments to distributed and mobile environments in line with technology advances. As disparate companies made their services available over the internet, the emergence of applications which utilised long-running selections of such services gave rise to the need for support for long-running transactions. Consequently, a need for models catering for the Web services environment arose because the models catering for the centralised environment were not fit for the new types of business application. For instance, a long-running transaction could run over hours or days, and it would be inappropriate to lock out all other transactions for this period of time. A lot of research has been carried out to address the limitations caused by centralised transaction models. A wide range of extended models have been proposed. Important early models are: *flat transaction model, nested transaction model, multilevel transaction model, Saga transaction model* and *spilt and join transaction model*. These models are discussed in the following sections.

2.3.1 Flat transaction model

According to Gray and Reuter in 1993, the flat transaction model caters for the uncomplicated form of transactions. The key to arranging an application for atomic execution, i.e. all-or-nothing, is to use a flat transaction. For example, Figure 2 shows a flat transaction in which all the actions or processes inside begin and commit at the same level. In this model there is no possibility of components of the transaction committing unless all other components also commit and all components must commit for the owning transaction to commit.

Figure 2 Flat transaction example (Gray and Reuter 1993)

2.3.2 Nested transaction model

A nested transaction model supports the division of large transactions into smaller transactions, referred to as sub-transactions. Moss introduced the nested transaction model in 1982 to take care of the flaws of flat transaction models (Moss 1982). Consider a transaction T_3 for a hen party arrangement, which comprises booking the dancers and booking the venue. Figure 3 shows T_3 in a nested transaction as the root transaction, the booking of the dancers is T_4, while booking of the venue is T_5; these are the sub-transactions. T_4 and T_5 further divided into T_6, booking of the clothing, and T_7, booking of instruments; these are sub-transactions of T_4 and T_5.

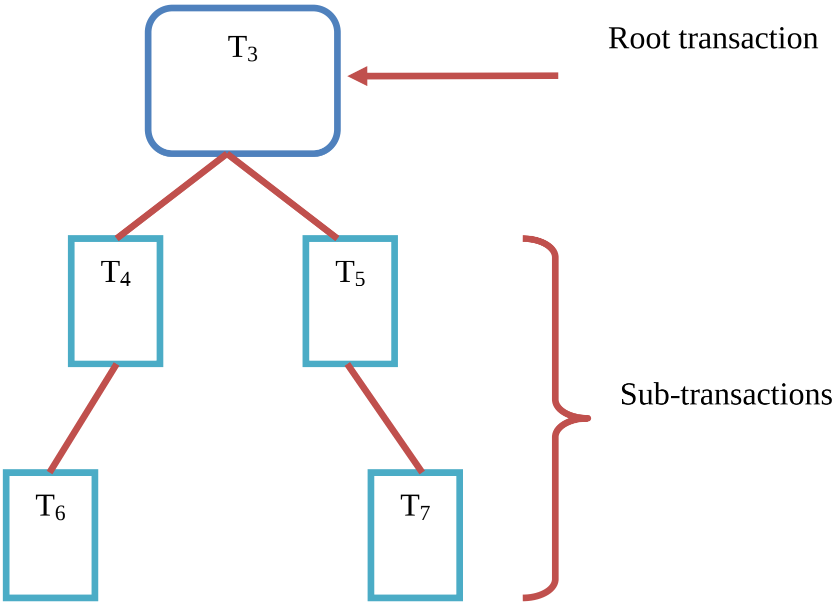
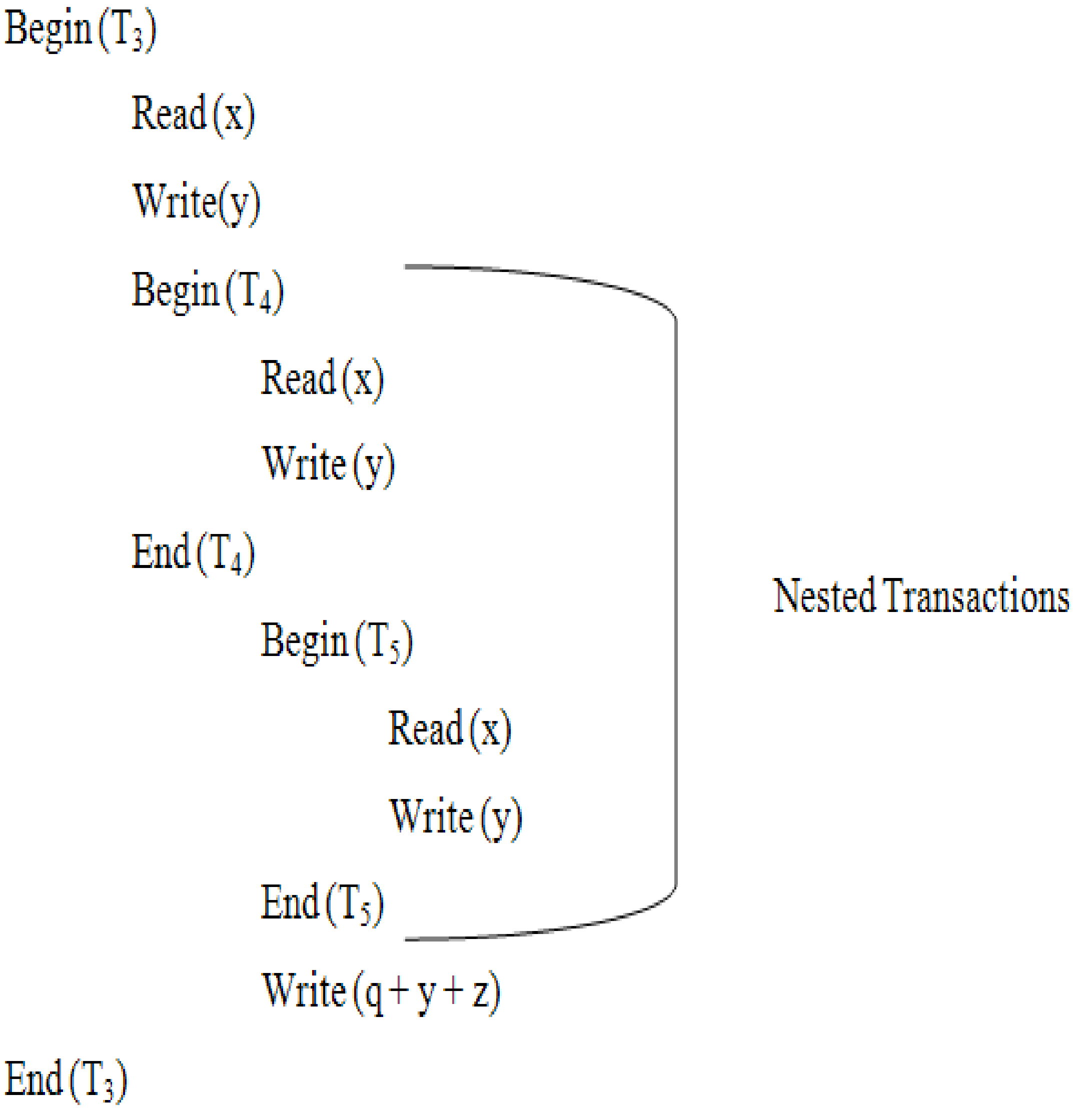


Figure 3 Nested transaction model example

The nested transaction model follows the *commit*, *rollback* and *visibility* rules. The *commit rule* implies that the results from the sub-transactions are available to the root transaction. For example, in the above scenario, the booking of the clothing transaction (T_6) results will be available to the booking of the dancers' transaction (T_4), and the results of the booking the dancers' transaction will be available to the booking of the hen party transaction (T_3). The *rollback rule* concentrates on making sure that when the root transaction aborts, the sub-transactions roll back, even if the sub-transactions have finished. That is to say, when transaction T_3 rolls back, T_4, T_5, T_6 and T_7 have to roll back, regardless of what they have done up until that point. This may be disadvantageous because time is wasted, which can be costly and is not a good thing for business. However, when T_5 rolls back it does not mean that T_3 has to roll back too. T_3 can simply ignore what happened to T_5 and continue, or it can abort, or it can execute T_5 again. The *visibility rule* states that T_4 and T_5 changes become visible to T_3 when T_4 and T_5 commit. If T_4 and T_5 are executing at the same time, changes made by each sub-transaction are not visible between them. However, if the executions are one after the other, changes are visible between them. Generally, T_4 and T_5 are not always consistent but T_3 is.



```
Begin (T3)
        Read (x)
        Write(y)
        Begin (T4)
                Read (x)
                Write (y)
        End (T4)
                Begin (T5)
                        Read (x)
                        Write (y)
                End (T5)
                Write (q + y + z)
End (T3)
```

Figure 4 Illustration of nested transactions

Unlike Figure 3, which shows that the transaction is at the same level in one block, Figure 4 shows that nested transactions are in different blocks and not at the same level.

2.3.3 Multilevel transaction model

Multilevel transaction models are different to nested transaction models, which have a fixed level of nesting (Weikum and Scheck 1992). Just as the nested transaction model takes care of the flaws of the flat transaction model, multilevel transaction models take care of some of the flaws of nested transaction models. Unlike nested transaction models, in multilevel transaction models sub-transactions can commit or abort autonomously and independently of the root transaction. The theory of transaction *compensation* attains this. Compensation can be expensive, especially if a lot of data needs to be compensated. Compensation transactions reverse the effects of already committed transactions. For example, in Figure 5 when transactions T_5 and T_4 commit, and if after they have committed T_3 fails, a compensation transaction is activated and T_4 and T_5 are compensated. Compensation transactions are useful, but having a lot of compensation would be expensive and complex. The similarity of the nested transaction model and the multilevel transaction model is that both are tree-based models. Compared to the nested transaction model tree, the multilevel transaction model tree is balanced. Nodes in a transaction tree match to operations at a certain level of abstraction in a layered system. Edges in a transaction tree signify the implementation of an operation. Concurrency control can be different in each level. The advantage of the multilevel transaction model is relaxation of serialisability for high concurrency.

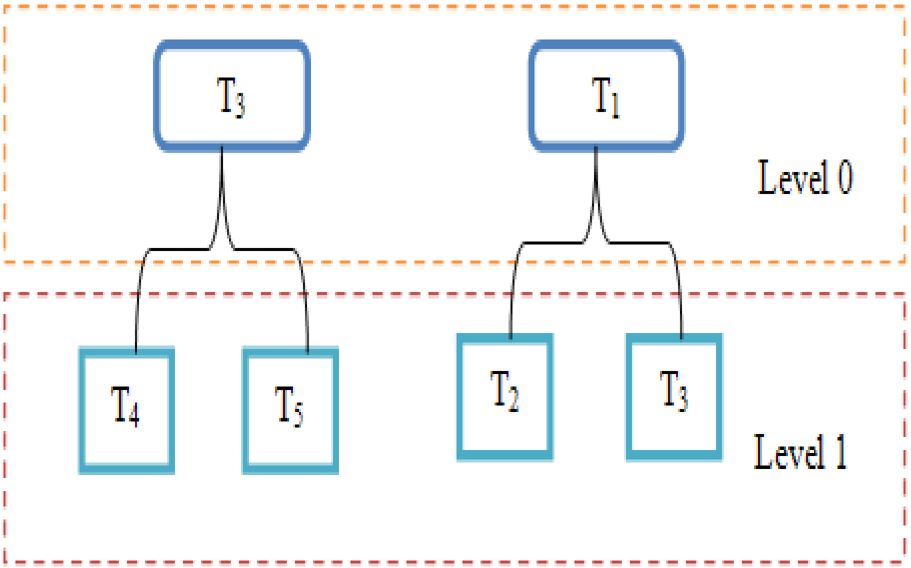


Figure 5 Multilevel transaction model

2.3.4 Saga transaction model

According to Garcia-Molina and Salem (1987), Saga is a transaction model that caters for long-running transactions by relaxing the isolation property. Figure 6 shows a Saga transaction made up of series of ACID sub-transactions and related compensating transactions. Sub-transactions are allowed to commit individually. The role of the compensation transaction is to undo the effect if the whole of the Saga transaction has to abort. The Saga transaction only commits when the sub-transaction (T) and related compensation transactions (ct) have committed. If the sub-transaction fails, the whole Saga transaction will compensate as Figure 7 demonstrates.

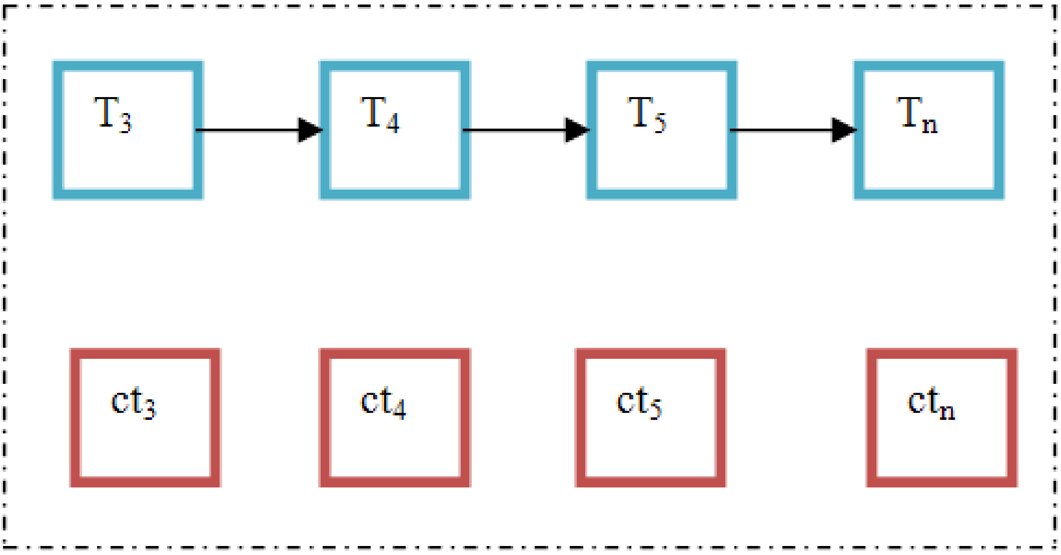


Figure 6 Saga successful transaction example

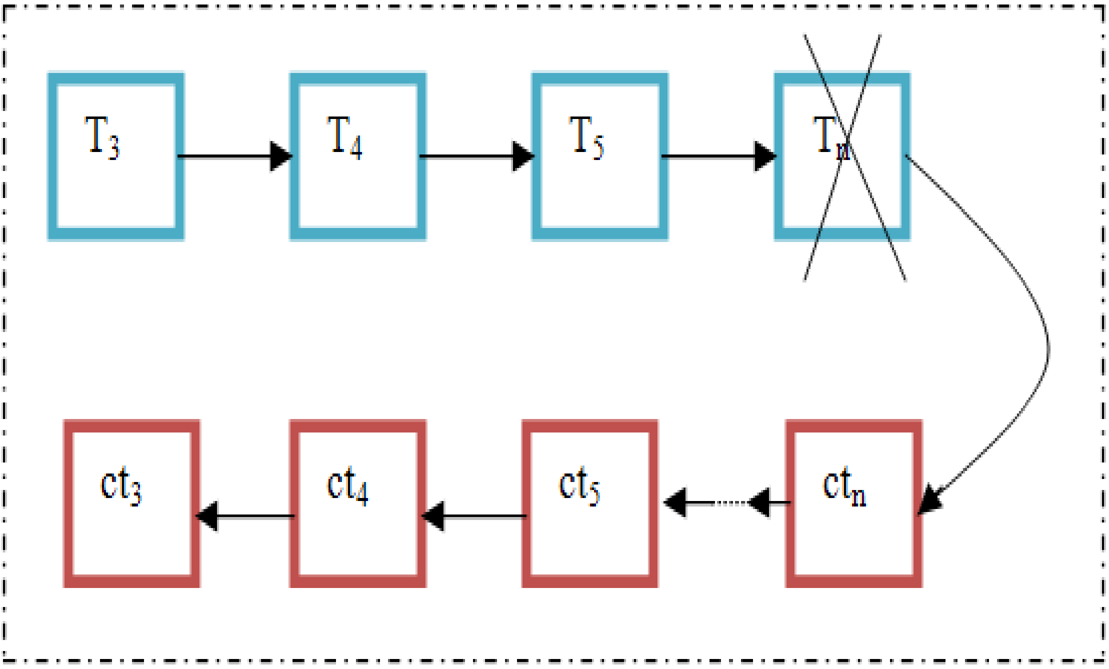


Figure 7 Saga unsuccessful transaction example

The Saga model has not been fully implemented but has influenced a number of models since it was proposed.

2.3.5 Split and join transaction model

In 1992, Kaiser and Pu introduced the split and join transaction model to aid open-ended executions related to transactions. Activities of long-running transactions are of indecisive duration, uncertain development and are interactive with other concurrent activities. As a result, the split and join transaction model focuses on these characteristics. Figure 8 shows how T3 breaks up into two serialisable transactions, T4 and T5. T4 and T5 results are later joined as one atomic unit.

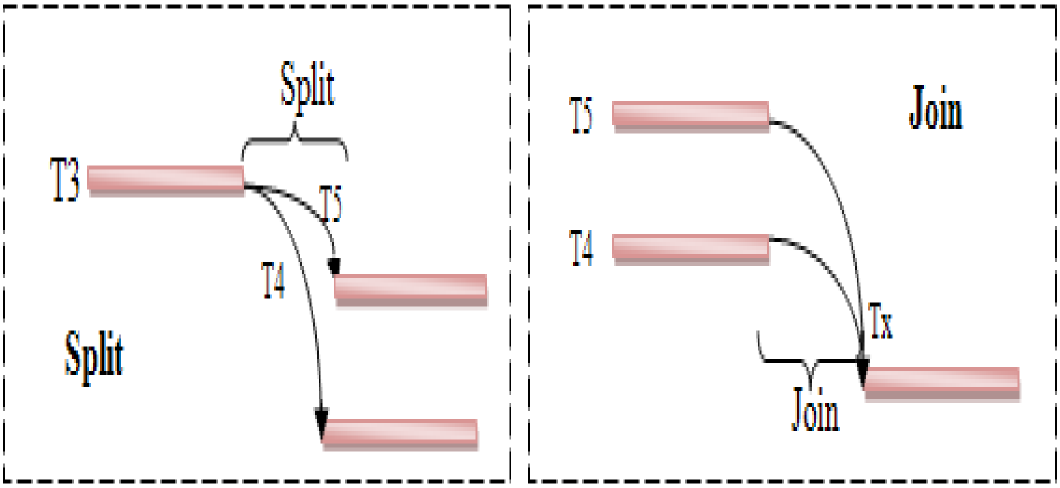


Figure 8 Illustration of the split and join transaction model

The split and join transaction model allows the transfer of resources from one transaction to another. This allows locks to be executed in parallel; hence this does not break the serialisation rule. The drawback of this model is that it follows the serialisation criterion. That is to say, isolation is not relaxed; transactions must be isolated when running. Unlike the split and join transaction model, the multilevel model relaxes serialisation for high concurrency. Relaxing serialisation is beneficial to the business as it increases throughput.

2.4 Web services

This section describes the architectural ideas and technologies of Web services. According to Limthanmaphon and Zhang in 2004, Web services are services accessible through the internet, which conduct transactions. The discovery, integration and execution of services is made possible by techniques defined by Web services standards (Chinnici et al. 2007). These standards include UDDI, WSDL XML and SOAP. UDDI (*Universal Description, Discovery, and Integration*) is an XML-based registry for businesses from different geographic locations to list themselves on the internet. UDDI can be compared to a telephone book's white, yellow and green pages. Using UDDI, businesses register the business name, product, location or the Web services offered. WSDL (*Web Service Description Language*) is an XML language that contains information about the interface, semantics and administration of a call to a Web service, while SOAP (*Simple Object Access Protocol*) is a protocol specification for exchanging structured information in the implementation of Web services in computer networks. Web services can be combined to make higher-level applications. Services can be combined in two ways, which are orchestration or choreography. Orchestration involves a coordinator Web service that directs other services involved in the cooperation. The coordinating Web service is aware that the other Web services are involved in the composition process, but the other involved Web services are not aware of the cooperation. Thus orchestration is centralised with explicit definitions of operations and order of invocation of Web services. Compared to orchestration, choreography is not controlled by a main coordinator. Each Web service taking part in the choreography knows whom they are interacting with and when to participate. Choreography is mainly used for collaboration globally, enabling business partners from different geographic areas to participate together. The invoked Web services involved in the collaboration must be aware of the business process, operations to execute, messages to exchange and the precise time of message exchange.

Important characteristics of Web services that are relevant to the topic of this research are that applications can be built using Web services offered by disparate organisations through the use of standard protocols and that Web services transactions, which form part of such applications, may be long-running which can cause resource-blocking. Consider again the hen party arrangements example of booking the venue and entertainment. An attempt to book the venue might be put on hold while the venue booking service waits for confirmation of a cancellation. This in turn might put the whole transaction on hold causing unacceptable delay unless an alternative transaction model which relaxes ACID properties is used.

2.5 Related work on transaction models which relax ACID properties

In recent years much work has focused on relaxing different ACID properties, chiefly with the motivation of providing a more flexible environment for Web transactions, given their specific characteristics. In the next sections, a selection of models will be discussed, grouped according to which ACID properties are relaxed. A summary of the findings of this section is given in Appendix A.

2.5.1 Relaxation of atomicity

In 2005, Bhiri, Perrin and Godart developed a model for Web services' composition which relaxes atomicity. The model caters for failure of atomicity required by designers in composite Web services. The researchers introduced Accepted Termination States (ATS). The ATS property is a correctness criterion for relaxing atomicity. ATS defines the Accepted Termination States of each component service. ATS is specified by designers and a composite service is not valid if there are some termination states that do not belong to the ATS specified by the designers. For example, the ATS can be *completed, failed, compensatable* or *completed aborted*. Component transactions will have different sets of ATS. If, for example, the state *failed* does not belong to the ATS set of accepted terminations of a component service, then the existing transaction property says it must be retriable. Again, if the state *compensatable* does not belong to the ATS, then the existing property states that there is no need to be compensatable. This model can be used by different interaction patterns in the same structured transaction. This can be a benefit since it is flexible and can cater for different situations. It is a one-size-fits-all model for atomicity relaxation.

In 2006, Ding, Wei and Huang introduced a model using new software called Internetware, which is designed for the open dynamic nature of the Web services environment. An Internetware application is composed of existing services, which are combined to form composite services according to the user's requirements. Since the model is made of primitive services, its transactional capability is normally described by two properties: *retriable* and *compensatable*. According to Ding, Wei and Huang, the existing research on transaction composite services (TCS) depends on the analysis of the composition structure and a handling mechanism in order to guarantee the atomicity. To enable relaxed atomicity the composition structure of TCS has to be analysed and there has to be a guarantee that there exists at least one must-succeed path after the non-compensatable service. The proposed model relaxes atomicity. Users are able to define different relaxed atomicity constraints for different TCS according to application-specific requirements, which include acceptable configuration and preference. This approach can handle complex application requirements, avoid unnecessary failure recoveries and perform the transaction management work automatically. Preference is used in places where more than one choice has to be made. The preferred one must succeed over the least preferred one. Consider transactions set up to examine different travel options. For example, where flight booking and train booking are invoked in parallel, the flight booking might be preferred and the train booking would then be compensated. Likewise, if money is an issue and a B & B and hotel are invoked in parallel, the B & B must succeed and the hotel must be compensated. Nevertheless, in a situation where there is no preference, any can succeed. The transaction management is done automatically by the system—not manually by the user. Since users define the relaxed criteria according to the application-specific requirements, users express their specific requirements through the set and order of TCS configurations, which must be acceptable. All acceptable TCS configurations are ordered according to preference. No matter which service succeeds or fails, the execution must end with a legal configuration.

2.5.2 Relaxation of isolation

Roberts and Srinivasan (2001) and Roberts et al. (2001) introduced Tentative Hold Policy (THP) in a W3C white paper. The basic concept behind the model was the support for long-running Web-based business transactions in which the entire business transaction may be made up of several component transactions, each perhaps operating at autonomous providers. In some cases, a business transaction may need all components to succeed for the complete transaction to be successful. Locking resources is not efficient for business as it could stop other customer transactions accessing the resource for a long time. This is known as blocking. As an alternative, the tentative hold policy was introduced with which a transaction might put a tentative but non-blocking hold on an item. If another transaction also wishes to hold the item, the first transaction is informed so transactions have an awareness of the amount of interest in an item. This approach is different to the two-phase commit as the hold on items is tentative rather than absolute, and thus throughput is improved. Since the publication of the tentative hold policy, a number of variations of transaction management methods for long-running, Web-based transactions have been developed.

Park and Choi in 2003 introduced a model based on THP which uses two-phase commit (2PC) protocol in combination with the THP protocol to ensure the atomicity of the transaction. The proposed model of Park and Choi adaptively determines the hold duration of the resources, which results in improved performance of the transaction. This can be compared to the THP model of Roberts et al. (2001), which had a fixed duration to hold the resources. A straightforward example is a situation in which Nancy and John want to access their joint account at the same time. With Park and Choi's model, the time the account will be held is not fixed, as opposed to the fixed time in the THP model. In Park and Choi's approach, the holding of the account by both is flexible to the environment. This results in fewer transactions being aborted and an improvement in performance.

In 2003, Younas and Iqbal showed that even if Web services are mainly used for developing and integrating business systems and applications, it is possible to support collaboration editing applications (CAEs) by introducing a transactional approach in collaborative applications. The proposed model is based on correctness criteria called SACReD (semantic atomicity, consistency, resiliency, durability). The model relaxes isolation by allowing component transactions to commit or abort independently, but it has a strict all-or-nothing policy, which means that every component transaction has to complete successfully if the composite transaction is to complete successfully, otherwise the composite transaction fails.

In 2005, Haller, Schuldt and Türker introduced a model that relaxes isolation in a peer-to-peer environment. They pointed out that the peer-to-peer environment typically enables access to Web services in different peer environments. The novelty of the model is that it ensures global correctness without depending on a serialisation graph. Global correctness is achieved through communication among the peers in which dependent transaction processes may be running. Rollback is used in case of failure.

Alrifai, Dolog and Nejdl in 2006 introduced a model that extends the WS-transaction protocol (Cabrera et al. 2001) for concurrency control in Web services environments. Agreement between the client and the service providers takes place before the service is invoked, i.e. the client composes the process and identifies the deadline, then contacts the service providers. The service providers which match the client deadline, whose local schedulers can be synchronised, are selected. Then the execution plan is produced. All the parties involved agree on the time for holding the exclusive lock during the execution of the commit protocol. The protocol is non-blocking in the sense that the output of the scheduling process is always a non-blocking schedule. The commit-order-preserving scheduler ensures the correctness of the concurrency execution. The mechanism avoids direct communication with the coordinators for security reasons. However, the Alrifai, Dolog and Nejdl model sends twice the number of messages, which can be time consuming and costly.

Yang, Liu and Ling in 2006 presented a transaction-aware protocol for a Web services transaction coordinator (taTHP). This is done by the coordinator being able to know the success probability of the transaction. This helps the coordinator to select the transaction that will be successful and reject those that might not be successful. This protocol grants maximum autonomy of isolation. This means that clients can think that they are the only ones making a reservation on the resource. Clients cannot see how many reservations have been made on a particular resource. For example, consider a case where three people make a transaction of reserving a ticket. Transactions A, B and C can reserve ticket number 456; the client of transaction A will not know that the transaction B client has made a reservation on the same ticket. It allows several clients to place a tentative hold on the same resource and confirm availability before the completion of a transaction, just like with THP. When a client executes the actual business transaction, the other clients will receive a notification informing them that the reservation is no longer valid. THP tries to introduce a maximum hold size and duration time in order to take care of the coordination situation, but it still has a problem in terms of coordination. This is an issue, since THP does not know which request should be granted and which ones should be rejected. This is because THP is not aware of the transaction context and the success probability of the transaction. Being aware of success probability can be vital since the resource manager will know which resources can be granted most effectively.

Böttcher, Gruenwald and Obermeier in 2006 presented a model that reduces the number of transaction aborts and blocks in a transaction. The model is based on the Web services transaction specification and is an extension of existing atomic commit protocols. The transaction enters the suspend phase after the read phase. The suspend phase is non-blocking and in this phase the resource manager can still abort the transaction if there is a need to grant a request to another transaction, and it is also used to reduce the number of aborts in a situation of missing votes or conflicts. The new approach identifies those sub-transactions that are repeatable or reusable instead of aborting and restarting all sub-transactions of a global transaction.

Zhao, Moser and Mellior-Smith in 2008 introduced a reservation-based, extended transaction protocol. Their protocol reserves (reservation phase) and cancels/confirms (confirmation/cancellation phase) to coordinate the tasks of business activities across multiple businesses. This protocol does not depend on compensation. The model demonstrates that the use of compensating transactions has a much higher probability of inconsistencies, especially when the data spreads across multiple enterprises, as is the intention of the Web services environment.

Kumar and Barvey in 2009 proposed a Non-Blocking Commit Protocol (NBCP). Each site, including coordinator and participant, maintains a database in its primary memory as a transaction database. Every database maintains a transaction ID, primary memory ID, transaction status, participant ID and vote from each participant. The transaction database is deleted automatically after the transaction completion. The backup which holds the replicate of the transaction database is kept in the primary memory backup. The primary memory backup works concurrently with coordinator, in case of failure or network delay. The coordinator and every participant also maintain a replicated copy of themselves. NBCP relaxes isolation. The model is based on the idea of the two-phase commit (2PC) protocol and is reliable in the sense that it can survive a coordinator or participant crash.

In 2009, Wang, Li and Min introduced a model that relaxes isolation while ensuring consistency in Web services transactions. The model extends the WS-Business Activity protocol and is based on a transaction dependency graph distributed over multiple nodes. The proposed graph named Web Services Transaction Dependency Coordination Protocol (WSTDCP) is able to identify any transaction in an inconsistent state, using dependency relations. The end-state dependency is sent to the coordinator, which will ensure dependencies are removed appropriately so that transactions can enter the complete state and compensation can take place if necessary.

2.5.3 Relaxation of atomicity and isolation

Different models that relax both atomicity and isolation have been introduced in different research. In 1993, Godart introduced a framework that strived to involve cooperative or collaborative work in transactions. Godart developed a framework to support collaboration between software developers, based on the software development process. This was the Coo approach which relaxes atomicity and isolation. The model relaxes atomicity in the sense that long-duration transactions can save their partially complete or halfway results. That is to say, long-running transactions save their intermediate results, making use of the principle of partial rollback and therefore have an advantage over traditional transaction models. This can be effective in the cooperative or business world, in cases where the system crashes or there is power failure. All work need not be lost; it is not all-or-nothing. Some of the components that have been committed that are parts of the whole transaction, can be saved. For example, T_3 is a transaction made up of sub-transactions T_4 and T_5. If a crash or power failure takes place, if T_4 has finished and committed, while T_5 is not yet committed, T_3 can still save the parts that have been completed by T_4. Therefore, what T_4 has already done will not be wasted. This saves time because when T_3 restarts it will only have T_5 to process, resulting in less time being needed for processing and a higher throughput.

Relaxed isolation allows several software processes to access these intermediate results at the same time, while not violating the correctness criteria. To cater for the incorrectness of data, which can be caused by dirty reading, the model uses three different consistency levels: *stable*, *semi-stable* and *unstable.* A *stable* object is one that has committed transaction results successfully and is completely consistent according to the business requirements. *Semi-stable* objects are those in which the transactions are in the process of generating tentative data but are violating some of the correctness criteria and can be seen as not consistent enough. Lastly, *unstable* objects are those that do not contain any correctness criteria at all. These objects are locked by some processes and cannot be accessed until they become stable or semi-stable. These three objects are stored in different databases. This model of correctness constraints and management of activities is tailor-made, which is good for this type of collaboration. The use of three different degrees of stability is a valuable idea, since it allows flexible support for collaborative work, which is missing from the traditional transaction models.

Agrawal, Abbadi and Singh in 1993 suggested another model, which contributes to the collaboration work and relaxes isolation and atomicity. Their main goal was to develop a transaction model by merging flexible transaction models from collaborative environments and semantic-based correctness criteria. They used a notion of relative atomicity, which is used to state how a co-action can be interleaved relatively to other co-actions without breaking the overall atomicity requirements for collaborative activities. The models use a *relative serialisability* correctness criterion to check for correctness execution, which is an additional relaxed criterion to the classic (conflict) serialisability (SR). Thus, the fundamental assumption is that any execution obeying the RSR criterion would maintain the consistency of the database, even if it is not serialisable. The model extends the standard 2-phase lock protocol (2PL) to assure the virtual serialisable execution. To handle conflicts, *push-forward* and *push-backward* locks must be acquired before the connecting operation sets a normal lock. A push-forward lock causes any conflicting operation to be delayed until the last operation of the actual atomic unit with which it conflicts is run; a pull-backward lock is used to move operations backward before the start of an atomic unit. A typical application area is the design environment. However, to be able to specify relative atomicity, one must know the complete sets of operations before the involved transactions can be executed. Because one must know the complete set of operations beforehand, i.e. before the transaction is executed, transactions which vary their operations according to circumstance are not well supported. This can be a disadvantage for business environments which need dynamic applications.

In 1995, Rusinkiewicz et al. introduced a model that relaxes atomicity and isolation. The model allows users to investigate several alternatives to solve problems. The model allows compensation, just like the other advanced transaction models.

Conradi et al. in 1997 used a similar concept to Godart's Coo model. They introduced the EPOS (Expert System for Program and ~og~ System Development) framework, which is for quality-assured software engineering. The framework has a database to manage the resources produced during the development stage and is similar to Coo in the sense that it uses workspaces (both private and common workspaces) and it uses the check-in and check-out mechanisms for interaction with other workspaces. To take care of concurrency, EPOS uses locks to control access to a shared workspace. In order for the users to know the actions that affect their work, awareness support is provided. Awareness mechanisms are used to support correctness execution and aid in taking care of access conflicts. EPOS uses nested transactions due to the fact that cooperative transactions are long-running. This echoes the ideas of Kim et al. in 1984 and Bancilhon, Kim and Korth in 1985.

The aim of Wäsch's 1999 work was to develop a transaction model and a specification language that would allow efficient information-sharing. The model is called CoAct and was developed based on an extension of existing advanced transaction models. Their motivation was to overcome the limitations imposed by the use of the standard ACID model. The requirements for the transaction model were distinct as they used four application scenarios: *cooperative authoring*, which was all about unplanned processes; *software engineering*, looking at semi-structured processes; *design for manufacturing*, using structured activities; and *workflow*, dealing primarily with automated business processes. The model relaxes atomicity and isolation. Isolation is relaxed by dividing work into packages, sending the packages to various workstations, where the work packages are executed in parallel and then returning and merging the output later into a single unit. The advantage of this model is that it has tried to cater for a variety of applications, unlike the Coo or EPOS models.

In 2004, Ramampiaro and Nydard came up with an interesting approach by proposing a model which provides transactional support that can be tailored to meet different needs or situations and can also be modified following changes made in the actual environment while work is performed. The model is called CAGIS-Trans(Cooperative Agents in a Global Information Space-Transactions). The model tries to meet the flaws of fixed criteria proposed by other researchers which make their models inadequate for cooperative work. For example, other models did not allow users to specify their relaxation. The proposed solution of the model relaxes isolation and atomicity. Unlike Conradi et al. (1997) Rusinkiewicz et al. (1995) and Agrawal, Abbadi and Singh (1993), whose models relax atomicity and isolation without customisation, in this case atomicity and isolation are customisable. This means that users can decide whether to relax atomicity or isolation or not. The drawback of not relaxing atomicity and isolation may be the cost of the rollbacks.

In 2004 Younas, Eaglestone and Chao introduced a protocol for e-business transaction management. The protocol was called Low Latency Resilient (LLR). The protocol relaxes atomicity and isolation. For correctness criteria, the protocol applies SACReD (Younas, Eaglestone and Holton 2000). The advantage of this protocol is allowance of flexible components, meaning that a stated alternative could be executed in case of abort of the current one. As a result, the number of transactions aborted is reduced. Since the protocol allows individual components which are independent to commit, this results in releasing the locks and reducing resource-blocking.

Fauvet et al. (2005) followed the direction of Robert et al. (2001) by proposing a THP model that tentatively makes resource reservations and avoids resource-blocking. Just like the original THP protocol, the model is aimed at ensuring atomicity of the transaction. However, unlike the model of Robert et al., the presented model defines different levels of atomicity. That implies that a transaction can still be committed even if some of its component transactions are aborted. The model therefore relaxes atomicity and isolation. This model is also similar to Bhiri, Perrin and Godart’s (2005) model.

Younas et al. in 2006 introduced the commit protocol that is an extension of SACReD (Younas, Eaglestone and Holton 2000). The protocol is used to ensure correctness and reliability in distributed systems. The protocol aims at improving performance while ensuring correctness and reliability. Transaction Commit Protocol for Composite Web Services (TCP4CWS) relaxes atomicity and isolation.

In 2008, Choi et al. presented a model that maintains consistency while relaxing isolation. The model is similar to the Alrifai, Dolog and Nejdl (2006) model and is intended to fit with a representative WS-transaction standard, for easy amalgamation into existing WS-transaction systems. The protocol uses a dependency management protocol called Web services Transaction Dependency management Protocol (WTDP) to detect inconsistency between dependent transactions. The WTDP detects the inconsistency states of transactions with the notion of end-state dependence and can recover them to the new consistency states. This inconsistency can happen in a case in which a dominant transaction fails or aborts before completion. This protocol allows participants to automatically supply related information to other participants. For example, participant B will give information that participant C needs. Let's say C needs some information about the income of B for his transaction. B will constantly be supplying that information during the process and that information will be closely monitored. If it happens that B fails, C will be able to detect this and both transactions will be rolled back to a consistent state. This model is better than Alrifai, Dolog and Nejdl's model (2006) since it sends half the messages of their model, which results in this model being more efficient.

2.5.4 Relaxation of consistency and atomicity and isolation or both

Although the majority of the research has been done in relaxing atomicity and isolation, Terry et al. in 1995 investigated a model that relaxes consistency. In this model, clients can read and write to any replica without the need for coordination. That allows some inconsistency. Inconsistency can be useful since it increases availability, in situations where it can be tolerated because no vital information is needed. For example, in a call centre application during peak hours, there might be no need to make some databases consistent, depending on the kind of data the database is holding. Nevertheless, in situations where it is vital to have consistency, such as calculating prescription dosages, this model is not appropriate. Consistency in this model is finally achieved by making certain that all update conflicts are resolved in a consistent manner by all servers. The final consistency check may be done at a later time following a busy transaction period.

Pitoura and Bhargava in 1999 produced a model that relaxes consistency. In this model the data or information that is situated in the same place is joined together to create clusters. Joint consistency is required from clusters that come from the same collection. Clusters are sites of distributed systems which are grouped together. Strongly connected sites are grouped together and the same applies to the weakly connected sites. Direct access is applied to local clusters to increase interaction between clusters and increase availability. Two types of transactions are supported: weak and strong transactions. The weakly consistent clusters are committed locally, and after committing the changes can only be seen by weak transactions of the same physical cluster. Inconsistency is dealt with by allowing controlled deviation among copies located in a weakly connected site. That is to say, consistency is relaxed in the sense that integrity-constraints are ensured only for data copies belonging to the same logical cluster.

Yu and Vahdah in 2002 proposed a model in which they looked at the classical strong and hopeful consistency model for replicated services. They argued that replicated services can benefit from some relaxed consistency. They introduced a model that captured three independent application metrics, numerical error, order error and staleness, which helped to capture consistency. A broad range of applications can express their consistency semantics and, with the help of an application-dependent algorithm, the target consistency level can be enforced. The optimistic approach used in this model has been proposed before, but it has its drawbacks since it provides no limit to the inconsistency of data exported to the client’s location and end user. On the other hand, the proposed model has limits to the level of inconsistency allowed by introducing consistency requirements. That is to say, the model can allow a certain level of inconsistency. If the level of consistency is violated, data cannot be passed and the operation will be blocked until the synchronisation of a remote replicate, as determined by the system’s consistency requirements.

Zhou, Jin and Zheng in 2004 followed a similar direction to Yu and Vahdah. They introduced the Tsinghua Object Data Store (TODS) in 2004. Their model was built as a cluster object storage system to support the building of internet services. The model relaxes consistency and, just like Yu and Vahdah’s model, different levels of consistency are supported. TODS allows the system to continue when part of its storage fails. For example, a system that has a lot of nodes will continue to run if some or one node fails. TODS uses replication. This allows the model to cater for the requirements of different services as it caters for different levels of consistency, meaning services with varying consistency requirements can make use of the model. Such differentiation is useful in the Web services environment.

In 2004, Zhou, Wang and Jia produced a model which emphasises that not all distributed applications require strict consistency. For instance, applications in retail and wholesale information storage and retrieval may not require strict consistency. They introduced a model that relaxes consistency in the form of replication of data but emphasised that when replication is used to improve access it can be expensive to maintain data consistency. As a result, they introduced the use of ordering constraints to express the corresponding set of operations provided by the replica group. The ordering constraints can be defined in four categories: *FIFO,* which states that requests sent by the same client are to be executed in the order they are sent; *causal ordering* which states that if two requests have the nature of relationship, this relationship should be kept at all replicas; *total ordering* which states that the request be delivered in a predefined way and the ordering has to be consistent with the replicates; and *total + causal ordering* which is the integration of total and causal ordering. FIFO and causal ordering are needed from the client's point of view, total ordering is often needed from the replica group's point of view, whereas total + causal ordering is used to give satisfaction to both parties: client and replicate group. This model is valuable in the sense that it improves the system efficiency and throughput and still maintains data consistency. The approach is interesting, as it differentiates the needs of the client and server in the maintenance of consistency.

Another model proposed by Lee et al. in 2009 relaxes consistency in the Web environment. The model uses a similar concept to the use of replicas, as found in the approach of Terry et al. (1995) to relaxing consistency. However, this is extended through the concept of lease time through which the maintenance of consistency is achieved. The model has three-tier hierarchies on which each group and node independently and adaptively chooses the proper lease time and the protocol for each proxy cache. The innovative part of this model is the fact that it uses adaptive multi-levels for lease duration.

Another approach proposed by Younas and Mostefaoui in 2010 looked at transaction management in a slightly different way, i.e. in the way of context awareness in mobile services transactions. The approach incorporates SACReD (Younas Eaglestone and Holton 2000) as the correctness criteria and relaxes atomicity, isolation and consistency. This breaks the barrier of classic ACID properties, which may be too strict. The added feature of this model, which is vital, is the fact that it is adaptive to the conditions and the users' needs. The system automatically adapts to the environment by taking into account the context information such as location.

2.5.5 Relaxation of durability and ACI

The evolution of technology has led to some big commercial companies introducing models that relax durability. IBM introduced (IBM SolidDB in 2009), which relaxes durability. This means that at the end of the transaction, data is not made permanent straight away. However, later, at a more convenient time, the database is brought up to date. The system permits three different durability alternatives: *strict durability*, *relaxed durability* and *adaptive durability*. Strict durability focuses on not allowing any durability; this behaves like the traditional model that makes data permanent at the end of the transaction. Adaptive durability is for HotStandby operations, i.e. the system configuration has two servers, the main server and the secondary server. The primary server is the one that executes all the jobs and the secondary server is the one to which data is sent. Therefore the secondary server contains the same information as the primary server. The primary server is a read-and-write database while the secondary is read only. In this system, an application can choose between relaxed or strict durability. The durability can be relaxed only when both servers are running; if not, the mode is strict durability. This system allows greater throughput because it relaxes durability. However, in some situations the system will have drawbacks. For example, imagine financial services systems in which a broker is busy evaluating equity position and is buying stock. While the broker is in the middle of the transaction process, the secondary server fails. Failure of the secondary server means that the mode of the system switches from adaptive relaxed durability to strict durability. Thus the transaction is delayed and the broker misses his purchase. In this kind of scenario, when time is critical, a delay due to strict durability caused by the failing of the secondary server can be unhelpful from a business point of view.

IBM continued on the research above and released SolidDB Universal Cache system in 2009. This is an improvement on the previous system. However, this model focuses more on using RAM to process all the information. Using RAM overcomes the difficulty of the traditional disk-based database system, which can be very slow in I/O access. The advantage of using memory to process information is an increase in processing speed, resulting in a good throughput. Again, the system provides the support for distributed transaction processing, through two-phase commit, by the use of the Java Transaction API (JTA) interface. This improvement lets a system be fully interoperable with principal application servers, such as the IBM Web Sphere Application Server, as they manage complex applications requiring multiple data sources. The benefit of this system is that not only does the model relax durability but it is also adaptable and can be used by any application, be it a standard database application, a Web services application, or a Web-based environment.

In 2009, (SYBASE 2009) introduced a system called the Adaptive Server Enterprise (ASE) system. Their system relaxed durability by providing two levels of durability: in-memory relaxation, just like the SolidDB Universal Cache system, or both in-memory and disk-based relaxation of durability, similar to IBM's SolidDB. The advantages of the system are that one can choose to use the in-memory mode to relax durability, or the disk-based mode to put some or all of the data in the memory or disk. In that way, if there is a failure, the databases can be made persistent. This enables relaxed durability databases to take advantage of many performance optimisations of in-memory databases. The difference with the IBM model is that, in normal operation, the ASE system does not write the logs at all. It is different from IBM Relaxed, in which transaction logs are written all the time.

Another player that came in to relaxing of durability is Oracle, with the introduction of (Oracle TimesTen in-memory database 2009). Oracle TimesTen in-memory database functions on databases that fit exclusively in physical memory, using standard SQL interfaces. The system uses transactional replication for high availability. The system takes advantage of managing data in memory, and optimising data structures and accessing algorithms. Thus database operations are executed very efficiently and as a result the system achieves dramatic gains in responsiveness and throughput. To take care of the lost update issue, the master database and the subscriber have an internal mechanism that will confirm that the updates have been successfully committed. Oracle TimesTen does this through providing two return-service options for applications to verify that the replicated data is consistent between the master and subscriber databases which are *the return receipt service* and *return twosafe service*. The *return receipt service* synchronises the application with the replication technique by blocking the application until replication confirms that the update has been received by the subscriber. The *return twosafe service* enables fully synchronous replication by blocking the application until replication confirms that the update has been both received and committed by the subscriber. Using *the return receipt service* trades some performance to ensure higher levels of data integrity and consistency between the master and subscriber databases.

2.5.6 Other approaches to performance improvement in transaction models

Some transaction models use other techniques to improve performance without relaxing ACID properties. Zhang et al. in 1999 introduced a model which uses a new timestamp ordering (NTO) approach that runs both classic transactions and long-running collaborative transactions in one system. In this model, in situations when there is a crash, the transaction will not fail—instead a new time stamp will be given and the transaction will incorporate recent updates and continue as normal. In long-running collaborative transactions, NTO uses high priority on the last read or write conflict in order to create the correctness criteria. This is through the concept of final serialisability, meaning only the last read or write are given priority.

Awan and Younas in 2004 proposed an approach for efficient commit in Web services transactions. The approach is called ‘priority commit protocols’. The model uses a priority active network scheduling mechanism at each network node based on head-of-line (HoL) scheduling mechanisms. The reason for using HoL is to reduce the queue delay at each network node. The priority scheduling gives preferential priority to high priority messages. The benefit of this approach is the improvement of the commit process by the reduction of queues. This is particularly beneficial where Web traffic is significant. However, the approach uses the strict ACID approach, which may be a drawback in certain situations.

In 2006, Younas and Chao moved in a slightly different direction by presenting a model to improve the performance of Web services transactions. The model is based on the new tentative commit protocol (TCP). TCP is based on the concept of tentative commit that allows transactions to tentatively commit on the shared data of Web services. In a situation where there is network or system failure, the transaction is cancelled. The protocol restricts tentative holds and thus may improve performance. Even though THP improves throughput by increasing the commit chances of a composite Web services transaction, the approach can sometimes be disadvantageous, since there may be performance degradation due to multiple tentative holds and communications. This is why TCP may be better in some contexts. The authors claim improvement in performance and in future work throughput will be measured.

On the other hand, Greenfield et al. in 2007 proposed a model called Promise which provides a mechanism that clients can use to guarantee that they can rely on the valiability of information resources remaining unaffected in the course of long-running applications. Promise is an agreement between the client's application and the services. The clients' application can agree on what resources they need in order to complete successfully. The Promise service will look at the request for the resources and decide either to grant the request for the promise of the resources or to deny it. Promise is similar to the ConTract model of Wachter and Reuter in 1992, which used expressing conditions to permit tasks within a workflow to complete successfully. The other model that is similar to Promise was introduced by Gawlick and Kinkade in 1995. This model reserves access to resources just like the Promise model.

2.6 Commercial and open source transaction protocols

Two well-known standards groups for Web services are W3C (2009) and OASIS (2009). These groups have been actively involved in developing protocols for Web services transactions. The use of the internet to perform transactions is common in the business environment. This area of application is generally known as e-commerce. As already discussed, protocols which support loosely coupled environments and do not follow the classic locking of the resources, are needed to support transaction processing in the new environment. W3C and OASIS have built on research in advanced transaction models and have produced some well-defined standard protocols to support the new requirements. The following sections introduce *Business Transaction Protocol* (BTP) and *Web Services Transactions* (WS-Transactions). Their relationship to the research is discussed as is the notion of middleware. A summary of the findings of this section is given in Appendix B.

2.6.1 Business Transaction Protocol (BTP)

Business Transaction Protocol is a specification realised by OASIS (Little and Freud 2003). BTP supports transaction synchronisation of participants of services presented by multiple independent companies as well as inside a single company. For handling the synchronisation of change of state, BPT uses the two-phase completion protocol. The two-phase completion protocol impedes transaction throughput because of the locking of the resources during the process, but it ensures consistency. However, in some cases, as discussed below, this approach is stronger than is necessary. Figure 9 shows the BTP Stack which supports two types of transactions. These are *atom transactions* and *cohesion transactions*.

Figure 9 BTP stack (Little and Freud 2003)

2.6.1.1 Atoms transactions

Atoms transactions are similar to the classic ACID model, which is that the whole transaction takes either place or nothing. Thus all the participants in the related Web services will see the same outcome, and either they accept it or reject it.

2.6.1.2 Cohesion transactions

The main idea in cohesion transaction was to relax atomicity and this allows certain work to be completed or cancelled based on the main business rules. Because of this it means that there could be a different transaction outcome to all-or-nothing. Thus transactions can complete even if some work has been rejected. In this case, compared to atoms transactions, the two-phase protocol gives the user a choice to define which atom participants or standalone participants to prepare or cancel. It is a good idea when using cohesion transactions to divide a work into units of transactions. This assists in situations where the business activity encounters some situations in which it is useful to cancel the atomic unit of a transaction with a warning and a *confirm-set*. A *confirm-set* is a set of all participants that have to confirm in order to terminate the business activity. Once the confirmation participants' answers have been determined, the whole cohesion transaction becomes an atoms transaction, resulting in all confirmation participants seeing the same outcome. BTP allows relaxation of atomicity and isolation, and by so doing, it allows tentative states of change during transaction processing. The completion of a transaction is either confirmation or cancelling. BTP does not state how to implement prepare, cancel or confirm. The advantage of BTP is being able to control time between phases, meaning the application is able to choose the interchange, which has been prepared before the termination. That is, BTP lets the participants inform the coordinator well in advance what the decision will be and when it will be taken. For example, the participants might say they will remain prepared for 24 hours and after that they will cancel. This is known as forward operation and in the case of group participation the services use the participants to supervise the outcome of the results. The participants can leave the transaction at any time after the participants have prepared. The leaving of a participant also shows that the participant is not interested in the outcome of transaction.

2.6.1.3 Qualifiers

To take care of the long-running and loosely coupled environment, BTP introduced *Qualifiers*. The main purpose of a Qualifier is to provide additional extended information within the protocol. BTP gives the user the flexibility to extend the Qualifiers' implementation to suit the application requirements. For instance, it provides Qualifiers like Time Out, when users can specify how long a transaction may be allowed to wait before it times out. Allowing flexibility is a good thing, since the protocol can be tailored, and this allows different applications with different needs to use the protocol.

2.6.1.4 Relationships in BTP

The relationships in BTP are of *Superior-Inferior* type. The superior is always the leader of the inferior. Superior is the one conveying the results. It can send the CONFIRM to some atoms and CANCEL to others, provided it is composer of the cohesion (see Figure 10).

Figure 10 BTP superior-inferior relationship (Little and Freud 2003)

Superiors can be composers, coordinators, sub-composers and sub-coordinators, and inferiors can be sub-composers, sub-coordinators and participants. The coordinator of the atoms sends the same outcome to the inferiors. In the superior-inferior hierarchy, the sub-composers and sub-coordinators are inferiors to parent nodes in the trees but superior to the children nodes.

2.6.1.5 BTP transaction participants

Just like in any other transaction, there are participants taking part in the transaction. BTP have different types of participants, which are the *initiator*, the *factory*, the *coordinator*, the *terminator*, the *services participant* and the *enroller*. The *initiator* is responsible for starting the transaction depending on the program request. The request containing information will be sent to the coordinator that will control the process. To start the coordinator the initiator uses *the factory*. Again, the factory generates the framework of the business transaction. The principal participant in BTP is the *coordinator*, which is responsible for taking care of two-phase commit protocol. The coordinator gets the information of the transaction outcome of the participants from the *terminator*. The communication system from the inferior side is the *services participant.* This works hand-in-hand within the *enroller* by passing the message it received from the initiator to it. The participants act according to the information context request they received from the enroller. The terminator gives the final decisions to the coordinator of *confirm* or *terminate*.

2.6.2 WS-Transactions (WS-Tx)

WS-Transaction is a specification developed by BEA, IBM and Microsoft which describes the means for transactional interoperability between domains and provides a mechanism to combine transactional groupings of Web services into applications (http://www.ibm.com/developerworks/library/specification/ws-tx/) Figure 11 shows the relevant components for WS-Transaction. WS-Transaction supports two transaction protocols: WS-AtomicTransaction for short duration ACID transactions; and WS-BusinessActivity for long duration business transactions (Cabrera et al. 2009a and 2009b). WS-Transaction also works with the WS-Coordination specification. WS-AtomicTransaction and WS-BusinessActivity can be combined in situations of business transactions that are generally long-running, and which can be made up of several sub-transactions that are atomic.

Figure 11 WS-Transaction components (Cabrera et al. 2001)

2.6.2.1 WS-Coordination

Web services require management concerning *transaction management*, *replication*, *workflow*, *caching* and *security*. WS-Coordination (Cabrera et al. 2009c) is responsible for the management of Web services, i.e. the outcome and the processing. The most important part of WS-Coordination is the provision of generic coordination communications for Web services.

The generic infrastructure or communication of WS-Coordination enables the possibility of plugging in specific coordination protocols, for example a protocol for transaction management for security. Little and Freund, in 2003, pointed out that these specific protocols work between the services. The coordinator or manager is responsible for directing all the messages to the correct participants. The coordinator is responsible for disseminating information about the votes in this case to all participants and ensuring that all of them get the information. The context message directed to the participants can be commit or abort, depending on the number of votes received from participants.

New participants can opt for this context message to include the location or the endpoint of the coordinator. The context also includes protocol-specific information in relation to the actual coordination protocol used. Contexts use a SOAP header to encode messages and WSDL to use a synchronous invocation style for sending requests. According to Cabrera et al. (2009 c)., the coordination framework includes three elements, *activation service*, *registry service* and *coordination service*, which represent the basic responsibilities of all different kinds of coordination protocols between collaborative services.

The *activation service* concentrates on creation of a new activity coordinator for a particular application instance. The activation service also enables the nesting of activities, indicating the association between new and existing activities. The coordinator uses a specific coordination protocol, for example protocol configuration and negotiation, which defines the negotiation between the Web services to determine which coordination service model is to be used. It also defines the process for communicating the results of a process. The *registry service* is responsible for guaranteeing that registered Web services are driven through to completion by using the selected protocol. The *coordination service* focuses on the definition and provision of processing patterns. For example, the strict ACID transaction service provides a protocol that defines a sequential processing: prepare, commit and rollback.

2.6.2.2 WS-AtomicTransaction (WS-AT)

There is a need to support short-running transactions. WS-AtomicTransaction (WS-AT) is a protocol designed for this purpose. The initiator process begins the transaction protocol and the transaction coordinator controls the transaction protocol. The general purpose of the protocol is to ensure that the initiator and the participants agree on the outcome of the transaction.

The WS-AT specification provides the description of the atomic transaction coordination type that is used with the extensible coordination framework described in WS-Coordination. This specification defines three specific agreement coordination protocols for the atomic transaction coordination type: completion, volatile two-phase commit and durable two-phase commit. The completion coordination type states that the completion protocol starts the commit processing, according to the participants registered by the protocol.

When the transaction is in process the coordinator will receive either a commit or rollback message and then executes the volatile 2PC protocol prior to proceeding through to the execution of the durable 2PC protocol. A status (either a committed or an aborted message) of the transaction is transmitted back to the initiator of the completion. Because WS-AT supports classic ACID transactions and is intended for short-duration interactions among trusted partners, the coordinator directs all participants to either commit or cancel using well-known 2PC protocol.

2.6.2.3 WS-BusinessActivity (WS-BA)

WS-AT specification works the same way as the traditional 2PC ACID transactions. Therefore it is too rigid and not practical for long-running transactions. To avoid issues caused by WS-AT in long-running business transactions, a second coordination type called WS-BusinessActivity specification (Cabrera et al. 2009 c) was introduced. This specification defines protocols that allow existing business process and work flow systems to interoperate. A business activity usually consumes many resources, spans multiple atomic transactions (even human interaction), and can require a long time to complete.

An important aspect of WS-Transaction (see Figure 12) that differentiates it from traditional transaction protocols is that a synchronous request/response model is not assumed. This model derives from the fact that WS-Transaction is layered upon the WS-Coordination protocol, the communication patterns of which are asynchronous by default. WS-Coordination provides only context management. It allows contexts to be created and activities to be registered with those contexts. WS-Transaction improves the context management framework provided by WS-Coordination in two ways. First, it extends the WS-Coordination context to create a transaction context. Second, it augments the activation and registration services with a number of additional services (*Completion*, *CompletionWithAck*, *PhaseZero*, *2PC*, *Outcome Notification*, *BusinessAgreement*, and *BusinessAgreementWithComplete*) and two protocol message sets (one for each of the transaction models supported in WS-Transaction) to build a fully-fledged transaction coordinator on top of the WS-Coordination protocol infrastructure. WS-BusinessActivity, unlike WS-AtomicTransactions, is proposed for long-duration transactions. The protocol provides ACID-relaxed transactions among loosely coupled systems where locking resources is impractical or not desirable.

Figure 12 WS-Transaction overview (Cabrera et al. 2001)

The advantage of the protocol is the fact that sub-transactions may commit autonomously of each other without having to wait for the root or parent transaction to commit. In case of a sub-transaction failure, the client driving this business process may decide whether the overall transaction should abort or simply ignore the failed sub-transaction. Compensating actions are used to undo completed child tasks in the case of transaction abort. On the other hand, the theory that all service operations can at all times be compensated is not rational. When the number of transactions having access to transitional results increases, the compensation of some operations becomes either too expensive or even impossible.

WS-BusinessActivity relaxes isolation, that is to say results of completed tasks within business activities can be seen prior to the completion of the business activity. These tasks are in fact tentative and when they need to be compensated, business logic is necessary to make it possible, especially if a business activity spans a long period and has numerous participants which rely on the outcomes of the task that is to be compensated. The WS-BusinessActivity specification provides the definition of two Business Activity coordination types: *AtomicOutcome* or *MixedOutcome*. These are to be used with the extensible coordination framework described in the WS-Coordination specification. AtomicOutcome deals with atomic outcomes and does not relax atomicity since all the participants in the direction commit or abort; it is all-or-nothing. The coordinators that deal with mixed outcomes relax atomicity, since they lead each individual participant to close or compensate. For example if T_1 is a parent transaction which has children T_2, T_3 and T_4. Let us say T_2 and T_3 can commit but T_4 cannot. The parent transaction T1 can still commit, even if some parts of the transaction, i.e. T_4, did not commit. The protocol maintains consistency by the use of compensation.

The WS-BusinessActivity protocol specification introduced the relaxation of isolation and atomicity, which is useful for long-running transactions. To maintain consistency the protocol relies on compensation, which may be costly, especially when compensation has to be used repeatedly to make the data consistent at the end of the transaction.

2.7 Comparison of the different approaches

The various approaches investigated have been compared in terms of: the ACID properties that are relaxed; distinguishing features; customisability; how inconsistency is handled; and whether second chances may be given to the user to reconsider requirements. The results of the comparison are tabulated in Appendices A and B. The results were analysed and a summary of the findings is provided in Table 1.

Table 1 provides summarised information about the models evaluated in the literature review in terms of their relaxation of ACID properties. There was no model that allowed relaxation of all properties (as shown in Table 1). Combinations of properties not shown in Table 1 were also relaxed by no models. Figure 13 represents the findings as a pie chart.

Table 1 Summary of relaxation of ACID properties

ACID Relaxation	**Number of Models**
No Relaxing of ACID properties	11
Atomicity Relaxation	5
Consistency Relaxation	7
Isolation Relaxation	12
Durability Relaxation	4
Atomicity and Isolation Relaxation	13
Atomicity, Consistency and Isolation Relaxation	1
Atomicity, Consistency, Isolation and Durability	0

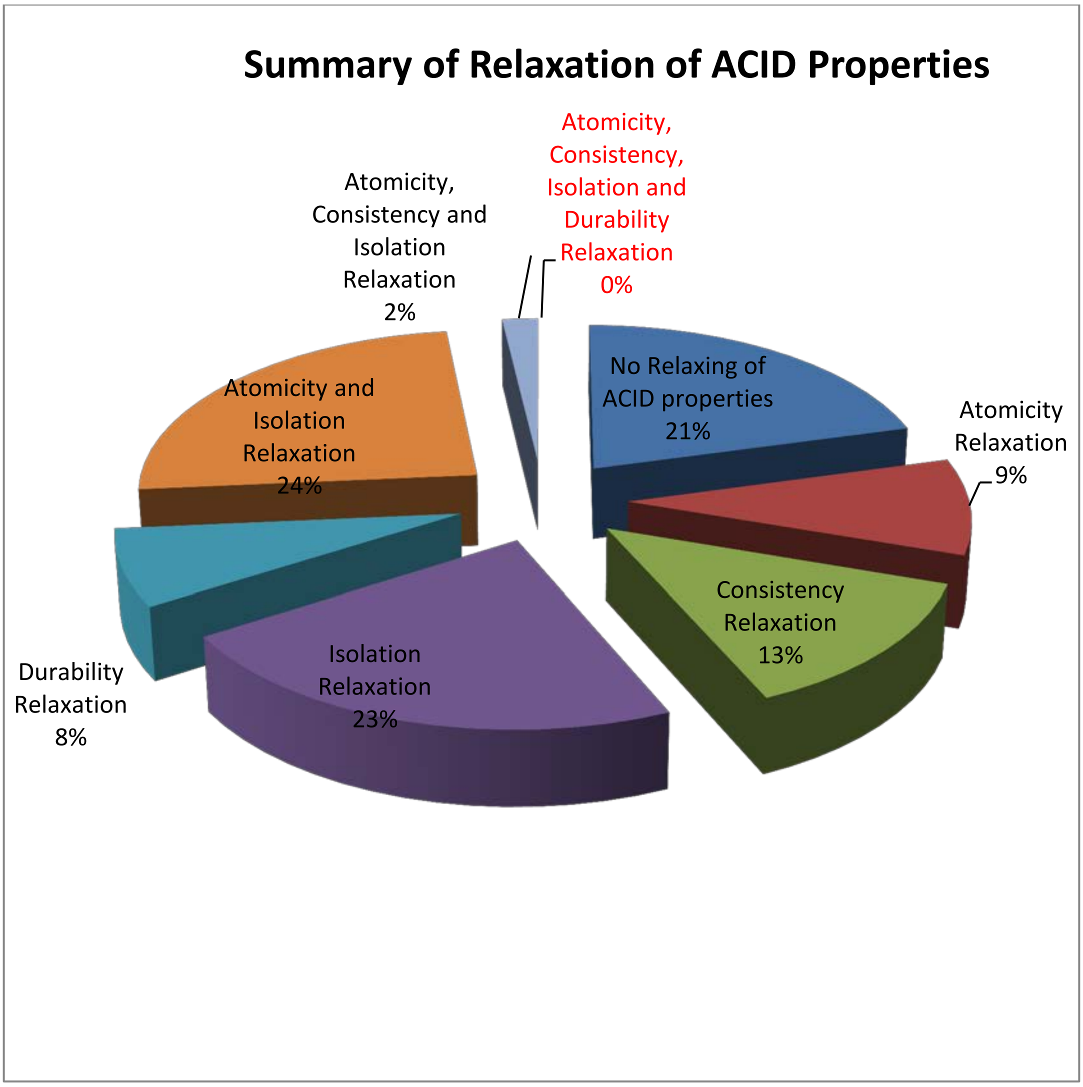


Figure 13 Pie chart showing summary of relaxation of ACID properties

So far no approach allows customisable relaxation of any combination of ACID properties nor permits service consumers to reconsider requirements if the relaxation requested is not compatible to the business requirements of the provider. It was felt that such a system would be beneficial, particularly for Web application developers, who may wish to experiment with various relaxation strategies. Hence the development of the Adaptable user-defined Transaction relaxing Approach (AuTrA) which is described in Chapters 3 and 4.

2.8 Summary

This chapter has provided an overview of the technologies and ideas from related work that are relevant to this thesis. Previous work from both academia and commerce in transaction management has been presented. Many models have been proposed previously for relaxing ACID properties in the context of Web services. The models proposed in the literature have benefits. For example, not holding resources when relaxing isolation is an advantage to long-running Web services applications, because holding resources can lead to deadlocks or can slow down processing time, which could in turn lead to loss of revenue. Similarly allowing sub-transactions to complete when the composite transaction fails can save on rollbacks without losing consistency in some cases. Most of the work that has been investigated has relaxed atomicity and isolation but some has relaxed consistency. Some major commercial players have also relaxed durability. Some of the work investigated allows users to select what relaxation might be appropriate for their applications. None of the related work allows relaxation of all ACID property with or without customisation. AuTrA will provide this facility. AuTrA is described in the next chapters.

Chapter 3: Requirements for a New Transaction Model

3.1 Introduction

This chapter introduces the Adaptable user-defined Transaction relaxing Approach (AuTrA). AuTrA is the system developed by this research to provide more flexible support for transactions in a Web services environment. Its main characteristic is the support for customisable relaxation of each of the ACID properties. In this chapter, firstly the motivation for developing AuTrA is considered. Then the relaxation of each of the ACID properties is discussed with consideration being given to correctness in AuTrA. Finally the idea of using application-specific criteria in addition to ACID criteria to improve composite transaction success rates is presented.

3.2 Motivation to develop AuTrA

The Adaptable user-defined Transaction relaxed Approach (AuTrA) model is proposed to support Web services transactions. This model builds upon previous work in allowing customisable relaxation of all four ACID properties, as well as enabling application-specific criteria to be used to increase the success rate of transactions. AuTrA enables varying relaxation of ACID properties, based on user-defined relaxation of atomicity and isolation, and adaptable relaxation of consistency and durability, based on provider requirements. In AuTrA, correctness of data is ensured by various techniques. These are compensation, synchronisation, tentative hold and relaxation specification. Additionally, application-specific criteria can be specified to reduce transaction abort and restart.

Web services transactions are often of long duration and may be initiated from or executed at globally disparate locations. It is therefore difficult to verify in advance the release of the resources held by transactions. This can affect the performance of the system, which in turn affects the business. This factor harshly affects the throughput of transactions. The following are generic dimensions that are relevant to transaction support for different areas of application:

- The interaction between the resources involved may be synchronous, i.e. occurring at the same time, meaning that two different users or transactions may require one resource at the same time; or the interaction may be asynchronous, namely a resource may not be required at the same time by different transactions.

- The users may not be in the same geographic location.
- Since the nature of work is long-running, the duration of the work may not be known in advance and is volatile.
- The resources used allow sharing.

The research question (see Section 1.4) was:

> ***Can transaction support for Web services be customised to suit the needs of varying applications and result in improved service?***

Building on from the research question, a research aim was established (see Section 1.5) and this was *to develop a system that increases throughput while maintaining the consistency and correctness required by particular applications.*

It was conjectured that the above aim could be achieved by relaxing some of the ACID requirements that are used in traditional transaction processing. Therefore, the research had to consider what properties could be relaxed. We see from the literature review that atomicity, isolation, consistency and even durability can be relaxed. The circumstances and consequences of relaxing these must be carefully considered. In related work, the most common of the criteria to be relaxed are atomicity and isolation. Some research has also investigated relaxing consistency. Few researchers have relaxed durability. However, the commercial companies have relaxed durability in in-memory databases. No work has relaxed all criteria within the same model with the particular relaxation pattern customised according to the user. Can all or any ACID properties be relaxed in one model if the need arises? And if so, is the application still a transaction? The answer to this question depends on the definition of a transaction. If the definition includes the requirements that ACID properties are maintained, then relaxing any can lead to discussion as to whether the process concerned may still be termed a transaction. However as we have seen in Chapter 2, many extended models of transaction processing have been developed which permit some relaxation of these properties thus implying a less restricted definition of a transaction. The definition of a transaction in this work is discussed in more detail in chapter 5. In the analysis of the past research it can be seen that it is very common that atomicity and isolation are relaxed in one model, or that each ACID property is relaxed individually in one model. This can be seen in appendices A and B and in the summary in Section 2.7. It can also be seen that no model allows relaxation of all or any of the ACID properties. Another question is, what type of user should have control over the transaction? Is it the consumer or the provider? The relationship between these parties, their data, the nature of the service and requirements of the application needs to be considered in order to answer this question. Is it the service provider or the consumer who should have the final say as to what can be relaxed, considering the business requirements and sometimes the criticality of the information? The consumer may want transactions to run quickly but the provider may not be prepared to allow this if it compromises data integrity. In the following sections, the issues of relaxing each of the ACID properties are explored.

3.3 User-defined atomicity

The AuTrA system gives a provision of both strict atomicity and fully relaxed atomicity. By strict atomicity, it is intended that the system works in the same way as the classic ACID model. This is important in a situation where there is a need to preserve atomicity. For example, consider that Chris has to make a bank transfer by debiting one bank account X and crediting the other, Y. Assume only the first transaction of debiting account X was completed, and that the crediting of account Y failed. In this case the whole of Chris' transaction has to fail. It is all-or-nothing, and in this case the data integrity is very important. If the whole transaction does not fail, it means that Chris will lose the money he debited from the account, and he will have to try again to credit account Y, by debiting account X for a second time.

Nevertheless, there are situations where relaxation of atomicity is desirable. For instance, there are scenarios in which there is no need to waste the work done before by aborting the whole transaction. For example, consider travel arrangements to go for a skiing holiday. The user will need a flight to the holiday location, a hotel and the rental of skis. The user might not mind whether all of the services are booked or not. When atomicity is relaxed, in the case where the booking of flight is completed but the ski and hotel bookings fail, the flight booking can be committed rather than be wasted by failing with the rest of the transaction.

Therefore, allowing the user to say if they want to relax atomicity or not should be provided by the system. The user of the transaction must decide whether it is important to maintain atomicity or not. In the case of a bank transfer, the user may insist on atomicity maintenance but in the case of the holiday booking the user may decide to relax atomicity. The user or service consumer therefore owns the decision over the relaxation of atomicity.

As much as the user has the power over the relaxation of atomicity, the system also allows the user to decide on the correctness of the composite transaction. This is done before the user starts the transaction processing. So if the user is saying, 'Yes, relax atomicity', it means that the user is happy that all or any of the bookings of flight, hotel and restaurant are done. That defines the correctness criteria of the transaction. In other words, it is okay if not all component transactions complete successfully. If the user wants the correctness criteria of the transaction to be different, the user might choose strict atomicity. It might be argued that a typical end-user does not have sufficient knowledge on whether or not to relax atomicity. This could be true, but the area in which AuTrA is intended to be applied is the area of middleware. It is envisaged that application developers will utilise AuTrA to develop systems that are user-friendly and that will support users in providing their appropriate application needs. Thus software developers will analyse the application area, and in discussion with the users will determine the types of application where relaxation of ACID properties like atomicity is appropriate. In any case, as will be explained later in Section 3.5, the service provider or data owner is in charge of consistency. Any request for relaxation received from a user that does not fit with the integrity requirements of the provider, will be refused.
Application systems to help the user specify relaxation requirements can be developed. For example, where an interface might say:

"You have asked us to book a flight, a hotel and skis. If we can't get all three do you want us to book some of the others anyway?"

The above is a question that can determine whether a user is prepared to relax atomicity or not and is a question posed in a user-friendly way.

Relaxation of atomicity mainly has an impact on the service user or consumer, which is the reason why the AuTrA gives the service consumer the final say over the relaxation of atomicity. If relaxation of atomicity would cause the underlying database to be inconsistent, then the criterion of consistency would come into play, which is controlled by the service provider (see Section 3.5). Hence correctness of the database can be maintained.

3.4 User-defined isolation

Similar to atomicity, the system will allow both full isolation and relaxed isolation. Full isolation protects executing transactions from seeing each other's incomplete results but does mean that transactions can be delayed for long periods. Isolation allows multiple transactions to read or modify data without knowing about each other because each transaction is isolated from the others. This is achieved using low-level synchronisation protocols on the underlying data. These can include two phase locking, time stamp ordering or pessimistic locking. For relaxation of isolation, the system mechanism processes the transactions concurrently. This is needed in areas where sharing of data is common and where, processing composite Web transactions would otherwise incur many delays through lock-outs.

AuTrA enables the user to specify whether isolation is relaxed or not. In AuTrA, if a user of a transaction specifies that isolation may be relaxed, the user is in effect saying that it does not mind if the transaction reads some data that may not be committed yet. In non-critical applications, this drawback may be insignificant and may be quite preferable than slower throughput. Provided the requested relaxation of isolation can be restricted to the requesting transaction in that only that requesting transaction is affected, then the relaxation of isolation can be owned by the user. If the relaxation of isolation could cause serious damage to the underlying database, the transaction user should not relax it. Thus it is the nature of the application, the function of the transaction and the user's awareness of this that will determine whether isolation should be relaxed or not.

To determine whether or not the system might relax isolation, the application developers will need to thoroughly analyse the application. In terms of user interface, the system, in the case of a holiday booking application, might ask:

"Do you mind whether we use information that might not be guaranteed to be the absolute latest information when we check whether flights, hotels and skis are available?"

AuTrA uses tentative hold when atomicity has to be maintained and isolation is relaxed. In this case, multiple holds can be allowed, but as soon as the resource is not available, the other parties holding the same resource will be notified. In this case, all the transactions will see the correct information but as soon as the correctness no longer holds, the other transaction will know that the resource is not available anymore.

Consumer relaxation requirements

Atomicity	Consistency	Isolation	Durability
No	Yes	No	No

Provider relaxation requirements

Atomicity	Consistency	Isolation	Durability
Yes	Yes	No	No

Figure 14 Non-matching consumer and provider relaxation requirements: Example 1.

If relaxation of isolation would cause the underlying database to be inconsistent, then the criterion of consistency would come into play, which is controlled by the service provider (see Section 3.5). Hence correctness of the database can be maintained.

3.5 Adaptable consistency

When it comes to relaxation of consistency, the system allows both full consistency and relaxation of consistency. It is important to leave data in a consistent manner. This applies to areas where inconsistent data can lead to catastrophic decisions being taken. That is the reason why there is still a need for strict consistency. For example, leaving financial data with figures that are more than what is really in an account can result in the account holder making big purchases, while in reality the money in the account is not enough to go ahead with the purchasing. However, there are some scenarios in which relaxation of consistency cannot do any harm and may instead add value like increased throughput. This implies to circumstances like booking ski equipment for hiring. In this case, even if the information is not correct, and the numbers of what is available are not correct, there might be a way of dealing with that. For example, imagine the ski equipment shop continues to book, even if the amount of equipment available is negative the shop owner might have a backup of borrowing from a neighbouring shop that has more than they need. With this mechanism in place the transactions can be processed even if it appears that consistency rules are being broken, as there is no possibility of catastrophic decisions being made. The worst event that can happen is that a hirer comes to the shop and no skis are available. Whilst annoying, that sort of event is not a critical one. In fact, businesses like airlines routinely overbook seats and shops take orders without being absolutely sure whether they can fulfil them or not. In these real-life situations, examples of compensating actions are refunds of money for airline seat bookings that are not honoured and cancellation of retail orders that cannot be fulfilled. Sometimes financial compensation may be offered.

In AuTrA the service provider owns relaxation of consistency. That is to say, unlike atomicity and isolation, the service provider has the final say over the relaxation of consistency. If the service provider sees that the relaxation of consistency damages data integrity, the service provider will not relax consistency. Even if the user wants to relax consistency, the service provider will not relax consistency if integrity is crucial. In this case the system, instead of rejecting the user transaction outright, will give the user a chance to rethink the requirements in line with the service provider’s needs.

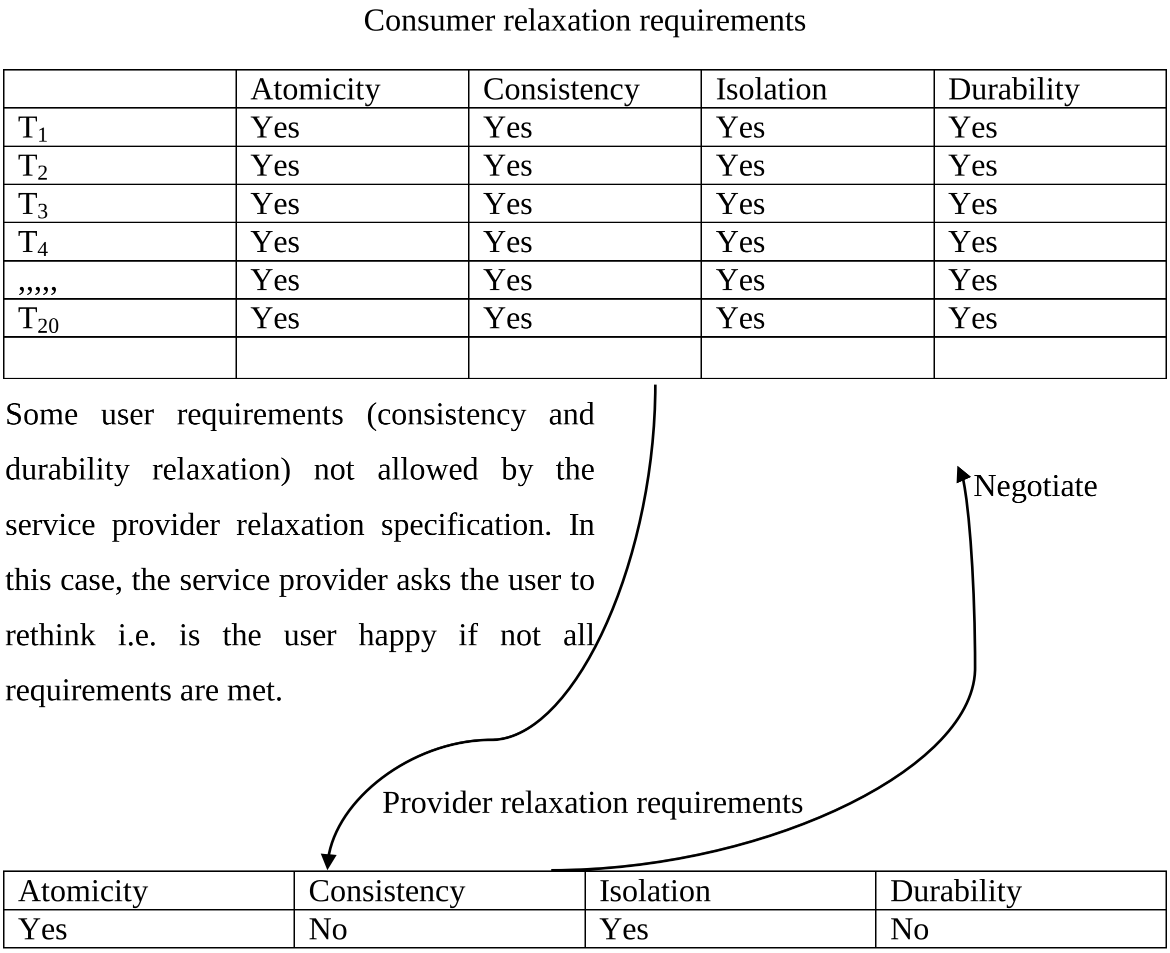
Database consistency is specified through consistency rules such as “the number of seats booked on plane X should not exceed the total number of seats on plane X”. In the case of replicated databases, a consistency requirement might be that the value of each replicated item is the same. Pre and post transaction rules might also be specified. For instance a precondition of ordering an item might be that the quantity of that item in stock is greater than 0. A post-condition might be that the quantity of that item in stock is 1 less than the quantity of that item in stock at the start of the transaction.

The service provider is responsible for consistency maintenance of the databases used. The consistency rules might be specified within the database but the service may be given the power to override them if the business considers this to be appropriate. It could be, however, that the database does not have a full set of consistency rules. The service provider can only maintain the rules that exist. That is, the service providers of component transactions have an agreement or “contract” with the database owner on what can be relaxed and what cannot.

A consistency definition in a database might be:

Consistency: A = number of resources (e.g. seats, capacity)
B = number of bookings
Consistency Requirement: B <= A

If relaxation of consistency is not permitted in the contract, consistency may not be relaxed by any subsequent transaction and the service provider will ensure this. However, if in the case of a particular application the business thinks it could be useful sometimes to relax a consistency requirement such as above, then the service provider is permitted to relax consistency. The service provider might then decide to relax rules like the above to increase throughput and improve business, for example thinking that even if stock is not available at the moment it can be procured later after the orders have been received. Thus the service provider in some circumstances will decide to relax consistency. In some circumstances the user may request that consistency be relaxed in order to increase throughput. In this case, the service provider would say yes or no depending on the back-end application requirement. If the data is such that consistency cannot be relaxed, then AuTrA will allow negotiation between the service provider and the user. The service provider will ask the user to rethink the requirement. If the user is prepared to accept that consistency cannot be relaxed, then the transaction can go ahead. Otherwise the transaction will be refused. The advantage of allowing negotiation is that it will lead to the acceptance of more transactions rather than just rejecting those where the consistency maintenance requirement varies between the consumer and the service provider. Figure 15 shows the relationship between consumer-specified requirements and provider relaxation specification in the context of AuTrA.



Consumer relaxation requirements

	Atomicity	Consistency	Isolation	Durability
T_1	Yes	Yes	Yes	Yes
T_2	Yes	Yes	Yes	Yes
T_3	Yes	Yes	Yes	Yes
T_4	Yes	Yes	Yes	Yes
,,,,,	Yes	Yes	Yes	Yes
T_{20}	Yes	Yes	Yes	Yes

Some user requirements (consistency and durability relaxation) not allowed by the service provider relaxation specification. In this case, the service provider asks the user to rethink i.e. is the user happy if not all requirements are met.

Provider relaxation requirements

Atomicity	Consistency	Isolation	Durability
Yes	No	Yes	No

Figure 15 Consumer and provider ACID relaxation specification

This mechanism of allowing the users to rethink the transaction requirements is useful for both the service provider and the user. That is, the service provider might not need to reject the transaction, resulting in costly compensation being avoided. On the other hand, in the situation where the user has chosen to relax atomicity, it means that the part of the transaction which does not violate the consistency requirements can commit even when other parts of the transaction have violated consistency requirements and need to be aborted.

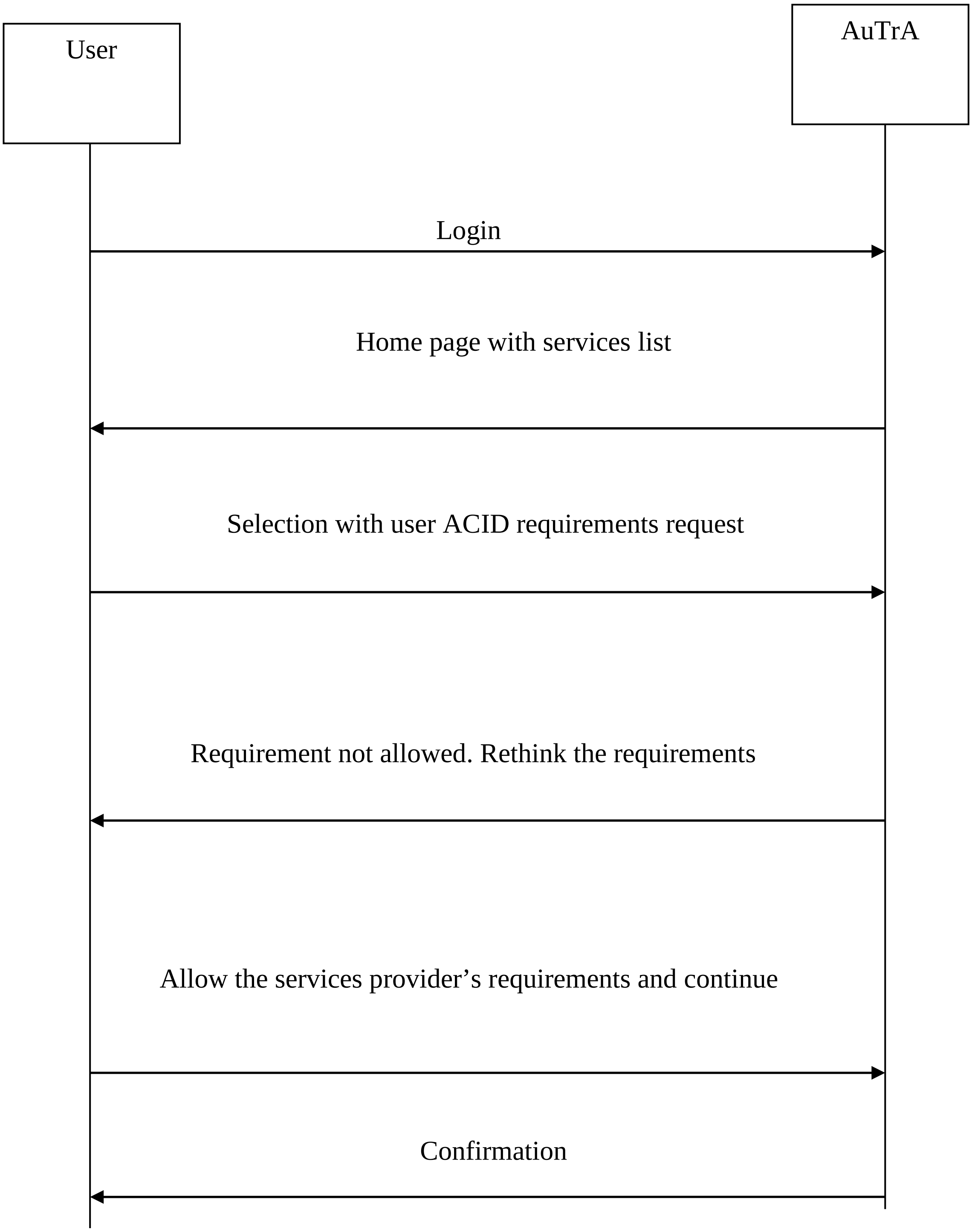

User
AuTrA
Login
Home page with services list
Selection with user ACID requirements request
Requirement not allowed. Rethink the requirements
Allow the services provider’s requirements and continue
Confirmation

Figure 16 Negotiation process

Additionally there are situations in which the user may not wish to relax consistency but the provider might. Note that atomicity and isolation are determined by the consumer; consistency and durability are determined by the provider. Consider the example shown in Figure 17: Jake's requirements include not relaxing consistency even if the provider is prepared to relax it. In this case, the user has the final say on what will be relaxed. The provider has the final say on consistency if the request is to relax consistency, but if the user requests that consistency be maintained, then it is maintained even if the provider was prepared to relax it.

Consumer relaxation requirements

Atomicity	Consistency	Isolation	Durability
No	No	No	No

Provider relaxation requirements

Atomicity	Consistency	Isolation	Durability
No	Yes	No	No

Figure 17 Non-matching consumer and provider relaxation requirements: Example 2

3.6 Adaptable durability

Data consistency is very important in business, and consistent data must be persistent at the end of the transaction. Nevertheless, there are some situations in which it is not necessary to save data. A system may have some mechanisms that assist in allowing full durability or relaxed durability. For full durability at the end of the transaction all the data has to be permanent. This is important in an area where the data needed is vital. Thus, not making the data permanent might make the whole business dangerous. For example, imagine that Doctor A at the end of the consultation does not save the information of Patient B, who is allergic to penicillin. After some weeks, Patient B goes to consult Doctor C. Doctor C prescribes penicillin to Patient B without knowing that it is harmful, because there is nothing in Patient B's record showing that; the transactions which were supposed to show that were not saved. In this scenario, durability must be strict.

However, there are some situations in which data does not need to be permanent. This is true in a situation where there is a large volume of data that needs to be processed a limited time. In this case, saving of data can be time-consuming, which is a drawback to business when fast transactions have to be processed. An example is in the case of sensors, where a lot of data will be coming in and, if the readings are coming very fast and saving is done, it can result in the whole process being slowed down. It might not matter if the writing of one or two readings is missed. Another example is the ski hire shop. Normally an order for ski hire comes in and it is written to the database and the user is issued with a slip to say the hire is agreed. If the system gets really busy and there is no time to update the database or if the database goes down, the shop may carry on taking orders and issuing agreements without updating the database. The owners may decide to update the database later when business is quieter using the agreement records sent by email, or even may decide not to update the database at all, hence relaxing durability altogether. In the latter case, the recording exactly of who has hired skis is not crucial to the business. Typically system mechanisms have two ways of not saving data, i.e. relaxing durability and not tidying up at the end, or relaxing durability and tidying it up at the end. By '*at the end'*, the time when everything is done and the server is not busy anymore is intended—in other words, when the server is free to do some more jobs. Then the system mechanism that allows the service providers to transfer the data from the memory into the permanent place might or might not make the data in the memory permanent, depending on the importance of durability to the application.

Similar to consistency relaxation, in AuTrA, the service provider owns relaxation of durability. The question that might arise when relaxing durability and not making the data changes permanent at the end is whether the data information is left inconsistent. This is could be the case. However, the system gives the service provider a choice of making the data consistent later (through delayed saving), and if the services provider chooses not to do so, it means that it is acceptable for the data to be left in an inconsistent way in that particular application. This latter situation would only arise if the consistency property is relaxed. The service provider can enforce consistency by not allowing the consistency property to be relaxed. However, relaxing durability even when maintaining consistency can still lead to inaccurate representation of the real-world situation. For example, a consistency requirement might be that the number of orders of Product A is less or equal to the number of Products A held in stock. Let us assume there are 20 instances of Product A in stock and orders for 15 of Product A. A new order comes in for 3 of Product A. This order does not break the consistency requirement as 18 is less than 20 and therefore is accepted but is not made durable. Thus the database still says there are only 15 orders for Product A, even though 18 orders have been accepted. This is an example of the database not representing an accurate picture of the real world even though the consistency requirement has been maintained.

Let us consider again how the consistency property can relate to the durability property. To illustrate the relaxation of durability, it is assumed that Perry decides to book 20 ski passes. The provider decides to relax durability and not to tidy up at the end of the transaction. The provider also does not relax the consistency requirement that before a booking is made there must be enough items in stock to satisfy the booking. Perry's booking results in nothing deducted being from the system, and coincidently there were only 20 tickets left, which means that when John comes to book 20 tickets, booking is possible and there will still be 20 tickets left afterwards as durability is still relaxed. The system allows the booking because there are tickets available according to the database and the consistency rule is not broken, as when the bookings are made there seems to be enough passes available. In this case, the maintenance of the consistency rule has not stopped an inaccurate representation of the real-world situation in the database.

3.7 Application-specific criteria

An additional characteristic of AuTrA is the use of application-specific criteria. Application-specific criteria concern the features that are unique to a particular application, as opposed to generic features that are applicable to any application. For example, when booking a restaurant one of the features that is not in a flight booking but is found in a restaurant booking is the type of cuisine. This feature is specific to the restaurant booking. On the other hand, there are specific aspects that are similar to both applications, like dates. Both the restaurant and flight bookings will be interested in dates, but they will be doing something different with regard to those dates, namely one will be interested in capturing features that can be attributed to food, such as cuisine type and food preference, and the other will be interested in capturing features to do with flights, such as seat preference and dietary needs.

The system allows the users to decide whether they want to use application-specific criteria or not in managing the transaction and in determining whether a component transaction should fail or not. The system is generic in such a way that it can cater for applications that have application-specific criteria or those without. The application-specific criteria are used when what the user wanted as a first choice is not available. The user has a choice of choosing the attributes that can be compromised and those that cannot be compromised. For example, the choice can be price of cuisine, type of cuisine, or the dates of the booking, so the user might say the dates have to stay as in the initial request. For example, the cuisine has to be vegetarian, no compromise, but anything else can be compromised to the best possibilities available. This means that a transaction that might otherwise fail will succeed when application-specific criteria are relaxed.

Application-specific criteria are based on the attributes that are specific to that application and that can be compromised. This means that if many attributes can be compromised, the read/write sequences will be complex. For example, John, who is booking a hotel and flight, might have a one application-specific criterion that can be compromised, which is date for hotel. Similarly, Kate might book the same resource like John, but for her, she might be prepared to compromise both dates of flight and hotel. Because there is extra reading and writing on Kate’s transaction (due to her transaction continuing because she has offered greater flexibility), Kate’s processing requires more time compared to John’s.

Having said that it is worth noting that Kate’s transaction is more flexible and the chances of her transaction failing because of services providers’ not meeting the requirements is less than that of John’s transaction.

3.8 Summary

This chapter has described the motivational concepts behind AuTrA. It delineates relaxation of each ACID property as provided in AuTrA. In AuTrA, relaxation of ACID properties is adaptable according to consumer requirements and services providers' relaxation specifications. An additional feature to ACID properties that has been introduced in AuTrA is application-specific property relaxation. Correctness of the data is defined by the service provider through the specification of the requirement of consistency.

Chapter 4: Presentation of AuTrA

4.1 Introduction

In this chapter, the architecture of the proposed system, AuTrA, is explained. AuTrA is the system developed by this research to evaluate whether customisable transaction support in the Web services environment can be useful. It has been implemented in ASP.Net C# and runs in Microsoft Visual Studio 2008 on an SQL server on an IIS Web server. It can be regarded as middleware, as it forms a layer between the service consumers' applications and the service providers' services. An outline of each AuTrA component is also given as well as an account of the process flows in the AuTrA system for service provider, service consumer and application developer. The notation used in the workflow diagrams follows the UML standard (Fowler 2003). Some screenshots illustrating parts of the implemented system are also given.

4.2 System overview

The structure of AuTrA is given in Figure 18, while Figure 19 shows the context of AuTrA . AuTrA has three types of user: a service provider, a service consumer, and an application developer/tester. The service provider provides services to AuTrA that can be offered to service consumers. A service consumer uses the services provided and may compose and run applications based on them. The application developer builds applications for service consumers by combining services. The application developer helps consumers specify appropriate relaxation criteria. The application developer may also test various relaxation criteria by creating batch files of transactions and running them in AuTrA and analysing the results. This will help the developer make appropriate judgement on which criteria to relax. The roles of service consumer and application developer overlap in that both may compose applications based on services.

Let us consider the composition of AuTrA. AuTrA has an interface for service consumers, service providers and application developers. It also has a core transaction management layer which consists of the following components: Reader, Requirements Tailor, Requirements Negotiator, Batch Manager, Processing Timer and Writer.

Requirements Tailor: This component will check whether the user's ACID relaxation request fits with the service provider's requirements.

Requirements Negotiator: This component will give the consumer an opportunity to re-specify the transaction requirements.

Batch Manager: This component is responsible for the main processing of the applications. In AuTrA, transactions can be run singly or in batches. The batch manager handles both modes and runs the transactions according to the specifications set, for instance using concurrent or non-concurrent processing or maintaining or relaxing the various criteria. The batch manager launches the applications and raises the calls to the Web services used. It also communicates with the other components and thus forms the central hub of AuTrA.

Reader: This component is responsible for reading from any input objects which include the batch files or online input devices.

Processing Timer: This component calculates the processing time of the transactions.

Writer: This component writes the output of the processing. In the case of a batch run, the output will be a file showing the outcome of all transactions in the run and the processing time. In the case of an individual transaction run, the output is written to the screen.

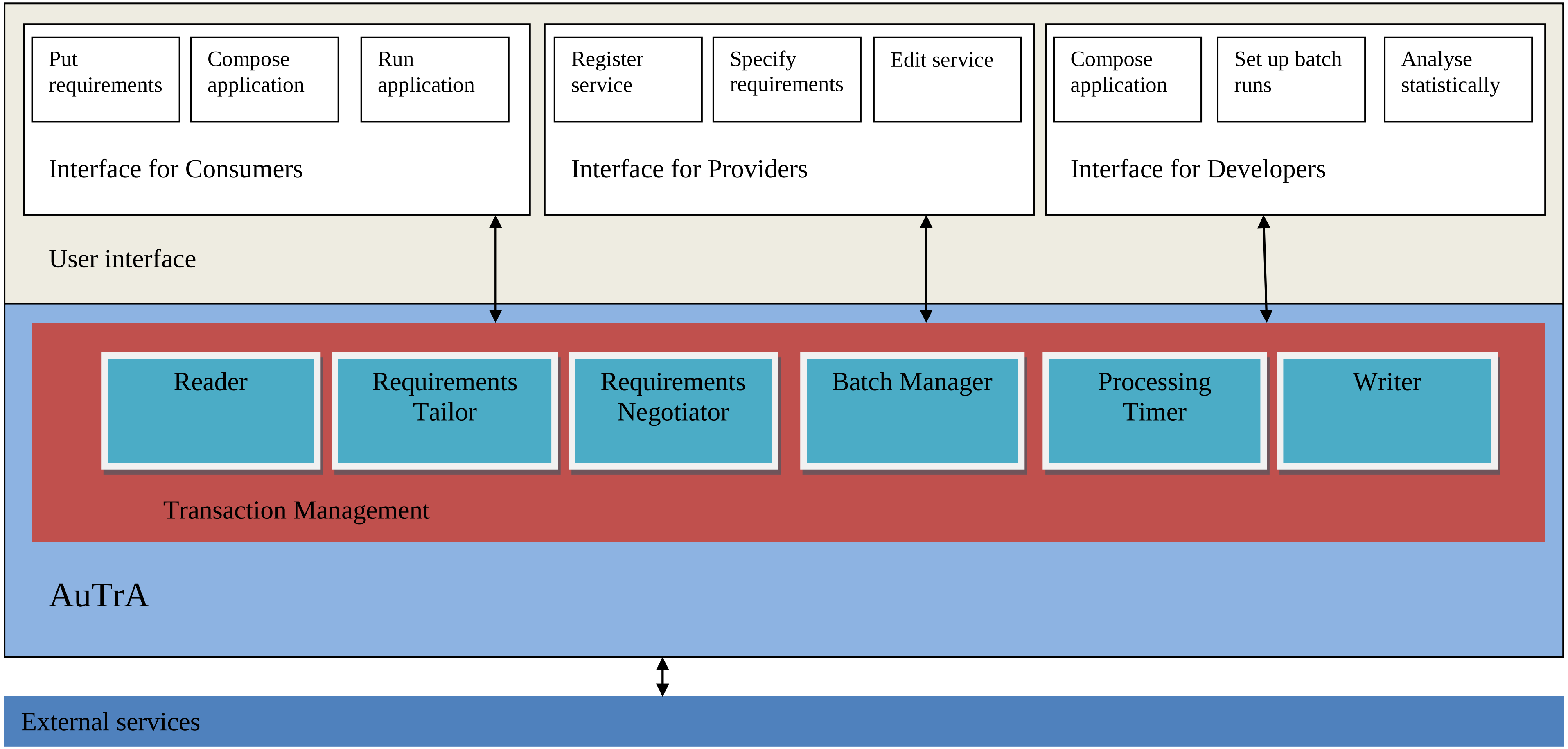


Figure 18 High level system diagram for AuTrA

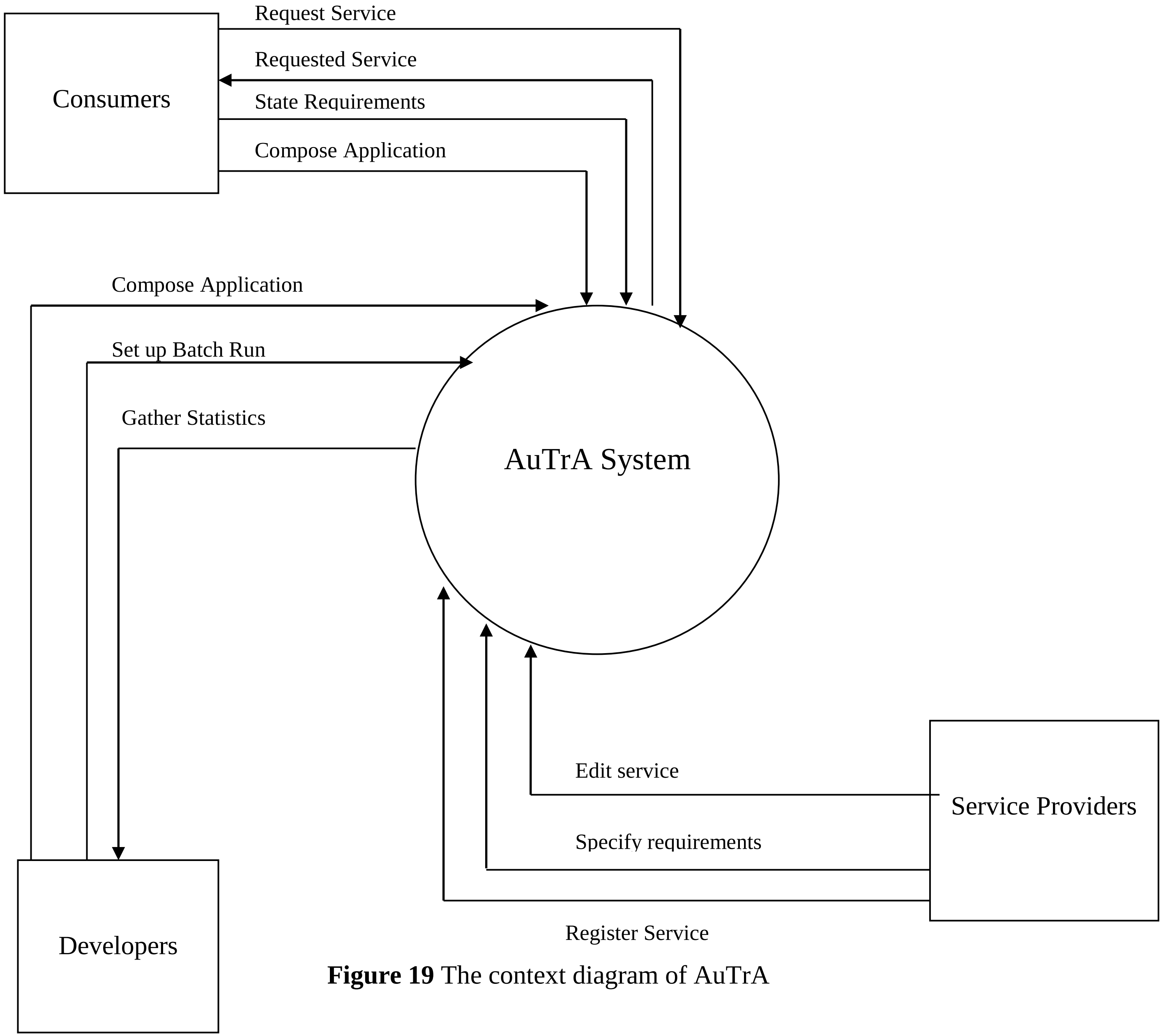


Figure 19 The context diagram of AuTrA

Let us consider the system workflows from the perspectives of each of the user categories. The proposed system workflow from a service consumer's perspective is presented in Figure 20, from the service provider's perspective in Figure 21 and from the developer's perspective in Figure 22.

From a service consumer's perspective, AuTrA has the following phases (see Figure 21):

Input User Requirements phase: This is where the consumer will have to put in the requirements of the service they request through a Graphical User Interface (GUI). After the users have put in what they want, the requirement will go to the Tailor Requirements Phase.

Tailor Requirements Phase: The system takes the consumer's requirements and checks that they satisfy the business requirements or criteria of the component services. For example, the business requirements or criteria might be to not relax durability and consistency. This phase will check the user's ACID specified request fits with the service provider's requirements.

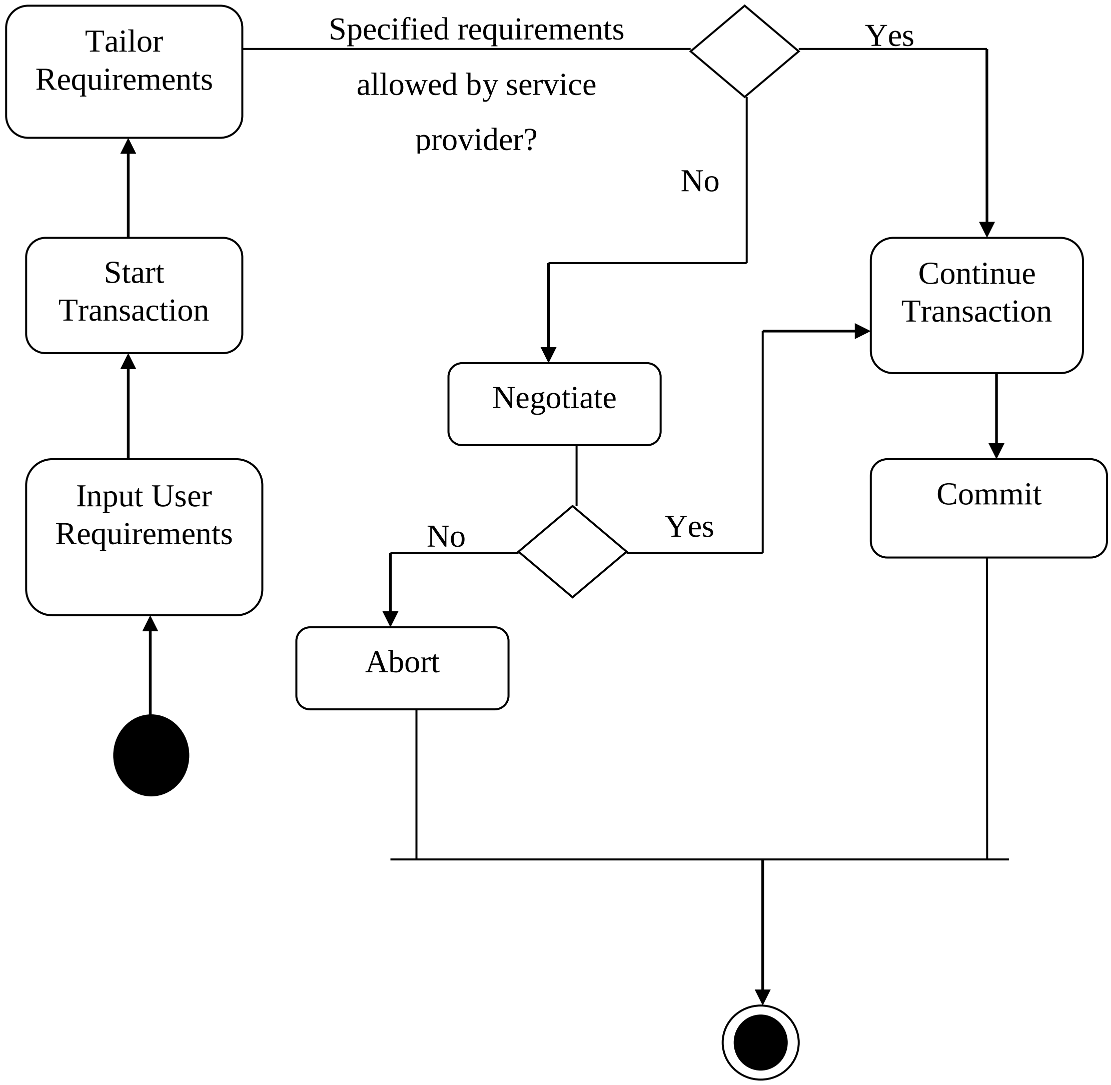


Figure 20 Proposed AuTrA system workflow: Consumer's perspective

Negotiation phase: Transactions that did not manage to go to the Continue process from the Tailor Requirements process will come to the Negotiation process. The user will be allowed an opportunity to re-specify their transaction requirements.

Commit phase: This is the process in which the transaction is committed, the data is saved, and confirmation is provided to the user. If durability is relaxed, data saving is not required. That is to say, the committed data is not made permanent, and therefore working against the classic characteristics of ACID transactions.

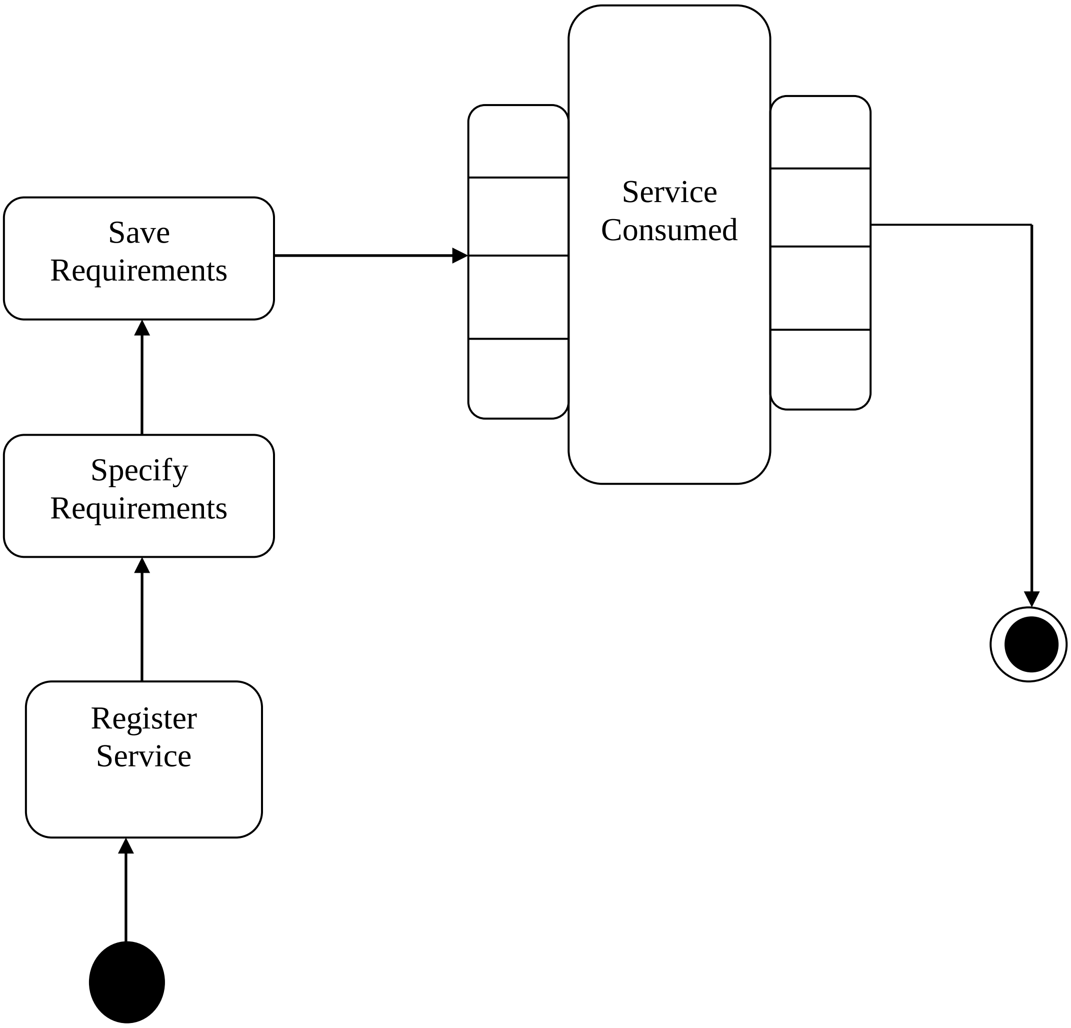


Figure 21 Proposed AuTrA system workflow: Service provider's perspective

From a service provider's perspective, AuTrA has the following phases (see Figure 21):

Register Service phase: This is where the providers who want to use AuTrA will register their service.

Specify Requirements Phase: This is where the service provider will have to put in the business requirements of the service they provide through a graphical user interface (GUI). After the services provider has put in required business requirements, the requirements will go to the Save Requirements phase.

Save Requirements Phase: This is the process in which the required business requirements are saved.

Service Consumed Phase: After the service has been set up it can be used repeatedly by consumers.

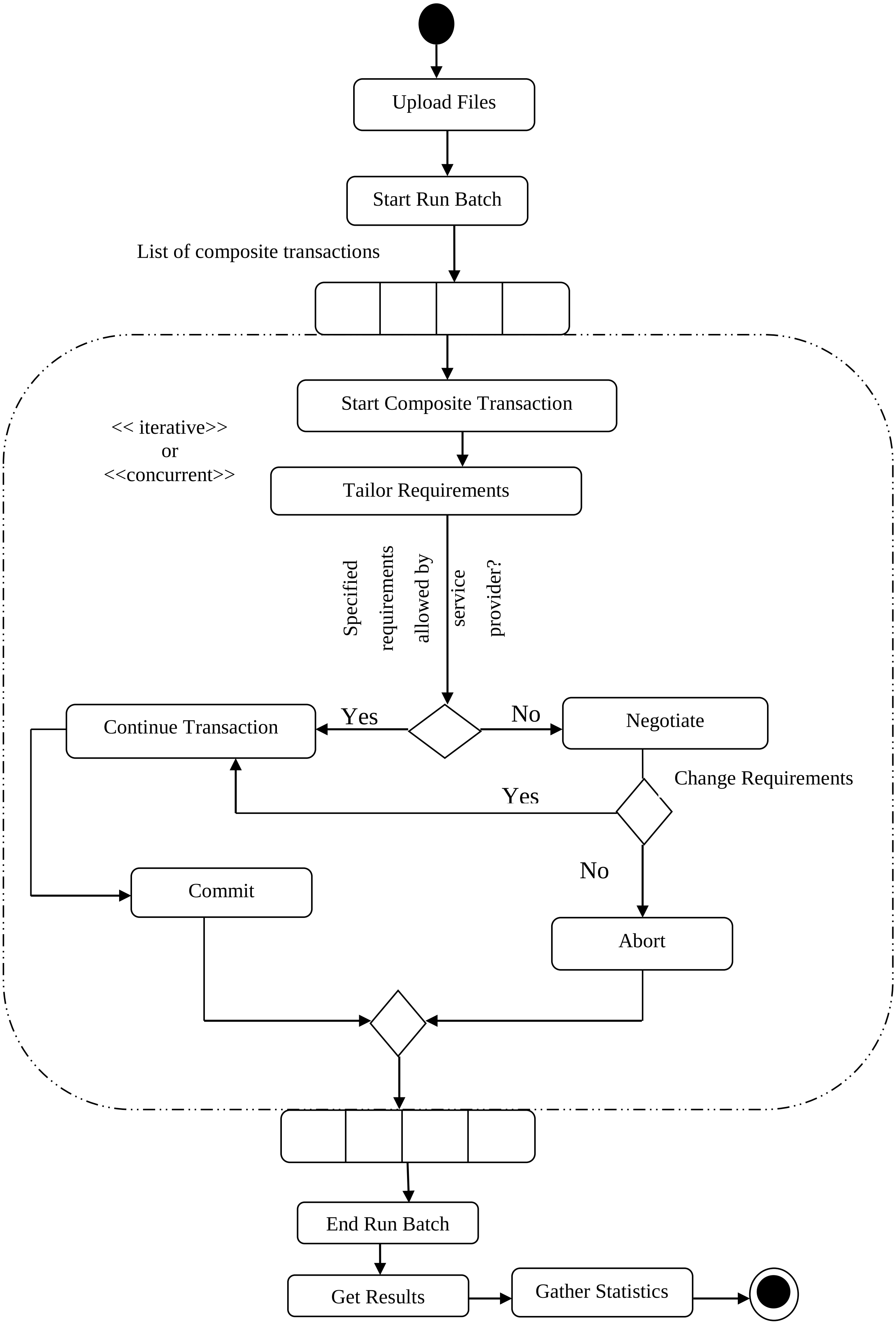


Figure 22 Proposed AuTrA system workflow: Developer’s perspective

Figure 22 shows the AuTrA system workflow from a developer’s perspective. In Figure 22 the main batch processing is shown as iterative or concurrent as the mode of processing will depend upon the relaxation specified. If isolation is relaxed the processing will be concurrent. Otherwise the processing will be iterative. From an application developer’s perspective, AuTrA has the following phases:

Upload File Phase: In this phase the developer will upload the files that are used to experiment with different relaxation choices.

Run Batch Phase: This phase is when the uploaded batches are run. During this process transactions will be committed or aborted and a record will be maintained of which transactions are successful and which are not, together with timings.

Get Results Phase: The results of the batch run will be obtained which will show which transactions were successful and which were not, as well as the processing times for the transactions. The results will be available for both composite and component transactions.

Gather Statistics Phase: The output of the batch transaction will be gathered as raw data and this will be statistically analysed. The statistically analysed data will help developers who are building the application to appropriately advise the users about relaxation.

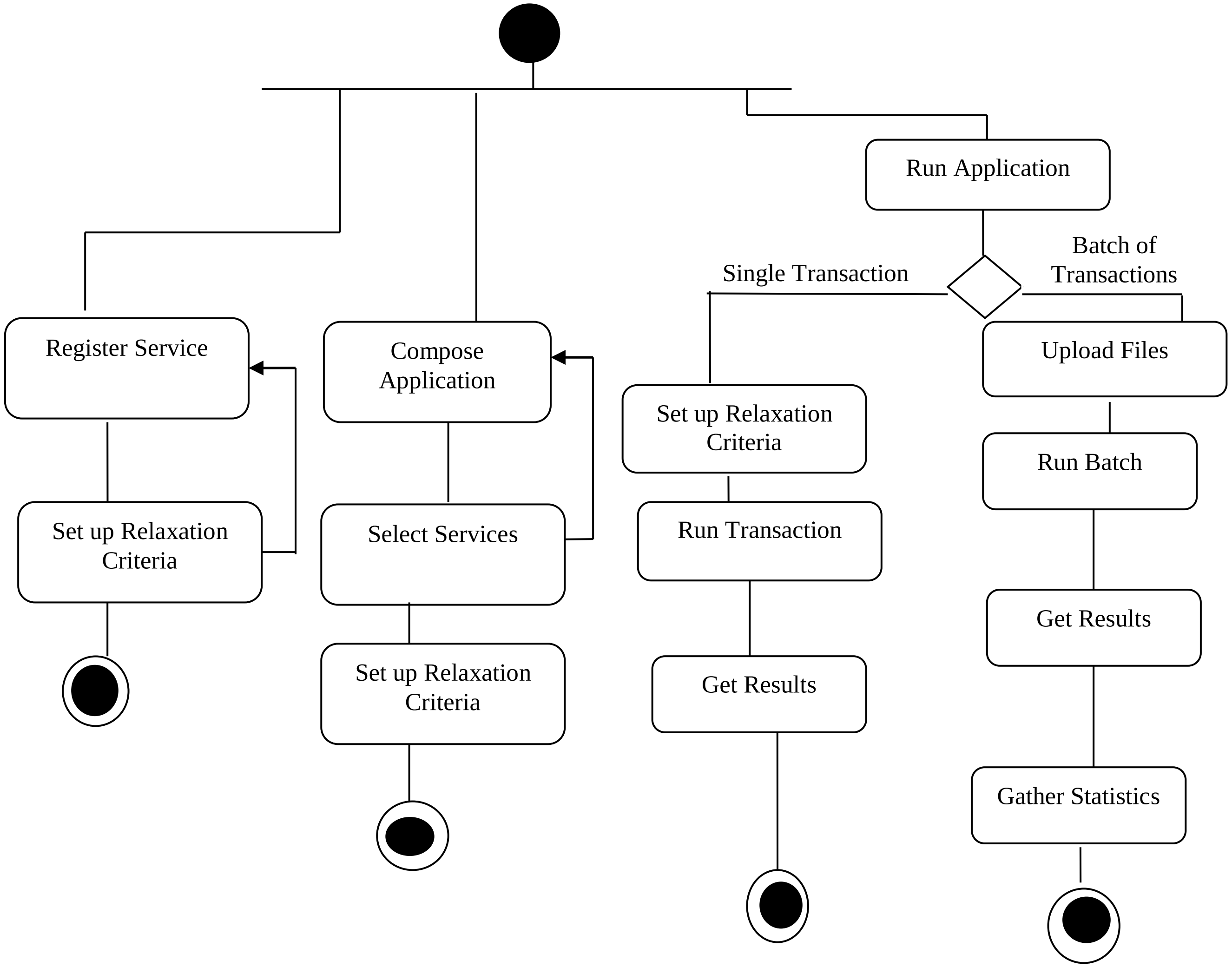


Figure 23 Detailed process flow of AuTrA system

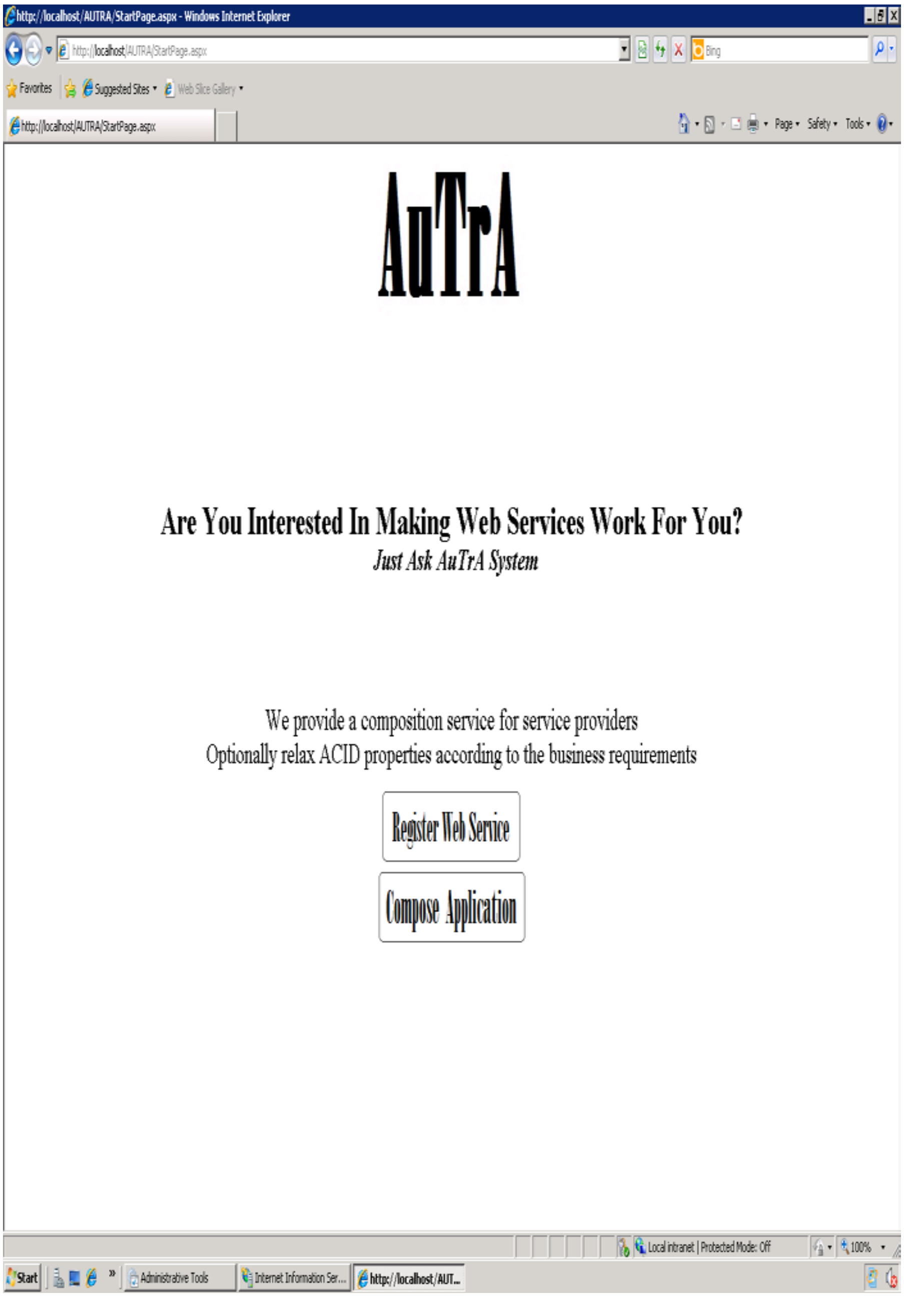
Figure 23 represents the complete workflow of AuTrA system workflow from all user perspectives. It can be seen that the system allows two forms of interaction: real-time or batch-processing-based. The real-time mode would be used by individual consumers online when requesting a service or composing an application. Batch processing would be used when, say, a company wished to process many transactions at an off-peak time. Batch processing may also be used by application developers for experimental purposes. In order to evaluate the AuTrA model, this research used batch processing to mimic a real situation, where a lot of users process transactions concurrently (see Chapter 5) .In this case AuTrA was configured to have random delays in launching transactions from a batch (to mimic live multi-user situations). AuTrA was also configured to allow simulated interaction with consumers for purposes of negotiation when there was a mismatch in relaxation specifications.

4.3 AuTrA implementation

AuTrA been implemented in ASP.Net C# and runs in Microsoft Visual Studio 2008 on an SQL server on an IIS Web server. AuTrA offers the following functionality:

- Service Registration
- Application Composition (by selecting registered services)
- ACID property relaxation specification by user
- ACID property relaxation specification by provider
- Application-specific criteria relaxation by user
- Negotiation of requirements
- Application execution coordination and monitoring
- Running of batch files of transactions
- Providing timings for transaction batch runs

Let us consider the service provider. The front page of the implemented AuTrA system for the service provider is shown in Figure 24.


http://localhost/AUTRA/StartPage.aspx - Windows Internet Explorer
http://localhost/AUTRA/StartPage.aspx
Bing
Favorites
Suggested Sites
Web Slice Gallery
http://localhost/AUTRA/StartPage.aspx
Page
Safety
Tools
AuTrA
Register Web Service
Compose Application
Local intranet | Protected Mode: Off
100%
Start
Administrative Tools
Internet Information Ser...
http://localhost/AUT...

Are You Interested In Making Web Services Work For You?

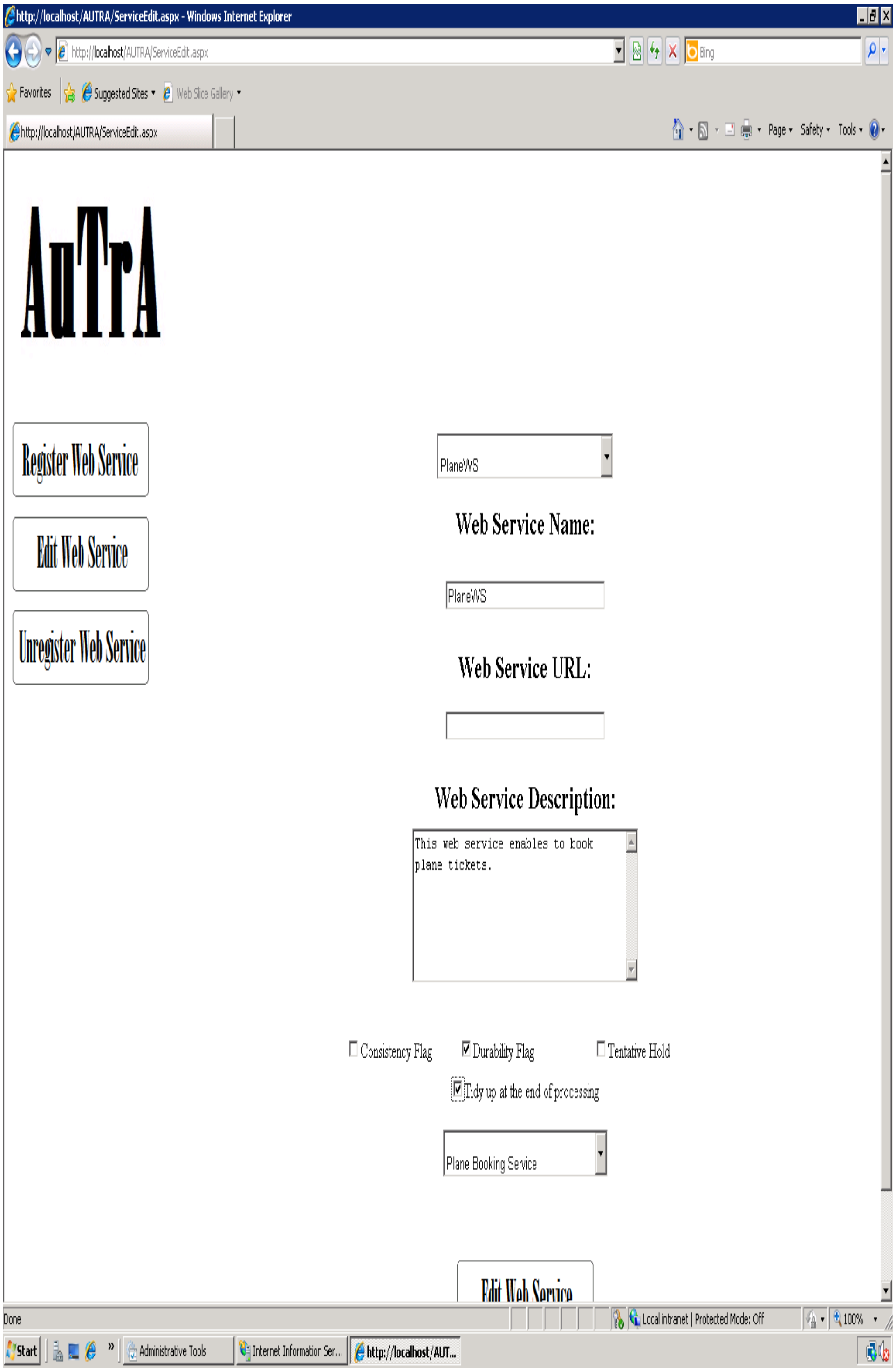
Just Ask AuTrA System

We provide a composition service for service providers
Optionally relax ACID properties according to the business requirements

Figure 24 AuTrA service provider front page

Figure 24 shows that when the service provider logs in they are able to register a new service, edit a service or delete a service. When choosing to register a new service the provider will have to provide the name of the service, the URL and the service description, which will give the user an overall picture of the service provided. Furthermore, because the provider is responsible for deciding whether or not to relax durability and consistency, the provider will have to specify its position on these items.

The position of the provider on these matters could in turn depend on the contracts that it has with its database owner clients. The provider can choose to edit the services according to their needs. For example, imagine that at the time the service provider first registered their services, durability and consistency could be relaxed. This position was consistent with the contracts the provider had with its database owner clients. Nevertheless, as time went on they realised that the relaxation of consistency was no longer acceptable, that only durability could be relaxed; the service provider can then edit the consistency from relaxed consistency to strict consistency. Additionally the provider can choose whether to tentatively hold the user transaction when atomicity and isolation are relaxed by the user. When durability is relaxed the service provider has a choice of tidying up the database at the end or not, as discussed in Section 3.5. The point being made here is that the service provider can maintain consistency and durability when these are essential to the needs of the business and can relax them when they are not. As shown in Figure 25, the service provider has a choice of relaxing consistency and durability (tidying up at the end or not), and using tentative hold. Tentative hold is used to maintain atomicity and to maintain consistency.



http://localhost/AUTRA/ServiceEdit.aspx - Windows Internet Explorer
http://localhost/AUTRA/ServiceEdit.aspx
Bing
Favorites
Suggested Sites
Web Slice Gallery
http://localhost/AUTRA/ServiceEdit.aspx
Page
Safety
Tools
AuTrA
Register Web Service
Edit Web Service
Unregister Web Service
Edit Web Service
Done
Local intranet | Protected Mode: Off
100%
Start
Administrative Tools
Internet Information Ser...
http://localhost/AUT...

PlaneWS

Web Service Name:

PlaneWS

Web Service URL:

Web Service Description:

This web service enables to book plane tickets.

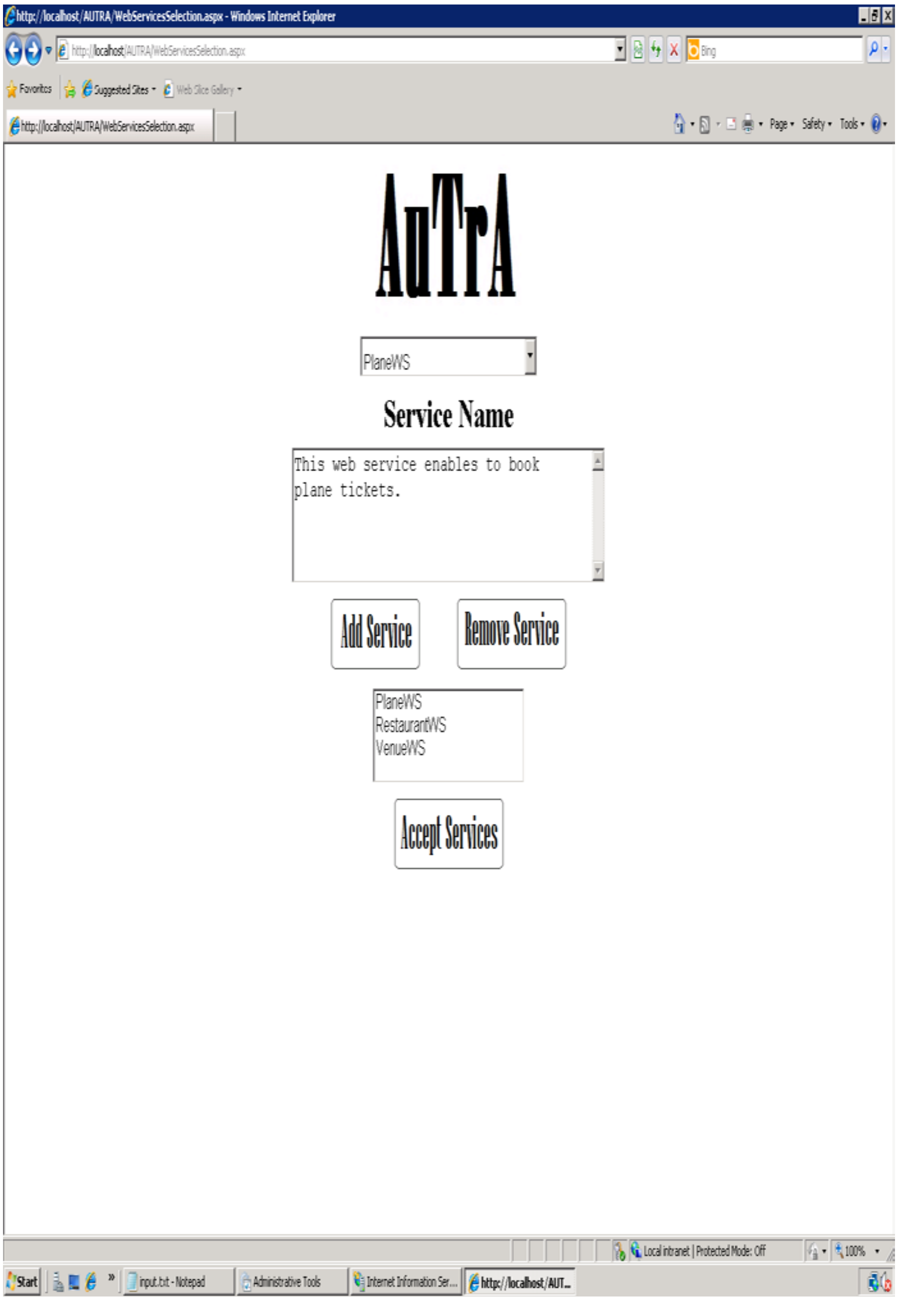
□ Consistency Flag ☑ Durability Flag □ Tentative Hold

☑ Tidy up at the end of processing

Plane Booking Service

Figure 25 Service provider options in AuTrA

Let us now consider the service consumer. The service consumers can choose to compose an application, adding services that they wish to include in the composition. Then, after composing application according to requirements, the consumer accepts the composite service. A composite Web service is a combination of individual services the consumer wants to use in a transaction. For example, in Figure 26, Sarah might choose to have a transaction that involves booking a flight to go to Spain for a hen party, a venue for the hen party and the restaurant to have a meal before the hen party.


http://localhost/AUTRA/WebServicesSelection.aspx - Windows Internet Explorer
http://localhost/AUTRA/WebServicesSelection.aspx
Bing
Favorites
Suggested Sites
Web Slice Gallery
Page
Safety
Tools
AuTrA
Add Service
Remove Service
Accept Services
Local intranet | Protected Mode: Off
100%
Start
input.txt - Notepad
Administrative Tools
Internet Information Ser...
http://localhost/AUT...

PlaneWS

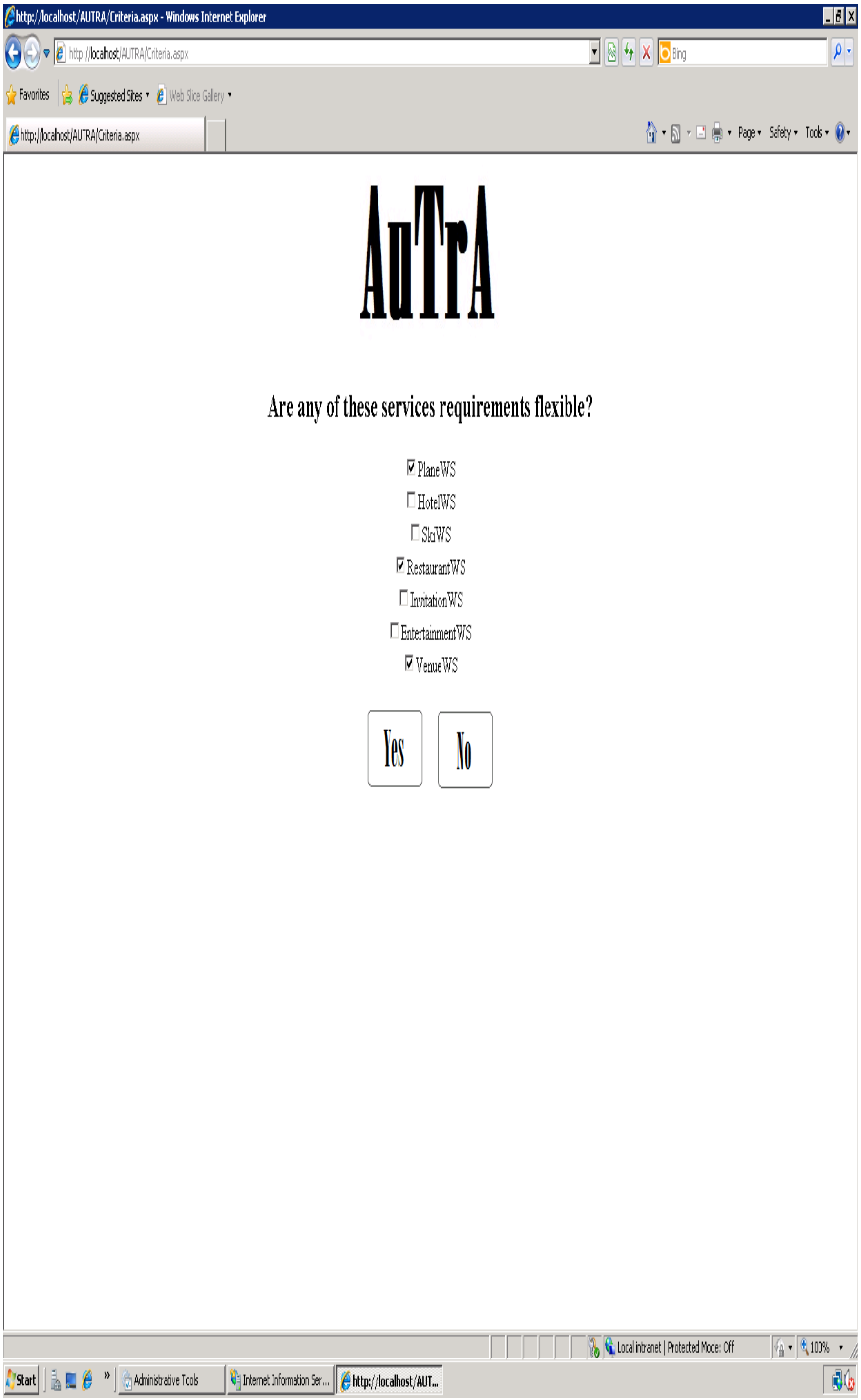
Service Name

This web service enables to book plane tickets.

PlaneWS
RestaurantWS
VenueWS

Figure 26 Application composition by service consumer

Service consumers may also choose application-specific criteria to be relaxed. Using the above scenario of Sarah's trip, Sarah will be able to choose the application-specific criteria that can be relaxed. Figure 27 shows how AuTrA offers Sarah a chance to relax some requirements.


http://localhost/AUTRA/Criteria.aspx - Windows Internet Explorer
http://localhost/AUTRA/Criteria.aspx
Bing
Favorites
Suggested Sites
Web Slice Gallery
http://localhost/AUTRA/Criteria.aspx
Page
Safety
Tools
Local intranet | Protected Mode: Off
100%
Start
Administrative Tools
Internet Information Ser...
http://localhost/AUT...

AuTrA

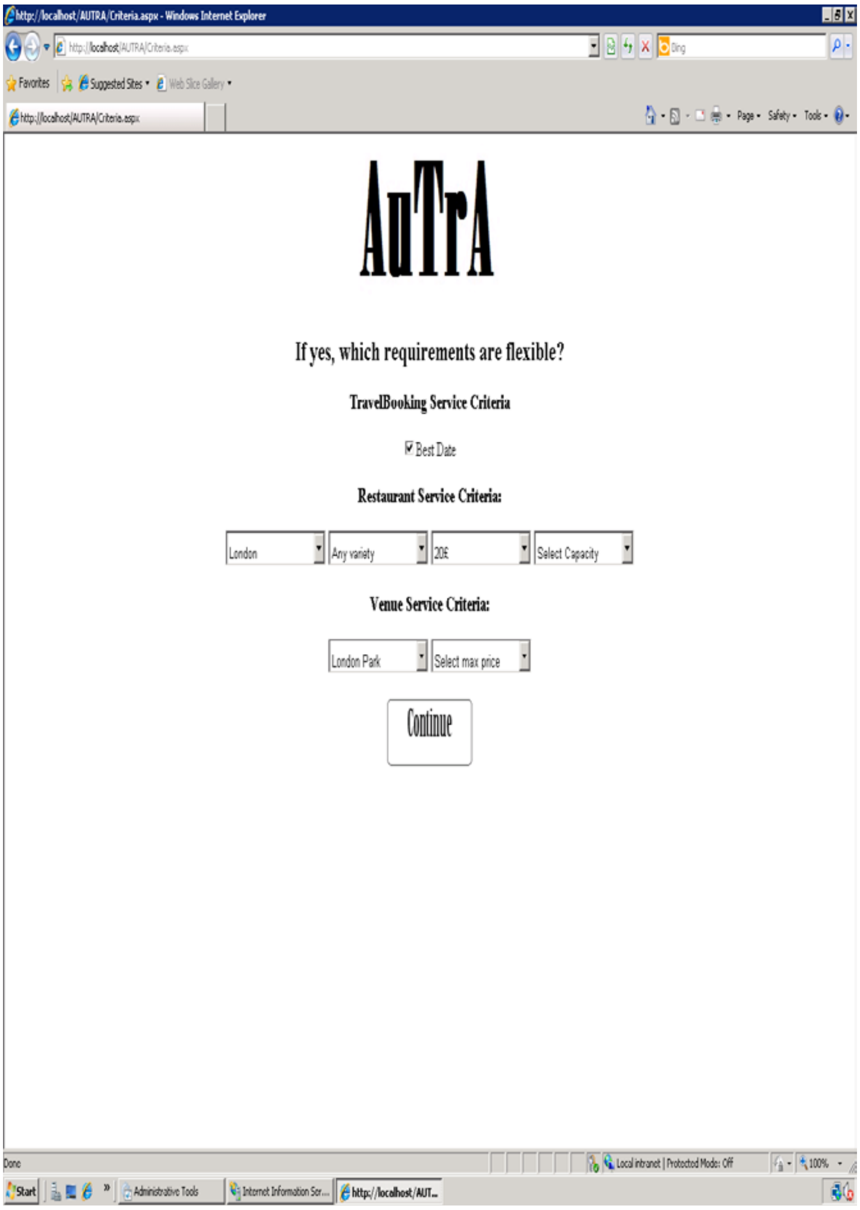
Are any of these services requirements flexible?

☑ PlaneWS
☐ HotelWS
☐ SkiWS
☑ RestaurantWS
☐ InvitationWS
☐ EntertainmentWS
☑ VenueWS

Yes No

Figure 27 Application-specific criteria relaxation

Note that Sarah had a choice of services that can allow application-specific criteria to be compromised. Even in those services that she has allowed to be compromised, it does not mean that all the application-specific criteria in those services will be compromised. She will choose which requirements she wants to be compromised. For example, Figure 28 illustrates the selection of criteria which can be flexible when running a particular application. The flexibility is determined by the service consumer according to own requirements.



http://localhost/AUTRA/Criteria.aspx - Windows Internet Explorer
http://localhost/AUTRA/Criteria.aspx
Bing
Favorites
Suggested Sites
Web Slice Gallery
Page
Safety
Tools
AuTrA
Done
Local intranet | Protected Mode: Off
100%
Start
Administrative Tools
Internet Information Ser...
http://localhost/AUT...

If yes, which requirements are flexible?

TravelBooking Service Criteria

☑ Best Date

Restaurant Service Criteria:

London | Any variety | 20£ | Select Capacity

Venue Service Criteria:

London Park | Select max price

Continue

Figure 28 Application-specific criteria selection

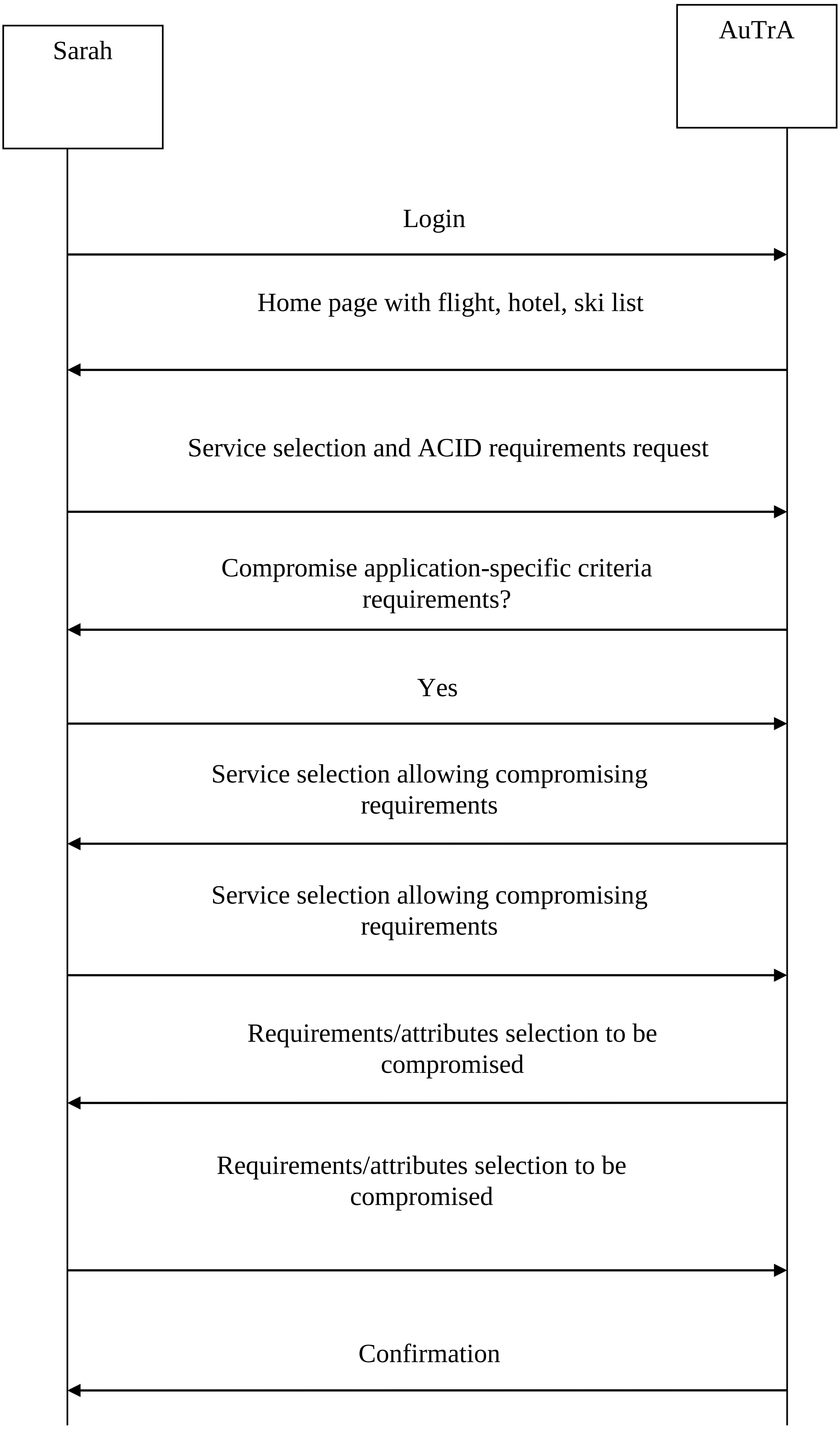
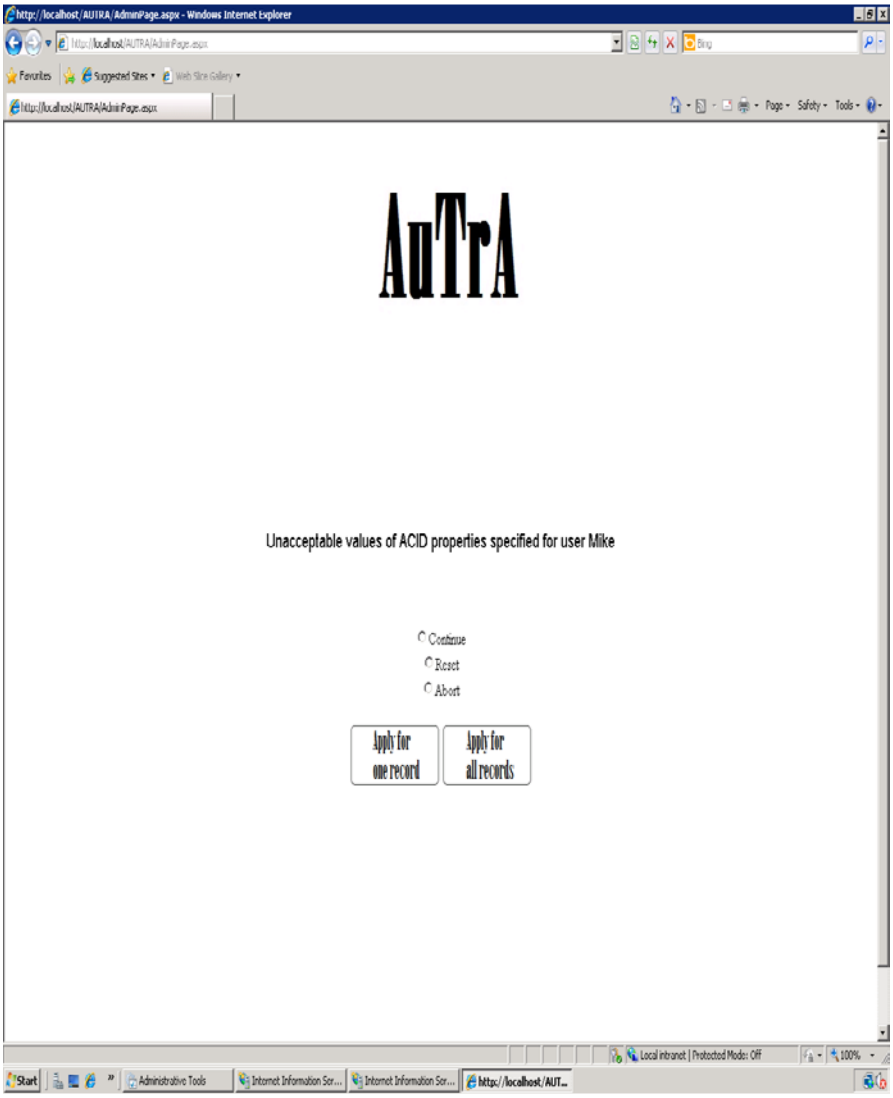

Sarah
AuTrA
Login
Home page with flight, hotel, ski list
Service selection and ACID requirements request
Compromise application-specific criteria requirements?
Yes
Service selection allowing compromising requirements
Service selection allowing compromising requirements
Requirements/attributes selection to be compromised
Requirements/attributes selection to be compromised
Confirmation

Figure 29 Application-specific criteria process

After the service consumer has specified all relaxation requirements, the application is run under the control of AuTrA. The processor checks whether the service provider allows what the consumer wants to relax in terms of consistency and durability. If allowed, the consumer’s transaction will finish with a summary of the transaction details. This process is shown in Figure 29. However, if not allowed, the transaction will go to a negotiation phase where the consumer is given a choice to re-specify the requirement, continue with the transaction, or abort. Keep in mind that the only thing Sarah can choose to change when re-specifying the requirement is the relaxing of atomicity or isolation. The others are the provider’s choice, or she must abort to restart the transaction. If she continues, the service provider’s requirements will be followed.

As mentioned above, AuTrA allows for some negotiation to take place when a user’s requirements are not in accord with the service provider’s criteria for consistency and durability. Let us consider an example. Mike has some requirements not allowed by the service provider. This means that Mike might have asked to relax consistency or durability and the services provider did not relax consistency and durability or either of them. In this situation the system has a mechanism that will respond to Mike to inform him that his request is not allowed. Mike has a choice of continuing — in this case his requirements will not be followed where they clash with those of the provider — or resetting his requirements to fit those of the service provider, or aborting (see Figure 30).



Unacceptable values of ACID properties specified for user Mike

Figure 30 AuTrA negotiation

4.4 Main classes in AuTrA

This section provides an overview of the main classes and methods used in AuTrA. The classes and methods can be grouped according to the main transaction management components in AuTrA (see Figure 18 in section 4.2, System overview). These components are Reader; Requirements Tailor; Requirements Negotiator; Batch Manager; Processing Timer; and Writer. The following sections provide more detail of the classes used in AuTrA.

4.4.1 Reader component

This component is responsible for opening and closing files as well as reading all inputs to AuTrA. The launch of the transaction is also done in this component. The launch is done with the communication between the ***StringParser*** class and the ***PositionableStreamReader*** class.

Class**: *File Reader***

Methods:

- void openFiles() – opens input file for each registered service.
- void closeFiles() – closes files opened by openFiles().
- User parseLine() – parses one line from each input file, creates new User object from obtained data and returns it.
- Boolean hasFinishedReading() – returns true if any of input file readers reached end of the file, otherwise returns false.
- void initializeParsers() – creates set of parsers corresponding to registered services.

Class: ***InputFormData***

Methods:

- void setName(String n)
- void setSurname(String s)
- void setAdd1(String a)
- void setAdd2(String a)
- void setAdd3(String a)
- void setMobile(String m)
- void setEmail(String e)

This class collects the consumer's information from the input form. Each service has its own specific implementation of this class.

Class: ***StringParser***

Methods:

- User parseUserDetails(PositionableStreamReader streamReader) – parses user information from input file and creates new User object which is returned.
- UserData parseLine(ref PositionableStreamReader streamReader, long id) – parses one line of the input file and creates UserData object from the obtained data.

Each service has its own implementation of the ***StringParser*** object.

Class: ***PositionableStreamReader***

Methods:

- PositionableStreamReader(String path) – position the reader in the full path of where the reading is supposed to take place.
- String readLine() – read the consumers input line by line and return the line.

4.4.2 Requirements Tailor component

This component is responsible for tailoring the consumer requirements according to the providers where necessary.

Class: ***ManagerImpl***

Methods:

- void checkFlags() – checks set of ACID properties requirements specified in the input file against the ones supported by chosen services.

4.4.3 Requirements Negotiator component

This component will handle negotiation in cases when the consumer requirements are not allowed by the provider. The consumer will be given a choice of rethinking the requirements.

Class: ***ManagerImpl***

Methods:

- void agreeServicesFlags() – agrees set of ACID properties requirements specified for chosen services.
- void agreeInputFlags() – agrees set of ACID properties requirements specified in the input files.

4.4.4 Batch Manager component

This component handles the batches and the online interaction. In the case of the online interaction, the running of the composite transaction is treated as if in a batch of 1.

Class: ***ManagerImpl***

Methods:

- void runManager(String mode) – starts manager operation in separate thread. The manager operation oversees the execution of each thread of work, where each thread of work represents a composite transaction.
- void doWork(object modeObj) – coordinates reading, buffer management and writing, and is called by runManager.

Class: ***Executer***

Methods:

- void executeUsersRIsolationFalse(List<User> listOfUsers, String mode) – executes calls to Web services for each user in the listOfUsers.
- void tentativeHold() – hold the resource for a certain amount of time.

Class: ***WebServicesCaller***

Methods:

- Boolean callWebServices(ref User user, String mode, ref List<UserData> servicesToRollback, ref List<UserData> notBookedServices) – executes each chosen service and saves information on which services completed successfully and which did not. The method returns true when all the services (component transactions) for a user's composite transaction have been executed.

Class: ***WebServicesConcurrentCaller***

Methods:

- void callWebServices(List<User> usersList, String mode, ref List<UserData> servicesToRollback, ref List<UserData> notBookedServices) – invokes callWebCaller() concurrently.
- void callWebCaller(Object user) – creates new instance of WebServicesCaller and invokes callWebServices() on it.

Class: ***BufferHandler***

Methods:

- void appendUser(User user) – adds user object to the buffer, generating random delay first if needed and flushing the buffer when full. The user object represents a composite transaction.
- void generateRandomDelay(int low, int high) – generates random delay within given range.
- void flush() – processes contents of the buffer, writes the output.
- void clearDictionary() – clears contents of the buffer.

Class: ***User***

Methods:

- List<UserData> userDataList() - lists the userdata for the user. The user data list shows which component transactions comprise the composite transaction and shows the relaxation requirements.

Class: ***UserData***

Methods:

- Boolean getrConsistencyFlag( ) – this will return true when the user requests to relax consistency.
- Boolean getrDurabilityFlag( ) – this will return true when the user requests to relax durability.
- Boolean getrAtomicityFlag( ) – this will return true when the user requests to relax atomicity.

- Boolean getrIsolationFlag( ) – this will return true when the user requests to relax isolation.
- Boolean getcriteriaFlag() – this will return true when the user compromises application-specific criteria.
- void setrConsistencyFlag(String s) – this sets the flag for relaxing consistency.
- void setrDurabilityFlag(String s) – this sets the flag for relaxing durability.
- void setrAtomicityFlag(String s) – this sets the flag for relaxing atomicity.
- void setrIsolationFlag(String s) – this sets the flag for relaxing isolation.
- void setcriteriaFlag(String s) – this sets the flag for application-specific criteria.

Each service has its own implementation of the ***UserData*** object

Class: ***Criteria***

- Abstract class which keeps information about the selected criteria; each service has its own implementation of this class.

Class: ***CriteriaUtil***

Method:

- void applyCriteria(ref UserData uData) – applies chosen criteria overriding ones specified in the input file.

4.4.5 Processing Timer component

This is responsible for obtaining high precision of time measurements of the transaction processing time.

Class: ***HiPerfTimer***

Methods:

- void start() – starts the timer when the transaction begins.
- void stop() – stops the timer according delay time set.
- double duration() – returns the duration of the transaction processing.

4.4.6 Writer component

This component writes the output of the consumers request to an output object. The output includes the outcome for each composite transaction, showing which component transactions were successful and which were not. The output also includes the transaction processing time.

Class: ***FileWriter***

Methods:

- void write(List<User> userList) – writes to the output file information about processing outcome for each User object from userList.
- void addTime() – adds to the output file information about the processing time.
- void writeOutput(Dictionary<int, List<User>> users) – writes processing outcome for each entry in the buffer to the output file. Dictionary is a system-defined name for the buffer.

4.5 Summary

AuTrA is a system that allows service providers to offer services to consumers. Consumers may combine these services to make applications or composite services. Providers may relax the ACID properties of consistency and durability. The consumer may relax the ACID properties of atomicity and isolation and may also request to relax consistency and durability. Consumers can also relax application-specific criteria. After composing the application from available services, the consumers can either fill in online forms to make an individual service request or use the batch processing mode, in which a user can upload a set of input files and in that way many service requests may be processed together. The next chapter describes the simulation model that was used to test the AuTrA model and system implementation.

Chapter 5: Simulation Model and Evaluation Strategy

5.1 Introduction

In this chapter the research discusses the key definitions, presents the simulation model and the simulation road map which leads to experiments and results presented in Chapter 6.

5.2 Definitions of the key terms in the research

This research uses the key terms *transaction* and *throughput* which are sometimes defined differently by different research. McGovern, Stevens, and Mathew in 2003 said "A transaction may be thought of as an interaction with the system, resulting in a change to the system state, while the interaction is in process of changing state, any number of events can interrupt the interaction, leaving the state change incomplete and system state in an inconsistent, undesirable form. Any change to the system within a transaction boundary, therefore has to ensure that the change leaves the system in a stable and consistent state. A transactional unit of work is one in which the following four fundamental transactional properties are satisfied: atomicity, consistency, isolation, and durability (ACID)." On the other hand, Younas and Iqbal state: "Transaction is defined as a unit of work wherein several operations can be treated as a logical work performed."

Definition 1: For this research a *transaction* is a process which executes a logical unit of work a system, intending a change of system state. The logical unit of work may consist of a number of individual operations. This research is considering composite Web services transactions. The transaction in this research therefore is a composite Web Service made up of component Web services. Each component Web service can be seen as an individual operation within the composite transaction and could be a transaction in the classical sense at the site of the service provider. Unlike McGovern, Stevens, and Mathew in 2003, which says the fundamental transactional properties of ACID have to be satisfied, this research uses an extended concept of a transaction in that a transaction in this research does not need to satisfy the ACID properties. This research considers a transaction which follows the rules of correctness agreed between the service consumer and service provider to be a transaction, even where this means relaxing classic ACID criteria.

In terms of throughput, Elnikety et al. in 2004 said “Throughput is the average number of successful requests that clients issue per unit time.” The definition for Elnikey et al.’s experimentation continues, “If a request fails or times out, it is not included in the measured throughput and response time, even though some components of the system may have executed parts of the request. Hence throughput is measured only for successful requests.” On the other hand, Alrifai et al. in 2009 said: “The overall throughput is measured by the number of terminated transactions per second.”

Definition 2: In the case of this research *throughput* is the number of successfully completed composite WS-transactions from a given batch of composite WS-transactions in which the time characteristics of long-running transactions are simulated. To count as a successfully completed composite transaction, at least one component transaction has to successfully complete. This is different from the definition by Elnikety et al., which says that all parts of the request have to be successful. Our definition of throughput is given below:

$$\underline{\text{Throughput}} = \frac{\text{Number of successfully completed composite transactions}}{\text{Total execution time (ms)}} \qquad \textbf{\textit{Equation 1}}$$

Definition 3: In the case of this research, *throughput unit time* is the average time it takes one transaction to complete. This is measured when a batch of transactions are run through the system. This measurement is directly related to throughput and is used as the defining measure in the experimentation. A lower throughput unit time indicates higher throughput.

$$\underline{\text{Throughput unit time}} = \frac{\text{Total execution time (ms)}}{\text{Number of successfully completed composite transactions}} \qquad \textbf{\textit{Equation 2}}$$

Definition 4: In the case of this research a *successfully completed composite transaction* is considered to be a composite transaction when at least one component has succeeded, provided that the correctness requirements set by the user allows for this. If the correctness criteria set by the user insists on all components of a composite transaction completing successfully then the composite transaction will only be considered to be successful if all components have completed successfully. However, if the correctness criteria have been set so that atomicity relaxation is allowed, then a composite transaction will be considered to be successfully completed if some of its component transactions have successfully completed.

Definition 5: In this research *successful completion of a component transaction* is considered to be when the consumer goal was satisfied. For instance, if the consumer wishes to book a restaurant on a particular day via a service and the booking is made, then that is considered to be a successfully completed transaction. If the booking is not made because of a lack of availability, then the component transaction would be considered to be unsuccessful. This understanding is in line with our common understanding of such consumer tasks. For instance, we might say to a friend, "Did you manage to buy that book?" and the friend might answer, "No – I'm afraid I was unsuccessful in that task." The research did not specifically consider other lower level types of failure such as logical error, system crash, or network failure. However, these types of failure would also result in the consumer goal not being satisfied. Thus at the AuTrA middleware level the transaction logic would still be applicable. AuTrA does not at present have the functionality to capture these sorts of lower level failures, as this aspect lies outside the scope of the research. Commercial middleware systems commonly have error-handling functionality, and so this component would be added if AuTrA were to be launched as a commercial product.

5.3 Simulation model

The AuTrA system was used as a test environment for a number of experiments. The experiments were designed to simulate long-running composite Web transactions occurring over a period of time. The simulation was achieved by composing an application from individual Web services which had previously been created and registered in AuTrA. To simulate such applications executed by various users over a lapsed period of time, a batch file approach with random system-generated delays was used. Thus sets of calls to the application with various parameters were put into batch files and the batch files were used as input to AuTrA. AuTrA then utilised programmed system delays to represent the long-running and event-based nature of such applications in real life. Within the batch files various parameters were set to show the relaxation requirements of both consumer and provider. During the batch run any requirements set by the user which do not meet the service provider's requirements will be changed so that they follow the service provider's requirements.

5.3.1 Simulation set-up

A personal computer was set up as a client-server machine to simulate Web usage. AuTrA was run on this platform, which was also running IIS, SQL Server and Visual Studio.net. Seven Web services were set up within AuTrA. The services were based around the idea of travel and event booking and were called: Flight service, Hotel service, Ski service, Entertainment service, Restaurant service, Invitation service and Venue service. The simulation involved the idea of people making holiday and event bookings over the Web. The services were registered within AuTrA. Figure 31 shows a screen shot of the AuTrA back-end directory showing these services listed. Database objects were set up on the server side to simulate the data banks held by service providers. These database objects are shown registered on the server in Figure 32. The seven Web services access these database objects during simulation in a similar way as such services would do in real deployment. The services and database objects were developed within ASP.net and written in C#. Appendix D shows the WSDL and a SOAP message for one of the services. For the purposes of experimentation, three applications (or composite transactions) were composed from these seven services. One application consisted of three Web services, another of four Web services and the other of seven Web services. Then for each application six sets of batches were set up for the experimentation. In each set there was a batch file for each service. The sizes of the six batch file sets for each composite transaction were varied as 20, 100, 200, 300, 400 and 500 transactions respectively. The experimental set up is summarised in Table 2.

Table 2 Simulation used in the evaluation

System Environment	Microsoft Internet Information Services Manager (IIS) SQL Server Visual Studio ASP.net (Version3.5.0.0)
Software Environment	AuTrA middleware (developed as part of this research)
Number of services	7 (developed as part of this research)
Names of services	*flight service*, *hotel service*, *ski service*, *restaurant service*, *entertainment service*, *invitation service*, *venue service*
Number of composite transaction types	3
Names of composite transaction types	Travel Plan application Travel and Party application Big Party Arrangements application
Number and names of services for the three composite transaction types	Travel Plan application - 3 services (*flight service, hotel service, ski service),* Travel and Party application – 4 services *(flight service, hotel service, restaurant service,* and *venue service)* Big Party Arrangements application -7 services (*flight service, hotel*

	service, ski service, restaurant service, venue service, entertainment service, and *invitation service)*
Number of batch file sets per composite transaction type	6
Number of composite transactions instances per batch file set	20,100,200,300,400,500
Number of batch files per batch file set	Various –one batch file per service (component transaction)
Delay factor	Random interval within a range
Transaction execution maximum interval	≤ 20 mins
Transaction execution minimum interval	0 mins
Attributes domain	Natural numbers, names, dates

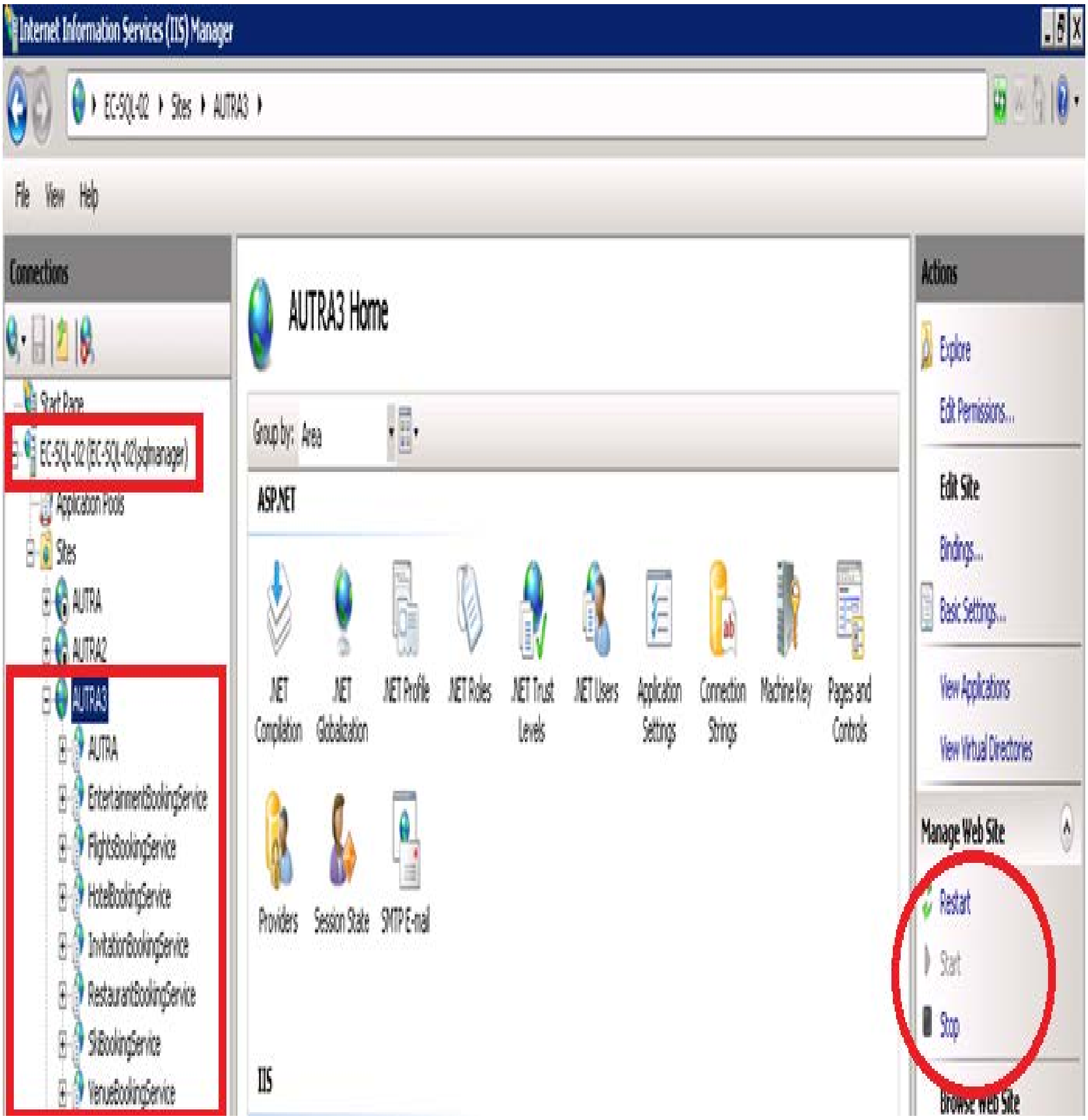
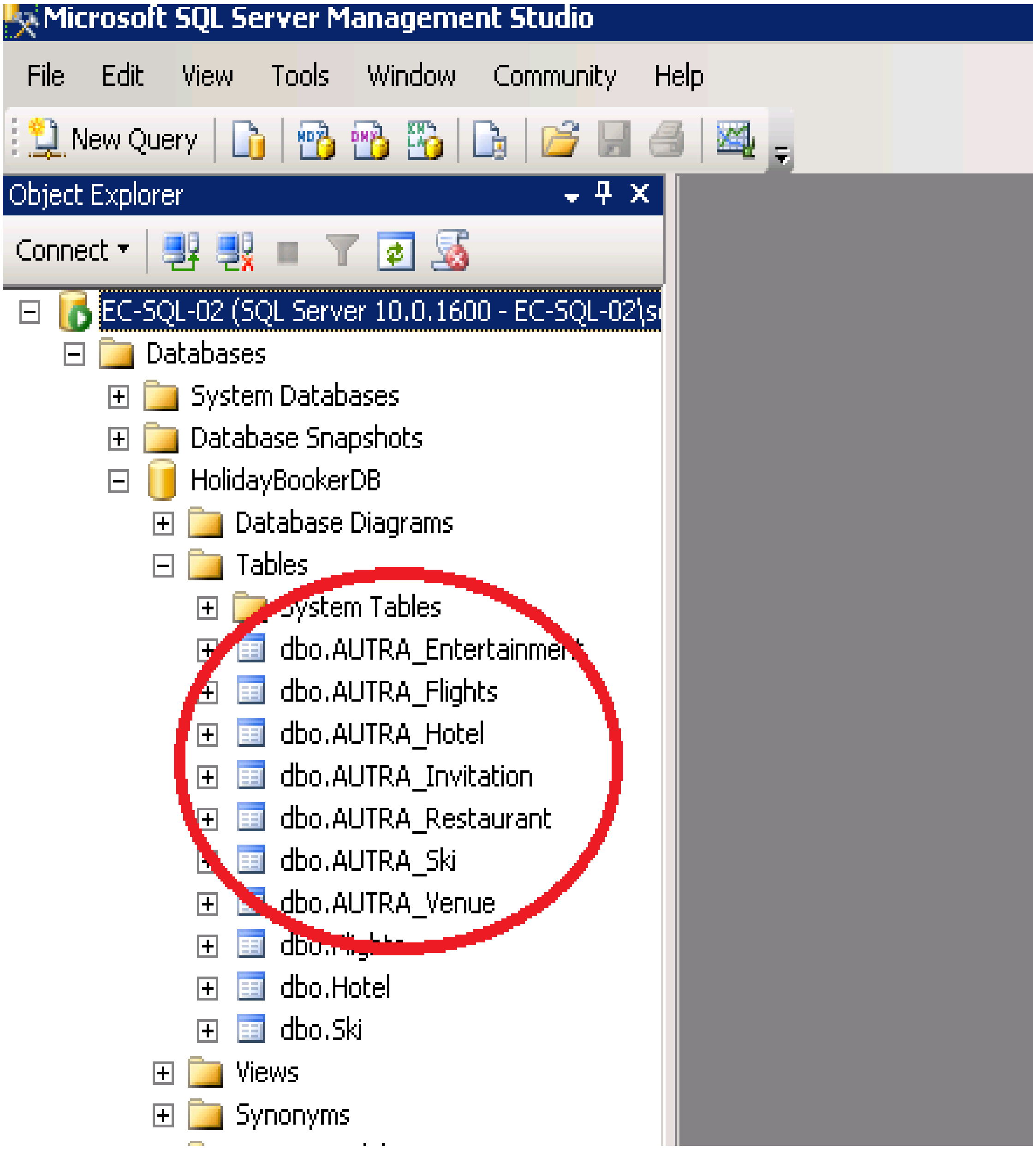


Figure 31 Web services registered with AuTrA



EC-SQL-02 (SQL Server 10.0.1600 - EC-SQL-02\s
Databases
System Databases
Database Snapshots
HolidayBookerDB
Database Diagrams
Tables
System Tables
dbo.AUTRA_Entertainment
dbo.AUTRA_Flights
dbo.AUTRA_Hotel
dbo.AUTRA_Invitation
dbo.AUTRA_Restaurant
dbo.AUTRA_Ski
dbo.AUTRA_Venue
dbo.Flights
dbo.Hotel
dbo.Ski
Views
Synonyms

Figure 32 Corresponding database of Web services registered with AuTrA

5.3.2 Simulation model settings and configuration

The simulation model had settings and configurations which were configured in a customised way. This was done in Web.config file and placed in the main directory of AuTrA. The following parameters were configured:

- *InputFilePath* – the file path (on the server) where the input files were uploaded.
- *OutputFilePath* – the file path (on the server) where the output files was created.

- *lRandomizerInterval* – the minimal time delay applied to each processed record.
- *hRandomizerInterval* – the maximal time delay applied to each processed record.
- *tentativeHold* – the amount of time the tentative hold was performed when applicable.
- *Buffer* – the maximum number of records processed at a time (the size of transaction set).
- *DataPath* – the path to the file which stored information about registered services.
- *<connection string>* tag **–** information about the connection string and name which is used by external Web services.

5.3.3 Mechanism for simulation relaxation of ACID and application-spefication properties

The mechanisms for simulation of relaxation and maintenance of the ACID and application-specific properties is given in Table 3

Table 3 Mechanisms for simulation relaxation in AuTrA

Property	Relaxation Method	Maintenance
Atomicity	Successful component transactions were allowed to commit even if other components belonging to the same composite transaction did not succeed	A composite transaction could only commit if all component transactions committed i.e. all-or-nothing. Tentative hold was used.
Consistency	Consistency rules were not followed and tentative hold was not used	Consistency rules were followed and tentative hold was used
Isolation	Component transactions from various composite transactions were interleaved without regard to serialisation. Locks and tentative hold were not used	Composite transactions were run in serial order
Durability	No saving was made of updates during main processing time but if “Tidy up” was used, saving was delayed to off-peak time	All updates were saved during main processing
Application-specific	Alternative criteria were used if user preference was not possible	No compromise of user preference was allowed

5.3.4 Simulation road map

The simulation process involves logging into AuTrA, composing an application, setting up an appropriate batch file with appropriate ACID and application-specific relaxation requirements, running the batch files, tailoring the requirements, negotiation if applicable and downloading results and then gathering statistics. Figure 33 shows a screen shot of the launching of the AuTrA system and Figure 34 provides an overview of the simulation process.

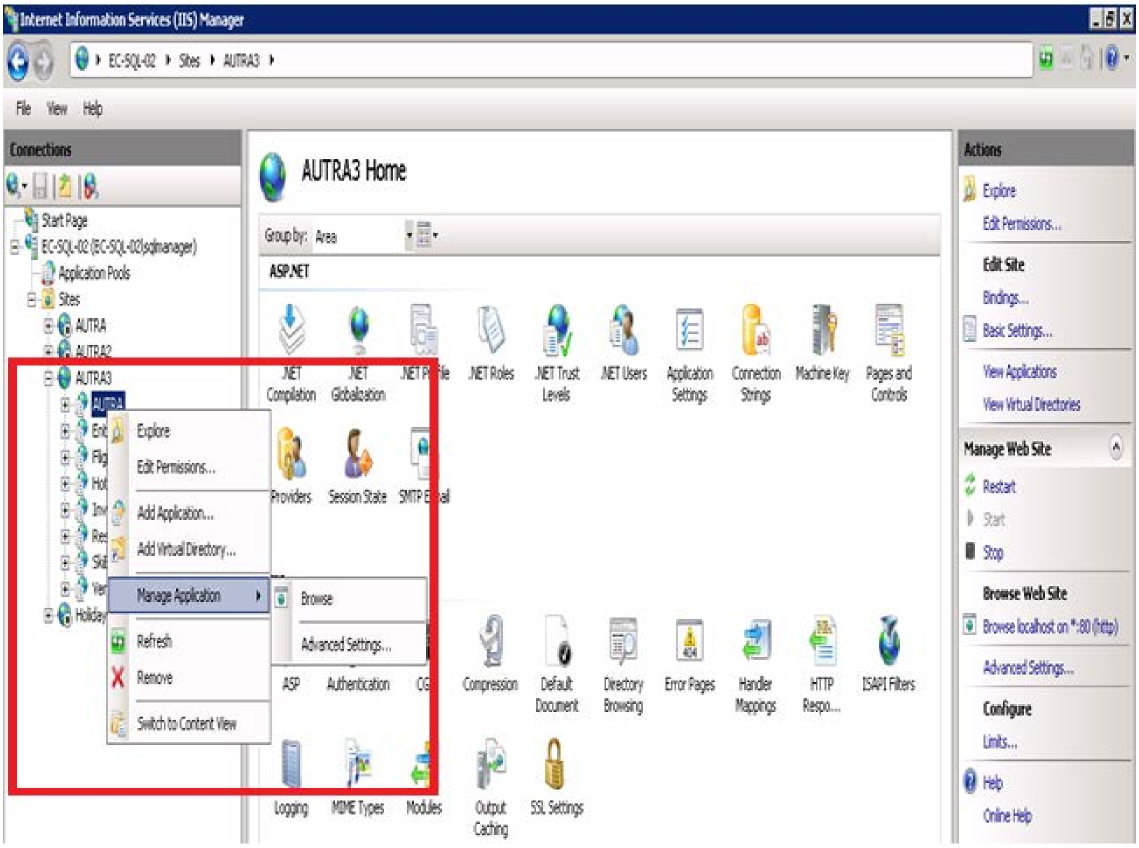


Figure 33 Launching the AuTrA simulation application

After opening the login page and logging in to AuTrA, the main page shown previously in Figure 24 will display. From here, applications can be composed and relaxation of properties can be specified. After this, the application can be run in real-time or batch mode. For the experimental simulation, a batch file approach was used to simulate multi-user interaction over a period of time. Random delays between starting transactions in the batch were configured to achieve this. New batch files can be set up offline for any new applications that are composed in AuTrA. Appendix E shows some example batch files used in the experimentation. AuTrA collects data during the run for analysis later. During the running, the consumer relaxation requirements are checked against those of the service provider. The code snippet in Figure 35 shows how requirements representing consumer requirements are checked against the provider requirements. The consumer requirements will be checked against the service provider requirements until the batch is finished. After all the transactions in the batch are processed, the total processing time will be returned with the list of transaction outcomes.

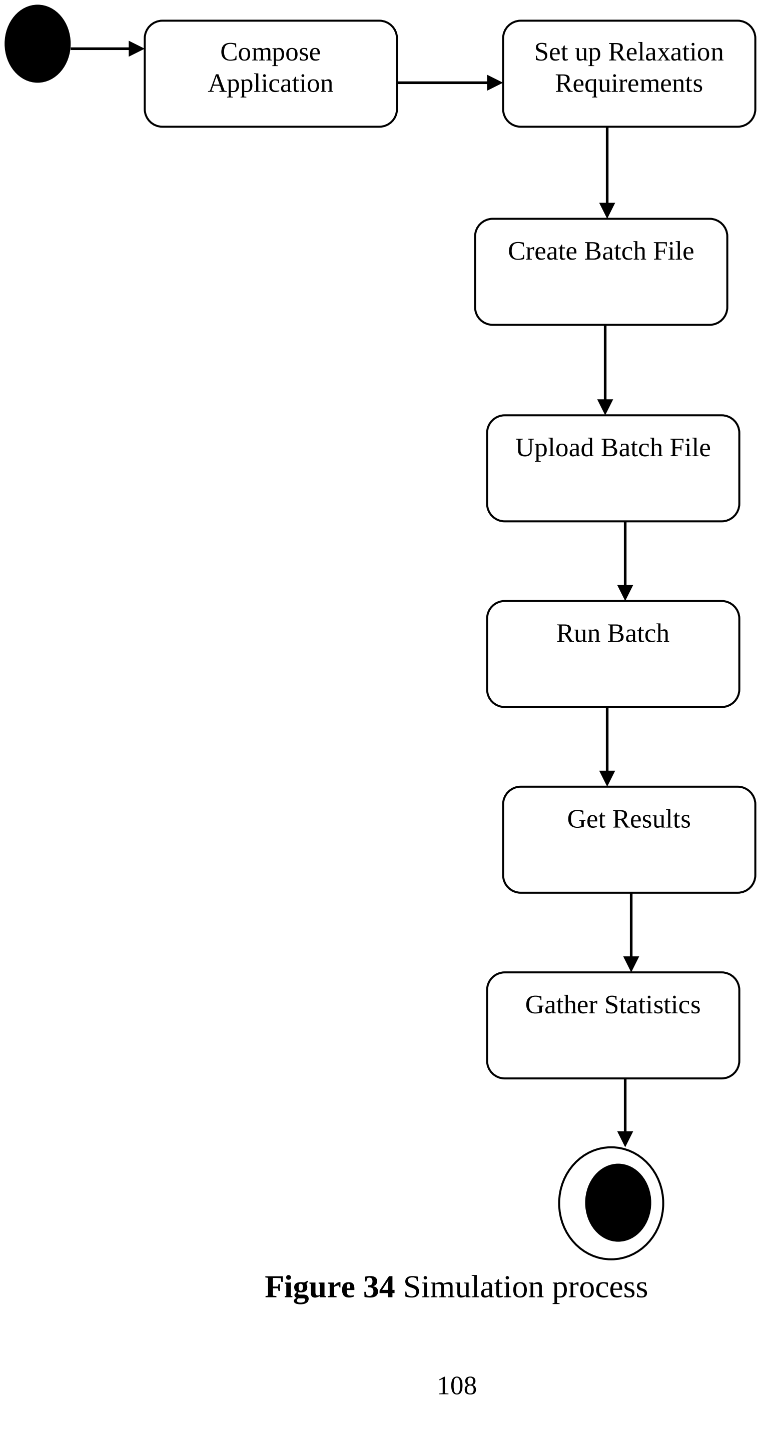


Figure 34 Simulation process

```
private void agreeInputFlags()
        {
            Boolean[] flags = new Boolean[] { true, true,
true, true  };

            foreach (var userData in tmpUser.userDataList)
            {
                if (!userData.rAtomicityFlag)
                {
                    flags[0] = false;
                }
                if (!userData.rIsolationFlag)
                {
                    flags[1] = false;
                }
                if (!userData.rConsistencyFlag)
                {
                    flags[2] = false;
                }
                if (!userData.rDurabilityFlag)
                {
                    flags[3] = false;
                }
   };
```

Figure 35 Snippet of code difference ACID relaxation

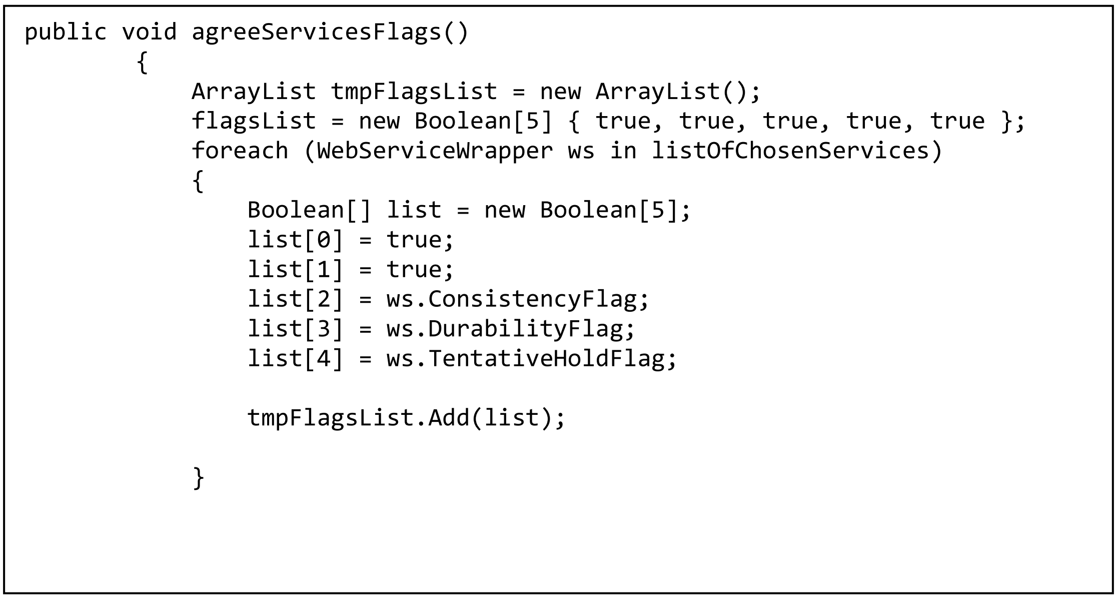
During the batch processing a random delay method was used which mimics the processing time of real-world transactions. As stated previously the minimum time set was 0 minutes and the maximum set for each transaction was <= 20 minutes. When tentative hold was checked during the simulation the tentative hold time was added to the maximum transaction processing time. As explained in Section 5.3.2, the hold time was specified in the Web.config file. When isolation was relaxed, transactions were processed concurrently and when isolation was not relaxed, transactions were processed one after the other. Figure 36 shows the snippet of code specifying the random delay.

```
private static void generateRandomDelay(int low, int high)
        {
            Random r = new Random();
            HiPerfTimer.delay = r.Next(low, high);

        }
```

Figure 36 Snippet of code for method of random delay

In cases where the simulation input file did not agree with ACID relaxation requirements of the service provider in terms of consistency and durability, the system asks the user to consider changing requirements; this is the Negotiation phase. If the user agrees to continue, their requirements for consistency and durability are changed to those of the provider. This code is shown in Figure 37.



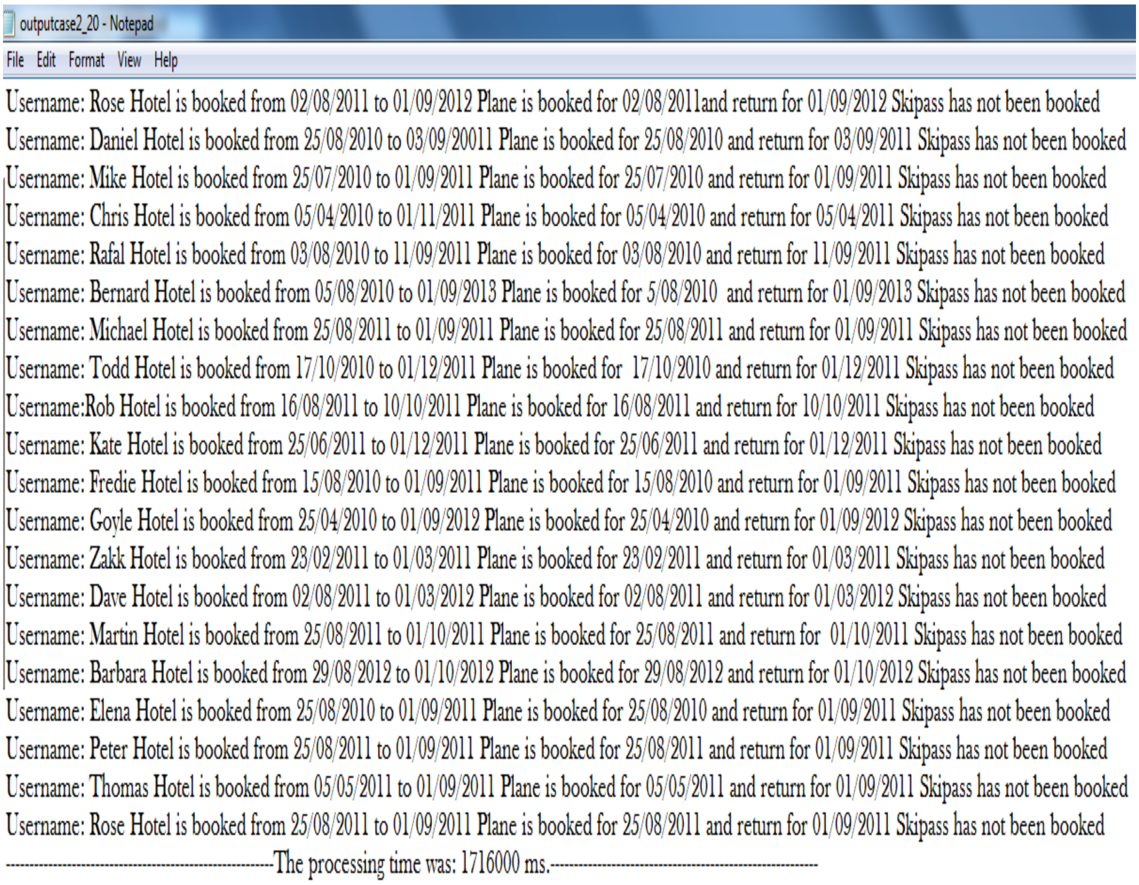
```
public void agreeServicesFlags()
        {
            ArrayList tmpFlagsList = new ArrayList();
            flagsList = new Boolean[5] { true, true, true, true, true };
            foreach (WebServiceWrapper ws in listOfChosenServices)
            {
                Boolean[] list = new Boolean[5];
                list[0] = true;
                list[1] = true;
                list[2] = ws.ConsistencyFlag;
                list[3] = ws.DurabilityFlag;
                list[4] = ws.TentativeHoldFlag;

                tmpFlagsList.Add(list);

            }
```

Figure 37 Snippet of code showing enforced agreement of provider requirements

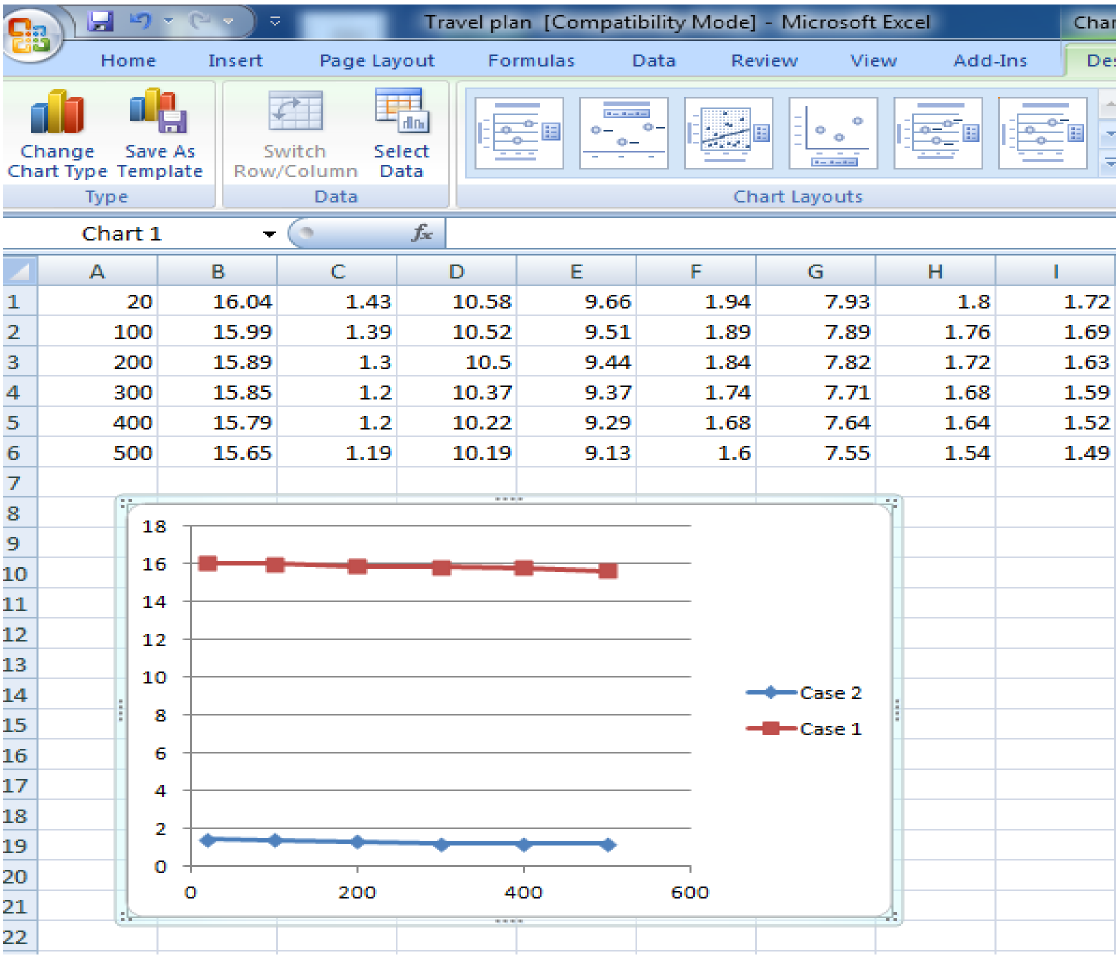
In the case where the simulation input file and the relaxation requirements of the service provider match, the model will process the transaction until the stop time (the maximum random delay time) is reached. The results will be committed in the database of the Web service if the durability was relaxed and tidying up at the end of the transaction is required. Then the output will be downloaded from the model in a text file. This file includes the processing time in milliseconds. Figure 38 shows an example of the output file contents.



```
Username: Rose Hotel is booked from 02/08/2011 to 01/09/2012 Plane is booked for 02/08/2011and return for 01/09/2012 Skipass has not been booked
Username: Daniel Hotel is booked from 25/08/2010 to 03/09/20011 Plane is booked for 25/08/2010 and return for 03/09/2011 Skipass has not been booked
Username: Mike Hotel is booked from 25/07/2010 to 01/09/2011 Plane is booked for 25/07/2010 and return for 01/09/2011 Skipass has not been booked
Username: Chris Hotel is booked from 05/04/2010 to 01/11/2011 Plane is booked for 05/04/2010 and return for 05/04/2011 Skipass has not been booked
Username: Rafal Hotel is booked from 03/08/2010 to 11/09/2011 Plane is booked for 03/08/2010 and return for 11/09/2011 Skipass has not been booked
Username: Bernard Hotel is booked from 05/08/2010 to 01/09/2013 Plane is booked for 5/08/2010  and return for 01/09/2013 Skipass has not been booked
Username: Michael Hotel is booked from 25/08/2011 to 01/09/2011 Plane is booked for 25/08/2011 and return for 01/09/2011 Skipass has not been booked
Username: Todd Hotel is booked from 17/10/2010 to 01/12/2011 Plane is booked for  17/10/2010 and return for 01/12/2011 Skipass has not been booked
Username:Rob Hotel is booked from 16/08/2011 to 10/10/2011 Plane is booked for 16/08/2011 and return for 10/10/2011 Skipass has not been booked
Username: Kate Hotel is booked from 25/06/2011 to 01/12/2011 Plane is booked for 25/06/2011 and return for 01/12/2011 Skipass has not been booked
Username: Fredie Hotel is booked from 15/08/2010 to 01/09/2011 Plane is booked for 15/08/2010 and return for 01/09/2011 Skipass has not been booked
Username: Goyle Hotel is booked from 25/04/2010 to 01/09/2012 Plane is booked for 25/04/2010 and return for 01/09/2012 Skipass has not been booked
Username: Zakk Hotel is booked from 23/02/2011 to 01/03/2011 Plane is booked for 23/02/2011 and return for 01/03/2011 Skipass has not been booked
Username: Dave Hotel is booked from 02/08/2011 to 01/03/2012 Plane is booked for 02/08/2011 and return for 01/03/2012 Skipass has not been booked
Username: Martin Hotel is booked from 25/08/2011 to 01/10/2011 Plane is booked for 25/08/2011 and return for  01/10/2011 Skipass has not been booked
Username: Barbara Hotel is booked from 29/08/2012 to 01/10/2012 Plane is booked for 29/08/2012 and return for 01/10/2012 Skipass has not been booked
Username: Elena Hotel is booked from 25/08/2010 to 01/09/2011 Plane is booked for 25/08/2010 and return for 01/09/2011 Skipass has not been booked
Username: Peter Hotel is booked from 25/08/2011 to 01/09/2011 Plane is booked for 25/08/2011 and return for 01/09/2011 Skipass has not been booked
Username: Thomas Hotel is booked from 05/05/2011 to 01/09/2011 Plane is booked for 05/05/2011 and return for 01/09/2011 Skipass has not been booked
Username: Rose Hotel is booked from 25/08/2011 to 01/09/2011 Plane is booked for 25/08/2011 and return for 01/09/2011 Skipass has not been booked
--------------------------------------------------The processing time was: 1716000 ms.--------------------------------------------------
```

Figure 38 Snippet of output showing bookings and processing time

The research took the processing time of the batch of transactions and used ***Equation (2)*** given in Section 5.2 to calculate the throughput unit time. The output of the different transaction sets (there were batches of 20, 100, 200, 300, 400 and 500 transactions for each application) were input into Excel for analysis and comparison. Each batch was run three times for each application and the average throughput unit times were calculated. Excel was used to produce graphs to illustrate the experimental results. Figure 39 provides a screenshot from this process. The results were analysed statistically to check significance. The results of the experiments are presented in Chapter 6 and the statistical significance testing is presented in Appendix C.



	A	B	C	D	E	F	G	H	I
1	20	16.04	1.43	10.58	9.66	1.94	7.93	1.8	1.72
2	100	15.99	1.39	10.52	9.51	1.89	7.89	1.76	1.69
3	200	15.89	1.3	10.5	9.44	1.84	7.82	1.72	1.63
4	300	15.85	1.2	10.37	9.37	1.74	7.71	1.68	1.59
5	400	15.79	1.2	10.22	9.29	1.68	7.64	1.64	1.52
6	500	15.65	1.19	10.19	9.13	1.6	7.55	1.54	1.49

Figure 39 Screen shot of the results analysis phase

5.3.5 Simulation road map summary

A batch file approach with configured delays was used to simulate the event-based nature and long-running nature of the types of transaction considered in this research. The user logs in and composes the required application from the services on offer. The user will be asked if any service criterion is flexible—if so, the user can choose the criteria that he/she allows to be compromised. Having developed the application, the user can upload batch files. AuTrA has experimental batch files set up for all registered services. The user could alternatively set up new experimental batch files. Then the relevant batch set can be run. The relaxation requirements are checked and if there are some requirements that are not allowed by the service provider the system goes to negotiation, after which the user either continues (which means changing requirements to agree with the service provider's requirements) or aborts the transaction. After the user has agreed with the service provider’s requirements the batch processing will continue to run until the whole batch is finished. Then the output will be downloaded and raw data collected.

5.4 Evaluation strategy

The strategy for evaluating whether the main aim of the research was achieved was experimental evaluation. The aim of the research was *to develop a system that increases throughput while maintaining the consistency and correctness required by particular applications.* Given the above aim it was conjectured that relaxing ACID properties and also supporting application-specific property relaxation would increase throughput. To test this conjecture, the experimental evaluation broke down into the following five main areas:

- Measuring the effect of relaxing atomicity, consistency and isolation properties in terms of throughput.
- Measuring the effect of relaxing durability and later tidying up in terms of throughput.
- Measuring the effect of application-specific property relaxation in terms of throughput.
- Measuring the effect of negotiation in terms of throughput.
- Measuring the effect of tentative hold in terms of throughput.

The above areas were tested to prove that AuTrA works, and also to shed more light on the effects of relaxation of ACID and application-specific criteria. The findings will enable better advice and support to be given to AuTrA users and also can be used by future researchers and developers working in this area. The following sections discuss each area of evaluation.

5.4.1 Evaluation of the effect of relaxation of atomicity, consistency and isolation

Experiments were performed to see if the relaxing of atomicity, consistency and isolation has any effect on transaction throughput. Furthermore it was intended to find the comparative and combined effects of relaxing these properties. As well as contributing to answering the research question, finding out such information would enable better advice and support to be provided for users of AuTrA. This area was covered by Experiment Set 1.

5.4.2 Evaluation of the effect of relaxing durability and later tidying up

This evaluation was carried out to find the effect on throughput of relaxing durability and then either tidying up or not tidying up at the end of the composite transaction. Tidying up refers to delayed saving. Updates may not be saved during peak processing time if durability is relaxed, but if tidying up is used, saving would occur later, theoretically at a less busy time. This area is covered in Experiment Set 2.

5.4.3 Evaluation of the effect of application-specific property relaxation

This evaluation was carried out to find out the effect on throughput of relaxing application-specific property requirements. These sorts of properties relate to individual component transactions. For instance, in booking a venue a consumer may prefer the location to be London but may be prepared to relax this requirement. Relaxing such a requirement may enable the composite transaction to succeed, whereas otherwise it would have failed. Such properties are not related to ACID properties but the research considered application-specific property relaxation to be a useful addition because at the application level, it has an effect on the success or otherwise of a composite transaction. This area was covered by Experiment Set 3.

5.4.4 Evaluation of the effect of negotiation

AuTrA includes a Negotiation phase. This allows consumers to rethink their requirements if these do not meet with the requirements of the service provider in terms of consistency and durability relaxation. It was conjectured that allowing negotiation and subsequent change in consumer requirements specification will achieve greater throughput because instead of aborting transactions when requirements did not match, such transactions could continue after successful negotiation and the changing of relaxation requirements. Therefore the research needed to find the effect of negotiation on transaction throughput. This area was covered by Experiment Set 4.

5.4.5 Evaluation of the effect of tentative hold

The research went further to explore the effect on throughput of tentative hold. Tentative hold is used to maintain atomicity. It can also be used to maintain consistency while relaxing isolation. Thus it was considered important to find out what effect its use had. This area was covered by Experiment Set 5.

5.5 Summary

In this chapter the key terms used in the research were defined. The simulation model of the research was introduced together with the simulation process and the evaluation strategy. In summary, AuTrA, the system developed in this research, was used as a platform for the simulation. AuTrA enables Web services to be registered, applications to be composed and batches of composite transaction instances to be executed and timed. The simulation consisted of a number of batch files representing different sets of component transaction instances. The batches were run varying ACID relaxation, application-specific relaxation, negotiation and tentative hold. This was done to evaluate the research aim. The results were recorded and analysed. These experimental results are discussed in the next chapter.

.

Chapter 6: Experimental Evaluation

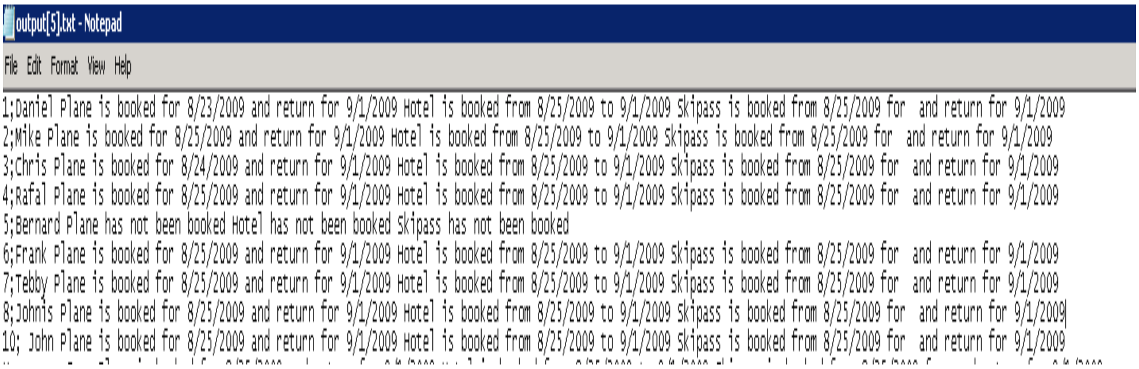
6.1 Introduction

This chapter presents the experimental results which were run to assess the effectiveness of the proposed system. The experiments fall into five groups which explore: the effects of relaxing the ACID properties and various combinations of these; the use of later tidy-up when relaxing durability; the relaxation of application-specific properties; the use of negotiation; and the use of tentative hold. This chapter discusses the scenarios and provides the results of the experiments.

6.2 Experimental set-up

The simulation platform is described in chapter 5. Three applications were set up using the Web Services registered. These applications were the Travel Plan Scenario, the Travel with Party Scenario and the Big Party Arrangements scenario. The applications were set up as composite Web transactions made up of component Web services.

In the simulation, batch files are processed by AuTrA using a configured delay mechanism to represent concurrent transactions starting at various times by different users over an extended period. There are six batch files of component transaction instances for each service in the composite transaction. These six batch files consist of 20, 100, 200, 300, 400 and 500 component transaction instances respectively. The component transaction instances across the batch files are linked via an identifier (the transaction number) to make up a composite transaction. The complete set of batch files for a composite transaction processing simulation is called a batch file set. The transaction instances in the batch file in the scenarios developed have the purpose of booking something e.g. a flight, hotel or restaurant. Underlying databases have been set up for each service representing the reservation status of the item being booked. The transaction instances in the batch files include some booking requests which will not be able to be fulfilled because of unavailability. If this happens the component transaction will be considered to be unsuccessful (see Figure 40). A random delay of between 0-20 minutes between transactions was used to represent the long-running nature of many Web transactions.



```
1;Daniel Plane is booked for 8/23/2009 and return for 9/1/2009 Hotel is booked from 8/25/2009 to 9/1/2009 Skipass is booked from 8/25/2009 for  and return for 9/1/2009
2;Mike Plane is booked for 8/25/2009 and return for 9/1/2009 Hotel is booked from 8/25/2009 to 9/1/2009 Skipass is booked from 8/25/2009 for  and return for 9/1/2009
3;Chris Plane is booked for 8/24/2009 and return for 9/1/2009 Hotel is booked from 8/25/2009 to 9/1/2009 Skipass is booked from 8/25/2009 for  and return for 9/1/2009
4;Rafal Plane is booked for 8/25/2009 and return for 9/1/2009 Hotel is booked from 8/25/2009 to 9/1/2009 Skipass is booked from 8/25/2009 for  and return for 9/1/2009
5;Bernard Plane has not been booked Hotel has not been booked Skipass has not been booked
6;Frank Plane is booked for 8/25/2009 and return for 9/1/2009 Hotel is booked from 8/25/2009 to 9/1/2009 Skipass is booked from 8/25/2009 for  and return for 9/1/2009
7;Tebby Plane is booked for 8/25/2009 and return for 9/1/2009 Hotel is booked from 8/25/2009 to 9/1/2009 Skipass is booked from 8/25/2009 for  and return for 9/1/2009
8;Johnis Plane is booked for 8/25/2009 and return for 9/1/2009 Hotel is booked from 8/25/2009 to 9/1/2009 Skipass is booked from 8/25/2009 for  and return for 9/1/2009
10; John Plane is booked for 8/25/2009 and return for 9/1/2009 Hotel is booked from 8/25/2009 to 9/1/2009 Skipass is booked from 8/25/2009 for  and return for 9/1/2009
```

Figure 40 Output with unsuccessful transaction examples

As stated in Chapter 5, the evaluation strategy included five different sets of experiments:

Set 1 – Experiments to measure effect of relaxing atomicity, consistency and isolation

Set 2 – Experiments to measure the effect of relaxing durability and later tidying-up when relaxing durability

Set 3 – Experiments to measure the effect of relaxation of application-specific properties

Set 4 – Experiments to measure the effect of negotiation

Set 5 – Experiments to measure the effect of tentative hold

In all experiment sets, AuTrA was used as a test platform and throughput unit time was used a comparative measure. According to the definitions given earlier, throughput is the number of successfully completed composite transactions over a given period of time. Throughput unit time is the average time taken to complete one composite transaction. Thus smaller throughput unit times indicate greater throughput. Three scenarios were set up to use in the experiments. Section 6.3 provides more detail about the scenarios.

6.3 Scenarios

Three different scenarios where assumed to carry out the experimental evaluation of AuTrA. All three scenarios were from the travel, tourism and leisure domain. Composite transaction types (or applications) were set up in AuTrA to represent the scenarios. The three composite transactions were the Travel Plan application, the Travel and Party application and the Big Party Arrangements application. They were composed of 3, 4 and 7 component transactions respectively. The reason for adopting three scenarios of different sizes was to demonstrate how AuTrA can be used to compose various sorts of composite transactions from an underlying set of component transactions. The scenarios were developed in this research for the purpose of experimentation and are similar to the types of scenarios used by other researchers in the same field. For instance, Younas et al. in 2006 used a travel scenario to demonstrate their model. The following sections describe the scenarios in more detail.

6.3.1 The Travel Plan application

The Travel Plan application is made up of three Web services related to travelling. Note that the scenarios were made up by the research. A description of the imaginary scenario follows:

SouthBots is an airline company that offers cheap cost tickets. The airline allows the users to use a Web-based environment to book the flights. SouthBots is an international company with offices around the continent. The headquarters are in London. Each office is connected to the head office through the internet. Booking of plane tickets can be done online or by going to travel agents.

Rainbow is a ski resort located in a popular area of Europe. Rainbow ski resort offers a magnificent panorama and, as it is built up around mountain villages, it offers a traditional atmosphere. Rainbow ski resort offers affordable prices for ski passes, lifts and equipment. Users can use the internet to book anything they need related to skiing.

Chalet Hotel is a family business and has been an attraction in the heart of Swiss resorts since the 18th century. The hotel has been home to thousands of well-known artists and other influential people. The hotel has many rooms. Booking of rooms is online or by telephone.

The three businesses have seen the advertisement of AuTrA system since they are all targeting the same market, i.e. people going for a ski holiday in Switzerland. The companies decided to join forces and use the AuTrA system in the belief that it will facilitate business. The companies have made some risk assessments of using AuTrA and optional relaxing of ACID properties; they make a contract. The services provider specification includes relaxing ACI (atomicity, isolation and consistency) properties but not D (durability).

SouthBots decided to relax consistency because the companies did not see any harm in not having consistent data at the end of the transaction. By relaxing consistency, in this case, booking is done even if the number of seats is negative. The airline has a backup of two planes always on standby with different capacities. That is, when a lot of seats have been booked beyond the capacity of the scheduled plane, a bigger plane on standby can be used, but if not a smaller standby plan can be used. The reason this plane can go when it is not scheduled is because the runway where the plane lands and takes off is owned by the airline and they have set aside time for this kind of situation for the plane to land when necessary.

The hotel, as advertised on the home page, has a lot of rooms available, but if there are more rooms booked than the ones available because of relaxation of consistency, the hotel will distribute residents to other hotels that have vacant rooms. This hotel is in partnership with other hotels, which have made a deal that excess bookings can be passed over to the group which has rooms available, and it has worked very well in aiding business.

The ski resort equipment provider also relaxed consistency because there is a local ski shop that rents equipment to ski resorts like Rainbow, so if it happens that there is more equipment booked than Rainbow has, Rainbow will borrow some equipment from another local shop. This helps because it means Rainbow does not have to turn down new customers when, although there is no ski equipment left in their shop, they can access supplies from elsewhere.

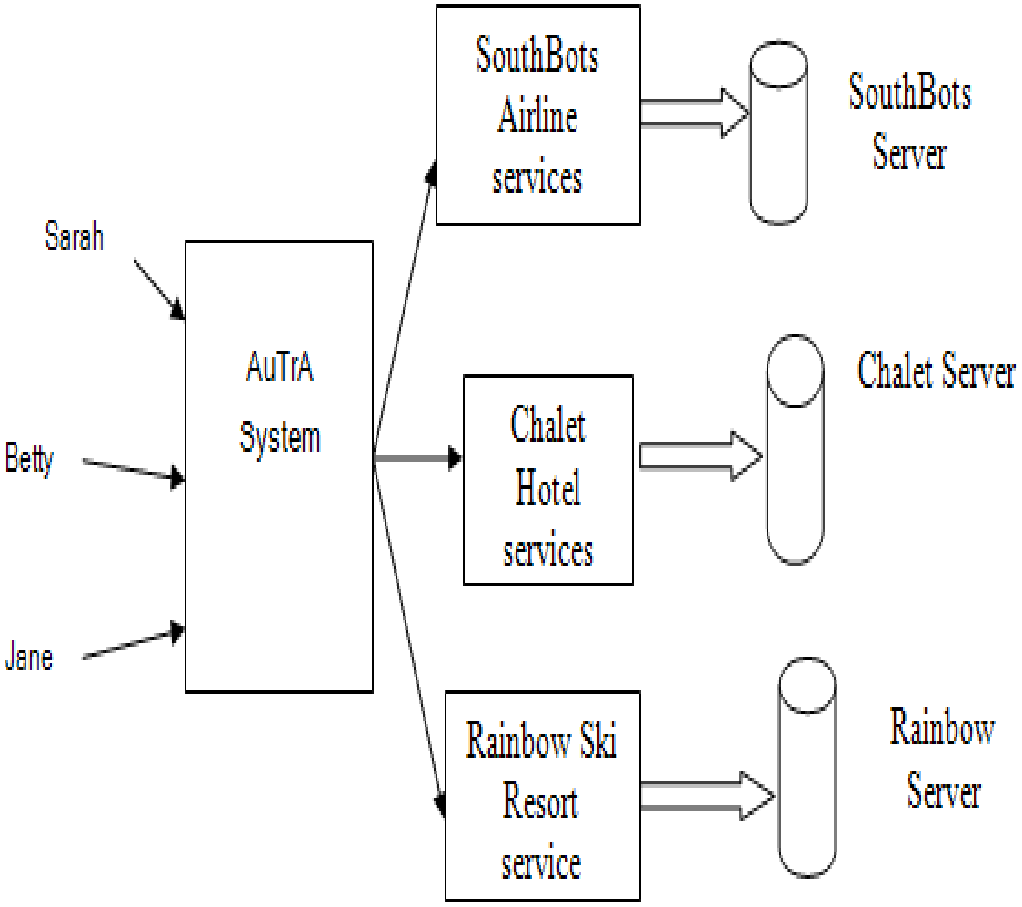


Figure 41 Scenario of composite Web services

Let us consider a scenario in which three users want to make travel arrangements to go for a skiing holiday to Switzerland. In reality, they will need a flight to the holiday location, a hotel and rental of ski equipment to go skiing. They all go to AuTrA Web services to make their holiday travel bookings. The assumption that the research makes is that the three users access AuTrA more or less at the same time. AuTrA has access to different services, which include component services for SouthBots Airline, Chalet Hotel and Rainbow Ski Resort. Services can be combined through AuTrA to form composite business transactions. The user requirements in terms of ACID property relaxation would be stated before the transaction is run.

AuTrA will process and execute the request of the output as shown in Figure 40, making sure that the business criteria or rules, i.e. the requirements set by the service provider, are met while also following the user's requirements, i.e. the requirements set by the service consumer. The consumer, on the one hand, and the data owner on the other control the relaxation of ACID properties.

Table 4 Examples of relaxation requests

	Relax Atomicity	Relax Isolation	Relax Consistency	Relax Durability
Sarah	Yes	Yes	No	No
Betty	No	No	No	No
Jane	Yes	No	No	No
T_4	Yes	Yes	Yes	Yes
. . .	. . .	. . .	. . .	. . .
T_{20}	Yes	Yes	Yes	Yes

Table 4 shows examples of possible relaxation requirements of the users of the travel plan application. Different users will have different relaxation selected depending on their needs. $T_{20,}$ for example, might be a transaction belonging to John, who relaxed all ACID properties. Other transactions might have relaxed different ACID properties. The table illustrates the possibility of different selections.

6.3.2 The Travel and Party application

The Travel and Party application consists of two of the previously described Travel Plan application services and two additional services related to party celebrations. The two services from the Travel Plan scenario are the flight service and the hotel service. The two additional party services are a restaurant service and a venue service, provided by Salut Restaurant and Asienhaus Garten respectively.

Salut Restaurant provides fine cuisine prepared with quality ingredients with a combination of flavours, imagination and, above all, a consistent standard of all these qualities throughout the meal. The restaurant is international with a lot of restaurants in Europe and it dominates the market in America. The use of internet technology for booking has contributed to the success of the restaurant.

Asienhaus Garten is a multi-room, multi-occasion venue, making it the perfect place for any event, day or night. Asienhaus Garten provides the perfect space and atmosphere for many different types of events. The experienced team are at the disposal of guests to provide the highest standards of coordination. The following areas are in Asienhaus Garten: the bar, which mixes a natural warm decor with contemporary design, giving a laid-back atmosphere for lunch, networking and drinks receptions; the restaurant, which can be perfectly matched to the client’s needs; the private dining area, which is suitable for personal services, with a stunning floor and glass ceiling; and Kakos Place, with glam decor, which is a perfect milieu for an evening of glamour. Bookings and reservations are processed online.

In summary, this application uses a *flight service, hotel service*, *restaurant service* and *venue service.*

6.3.3 The Big Party Arrangements application

The Big Party Arrangements application consists of the previously described Travel Plan application services and the Travel and Party application services combined with two other services related to party celebrations. The two additional party services are provided by Alfredo Entertainment and Liebe Creations.

Alfredo Entertainment established itself as a mobile entertainment service. For that reason, it has a competitive edge over its competitors that are not mobile. The company specialises in any kind of entertainment that is thinkable, for all events, be they children's parties, wedding events, birthdays, hen parties, stag nights and many more. The company has the knowledge, the experience and the personnel to satisfy their clientele's entertainment needs. The online services offer a selection of services, quotes, bookings and calendar.

Liebe Creations has been creating event cards for the past 20 years. The business supplies any invitation cards, menus and thank you cards, which are custom-made and handmade. The business supplies both the local and international market. To make the process easy everything is done online.

The research assumes that the seven providers team up to offer a composite service for a user making some party arrangements. The services that the user might need are shown in Figure 42. They are: *flight service, hotel service, ski service, restaurant service, entertainment service, invitation service* and *venue service.*

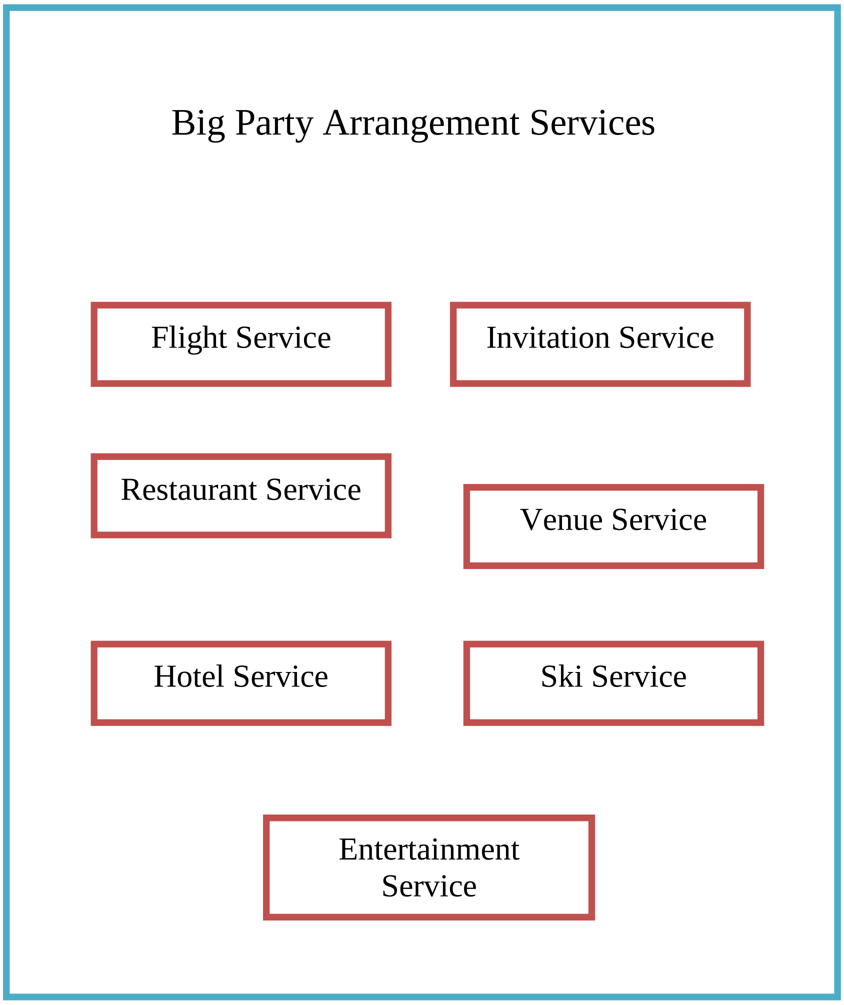


Figure 42 Big Party Arrangements composition

6.4 The Experiments

In this section, 15 experiments are presented grouped according to the evaluation strategy outlined in Chapter 5. A number of user relaxation cases were set up as shown in Table 5. There are a number of combinations of relaxation criteria that the consumer can request in the transaction processing, but only the ones shown in Table 5 are presented in the experiments as these are representative for the purposes of evaluation. Each experiment was carried out in the context of one of the scenarios outlined in section 6.3. The experiments measure the throughput unit time (see section 5.2, Definitions of the key terms in the research). A lower throughput unit time indicates increased throughput.

Table 5 Consumer relaxation combinations used in the experiments

Cases	Relaxed Atomicity	Relaxed Consistency	Relaxed Isolation	Relaxed Durability	Application-Specific Criteria	Scenario used
Case 1	No	No	No	No	Not used	Travel Plan
Case 2	Yes	Yes	Yes	No	Not used	Travel Plan
Case 3	Yes	No	No	No	Not used	Travel Plan Travel and Party
Case 4	No	Yes	No	No	Not used	Travel Plan Travel and Party
Case 5	No	No	Yes	No	Not used	Travel Plan Travel and Party

Case 6	Yes	Yes	No	No	Not used	Travel Plan Big Party
Case 7	Yes	No	Yes	No	Not used	Travel Plan Big Party
Case 8	No	Yes	Yes	No	Not used	Travel Plan Big Party
Case 9	No	No	No	No	Not used	Travel and Party
Case 10	No	No	No	Yes	Not used	Travel and Party Big Party
Case 11	Yes	No	No	Yes	Not used	Travel and Party
Case 12	No	No	No	No	No	Big Party
Case 13	Yes	Yes	Yes	Yes	Yes	Big Party
Case 14	Yes	Yes	Yes	Yes	No	Big Party

Case 15	Yes	Yes	Yes	No	Yes	Big Party
Case 16	Yes	Yes	No	Yes	No	Big Party
Case 17	No	Yes	Yes	Yes	No	Big Party
Case 18	Yes	No	Yes	Yes	No	Big Party
Case 19	Yes	Yes	Yes	No	No	Big Party
Case 20	No	No	No	No	Yes	Big Party

6.4.1 Set 1 - Experiments to measure effect of relaxing atomicity, consistency and isolation

In this section, a number of experimental results based on the Travel Plan application are reported. For these experiments the focus is on the services providers' relaxation specification shown in Table 6.The experiments are conducted to show the effect of relaxation of combinations of atomicity, consistency and isolation properties.

Table 6 Services provider relaxation requirements – Experiment Set 1

Relax Atomicity	Relax Consistency	Relax Isolation	Relax Durability
YES	YES	YES	NO

6.4.1.1 Experiment 1

In this experiment, the research concentrated on two cases: Case 1 (when no ACID properties are relaxed) and Case 2 (when atomicity, consistency and isolation are relaxed).

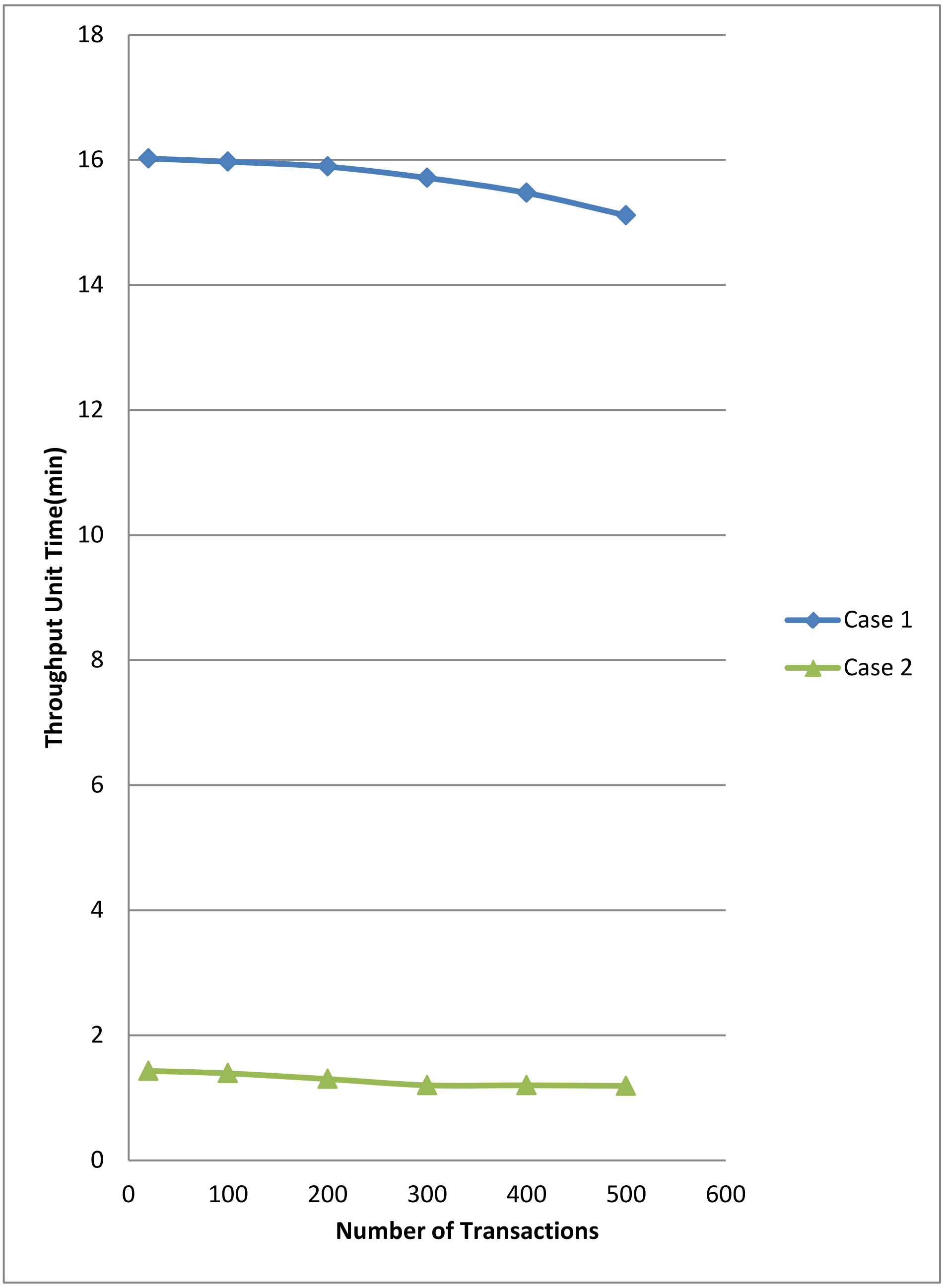


Figure 43 Case 1 and Case 2 ACI relaxation

From the experiment, it is clear that the relaxing of atomicity, consistency and isolation (ACI) gives a better throughput than when none of the ACID properties is relaxed. The statistical analysis shows a significant difference and is described in Appendix C.

6.4.1.2 Experiment 2

For this experiment, the research explored three cases: Case 3 (which relaxes only atomicity), Case 4 (which relaxes only consistency) and Case 5 (which relaxes only isolation). The main point of this experiment was to investigate the individual effect of relaxation of each of atomicity, consistency and isolation and compare these to see which property when relaxed gives the greatest throughput.

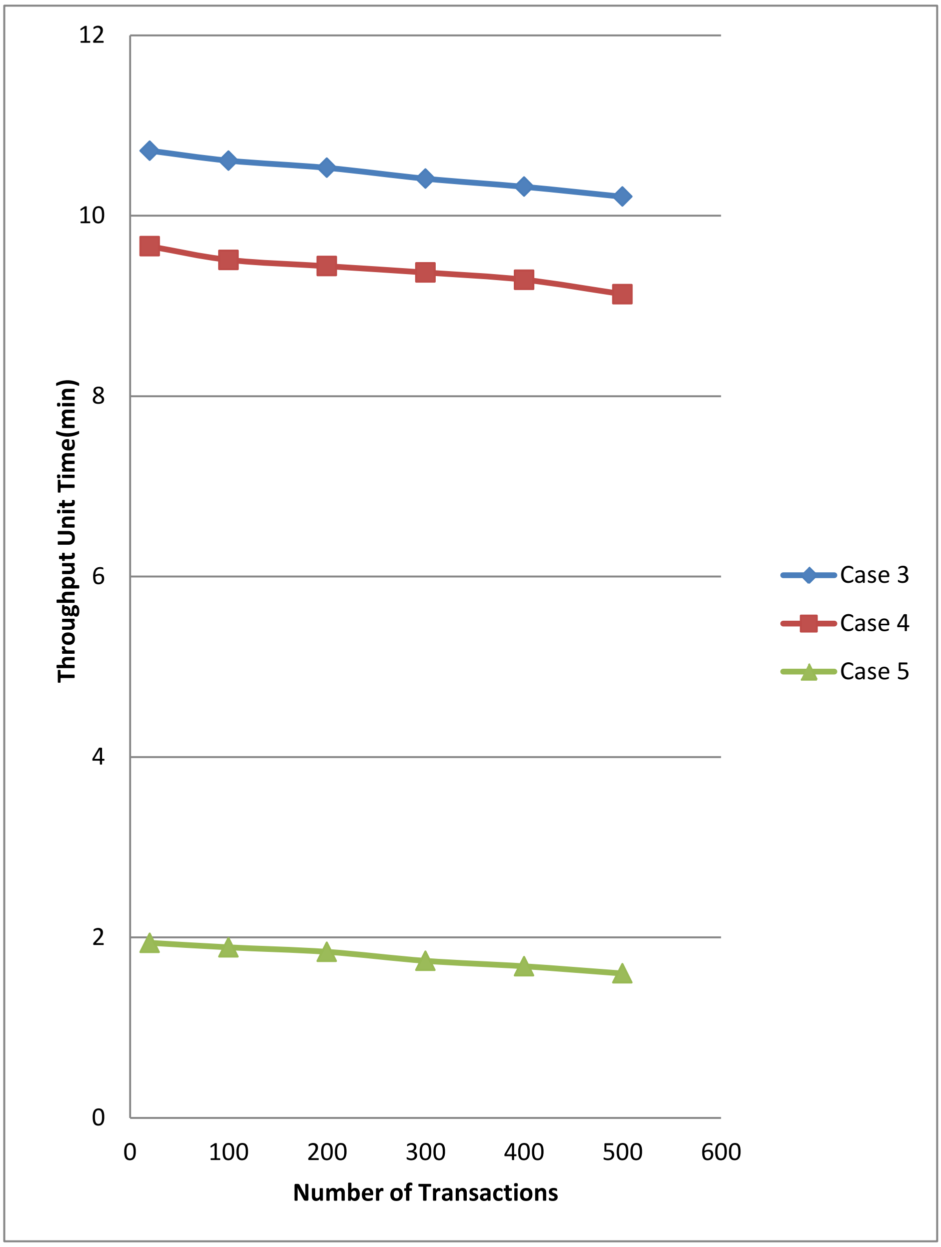


Figure 44 Cases 3, 4 and 5 ACI relaxation

The experiments show that atomicity relaxation produced the least throughput compared to consistency and isolation. Isolation has the greatest throughput. This is because relaxing isolation allows the transactions to be processed concurrently. The maintenance of strict isolation requires a serialisable schedule, which means that transactions often have to wait for each other to finish. The differences between the lines in the graph were found to be statistically significant as evidenced in Appendix C.

6.4.1.3 Experiment 3

Here the research looked at three cases: Case 6 (relaxes atomicity and consistency), Case 7 (relaxes atomicity and isolation) and Case 8 (relaxes consistency and isolation). The research investigated this to see the effect of each ACI property on the others. That is, the aim was to find out which combination of ACI property relaxation gives the best throughput and which combinations do not. Note that even if the throughput is not the greatest when compared with other combinations, there is improved throughput compared to not relaxing ACID properties at all.

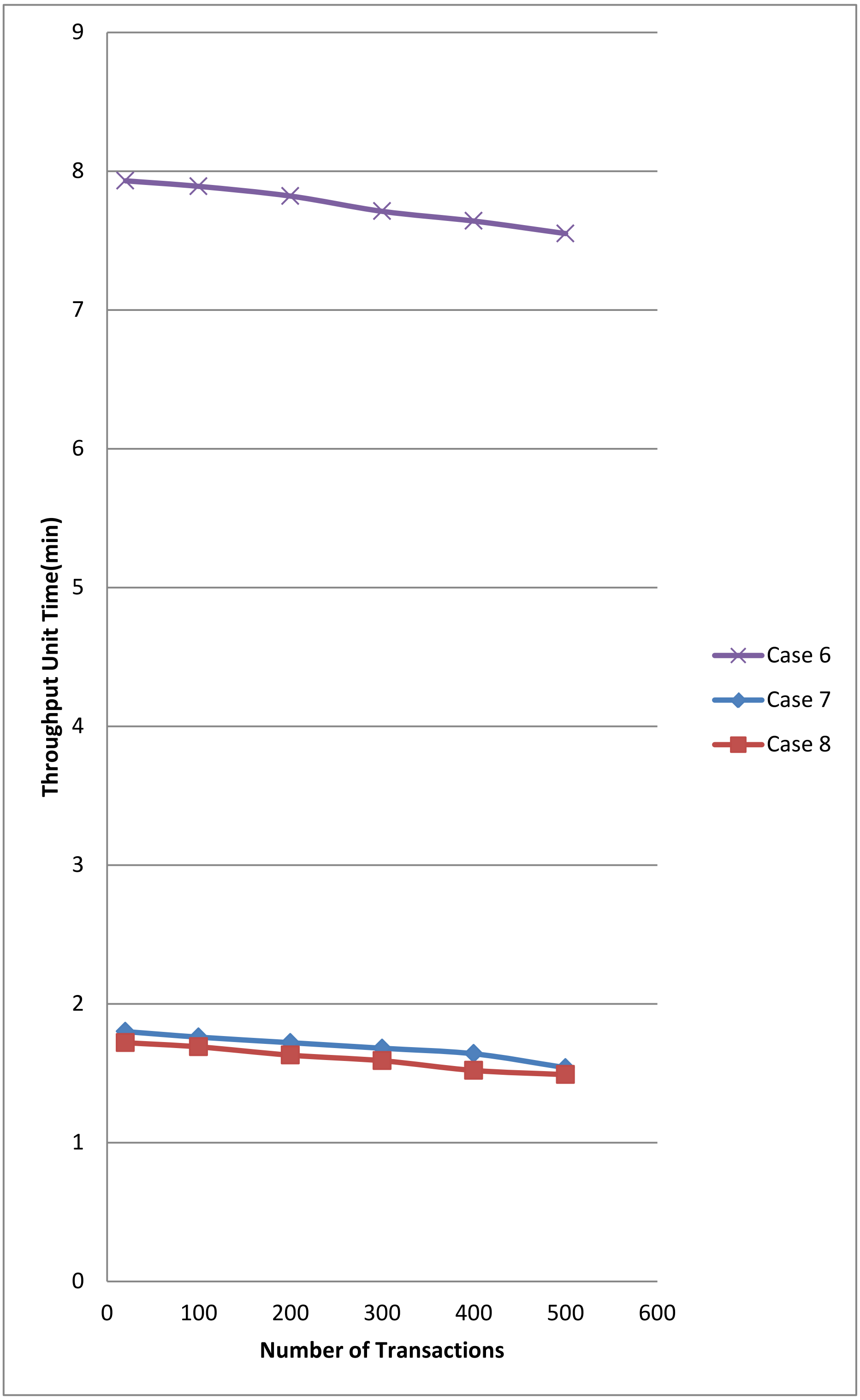


Figure 45 Cases 6, 7 and 8 ACI relaxation

The experiment shows that the combination of relaxing consistency and isolation has the best throughput. The throughput resulting from the relaxation of atomicity and isolation is close to that resulting from the relaxation of isolation and consistency. Looking at the individual relaxation of ACI properties it is clear that even if the relaxation of consistency produces better throughput compared to atomicity relaxation, it is not so much of a difference and including isolation in the combination narrows the gap more. Again the differences between all the lines in the graph were found to be statistically significant (see Appendix C).

6.4.1.4 Experiment 4

In this experiment, the research investigated Case 2 (when atomicity, consistency and isolation are relaxed), Case 6 (relaxation of atomicity and consistency), Case 7 (relaxation of atomicity and isolation) and Case 8 (relaxation of consistency and isolation). The main question behind this experiment was: Does the number of relaxed ACI properties have an influence on transaction throughput?

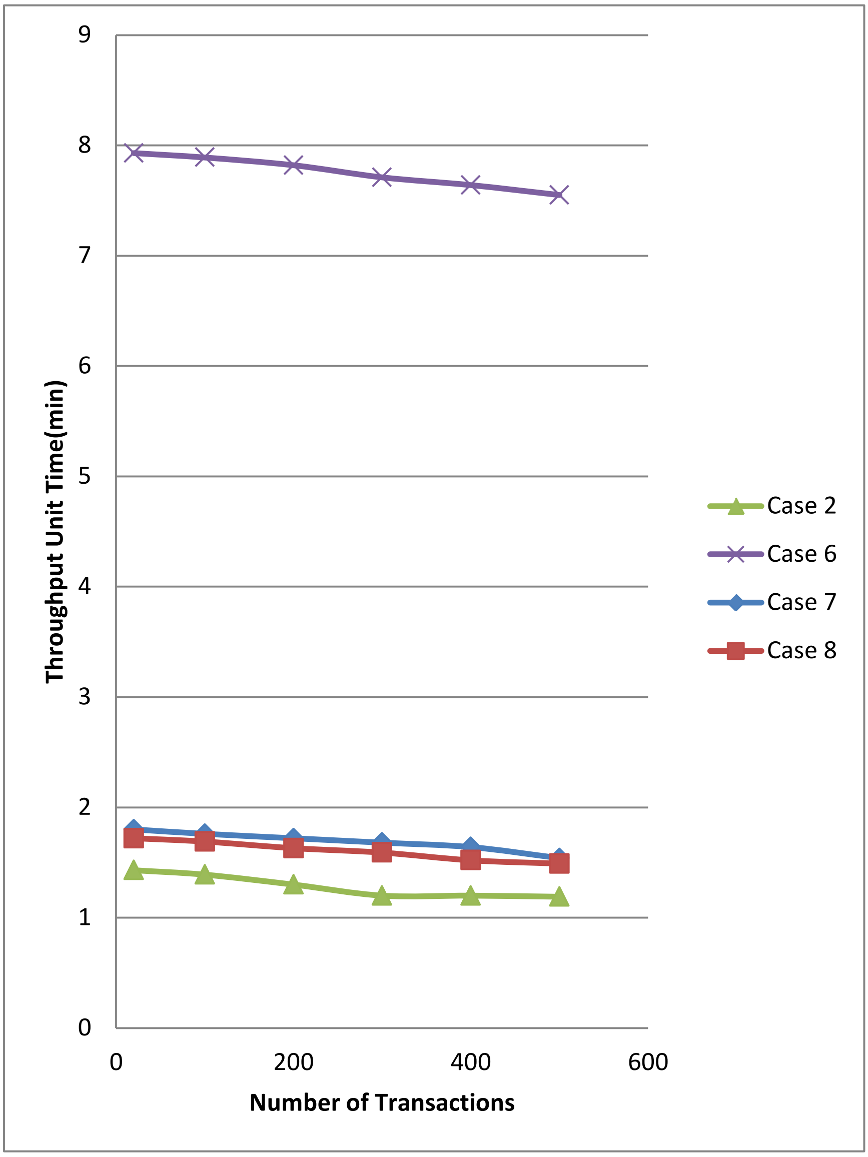


Figure 46 Case 2, Case 6, Case7 and Case 8 ACI relaxation

The experiment shows that when a number of ACID properties are relaxed, a better throughput is achieved compared to when fewer ACID properties are relaxed. Again the differences in the lines in the graph were found to be statistically significant (see Appendix C).

6.4.1.5 Summary of Set 1 experiments

These experiments demonstrated that relaxing all ACI properties improved throughput. It also showed that relaxing any of atomicity, consistency and isolation also improved throughput. The difference between relaxing any property and not relaxing it was statistically significant (see Appendix C).

6.4.2 Set 2 – Assessing the effect of durability relaxation with or without tidy-up

This set of experiments was based on the Travel and Party Arrangements application, with the exception of Experiment 9 which was based on the Big Party Arrangements application. This set of experiments focused on assessing the effect on throughput of relaxing the durability property. The set also investigated the effect of later tidy-up (i.e. delayed saving). The service provider requirements specified in Table 7 were assumed.

Table 7 Service provider relaxation requirements – Experiment Set 2

Relax Atomicity	Relax Consistency	Relax Isolation	Relax Durability
YES	YES	YES	YES

6.4.2.1 Experiment 5

The research measured throughput in Case 9 (when none of the ACID properties is relaxed) and Case 10 (where durability alone is relaxed). In this experiment tidying up of the data at the end of the transaction processing was not done. That is, the data was not saved at a later stage.

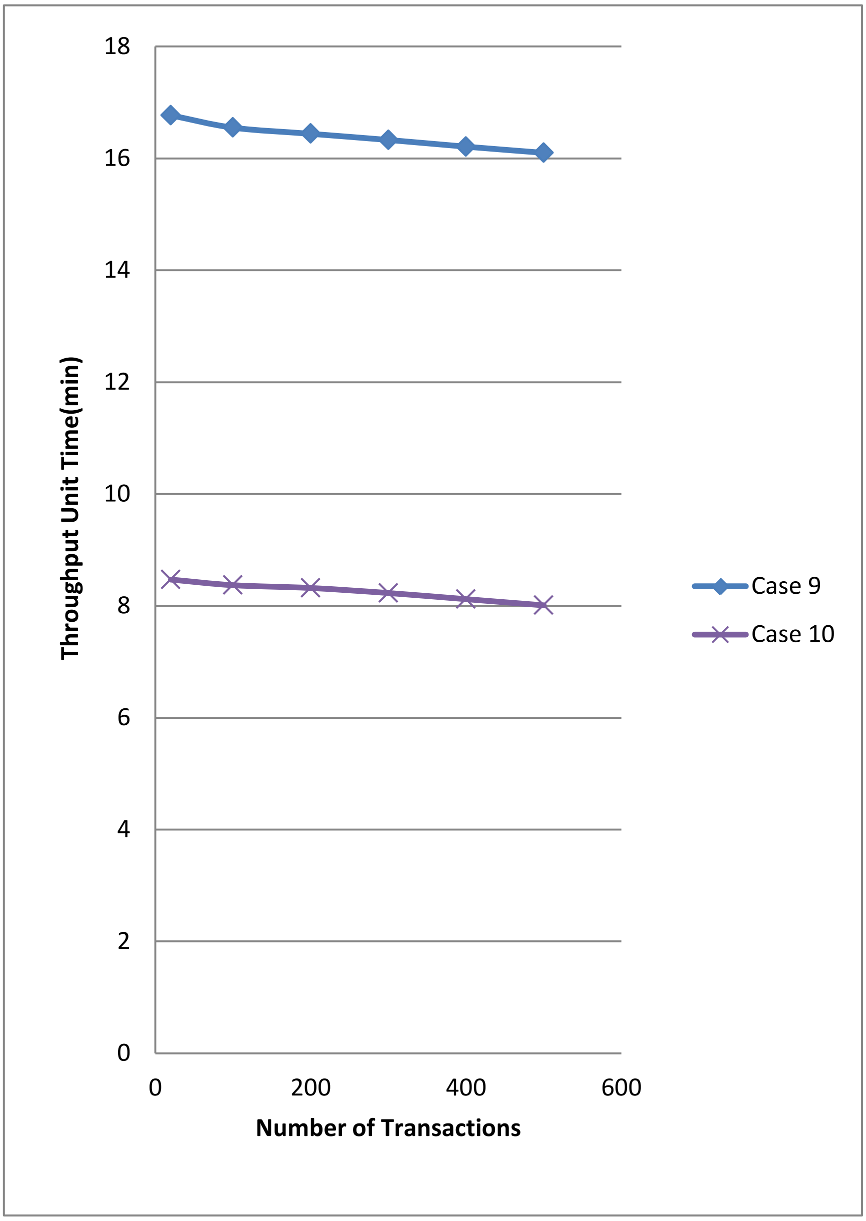


Figure 47 Case 9 and Case 10 ACID relaxation

The experiment shows that the relaxation of durability increases throughput, which can be a benefit for the processing during peak time. However data is not saved. Statistical analysis shows that the difference between the lines in the graph is significant (see Appendix C).

6.4.2.2 Experiment 6

The experiment focused on Case 11 (relaxing atomicity and durability), and compared it with Case 10 (where durability alone is relaxed). There was no saving at the end of the transaction processing period.

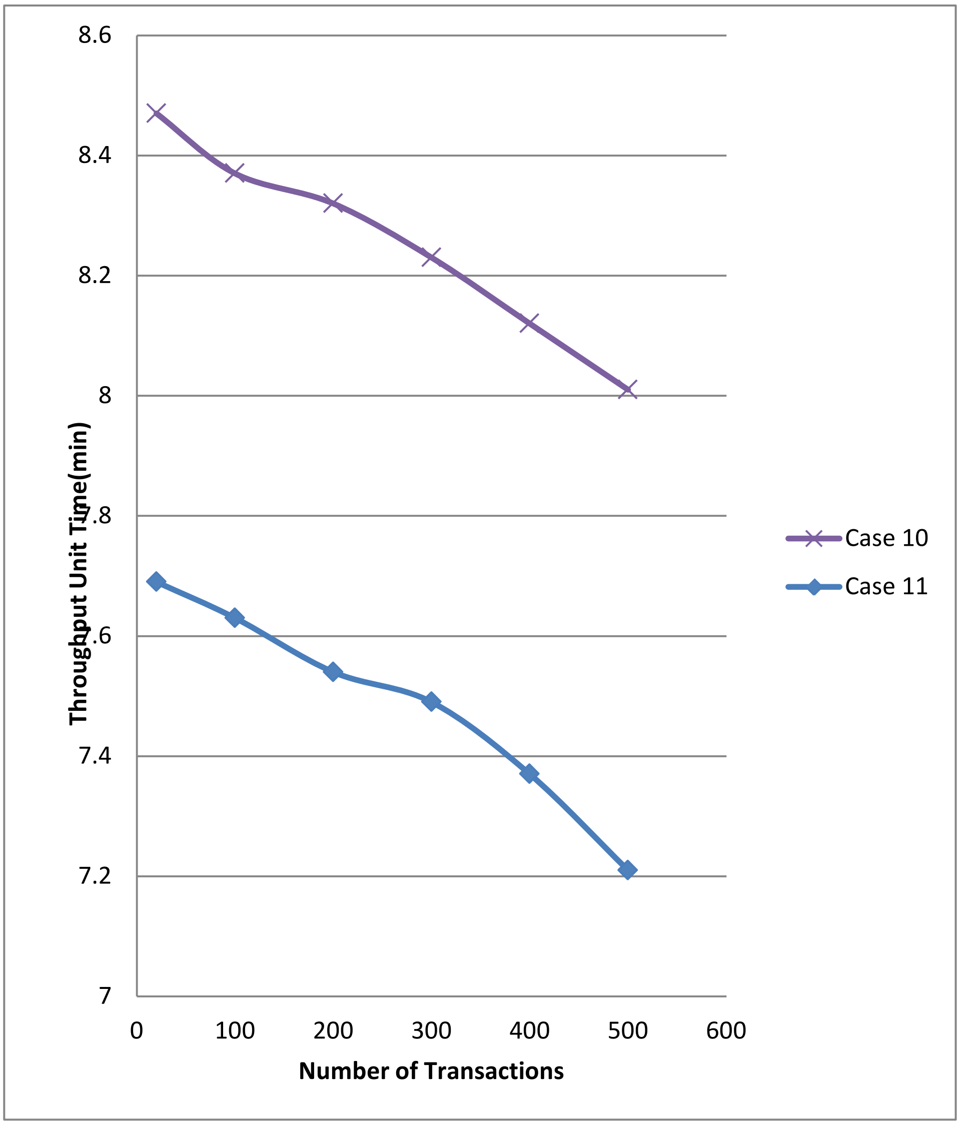


Figure 48 Case 10 and Case 11 ACID relaxation

The experiment shows that relaxing atomicity and durability achieves better throughput than relaxing durability alone. Statistical analysis shows that the difference between the lines in the graph is significant (see Appendix C).

6.4.2.3 Experiment 7

In this experiment, the research investigated the impact that tidying-up will have on the throughput of the transactions. The research repeated Experiment 6, but this time, although durability was relaxed during processing, there was a tidy-up at the end of the transaction processing period so that data was saved.

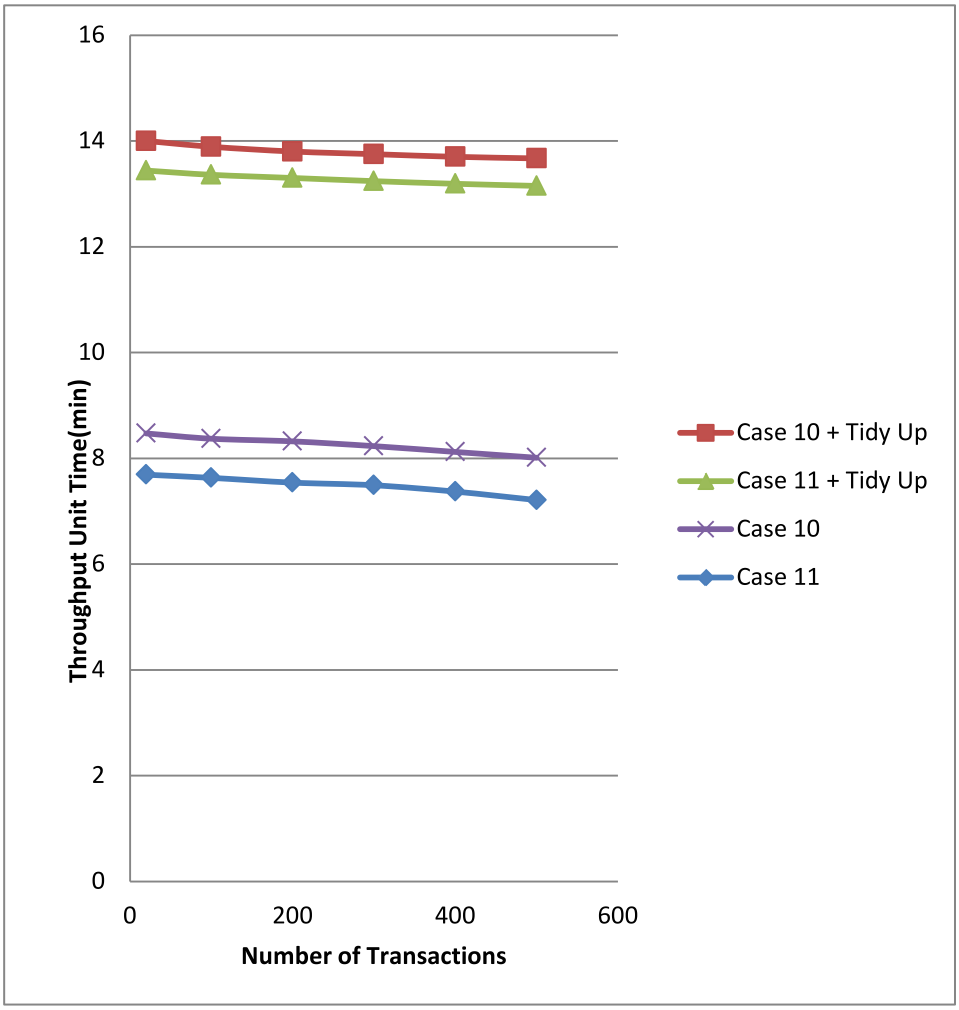


Figure 49 Case 10 and Case 11 relaxation with tidying-up

The tidying up of the data has an impact on the throughput compared to not tidying up. Thus throughput decreases when tidy-up is included. Statistical analysis shows that the differences between the lines in the graph are significant (see Appendix C).

6.4.2.4 Experiment 8

This experiment compared Case 3 (relaxing atomicity), Case 4 (relaxation of consistency), Case 5 (relaxation of isolation) and Case 10 (relaxation of durability). The experiment was performed to find out how relaxing durability compared to other relaxations.

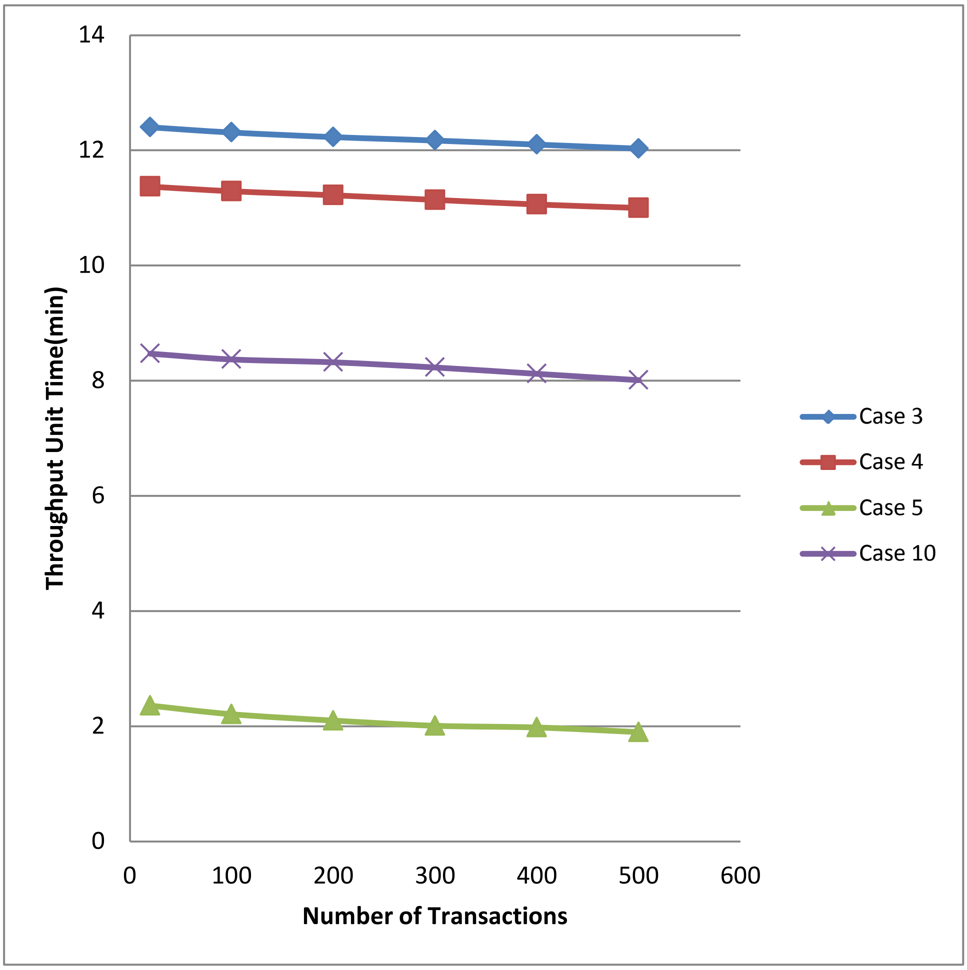


Figure 50 Case 3, Case 4, Case 5 and Case 10 ACID relaxation

The results show that durability has better throughput when relaxed compared to atomicity and consistency. However durability has lower throughput compared to isolation. There is a significance difference between the lines of the graphs (see Appendix C).

6.4.2.5 Experiment 9

For this experiment, the exploration was on Case 6 (relaxing atomicity and consistency), Case 7 (relaxing atomicity and isolation), Case 8 (relaxing consistency and isolation), Case 16 (relaxing atomicity, consistency and durability), Case 17 (relaxation of consistency, isolation and durability) and Case 18 (relaxation of atomicity, isolation and durability). The purpose of this experiment was to see the effects of relaxation of durability with other combinations of relaxation.

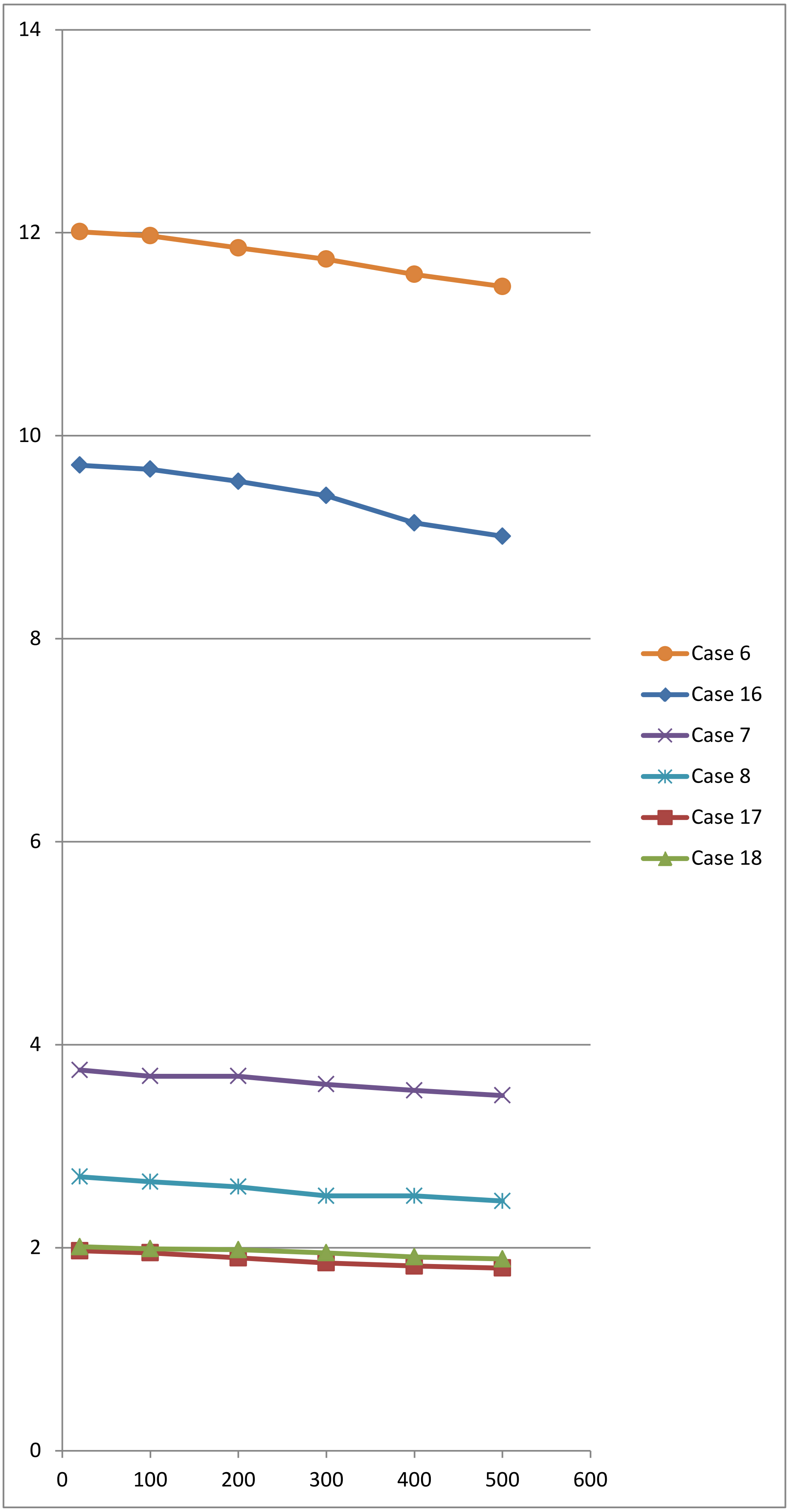


Figure 51 Case 6, Case 7, Case 8, Case 16, Case 17 and Case 18

The statistical analysis showed there is significant difference between the lines in the graphs. All cases were statistically significantly different from one another apart from Cases 17 and 18. The cases which directly compared relaxing durability against not relaxing durability were: Case 6 versus Case 16; Case 7 versus Case 18; and Case 8 versus Case 17. The cases in each of these pairs were all significantly different and this showed that relaxing durability increases throughput when used in combination with other relaxations. The two cases with the best throughput were Case 17 and Case 18. Both these included relaxation of isolation showing again the strong effect of isolation relaxation. The different factors in Case 17 and Case18 were the relaxation of consistency in Case 17 and the relaxation of atomicity in Case 18. The fact that Case 17 and Case 18 were not significantly different shows that atomicity and consistency relaxation have similar effects when combined with isolation and durability relaxation.

6.4.2.6 Summary of Set 2 experiments

These experiments demonstrated that relaxing durability improved throughput. When relaxing durability with atomicity, greater throughput was achieved. Tidying up at the end of the batch run, i.e. saving any data that was not saved during the batch run, decreases throughput, but in many situations this will be a necessary task. The difference between the graph lines resulting from this set of experiments was found to be statistically significant.

6.4.3 Set 3 – Assessing effect of relaxation of application-specific properties

Table 8 Service provider relaxation requirements – Experiment Set 3

Relax Atomicity	Relax Consistency	Relax Isolation	Relax Durability
YES	YES	YES	YES

This set of experiments was based on the Big Party Arrangements application. It focused on assessing the effect on throughput of relaxing application-specific properties. The service provider relaxation requirements shown in Table 8 were assumed.

6.4.3.1 Experiment 10

For this experiment the research evaluated the scenario where all seven services were selected and used Case 12 (relaxing none of the ACID properties and not relaxing any application-specific criteria) and Case 13 (relaxing all ACID properties and relaxing the application-specification criteria). The application-specific criteria are: *price, capacity of the restaurant, location of the restaurant, type of cuisine, location of the venue, price of the venue, location of the entertainment, price of the entertainment, price of invitation*, and *best possible date of flight, hotel* and *ski*. The customer might be prepared to compromise on these, in other words to relax them in terms of not insisting on a particular value for them, e.g. being prepared to fly on 5th August instead of 6th August. Such compromising should increase throughput.

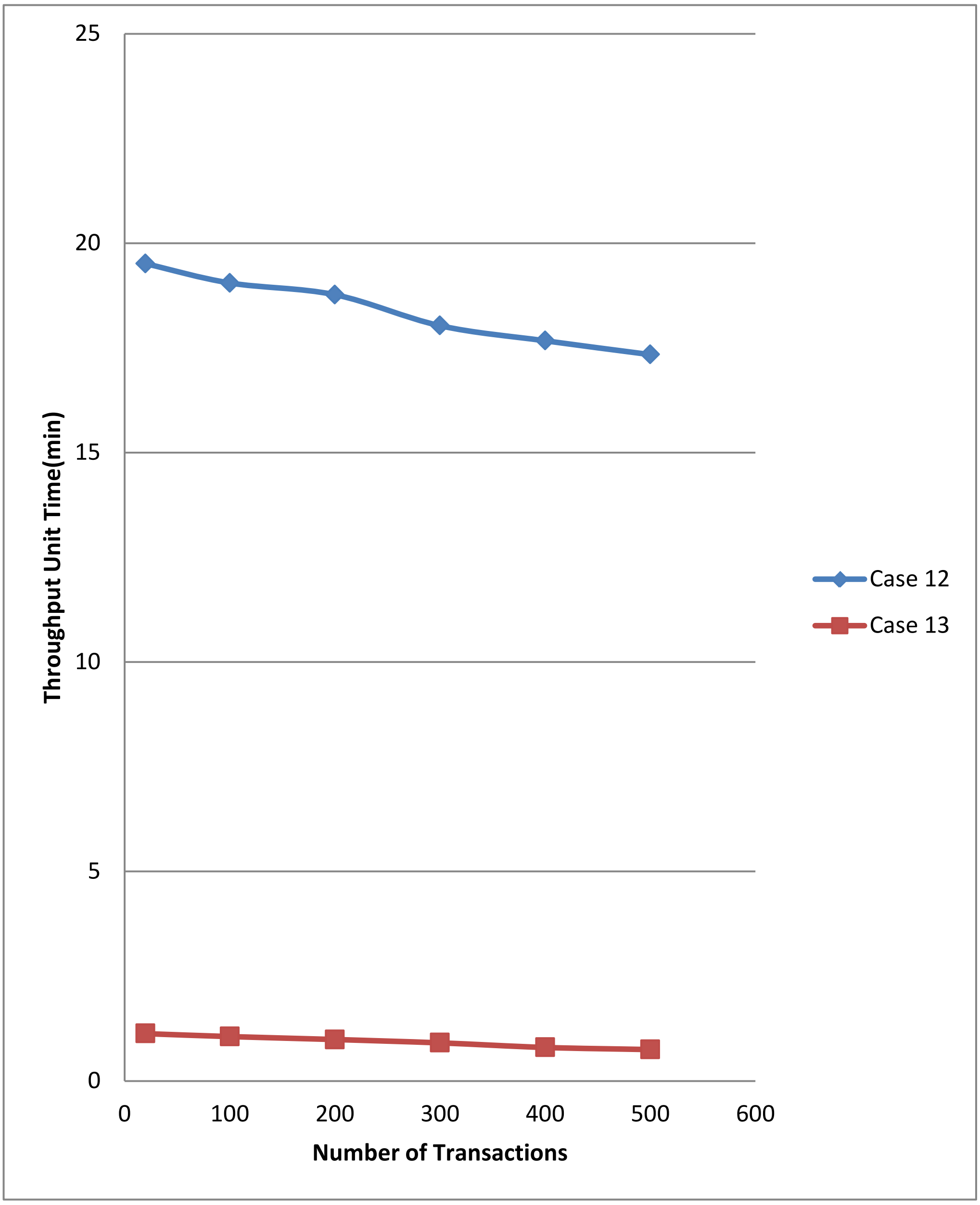


Figure 52 Case 12 and Case 13 ACID relaxation

The experiment shows that relaxing all ACID properties with application-specific criteria results in increased throughput. Statistical analysis shows that the differences between the lines in the graph are significant (see Appendix C).

6.4.3.2 Experiment 11

The experiment measured throughput in Case 13 (relaxing all ACID properties and the application-specification criteria), Case 14 (relaxation of all ACID properties but not application-specific criteria), Case 15 (relaxing atomicity, consistency, isolation and application-specific criteria) and Case 19 (relaxing atomicity, consistency and isolation and not application-specific criteria). These cases were chosen because the research wanted to investigate the effect of application-specific criteria relaxation on throughput.

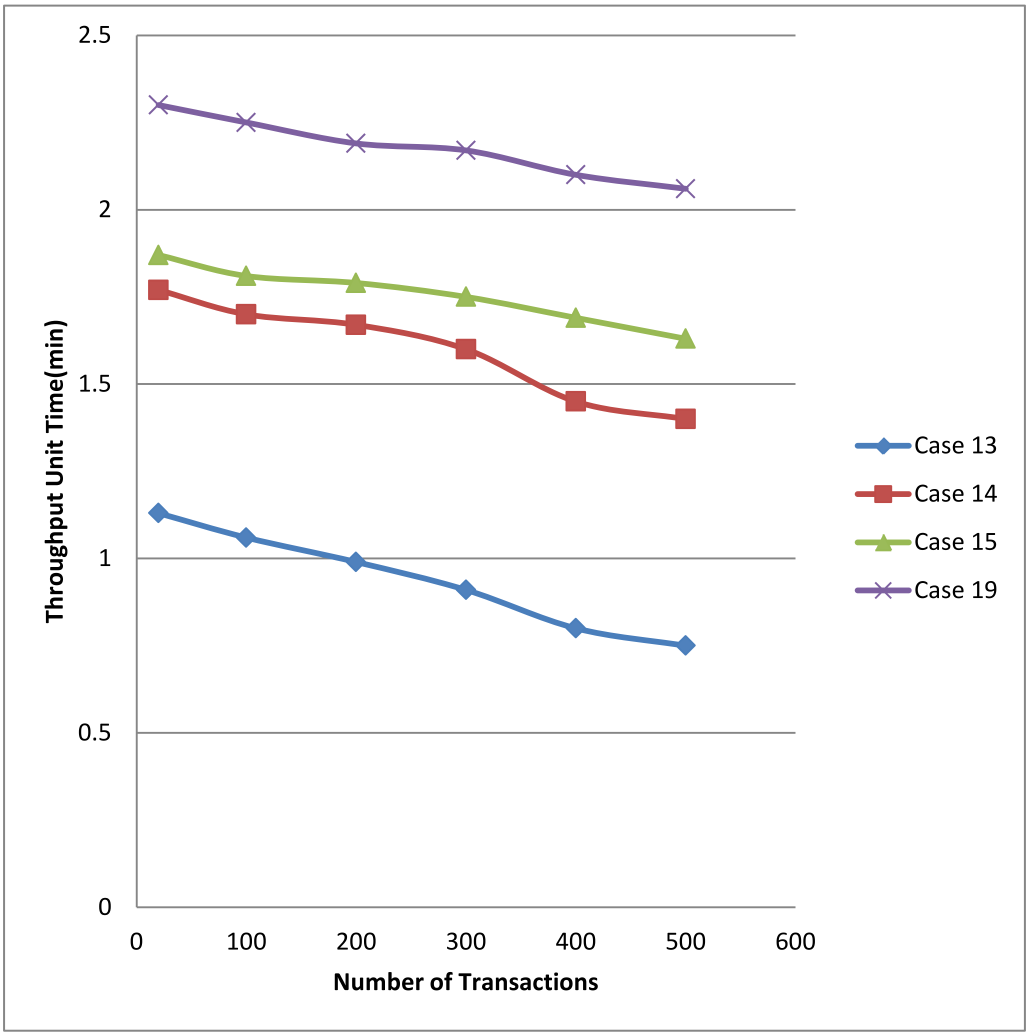


Figure 53 Case 13, Case 14, Case 15 and Case 19 ACID relaxation

Looking at experimental results it is clear that including application-specific criteria increases the throughput of transactions; the results of Case 13 are better that the results of Case 14 and the results of Case 15 are better those of Case 19. The experiment also shows that relaxing all ACID properties with application-specific criteria improves throughput compared with just relaxing ACI with application-specific criteria. Thus relaxing durability is shown again to be effective in improving throughput. Statistical analysis shows that these differences are statistically significant (see Appendix C).

6.4.3.3 Experiment 12

In this experiment, the research further measured the impact of relaxation of application-specific criteria by comparing Case 20 (relaxation of application-specific criteria but no relaxation of ACID criteria) with Case 12 (no relaxation of application-specific criteria or ACID criteria). Application-specific criteria for this experiment are *price, capacity of the restaurant, location of the restaurant, type of cuisine, location of the venue, price of the venue, location of the entertainment, price of the entertainment, price of invitation, and date of flight, hotel and ski*. In Case 20 the user is prepared to compromise on any of these but not on ACID property maintenance.

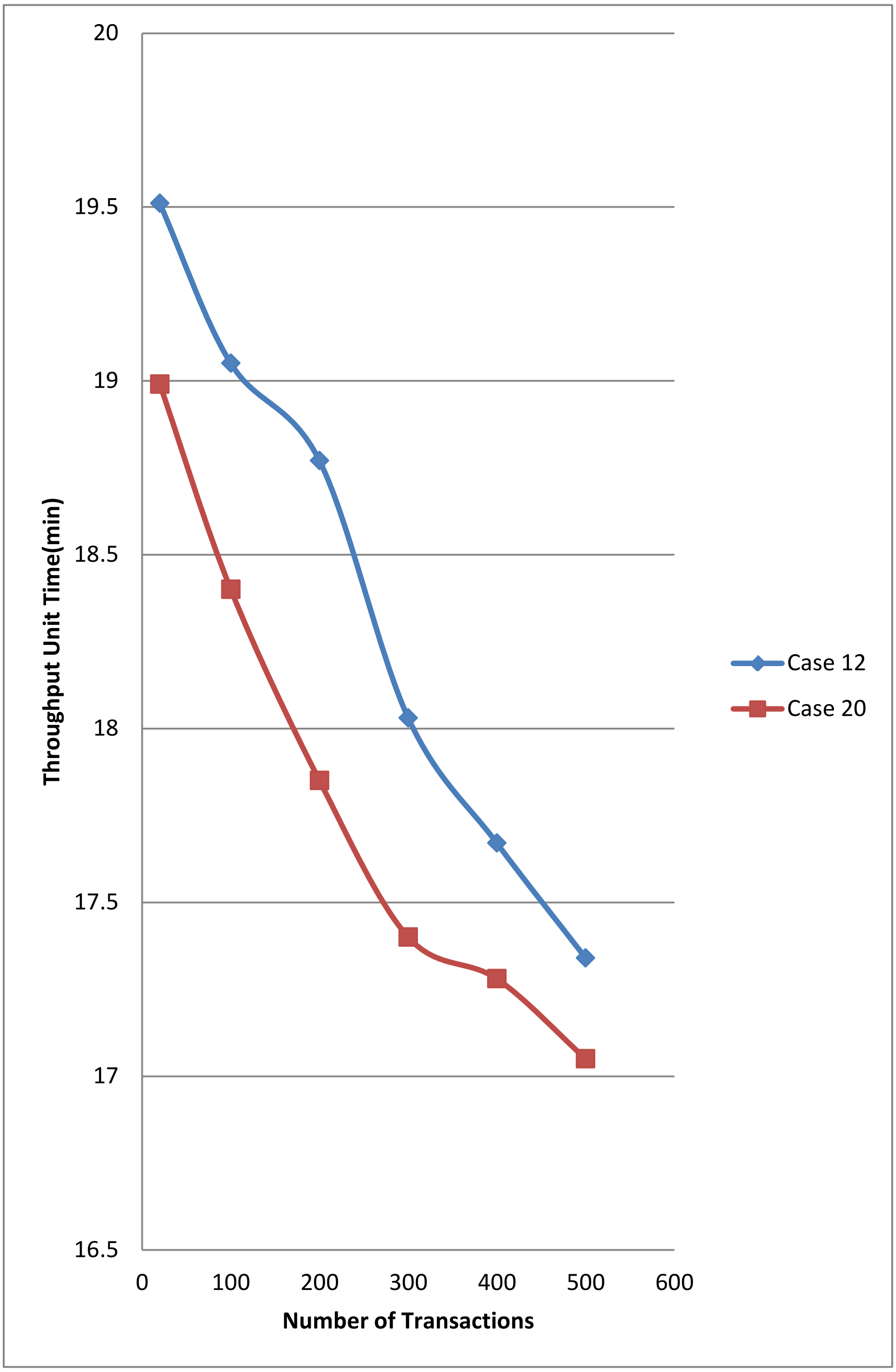


Figure 54 Case 12 and Case 20 ACID relaxation

The research shows that relaxation of application-specific criteria increases the throughput of the transaction. The results of Case 20 were better than for Case 12 in terms of throughput of transactions. Statistical analysis shows that these differences are statistically significant (see Appendix C).

6.4.3.4 Summary of Set 3 experiments

The experiments in Set 3 provide results which show that application-specific criteria relaxation is effective in improving throughput. The results were shown to be statistically significant.

6.4.4 Set 4 – Effect of negotiation on throughput

Table 9 Service provider relaxation requirements – Experiment Set 4

Relax Atomicity	Relax Consistency	Relax Isolation	Relax Durability
YES	NO	YES	NO

This set consists of just one experiment which was based on the Big Party Arrangements application and investigated the effect of negotiation. The service provider relaxation specification assumed is shown in Table 9.

6.4.4.1 Experiment 13

In this experiment the effect of negotiation on throughput is assessed. Negotiation allows differences between consumer and provider relaxation specifications to be resolved during processing, thus allowing the possibility of continuing the transaction rather than aborting. Assume that the service providers of the Party Arrangements Application found that it is not good for the business to have inconsistency of data at the end of the transaction, and for that reason the service providers change the relaxation of ACID properties set previously to the one shown in Table 8. The consumer's requirements Case 13 (relaxing all ACID properties and the application-specification criteria) and Case 14 (relaxing all ACID properties but not the application-specification criteria) was surveyed.

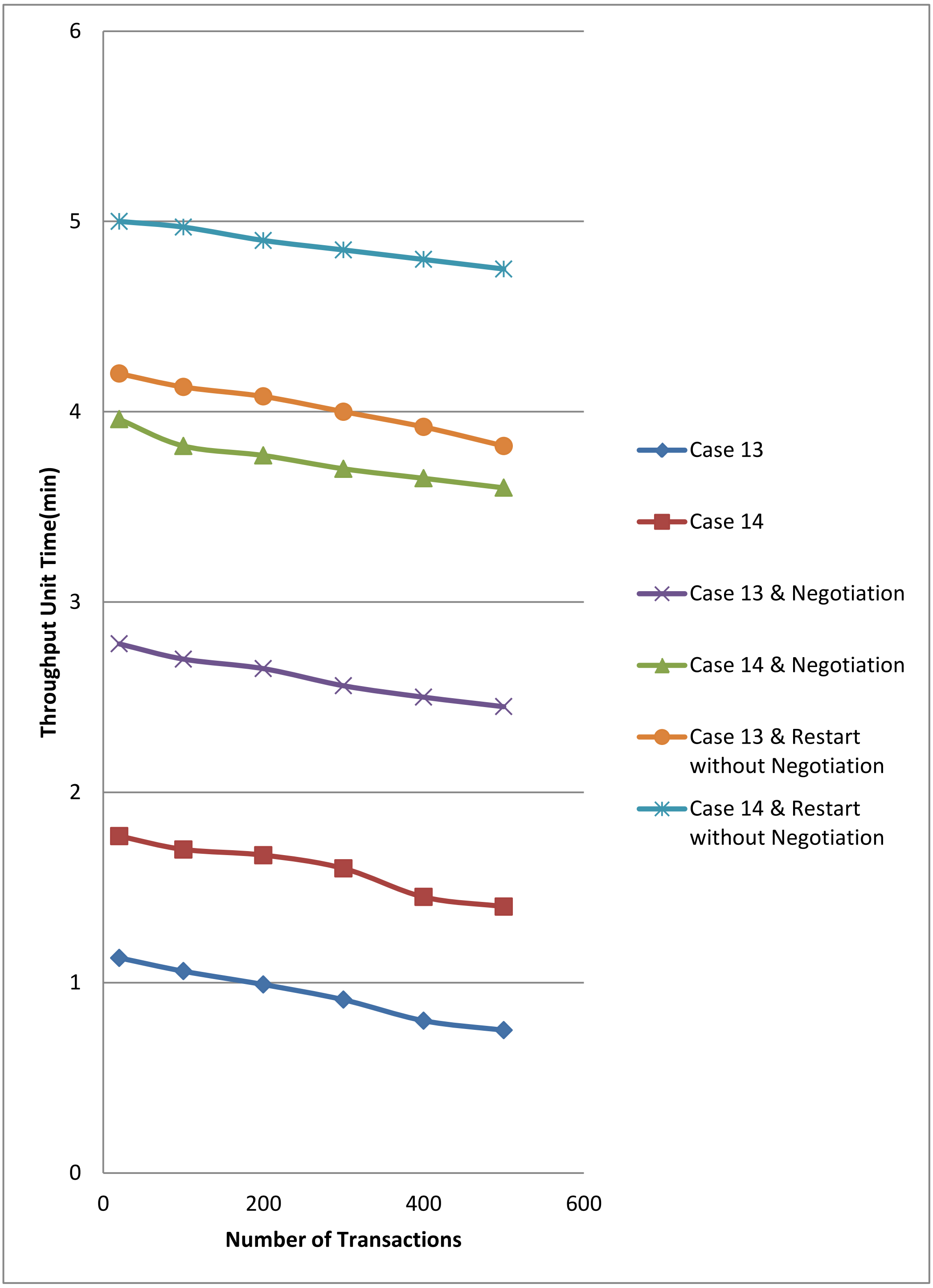


Figure 55 Case 13 and Case 14 ACID relaxation

The experiment shows that negotiation results in lower throughput compared to when a transaction does not go through negotiation. This is expected, as negotiation takes time. If negotiation succeeds the transaction will continue. If negotiation fails the transaction will have to restart. If one compares negotiation with restarting, it can be seen that negotiation has a better throughput than restarting. This is because in many cases negotiation will avoid an abort and restart and instead will allow a transaction to continue after a change in relaxation specification. Statistical analysis shows that the differences between all the lines in the graph are significant (see Appendix C).

6.4.4.2 Summary of Set 4 experiments

This set of experiments has shown that negotiation can be useful for improving throughput. Although negotiation takes additional processing time in the given batch run, it can avoid a transaction having to be aborted and restarted, which saves time in the bigger picture.

6.4.5 Set 5 – Assessing effect of tentative hold

Table 10 Service provider relaxation requirements – Experiment Set 5

Relax Atomicity	Relax Consistency	Relax Isolation	Relax Durability
YES	NO	NO	YES

This experiment set is based on the Big Party Arrangement scenario with Case 4, Case 5 and Case 10 user requirements assumed and 6 minutes tentative hold configured.

6.4.5.1 Experiment 14

The focus of this experiment was to analyse the effect of tentative hold. A variety of cases were chosen and for each case the use of tentative hold was measured. In this experiment, throughput is measured for Case 4 (just consistency relaxed), Case 5 (just isolation relaxed) and Case 10 (just durability relaxed) with relaxation of application-specific criteria and varying inclusion of tentative hold.

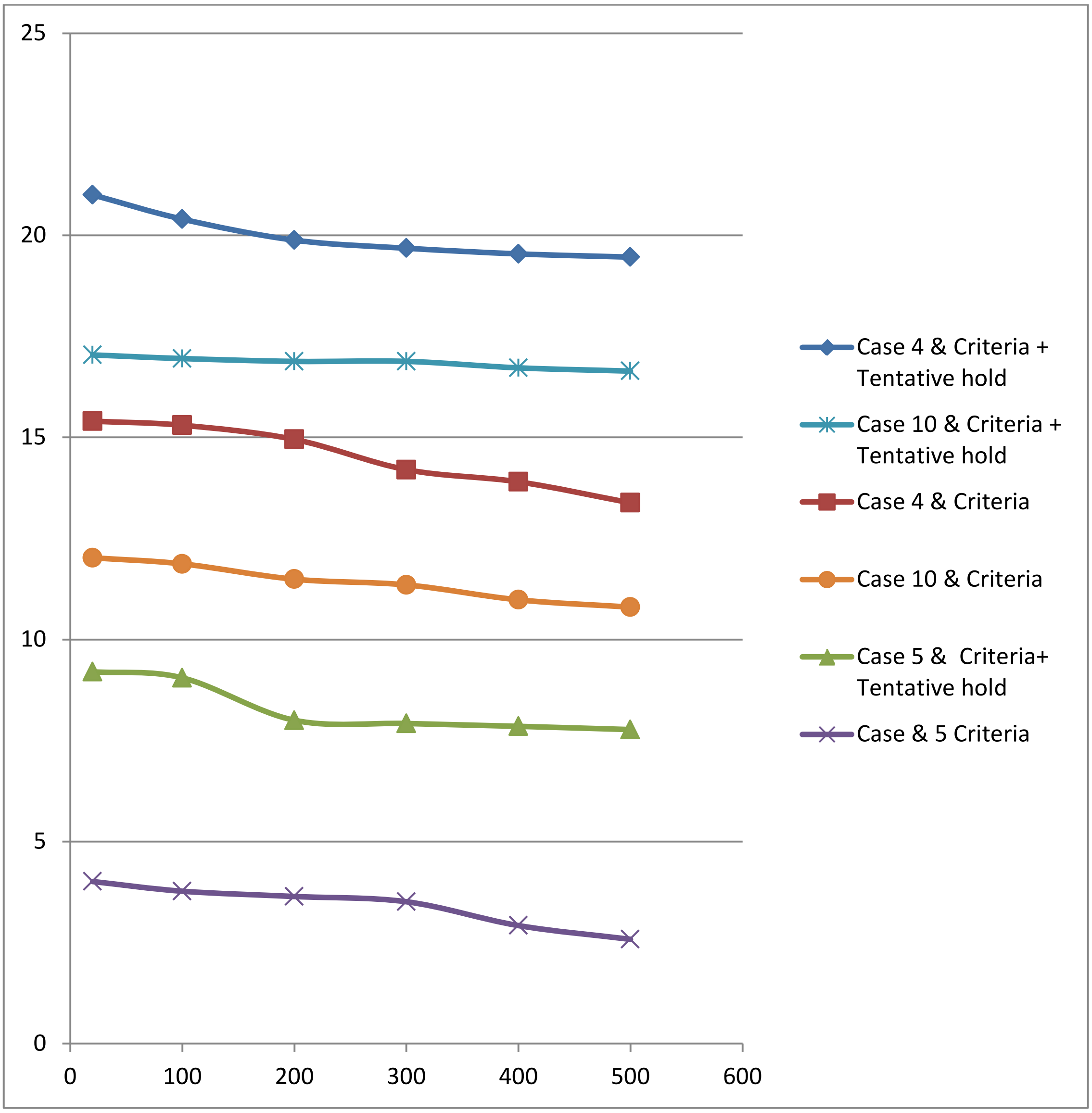


Figure 56 ACID and application-specific relaxation with tentative hold

It is clear that the tentative hold adds a considerable amount of processing time to the transaction. Statistical analysis confirms that this difference is significant (see Appendix C).

6.4.5.2 Experiment 15

The focus of this experiment was to analyse the effect of negotiation and tentative hold. The same cases were chosen as those used in Experiment 14 but this time the effect of both negotiation and tentative hold was measured. Note that tentative hold depends on the time the resource is set to be held. For example, the flight service might say that the ticket can be put on hold for two days, and when two days are up, the resource has to be confirmed or aborted. The more time the resource is held, the less the throughput.

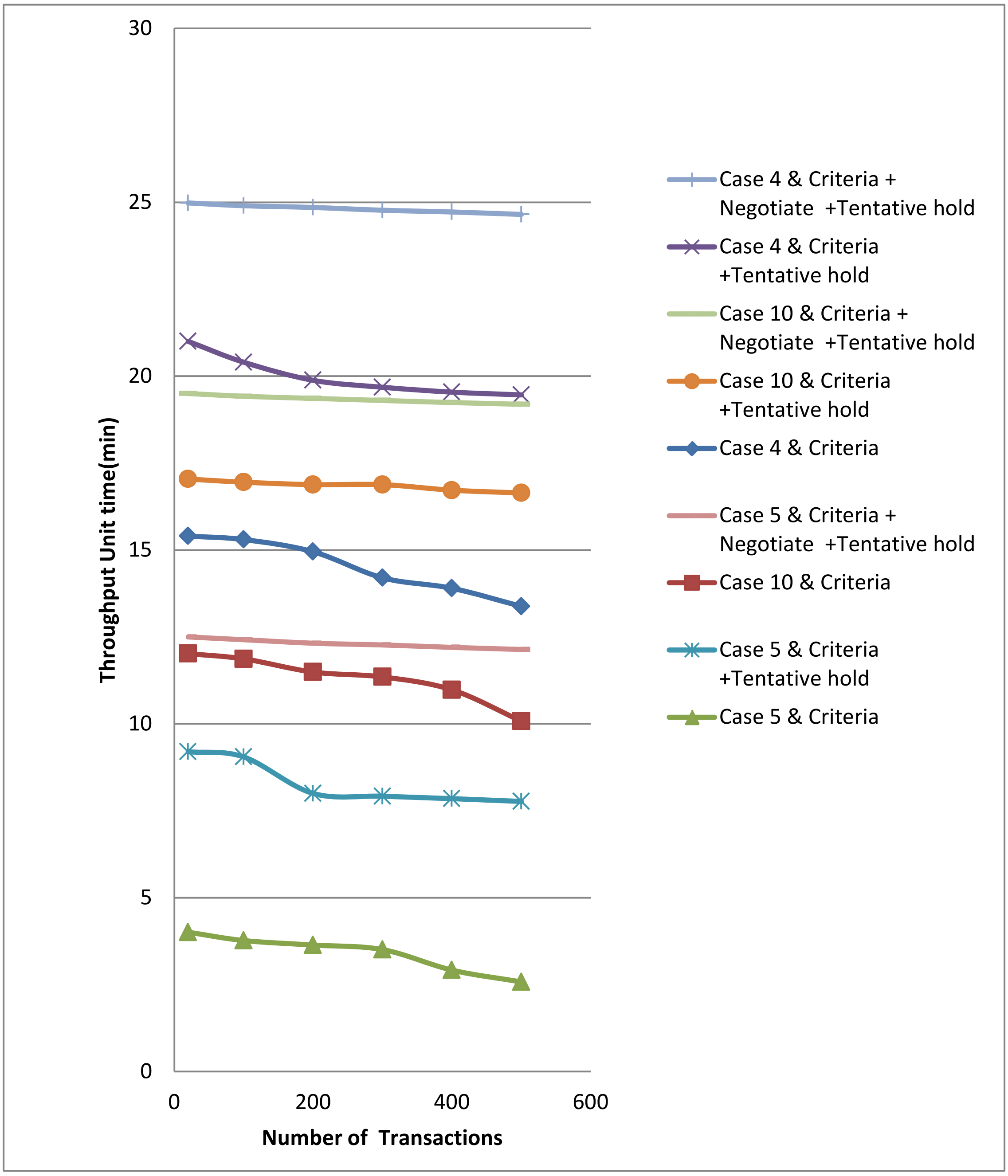


Figure 57 ACID, application-specific criteria, negotiation and tentative hold

From the experiment it is apparent that negotiation with tentative hold reduces the throughput of the transaction, depending on the time set for the resource to be on hold. Statistical analysis shows that the differences between the lines in the graph are statistically significant (see Appendix C)

6.4.5.3 Summary of Set 5 experiments

These experiments demonstrated that using tentative hold decreased throughput. However, tentative hold is necessary to ensure atomicity and also to maintain consistency when isolation is relaxed. Thus it will be needed in some circumstances in spite of decreasing throughput. The difference between the graph lines in this experiment set was statistically significant.

6.5 Summary

In this chapter, an experimental evaluation of AuTrA was conducted using assumed scenarios. The different scenarios assumed varied consumer and provider requirements for the purposes of evaluation of the system. Statistical analysis showed that the findings of all the experiments have statistical significance. Thus part of the research question was answered, namely that throughput could be increased by customisation of ACID and application-specific criteria, hence improving service. The results of the experiments are discussed in the next chapter.

Chapter 7: Discussion, Conclusion and Future Work

7.1 Introduction

This chapter discusses the results from the previous chapter and compares AuTrA with related work. In section 7.2, the discussion focuses on issues of throughput with respect relaxing ACID and application-specific properties. Section 7.3 discusses how AuTrA has answered the research question. Section 7.4 compares AuTrA with other models and section 7.5 comments on correctness when relaxing ACID properties. Section 7.6 looks back at this research in the context of the transition from centralised database to Web services, while section 7.7 discusses the targeted users of AuTrA. A conclusion to the research is given in section 7.8 and section 7.9 discusses future work.

7.2 Discussion

The experimental results verify that relaxation of ACID properties outperforms the classic ACID model, which is very rigid and is not suitable for long-running transactions, such as those found in Web service environments. It is clear that the number of ACID properties relaxed has an impact on the transaction throughput. That is, the more ACID properties are relaxed the better the transaction throughput. In addition, each individual ACID property generates a different level of throughput and this is a key to the throughput level of different combinations. Relaxing isolation has the best throughput of all the properties. This is because the transactions are processed concurrently; there is no waiting, which reduces throughput significantly. Relaxed durability is second to relaxed isolation. This is because when durability is relaxed there is no input/output (I/O) for writing which saves time. With atomicity and consistency relaxation, there is saving at the end of the transaction and that is the reason why relaxing these two have lower throughput compared to durability. When tidy-up (delayed saving) is used with durability relaxation, the increase in throughput is less than when tidy-up is not used. However in most circumstances it will be necessary to use tidy-up to preserve data integrity.

The consequence of application-specific criteria relaxation on the improvement of the service is a matter worthy of discussion. When the number of criteria which may be relaxed is greater, it means there will be more searches, reads and writes resulting in longer transaction-processing time. However, relaxation of application-specific criteria means that more transactions are successful, hence an improvement in throughput as shown in the experimentation. Overall the experiments showed that relaxing application-specific criteria improved throughput.

Negotiation is about giving the users who wished to relax either consistency or durability when the provider does not allow that, an opportunity to change their minds. If negotiation is successful, fewer ACID properties are relaxed. As seen from the results, the fewer ACID properties that are relaxed, the lower the transaction throughput. Allowing negotiation can be seen as a disadvantage in terms of throughput because of the reduction in the number of ACID properties relaxed and also the process of a provider and consumer negotiating adds time to the whole transaction. However negotiation may avoid restart, which would be even more time-consuming because the transaction concerned would have to start from the beginning rather than negotiating and continuing. Furthermore, successful negotiation avoids transaction abort, which includes compensation and is costly. Generally, this type of negotiation adds value to the transaction processing.

The novelty of the AuTrA system is the support for relaxation of ACID or application-specific properties according to the application's requirements and with the aim of increasing throughput. This allows service consumers to request to relax any ACID property but the service providers must define requirements regarding consistency and durability. This is important and fundamental in relation to maintaining correctness.

7.3 Answering the research question

To find whether the research aim has been achieved or not, let us reiterate the research question the thesis has endeavoured to answer:

Can transaction support for Web services be customised to suit the needs of varying applications and result in improved service?

Without a doubt, the research question has been answered through investigation of the problem, definition of the requirements, design and implementation of the AuTrA system as a prototype Web transaction management framework, and experimental evaluation of AuTrA using scenarios. AuTrA optionally relaxes ACID properties according to the users' needs, thereby meeting business requirements. This makes the AuTrA system suitable for varying applications and varying transaction requirements.

A research aim was developed from the research question. This was:

To develop a system that increases throughput while maintaining the consistency and correctness required by particular applications.

It was conjectured that the above research aim could be achieved by relaxing the ACID properties that are used in traditional transaction processing. A transaction management framework AuTrA and associated experimentation was designed to test this conjecture. The experimentation has shown that AuTrA enables throughput to increase while implementing safeguards to protect the data according to the requirements of the service provider. This is achieved through appropriate relaxation of ACID properties. A further conjecture was that relaxation of application-specific properties could also increase throughput. Experiments were designed to test this and it has been shown that relaxation of application-specific properties can indeed increase throughput.

From the research question arises another question:

> *Has any other research been done which also answers the thesis research question?*

From the related work, it is clear that a considerable amount of work has been done in relation to transaction management. This prompted the research to further investigate related work and identify the elements important in answering the research question. That is, this research discovered what is missing in the existing related work, has improved previous work and hopefully taken understanding of the subject area further forward.

The outline of how the research answered the research question in terms of transaction processing requirements is summarised in Table 11.

Table 11 Summary of how the AuTrA system answers the research question

Transaction processing requirements	How it is supported by the research
Customisation: User-defined atomicity and isolation	Allows users to specify the requirements
Correctness: Maintaining transaction consistency	Provider-specified relaxation
Customisation: Adaptable consistency and durability	Provider-specified relaxation
Customisation: Non-ACID improvement of throughput	Use of application-specific criteria
Improvement of throughput	Experiments show how relaxation of various properties and combinations of these improves throughput

7.4 Comparison with other models

The AuTrA system is different from other models. Detailed comparisons are provided in Appendix A, Appendix B and section 2.7, Comparison of the different approaches. The major differences are recapped in this section:

- **Optional relaxation of all ACID properties:** No other model relaxes all ACID properties. Some models like CaGIS-Trans customise relaxation of ACID properties, but only cater for atomicity and isolation. Ding, Wei and Huang's (2006) model also customises relaxation but caters for atomicity only.

- **Negotiation:** No other model, to the researcher's knowledge, gives the user a choice to rethink requirements.

- **Application-specific criteria relaxation:** No other model has formalised this aspect as a means of improving transaction throughput to the researcher's knowledge.

- The difference between standard protocols (such as WS-BA and BTP) and AuTrA is that relaxation of ACID properties in AuTrA is not required to be hardcoded like in the standard protocols. That is to say, for the standard protocols to relax ACID properties it has to be hardcoded by the programmer, which means it cannot be changed when the user wants to relax something different as may be required when needs change. Also these standard protocols do not include negotiation and application-specific criteria which might be useful to business in terms of increasing throughput.

7.5 Correctness when relaxing a property

A question arises of how AuTrA maintains correctness if ACID criteria are relaxed. In the classical database field, it is well known that consistency can be achieved through serialisation of concurrent transaction schedules (Silberschatz, Korth, and Sudarshan 2010). However, research has been done which shows that serialisation is a drawback when it comes to non-atomic transactions. For example Ramamritham and Chrysanthis in 1996 proposed a model which has more flexible correctness criteria. They developed a categorisation of different correctness criteria centred on database consistency requirements and transaction correctness properties. This model showed that relaxed serialisation is needed in distributed transactions. Another model which relaxed serialisation was the one by Guo, Tang, and Li, 2007. They argued that serialisation is too strict a correctness criterion for autonomous distributed systems. They proposed a model called weak serialisation. Their model separated the transactions into atomic transaction units according to application information. Interleaving amongst atomic transactions was allowed to increase parallelism. This model is a non-serialisation model and maintains consistency at higher levels of semantics.

Apart from the use of serialisation to maintain consistency, other different consistency methods like the use of data integrity or defining correctness rules can be used. For example, in flight-booking the number of bookings should not exceed the number of seats available. When relaxing durability and tidying up later, the “correct” database is produced from the memory transaction log, and when not tidying up it can be produced from the last checkpoint in the past transaction history log on secure storage (the absolute latest held in the memory log may be lost).

In terms of relaxing atomicity there is no loss of data consistency in the database. However, the user might receive inconsistent data when relaxing isolation that could have an effect on the correctness of the database. Therefore relaxation of isolation should only be used in non-critical applications. AuTrA leaves it to the user to decide whether the application is critical or not, but since a bad judgement could affect database correctness, the service provider has final decision on consistency and may use tentative hold in order to take care of incorrectness of data that might happen when tentative hold is not applied. Tentative hold will slow throughput but is necessary in some circumstances.

When it comes to consistency the correctness of the database is defined in terms of consistency rules, e.g. a rule might be the allowing of bookings even if the number of bookings exceeds the number of seats available. In other words, an inconsistent database according to the real world is deemed correct according to the business application, in that it meets the business rules. For instance, some airlines might deliberately overbook their flights, meaning that the number of bookings exceeding the number of seats is acceptable in the database. Sometimes overbooking may be to the level of a certain number of seats. The rule may be that the number of bookings must be less or equal to the number of seats plus ten. The point here is that consistency is relative to the application, and what is deemed acceptable in one application might be deemed as dangerously unacceptable in another. AuTrA allows flexibility, in that consistency can be relaxed according to the application requirements.

7.6 Transition from database to Web services

It is clear that the advance of technology has led to globalisation, more complex transactions, and the development of transaction management theory and practice from traditional centralised databases to decentralised internet environments. In today’s environment a transaction may involve businesses which are geographically located in different places, even in different continents. Such transactions are typically long-running and the centralised database approach cannot take care of this sort of transaction because originally it was designed for simpler centralised database transactions. The ACID properties are well established for transaction management in traditional databases. It was thought that this type of support would also be useful for Web business transactions. However, research showed that the ACID approach is too rigid for Web transactions. For this reason, this research has built on the findings of other research to develop a new transaction model that is flexible and suitable for the new environment.

7.7 Targeted users of AuTrA

It is envisaged that AuTrA will be used primarily in a software development house. Software developers will be able to use AuTrA to build applications for customers from various Web services that providers have offered up to the internet. The software developers will establish the user requirements regarding relaxation through analysis and discussion with users and will build suitable interfaces that support such requirements.

User-friendly interfaces can be built so that in some cases users can specify relaxation requirements dynamically. For instance, there could be a question such as: “If it is not possible to book the flight, hotel and skis, do you want the system to book whatever it can?” This type of question can be used to find out if a user is prepared to relax atomicity. To find out if a user is prepared to relax isolation the following sort of question can be asked: “Do you need your information quickly? – There may be other users updating the data at the moment – does it matter if the data provided to you is not absolutely the latest data? – If it doesn’t matter then your transaction can be run quicker.”

AuTrA will also be useful to system developers who might be experimenting with the response times for the system under development. Developers might want to use AuTrA to relax some of the ACID properties to find out the response time of each ACID property while the system is being developed, and depending on this decide on how to advise the customer on the final version and use of their application.

Another type of user might be a software-hosting company which hosts systems and applications for others in the cloud. The clients for their hosting company might be interested in high throughput for their application, for instance, sensor applications which gather data in real time and need fast processing. Strict durability and consistency is likely to be unnecessary in these types of application. The hosting company has possibilities of using AuTrA to relax some ACID properties that will suit the business needs of their clients’ applications and systems.

7.8 Conclusion

Efficient online transaction management is essential for the modern online business. Technology evolution has delivered Web services, which have become an essential component in the infrastructure for such business. Improvement in Web services transaction management is the key to smoothing the progress of business in terms of better transaction processing and hence better throughput. Much interesting work has lately contributed to the development of improved Web services transaction management. This has been described in the earlier part of this thesis, but it was noted that there was still room for a more flexible approach to Web transaction management and hence the formulation of the research question of this thesis.

The thesis has demonstrated how the research question has been answered. Experimental evaluation of AuTrA was performed and the results show that relaxation of ACID properties increases throughput, resulting in business enhancement. The experiments also showed that relaxation of non-ACID, application-specific requirements improve the throughput of transactions. AuTrA points out that consistency and durability of data is the responsibility of the business or service provider. That is to say, the providers set the relaxation specification that maintains consistency and durability. Users can opt to relax atomicity and isolation.

One might argue that if relaxing all ACID properties improves the service because it increases throughput, why not relax all ACID properties at all times? The research reiterates that relaxing consistency and durability sometimes might affect the data integrity of the business application. As a result, for the sake of the consistency of data, not all ACID properties can be relaxed, even if increased throughput is desired. In this case, the business must choose to maintain consistency of data over the throughput.

AuTrA has answered the research question showing that it is possible to optionally relax ACID properties according to the business requirements, while maintaining the correctness needed by the application. There is support for relaxation to be specified either by service provider or service consumer. The provider's relaxation specification determines which ACID properties can be relaxed in terms of consistency and durability, while the user determines which ACID properties can be relaxed with regard to atomicity and isolation. Negotiation gives the consumer a chance to rethink the requirements if they disagree with the provider's requirements. Another feature is application-specific criteria relaxation which supports alternative consumer requirements if the first requirement cannot be met. Tentative hold plays a role of maintaining consistency when isolation is relaxed and also when atomicity is not relaxed. AuTrA is a Web-based product, designed for the fast growing area of Web services, now becoming known as “the cloud”, where the growth of global and virtual businesses can flourish. AuTrA makes an important contribution to this area, which in the future will need to support many different types of application and will need innovative mechanisms to do so.

7.9 Future work

Whilst AuTrA has made some important contributions to the area of transaction management in Web services, there are still some issues that need to be addressed in the future. One of those issues is the dependency relationships between transactions which AuTrA at present does not consider. At the moment, the AuTrA system assumes that there is no dependency between the transactions, but if the AuTrA is to be used by any application, dependency mechanisms that will maintain consistency if needed by the business requirement have to be implemented. At present, AuTrA enables composition of applications through sequential composition only. More complex workflow composition facilities need to be added.

Another issue that needs attention in the future is consistency from the services consumer’s point of view. The AuTrA system takes care of consistency from the services provider’s side, but it does not have a mechanism in place that caters for consistency of information the services requester receives. For example, in a situation where isolation is relaxed, dirty reading may occur, and the consumer might receive incorrect information caused by dirty reading and might act upon it. At present, AuTrA handles this case by relaxing isolation only at the consumer’s request based on the assumption that the consumer understands the problem and is prepared to accept some risk of incorrect data. In many circumstances this is a reasonable assumption, but there could be other circumstances in which an additional safety net may be useful, notwithstanding that the inclusion of such a safety net would slow down processing. In the future, AuTrA should have some additional mechanism that will verify the consistency of information before the service requester receives it in cases with which it is felt that this feature would be useful.

Again, in the agenda for future work is the released AuTrA to be used by everyday users, which means they will be people who do not know anything about ACID properties and relaxation. Therefore, for the system to be used by the end user, the relaxation of ACID properties has to be communicated in simple English. For example, if the user wants to relax atomicity when booking a flight, a hotel and skis, the interface might offer a question as follows: “We might not be able to book everything. Which of the following is it essential that we book?” Then there would be a check box for the user to tick the components that are essential in case it is not possible to book all. This would be relaxing atomicity, because only part of the booking may be done. Relaxing isolation may be communicated by words such as the following: “In processing your transaction, we will access the very latest data but some of the data may not yet be fully verified. Is it okay to proceed in this way or alternatively would you prefer a slower process which accesses only verified data, even if it may not be the very latest data?” The sentiment could alternatively be communicated via option boxes. The application would only allow the user to ask for relaxation of atomicity and isolation, and the service provider would determine whether to relax consistency and durability. This would ensure the consistency and correctness of the database. The area of involving end users in the transaction configuration process will require further research and will be application-dependent to ensure that safely critical systems are not compromised. At present, AuTrA is seen primarily as a software developer’s tool.

7.10 Closing remark

More and more businesses are seeking an edge that can help them to be better than their competitors. As a result, the internet and e-commerce future will depend on high transaction throughput to meet the demands of businesses seeking that elusive edge. AuTrA delivers adaptable relaxation of ACID properties according to the business requirements, as well as using application-specific criteria to increase transaction throughput while still maintaining application-defined consistency. Thus it is hoped that this research work makes a positive contribution towards fulfilling future requirements of internet-based business systems as we move deeper into the internet age.

References

- Agrawal, D., Abbadi, A. E. and Singh, A. K. (1993). ‘Consistency and Orderability: Semantics-Based Correctness Criteria for Databases’. *ACM Transactions on Database Systems*, 18 (3), 460–486.
- Alrifai, M., Dolog, P. and Nejdl, W. (2006). ‘Transaction Concurrency Control in Web Service Environment’ in Bernstein, A., Gschwind, T. and Zimmermann, W. (eds.) *Proceedings of 4th IEEE European Conference of Web Services* held 4–6 December 2006 at Zurich IEEE Computer Society, 109–118.
- Awan, I. and Younas, M. (2004). ‘Analytical Modelling of Priority Commit Protocol for Reliable Web Applications.’ *19th ACM Symposium on Applied Computing* held 14–17 March 2004 at Nicosia. New York: ACM Press, 313–317.
- Bancilhon, F., Kim, W. and Korth, H. F. (1985). ‘A model of CAD transactions’ in Pirotte, A. and Vassiliou, Y. (eds.) *Proceedings of 11th International Conference on Very Large Data Bases* held 21–23 August 1985 at Stockholm. Morgan Kaufmann, 25–33.
- Bernstein, P. Hadzilacos, V. and Goodman, N. (1987). ‘*Concurrency Control and Recovery in Database Systems*.’ Reading: Addison-Wesley.
- Bernstein P. A. (1993). ‘Middleware – An Architecture for Distributed System Services.’ *Digital Equipment Corp., Cambridge Research Lab., Report* CRL 93/6.
- Bernstein P. A. (1996). ‘Middleware: A Model for Distributed System Services.’ *Communications of the ACM* 39 (2), 86–98.
- Bhiri, S., Perrin, O. and Godart, C. (2005) ‘Ensuring Required Failure Atomicity of Composite Web Services’ in Ellis, A. and Hagino, T. (eds.) *Proceedings of the 14th International Conference on World Wide Web, ACM WWW* held 10–14 May 2005 at Chiba. New York: ACM, 138–147.
- Biliris, A., Dar, S., Gehani, N. H., Jagadish, H. V. and Ramamritham, K. (1994). ‘ASSET: A System for Supporting Extended Transactions’ in Snodgrass, R. T. and Winslett, M. (eds.) *Proceedings of the ACM SIGMOD International Conference on Management of Data* held 24–27 May 1994 at Minneapolis. ACM Press, 44–54.
- Britannica Concise Encyclopaedia. Copyright 1994–2008.

- Böttcher, S., Gruenwald, L. and Obermeier, S. (2006) ‘Reducing Sub-Transaction Abort and Blocking Time within Atomic Commit Protocols.’ *Lecture Notes on Computer Science*. Springer, 4042/2006, 59–72.
- Cabrera, F., Copeland, G., Cox, B., Freund, T., Klein, J., Storey, T. and Thatte, S. (2001). ‘Web Services Transaction (WS-Transaction).’ *BEA Systems, International Business Machines Corporation, Microsoft Corporation, Inc.*, <http://www.ibm.com/developerworks/library/wstranspec>. [28 November 2010].
- Cabrera, L. F., Copeland, G., Cox, W., Feingold, M., Freund, T., Johnson, J., Kaler, C., Klein, J., Langworthy, D., Nadalin, A., Orchard, D., Robinson, I. Shewchuk, J., Storey, T. and Thatte, S. (2009a) ‘Web Services Atomic Transaction Framework (WSAtomicTransaction)’.<http://download.boulder.ibm.com/ibmdl/pub/software/ dw/library/WS-Atomic Transaction.pdf.> [12 February 2010]
- Cabrera, L. F., Copeland, G., Cox, W., Feingold, M., Freund, T., Johnson, J., Kaler, C., Klein, J., Langworthy, D., Nadalin, A., Orchard, D., Robinson, I., Shewchuk, J., Storey, T. and Thatte, S. (2009b). ‘Web Services Business Activity Framework (WSBusinessActivity)’.<http://download.boulder.ibm.com/ibmdl/pub/software/dw/ library/ WSBusinessActivity.pdf.> [12 February 2010]
- Cabrera, L. F., Copeland, G., Cox, W., Feingold, M., Freund, T., Johnson, J., Kaler, C., Klein, J., Langworthy, D., Nadalin, A., Orchard, D., Robinson, I. Shewchuk, J., Storey, T. and Thatte, S. (2009c). ‘Web Services Coordination Framework (WS-Coordination)’. http://download.boulder.ibm.com/ibmdl/ pub/software/dw/library/ WS-Coordination.pdf. [12 February 2010]
- Cambridge Dictionary Online (2011). Cambridge: Cambridge University Press. [2011]
- Ceri, S. and Pelagatpi, G. (1984). ‘Distributed Databases: Principles and Systems.’ New York: McGraw-Hill.
- Chen, P. P-S. (1976). ‘The Entity-Relationship Model – Toward a Unified View of Data.’ *ACM Transactions on Database Systems* 1 (1), 9–36.
- Chinnici, R., Moreau, J-J., Ryman, A. and Weerawarana, S. (2007). ‘Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, World Wide Web Consortium, Recommendation’. http://www.w3.org/TR/2007/REC-wsdl20-20070626. [12 May 2010]

- Choi, S., Jang, H., Kim, J., Kim, S. M., Song, J. and Lee, Y. (2005) 'Maintaining Consistency Under Isolation Relaxation of Web Services Transactions' in Anne H.H., Ngu, Kitsuregawa, M., Neuhold. E.J., Chung J-Y., Sheng, Q.Z. (eds.) *International Conference of Web Information System Engineering Proceedings of Lecture Computer Science* held on 20–22 November 2005 at New York. Springer 3807/2005, 245–257.
- Chrysanthis, P. K. and Ramamritham, K. (1990). 'ACTA: A Framework for Specifying and Reasoning about Transaction Structure and Behaviour' in Garcia-Molina, H. and Jagadish, H. V. (eds.) *Proceedings of the ACM, SIGMOD International Conference on Management of Data* held on June 1990 at Atlanta City. ACM Press, 19(2) 194–203.
- Codd, E. F. (1990) *'The Relational Model for Database Management.'* 2nd edn. Reading, Mass. Los Altos, CA: Addison Wesley Publishing.
- Conradi, R., Larsen, J. O., Nguyen, M., Wang, A. I. and Liu, C. (1997). 'Transaction Models for Software Engineering Database.' *Proceedings of Dagstuhl Workshop on Software Engineering Databases* held on 17–21 March 1997, FRG.
- Ding, X., Wei, J. and Huang, T. (2006). 'User-defined Atomicity Constraints: A More Flexible Transaction Model for Reliable Service Composition' in Liu. Z and J. He (ed.) *8th International Conference on Formal Engineering Methods. Proceedings of Lecture Notes in Computer Science* held on 1–3 November 2006 at Macao. Springer, *4260/2006*, 168–184.
- Elnikety, S. Tracey, J. Nahum, E and Zwaenepoel, W. (2004). "A Method for Transparent Admission Control and Request Scheduling in E-Commerce Web Sites" in Feldman, S.I. Uretsky, M. Najork, M. and Wills, C.E (eds.) *Proceedings of 13th World Wide Web Conference* held on 17–22 Manhattan. ACM, 276–286.
- Elmagarmid, A. K., (ed.) (1992). 'Transaction Models for Advanced Database Applications.' San Mateo, CA: Morgan Kaufmann.
- Eswaran, K., Grary, J., Lorie, R. and Traiger, I. (1976). 'The Notions of Consistency and Predicate Locks in a Database System' in Morgan H. (ed.) *Communication of the ACM* 19(11), 624–632.
- Fauvet, M. C., Duarte, H., Dumas, M. and Benatallah, B. (2005). 'Handling Transactional Properties in Web Service Composition' in Anne H.H., Ngu, Kitsuregawa, M., Neuhold. E.J., Chung J-Y., Sheng, Q. Z. (eds.) *Proceedings of 6th*

International Conference on Web Information Systems Engineering held on 20–22 November 2005 at New York. Springer: 3806/2005, 273–289.

- Fowler, M. (2003). '*UML Distilled: A Brief Guide to the Standard Object Modelling Language*'. 3rd edn. London: Addison-Wesley Professional
- Fussell, D., Kedem, Z. M. and Silberschatz, A. (1981). 'A Deadlock Removal Using Partial Rollback in Database Systems' in Lein, Y. E. (ed.). *Proceedings of the ACM SIGMOD International Conference on Management of Data* held on 29 April–1 May 1981 at Ann Harbor. ACM Press, 65–73.
- Garcia-Molina, H. and Salem, K. (1987). 'Sagas' in Dayal, U. and Traiger, I. L. (eds.) *Proceedings of the ACM International Conference on Management of Data, SIGMOD 87*, held on May 27–29 at San Francisco. ACM Press, 249–259.
- Gawlick, D. and Kinkade, D. (1985). 'Varieties of Concurrency Control in IMS/VS Fast Path'. *Database Engineering Bulletin* 8(2), 3–10.
- Glass, G. V. and Hopkins, K. D. (1984). '*Statistical Methods in Education and Psychology*'. 2nd edn. Englewood Cliffs, N. J. : Prentice-Hall.
- Godart, C. (1993) 'Coo: A Transactional Model to Support Cooperating Software Developers Coordination' in Sommerville, I. and Paul, M. (eds.) *4th European Software Engineering Conference, Proceedings Lecture Notes in Computer Science.* held on 13–17 September at Garmisch-Partenkirchen. Springer, 717, 361–379.
- Gray, J., McJones, P. R., Blasgen, M. W., Lindsay, B. G., Lorie, R. A., Price, T. G., Putzolu, G. R. and Traiger, I. L. (1981). 'The recovery manager of the system R database manager'. *ACM Computing Surveys*, 13(2), 223–243.
- Gray, J. and Reuter, A. (1993). '*Transaction Processing: Concepts and Techniques'*. San Mateo, CA. Morgan Kaufmann.
- Green, P. E., Tull, D. S. and Albaum, G. (1993). '*Research for Marketing Decisions' 5th edn.*, Upper Saddle River, N. J. and Prentice-Hall.
- Greenfield, P., Fekete, A., Jang, J., Kuo, D. and Nepal, S. (2007) 'Isolation support for services-based applications: a position paper.' in Roscoe, T (ed.) *Proceedings of 3rd Biennial Conference on Innovative Data Systems Research (CIDR)* held on 7–10 January 2007 at Asilomar. 314–323.
- Guo, Y., Xi J., Tang, D. and Li, X. (2007). 'Correctness Criterion for Transaction Management in Loosely Coupled System' in Li. K., Jesshope, C. R., Jin, H. and Gaudiot, J-L. (eds.) *IFIP International Conference on Network and Parallel*

Computing-Workshop. *Proceedings of Lecture Notes in Computer Science* held on September 18–21 2007 at Dalian. Springer, 975–982.

- Haller, K. Schuldt, H. and Türker, C. (2005). 'Decentralised coordination of transactional processes in peer-to-peer environments' in Herzog, O., Schek, H-J., Fuhr, N., Chowdhury, A. and Teiken, W. (eds.) *Proceedings of the ACM CIKM, International Conference on Information and Knowledge Management* held on 31 October to 5 November 2005, at Bremen. ACM, 28–35.
- IBM SolidDB (2009) [1 August 2010]
- Kaiser, G. E. and Pu, C. (1992). 'Dynamic Restructuring of Transactions' in Elmagarmid, A. K., (ed.) *Database Transaction Models for Advanced Applications*. San Mateo, CA. Morgan Kaufmann Publishers, 265–295.
- Kerlinger F. N. (1973). *'Foundations of Behavioural Research'* 2nd edn. London: Holt, Rinehard and Winston.
- Kim, W., Lorie, R. A., McNabb, D. and Plouffe, W. (1984). 'A Transaction Mechanism for Engineering Design Databases' in Dayal, U., Schlageter, G. and Seng, L. H. (eds.) *Proceedings of the 10th International Conference on Very Large Data Bases* held on 27–31 August 1984 at Singapore. Morgan Kaufmann, 355–362.
- Kohler W. (1981). 'A Survey of Techniques for Synchronisation and Recovery in Decentralised Computer Systems', *ACM Computing Survey*. New York, N.Y., 13(2), 149–183.
- Kumar, S. and Barvey, S. (2009). 'Non-Blocking Commit Protocol (NBCP)'. *International Journal of Computer Science and Network Security* 9(8), 172–177.
- Kung H. T., and Robinson, J. T. (1981) 'On Optimistic Methods for Concurrency Control'. *ACM Transactions on Database Systems*, 6(2), 213–226.
- Lee, B., Lim, S., Kim J. H. and Fox, G. C. (2009). 'Lease-based Consistency Schemes in the Web Environment.' *Future Generation Computer Systems* 25 (1), 8–19.
- Limthanmaphon, B. and Zhang, Y. (2004). 'Web Service Composition Transaction Management' in Schewe, K-D and Williams, H.E. (eds.). *Proceedings of 15th Australasian Database Conferences* held on 18–22 January 2004 at Dunedin. Australian Computer Society, 171–179.
- Lin, Y. and Son, S. H. (1990). 'Concurrency Control in Real-Time Database by Dynamic Adjustment of Serialization Order.' *Proceedings of the 11th IEEE Real-*

Time Systems Symposium held on 5–7 December 1990 at Lake Buena Vista. IEEE Computer Society Press, 104–112.

- Little, M. and Freund, T. (2003). ‘A Comparison of Web Services Transaction Protocols’. IBM DeveloperWorks article, http://www-106.ibm.com/ developerworks/Webservices/library/ws-comproto/.> [12 February 2010]
- Luck, D. J. and Rubin, R. S. (1992). ‘*Marketing Research’*. New Delhi: Prentice-Hall of India.
- Lynch, N. A. (1983). ‘Multilevel atomicity – A New Correctness Criterion for Database Concurrency Control.’ *ACM Transactions on Database Systems*, 8 (4). 484–502.
- McGovern, J., Tyagi, S., Stevens, M. and Mathew, S. (2003) ‘*Java Web Services Architecture.’* San Fransisco, CA. Morgan Kaufmann.
- Moss, J. E. B. (1982). ‘Nested Transactions and Reliable Distributed Computing.’ *Proceedings of the 2nd IEEE Symposium on Reliability in Distributed Software and Database Systems, SRDSDS 1982* held on August 1982 at Pittsburgh. ACM New York, N.Y. 33–39.
- Navathe, S. B. and Schkolnick, M. (1978). ‘View Representation in Logical Database Design’ in Lowenthal, E. I. and Dale, N. B. (eds.) *Proceedings of SIGMOD Conference* held on 31 May–2 June 1978 at Austin. ACM, 144–156.
- Nodine, M. H. and Zdonik, S. B. (1992). ‘Cooperative Transaction Hierarchies: Transaction Support for Design Applications’. *VLDB Journal*, 1 (1), 41–80.
- Oracle TimesTen in-memory database (2009) [1 August 2010].
- Park, J. and Choi, K. S. (2003). ‘Design of an Efficient Tentative Hold Protocol for Automated Coordination of Multi-Business Transactions.’ *Proceedings IEEE International Conference on Electronic Commerce* held on 24–27 June 2003 at Newport Beach, CA. IEEE Computer Society, 215–222.
- Peltz, C. (2003). ‘Web Services Orchestration and Choreography.’ Hewlett-Packard Company. CA, USA. *IEEE Computer Society* Press 36 (10), 46–52.
- Pitoura, E. and Bhargava, B. (1999). ‘Data Consistency in Intermittently Connected Distributed Systems.’ *IEEE Transactions on Knowledge and Data Engineering* 11 (6). 896–915.

- Rahman, Q. A., Abubakar, A., Sirajuddin, S., Islam, S. M. S. and Razzaque, M. A. (2006). ‘Marker-free Human Motion Capture.’ .Crawley: The University of Western Australia
- Ramamritham, K. and Chrysanthis, P. K. (1996). A Taxonomy of Correctness Criteria in Database Applications.’ *VLDB* (*Very Large Data Bases*) 5 (1),85–97.
- Ramampiaro, H. and Nydard, M. (2004). ‘CAGIS-Trans: Providing Adaptable Transactional Support for Cooperative Work-Extended Treatment.’ *Information Technology and Management*, 5(1–2), 23–64.
- Roberts, J., and Srinivasan, K. (2001). ‘Tentative hold policy part 1: white paper.’ <http://www.w3.org/TR/tenthold-1.> [28 November 2010]
- Roberts, J., Collier, T., Malu, P. and Srinivasan, K. (2001). ‘Tentative hold protocol, part 2: technical specification.’ <http://www.w3.org/TR/2001/NOTEtenthold-2-20011128/.> [28 November 2011]
- Rosenkrantz, D., Stearns, R. and Lewis, P. (1978). ‘System-level Concurrency Control for Distributed Database Systems.’ *ACM Transactions. Database System* 3 (2), 178–198.
- Ruh, W., Maginnis, F. and Brown, W. (2000). ‘*Enterprise Application Integration: A Wiley Tech Brief*.’ New York, N.Y., USA: John Wiley & Sons, Inc.
- Rusinkiewicz, M., Klas, W., Tesch, T., Walsch, J. and Muth, P. (1995). ‘Towards a Cooperative Transaction Model: The Cooperative Activity Model’ in Dayal, U., Gray, P. M. D., Nishio, S. (eds.) *Proceedings of 21st on VLDB (Very Large Data Base) Conference* held on 11–15 September 1995 at Zurich. Morgan Kaufmann, 194–205.
- Silberschatz, A., Korth, H. F. and Sudarshan, S. (2010). ‘*Database System Concepts.*’ 6th edn. New York: McGraw-Hill.
- Spahni, S., Scherrer, J. R., Sauquet, D. and Sottile, P. A. (1998). ‘Consensual Trends for Optimizing the Constitution of Middleware.’ *ACM SIGCOMM Computer Communication Review* 28 (5).
- Sybase (2009). ‘Getting Started with In-Memory Database in Adaptive Server Enterprise 15.5’ [1 August 2010].
- Sybase (2009). ‘ ASE In-Memory Databases. Business White Paper’ [1 August 2010].

- Terry, D. B., Theimer, M. M., Petersen, K., Demers, A. J., Spreitzer, M. J. and Hauser, C. H. (1995). ‘Managing Update Conflicts in Bayou, A Weakly Connected Replicated Storage System.’ in Jones, M.B. (ed.) *Proceedings of 15th ACM Symposium on Operating Systems Principles* held on 3–6 December 1995 at Copper Mountain Resort. ACM New York, 172–183.
- Wang, X-J., Li, X-M. and Min, L-J. (2009). ‘Ensuring Consistency of Web services Transaction.’ *The Journal of China Universities of Posts and Telecommunications* 16 (4), 59–66.
- Wächter, H. and Reuter, A. (1992). ‘The ConTract Model’ in Elmagarmid, A.K. (ed.) *Database Transaction Models for Advanced Applications*. San Mateo, CA: Morgan Kaufmann, 219–263.
- Weikum, G. and Scheck, H. J. (1992). ‘Concepts and Applications of Multilevel Transactions and Open Nested Transactions’ in Elmagarmid, A. K. (ed.) *Database Transaction Models for Advanced Applications*. San Mateo, CA: Morgan Kaufmann, 515–553
- Wäsch, J. (1999). ‘Transactional Support for Cooperative Applications’, PhD thesis, GMD/IPSI and Darmstadt University of Technology.
- Yang, X. W., Liu, Z. and Ling, W. X. (2006) ‘taTHP: A Transaction-Aware Protocol for Web Services Transaction Coordination.’ *Proceedings of TENCON 2006 IEEE Region 10 Conference* held on 14–17 November 2006 at Hong Kong IEEE, 1–4.
- Younas, M., Eaglestone, B. and Holten, R. (2000). ‘A Review of Multidatabase Transactions on the Web. From the ACID to the SACReD’ in Lings, B. and Jeffery, K.G. (ed.). *British National Conference on Databases*, BNCOD. *Proceedings of Lecture Notes in Computer Science* held on 3–5 July at Exeter. Springer, 140–152.
- Younas, M. and Iqbal, R. (2003). ‘Developing Collaborative Editing Applications Using Web services.’ *IEEE Distributed Systems Online Journal of Collaborative Computing* 4 (9).
- Younas, M., Eaglestone, B. and Chao, K-M. (2004) ‘A low-latency resilient protocol for e-business transactions.’ *International Journal of Web Engineering and Technology* 1(3), 278–296.
- Younas, M., Chao, K. M., Lo, C. C. and Li, Y. (2006). ‘An Efficient Transaction Commit Protocol for Composite Web Services’ in Martins, D. (ed.) *Proceedings of*

The IEEE 20th International Conference on Advanced Information Networking and Applications held on 18–20 April at Vienna. IEEE Computer Society, 591–596.

- Younas, M. and Chao, K. (2006). 'A Tentative Commit Protocol for Composite Web Services.' *Journal of Computer System Science* 72 (7), 1226–1237.
- Younas, M., Awan, I. and Duce, D. (2006). 'An efficient composition of Web services with active network support.' *Expert Systems with Applications* 31 (4), 859–869.
- Younas, M. and Mostéfaoui, S. K. (2010) 'Context-aware Mobile Services Transactions' in Hussain, F. (ed.) *Proceedings of 24th IEEE International Conference on Advanced Information Networking and Applications* held 20–23 April 2010 at Perth. IEEE Computer Society, 705–712.
- Yu, H. and Vahdah, A. (2002). 'Design and Evaluation of Conit-based Continuous Consistency for Replicated Services.' *ACM Transactions on Computer Systems* 20 (3), 239–282.
- Zhang, Y., Kambayashi, Y., Jia, X., Yang Y. and Sun, C. (1999). 'On Interactions Between Co-existing Traditional and Cooperative Transactions.' *International Journal of Cooperative Information Systems* 8 (2–3), 87–10.
- Zhao, W., Moser, L. E. and Melliar-Smith, P. M. (2008) 'A Reservation-Based Extended Transaction Protocol.' *IEEE Transactions on parallel and distributed systems* 19, (2), 188–203.
- Zhou, F., Jin, C. and Zheng, W. (2004). 'TODS: Cluster Object Storage Platform Designed for Scalable Services.' *Future Generation Computer Systems* 20(4), 549–563.
- Zhou, W., Wang, L. and Jia, W. (2004). 'An Analysis of Update Ordering in Distributed Replication Systems.' *Future Generation Computer Systems* 20 (4), 565–590.

Appendix A – Comparison of different Web-based transaction management models

Table A1 summarises the difference and similarities between the various Web-based transaction models that have been studied which includes authors and year of publication.

Table A 1 Comparison of different transaction Web-based transaction models

Year	Model	Author(s)	Relaxation				Distinguishing feature(s)	Customisable	Management of inconsistency	Reconsidering the requirements by users
			A	C	I	D				
1982	Nested transaction	Moss	N	N	N	N	Sub-transactions(parent and child)	N	N/A	N
1983	Modulation specification	Lynch	Y	N	N	N	Multilevel atomicity	N	N/A	N
1987	Saga	Garcia-Molina and Salem	Y	N	Y	N	Compensation similar to nested transaction but caters for long transactions	N	Compensation	N
1990	ACTA	Chrysanthis and Ramamritham	N	N	N	N	Customisable, captures semantics and rationale for the recovery properties of the composite transactions; commit and abort-dependency	Y	N/A	N

1992	Open-nested and multilevel	Weikum and Schek	Y	N	Y	N	Use of semantics-based concurrency control; serialisability as the correctness criterion	N	Serialisability as the correctness criterion	N
1992	Cooperation transaction hierarchy	Nodine and Zdonik	N	N	N	N	Three levels of the tree: root, transaction groups, cooperation transactions; user-defined criteria (patterns and conflicts)	Y	N/A	N
1992	Split and join	Keiser and Pu	N	N	N	N	*Splits* a running transaction into two or more transactions and later *joins* transactions by merging their resources	N	N/A	N
1993	Coo	Godart	Y	N	Y	N	Three consistency levels: stable, semi-stable, unstable	N	Safety constraints	N
1993	Semantics-based correctness criteria	Agrawal, Abbadi and Singh	Y	N	N	N	Relative atomicity	N	N/A	N
1994	ASSET (A System Supporting Extended Transactions)	Biliris et al.	N	N	N	N	Introduced new primitives: delegation, dependency, conflict set	N	N/A	N
1995	Cooperative	Rusinkiewicz	Y	N	N	N	User investigation	Y	N/A	N

	transaction	et al.					facility; retraction of decision; compensation			
1995	Bayou	Terry et al.	N	Y	N	N	Use of replicate to allow relaxation of consistency	N	Update conflicts resolved in a consistent manner by all servers	N
1997	Software engineering database	Conradi et al.	N	N	Y	N	Software engineering focus; two types of locking: mandatory and cooperative; user awareness; conflict change detection	N	User awareness; conflict change detection	N
1997	RTF(Reflective Transaction Framework)	Barga and Pu	N	N	N	N	Transactional adapters: transaction manager adapter, lock adapter, conflict adapter, log adapter	N	N/A	N
1999	Data consistency	Pitoura and Bhagava	N	Y	N	N	Clusters: weak and strong transactions	N	Data integrity constraints are ensured only for data copies belonging to the same logical	N

									clusters	
1999	Transactional Support for Cooperative Applications	Wäsch	Y	N	Y	N	Caters for ad-hoc activities	Y	Not discussed	N
1999	NTO (New Timestamp Ordering)	Zhang et al.	N	N	N	N	Use of high priority on the last read or last write conflict write for correctness criteria	Y	N/A	N
2001	Tentative Hold Protocol	Robert and Srinivasan	N	N	Y	N	Tentative non-blocking holds; user awareness	N	Tentative Hold	N
2002	Design and Evaluation of Conit-based Continuous Consistency for Replicated Services.	Yu and Vahdah	N	Y	N	N	Replicate services; consistency spectrum	N	Allowance of certain inconsistency level; If this is passed operation will be blocked until synchronisations of the replicate as determined by the system consistency	N
2003	SACReD	Younas and Iqbal	N	N	Y	N	Collaborative editing focus; semantics atomicity; resilience	N	Correctness criteria based on SACReD	N
2003	BTP	Little and Freud	Y	N	Y	N	Two sub-protocols (atoms and cohesion)	Y	Compensation	N
2003	Efficient THP	Park and Choi	N	N	Y	N	Optimisation of THP through adaptive hold duration	Y	Tentative hold	N
2004	TODS	Zhou, Jin and	N	Y	N	N	Different levels of	N	Local consistency	N

		Zheng					consistency			
2004	Update ordering	Zhou, Wang and Jia	N	Y	N	N	Replication with data update ordering	N	Differentiates needs of clients and server in maintaining consistency	N
2004	Priority Commit Protocol	Awan and Younas	N	N	N	N	Head-of-line (HoL) scheduling mechanisms at network nodes.	N	N/A	N
2004	Low-latency resilient	Younas Eaglestone and Chao	Y	N	Y	N	Flexible components transaction, i.e. use of alternatives	Y	Correctness criteria based on SACReD	N
2004	CAGIS-Trans	Ramampiaro and Nydard	Y	N	Y	N	Focus on cooperative work; adaptability of atomicity and isolation relaxation;	Y	Users specify suitable correctness constraints	N
2005	Accepted termination states	Bhiri, Perrin and Godart	Y	N	N	N	Flexible definition by user acceptable termination states	Y	N/A	N
2005	Composite model	Fauvet et al.	Y	N	Y	N	Based on THP; different levels of atomicity.	Y	N/A	N
2005	Decentralised coordination of transaction process in peer-to-peer environment	Haller, Schuldt, and Türker	N	N	Y	N	Global correctness without depending on global serialisation graph.	N	Correctness ensured decentralised serialisation graph testing	
2005	A framework for ensuring	Choi et al.	N	N	Y	N	Web services transaction dependency	N	The mechanism effectively detects	N

	consistency of Web services transactions						management Protocol (WTDP).		inconsistent states of transactions with a notion of a *completion dependency* and recovers them to consistent states	
2006	User-defined atomicity	Ding ,Wei and Huang	Y	N	N	N	Adaptive user-defined atomicity	Y	N/A	N
2006	Tentative commit protocol	Younas and Chao	N	N	N	N	Similar to THP but tentative commit replaces tentative holds	N	N/A	N
2006	Transactions concurrency control in Web services environment	Alrifai, Dolog and Nejdl	N	N	Y	N	Non-blocking scheduler	N	Commit order preserving transaction scheduler	N
2006	Transaction Commit Protocol for Composite Web Services	Younas et.al	Y	N	Y	N	Alternative transaction	N	Tentative commit	N
2006	Transaction awareness protocol	Yang, Liu and Ling	N	N	Y	N	THP transaction context awareness; success probability	Y	Tentative hold	N

2006	Reducing sub-transaction abort and blocking time within atomic commit protocol	Böttcher, Gruenwald, and Obermeier, S.	N	N	Y	N	Non-blocking and reduction of transaction aborts	N	Not discussed	N
2006	Network-based Composition (NetCom)	Younas, Awan and Duce	N	N	N	N	P2P architecture	N	N/A	N
2007	Promise	Greenfield et al.	N	N	N	N	Resources held based on promise; no other transaction allowed to see a promised resource; provides isolation	N	N/A	N

2007	Tentative Hold Protocol (THP)	Robert et al.	N	N	Y	N	Compensation	N	Compensation	N
2008	Reservation-based extended protocol	Zhao, Moser and Melliar-Smith	N	N	Y	N	No use of compensation	N	Not discussed	N
2009	WS-BusinessActivity	Cabrera et al. 2009b	Y	N	Y	N	Coordinates a set of distributed Web services to reach a jointly outcome; includes WS-Coordinator	Y	Compensation	N
2009	Non-Blocking Commit Protocol (NBCP)	Kumar and Barvey	N	N	Y	N	Two-phase commit (2PC) protocol	N	Not discussed	N
2009	Lease-based consistency	Lee et al.	N	Y	N	N	Replicates; lease time adaptively	Y	Lease time	N
2009	Ensuring consistency on Web services transactions	Wang et al.	N	N	Y	N	Web services transaction dependency coordination protocol (WSTDCP)	N	End state dependency	N
2009	IBM SolidDB		N	N	N	Y	HotStandby	Y	Delayed saving	N
2009	IBM Universal Cache		N	N	N	Y	In-memory	Y	Dual Database	N
2009	ASE		N	N	N	Y	In-memory and in-memory/disk-based	Y	Dual Database	N
2009	Oracle Ten-Times		N	N	N	Y	Transaction replicates	Y	Replicates	N

2010	Context aware	Younas and Mostefaoui	Y	Y	Y	N	Context awareness	Y	Correctness criteria based on SACReD	N
2011	AuTrA	Khachana, James and Iqbal	Y	Y	Y	Y	Full adaptability to relax ACID properties depending on the requirements and contracts; use of negotiation phase	Y	Service provider’s specification	Y

Appendix B – Comparison of key features of BTP, WS-Tx (BusinessActivity) and AuTrA

This appendix provides a summary table which compares business standard protocols with AuTrA. The following table summarises the differences and similarities.

Table B 1 Comparison of key features of BTP, WS-Tx (BusinessActivity) and AuTrA

	BTP	**WS-BusinessActivity**	**AuTrA**
Distinguishing Feature	Two sub-protocols (atoms and cohesion)	Coordinates a set of distributed Web services (example, Atomic Transactions) to reach a joint outcome; includes WS-Coordination; dynamic	Full adaptability to relax ACID properties depending on the requirements and contracts; use of negotiation phase
Atomicity Relaxation	Yes	Yes	Yes
Consistency Relaxation	No	No	Yes
Isolation Relaxation	Yes	Yes	Yes
Durability Relaxation	No	No	Yes
Customisable	No	No	Yes
Review of the Characteristics Entered by the User	No	No	Yes
Management of Inconsistency	Compensation	Compensation	service providers application contracts

Appendix C – Raw data and evidence of statistical analysis of the experiments

This appendix presents the raw data and the statistical analysis of the experiments. There are different methods that can be used to statistically analyse the data like CHITEST, t-test, ANOVA (Analysis of Variance), and Tukey Test.

CHITEST – **chi-squared (χ^2) test**. Chi-squared tests are only appropriate for frequency data, i.e. counts. Since the research data are not frequency data, this test is not suitable for the analysis.

t-test – This is used to compare two groups (referred to as Cases in this research) of observations, but since the research has more than two groups to compare, this is not the correct method to statistically analyse the data, because of multiple comparison which can lead to high error rates.

ANOVA– This is used to find the significance of differences between groups. This method was suitable in this research because the raw data consists of many groups (in this research these are called Cases). ANOVA can be used for experiments that involve single or multiple factors. In the case of this research there were 2 factors which influenced the results. These were the cases (identified by the particular relaxation specification) and the transaction batch size. Thus 2 way ANOVA was the correct method to use for this research. ANOVA assumes that the data from the different groups come from populations where the observations have a normal distribution and the variance is the same for each group. ANOVA can determine that there is significant difference between groups but does not show which of the groups are statistically significantly different.

Tukey's test – also referred to Tukey's *HSD (Honest Significant Difference)* test is a multiple comparison method used following ANOVA to find which means are significantly different from one another. Tukey's test and t-test are based on a similar formula. However Tukey's test is more acceptable than t-test, because it has a lower rate of error when doing multiple comparisons. Therefore the Tukey test is more suitable for multiple comparisons. Tukey’s test is more specific than ANOVA. Tukey’s test shows which groups from a set are statistically significantly different.

In statistical significance testing, the **p-value** is the probability of getting a test statistic at least as large as the one that was actually observed, assuming that the null hypothesis is true. The null hypothesis is rejected when the p-value is less than the significance level α, which is often 0.05 or 0.1. The result is said to be statistically significant when the null hypothesis is rejected. ANOVA reports a p-value and an F-ratio. The F-ratio describes the variance of the group means. The F-ratio is calculated as a factor of the largest variance over the smallest variance of the group means across the groups. The null hypothesis is rejected if the F-ratio is higher than a crucial value as given in established statistical tables and dependent on degrees of freedom (DF). In the results provided below, the F-ratio is presented as $F_{x,y}$ where $_X$ and y are the relevant DFs of the factor under consideration, namely Cases, and the Error respectively. The Error refers to the interaction between the factors, Cases and Transactions.

This research used Minitab for undertaking the ANOVA and Tukey tests.

Statistical Significance Testing

In the following sections the data from each experiment presented in Chapter 6 is processed in order to check statistical significance. For each experiment a table is given which shows the number of transactions in each batch and the throughput unit time for each case tested.

Experiment 1 Raw data and Tukey analysis output

Table C 1 Case 1 and Case 2 Raw Data

No. of transactions in a set	**Case 1**	**Case 2**
20	16.04	1.43
100	15.99	1.39
200	15.89	1.30
300	15.85	1.20
400	15.79	1.20
500	15.65	1.19

H0 – There is no difference in transaction throughput between groups, i.e. between (a) relaxing no ACID properties and (b) relaxing Atomicity, Consistency and Isolation.

H1– There is a difference in transaction throughput between groups, i.e. between (a) relaxation of no ACID properties and (b) relaxing Atomicity, Consistency and Isolation.

General Linear Model: Throughput Unit Time versus Transactions; Cases

```
Factor        Type   Levels  Values
Transactions  fixed       6  20; 100; 200; 300; 400; 500
Cases         fixed       2  Case1; Case2


Analysis of Variance for Throughput Unit Time, using Adjusted SS for Tests

Source        DF  Seq SS  Adj SS  Adj MS          F      P
Transactions   5    0,14    0,14    0,03      13,92  0,006
Cases          1  638,02  638,02  638,02  307727,09  0,000
Error          5    0,01    0,01    0,00
Total         11  638,18


S = 0,0455339   R-Sq = 100,00%   R-Sq(adj) = 100,00%


Unusual Observations for Throughput Unit Time

     Throughput
Obs   Unit Time      Fit  SE Fit  Residual  St Resid
  6     15,6500  15,7117  0,0348   -0,0617     -2,10 R
 12      1,1900   1,1283  0,0348    0,0617      2,10 R

R denotes an observation with a large standardized residual.


Grouping Information Using Tukey Method and 95,0% Confidence

Cases  N    Mean  Grouping
Case1  6  15,868  A
Case2  6   1,285    B
```

```
Means that do not share a letter are significantly different.


Tukey Simultaneous Tests
Response Variable Throughput Unit Time
All Pairwise Comparisons among Levels of Cases
Cases = Case1  subtracted from:

       Difference       SE of            Adjusted
Cases    of Means  Difference  T-Value    P-Value
Case2      -14,58     0,02629   -554,7     0,0000
```

The research reports from ANOVA , the F ratio, with both DFs , and the p-value. The research also reports the results from Tukey. There were significant differences between cases (2 way ANOVA gives $F_{1, 5}$ = 307727.09 and p<0.001). Tukey’s HSD test confirmed both cases were significantly different from one another.

Interpreting the results

In the ANOVA table, the p-value <0.001 and Tukey ’s HSD test confirmed all cases were significantly different from one another so the research rejects the null hypothesis that “There is no difference in transaction throughput between groups, i.e. between (a) relaxation of no ACID properties and (b) relaxing Atomicity, Consistency and Isolation.”

Experiment 2 Raw data and Tukey analysis output

Table C 2 Case 3, Case 4 and Case 5 Raw Data

No. of transactions in a set	**Case 3**	**Case 4**	**Case 5**
20	10.58	9.66	1.94
100	10.52	9.51	1.89
200	10.50	9.44	1.84
300	10.37	9.37	1.74
400	10.22	9.29	1.68
500	10.19	9.13	1.60

H0 – There is no difference in transaction throughput between groups, i.e. between (a) relaxing Atomicity and (b) relaxing Consistency and (c) relaxing Isolation.

H1 – There is a difference in transaction throughput between groups, i.e. between (a) relaxing Atomicity and (b) relaxing Consistency and (c) relaxing Isolation.

General Linear Model: Throughput Unit Time versus Transactions; Cases

```
Factor        Type   Levels  Values
Transactions  fixed       6  20; 100; 200; 300; 400; 500
Cases         fixed       3  Case3; Case4; Case5
```

```
Analysis of Variance for Throughput Unit Time, using Adjusted SS for Tests

Source        DF   Seq SS   Adj SS   Adj MS         F      P
Transactions   5    0,371    0,371    0,074     47,81  0,000
Cases          2  266,501  266,501  133,251  85845,04  0,000
Error         10    0,016    0,016    0,002
Total         17  266,888


S = 0,0393983   R-Sq =  99,99%   R-Sq(adj) = 99,99%


Unusual Observations for Throughput Unit Time

     Throughput
Obs   Unit Time     Fit  SE Fit  Residual  St Resid
  7      9,6600  9,6006  0,0263    0,0594      2,02 R

R denotes an observation with a large standardized residual.


Grouping Information Using Tukey Method and 95,0% Confidence

Cases  N    Mean  Grouping
Case3  6  10,397  A
Case4  6   9,400    B
Case5  6   1,782      C

Means that do not share a letter are significantly different.


Tukey Simultaneous Tests
Response Variable Throughput Unit Time
All Pairwise Comparisons among Levels of Cases
Cases = Case3  subtracted from:

       Difference       SE of           Adjusted
Cases    of Means  Difference  T-Value   P-Value
Case4      -0,997     0,02275    -43,8    0,0000
Case5      -8,615     0,02275   -378,7    0,0000
```

```
Cases = Case4  subtracted from:

       Difference       SE of            Adjusted
Cases    of Means  Difference  T-Value    P-Value
Case5      -7,618     0,02275   -334,9     0,0000
```

The research reports from ANOVA , the F ratio, with both DFs, and the p-value. The research also reports the results from Tukey. The research reports the F ratio, with both DFs, and the p-value. The research also reports the results from Tukey. There were significant differences between cases (2 way ANOVA gives $F_{2, 10} = 85845.04$, $p<0.001$). Tukey’s HSD test confirmed all cases were significantly different from one another.

Interpreting the results

In the ANOVA table, the p-value <0.001 and Tukey ’s HSD test confirmed all cases were significantly different from one another so the research rejects the null hypothesis that “There is no difference in transaction throughput between groups, i.e. between (a) relaxing Atomicity and (b) relaxing Consistency and (c) relaxing Isolation.”

Experiment 3 and Experiment 4 Raw Data and Tukey Analysis Output

Table C 3 Case 2, Case 6, Case 7 and Case 8 Raw Data.

No. of transactions in a set	**Case 2**	**Case 6**	**Case 7**	**Case 8**
20	1.43	7.93	1.80	1.72
100	1.39	7.89	1.76	1.69
200	1.30	7.82	1.72	1.63
300	1.20	7.71	1.68	1.59
400	1.20	7.64	1.64	1.52
500	1.19	7.55	1.54	1.49

H0 - There is no difference in transaction throughput between groups, i.e. between (a) relaxing Atomicity, Consistency and Isolation and (b) relaxing Atomicity with Consistency and (c) relaxing Atomicity with Isolation and (d) relaxing Consistency with Isolation

H1- There is a difference in transaction throughput between groups, i.e. between (a) relaxing Atomicity, Consistency and Isolation and (b) relaxing Atomicity with Consistency and (c) relaxing Atomicity with Isolation and (d) relaxing Consistency with Isolation.

General Linear Model: Throughput Unit Time versus Transactions; Cases

```
Factor        Type   Levels  Values
Transactions  fixed       6  20; 100; 200; 300; 400; 500
Cases         fixed       4  Case2; Case6; Case7; Case8


Analysis of Variance for Throughput Unit Time, using Adjusted SS for Tests

Source        DF   Seq SS   Adj SS  Adj MS         F      P
Transactions   5    0,232    0,232   0,046     36,30  0,000
Cases          3  175,176  175,176  58,392  45747,76  0,000
Error         15    0,019    0,019   0,001
Total         23  175,427


S = 0,0357266   R-Sq = 99,99%   R-Sq(adj) = 99,98%


Unusual Observations for Throughput Unit Time

     Throughput
Obs   Unit Time      Fit   SE Fit  Residual  St Resid
  6     7,55000  7,61458  0,02188  -0,06458     -2,29 R

R denotes an observation with a large standardized residual.


Grouping Information Using Tukey Method and 95,0% Confidence

Cases  N   Mean  Grouping
Case6  6  7,757  A
Case7  6  1,690    B
Case8  6  1,607      C
Case2  6  1,285        D

Means that do not share a letter are significantly different.


Tukey Simultaneous Tests
Response Variable Throughput Unit Time
```

```
All Pairwise Comparisons among Levels of Cases
Cases = Case2  subtracted from:

       Difference       SE of            Adjusted
Cases    of Means  Difference  T-Value   P-Value
Case6      6,4717     0,02063   313,75    0,0000
Case7      0,4050     0,02063    19,63    0,0000
Case8      0,3217     0,02063    15,59    0,0000


Cases = Case6  subtracted from:

       Difference       SE of            Adjusted
Cases    of Means  Difference  T-Value   P-Value
Case7      -6,067     0,02063   -294,1    0,0000
Case8      -6,150     0,02063   -298,2    0,0000


Cases = Case7  subtracted from:

       Difference       SE of            Adjusted
Cases    of Means  Difference  T-Value   P-Value
Case8    -0,08333     0,02063   -4,040    0,0053
```

The research reports from ANOVA , the F ratio, with both DFs, and the p-value. The research also reports the results from Tukey.. There was significant differences between cases (2 way ANOVA gives $F_{3,\ 15}$ = 45747.76, $p<0.001$). Tukey’s HSD test confirmed all cases were significantly different from one another.

Interpreting the results

In the ANOVA table, the p-value <0.001 and Tukey 's HSD test confirmed all cases were significantly different from one another so the research rejects the null hypothesis that "There is no difference in transaction throughput between groups, i.e. between (a) relaxing Atomicity, Consistency and Isolation and (b) relaxing Atomicity with Consistency and (c) relaxing Atomicity with Isolation and (d) relaxing Consistency with Isolation."

Experiment 5 Raw data and Tukey analysis output

Table C 4 Case 9 and Case 10 Raw Data

No. of transactions in a set	**Case 9**	**Case 10**
20	16.77	8.47
100	16.55	8.37
200	16.44	8.32
300	16.33	8.23
400	16.21	8.12
500	16.10	8.01

H0 – There is no difference in transaction throughput between groups, i.e. between (a) relaxing no ACID properties and (b) relaxing Durability

H1– There is a difference in transaction throughput between groups, i.e. between (a) relaxing no ACID properties and (b) relaxing Durability.

General Linear Model: Throughput Unit Time versus Transactions; Cases

```
Factor        Type   Levels  Values
Transactions  fixed       6  20; 100; 200; 300; 400; 500
Cases         fixed       2  Case10; Case9


Analysis of Variance for Throughput Unit Time, using Adjusted SS for Tests

Source        DF   Seq SS   Adj SS   Adj MS         F      P
Transactions   5    0,418    0,418    0,084     24,61  0,002
Cases          1  199,105  199,105  199,105  58675,21  0,000
Error          5    0,017    0,017    0,003
Total         11  199,539


S = 0,0582523   R-Sq = 99,99%   R-Sq(adj) = 99,98%


Unusual Observations for Throughput Unit Time

     Throughput
Obs   Unit Time      Fit  SE Fit  Residual  St Resid
  1     16,7700  16,6933  0,0445    0,0767      2,04 R
  7      8,4700   8,5467  0,0445   -0,0767     -2,04 R

R denotes an observation with a large standardized residual.


Grouping Information Using Tukey Method and 95,0% Confidence

Cases   N    Mean  Grouping
Case9   6  16,400  A
Case10  6   8,253    B

Means that do not share a letter are significantly different.


Tukey Simultaneous Tests
Response Variable Throughput Unit Time
All Pairwise Comparisons among Levels of Cases
```

```
Cases = Case10  subtracted from:

       Difference       SE of            Adjusted
Cases    of Means  Difference  T-Value    P-Value
Case9       8,147     0,03363    242,2     0,0000
```

The research reports from ANOVA , the F ratio, with both DFs, and the p-value. The research also reports the results from Tukey. There was significant differences between cases (2 way ANOVA gives $F_{1,\ 5}$ = 58675.21, p<0.001). Tukey’s HSD test confirmed all cases were significantly different from one another.

Interpreting the results

In the ANOVA table, the p-value <0.001 and Tukey’s HSD test confirmed all cases were significantly different from one another so the research rejects the null hypothesis that “There is no difference in transaction throughput between groups, i.e. between (a) relaxing no ACID properties and (b) relaxing Durability.”

Experiment 6 and Experiment 7 Raw Data and Tukey Analysis Output

Table C 5 Case 10 & Tidy-up, Case 10, Case 11 & Tidy-up and Case11 Raw Data

No. of transactions in a set	**Case 10 & Tidy-up**	**Case 10**	**Case 11 Tidy-up**	**Case 11**
20	14.00	8.47	13.44	7.69
100	13.89	8.37	13.36	7.63
200	13.80	8.32	13.30	7.54
300	13.75	8.23	13.24	7.49
400	13.70	8.12	13.19	7.37
500	13.67	8.01	13.15	7.21

H0 – There is no difference in transaction throughput between groups, i.e. between (a) relaxing Durability and (b) relaxing Durability with Tidy-up and (c) relaxing Durability and Atomicity and (d) relaxing Durability and Atomicity with Tidy-up

H1– There is a difference in transaction throughput between groups, i.e. between (a) relaxing Durability and (b) relaxing Durability with Tidy-up and (c) relaxing Durability and Atomicity and (d) relaxing Durability and Atomicity with Tidy-up.

General Linear Model: Throughput Unit Time versus Transactions; Cases

```
Factor        Type   Levels  Values
Transactions  fixed       6  20; 100; 200; 300; 400; 500
Cases         fixed       4  Case10; Case10+ Tidy Up; Case11; Case11+ Tidy Up
```

```
Analysis of Variance for Throughput Unit Time, using Adjusted SS for Tests

Source        DF   Seq SS   Adj SS  Adj MS         F      P
Transactions   5    0,407    0,407   0,081     44,92  0,000
Cases          3  195,465  195,465  65,155  35975,24  0,000
Error         15    0,027    0,027   0,002
Total         23  195,899


S = 0,0425572   R-Sq = 99,99%   R-Sq(adj) = 99,98%


Unusual Observations for Throughput Unit Time

     Throughput
Obs   Unit Time     Fit  SE Fit  Residual  St Resid
 24      7,2100  7,2925  0,0261   -0,0825     -2,45 R

R denotes an observation with a large standardized residual.


Grouping Information Using Tukey Method and 95,0% Confidence

Cases             N    Mean  Grouping
Case10+ Tidy Up   6  13,802  A
Case11+ Tidy Up   6  13,280    B
Case10            6   8,253      C
Case11            6   7,488        D

Means that do not share a letter are significantly different.


Tukey Simultaneous Tests
Response Variable Throughput Unit Time
All Pairwise Comparisons among Levels of Cases
Cases = Case10  subtracted from:

                 Difference       SE of           Adjusted
Cases              of Means  Difference  T-Value   P-Value
Case10+ Tidy Up      5,5483     0,02457   225,81    0,0000
Case11              -0,7650     0,02457   -31,14    0,0000
```

```
Case11+ Tidy Up      5,0267     0,02457   204,58    0,0000


Cases = Case10+ Tidy Up  subtracted from:

                 Difference       SE of           Adjusted
Cases              of Means  Difference  T-Value   P-Value
Case11               -6,313     0,02457   -256,9    0,0000
Case11+ Tidy Up      -0,522     0,02457    -21,2    0,0000


Cases = Case11  subtracted from:

                 Difference       SE of           Adjusted
Cases              of Means  Difference  T-Value   P-Value
Case11+ Tidy Up       5,792     0,02457    235,7    0,0000
```

The research reports from ANOVA , the F ratio, with both DFs, and the p-value. The research also reports the results from Tukey. There was significant differences between cases (2 way ANOVA gives $F_{3,15} = 35975.24$, $p<0.001$). Tukey's HSD test confirmed all cases were significantly different from one another.

Interpreting the results

In the ANOVA table, the p-value <0.001 and Tukey's HSD test confirmed all cases were significantly different from one another so the research rejects the null hypothesis that "There is no difference in transaction throughput between groups, i.e. between (a) relaxing Durability and (b) relaxing Durability with Tidy-up and (c) relaxing Durability and Atomicity and (d) relaxing Durability with Atomicity with Tidy-up."

Experiment 8 Raw Data and Tukey Analysis Output

Table C 6 Case 3, Case 4 and Case 5 and Case 10 Raw Data

No. of transactions in a set	**Case 3**	**Case 4**	**Case 5**	**Case 10**
20	12.40	11.37	2.36	8.47
100	12.31	11.29	2.21	8.37
200	12.23	11.14	2.10	8.32
300	12.17	11.05	2.01	8.23
400	12.10	11.06	1.98	8.12
500	12.03	11	1.90	8.01

H0 – There is no difference in transaction throughput between groups, i.e. between (a) relaxing Atomicity, and (b) relaxing Consistency, (c) relaxing Isolation, (d) relaxing Durability.

H1 – There is a difference in transaction throughput between groups, i.e. between (a) relaxing Atomicity, and (b) relaxing Consistency, (c) relaxing Isolation, (d) relaxing Durability.

General Linear Model: Throughput Unit Time versus Transactions; Cases

```
Factor        Type   Levels  Values
Transactions  fixed       6  20; 100; 200; 300; 400; 500
Cases         fixed       4  Case 10; Case 3; Case 4; Case 5
```

```
Analysis of Variance for Throughput Unit Time, using Adjusted SS for Tests

Source        DF   Seq SS   Adj SS   Adj MS         F      P
Transactions   5    0,570    0,570    0,114     20,59  0,000
Cases          3  371,031  371,031  123,677  22328,82  0,000
Error         15    0,083    0,083    0,006
Total         23  371,684


S = 0,0744237   R-Sq = 99,98%   R-Sq(adj) = 99,97%


Unusual Observations for Throughput Unit Time

     Throughput
Obs   Unit Time     Fit  SE Fit  Residual  St Resid
 20      8,7000  8,4958  0,0456    0,2042      3,47 R

R denotes an observation with a large standardized residual.


Grouping Information Using Tukey Method and 95,0% Confidence

Cases    N    Mean  Grouping
Case 3   6  12,207  A
Case 4   6  11,152    B
Case 10  6   8,308      C
Case 5   6   2,093        D

Means that do not share a letter are significantly different.


Tukey Simultaneous Tests
Response Variable Throughput Unit Time
All Pairwise Comparisons among Levels of Cases
Cases = Case 10  subtracted from:

        Difference       SE of           Adjusted
Cases     of Means  Difference  T-Value   P-Value
```

```
Case 3       3,898     0,04297     90,7    0,0000
Case 4       2,843     0,04297     66,2    0,0000
Case 5      -6,215     0,04297   -144,6    0,0000


Cases = Case 3  subtracted from:

        Difference       SE of           Adjusted
Cases     of Means  Difference  T-Value   P-Value
Case 4       -1,05     0,04297    -24,6    0,0000
Case 5      -10,11     0,04297   -235,4    0,0000


Cases = Case 4  subtracted from:

        Difference       SE of           Adjusted
Cases     of Means  Difference  T-Value   P-Value
Case 5      -9,058     0,04297   -210,8    0,0000
```

The research reports from ANOVA , the F ratio, with both DFs, and the p-value. The research also reports the results from Tukey. There was significant differences between cases (2 way ANOVA gives $F_{3,15}$ = 22328,82, p<0.001). Tukey’s HSD test confirmed all cases were significantly different from one another.

Interpreting the Results

In the ANOVA table, the p-value <0.001 and Tukey’s HSD test confirmed all cases were significantly different from one another so the research rejects the null hypothesis that “There is no difference in transaction throughput between groups, i.e. between (a) relaxing Atomicity, and (b) relaxing Consistency, (c) relaxing Isolation, (d) relaxing Durability.”.

Experiment 9 Raw Data and Tukey Analysis Output

Table C 7 Cases 6, Case 7 and Case 8, Case 16, Case 17 and Case 18 Raw Data

No. of transactions in a set	**Case 6**	**Case7**	**Case 8**	**Case 16**	**Case 17**	**Case 18**
20	12.01	3.75	2.70	9,71	1.97	2.01
100	11.97	3.69	2.65	9,67	1.95	1.99
200	11.85	3.69	2.60	9,53	1.90	1.98
300	11.74	3.61	2.51	9,41	1.85	1.95
400	11.59	3.55	2.51	9,14	1.82	1.91
500	11.47	3.50	2.46	9,01	1.80	1.89

H0 – There is no difference in transaction throughput between groups, i.e. between (a) relaxing Atomicity and Consistency (b) relaxing Consistency and Isolation (c) relaxing Atomicity and Isolation (d) relaxing Atomicity, Consistency and Durability and (e) relaxing Consistency, Isolation and Durability and (f) relaxing Atomicity, Isolation and Durability.

H1 – There is a difference in transaction throughput between groups, i.e. between (a) relaxing Atomicity and Consistency (b) relaxing Consistency and Isolation (c) relaxing Atomicity and Isolation (d) relaxing Atomicity, Consistency and Durability and (e) relaxing Consistency, Isolation and Durability and (f) relaxing Atomicity, Isolation and Durability.

General Linear Model: Throughput Unit Time versus Transactions; Cases

```
Factor        Type   Levels  Values
Transactions  fixed       6  20; 100; 200; 300; 400; 500
Cases         fixed       6  Case 16; Case 17; Case 18; Case 6; Case 7; Case 8


Analysis of Variance for Throughput Unit Time, using Adjusted SS for Tests

Source        DF   Seq SS   Adj SS   Adj MS         F      P
Transactions   5    0,535    0,535    0,107     11,45  0,000
Cases          5  550,610  550,610  110,122  11791,91  0,000
Error         25    0,233    0,233    0,009
Total         35  551,378


S = 0,0966374   R-Sq = 99,96%   R-Sq(adj) = 99,94%


Unusual Observations for Throughput Unit Time

     Throughput
Obs   Unit Time     Fit  SE Fit  Residual  St Resid
 24      9,0100  9,2303  0,0534   -0,2203     -2,74 R

R denotes an observation with a large standardized residual.


Grouping Information Using Tukey Method and 95,0% Confidence

Cases    N    Mean  Grouping
Case 6   6  11,767  A
Case 16  6   9,412    B
Case 7   6   3,632      C
Case 8   6   2,572        D
Case 18  6   1,955          E
Case 17  6   1,882          E

Means that do not share a letter are significantly different.
```

```
Tukey Simultaneous Tests
Response Variable Throughput Unit Time
All Pairwise Comparisons among Levels of Cases
Cases = Case 16  subtracted from:

         Difference       SE of            Adjusted
Cases      of Means  Difference  T-Value    P-Value
Case 17      -7,530     0,05579   -135,0     0,0000
Case 18      -7,457     0,05579   -133,6     0,0000
Case 6        2,355     0,05579     42,2     0,0000
Case 7       -5,780     0,05579   -103,6     0,0000
Case 8       -6,840     0,05579   -122,6     0,0000


Cases = Case 17  subtracted from:

         Difference       SE of            Adjusted
Cases      of Means  Difference  T-Value    P-Value
Case 18     0,07333     0,05579    1,314     0,7745
Case 6      9,88500     0,05579  177,171     0,0000
Case 7      1,75000     0,05579   31,366     0,0000
Case 8      0,69000     0,05579   12,367     0,0000


Cases = Case 18  subtracted from:

        Difference       SE of            Adjusted
Cases     of Means  Difference  T-Value    P-Value
Case 6      9,8117     0,05579   175,86     0,0000
Case 7      1,6767     0,05579    30,05     0,0000
Case 8      0,6167     0,05579    11,05     0,0000


Cases = Case 6  subtracted from:

        Difference       SE of            Adjusted
Cases     of Means  Difference  T-Value    P-Value
Case 7      -8,135     0,05579   -145,8     0,0000
Case 8      -9,195     0,05579   -164,8     0,0000
```

```
Cases = Case 7  subtracted from:

        Difference       SE of            Adjusted
Cases     of Means  Difference  T-Value    P-Value
Case 8      -1,060     0,05579   -19,00     0,0000
```

The research reports from ANOVA , the F ratio, with both DFs, and the p-value. The research also reports the results from Tukey. There was significant differences between cases (2 way ANOVA gives $F_{5,25}$ = 11791,91, p<0.001). Tukey's HSD test confirmed all cases were significantly different from one another apart from Case 17 and Case 18.

Interpreting the Results

In the ANOVA table, the p-value <0.001 and Tukey's HSD test confirmed all cases were significantly different from one another so the research rejects the null hypothesis that "There is no difference in transaction throughput between groups, i.e. between (a) relaxing Atomicity and Consistency (b) relaxing Consistency and Isolation (c) relaxing Atomicity and Isolation (d) relaxing Atomicity, Consistency and Durability and (e) relaxing Consistency, Isolation and Durability and (f) relaxing Atomicity, Isolation and Durability".

Experiment 10 Raw data and Tukey analysis output

Table C 8 Case 12 and Case 13 Raw Data

No. of transactions in a set	**Case 12**	**Case 13**
20	19.51	1.13
100	19.05	1.06
200	18.77	0.99
300	18.03	0.91
400	17.67	0.80
500	17.34	0.75

H0 – There is no difference in transaction throughput between groups, i.e. between (a) relaxing all ACID properties and application-specific criteria and (b) relaxing no ACID properties and no application-specific criteria

H1– There is a difference in transaction throughput between groups, i.e. between (a) relaxing all ACID properties and application-specific criteria and (b) relaxing no ACID properties and no application-specific criteria.

General Linear Model: Throughput Unit Time versus Transactions; Cases

```
Factor        Type   Levels  Values
Transactions  fixed       6  20; 100; 200; 300; 400; 500
Cases         fixed       2  Case 12; Case 13


Analysis of Variance for Throughput Unit Time, using Adjusted SS for Tests

Source        DF  Seq SS  Adj SS  Adj MS        F      P
Transactions   5    2,47    2,47    0,49     2,02  0,230
Cases          1  914,03  914,03  914,03  3730,51  0,000
Error          5    1,23    1,23    0,25
Total         11  917,73


S = 0,494990   R-Sq = 99,87%   R-Sq(adj) = 99,71%


Grouping Information Using Tukey Method and 95,0% Confidence

Cases    N     Mean  Grouping
Case 12  6  18,3950  A
Case 13  6   0,9400    B

Means that do not share a letter are significantly different.


Tukey Simultaneous Tests
Response Variable Throughput Unit Time
All Pairwise Comparisons among Levels of Cases
Cases = Case 12  subtracted from:

         Difference       SE of           Adjusted
Cases      of Means  Difference  T-Value   P-Value
Case 13      -17,46      0,2858   -61,08    0,0000
```

The research reports from ANOVA , the F ratio, with both DFs, and the p-value. The research also reports the results from Tukey. There was significant differences between cases (2 way ANOVA gives $F_{1,5} = 3730.51$, $p<0.001$). Tukey's HSD test confirmed all cases were significantly different from one another.

Interpreting the results

In the ANOVA table, the p-value <0.001 and Tukey's HSD test confirmed all cases were significantly different from one another so the research rejects the null hypothesis that "There is no difference in transaction throughput between groups, i.e. between (a) relaxing all ACID properties and application-specific criteria and (b) relaxing no ACID properties and no application-specific criteria."

Experiment 11 Raw data and Tukey analysis output

Table C 9 Case 13, Case 14, Case 15 and Case 19 Raw Data

No. of transactions in a set	**Case 13**	**Case 14**	**Case 15**	**Case 19**
20	1.13	1.63	1.87	2.30
100	1.06	1.59	1.81	2.25
200	0.99	1.55	1.79	2.19
300	0.91	1.49	1.75	2.17
400	0.80	1.45	1.69	2.10
500	0.75	1.40	1.63	2.06

H0 – There is no difference in transaction throughput between groups, i.e. between (a) relaxing all ACID properties and application-specific criteria and (b) relaxing all ACID properties and no application-specific criteria and (c) relaxing Atomicity, Consistency, Isolation and application-specific criteria and (d) relaxing Atomicity, Consistency, Isolation and no application-specific criteria.

H1– There is a difference in transaction throughput between groups, i.e. between (a) relaxing all ACID properties and application-specific criteria and (b) relaxing all ACID properties and no application-specific criteria and (c) relaxing Atomicity, Consistency, Isolation and application-specific criteria and (d) relaxing Atomicity, Consistency, Isolation and no application-specific criteria.

General Linear Model: Throughput Unit Time versus Transactions; Cases

```
Factor        Type   Levels  Values
Transactions  fixed       6  20; 100; 200; 300; 400; 500
Cases         fixed       4  Case 13; Case 14; Case 15; Case 19


Analysis of Variance for Throughput Unit Time, using Adjusted SS for Tests

Source        DF   Seq SS   Adj SS   Adj MS        F      P
Transactions   5  0,21018  0,21018  0,04204    41,71  0,000
Cases          3  4,80763  4,80763  1,60254  1590,18  0,000
Error         15  0,01512  0,01512  0,00101
Total         23  5,03293


S = 0,0317455   R-Sq = 99,70%   R-Sq(adj) = 99,54%


Unusual Observations for Throughput Unit Time

     Throughput
Obs   Unit Time      Fit   SE Fit  Residual  St Resid
```

```
 19     1,13000  1,07417  0,01944   0,05583      2,22 R
 23     0,80000  0,85167  0,01944  -0,05167     -2,06 R
 24     0,75000  0,80167  0,01944  -0,05167     -2,06 R

R denotes an observation with a large standardized residual.


Grouping Information Using Tukey Method and 95,0% Confidence

Cases    N    Mean  Grouping
Case 19  6  2,1783  A
Case 15  6  1,7567    B
Case 14  6  1,5183      C
Case 13  6  0,9400        D

Means that do not share a letter are significantly different.


Tukey Simultaneous Tests
Response Variable Throughput Unit Time
All Pairwise Comparisons among Levels of Cases
Cases = Case 13  subtracted from:

         Difference       SE of           Adjusted
Cases      of Means  Difference  T-Value   P-Value
Case 14      0,5783     0,01833    31,55    0,0000
Case 15      0,8167     0,01833    44,56    0,0000
Case 19      1,2383     0,01833    67,56    0,0000


Cases = Case 14  subtracted from:

         Difference       SE of           Adjusted
Cases      of Means  Difference  T-Value   P-Value
Case 15      0,2383     0,01833    13,00    0,0000
Case 19      0,6600     0,01833    36,01    0,0000


Cases = Case 15  subtracted from:

         Difference       SE of           Adjusted
```

```
Cases       of Means  Difference  T-Value   P-Value
Case 19       0,4217     0,01833    23,01    0,0000
```

The research reports from ANOVA , the F ratio, with both DFs, and the p-value. The research also reports the results from Tukey. There was significant differences between cases (2 way ANOVA gives $F_{3, 15} = 1590.18$, $p<0.001$). Tukey's HSD test confirmed all cases were significantly different from one another.

Interpreting the results

In the ANOVA table, the p-value <0.001 and Tukey's HSD test confirmed all cases were significantly different from one another so the research rejects the null hypothesis that "There is no difference in transaction throughput between groups, i.e. between (a) relaxing all ACID properties and application-specific criteria and (b) relaxing all ACID properties and no application-specific criteria and (c) relaxing Atomicity, Consistency, Isolation and application-specific criteria and (d) relaxing Atomicity, Consistency, Isolation and no application-specific criteria."

Experiment 12 Raw data and Tukey analysis output

Table C 10 Case 12 and Case 20 Raw Data

No. of transactions in a set	**Case 12**	**Case 20**
20	19.51	18.97
100	19.05	18.46
200	18.77	17.60
300	18.03	17.47
400	17.67	17.45
500	17.20	17.20

H0 – There is no difference in transaction throughput between groups, i.e. between (a) relaxing all ACID properties and application-specific criteria and (b) relaxing no ACID properties but relaxing application-specific criteria

H1 – There is a difference in transaction throughput between groups, i.e. between (a) relaxing all ACID properties and application-specific criteria and (b) relaxing no ACID properties but relaxing application-specific criteria

General Linear Model: Throughput Unit Time versus Transactions; Cases

```
Factor        Type   Levels  Values
Transactions  fixed       6  20; 100; 200; 300; 400; 500
Cases         fixed       2  Case 12; Case 20


Analysis of Variance for Throughput Unit Time, using Adjusted SS for Tests

Source        DF  Seq SS  Adj SS  Adj MS      F      P
Transactions   5  5,6692  5,6692  1,1338  17,12  0,004
Cases          1  0,8640  0,8640  0,8640  13,05  0,015
Error          5  0,3311  0,3311  0,0662
Total         11  6,8643


S = 0,257320   R-Sq = 95,18%   R-Sq(adj) = 89,39%


Grouping Information Using Tukey Method and 95,0% Confidence

Cases    N   Mean  Grouping
Case 12  6  18,40  A
Case 20  6  17,86    B

Means that do not share a letter are significantly different.


Tukey Simultaneous Tests
Response Variable Throughput Unit Time
All Pairwise Comparisons among Levels of Cases
Cases = Case 12  subtracted from:

         Difference       SE of           Adjusted
Cases      of Means  Difference  T-Value   P-Value
Case 20     -0,5367      0,1486   -3,612    0,0153
```

The research reports from ANOVA , the F ratio, with both DFs, and the p-value. The research also reports the results from Tukey. There were significant differences between cases (2 way ANOVA gives $F_{1,5} = 13.05$, $p<0.05$) and Tukey's HSD test confirmed both cases were significantly different from one another.

Interpreting the results

In the ANOVA table, the p-value <0.05 and Tukey's HSD test confirmed all cases were significantly different from one another so the research rejects the null hypothesis that "There is no difference in transaction throughput between groups, i.e. between (a) relaxing all ACID properties and application-specific criteria and (b) relaxing no ACID properties but relaxing application-specific criteria."

Experiment 13 Raw data and Tukey analysis output

Table C 4 Case 13, Case 14, Case 13 & Restart, Case 13 & Negotiation, Case 14 & Restart and Case 14 & Negotiation Raw Data

No. of transactions in a set	**Case 13**	**Case 14**	**Case13& Restart**	**Case13& Negotiation**	**Case14& Restart**	**Case14& Negotiation**
20	1.13	1.63	4.10	2.88	5.00	3.96
100	1.06	1.59	4.05	2.69	4.99	3.88
200	0.99	1.55	4.02	2.56	4.92	3.85
300	0.91	1.49	4.00	2.33	4.88	3.79
400	0.80	1.45	3.96	2.28	4.86	3.75
500	0.75	1.40	3.95	2.25	4.80	3.72

H0 – There is no difference in transaction throughput between groups, i.e. between (a) relaxing all ACID properties and application-specific criteria and (b) relaxing all ACID properties and no application-specific criteria and (c) relaxing all ACID properties and application-specific criteria with Restart and (d) relaxing all ACID properties and no application-specific criteria with Restart and (e) relaxing all ACID properties and application-specific criteria with Negotiation and (f) relaxing all ACID properties and no application-specific criteria with Negotiation.

H1– There is a difference between groups, i.e. between (a) relaxing all ACID properties and application-specific criteria and (b) relaxing all ACID properties and no application-specific criteria and (c) relaxing all ACID properties and application-specific criteria with Restart and (d) relaxing all ACID properties and no application-specific criteria with Restart and (e) relaxing all ACID properties and application-specific criteria with Negotiation and (f) relaxing all ACID properties and no application-specific criteria with Negotiation.

General Linear Model: Throughput Unit Time versus Transactions; Cases

```
Factor        Type   Levels  Values
Transactions  fixed       6  20; 100; 200; 300; 400; 500
Cases         fixed       6  Case  13  Restart without Negotiation; Case 13;
                             Case 13 with Negotiation; Case 14; Case 14
                             Restart without Negotiation; Case 14  with
                             Negotiation


Analysis of Variance for Throughput Unit Time, using Adjusted SS for Tests

Source        DF   Seq SS   Adj SS   Adj MS        F      P
Transactions   5   0,4148   0,4148   0,0830    14,54  0,000
Cases          5  72,1509  72,1509  14,4302  2530,23  0,000
Error         25   0,1426   0,1426   0,0057
Total         35  72,7082
```

```
S = 0,0755189   R-Sq = 99,80%   R-Sq(adj) = 99,73%


Unusual Observations for Throughput Unit Time

     Throughput
Obs   Unit Time      Fit   SE Fit  Residual  St Resid
 19     2,88000  2,66444  0,04174   0,21556      3,43 R

R denotes an observation with a large standardized residual.


Grouping Information Using Tukey Method and 95,0% Confidence

Cases                                  N    Mean  Grouping
Case 14  Restart without Negotiation   6  4,9083  A
Case  13  Restart without Negotiation  6  4,0133    B
Case 14  with Negotiation              6  3,8250      C
Case 13 with Negotiation               6  2,4983        D
Case 14                                6  1,5183          E
Case 13                                6  0,9400            F

Means that do not share a letter are significantly different.


Tukey Simultaneous Tests
Response Variable Throughput Unit Time
All Pairwise Comparisons among Levels of Cases
Cases = Case  13  Restart without Negotiation  subtracted from:

                                      Difference       SE of           Adjusted
Cases                                   of Means  Difference  T-Value   P-Value
Case 13                                   -3,073     0,04360   -70,49    0,0000
Case 13 with Negotiation                  -1,515     0,04360   -34,75    0,0000
Case 14                                   -2,495     0,04360   -57,22    0,0000
Case 14  Restart without Negotiation       0,895     0,04360    20,53    0,0000
Case 14  with Negotiation                 -0,188     0,04360    -4,32    0,0027


Cases = Case 13  subtracted from:
```

```
                                     Difference       SE of             Adjusted
Cases                                  of Means  Difference  T-Value    P-Value
Case 13 with Negotiation                 1,5583     0,04360    35,74     0,0000
Case 14                                  0,5783     0,04360    13,26     0,0000
Case 14  Restart without Negotiation     3,9683     0,04360    91,01     0,0000
Case 14  with Negotiation                2,8850     0,04360    66,17     0,0000


Cases = Case 13 with Negotiation  subtracted from:

                                     Difference       SE of             Adjusted
Cases                                  of Means  Difference  T-Value    P-Value
Case 14                                 -0,9800     0,04360   -22,48     0,0000
Case 14  Restart without Negotiation     2,4100     0,04360    55,27     0,0000
Case 14  with Negotiation                1,3267     0,04360    30,43     0,0000


Cases = Case 14  subtracted from:

                                     Difference       SE of             Adjusted
Cases                                  of Means  Difference  T-Value    P-Value
Case 14  Restart without Negotiation      3,390     0,04360    77,75     0,0000
Case 14  with Negotiation                 2,307     0,04360    52,90     0,0000


Cases = Case 14  Restart without Negotiation  subtracted from:

                          Difference       SE of             Adjusted
Cases                       of Means  Difference  T-Value    P-Value
Case 14  with Negotiation     -1,083     0,04360   -24,85     0,0000
```

The research reports from ANOVA , the F ratio, with both DFs, and the p-value. The research also reports the results from Tukey. There were significant differences between cases (2 way ANOVA gives $F_{5,25}$ = 2530.23, p<0.001) and Tukey's HSD test confirmed all cases were significantly different from one another.

Interpreting the results

In the ANOVA table, the p-value <0.001 and Tukey's HSD test confirmed all cases were significantly different from one another so the research rejects the null hypothesis that "There is no difference in transaction throughput between groups, i.e. between (a) relaxing all ACID properties and application-specific criteria and (b) relaxing all ACID properties and no application-specific criteria and (c) relaxing all ACID properties and application-specific criteria with Restart and (d) relaxing all ACID properties and no application-specific criteria with Restart and (e) relaxing all ACID properties and application-specific criteria with Negotiation and (f) relaxing all ACID properties and no application-specific criteria with Negotiation."

Experiment 13 and Experiment 14 Raw Data and Tukey Analysis Output

Table C 5 Case 4 & Criteria + Negotiation + Tentative Hold, Case 4 & Criteria + Tentative Hold, Case 4 & Criteria, Case 5 & Criteria + Negotiation + Tentative Hold, Case 5 & Criteria + Tentative Hold, Case 5 & Criteria, Case 10 & Criteria + Negotiation + Tentative Hold, Case 10 & Criteria + Tentative Hold, Case 10 & Criteria

No. of transactions in a set	**Case 4 & Criteria+ Negotiation+ Tentative Hold**	**Case 4 & Criteria+ Tentative Hold**	**Case 4 & Criteria**	**Case 5 & Criteria+ Tentative Hold**	**Case 5 & Criteria+ Negotiation+ Tentative Hold**	**Case 5 & Criteria**	**Case 10 & Criteria+ Negotiation+ Tentative Hold**	**Case 10 & Criteria Tentative Hold**	**Case 10 & Criteria**
20	24.98	21.00	15.40	9.20	12.50	4.01	19.50	17.04	12.02
100	24.90	20.40	15.30	9.05	12.42	3.77	19.42	16.95	11.87
200	24.85	19.88	14.95	8.00	12.32	3.64	19.36	16.88	11.49
300	24.77	19.68	14.20	7.92	12.27	3.51	19.30	16.88	11.35
400	24.72	19.54	13.90	7.85	12.20	2.92	19.24	16.72	10.98
500	24.65	19.46	13.38	7.77	12.14	2.58	19.19	16.64	10.08

H0 – There is no difference in transaction throughput between groups, i.e. between (a) relaxing Consistency and application-specific criteria with Negotiation and Tentative Hold and (b) relaxing Consistency and application-specific criteria with Tentative Hold and without Negotiation and (c) relaxing Consistency and application-specific criteria with neither Negotiation nor Tentative Hold (d) relaxing Isolation and application-specific criteria with Negotiation and Tentative Hold and (e) relaxing Isolation and application-specific criteria with Tentative Hold and without Negotiation and (f) relaxing Isolation and application-specific criteria with neither Negotiation nor Tentative Hold and (g) relaxing Durability and application-specific criteria with Negotiation and Tentative Hold, and (h) relaxing Durability and application-specific criteria with Tentative Hold and without Negotiation and (i) relaxing Durability and application-specific criteria with neither Negotiation nor Tentative Hold.

H1 – There is a difference in transaction throughput between groups, i.e. between (a) relaxing Consistency and application-specific criteria with Negotiation and Tentative Hold and (b) relaxing Consistency and application-specific criteria with Tentative Hold and without Negotiation and (c) relaxing Consistency and application-specific criteria with neither Negotiation nor Tentative Hold (d) relaxing Isolation and application-specific criteria with Negotiation and Tentative Hold and (e) relaxing Isolation and application-specific criteria with Tentative Hold and without Negotiation and (f) relaxing Isolation and application-specific criteria with neither Negotiation nor Tentative Hold and (g) relaxing Durability and application-specific criteria with Negotiation and Tentative Hold, and (h) relaxing Durability and application-specific criteria with Tentative Hold and without Negotiation and (i) relaxing Durability and application-specific criteria with neither Negotiation nor Tentative Hold.

General Linear Model: Throughput Unit Time versus Transactions; Cases

```
Factor        Type   Levels  Values
Transactions  fixed       6  20; 100; 200; 300; 400; 500
Cases         fixed       9  Case 10& Criteria; Case 10& Criteria + Tentative
                             hold; Case 10& Criteria +Negotiation + Tentative
                             hold; Case 4& Criteria; Case 4& Criteria
                             +Negotiation + Tentative  hold; Case 4& Criteria
                             Tentative  hold; Case 5& Criteria; Case 5&
                             Criteria  + Tentative  hold; Case 5& Criteria
                             +Negotiation + Tentative  hold
```

```
Analysis of Variance for Throughput Unit Time, using Adjusted SS for Tests

Source        DF   Seq SS   Adj SS  Adj MS        F      P
Transactions   5     7,45     7,45    1,49    14,63  0,000
Cases          8  2052,10  2052,10  256,51  2519,41  0,000
Error         40     4,07     4,07    0,10
Total         53  2063,62


S = 0,319084   R-Sq = 99,80%   R-Sq(adj) = 99,74%


Unusual Observations for Throughput Unit Time

     Throughput
Obs   Unit Time      Fit  SE Fit  Residual  St Resid
 30     13,3800  13,9735  0,1625   -0,5935     -2,16 R
 42     10,0800  10,7502  0,1625   -0,6702     -2,44 R

R denotes an observation with a large standardized residual.


Grouping Information Using Tukey Method and 95,0% Confidence

Cases                                             N    Mean  Grouping
Case 4& Criteria +Negotiation + Tentative  hold   6  24,812  A
Case 4& Criteria Tentative  hold                  6  19,993    B
Case 10& Criteria +Negotiation + Tentative  hold  6  19,335      C
Case 10& Criteria + Tentative  hold               6  16,852        D
Case 4& Criteria                                  6  14,522          E
Case 5& Criteria +Negotiation + Tentative  hold   6  12,308            F
Case 10& Criteria                                 6  11,298              G
Case 5& Criteria  + Tentative  hold               6   8,298                H
Case 5& Criteria                                  6   3,405                  I

Means that do not share a letter are significantly different.


Tukey Simultaneous Tests
Response Variable Throughput Unit Time
```

```
All Pairwise Comparisons among Levels of Cases
Cases = Case 10& Criteria  subtracted from:

                                                  Difference       SE of
Cases                                               of Means  Difference
Case 10& Criteria + Tentative  hold                    5,553      0,1842
Case 10& Criteria +Negotiation + Tentative  hold       8,037      0,1842
Case 4& Criteria                                       3,223      0,1842
Case 4& Criteria +Negotiation + Tentative  hold       13,513      0,1842
Case 4& Criteria Tentative  hold                       8,695      0,1842
Case 5& Criteria                                      -7,893      0,1842
Case 5& Criteria  + Tentative  hold                   -3,000      0,1842
Case 5& Criteria +Negotiation + Tentative  hold        1,010      0,1842

                                                           Adjusted
Cases                                             T-Value   P-Value
Case 10& Criteria + Tentative  hold                 30,14    0,0000
Case 10& Criteria +Negotiation + Tentative  hold    43,62    0,0000
Case 4& Criteria                                    17,50    0,0000
Case 4& Criteria +Negotiation + Tentative  hold     73,35    0,0000
Case 4& Criteria Tentative  hold                    47,20    0,0000
Case 5& Criteria                                   -42,85    0,0000
Case 5& Criteria  + Tentative  hold                -16,28    0,0000
Case 5& Criteria +Negotiation + Tentative  hold      5,48    0,0001


Cases = Case 10& Criteria + Tentative  hold  subtracted from:

                                                  Difference       SE of
Cases                                               of Means  Difference
Case 10& Criteria +Negotiation + Tentative  hold        2,48      0,1842
Case 4& Criteria                                       -2,33      0,1842
Case 4& Criteria +Negotiation + Tentative  hold         7,96      0,1842
Case 4& Criteria Tentative  hold                        3,14      0,1842
Case 5& Criteria                                      -13,45      0,1842
Case 5& Criteria  + Tentative  hold                    -8,55      0,1842
Case 5& Criteria +Negotiation + Tentative  hold        -4,54      0,1842

                                                           Adjusted
Cases                                             T-Value   P-Value
Case 10& Criteria +Negotiation + Tentative  hold    13,48    0,0000
Case 4& Criteria                                   -12,65    0,0000
Case 4& Criteria +Negotiation + Tentative  hold     43,21    0,0000
Case 4& Criteria Tentative  hold                    17,05    0,0000
Case 5& Criteria                                   -72,99    0,0000
Case 5& Criteria  + Tentative  hold                -46,43    0,0000
Case 5& Criteria +Negotiation + Tentative  hold    -24,66    0,0000


Cases = Case 10& Criteria +Negotiation + Tentative  hold  subtracted from:

                                                 Difference       SE of
Cases                                              of Means  Difference
Case 4& Criteria                                      -4,81      0,1842
Case 4& Criteria +Negotiation + Tentative  hold        5,48      0,1842
Case 4& Criteria Tentative  hold                       0,66      0,1842
Case 5& Criteria                                     -15,93      0,1842
Case 5& Criteria  + Tentative  hold                  -11,04      0,1842
Case 5& Criteria +Negotiation + Tentative  hold       -7,03      0,1842

                                                          Adjusted
Cases                                            T-Value   P-Value
Case 4& Criteria                                  -26,13    0,0000
Case 4& Criteria +Negotiation + Tentative  hold    29,73    0,0000
Case 4& Criteria Tentative  hold                    3,57    0,0236
Case 5& Criteria                                  -86,47    0,0000
Case 5& Criteria  + Tentative  hold               -59,91    0,0000
Case 5& Criteria +Negotiation + Tentative  hold   -38,14    0,0000


Cases = Case 4& Criteria  subtracted from:

                                                 Difference       SE of
Cases                                              of Means  Difference
Case 4& Criteria +Negotiation + Tentative  hold       10,29      0,1842
Case 4& Criteria Tentative  hold                       5,47      0,1842
Case 5& Criteria                                     -11,12      0,1842
Case 5& Criteria  + Tentative  hold                   -6,22      0,1842
Case 5& Criteria +Negotiation + Tentative  hold       -2,21      0,1842

                                                          Adjusted
Cases                                            T-Value   P-Value
```

```
Case 4& Criteria +Negotiation + Tentative  hold    55,86    0,0000
Case 4& Criteria Tentative  hold                   29,70    0,0000
Case 5& Criteria                                  -60,34    0,0000
Case 5& Criteria  + Tentative  hold               -33,78    0,0000
Case 5& Criteria +Negotiation + Tentative  hold   -12,01    0,0000


Cases = Case 4& Criteria +Negotiation + Tentative  hold  subtracted from:

                                                 Difference       SE of
Cases                                              of Means  Difference
Case 4& Criteria Tentative  hold                      -4,82      0,1842
Case 5& Criteria                                     -21,41      0,1842
Case 5& Criteria  + Tentative  hold                  -16,51      0,1842
Case 5& Criteria +Negotiation + Tentative  hold      -12,50      0,1842

                                                          Adjusted
Cases                                            T-Value   P-Value
Case 4& Criteria Tentative  hold                   -26,2    0,0000
Case 5& Criteria                                  -116,2    0,0000
Case 5& Criteria  + Tentative  hold                -89,6    0,0000
Case 5& Criteria +Negotiation + Tentative  hold    -67,9    0,0000


Cases = Case 4& Criteria Tentative  hold  subtracted from:

                                                 Difference       SE of
Cases                                              of Means  Difference
Case 5& Criteria                                     -16,59      0,1842
Case 5& Criteria  + Tentative  hold                  -11,69      0,1842
Case 5& Criteria +Negotiation + Tentative  hold       -7,68      0,1842

                                                          Adjusted
Cases                                            T-Value   P-Value
Case 5& Criteria                                  -90,04    0,0000
Case 5& Criteria  + Tentative  hold               -63,48    0,0000
Case 5& Criteria +Negotiation + Tentative  hold   -41,72    0,0000


Cases = Case 5& Criteria  subtracted from:
```

```
                                                 Difference       SE of
Cases                                              of Means  Difference
Case 5& Criteria  + Tentative  hold                   4,893      0,1842
Case 5& Criteria +Negotiation + Tentative  hold       8,903      0,1842

                                                          Adjusted
Cases                                            T-Value   P-Value
Case 5& Criteria  + Tentative  hold                26,56    0,0000
Case 5& Criteria +Negotiation + Tentative  hold    48,33    0,0000


Cases = Case 5& Criteria  + Tentative  hold  subtracted from:

                                                 Difference       SE of
Cases                                              of Means  Difference
Case 5& Criteria +Negotiation + Tentative  hold       4,010      0,1842

                                                          Adjusted
Cases                                            T-Value   P-Value
Case 5& Criteria +Negotiation + Tentative  hold    21,77    0,0000
```

The research reports from ANOVA , the F ratio, with both DFs, and the p-value. The research also reports the results from Tukey. There was significant differences between cases (2 way ANOVA gives $F_{8,\ 40}$ = 2519.41, p<0.001) and Tukey’s HSD test confirmed all cases were significantly different from one another.

Interpreting the results

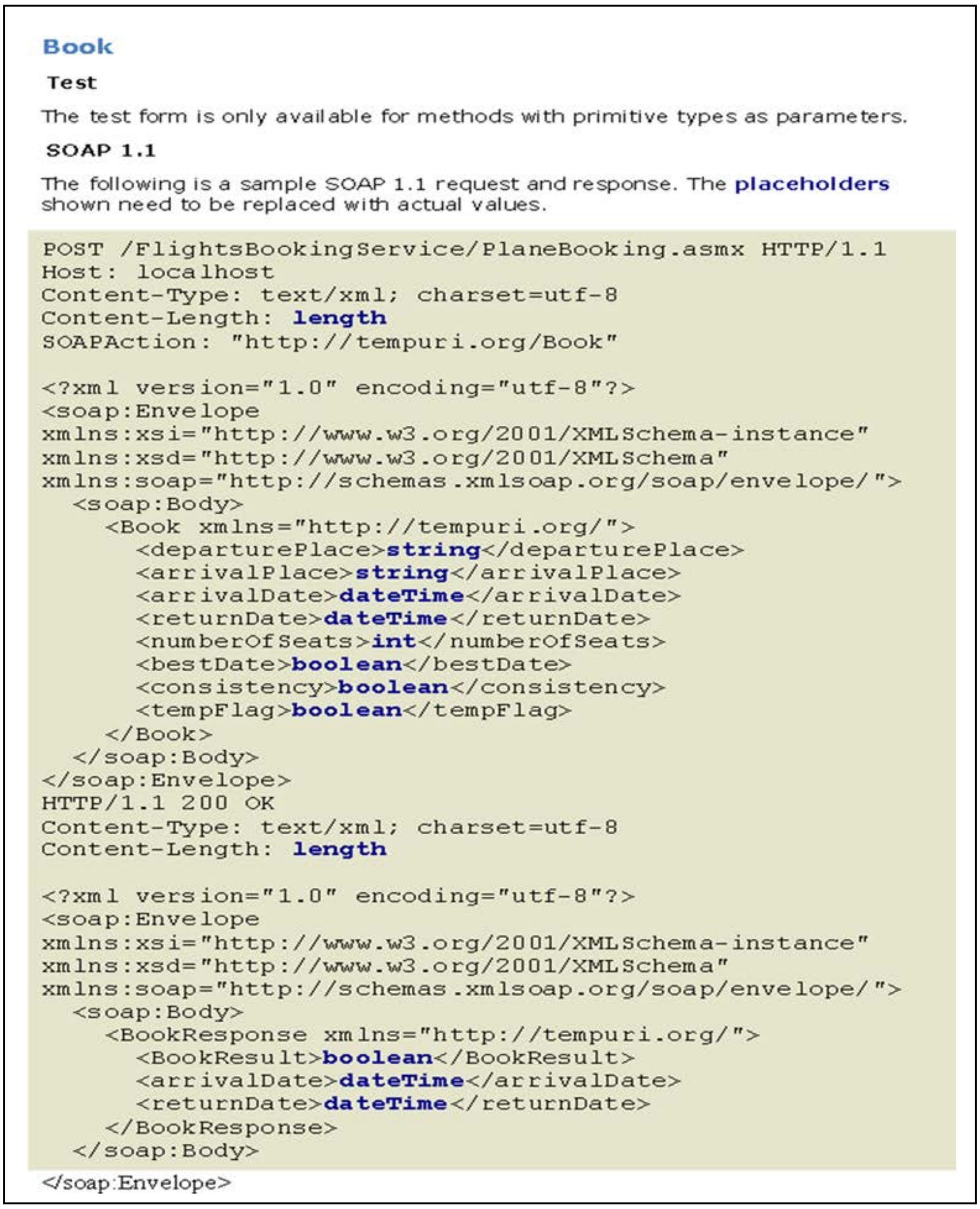
In the ANOVA table, the p-value <0.001 and Tukey’s HSD test confirmed all cases were significantly different from one another so the research rejects the null hypothesis that “between (a) relaxing Consistency and application-specific criteria with Negotiation and Tentative Hold and (b) relaxing Consistency and application-specific criteria with Tentative Hold and without Negotiation and (c) relaxing Consistency and application-specific criteria with neither Negotiation nor Tentative Hold (d) relaxing Isolation and application-specific criteria with Negotiation and Tentative Hold and (e) relaxing Isolation and application-specific criteria with Tentative Hold and without Negotiation and (f) relaxing Isolation and application-specific criteria with neither Negotiation nor Tentative Hold and (g) relaxing Durability and application-specific criteria with Negotiation and Tentative Hold, and (h) relaxing Durability and application-specific criteria with Tentative Hold and without Negotiation and (i) relaxing Durability and application-specific criteria with neither Negotiation nor Tentative Hold."

Summary

In all experiments using the ANOVA test, there is high significant difference between the groups in the experiment and furthermore Tukey revealed the significance between each case in the group in the experiments. Thus it is clear that relaxing ACID properties makes a significant difference in throughput compared to not relaxing ACID properties. Relaxing additional application-specific criteria also makes a significant difference compared to not relaxing these criteria. Use of Tentative Hold decreases throughput significantly. Use of Negotiation, when compared to abort and restart, increases throughput significantly.

Appendix D - AuTrA technologies examples

This appendix shows examples of the SOAP and WSDL technologies used in the AuTrA system.



Book

Test

The test form is only available for methods with primitive types as parameters.

SOAP 1.1

The following is a sample SOAP 1.1 request and response. The **placeholders** shown need to be replaced with actual values.

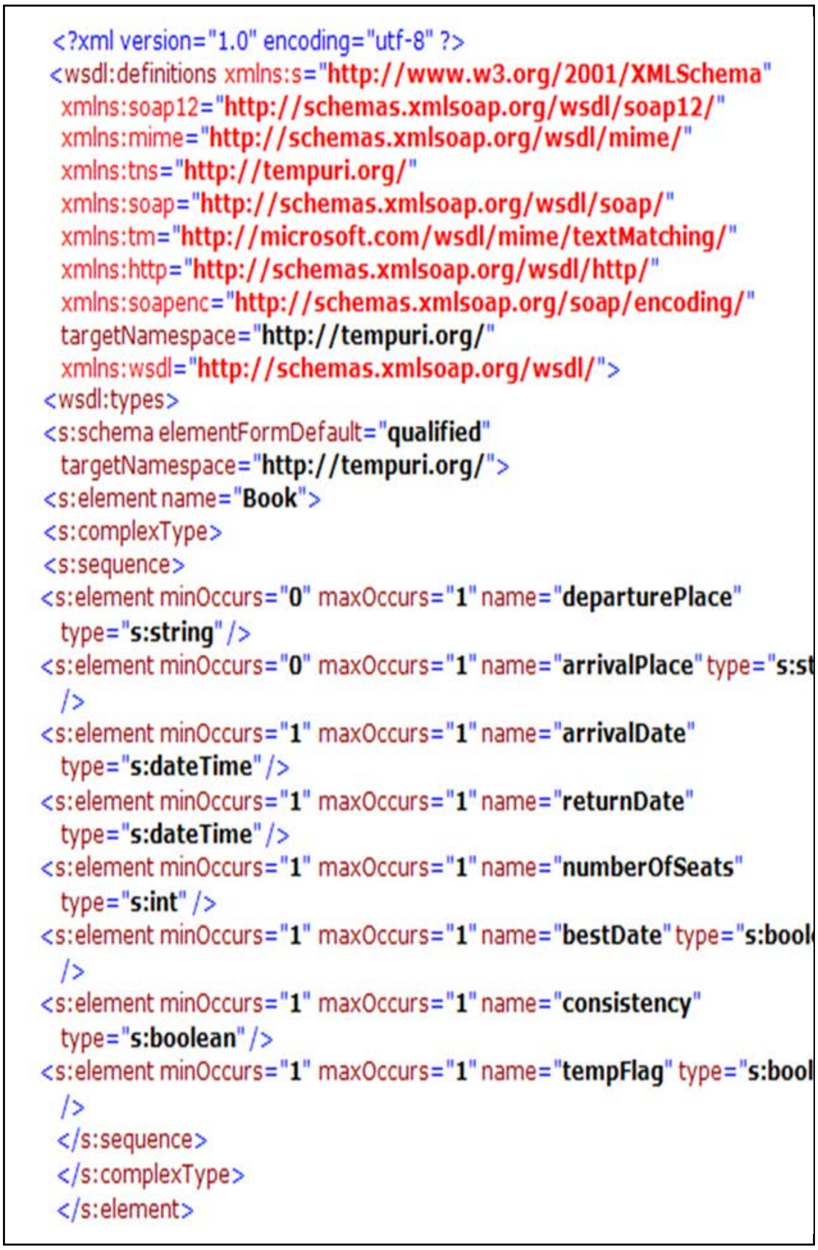
```
POST /FlightsBookingService/PlaneBooking.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/Book"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Book xmlns="http://tempuri.org/">
      <departurePlace>string</departurePlace>
      <arrivalPlace>string</arrivalPlace>
      <arrivalDate>dateTime</arrivalDate>
      <returnDate>dateTime</returnDate>
      <numberOfSeats>int</numberOfSeats>
      <bestDate>boolean</bestDate>
      <consistency>boolean</consistency>
      <tempFlag>boolean</tempFlag>
    </Book>
  </soap:Body>
</soap:Envelope>
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <BookResponse xmlns="http://tempuri.org/">
      <BookResult>boolean</BookResult>
      <arrivalDate>dateTime</arrivalDate>
      <returnDate>dateTime</returnDate>
    </BookResponse>
  </soap:Body>
</soap:Envelope>
```

Figure D1 Flight booking service: SOAP message

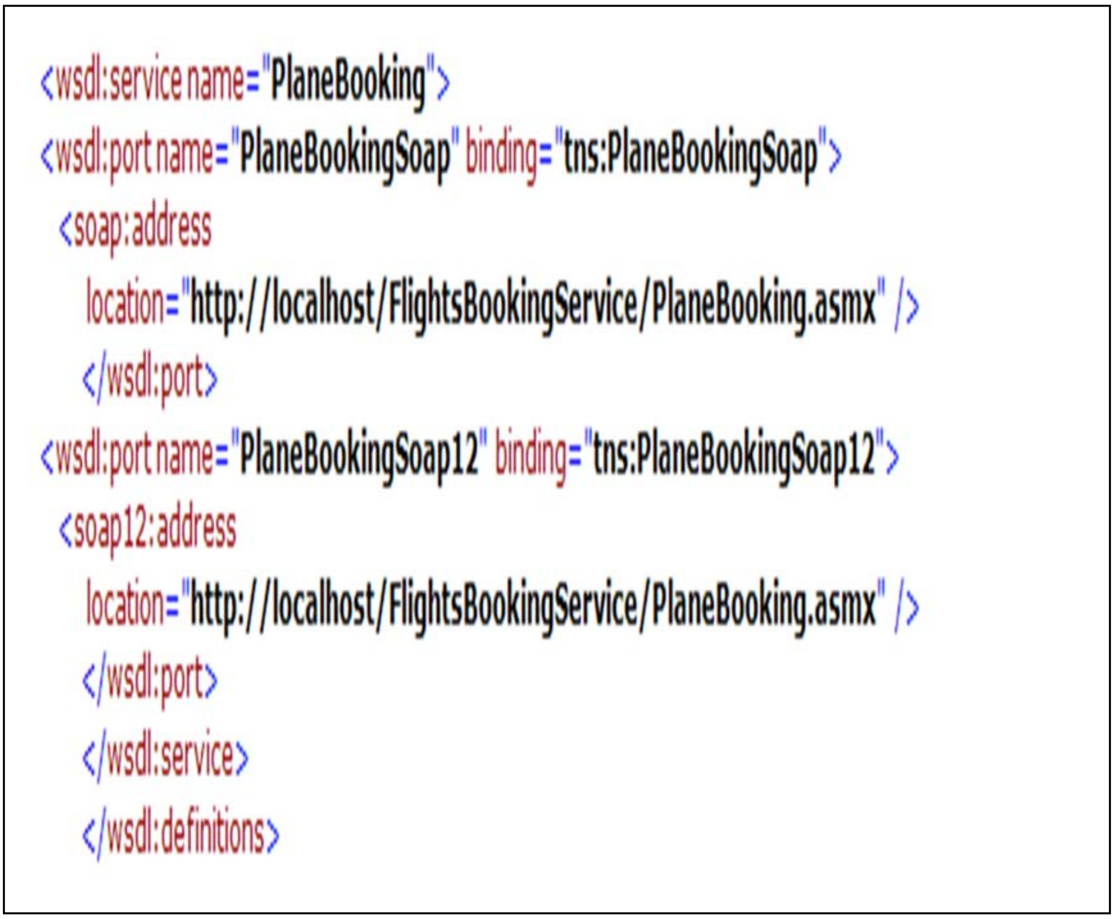
With the help of SOAP, the users can post the request and transport it to the correct end point. (Figure D1)



```
<?xml version="1.0" encoding="utf-8" ?>
<wsdl:definitions xmlns:s="http://www.w3.org/2001/XMLSchema"
  xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
  xmlns:tns="http://tempuri.org/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  targetNamespace="http://tempuri.org/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
<wsdl:types>
<s:schema elementFormDefault="qualified"
  targetNamespace="http://tempuri.org/">
<s:element name="Book">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="departurePlace"
  type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="arrivalPlace" type="s:st
  />
<s:element minOccurs="1" maxOccurs="1" name="arrivalDate"
  type="s:dateTime" />
<s:element minOccurs="1" maxOccurs="1" name="returnDate"
  type="s:dateTime" />
<s:element minOccurs="1" maxOccurs="1" name="numberOfSeats"
  type="s:int" />
<s:element minOccurs="1" maxOccurs="1" name="bestDate" type="s:bool
  />
<s:element minOccurs="1" maxOccurs="1" name="consistency"
  type="s:boolean" />
<s:element minOccurs="1" maxOccurs="1" name="tempFlag" type="s:bool
  />
 </s:sequence>
 </s:complexType>
 </s:element>
```

Figure D2 Flight booking service: WSDL data type definition

Figure D2 shows the WSDL example of a flight booking service which has been registered in AuTrA.



```
<wsdl:service name="PlaneBooking">
<wsdl:port name="PlaneBookingSoap" binding="tns:PlaneBookingSoap">
  <soap:address
    location="http://localhost/FlightsBookingService/PlaneBooking.asmx" />
    </wsdl:port>
<wsdl:port name="PlaneBookingSoap12" binding="tns:PlaneBookingSoap12">
  <soap12:address
    location="http://localhost/FlightsBookingService/PlaneBooking.asmx" />
    </wsdl:port>
    </wsdl:service>
    </wsdl:definitions>
```

Figure D3 Flight booking service: WSDL concrete segment

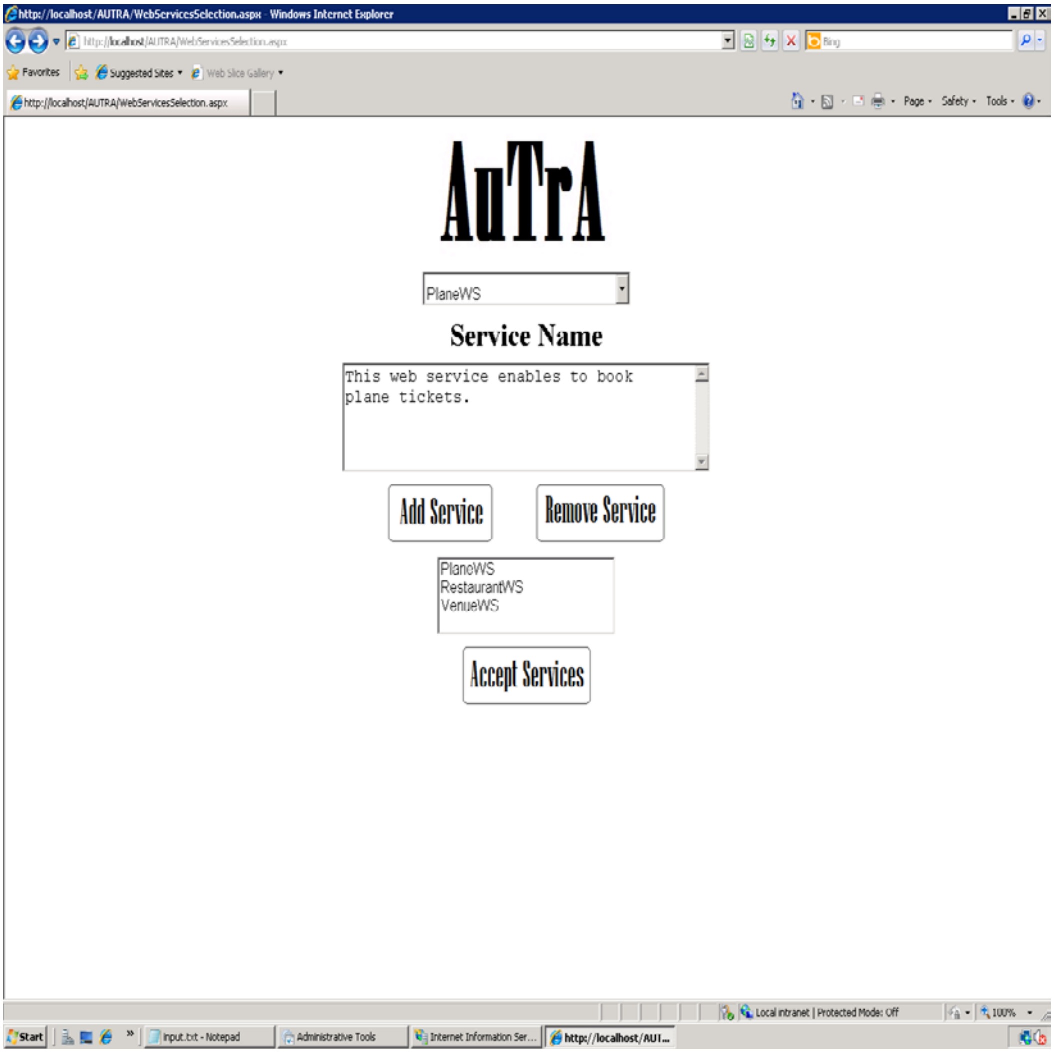
The binding of the abstract to the concrete segment is through the port address of the PlaneBookingSoap port as shown in Figure D3.

Appendix E – Snippets of some of the inputs used in the experiments

This appendix shows some snippets of the inputs used in the experiments. In section E1 the user interface for transaction composition is shown and in section E2 example snippets of the batch files used as inputs to the experiments are shown.

E1 User interface for transaction composition

When the user wants to book flight, hotel and ski in one composite transaction, he or she has to choose which services he wants. For example, Figure E1 shows that the user wants to book flight, restaurants and venue in one transaction, meaning the composite service of the user's transaction will consist of three component services.



PlaneWS

This web service enables to book plane tickets.

PlaneWS
RestaurantWS
VenueWS

Figure E1 Composite service realisation

After the user accepts the services from which to create a composition service, the user has to input the required information, like date, number of travellers and number of rooms required. Different Web services will require different information.

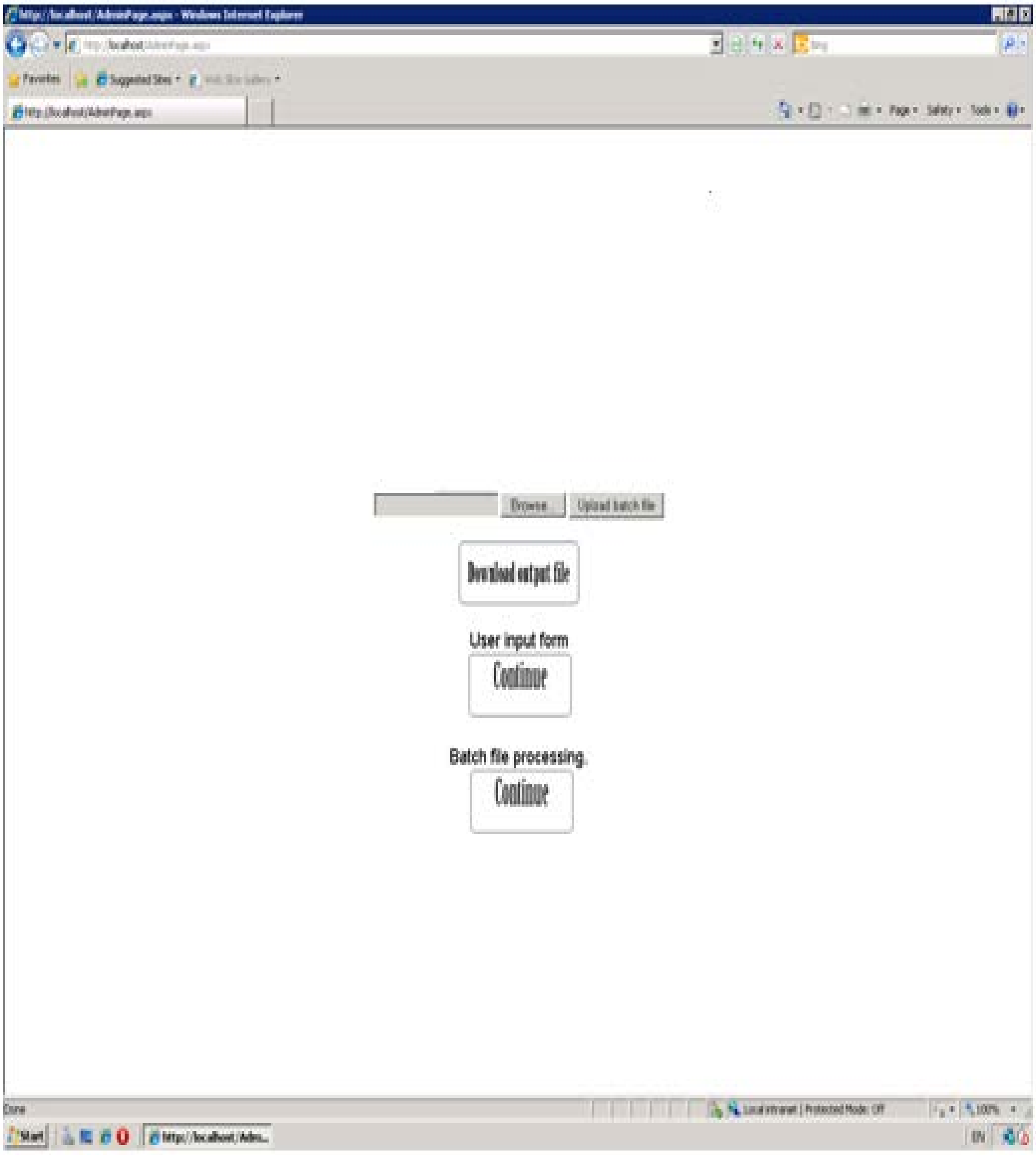
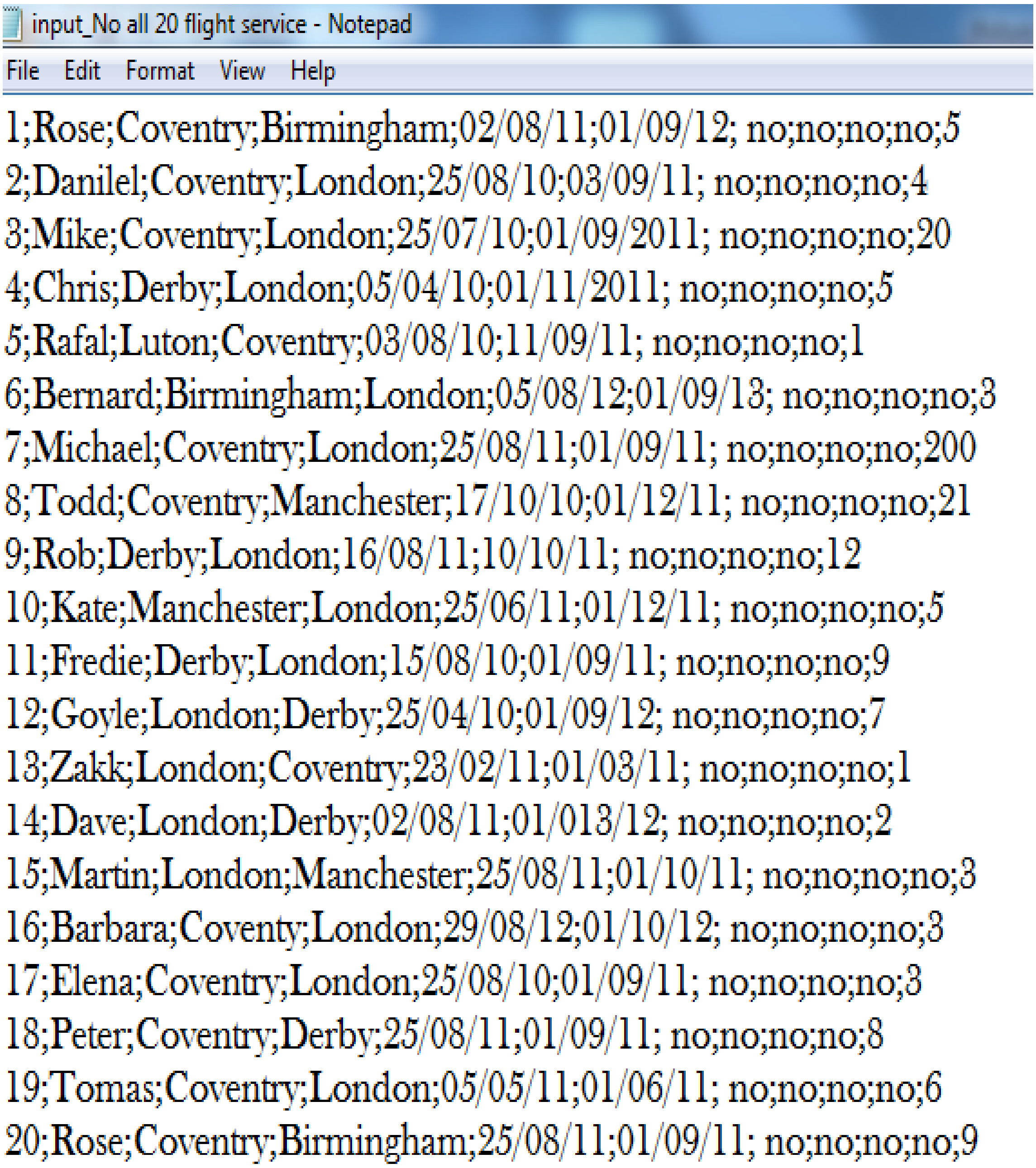


Figure E2 Input upload

Input batch files can be uploaded using the upload button shown in Figure E2. The continue button will run the composite service. The system then breaks down the input to the composite service into inputs to component services for processing.

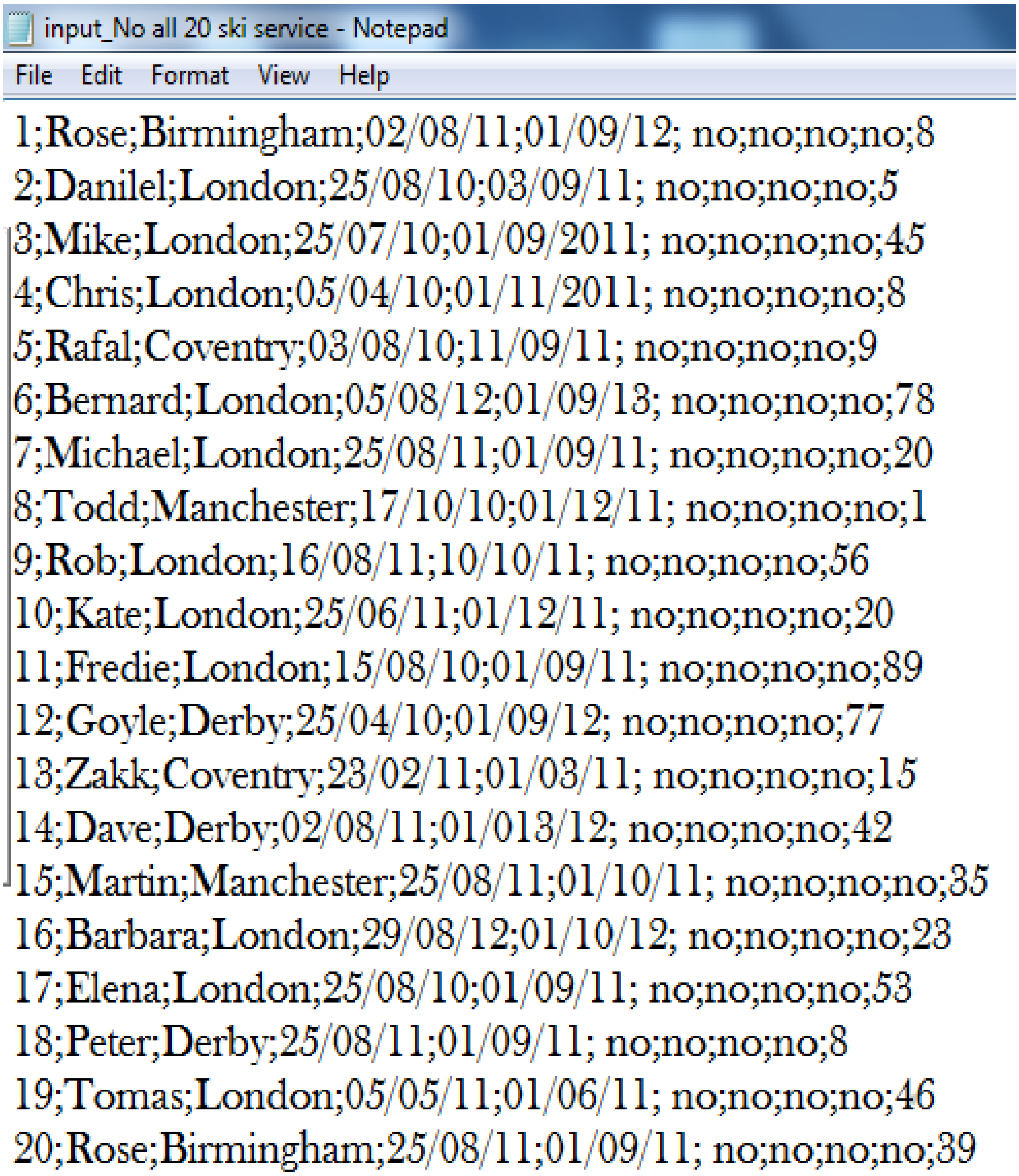
E3 Example Batch File Input

Figure E3 shows example batch file input where none of the ACID properties are relaxed in a composite transaction made up of flight service, hotel service and ski service.


input_No all 20 flight service - Notepad
File Edit Format View Help

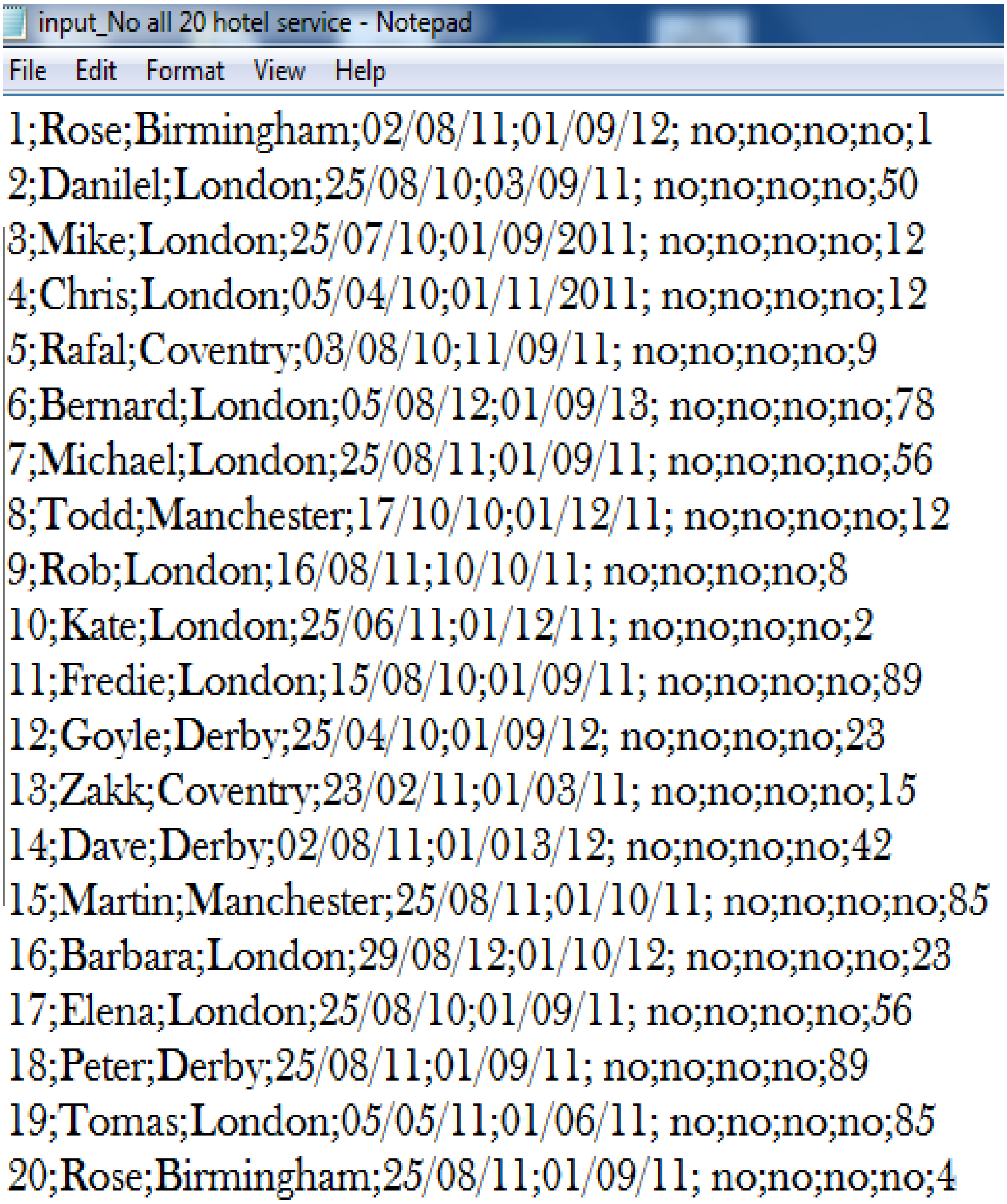
```
1;Rose;Coventry;Birmingham;02/08/11;01/09/12; no;no;no;no;5
2;Danilel;Coventry;London;25/08/10;03/09/11; no;no;no;no;4
3;Mike;Coventry;London;25/07/10;01/09/2011; no;no;no;no;20
4;Chris;Derby;London;05/04/10;01/11/2011; no;no;no;no;5
5;Rafal;Luton;Coventry;03/08/10;11/09/11; no;no;no;no;1
6;Bernard;Birmingham;London;05/08/12;01/09/13; no;no;no;no;3
7;Michael;Coventry;London;25/08/11;01/09/11; no;no;no;no;200
8;Todd;Coventry;Manchester;17/10/10;01/12/11; no;no;no;no;21
9;Rob;Derby;London;16/08/11;10/10/11; no;no;no;no;12
10;Kate;Manchester;London;25/06/11;01/12/11; no;no;no;no;5
11;Fredie;Derby;London;15/08/10;01/09/11; no;no;no;no;9
12;Goyle;London;Derby;25/04/10;01/09/12; no;no;no;no;7
13;Zakk;London;Coventry;23/02/11;01/03/11; no;no;no;no;1
14;Dave;London;Derby;02/08/11;01/013/12; no;no;no;no;2
15;Martin;London;Manchester;25/08/11;01/10/11; no;no;no;no;3
16;Barbara;Coventy;London;29/08/12;01/10/12; no;no;no;no;3
17;Elena;Coventry;London;25/08/10;01/09/11; no;no;no;no;3
18;Peter;Coventry;Derby;25/08/11;01/09/11; no;no;no;no;8
19;Tomas;Coventry;London;05/05/11;01/06/11; no;no;no;no;6
20;Rose;Coventry;Birmingham;25/08/11;01/09/11; no;no;no;no;9
```

Figure E3 Flight input



```
1;Rose;Birmingham;02/08/11;01/09/12; no;no;no;no;8
2;Danilel;London;25/08/10;03/09/11; no;no;no;no;5
3;Mike;London;25/07/10;01/09/2011; no;no;no;no;45
4;Chris;London;05/04/10;01/11/2011; no;no;no;no;8
5;Rafal;Coventry;03/08/10;11/09/11; no;no;no;no;9
6;Bernard;London;05/08/12;01/09/13; no;no;no;no;78
7;Michael;London;25/08/11;01/09/11; no;no;no;no;20
8;Todd;Manchester;17/10/10;01/12/11; no;no;no;no;1
9;Rob;London;16/08/11;10/10/11; no;no;no;no;56
10;Kate;London;25/06/11;01/12/11; no;no;no;no;20
11;Fredie;London;15/08/10;01/09/11; no;no;no;no;89
12;Goyle;Derby;25/04/10;01/09/12; no;no;no;no;77
13;Zakk;Coventry;23/02/11;01/03/11; no;no;no;no;15
14;Dave;Derby;02/08/11;01/013/12; no;no;no;no;42
15;Martin;Manchester;25/08/11;01/10/11; no;no;no;no;35
16;Barbara;London;29/08/12;01/10/12; no;no;no;no;23
17;Elena;London;25/08/10;01/09/11; no;no;no;no;53
18;Peter;Derby;25/08/11;01/09/11; no;no;no;no;8
19;Tomas;London;05/05/11;01/06/11; no;no;no;no;46
20;Rose;Birmingham;25/08/11;01/09/11; no;no;no;no;39
```

Figure E4 Ski input


input_No all 20 hotel service - Notepad
File Edit Format View Help

```
1;Rose;Birmingham;02/08/11;01/09/12; no;no;no;no;1
2;Danilel;London;25/08/10;03/09/11; no;no;no;no;50
3;Mike;London;25/07/10;01/09/2011; no;no;no;no;12
4;Chris;London;05/04/10;01/11/2011; no;no;no;no;12
5;Rafal;Coventry;03/08/10;11/09/11; no;no;no;no;9
6;Bernard;London;05/08/12;01/09/13; no;no;no;no;78
7;Michael;London;25/08/11;01/09/11; no;no;no;no;56
8;Todd;Manchester;17/10/10;01/12/11; no;no;no;no;12
9;Rob;London;16/08/11;10/10/11; no;no;no;no;8
10;Kate;London;25/06/11;01/12/11; no;no;no;no;2
11;Fredie;London;15/08/10;01/09/11; no;no;no;no;89
12;Goyle;Derby;25/04/10;01/09/12; no;no;no;no;23
13;Zakk;Coventry;23/02/11;01/03/11; no;no;no;no;15
14;Dave;Derby;02/08/11;01/013/12; no;no;no;no;42
15;Martin;Manchester;25/08/11;01/10/11; no;no;no;no;85
16;Barbara;London;29/08/12;01/10/12; no;no;no;no;23
17;Elena;London;25/08/10;01/09/11; no;no;no;no;56
18;Peter;Derby;25/08/11;01/09/11; no;no;no;no;89
19;Tomas;London;05/05/11;01/06/11; no;no;no;no;85
20;Rose;Birmingham;25/08/11;01/09/11; no;no;no;no;4
```

Figure E5 Hotel input

The snippets shown in Figures E3, E4 and E5 were part of the input used in Experiment 1 Case 1 of the Travel Plan application.

Appendix F – Acronyms

2PC	2-Phase Commit
ACI	Atomicity, Consistency, Isolation
ACID	Atomicity, Consistency, Isolation, Durability
ACTA	A Comprehensive TransAction Framework for Extended Transactions
ANOVA	Analysis of Variance
API	Application Programming Interface
API(JTA)	Application Programming Interface (Java Transaction API)
ASE	Adaptive Server Enterprise
ASSET	A System Supporting Extended Transactions
ATS	Accepted Termination States
AuTrA	Adaptable user-defined Transaction relaxing Approach
BTP	Business Transaction Protocol
CAEs	Collaboration Editing Applications
CAGIS-Trans	Cooperative Agents in a Global Information Space-Transactions
COO	COOperating software developers COOrdination
CSCW	Computer Supported Cooperative Work
DF	Degrees of Freedom
FIFO	First in First Out

EPOS	Expert System for Program and ~og~ System Development
HoL	Head of Line
HSD	Honest Significant Difference
IIS	Internet Information Server
IMSD	Information Management System Dynamics
JTA	Java Transaction API
LLR	Low-Latency Resilient
NBCP	Non-Blocking Commit Protocol
NTO	New Timestamp Ordering
RTF	Reflective Transaction Framework
SACReD	Semantic Atomicity, Consistency, Resiliency, Durability
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
taTHP	transaction-aware Tentative Hold Protocol
TCS	Transaction Composite Services
TCP4CWS	Transaction Commit Protocol for Composite Web Services
THP	Tentative Hold Protocol
TODS	Tsinghua Object Data Store
UDDI	Universal Description, Discovery, and Integration
URL	Uniform Resource Locator

WS-AT	WS-AtomicTransaction
WS-BA	WS-BusinessActivity
WSDL	Web Services Description Language
WS-Tx	Web Services Transactions
WTDP	Web Services Transaction Dependency management Protocol
WSTDCP	Web Services Transaction Dependency Coordination Protocol
XML	Extensible Markup Language