A Comparative Evaluation of Specification Techniques for Electronic Commerce Systems

by

Bamidele Ola

A thesis
presented to the University of Manitoba
in partial fulfilment of the
requirements for the degree of
Master of Science
in
Computer Science

Winnipeg, Manitoba, Canada, 2004

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION

A COMPARATIVE EVALUATION OF SPECIFICATION TECHNIQUES FOR ELECTRONIC COMMERCE SYSTEMS

BY

Bamidele Ola

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirement of the degree

Of

MASTER OF SCIENCE

Abstract

E-commerce has become a very popular form of transacting business. A combination of factors such as high volume of transactions, high revenues, and increasing patronage of e-commerce activities have led to the hasty development and deployment of e-commerce systems by e-commerce service providers. The reliability of e-commerce systems has been queried in some recent scenarios. The approaches used in developing these applications are informal, ad hoc, and intuitive, which result in e-commerce systems that fail to meet both the users' and customers' expectations. The application of rigorous formal specification techniques in developing e-commerce systems would result in reliable applications processing that can guarantee the correctness of e-commerce transactions.

This thesis pragmatically evaluates five major specification techniques (including two variants of one of the techniques) amenable to specifying e-commerce systems. In this research, I identify the requirements necessary for a specification technique to completely specify all the components of an e-commerce system, and classify these requirements. Thereafter, I examine existing specification techniques applicable to specifying e-commerce systems, and provide a taxonomy for these techniques. Furthermore, I use the five major specification techniques for evaluation to specify the ordering process in a typical business-to-consumer (B2C) e-commerce scenario. Lastly, an evaluation of the specification techniques was carried out using the evaluation criteria derived from the identified requirements and a fuzzy logic framework developed in this research.

List of Publications

1. Sylvanus A. Ehikioya and **Bamidele Ola**, An Evaluation of Specification Methods for Electronic Commerce Systems. *2nd International Conference on Computer Science and its Applications, (ICCSA-2004)*, San Diego, California, USA, June 28 - 30, 2004.

2. Sylvanus A. Ehikioya and **Bamidele Ola**, A Comparison of Formalisms for Electronic Commerce Systems. *Alternative Approaches in Software Engineering, A special session at ICCC 2004 - IEEE International Conference on Computational Cybernetics, (AASE-ICCC 2004)*, Vienna, Austria, August 30 - September 1, 2004.

3. Sylvanus A. Ehikioya and **Bamidele Ola**, A Framework for Evaluating Specification Methods for Electronic Commerce Systems. *13th International Conference on Information Systems Development, Advances in Theory, Practice and Education, (ISD' 2004)* Vilnius, Lithuania, September 9 - 11, 2004, Kluwer Academic Publishers.

4. Sylvanus A. Ehikioya and **Bamidele Ola**, A Comparative Study of Specification Methods for Electronic Commerce Systems. *3rd ACS/IEEE International Conference on Computer Systems and Applications (AICCSA-05)* Cairo, Egypt, January 3 - 6, 2005.

Dedication

This thesis is dedicated to God the giver of life and from whom all good things flow and to my mother for all her care, support, and encouragement.

Acknowledgements

First and foremost, I would like to acknowledge my advisor, Dr. S.A. Ehikioya, for accepting me as a Masters student in the Computer Science Department and went beyond the call of duty to facilitate my coming to Canada. In addition, I sincerely, profoundly, and interminably thank my advisor for his most useful comments, suggestions, corrections, and support as this research evolves; and more importantly the timely manner with which he reviewed this work.

I want to thank my examiners, Dr. Vojislav B. Misic of the department of Computer Science and Professor Robert D. McLeod of the department of Electrical and Computer Engineering for accepting to serve in my thesis committee, despite their obviously tight schedules.

I also want to thank my Pastor, Tokunboh Okunnu and his family for their prayers, support, and encouragement as well as my friends: David Allenotor, Lanre Ojo, John Akinyemi, Yinka Komolafe, and a host of others too numerous to mention for all their support and encouragement. May the good Lord grant you all your heart desires.

Lastly, I want to thank my parents and younger siblings for their understanding and patience this period I choose to pursue this personal agenda.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Electronic commerce (e-commerce) entails the buying and selling of goods and services using the Internet. E-commerce has profoundly changed conventional businesses. E-commerce transactions are carried out online over the Internet and has broken barriers associated with traditional businesses. E-commerce has grown rapidly and individuals, private sector organizations, and government agencies currently provide e-commerce services.

E-commerce enables e-commerce merchants (providers of e-commerce services) to make goods and services available to a wider range of customers at a cheaper, timely, effective, and efficient manner. Also, e-commerce affords customers to access a wider range of products. The products could be cheaper and the customer's access to products is faster and easier in comparison with conventional commerce. However, a reliable and dependable e-commerce system is fundamental to maximizing the full potentials of these benefits accruable to customers and e-commerce merchants. Alagar and Li [2] as well as Ehikioya [31] underscores the need to develop secure (providing a framework for guaranteeing security of customers' and merchants' data), reliable (producing exact results in a consistent manner), and error-free e-commerce applications that guarantee the correctness (producing consistent and correct operational and transactional results) of online transactions.

1.1 Problem Statement

Some essential characteristic behaviour of e-commerce systems[1] increase their complexity. Also, these essential characteristic behaviours make e-commerce systems amenable to formal specification. This resultant complexity leads to many situations where errors (unanticipated behaviours in a software application) and faults (manifested errors in a software application) could occur in e-commerce transactions. Some scenarios where errors have surfaced in e-commerce applications include: more than one customer attempting to buy the same item, the system displaying an already sold item as available item, and the system displaying out-of-inventory items. The mitigation of such errors in e-commerce applications is critical to the survival of e-commerce. These errors are mostly hidden and are unnoticed at development time and are triggered after system deployment under unanticipated scenarios, which may result in subtle mistakes and sometimes system failures [31, 105].

Techniques used in developing most prevailing e-commerce applications do not conform to the generic software development process. Guaranteeing the correctness (entails various stages in e-commerce transactions producing consistent results) of an e-commerce application is difficult. This difficulty is due to the existence of many situations in e-commerce transactions where errors could emerge, thereby leading to unanticipated system failures. Ehikioya [29, 31, 32] as well as Song and Campos [92] posit that the application of formal specification techniques would facilitate the development of e-commerce systems that are error-free, reliable, and dependable, and would guarantee correctness of e-commerce transactions.

Many specification techniques that could be used for specifying e-commerce systems exist in the literature. However, current specification techniques that have been used to specify e-commerce systems only have attributes that can be used in specifying certain components of e-commerce systems (discussed in Section 2.1.3). For instance, some specification techniques have attributes that can be used to model either user interfaces

[1] These characteristics include concurrency, distribution, dynamism, and real-time behaviour (see Section 3.1 for detail discussion).

or data components of an e-commerce system, while other specification techniques have attributes to model the dynamic and real-time behaviour of e-commerce systems. The problem is that none of the existing specification techniques has all the required functionalities to completely specify all the components of an e-commerce system. Thus, the following questions are the subject of this thesis:

i. Which specification techniques are suitable for specifying e-commerce systems?

ii. What characteristic features of e-commerce are necessary to specify a reliable/dependable e-commerce system?

iii. What are the functionalities required of a specification technique to completely specify all the various components of an e-commerce system?

iv. What are the relative strengths and weaknesses of existing specification techniques that can be used to specify e-commerce systems?

v. Which functionalities should designers of specification techniques incorporate in formalisms for specifying e-commerce systems?

vi. Which functionalities should developers of e-commerce look for in a specification technique they want to use in specifying an e-commerce system?

1.2 Objectives

The primary objective of this research is to evaluate specification techniques amenable to specifying e-commerce systems. To achieve this objective, this thesis:

- identifies the requirements necessary for a specification technique to completely specify all the components of an e-commerce system and provides a taxonomy for the requirements;

- examines existing specification techniques for e-commerce systems and provides a taxonomy for the techniques;

- formally specifies a sample business-to-consumer (B2C) e-commerce system case study using some of the specification techniques for evaluation in an effort to assess the strengths and weaknesses of the techniques; and
- develops a simple fuzzy logic framework used to evaluate specification techniques for e-commerce systems.

1.3 Overview of Solution Methodology

This thesis discusses the characteristics that make e-commerce systems amenable to formal specification and presents the requirements needed for a specification technique to completely specify all the components of an e-commerce system and provide a taxonomy for these requirements. The development of these requirements entails carrying out an in-depth examination of e-commerce applications and the requirements of e-commerce applications. Thereafter, questionnaires (on these requirements necessary for a specification technique suitable for specifying e-commerce systems) were administered to some designers of e-commerce systems and academic researchers with focus on the low-level aspects of e-commerce systems. Based on the responses received from the questionnaires, the initial requirements were refined and subsequently evolve the requirements into the criteria for the evaluation of specification techniques for e-commerce applications. Thereafter, existing and emerging specification techniques for e-commerce systems were examined and provided a taxonomy for these techniques.

This thesis adopts a pragmatic and systematic approach in this evaluation of specification techniques for e-commerce systems. To underscore the pragmatic and systematic approach as well as lend more credence to our evaluation, five of the specification techniques for evaluation were used to specify the requirements of the *ordering process* in a typical B2C e-commerce domain. Specifically, this thesis evaluates Petri nets [79], Statecharts [47], Finite State Machines (FSMs) [61], the Z specification notation [108] and some of its widely used variants, Concurrent Z [35] and Real-Time Z [14], as well as Unified Modelling Language (UML) [68, 100]. The decision to evaluate these specification

techniques was influenced by:

- the popularity of the techniques among academic and industrial researchers (i.e., Z, Real-Time Z, Concurrent-Z, FSMs, and Petri nets) and industrial practitioners (i.e., Z, UML, and Statecharts);
- the maturity of the specification techniques, evidenced by the standardization of the techniques; e.g., UML by the Object Management Group (OMG), Z by the International Standards Organization (ISO), and Petri nets by ISO as well as the International Electrotechnical Commission (IEC);
- the availability of verification and validation tools for the techniques, e.g., Z/EVES [85] and fuzz [97] for Z, GDPro [42] for UML, Statemate [48] for Statecharts, and NET-LAB [22] for Petri nets; and
- the existence in the literature (see Section 2.3.1) of the application of the specification techniques or their variants in specifying various components of e-commerce systems.

In addition, a fuzzy logic framework to evaluate the specification techniques was developed, followed by an evaluation of the specification techniques in conjuction with some specialists. Then, a discussion of the findings of the evaluation and a summary of the results in a tabular form.

1.4 Contributions

The main contributions of this thesis include:

i. presents the requirements necessary for a specification technique to completely specify all the components of an e-commerce system and provides a taxonomy for these requirements;

ii. introduces a taxonomy of formal specification techniques for e-commerce systems; this taxonomy provides a broad overview of formal specification techniques for e-commerce systems and their relationships;

iii. presents a simple fuzzy logic framework for evaluating specification techniques for e-commerce systems;

iv. presents e-commerce system designers with the strengths and weaknesses of these formal specification techniques, thereby assisting them in their choice of the formal specification technique to use in specifying their e-commerce applications; and

v. present designers of e-commerce specification techniques with the functionalities required in any specification technique they are developing or hope to develop for modelling e-commerce systems.

1.5 Significance of the Research

This research is significant for the following reasons:

i. It presents the characteristics that make e-commerce systems amenable to formal specification. These peculiar characteristics of e-commerce systems emphasize why certain functionalities should be present in a specification technique that could be used in specifying e-commerce systems;

ii. It presents the requirements required of a specification technique to completely specify all the components of an e-commerce system and provides a taxonomy for these requirements. These requirements are functionalities that should be present in a grand unified specification technique that would be used to specify all the various components of an e-commerce system;

iii. It presents a taxonomy for specification techniques applicable to specifying e-commerce transactions. This taxonomy provides a broad synopsis of specification techniques applicable to capturing the behaviour of e-commerce systems; and

iv. It presents an objective evaluation of specification techniques for e-commerce systems. The use of fuzzy logic principles in evaluating the specification techniques reduces impreciseness, vagueness, and subjectivity associated with non-fuzzy (crisp) evaluation ratings.

1.6 Limitations

The scope of this research is restricted to the evaluation of specification techniques for e-commerce systems using a systematic and pragmatic approach. As a result, only sample specifications composed in some (rather than all existing) specification techniques for a phase (rather than a complete) e-commerce transaction are provided. In addition, the translation of the specifications into implementation as well as the implementation of the fuzzy logic evaluation framework are beyond the scope of this thesis.

1.7 Organization of the Thesis

This chapter (Chapter 1) presents the statement of the problem of this research, the objectives, solution methodology, contributions, significance, and limitations of this research. Chapter 2 presents a review of literature on e-commerce, formal specification, specification techniques, an overview of some of the specification techniques for evaluation, and fuzzy logic. This chapter also presents a review of relevant research work on formal specification of e-commerce systems, evaluation of specification techniques, and taxonomy of specification techniques and various aspects of e-commerce. Chapter 3 presents the distinguishing characteristic attributes of e-commerce systems, the requirements required of a specification technique to completely specify all the components of an e-commerce system, and a taxonomy of these requirements. This chapter also presents a taxonomy for specification techniques for e-commerce systems. Chapter 4 presents the pseudocode of the *ordering process* and the specifications of the requirements in the pseudocode composed in five of the specification techniques evaluated. Chapter 5 presents the evalu-

ation approach adopted in this research, the evaluation criteria, the fuzzy logic evaluation framework, and the findings of the evaluation and the evaluation summary. Finally, Chapter 6 presents the conclusion of the research and provides a roadmap for future work.

Chapter 2

Background and Related Work

This chapter presents a review of the literature relevant to this research work and the related research work.

2.1 Background

This section presents a review of literature on e-commerce, formal specification, specification techniques, and fuzzy logic.

2.1.1 Overview of E-commerce

Laudon and Traver [59] state that e-commerce is the exchange of goods and services between and among organizations as well as individuals using the services of the Internet such as the World Wide Web. The main advantage of e-commerce to e-commerce merchants is making goods and services available to a wider range of customers, at a cheaper, timely, effective, and efficient manner. On the other hand, e-commerce affords customers to have a faster and easier access to a wider range of products.

2.1.2 Categories of E-commerce Systems

Various attempts have been made to classify the categories, models, or forms of e-commerce. Rajput [80] identifies business-to-business (B2B) and business-to-consumer (B2C) domains based on the nature of transactions. Other e-commerce application domains include consumer-to-consumer (C2C), business-to-government, and consumer-to-government domains. These various categories of e-commerce systems (shown in Figure 2.1) are briefly descibed below.

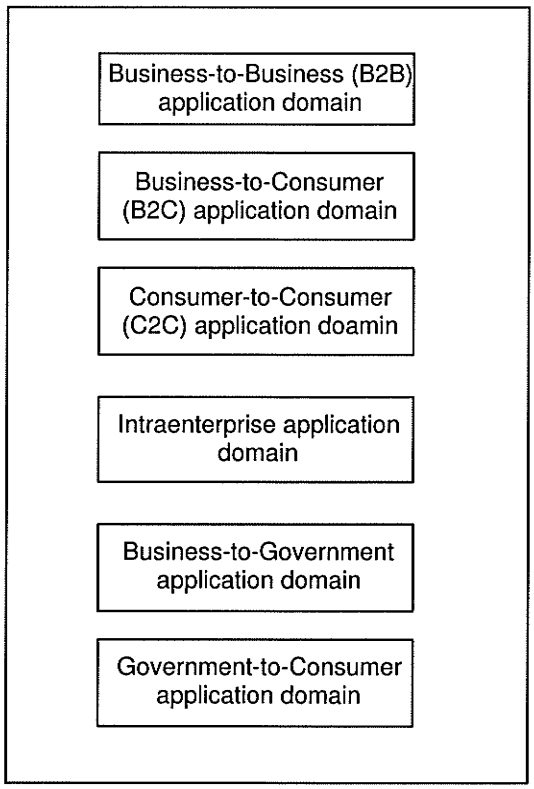

Figure 2.1: Categories of E-commerce

Business-to-Business domain: In B2B application domain, business transaction occurs between two or more business entities. Electronic Data Interchange (EDI) — (transmission of information and data between two or more organizations through networks such as the Internet) — is the most popular form of B2B e-commerce.

eHubs [54] (third-party e-commerce websites that mediate between companies involved in B2B transactions) is another form of B2B that is becoming popular. Some business activities in this domain include inventory control, logistics management, supply-chain management, and buying and selling of organization assets [80]. The users (usually trading and financial organizations' employees) in this domain are fewer than the users in B2C domain. According to Rajput [80], the risks in B2B e-commerce domain is quite high due to the high amount of money involved in the few transactions carried out, which necessitates the need for a foolproof and secure payment medium.

Business-to-Consumer domain: B2C e-commerce application domain (the thrust of this thesis) is the most popular e-commerce application domain. In this domain, goods such as books, computer peripherals, drugs, etc., and services such as news item, entertainment services, games, etc., are sold online by businesses to customers using the Internet [80]. The users (usually customers that request goods or services) in this domain are very many in comparison with the B2B domain. The businesses involved in B2C domain are usually referred to as e-commerce merchants. E-commerce merchants normally publish and display description of goods and services on their websites. The type of transactions in B2C domain, in contrast with B2B domain, usually involve not too high amount of money, due to the limits on acceptable payment mechanisms such as credit cards and debit cards. Online virtual stores are a popular form of B2C e-commerce. A typical B2C e-commerce application provides a framework that enables users to browse a website, search and find items to purchase, negotiate the price of items, compare prices with other websites, add items to a shopping cart, checkout the items (i.e., purchase the items), and make payment for items purchased. Then, the system enables e-commerce merchants to update their inventory, verify payments provided by users', and plan logistics for shipping items to the user. Examples of a B2C system include the online stores amazon.com, barnesandnoble.com, and bookfinder.com.

Consumer-to-Consumer domain: In C2C domain, business is transacted between two

or more individuals. There may be a third party website, which acts as an intermediary between the sellers of goods and services and the prospective buyers. The intermediary also provides secure payment mechanisms and other logistics for the participants and transactions in this domain. Online auctions are a popular form of C2C e-commerce. The financial risks in this domain could be high or low depending on the amount of money involved in the transactions. Examples of a C2C system include the online auction stores ebay.com and auctions.yahoo.com.

Business-to-Government domain: This is a specialised form of B2B e-commerce. In this domain, business transaction is between business entities and governmental agencies and vice versa. Corporate businesses bidding for government contracts' using the Internet fit into this domain. Also, the financial risks in this domain could be high or low depending on the amount of money involved in the transactions.

Government-to-Consumer domain: This is a specialised form of B2C e-commerce. In this domain, government agencies buy or provide goods and services, or information to individuals and vice versa. The financial risks in this domain could be high or low depending on the amount of money involved in the transactions.

2.1.3 Components of E-commerce Systems

A typical e-commerce system consists of user interfaces, a transaction management system, a payment system, a data management system, multimedia artefacts, security framework, logistics, and telecommunications/networking components (see Figure 2.2).

These various components that make up an e-commerce system are discussed below.

- The user interface component consists of mainly the website (provides the medium of communication and business transaction between e-commerce merchants' and customers') and the human-computer interaction interface. The user interface components must possess adequate functionalities and content that will enhance

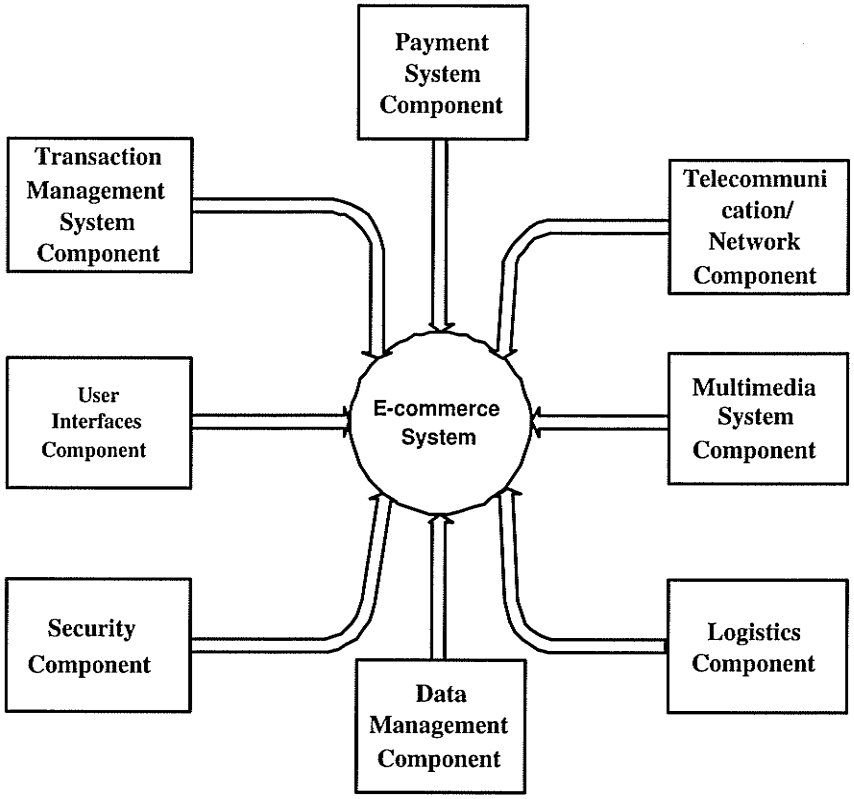

Figure 2.2: Components of a Typical E-commerce System

user friendliness, efficiency, navigability, and maintainability of the e-commerce system.

- The transaction management system is a critical component of an e-commerce system. This component computes taxes and other costs, processes customers' information, as well as ensures that e-commerce transactions conform to the atomicity, consistency, isolation and durability (ACID) properties.

- The payment system provides a framework for accepting payments from customers, verifying payments, and processing payments. The payment system must be secure, robust, flexible, and fool-proof.

- The data management system consists of the low-level abstract functionalities to manage data in an e-commerce system.

- Multimedia artefacts consist of the multimedia components such as graphic, video, and sound available in e-commerce systems. The multimedia artifacts are used to enhance the description of items and services available for sale.

- The security framework must ensure the integrity, privacy, confidentiality, and overall security of e-commerce transactions and applications. In particular, the security framework must have components for authorizing and authenticating users, as well as encrypting users' sensitive information.

- Logistics consists of issues relating to shipping and delivering items to customers, verifying if customers received shipments, and other warehouse related activities.

- Telecommunications/Networking component consists of the interconnectivity framework, which encompasses data, voice, and video transmission. Since e-commerce applications are network-centric and network-intensive, this component must be scalable, robust, dependable, and posses adequate bandwidth that will guarantee a seamless transmission of information across various components of e-commerce systems.

2.1.4 Desired Attributes of E-commerce Applications

E-commerce systems must possess some basic features in order to meet the requirements and expectations of customers' and providers of e-commerce services. Ehikioya [31] identifies some desirable characteristics of e-commerce applications, which include:

Fail-safety: The fail safe property gives a system an appreciable level of correctness and resilience. It also gives a system the ability to gracefully recover from a failure state to a pre-failure state and continue the processing of a transaction to the end.

Guaranteed correctness of operations: The various stages in e-commerce transactions must produce consistent results. The e-commerce system must ensure payment correctness, delivery correctness, product atomicity correctness, operational and transactional correctness, as well as computational results correctness.

Dependability: Randell [81] states that dependability allows customers to rely on the services provided by a system. Dependability can be quantified in terms of correctness (ability to produce consistent results), reliability (ability to produce exact results in a consistent manner), and availability (the system having little or no down time).

Security: E-commerce applications must provide a framework that guarantees the security of customers' and merchants' data.

Scalability: E-commerce applications must be designed in such a way that a new functionality is easy to add after system deployment.

Easily deployed: E-commerce applications must not take too long to be put into productive use and should be easy to deploy.

These desirable attributes of e-commerce systems can be achieved through the use of techniques based on rigour and sound formal basis.

2.1.5 Formal Specification

According to Alagar and Periyasamy [3], formal specification is the application of mathematics and logic-based techniques to represent information necessary for developing software applications. The techniques used in formal specification include: semantic and syntactic analysis of specifications (analyzing the syntax and semantics of a formal specification), specification validation (validating a specification against its' requirements), and formal verification (applying formal verification tools to verify and type-check a specification for errors) [3].

Hull [51] identifies the major parts of system behaviour that are mostly specified to include: concrete interfaces, concurrency, abstract functionality, performance, availability, reliability, and maintainability. Alagar and Periyasamy [3] identify some reasons why system developers specify software applications, which include:

- to clearly illustrate the validity of a system property,
- to concisely depict the interface of system components and examine hidden behaviours,
- to localize the complexity of a system design and illustrate requisite behaviours inherent in the design, and
- to contain effects of change and prove that the resultant application satisfies its initial system requirements.

There is need to adopt formal specification techniques in developing complex and safety-critical applications. The application of formal specification to specify software systems is not a complete panacea to all problems affecting software development. Ehikioya [29, 31] and Hull [51] identify the following as benefits derivable from using formal specification:

- reveals ambiguities, incompleteness, unstated assumptions, and contradictions in requirements analysis,
- allows system analysis and refinement into design and code,
- makes a system easier to test and understand,
- allows verification of correctness, thereby improving the system's reliability,
- provides abstraction for the functionalities required to manage other features of the system,
- represents various components of the system simply, and
- provides internal consistency, completeness, and allows hierarchical representation.

2.1.6 Overview of Fuzzy Logic

Fuzzy logic [55, 110] is the application of fuzzy set (a group of objects with no clear boundary between objects that are in a group and objects that are not in the group) theories and techniques to capture imprecision, uncertainty, and vagueness associated with linguistic terms in natural language. Characteristic functions (map elements of a set to 0 or 1) are used to define crisp (nonfuzzy) sets. Elements of crisp sets are either member of a crisp set or are not members. Values in a crisp set are either true or false. An element is a member of a crisp set if it has a value of 1, and does not belong to a set if it has a value of 0. Membership functions are use to uniquely define fuzzy sets. The membership function describes the degree or extent to which an object belongs to a fuzzy set, it assigns a numerical value to elements in a fuzzy set and maps nonfuzzy inputs from a particular domain to membership values, which lie on the interval $[0, 1]$. In sharp contrast with conventional set theory, fuzzy logic goes beyond the boundaries of true or false and deals with the degree of truth or falsity. For instance, if a fuzzy set B is defined on a universal set Z, for every element z of Z, the symbol $B(z)$ represents the degree of membership of z in B. Generally, membership functions are used to describe sets, which satisfy to various degrees, properties that can be described using linguistics terms like *small, long, great, little, tall, short, etc.*

According to Dubois and Prade [27], a fuzzy logic system consists of a rule base (group of fuzzy rules that forms conditional If-Then statements), membership functions, and an inference procedure that conforms to arithmetic rules (addition, subtraction, and multiplication of fuzzy numbers). Fuzzy rules describe actions that should be taken when given different fuzzy inputs. An instance of a fuzzy rule is: If $(z = A)$ Then $(y = B)$. Generally, a fuzzy logic system is a system whose variables belong to a range of fuzzy numbers [55]. Linguistic variables are fuzzy numbers that represent fuzzy linguistic terms in a certain universe of discourse. Linguistic variables are described in terms of base variables — physical or numerical variables having real number values within a specified range. According to Klir [55], every linguistic variable has a name (must be relevant to the meaning of the base variable), a base variable with its range of values, a

set of linguistic terms (that refer to values of the base variable), and a semantic rule that assigns meaning to each linguistic term. The states of variables in fuzzy logic systems can be represented using graphical, tabular, triangular, or trapezoidal representations.

The application of fuzzy logic helps in eliminating vagueness, imprecision, and uncertainty associated with linguistic terms in natural language. To achieve these benefits, fuzzy logic undergoes processing. The steps involved in processing fuzzy logic include fuzzyfication, rule evaluation, and defuzzyfication.

Fuzzyfication: This step entails transforming crisp inputs into fuzzy inputs, after determining the membership function of the corresponding fuzzy inputs. Thereafter, assign fuzzy labels in the universe of discourse to every crisp input.

Rule evaluation: Rule evaluation (fuzzy inference) uses the min-max inference scheme (fuzzy set operations for intersection and union) to obtain a numerical value, which represents the truth for an action based on the fuzzy rules. The min-max inference scheme compares two fuzzy values. The output of this step is a fuzzy output.

Defuzzyfication: This is the last step in processing fuzzy logic. In this step, the expected output value is obtained by isolating a crisp value in the universe of discourse of the output fuzzy sets.

2.1.7 Overview of Specification Techniques

Specification techniques are methods that use mathematical and non-mathematical based techniques to describe system behaviour. Specification techniques can be broadly classified into the following categories: formal, semi-formal, and informal specification techniques. Formal specification techniques are methods premised on mathematics and have well-defined syntax and semantics. Some formal specification techniques are Z, Communicating Sequential Processes (CSP) [13, 50], LOTOS [62], and RAISE [45]. Specifications derived from the use of formal specification techniques present abstract and precise descriptions of system behaviours and properties. Semi-formal specification techniques

are methods that use non-rigorous and graphical structures such as diagrams and graphs to model system behaviour, while informal specification techniques use non-formal notations such as natural languages, pseudocodes, and informal descriptions to describe system behaviour. Some semi-formal specification techniques include UML, Statecharts, and Petri nets, while pseudocodes and natural language descriptions written in English are instances of informal specification techniques.

2.2 Specification Techniques Fundamentals

This section presents the fundamentals of the various specification techniques used in specifying the sample case study in this research.

2.2.1 Petri nets

Petri nets [79, 83] are graphical but mathematical modelling tools that consist of places (for modelling conditions), transitions (for modelling events), and arcs to link the places with the transitions. Input arcs link places with transitions, while output arcs begin at a transition and terminate at a place. The number of tokens in each place gives the current state of the modelled system. Places contain tokens, while transitions are active components used for modelling events or activities, which might occur, thereby modifying system state. The Petri net is marked at this state.

Transitions are allowed to fire if all preconditions (inputs) for the activity or event have been met, while postconditions are output conditions. This implies that there are enough tokens available in input places. When a transition is fired, it takes tokens from its input places and adds them to some of its output places. Tokens are used in Petri nets to simulate the dynamic and concurrent processes in a system. In a Petri nets model, a circle $\bigcirc$ represents a place, a bar | represents a transition, while an arrow $\rightarrow$ is used to connect places with transitions.

The process of collating (counting) election votes by the returning officer of an election is used to illustrate modelling with Petri nets. After an election takes place, the returning officer receives the votes cast in ballot boxes from the various voting centers, then the returning officer collates the results and sends the results to the candidates in the election. The conditions (places description) and events (transitions description) of this example are given in Table 1, while Table 2 shows the preconditions and postconditions of each event in the example. Figure 2.3 shows the Petri net model of the election votes counting example.

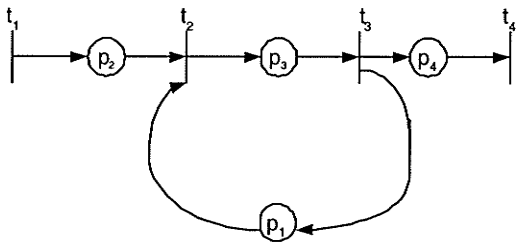

Figure 2.3: Petri-nets Model of Collating Election Results

Place Descriptions	**Transition Descriptions**
p_1: returning officer awaits election results	t_1 : election results arrive from polling centers
p_2: election results are awaiting collation	t_2 : returning officer begins collating election results
p_3: returning officer is collating election results	t_3 : returning officer finishes collation of election results
p_4: election results are ready	t_4 : returning officer sends results to candidates

Table 1: Places and Transitions Descriptions for Collating Election Results.

Events	**Preconditions**	**Postconditions**
t_1	—	p_2
t_2	p_1, p_2	p_3
t_3	p_3	p_4, p_1
p_4	p_4	—

Table 2: Events, Preconditions and Postconditions.

2.2.2 Z Notation

The Z notation [97, 108] is based on first-order predicate logic and set theory (includes standard set operators, cartesian products, power sets, and set comprehensions). Set theory provides the framework for composing abstract structures for the system, while first-order predicate logic is use to describe system behaviour.

The Z notation captures the abstract functionality of a system's behaviour by describing pre-conditions, post-conditions, and invariants. Z consists essentially of schemas — an encapsulated structure associated with certain properties. Schemas consist of a schema name, the variables declaration section, and a predicate section (which contains the constraints on the variables in the declaration section). Schemas are used to describe system states, changes in system states, represent types, operations, define system properties, and reason about possible refinement of a system design. The use of types is another feature of the Z notation. Objects in Z are associated with distinct types; types provides a veritable link to programming and facilitates easy development of type-checking tools for the notation.

The contact list of the customers of an organization is used to illustrate the use of the Z notation. The contact list stores the names and addresses of customers of an organization. When a new customer starts doing business with the organization, the name and address of the customer is added to the contact list. Also, when a customer stops doing business with the organization, the name and address of the customer is removed from the contact list. To begin the specification, the following basic types are defined.

$$[CHAR, STRING]$$

First, I describe the state space of system using the schema:

CustomerContact

$$custid, name : \text{seq } CHAR;$$
$$address : STRING$$

$$\#name > 0 \land \#address > 0$$

CustomerContact is the name of the schema, the first part of the schema shows the declaration of the variables *name*, *custid*, and *address* and their associated types. The variable *custid* is introduce to uniquely identify the ustomers, since two or more customers may have the same name. The second part of the schema (predicate section) shows the constraints on the variables. In this case, the constraints include the number of characters in the name and address of a customer must be greater than 0. The symbol $\wedge$ is used to represent *logical and* in Z.

The schema *CustomerContactList* describes the set of all customers of the organization in the contact list. *CustomerContactList* has only one state variable *Customers* defined as a set of *CustomerContact*. The schema *initCustomerContactList* describes the initial state of the contact list when no information has been added. The declaration Δ *CustomerContactList* shows that the schema is describing a change of state.

CustomerContactList

$$Customers : \mathbb{P}\ CustomerContact$$

initCustomerContactList

$$\Delta CustomerContactList$$

$$Customers = \emptyset$$

An important constraint is that the customers of the organization must be unique, i.e, no two customers should have the same *custid*. The uniqueness of customers constraint is defined as follows:

$$\forall a, b : CustomerContact \bullet a \neq b \Rightarrow a.custid \neq b.custid$$

Now, some operations that can be performed on the contact list are described below. When a a customer starts doing business with an organization, his/her contact information is added to the contact list. The schema below captures this operation.

$$
\begin{array}{|l}
\textit{AddCustomer} \\
\hline
\Delta \textit{CustomerContactList} \\
\textit{newname}?, \textit{name}' : \text{seq } \textit{CHAR}; \\
\textit{newaddress}?, \textit{address}' : \textit{STRING} \\
\hline
(\forall\, a : \textit{CustomerContact} \mid a \in \textit{Customers} \bullet \\
a.\textit{name} \neq \textit{newname}? \land a.\textit{address} \neq \textit{newaddress}?) \\
\textit{name}' = \textit{name} \cup \{\textit{newname}?\} \land \textit{address}' = \textit{address} \cup \{\textit{newaddress}?\} \\
\hline
\end{array}
$$

The variables *newname*? and *newaddress*? are the new input variables to the operation, while *name'* and *address'* are the state of the variables after adding the name and address of the new customer. In Z, the name of input variables end with a question mark, while the name of output variables end with the exclamation mark.

Also, when a customer decides to leave the organization, the name and address of the customer is removed from the contact list. The schema below captures this operation.

$$
\begin{array}{|l}
\textit{RemoveCustomer} \\
\hline
\Delta \textit{CustomerContactList} \\
\textit{name}', \textit{removename}? : \text{seq } \textit{CHAR}; \\
\textit{address}', \textit{removeaddress}? : \textit{STRING} \\
\hline
(\forall\, a : \textit{CustomerContact} \mid a \in \textit{Customers} \bullet \\
a.\textit{name} = \textit{removename}? \land a.\textit{address} = \textit{removeaddress}?) \\
\textit{name}' = \textit{name} \setminus \{\textit{removename}?\} \land \textit{address}' = \textit{address} \setminus \{\textit{removeaddress}?\} \\
\hline
\end{array}
$$

The variables *removename*? and *removeaddress*? are the new input variables to the operation, while *name'* and *address'* are the state of the variables after removing the name and address of an old customer.

2.2.3 Statecharts

Statecharts [47] is a graphical specification modelling tool for describing states and transitions in a modular way. Statecharts is based on the conventional state machine model, which describes a system's behaviour using state transition diagrams. States are represented with rounded boxes and arrows $\rightarrow$ represent transitions between states.

Gomaa [44] identifies the following types of statecharts: flat, hierarchical, concurrent, and top-level statecharts.

Flat statecharts: This is the traditional statecharts in which nodes or boxes represent states and arcs represent transitions between states. However, flat statecharts results in explosion of states and transitions, making them difficult to read, comprehend, and understand [44].

Hierarchical statecharts: Hierarchical statecharts solves the problem of explosion of states and transitions in flat statecharts. In hierarchical statecharts, higher-level superstates are decomposed into lower-level substates. Every transition into a superstate is a single transiton into one of the lower-level substates in the superstate. Also, every transition from a superstate must originate from one of the lower-level substates.

Concurrent statecharts: This is a form of hierarchical statecharts, in which a statechart is decomposed into at least two concurrent susbstates (the concurrent susbstates are usually separated by a dash line). The activities in the concurrent substates need not occur concurrently.

Top-level statecharts: This is also a variant of hierarchical statecharts which shows only events and excludes the actions that causes the events.

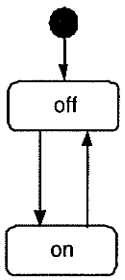

Figure 2.4: Statecharts Model of a Simple Light Switch Operation

The operation of a simple light switch is used to illustrate the modelling power of statecharts. The light switch works as follows: the light switch is initially in the *off* state, when it is pressed, it goes to the *on* state, then the light comes on. When the switch

is released, it goes to the *off* state and the light goes off. Figure 2.4 shows a simple statechart model of a simple light switch operation.

2.2.4 Finite State Machines

Finite state machines (FSMs) [61] are powerful abstractions for modelling system behaviour. An FSM consists of a set of states — including a start state as well as some terminal state(s), a set of input events, a set of output events, and a state transition function. The state transition function takes a current state and an input event and returns a new set of output events and the next state. There are two types of FSMs — deterministic and non-deterministic finite state machines. In deterministic FSMs, there is at most a single transition for each possible input for a specific state, while in non-deterministic FSMs there may be more than a single transition from a given state for a given input.

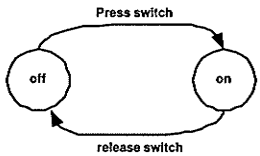

Figure 2.5: FSMs Model of a Simple Light Switch Operation

In FSMs representation, a circle ◯ is use to represents a state , while an arrow → is use to represent transitions between states. Figure 2.5 is a simple deterministc FSM diagram illustrating the light switch operation. In figure 2.5, either of the states be the start state or end state.

2.2.5 Unified Modelling Language

UML uses diagrams to model system behaviour and properties. UML is used for developing object-oriented and component-based systems and is rich in semantics and notations suitable for modelling systems of different architectural complexities across various domains. The various diagrams in UML are briefly discussed below.

Use case diagrams: Use case diagrams are used to depict actors, use cases, and their relationships. A use case diagram describes the interactions between the actors and the various system components. In a use case diagram, a stick-like figure is use to represents actors, an ellipse to represents use cases, while arrows represent their relationships.

Class diagram: A class diagram shows static data view for part of a system as well as shows classes and their relationship with one another. In a class diagram, rectangular boxes represent classes, while arcs represent their relationship.

Object diagram: An object diagram is related to class diagrams and it is used to show particular instances of classes in a system at run-time. Rectangular boxes are used to represent objects, while arrows are used to represent their relationships.

Sequence diagram: A sequence diagram shows objects and how they relate in a chronological manner. It also shows the interaction between objects/classes in systems for each use case. Rectangular boxes are used to represent objects while labelled horizontal arrows represent messages.

Collaboration diagram: A collaboration diagram shows objects and their relationship. A collaboration diagram uses numbers to indicate messages that depict relationship between objects. Rectangular boxes are used to represent objects while arcs are used to represent their relationships.

Statechart diagrams: They are used to show the states, transitions, events, and behaviour of a class. Rounded boxes are use to represent states and arrows represent transitions between states.

Activity diagrams: Activity diagrams show the sequence of activities in a use case. They show activities, state transitions, and events. Rounded boxes are used to represent activities and arrows are used to represent transitions.

Deployment diagrams: A deployment diagram shows how software systems are distributed across an enterprise, shows components, nodes, and their relationship with

one another. Cube shape boxes represent the distribution of the software systems while arcs represent their relationships.

Component diagrams: Component diagrams show architectures of software systems and their associated dependencies. Rectangles with tabs are use to represent the architectures of software systems and arrows are use to represent their dependencies.

Since the various UML diagrams are easy to read, interprete, and understand as well as the high number of diagrams in the notation and the consequent space constraints, I omit the use of examples to illustrate the diagrams.

2.3 Related Work

This section highlights previous research related to my research work. This section begins with a discussion of research efforts on formal specification of e-commerce systems and subsequently discuss the evaluation of formal specification techniques in the literature. Thereafter, a review of taxonomies of some aspects of e-commerce systems and taxonomies of specification techniques in the literature is provided.

2.3.1 Formal Specification of E-commerce Systems

Several research work have been carried out on the application of formal specification methods such as Z, UML, FSM, Statecharts, Model checking [23], RAISE, Timed CSP [86], and Coloured Petri nets (CPN) [57] to model various aspects and components of e-commerce applications. Some of these research efforts are discussed below.

Patra and Moore [75] apply the RAISE specification language to formally specify (buyer, seller, broker, product) agents operating in a virtual electronic market place. They use the open-cry e-commerce auction to illustrate the applicability of their methodology. Kumar and Feldman [58] formally model price negotiations in e-commerce auctions

using FSM. They captured the interaction and relationship between the building blocks of different e-commerce auctions. Rohm *et al.* [84] introduce and use the modelling security semantics (MOSS) methodology to specify security and fairness issues in e-commerce applications. Ehikioya [29] as well as Ehikioya and Barker [33] use the event-based Timed CSP [86] to specify and verify the transactional requirements of an electronic shopping mall, an early precursor of electronic commerce. They also specify causal dependencies and the necessary concurrency controls protocols to guarantee correct and reliable e-commerce transaction processing.

The research work reported in [29, 33, 58, 75, 84] use different specification techniques to capture functional as well as non-functional requirements of e-commerce applications, though [58, 75] focus on e-commerce auctions.

The research work in [25, 70, 71] apply identical specification technique to model functional behaviour of e-commerce applications. Cost *et al.* [25] apply CPN to formally model conversations among agents in an e-commerce system. Ouyang *et al.* [71] use both CPN and FSM to model a trading service based on the Internet Open Trading Protocol[1] (IOTP) [15], while Ouyang *et al.* [70] present a formal and executable specification of transactions in IOTP using CPN.

Ehikioya [30] applies the Z notation to formally specify the correctness and reliability requirements in satisfying a customer's order from data distributed across the Internet. He also describes techniques for guaranteeing and improving the dependability of e-commerce systems. Ehikioya [31] applies informal descriptions, UML, and Z to formally specify and verify an abstract model of an e-commerce system. He uses Z to model the concrete definitions of objects and calculations of the conditions of operations of the e-commerce system, and UML diagrams to capture the relationships between, as well as the interactions among, the subcomponents of the e-commerce system. Ehikioya [32] also uses Z to formally specify the key transactional processing requirements for B2B and B2C e-commerce applications.

The various research efforts in [30, 31, 32] model various functional requirements of

[1] A trading protocol for B2C e-commerce transactions.

e-commerce applications. The research work in [31, 32] illustrate how the formal specifications can be used to create real-life e-commerce systems using a suitable programming language, while the research work in [30] focus on correctness and reliability issues in e-commerce applications.

Ehikioya and Hiebert [34] apply UML to formally specify the correctness and reliability requirements of an e-commerce system and they present an architecture, as well as a design of a prototype e-commerce system. Pastor *et al.* [74] introduce an extension to UML, and they apply the extension to UML to formally specify the navigation design of e-commerce websites. They also introduce object-oriented conceptual modelling techniques for specifying certain components of an e-commerce system. Franz *et al.* [39] examine the business, system engineering, and communication technology requirements for e-commerce applications, and use UML to formally specify and define interfaces between the business, structure, and communication components of e-commerce applications. They also investigate the differences and similarities between these three components and present an integrated architecture of the components. Li *et al.* [60] present a formal application of UML to the analysis and design of e-commerce systems using an online ticketing system as a case study. They subsequently demonstrate how the formalization can be used to verify the correctness of functional requirements (e.g., products booking and payment processing), safety, and liveness constraints of the online ticketing system.

The research work in [34, 60] use UML to capture various functional requirements of e-commerce applications, while [39, 74] use UML to model non-functional requirements of e-commerce applications. Franz *et al.* [39] approach the modelling of non-functional requirements of e-commerce systems from an interdisciplinary perspective. This interdisciplinary approach differentiates [39] from the approach in [34, 60, 74].

Ray and Ray [82] use the FDR model checking tool [36] to model and subsequently verify that an e-commerce protocol (for online transactions involving digital items) satisfies money atomicity, goods atomicity, and validated receipts properties. They also use FDR to detect violations of these properties during website failures and communica-

tion link failures in an e-commerce system, and develop a mechanism to safeguard these properties during site or link failures. Fu *et al.* [40] apply model checking to formally specify and verify the correctness of the internal logic of e-commerce workflow specifications. They use three different model checking techniques to guide the construction of a composite e-commerce service that guarantees desirable properties such as deadlock avoidance and limits on resource usage. Wang *et al.* [105] use some model checking tools to formally specify and verify e-commerce systems using an online ticket sales system as a case study. They also apply model checking to demonstrate the complexity of e-commerce systems, as well as detect subtle flaws and errors in the ticket reservation, ticket payment, and ticket cancellation processes of the online ticket sales system. The research work in Song *et al.* [94] apply model checking and design patterns to formally specify and verify an e-commerce case study. Song *et al.* [95] as well as Song and Campos [93] introduce UML-CAFE (an integration of UML, the Formal-CAFE methodology [77], and design patterns) and use it to specify and verify an e-commerce case study. Pereira *et al.* [76] apply a symbolic model checking methodology, which they introduced in [77], to formally specify an e-commerce system using a virtual store as a case study. Tygar [99] introduces the atomicity property in e-commerce transactions and emphasizes the importance of atomicity in e-commerce systems. Subsequently, Pereira *et al.* [76] also automatically verify that the specifications of the e-commerce system satisfy the atomicity, isolation, and consistency properties.

The research efforts in [76, 93, 94, 95] use techniques based on the Formal-CAFE methodology to specify and verify various e-commerce case studies. The research work in [82, 105] use various case studies to illustrate how e-commerce applications can benefit from formal specification and verification. The formal specifications in [76, 105] are generic and can be customized to model other e-commerce case studies.

Yolum and Singh [109] introduce Commitment-based Finite State Machines (CFSM) — premised on meaning and not sequence of actions. They use CFSM to model and analyse the NetBill protocol to illustrate their agent-based approach, which enhances the flexibility of executing the protocol. Papa *et al.* [73] use an integration of logic and

process calculus to formally analyze and verify security and reliability properties of the NetBill [26, 87] e-commerce payment protocol. Bolignano and Dyade [11] apply finite automaton and the filtering function of model checking to specify the payment properties of the Chip-Secure Electronic Transaction (C-SET) protocol — a smartcard-based payment protocol. Bolignano and Dyade [11] also formally verify different trustability hypotheses of the C-SET protocol. Heintze *et al.* [49] apply FDR model checker to formally verify if goods atomicity and money atomicity properties hold for the NetBill and Digicash [20, 21] e-commerce payment protocols. Lowe [63] applies model checking to formally verify the completeness property of some e-commerce security protocols.

The research efforts in [11, 49, 73, 109] use various model checking tools, CFSM, an integration of logic and process calculus to formally specify and verify different properties of e-commerce payment protocols, while [63] applies model checking to formally verify e-commerce security protocols. The techniques used in [11] and [49] are generic and can be applied to verify properties of other e-commerce protocols.

The application of diverse specification techniques in [11, 25, 29, 30, 31, 32, 33, 34, 39, 40, 49, 58, 60, 63, 70, 71, 74, 73, 75, 76, 82, 84, 94, 95, 93, 105, 109] to model various e-commerce components and case studies reflect the growing acceptance, popularity, and importance of the use of specification techniques in modelling e-commerce systems. These research efforts on the formal specification of e-commerce systems underscore:

i. specifications reveal ambiguities, inconsistencies, contradictions, incompleteness, and unstated assumptions in e-commerce systems,

ii. specifications enhance the understanding and testability of e-commerce systems,

iii. specifications allow verification of correctness of e-commerce systems, thereby enhancing their reliability, and

iv. specifications provide internal consistency, completeness, and allow hierarchical representation of e-commerce systems.

2.3.2 Evaluation of Specification Techniques

This section reviews research work on the evaluation of various specification techniques in the literature.

Mills [66] experimentally evaluates specification techniques for improving functional testing using an automobile cruise control system as a case study. The experiment compares the effectiveness of software testing with three different sets of software specifications — natural language description, natural language description combined with real-time structured analysis (RTSA) [65], and natural language description combined with real-time structured analysis and an executable specification.

Ardis *et al.* [4, 5] evaluate six formal specification techniques (Modechart [52], Virtual Finite State Machine (VFSM) [104], ESTEREL [7], Basic LOTOS [62], Z, Specification and Description Language (SDL) [96]) and a programming language (C language) for reactive systems using a telephone switching system as a case study. They base their evaluation of the formal specification techniques on the following criteria: applicability, implementability, testability, checkability, maintainability, modularity, level of abstraction, soundness, verifiability, run-time safety, tools maturity, looseness, learning curve, language maturity, data modelling, and discipline. Their findings indicate that the Z notation and the C programming language are the least well suited specification techniques for the application domain they considered in their work.They stated that ESTEREL is the most expressive of the specification techniques evaluated by them.

van Harmelen *et al.* [101] evaluate a specific formal specification technique for knowledge based systems, by investigating the usability of $(ML)^2$ [102] — a knowledge-based systems (KBS) formal modelling language used to represent Knowledge Analysis and Design Support (KADS) conceptual models of expertise. They evaluate $(ML)^2$ by designing a set of evaluation criteria (expressivity, frequency of errors, redundancy, locality of changes, reusability, and support) and applying two different case studies to score the formal modelling technique against the evaluation criteria.Their evaluation findings show that $(ML)^2$ scored high on most of their evaluation criteria. Their findings also indicate

a close correlation between the structure of informal KADS models and formal $(ML)^2$ models.

Knight *et al.* [56] evaluate three formal specification techniques (Z, Statecharts, and Prototype Verification System (PVS) [72]) using a safety-critical application (University of Virginia nuclear reactor) as case study. They assess the specification techniques using the following criteria: coverage, integration with other components, group development, evolution, usability, compatibility with design tools and methods, communication of desired characteristics to designers, facilitation of design process, implementation performance, maintenance comprehension, and maintenance changes.

Benyoucef and Keller [9] present an evaluation of five specification techniques (natural language, agents coordination language (ACL) [19], FSM, OMG negotiation facility [69], and Statecharts) for describing negotiation processes in e-commerce auctions. Their evaluation criteria include: formal basis, serialization, visualization, executability, popularity, clarity, completeness, and convertibility. Their evaluation results show that Statecharts is best suited for capturing the rules governing negotiation processes in e-commerce auctions.

The research work in this thesis differs from [4, 5, 56, 66, 101] because it evaluates specification techniques for a different application domain, although the case studies in [4, 5, 56, 66] have most of the characteristic attributes peculiar to e-commerce systems, such as concurrency, dynamism, and real time behaviour. Also, this research work differs completely from [101] which compares a single formal specification technique for different case studies, while my work compares different specification techniques using one case study. The specification techniques this thesis evaluates differ completely from the techniques in [66, 101], and differ partially from the techniques evaluated in [4, 5, 9, 56]. The evaluation criteria used in this thesis differ sharply from the criteria used in [9, 56, 66, 101], while a few of the criteria are similar to those used in [4, 5]. Also, the research work in [4, 5, 56, 66, 101] adopt experimental evaluation approaches, whereas this thesis adopts a pragmatic evaluation approach based on the experimental use of the techniques and the theoretical attributes of the techniques. In

addition, the research work in this thesis differs specifically from the research work reported in [9] as follows:

i. Although [9] and this research work address the e-commerce domain, [9] focuses on negotiations in e-commerce auctions (C2C) e-commerce domain, while this thesis focuses on the (B2C) e-commerce domain, which is more popular, more widely patronized by users, and attracts a higher volume of e-commerce transactions.

ii. The specification techniques evaluated in [9] are all not rigorous (i.e., not based on mathematics), while some of the techniques evaluated in this thesis are based on mathematical rigour.

In contrast with the use of non-fuzzy principles in evaluating the specification techniques in [4, 5, 9, 56, 66, 101], which results in subjective judgement, this thesis adopts fuzzy logic theory in assessing the specification techniques. The application of fuzzy logic in evaluating the specification techniques enhances the objectivity, linguistic precision, and clarity of the evaluation.

2.3.3 Taxonomy of E-commerce and Specification Techniques

This section briefly discusses some research work on the classification of various aspects of e-commerce systems and specification techniques.

Kaplan and Sawhney [53] develop a taxonomy for classifying B2B e-commerce hubs using value creation mechanism, purchase situation, and bias of the market-maker as the classification criteria. Bartelt and Lamersdorf [8] present a multi-criteria taxonomy for e-commerce business models and systematically classify the models using suppliers, mediators and customers and their respective roles as carrier or initiator of the business model as the classification scheme. Vijayaraghavan [103] presents an extensive and exhaustive taxonomy of probable and actual failure points as well as potential and existing risks for e-commerce systems. This taxonomy facilitates the generation of issues to test for in e-commerce systems. Misic and Zhao [67] adopt a classification scheme premised

on linguistics and develop a conceptual framework for evaluating reference models for e-commerce systems.

Existing taxonomies of specification techniques in the literature include Wieringa [106] (a taxonomy of object-oriented and structured software specification techniques) and Wing [107] (a general taxonomy of specification techniques). Wieringa [106] classifies object-oriented and structured software specification techniques into two broad categories, which include: external interaction techniques and internal decomposition techniques. He further sub-classifies external interaction techniques as function specification techniques, behaviour specification techniques, and communication specification techniques. Wing [107] use the characteristics of specification techniques as the criteria to classify the techniques. She identifies the following categories of specification techniques: model-oriented techniques, property-oriented techniques, visual techniques, executable techniques, and tool-supported techniques.

Chapter 3

Specifying E-commerce Systems

This chapter presents the distinguishing essential characteristic attributes of e-commerce systems, the requirements required of a specification technique to completely specify all the components of an e-commerce system, and a taxonomy of these requirements. Also, this chapter presents a taxonomy of specification techniques for e-commerce systems.

3.1 Essential Characteristics of E-commerce Systems

E-commerce systems enable customers to make online purchases. A typical B2C e-commerce system allows customers to search and find items to purchase, negotiate the price of items, add items to a shopping cart, checkout items (i.e., purchase items), and pay for items purchased; the system also allows e-commerce merchants to update their inventory, verify customers' payment methods and plan logistics for shipping items to the customer.

E-commerce systems are inherently complex. The complexity of e-commerce systems results from the concurrent, distributed, dynamic, and real-time behaviour of e-commerce transactions. These characteristics (discussed below) underscore the need for certain attributes in a specification technique applicable to specifying e-commerce systems.

Concurrency: Many processes (a unit of concurrent activity) in e-commerce transactions may execute concurrently. Astesiano *et al.* [6] state that this interaction among concurrently executing processes may involve communication, synchronization, cooperation, acting in parallel, and competing for resources with other processes and the environment. For example, debiting a customer's credit card account and crediting the e-commerce service provider's credit card company could occur concurrently. However, concurrency controls must be in place to preserve transaction isolation.

Distribution: E-commerce systems are inherently distributed. E-commerce systems are usually distributed over possibly heterogeneous databases, web servers, networks, and operating systems across various locations. E-commerce transactions are characterized by complex data access patterns. Also, distributed e-commerce transaction processes can be invoked directly or indirectly from remote locations.

Dynamism: Dynamic systems, according to Zamulin [111], have a number of states and changes are made to these states from time to time. When e-commerce transactions occur, various database tables are updated, therefore, changing the database state. In e-commerce systems, this dynamic behaviour results in data dependencies as well as the need for proper synchronization and communication among the various subcomponents of the system.

Real-time Behaviour: E-commerce transactions occur in real time. Real time implies the interaction between the e-commerce system and its environment occurs instantaneously. The real-time behaviour of e-commerce is influenced by the input environment and the application processing requirements. The system captures input from customers in real time. The real-time requirements of a system usually make some constraints on the input environment and output environment of the system.

Formally, these essential characteristic behaviours that make e-commerce system complex and amenable to formal specification are a quintuple, $\langle Co, Di, Dy, Rt, Ecb \rangle$

where Co, Di, Dy, and Rt are sets representing the concurrent, distributed, dynamic, and real-time behaviour of e-commerce systems, respectively, while $Ecb = Co \times Di \times Dy \times Rt$ represents the relation between them. These four characteristics underscore the need for certain features in a specification technique for specifying e-commerce systems.

3.2 Formal Specification Techniques Requirements

In developing the requirements necessary for a specification technique to completely specify all the components of an e-commerce system, an exhaustive examination of e-commerce applications and the requirements of e-commerce applications was carried out. Thereafter, fifty questionnaires (on these requirements necessary for a specification technique suitable for specifying e-commerce systems) were administered to some designers of e-commerce systems and academic researchers with focus on the low-level aspects of e-commerce systems. Based on the responses from the forty-two questionnaires returned by the respondents, the requirements were refined and subsequently evolved into the criteria for the evaluation of specification techniques for e-commerce applications.

The requirements are classified into critical, essential, and secondary requirements (see Figure 3.1). Critical requirements are the functionalities necessary to capture the core distinguishing characteristics of e-commerce applications - $\langle Co, Di, Dy, Rt \rangle$, while essential requirements are functionalities needed to capture the abstract functional properties and non-functional properties of e-commerce systems. Secondary requirements are non-technical features dealing with the usability of the specification technique.

3.2.1 Critical Requirements

The critical requirements consist of two broad properties: non-determinism and timing mechanism. These are briefly explained below.

Nondeterminism: The specification technique should have nondeterministic constructs for indirectly accessing data and unlimited choice from a set of attributes. Nonde-

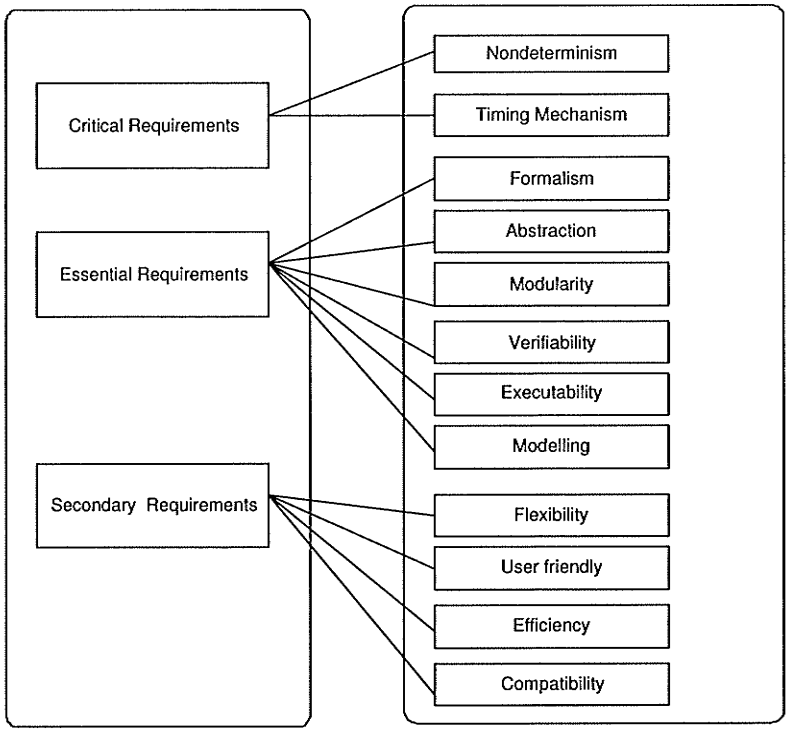

Figure 3.1: Taxonomy of Requirements for E-commerce Specification Techniques

terminism is necessary to capture and model concurrency, synchronization, as well as the parallel behaviour of an e-commerce application.

Timing mechanism: The specification technique should have constructs to capture and model time-dependent objects and actions. The technique should also have constructs for describing and analysing the sequences of system states, as well as constructs for capturing temporal system properties in future system states. Timing mechanism is necessary to specify time, associate time with object operations as well as capture dynamic and real-time properties of e-commerce applications.

3.2.2 Essential Requirements

This section discusses the essential requirements.

Formalism: A formal specification technique for e-commerce applications must have

well-defined syntax and formal semantics. This attribute will enable the specification technique to handle ambiguities, incompleteness, and contradictions in e-commerce applications.

Abstraction: The specification technique should have potent primitives for describing and manipulating data at the logical level. The technique should also have constructs for defining objects in e-commerce systems by a set of associations with which it interacts with other objects. The method should possess abstract operations that can create, destroy, access, and relate objects. This attribute makes it feasible to have precise, consistent, correct, and unambiguous specifications. This attribute is necessary to capture the functional properties of e-commerce systems, such as data and processing behaviours.

Modularity: The specification technique should have constructs for composing smaller specifications and extending the specifications, thereby enabling the derivation of large and complex specifications from smaller specifications. The inherent complexity of e-commerce system underscores the need for this attribute in any formal technique applicable to specifying e-commerce applications.

Modelling: The specification technique should have constructs for specifying data objects and the interactions among the components of the data objects. Also, the technique should have constructs for describing the actions active data objects should perform in e-commerce applications. This attribute is necessary to capture the non-functional properties of e-commerce systems such as user interfaces, human-computer interaction and system usability besides the modelling of process interactions and logic.

Verifiability: The specification technique for specifying e-commerce applications should have mechanisms for type-checking, verifying, and validating specifications. It is imperative to verify if specifications are well-defined. It is also necessary to ensure that a system meets it specification and system requirements. Proof of correctness

mandate by a formal technique ensures that specifications created using the techniques are verifiably correct.

Executability: The specification technique for specifying e-commerce applications should have the capability of generating testable executable code, which can be simulated and easily translated into the system implementation.

3.2.3 Secondary Requirements

This secondary requirements are briefly described below.

Flexibility: The specification technique for specifying e-commerce applications should be flexible to use by providing simple alternative constructs to most of the constructs.

User friendly: The specification technique for specifying e-commerce applications should be relatively easy to learn and use by an average e-commerce application developer. It should have appropriate tool support like good editors, type checkers, and true type fonts among others, which will enhance its user friendliness. This (tool) support will go a long way in increasing the usage of formal specification techniques by industrial e-commerce practitioners.

Efficiency: The specification technique for specifying e-commerce applications should produce efficient specifications and not obviously verbose specifications.

Compatibility: The specification technique for specifying e-commerce applications should be compatible with past and existing software engineering practices.

3.3 Taxonomy of Specification Techniques for E-commerce Systems

According to Fettke and Loos [37], a taxonomy establishes relationship between classification schemes (set of attributes necessary to classify objects of an application area, which depends on a classification principle — states the conceptual structure of the classification scheme) and classification objects. Taxonomies generally enhance the clarity and understanding of classified objects.

Existing taxonomies of specification techniques [106, 107] do not specifically address the e-commerce domain and they do not take into cognizance emerging specification techniques amenable to specifying e-commerce applications. The taxonomy for e-commerce specification techniques provided in this thesis builds on these existing taxonomies [106, 107] and takes into cognizance emerging general specification techniques and specification techniques developed specifically for the e-commerce domain. The need for a taxonomy for specification techniques applicable to specifying e-commerce systems is underscored by:

i. The need to reveal categories of not fully exploited and explored specification techniques for e-commerce systems,

ii. To provide a much deeper clarity and understanding of the specification techniques for e-commerce systems,

iii. To provide a broad overview of specification techniques for e-commerce systems and their relationships, and

iv. The existence in the literature of taxonomies for various aspects of e-commerce such as reference models [67], risks and failures [103], and business models [8, 53].

The taxonomy for e-commerce specification techniques (see Figure 3.2) introduces

domain-oriented and integrated specification methods and inherits model-based, algebraic, graphical, executable, and informal specification methods.

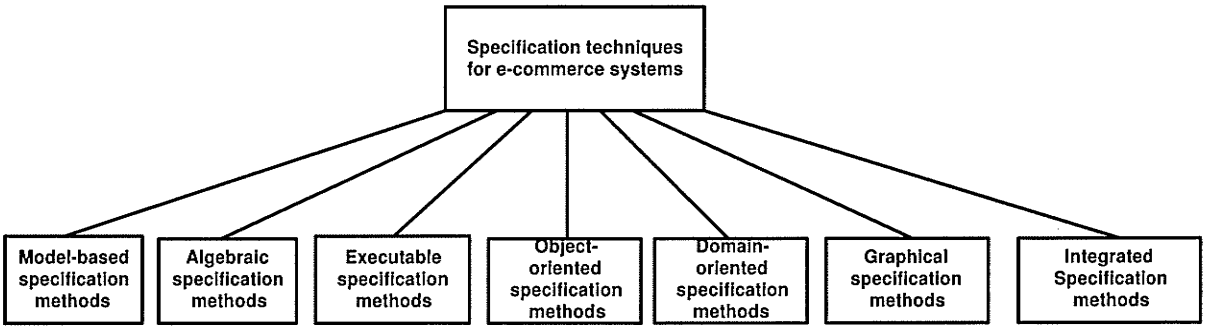

Figure 3.2: Taxonomy of E-commerce Specification Techniques

3.3.1 Domain-oriented Specification Methods

These are specification techniques (formal or otherwise) that have been designed, currently being designed, and those that will be designed to specifically model requirements of e-commerce applications. Some specification techniques in this category include Formal-CAFE [77] and UML-CAFE [93, 95]. Although specification techniques in this domain are aimed solely at e-commerce transactions, the existing techniques in this category do not have all the requirements (developed in this research) that are necessary to capture all the components of e-commerce systems. For instance, Formal-CAFE does not have constructs for describing real-time constraints and non-functional requirements such as user interfaces and usability. Also, the Formal-CAFE methodology which is based on model checking is not easy and flexible to use. UML-CAFE (an integration of UML, Formal-CAFE and model checking design patterns [94]) is a promising specification technique for describing most components of an e-commerce system. However, its current implementation does not have constructs for handling the lower level abstract functionalities of system behaviour.

3.3.2 Integrated Specification Methods

The specification techniques in this category integrate (combine) more than one specification technique into a single specification technique and introduce concepts such as time, concurrency, parallelism, non-determinism, real-time constraints, and synchronization into an existing technique. These techniques emerge in the literature during the advent of the object-orientation, distributed systems, dynamic, parallel, data synchronization, and concurrency paradigm. Since e-commerce applications exhibit most of these attributes, the techniques classified in this category are amenable to specifying e-commerce transactions. Examples of integrated specification methods include: CSP-OZ [38] (an integration of CSP [13, 50] and Object-Z [18, 28]) and CSP2B [16] (an integration of CSP and B-method [1]), among others. Examples of specification techniques that introduce concepts such as time, concurrency, parallelism, non-determinism, real-time constraints, and synchronization into an existing technique include Real-Time Object Z [78, 88, 89], Timed Communicating Object Z (TCOZ) [64], Timed CSP [86], Concurrent Z [35], Real-Time Z [14, 98], among others.

3.3.3 Model-based Specification Methods

Model-based formal specification methods specify a system in terms of state models, using mathematical primitives such as sets, functions, and sequences [91]. State changes are easy to define and the specification of operations on states may be grouped with corresponding state definitions. Also, the formal specification for each operation can be grouped together. Model-based formal specification methods are rich in mathematical notations. Examples of model-based formal specification methods are Z, Vienna Development Method (VDM) [10], RAISE, CSP, PVS, and Model checkers. CSP and Model checkers are used for modelling concurrent systems, while Z and VDM are used in specifying sequential systems.

3.3.4 Algebraic Specification Methods

In algebraic specification methods, objects are formally specified in terms of relationships between operations that act on objects. Algebraic specification is mostly used in interface specification, where large systems are decomposed into subsystems with well-defined interfaces between the subsystems. The specification of subsystem interfaces allows independent development of different subsystems [91]. Some algebraic specification methods include LARCH [46], OBJ [43], and LOTOS [12]. LOTOS is used for specifying concurrent systems, while LARCH and OBJ are used for specifying sequential systems. A drawback of algebraic specification techniques is the difficulty in proving mathematically the completeness and consistency of algebraic specifications.

3.3.5 Executable Specification Methods

According to Fuchs [41], executable specification methods apply logic-based techniques to model functional requirements of a system's behaviour. Executable specifications form the basis for implementations and do not constitute implementations. A major attribute of executable specification methods is the capability to translate specification into source codes. Executable specification tools are used to derive executable specifications of system behaviour [29]. Some executable specification methods include Maude [24] and NP-SPEC [17]. The drawbacks of executable specification techniques is their deficiency in constructs for modelling non-functional requirements (such as real-time constraints and user interfaces) of system behaviour.

3.3.6 Graphical Specification Methods

Graphical specification methods use graphical structures such as diagrams and graphs to model system behaviour. The techniques in this category are sometimes called visual modelling techniques. Graphical representations are useful in helping to visualize structures, composition, and relationships between elements [90]; and for modelling

user interfaces and external interactions. Graphical specification methods are also useful in modelling the real-time, concurrency, and distributed behaviours as well as dynamic views of a system. Some graphical specification methods are UML, Statecharts, Modecharts (an extension of Statecharts, redefined in real time logic to provide timing for events, thereby providing constructs for specifying real-time systems), and Petri nets. Graphical specification techniques can be used to specify user interfaces of e-commerce applications. Some graphical specification techniques are rich in structures, which can be used to capture the distributed, real-time, and concurrent behaviour of e-commerce systems. Some advantages of these methods include: enhancing ease of understanding by users, and making visualization of a system and its components easier. A major drawback of graphical specification techniques is their deficiency in constructs for handling functional requirements of system behaviour.

Chapter 4

Specifications

This chapter presents the pseudocode of the *ordering process* and the specifications of the requirements in the pseudocode using five of the specification techniques for evaluation.

4.1 Pseudocode for Ordering Process

The *ordering process* encompasses the stages involved in making and processing an order in a typical B2C e-commerce scenario. In making an order, a user browses an e-commerce enabled website, searches and finds items to purchase, negotiates the price of items, adds items to a shopping cart, checkout the items (i.e., purchases the items), and makes payment for items purchased. While processing an order, the e-commerce system verifies payments provided by users', updates the inventory, and plans logistics for shipping items to the user.

The following pseudocode describes the steps involved in the *ordering process* of a typical B2C e-commerce transaction. The pseudocode serves as the basis of the running sample requirements (functional) specifications specified using the selected specification techniques. For the sake of clarity, much detail is avoided and limits the options possible in certain circumstances. For instance, several modes of payment might be possible but have chosen to be silent on these. However, the processing required to fulfill an order, in

general, follows the pseudocode specified below.

Pseudocode : *Ordering Process*

```
BEGIN {OrderingPro};
    Item(s) to buy
    BEGIN {item search}
    REPEAT
        User searches for item in an e-commerce system
            IF item is found THEN
                {
                View item details
                Negotiate/check price of item
                        IF price is satisfactory to user THEN
                            Put item in shopping cart
                }
    UNTIL {user does not want more items to buy}
    END {item search}
    Edit shopping cart
    IF shopping cart is empty THEN
        exit e-commerce system
    ELSE
    {
        Checkout cart
        Logs in to secure part of e-commerce site
        Makes payment
            IF payment is accepted THEN
            {
                Process Order
                Update Database and Inventory
                Send/deliver item(s) to user
            }
            ELSE
            {
                Notify user of problem
                Request other payment mechanism
            }
    }
END{OrderingPro}
```

4.2 Specifications for the Ordering Process

This thesis adopts a pragmatic and systematic approach in the evaluation of specification techniques for e-commerce systems. To underscore this pragmatic and systematic approach as well as lend more credence to this evaluation, five of the major specifi-

cation techniques for evaluation were used to specify the requirements (stated in the pseudocode) of the *ordering process* in a typical B2C e-commerce scenario.

The subsequent subsections present some fundamentals of each of the specification techniques for evaluation and their respective specifications of the *ordering process*.

4.2.1 Z Specifications

The Z notation, as described earlier in Section 2.2.2, is based on first-order predicate logic and set theory. This thesis adopts a combination of Z and informal descriptions instead of Real-Time Z and Concurrent Z in the specification because tools for proving the correctness of specifications composed in Real-Time Z and Concurrent Z are not widely available. The Z specification of the ordering process described in the pseudocode is presented below.

In addition to the basic types available in Z, the following types are used in the specification:

$$[CHAR, STRING, PHONENO, BOOLEAN, DIGIT]$$

Table 3 shows some abbreviations used in the specification and their respective meanings.

Abbreviation	Meaning
PM	Payment Mechanism
itemtotcost	Total cost of Items purchased
upwd	User's Password
itemdesc	Description of item
usadr	User's Contact Address
itemid	Identification of Item
itemname	Name/description of Item
itemuprice	Unit Price of Item
itemqty	Quantity of Items purchased
usn	User's Username
nou	User's Real Name

Table 3: Abbreviations Used and Their Meanings.

Also, the following types used in the specification are defined. I represent monetary values with the variable REAL. Since Z lacks the primitive real type, integer values are

used for REAL in this specification. However, an implementation of the specification should use the appropriate variable type to represent monetary values so as to eradicate round-off errors that may occur during system testing.

$$REAL == \mathbb{N}$$
$$DIGIT ::= 0..9$$
$$PHONENO ::= \text{seq}\ DIGIT$$
$$BOOLEAN ::= true \mid false$$

The types of acceptable payments are defined by PM.

$$PM ::= Credit\_Card \mid Debit\_Card \mid Cheque \mid Others$$

Other user-defined types, which are used in the specification are given below.

$$EditCartStatus ::= empty \mid not\_empty$$
$$PaymentStatus ::= accept \mid decline$$

The structure of some objects that participate in the *ordering process* in a typical B2C e-commerce scenario and the constraints on the objects are given below.

The major acceptable mode of payment allowed by the system is credit card. The schema *CrCard* defines a major credit card such as VISA, MASTERCARD, etc or a merchant's credit card.

CrCard

$$CrCardNo, CrCardType, CrCardIsDt, CrCardExDt : STRING$$
$$CrCardBal, CrCardLim : REAL$$
$$CrCardCrncy : \text{seq}\ CHAR$$

$$CrCardBal \leq CrCardLim$$

The uniqueness constraint holds for the credit cards, i.e. no two credit cards should have the same number.

$$\forall a, b : CrCard \bullet a \neq b \Rightarrow a.CrCardNo \neq b.CrCardNo$$

Initially, a *User*, which will search for an item and the constraints on the *User* object are described in the schema below. The definition of a user encompasses users that search for an item, make order and ultimately purchase the item as well as casual users that just search for item(s) with no intention of buying the item(s).

User

$$usn, usadr, upwd, emailadr : \text{seq } CHAR$$
$$phone, fax : PHONENO$$
$$CrCrdNo, CrCardType, CrCardExDt : STRING$$
$$StatusFlag : BOOLEAN$$
$$paymode : PM$$

$$\#usn \geq 6 \land \#usn \leq 8$$
$$\#upwd \geq 6 \land \#upwd \leq 8$$
$$usn \neq upwd$$
$$\#phone \geq 7 \land \#fax \geq 7$$

In the schema *User*, a user's username cannot be the same as the user's password. Also, the number of characters in a user's username and password must have a minimum of six characters and a maximum of eight characters.

The following additional constraints on the *User* schema are defined: every user making an order is unique, and has a unique user's password.

$$\forall x, y : User \bullet x \neq y \Rightarrow x.usn \neq y.usn \land x.upwd \neq y.upwd$$

These constraints are my design decisions. All of them may not necessarily be applicable in general, e.g., the requirement that user passwords be unique and the number of characters in their username and password.

New users must be created in the system, before they can make use of the system. The new users must belong a group of valid system users. The schema *NUser* describes these requirements.

NUser

$$UserGroup, UserGroup' : \mathbb{P}\ User$$
$$NUser? : User$$
$$usn, upwd : \text{seq } CHAR$$

$$NUser? \notin UserGroup$$
$$(\forall a : UserGroup \bullet a.usn \neq NUser?.usn$$
$$\land\ a.upwd \neq NUser?.upwd$$
$$\land\ UserGroup' = UserGroup \cup \{NUser?\})$$
$$NUser? \in UserGroup'$$

The new user is added to the updated group of valid system users.

The schema *ActiveUser* describes a user who is online i.e. currently active in the system.

ActiveUser

$$ActiveU : User$$
$$UserGroup : \mathbb{P}\ User$$
$$StatusFlag : BOOLEAN$$

$$ActiveU.StatusFlag = true$$

Some operations on the *User* schema includes revoking users' access to the system by removing a user from the group of valid users. The schema *RUser* captures this scenario.

RUser

$$UserGroup, UserGroup' : \mathbb{P}\ User$$
$$RUser? : User$$
$$usn, upwd : \text{seq}\ CHAR$$

$$RUser? \in UserGroup$$
$$(\exists_1\ b : UserGroup \bullet b.usn \neq RUser?.usn$$
$$\land\ a.upwd \neq NUser?.upwd$$
$$\land\ UserGroup' = UserGroup \setminus \{RUser?\})$$
$$RUser? \notin UserGroup'$$

The removed user is deleted from the updated group of system users.

The schema *Item* describes an item a user can search for in an e-commerce system. Each item belongs to a specific group, this grouping allows items to be searched for displayed using their various groups.

Item

$$itemid : \text{seq}\ CHAR$$
$$itemname, itemdesc, itemgroup : STRING$$
$$itemuprice, itemtotcost, itemcprice, itemsprice : REAL$$
$$itemqty : \mathbb{N}_1$$

$$\#itemname > 0$$
$$\#itemid > 0$$
$$\#itemgroup > 0$$
$$itemcprice > 0 \land itemsprice > 0$$
$$itemcprice \leq itemsprice$$

Each item must be distinct. This constraint is captured by:

$$\forall x, y : Item \bullet x \neq y \Rightarrow x.itemid \neq y.itemid$$

The variable *Itemdb* is the item database, which consists of items available to users at any point in time. Its signature is given as:

$$Itemdb : \mathbb{P}\, Item$$

Each item in the database must be unique. This constraint on *Itemdb* is given by:

$$\forall w : Itemdb \bullet (\forall x, y : Item \mid x \in w \land y \in w \bullet x \neq y \Rightarrow x.itemid \neq y.itemid)$$

Some operations on the *Item* objects include adding an item to the database (captured by the schema *AItem*), removing an item from the database (captured by the schema *RItem*), and updating item information in the database by modifying some attributes of an item (captured by the schema *UItem*). The schemas for these operations are given below.

AItem

$$items, items' : \mathbb{P}(Item \times \mathbb{N}_1)$$
$$newitem? : Item$$

$$newitem? \notin items$$
$$(\forall c : items \bullet c.itemid \neq newitem?.itemid$$
$$\land\ items' = items \cup \{newitem?\})$$
$$newitem? \in items'$$

The newly added item is included in the database.

RItem

$$items, items' : \mathbb{P}(Item \times \mathbb{N}_1)$$
$$ritem? : Item$$

$$ritem? \in items$$
$$(\exists_1 c : items \bullet c.itemid \neq ritem?.itemid$$
$$\land\ items' = items \setminus \{ritem?\})$$
$$ritem? \notin items'$$

The removed item is deleted from the database.

UItem

$$items, items' : \mathbb{P}(Item \times \mathbb{N}_1)$$
$$newitem? : Item$$

$$newitem? \in items$$
$$(\exists_1 d : items \bullet d.itemid \neq newitem?.itemid$$
$$\land\ items' = items \setminus \{d\} \cup \{newitem?\})$$

The item information is updated. The schema *Totcost* computes the total cost of item(s) purchased.

Totcost

$$
\begin{array}{l}
items : \mathbb{P}(Item \times \mathbb{N}_1) \\
itemtotcost!, itemsprice : REAL \\
itemtqy : \mathbb{N}_1;\ itemid : STRING \\
\hline
\exists\, a, item? : items \bullet \\
a.itemid = item?.itemid \land \\
itemtotcost! = item?.itemqty * item?.itemsprice
\end{array}
$$

Each item belongs to a a virtual or phyical store where they items are kept before they are sold. The schema *Store* defines a store in the e-commerce system.

Store

$$
\begin{array}{l}
storeid : \text{seq}\ CHAR \\
storename : STRING \\
users : \mathbb{P}\ User \\
items : \mathbb{P}(Item \times \mathbb{N}_1) \\
\hline
\#storeid > 0 \\
\#storename > 0
\end{array}
$$

Each store in the system must not be the same. The uniqueness constraint is given by:

$$\forall\, a, b : Store \bullet a \in Store \land b \in Store \Rightarrow a.storeid \neq b.storeid$$

Initially, the Store is empty i.e., does not contain any item. The schema *InitStore* describes this requirement.

InitStore

$$
\begin{array}{l}
\Delta InitStore \\
\hline
\#items = \emptyset
\end{array}
$$

An electronic shopping cart keeps track of all the items a user wants to buy. The schema *ShoppingCart* defines a shopping cart in the e-commerce system.

ShoppingCart

$$
\begin{array}{l}
cartid : \text{seq}\ CHAR \\
carts : \mathbb{P}\ ShoppingCart \\
user : User \\
items : \mathbb{P}(Item \times \mathbb{N}_1) \\
time : REAL
\end{array}
$$

Initially, the shopping carty is empty i.e., does not contain any item. The schema *Init-ShoppingCart* describes this requirement.

InitShoppingCart

$$\Delta InitShoppingCart$$

$$\#items = \emptyset$$

Another constraint on the electronic shopping cart is that all carts must be unique. The uniqueness constraint is given by:

$$\forall x, y : carts \bullet x \in ShoppingCart \land y \in ShoppingCart \Rightarrow x.cartid \neq y.cartid$$

Some operations on the shopping cart involves a user adding or removing an item to a shopping cart, and the system displaying the contents of the shopping cart. The schema *DisplayCart* displays the content of a user's shopping cart.

DisplayCart

$$ItemGroup, CartGroup : \mathbb{P}\, Item$$

$$CartGroup \subseteq ItemGroup \land$$
$$\forall a : CartGroup? \bullet b.ItemGroup \bullet$$
$$b.itemid = a.itemid$$

A user may add or put an item to the electronic shopping cart. The schema *PutToCart* captures this operation.

PutToCart

$$User;\ Item;\ ShoppingCart$$
$$items, items' : \mathbb{P}(Item \times \mathbb{N}_1)$$
$$usn?, itemid, itemid? : \text{seq}\ CHAR$$
$$itemqty, itemqty? : \mathbb{N}_1$$

$$(\exists a : ShoppingCart;\ b : Item \mid$$
$$d \in carts \land b.user.usn = usn? \land$$
$$b.itemid = itemid? \bullet$$
$$d.items' = d.items \cup \{(b, itemqty?)\} \lor$$
$$(\exists c : \mathbb{N}_1 \mid (b, c) \in d.items \bullet$$
$$d.items' = d.items \setminus \{(b, c)\} \cup \{(b, c + itemqty?)\}))$$

Also, a user might decide to remove an item initially added to a shopping cart. The schema *RFCart* describes this operation.

RFCart

$$User;\ ShoppingCart$$
$$items, items' : \mathbb{P}(Item \times \mathbb{N}_1)$$
$$usn?, itemid, itemid? : \text{seq } CHAR$$
$$itemqty, itemqty? : \mathbb{N}_1$$

$$(\exists\, a : ShoppingCart;\ b : Item;\ c : \mathbb{N}_1 \mid$$
$$d \in carts \land (b, c) \in d.items \land$$
$$b.user.usn = usn? \land b.itemid = itemid? \bullet$$
$$d.items' = d.items \setminus \{(b, c)\} \cup \{b, c - itemqty?)\})$$

The schema below describes an *Order* made by a user.

Order

$$orderid : \text{seq } CHAR$$
$$paymode : PM$$
$$orderitems : \mathbb{P}\, Item$$
$$status : STRING$$

$$\#orderid > 0$$

Orders made by the same or different users must be distinct. This constraint is given by:

$$\forall\, x, y : Order \bullet x \neq y \Rightarrow x.orderid \neq y.orderid$$

A *UserTransaction* represents an activity to be carried out by one user. There can be different user transactions at any point in time, since different users could order different or identical items simultaneously.

UserTransaction

$$user : User$$
$$order : Order$$
$$activity : STRING$$

It is necessary to link a *UserTransaction* and the associated *Order* with the appropriate user. The constraint below captures this requirement.

$$\forall\, a : Order;\ b : UserTransaction;\ c : User \mid b.order = a \bullet b.user = c$$

Some schema definitions and axiomatic definitions used subsequently in the specifications are presented below.

Users login to the secure part of the e-commerce system by providing their username and password before they can access the system. The schema definition *UserLogIn* describes the requirement.

UserLogIn

$$
\begin{array}{l}
LoginUser? : User \\
UserGroup, ActiveUserGroup, ActiveUserGroup' : \mathbb{P}\ User \\
usn : \text{seq}\ CHAR \\
StatusFlag : BOOLEAN \\
\hline
\exists_1\ a : UserGroup \bullet a.usn = LoginUser?.usn \\
a.upwd = LoginUser?.upwd \\
LoginUser?.StatusFlag = true \\
ActiveUserGroup' = ActiveUserGroup \cup \{a\}
\end{array}
$$

Users log out of the secure part of the e-commerce system after using the system. The schema definition *UserLogOut* describes the requirement.

UserLogOut

$$
\begin{array}{l}
LogoutUser? : ActiveUser \\
UserGroup, ActiveUserGroup, ActiveUserGroup' : \mathbb{P}\ Userusn : \text{seq}\ CHAR \\
StatusFlag : BOOLEAN \\
\hline
\exists_1\ b : UserGroup \bullet b.usn = LogoutUser?.usn \\
b.StatusFlag = false \\
ActiveUserGroup' = ActiveUserGroup \setminus \{b\}
\end{array}
$$

The axiomatic definition *DecreaseInv* updates the quantity of items in inventory by removing the quantity of item(s) purchased.

$$
\begin{array}{l}
DecreaseInv : \text{seq}\ CHAR \times \mathbb{N}_1 \rightarrow \mathbb{N} \\
\hline
\exists\ usn : \text{seq}\ CHAR \\
itemqty, quantity, quantity' : \mathbb{N}_1 \\
a, b : Item;\ \forall\ x : \mathbb{N} \bullet \\
DecreaseInv(usn, itemqty) = x \Leftrightarrow \\
a.itemid = usn \land a.itemid = b.itemid \land \\
a.quantity' = a.quantity - itemqty
\end{array}
$$

The axiomatic definition *IncrementInv* updates the quantity of items in inventory when a user cancels an order by adding the initial quantity of item(s) purchased to the quantity of items in inventory.

$$
\begin{array}{|l}
IncrementInv : \text{seq}\ CHAR \times \mathbb{N}_1 \rightarrow \mathbb{N}_1 \\
\hline
\exists\, usn : \text{seq}\ CHAR \\
itemqty, quantity, quantity' : \mathbb{N} \\
x, y : Item;\ \forall\, a : \mathbb{N}_1 \bullet \\
IncrementInv(usn, itemqty) = a \Leftrightarrow \\
x.itemid = usn \land x.itemid = y.itemid \land \\
x.quantity' = x.quantity + itemqty
\end{array}
$$

The axiomatic definition *Valbal* checks if a user has sufficient money in their credit card to defray the cost of item(s) purchased.

$$
\begin{array}{|l}
Valbal : \text{seq}\ CHAR \times REAL \rightarrow BOOLEAN \\
\hline
\exists\, usn : \text{seq}\ CHAR;\ itemtotcost : REAL \bullet \\
valbal(usn, itemtotcost) = true \Leftrightarrow \\
(\exists\, a : CrCard \mid a.CrCardNo = usn \bullet \\
a.CrCardBal + itemtotcost \leq a.CrCardLim)
\end{array}
$$

The axiomatic definition *Drcrcard* debits a user's credit card by subtracting the total cost of item(s) purchased from the credit card balance.

$$
\begin{array}{|l}
Drcrcard : \text{seq}\ CHAR \times REAL \rightarrow REAL \\
\hline
\exists\, a, b : CrCard \\
itemtocost : REAL \\
usn : \text{seq}\ CHAR;\ \forall\, x : REAL \bullet \\
Drcrcard(usn, itemtotcost) = x \Leftrightarrow \\
a.CrCardNo = usn \land \\
a.CrCardNo = b.CrCardNo \land \\
a.CrCardBal = b.CrCardBal - itemtotcost
\end{array}
$$

The axiomatic definition *Crcrcard* credits a user's credit card by refunding the initial total cost of item(s) purchased to the credit card balance.

$$
\begin{array}{|l}
Crcrcard : \text{seq}\ CHAR \times REAL \rightarrow REAL \\
\hline
\exists\, p, q : CrCard \\
itemtocost : REAL \\
usn : \text{seq}\ CHAR;\ \forall\, a : REAL \bullet \\
Crcrcard(usn, itemtotcost) = a \Leftrightarrow \\
p.CrCardNo = usn \land \\
p.CrCardNo = q.CrCardNo \land \\
p.CrCardBal = q.CrCardBal + itemtotcost
\end{array}
$$

The axiomatic definition *Valusr* checks if the user of the e-commerce system is a valid system user. A valid system user has a correct username, password, and is a member of a store.

$$
\begin{array}{|l}
Valusr : \text{seq}\ CHAR \times \text{seq}\ CHAR \times \text{seq}\ CHAR \rightarrow BOOLEAN \\
\hline
stores : \mathbb{P}(Store \times \mathbb{N}_1) \\
\forall\, a : BOOLEAN \\
usn, upwd, storeid : \text{seq}\ CHAR \\
b : User;\ c : Store \bullet a = true \Leftrightarrow \\
valusr(usn, upwd, storeid) \Rightarrow \\
(c \in stores \land c.storeid = storeid) \land \\
(b \in c.users) \land (b.usn = usn) \land \\
b.upwd = upwd
\end{array}
$$

The axiomatic definition *Valitem* checks if the correct item(s) ordered by the user is the actual item offered for sale to the user.

$$
\begin{array}{|l}
Valitem : STRING \times STRING \rightarrow BOOLEAN \\
\hline
stores : \mathbb{P}(Store \times \mathbb{N}_1) \\
\forall\, a : BOOLEAN \\
itemid, storeid : STRING \\
b : Item;\ c : Store \bullet a = true \Leftrightarrow \\
valitem(itemid, storeid) \Rightarrow \\
(d \in stores \land d.storeid = storeid) \land \\
(b \in c.Itemdb) \land (b.itemid = itemid)
\end{array}
$$

The axiomatic definition below *CheckItemPrice* checks if the price of an item and the conditions under which the price is acceptable or unacceptable to the user.

$$
\begin{array}{|l}
OK : \text{seq}\ CHAR; \\
CheckItemPrice : Item \rightarrow BOOLEAN \\
\hline
\forall\, a : Item \bullet \\
CheckItemPrice(a) = true \Leftrightarrow a.itemprice = OK \land \\
CheckItemPrice(a) = false \Leftrightarrow a.itemprice = \neg\ OK
\end{array}
$$

The axiomatic definition *SearchMoreItem* searches for other item(s) in the e-commerce system, when the initial item sought by the user is not available.

$$SearchMoreItem : \text{seq}\ CHAR \times STRING \rightarrow \text{seq}\ CHAR$$

$$\begin{aligned}
&\forall\, b, itemid? : \text{seq}\ CHAR;\ itemname? : STRING \\
&c : Store \bullet b \Leftrightarrow \\
&SearchMoreItem(itemid, itemname) \Rightarrow \\
&\exists\, a, p : Item \mid a \in items \wedge p \in items \bullet \\
&a.itemid \neq p.itemid? \wedge a.itemname \neq p.itemname?
\end{aligned}$$

The axiomatic definition *EditShoppingCart* returns a — (EditCartStatus) of a shopping cart after a user edits the content of the shopping cart.

$$EditShoppingCart : \mathbb{N} \rightarrow EditCartStatus$$

Table 4a presents a summary of the informal semantics of functions used in the specifications.

Function Name	**Informal Description**
UserExit(c)	User with username c exit the e-commerce system.
CheckOutCart(b)	User with username b checkout shopping cart.
MakePayment(b, c)	User with username b makes payment with payment mechanism c.
NotifyUser(b)	Notify user with username b (e.g., of payment problem).
AskOtherPayment(b)	Ask user with username b to provide other payment mechanisms.
SendItem(a, b)	Send/deliver item a to user with username b.

Table 4a: Informal Semantics of Functions Used in the Specification.

Next, the specifications of some operations performed by a user in the *ordering process* in a typical B2C e-commerce scenario are given. First, a user searches for a specific item by either its name, description, or id, in the e-commerce system. There are two possible scenario when a user searches for an item in an e-commerce system: the user may find the item or not find the item. The schema *FindItem* captures this requirement.

FindItem

$$
\begin{array}{l}
Itemdb \\
found, name?, itemdesc?, outcome!, itemname : STRING \\
itemid? : \text{seq } CHAR \\
\hline
\forall x : Item \bullet x \in Itemdb \land \\
\textbf{if } (x.itemname = name?) \lor \\
\qquad x.itemid = itemid? \lor \\
\qquad x.itemdesc = itemdesc? \\
\textbf{then } (outcome! = found) \\
\textbf{else } (outcome! = \neg\, found)
\end{array}
$$

The schema *SearchItem* is an operation that uses the name and price of an item to search for the item in the database.

SearchItem

$$
\begin{array}{l}
items : \mathbb{P}(Item \times \mathbb{N}_1) \\
Store \\
itemname?, itemdesc?, itemgroup? : STRING \\
itemid?, storeid? : \text{seq } CHAR \\
itemsprice? : REAL \\
\hline
\exists a : Item \mid a \in items \bullet \\
a.itemid = itemid? \land \\
a.itemname = itemname? \land \\
a.itemsprice = itemsprice? \land \\
a.itemdesc = itemdesc? \land \\
a.itemgroup = itemgroup? \land \\
a.storeid = storeid?
\end{array}
$$

When a user finds the item searched for, the user views the item's description. However, if the item searched for is not found, the user may search for more item(s) in the current e-commerce system. The schema *AfterFindItem* captures this requirement.

AfterFindItem

$$
\begin{array}{l}
name?, desc?, itemname, itemdesc : STRING \\
outcome!, found : STRING \\
itemsprice, price? : REAL \\
\hline
\forall p : Item \bullet \\
\textbf{if } (outcome! = found) \\
\textbf{then } (\#name? > 0 \land \\
\quad (\exists x : Item \bullet x \in Itemdb \land \\
\quad x.itemname = name? \land x.itemdesc = desc? \\
\quad x.itemsprice = price?)) \\
\textbf{else } SearchMoreItem(p.itemid, p.itemname)
\end{array}
$$

After the user views the description of the item, the user checks the price of the item. The user puts the item into his/her shopping cart if the price of the item is acceptable. However, if the price of the item is unacceptable the user may search for more item(s) in the e-commerce system.

UserPutItemCart

$$
\begin{array}{l}
itemid : \text{seq } CHAR \\
itemname : STRING \\
items', items : \mathbb{P}(Item \times \mathbb{N}_1) \\
\hline
\forall a, p : Item \bullet \\
\textbf{if } (CheckItemPrice(a) = true) \\
\textbf{then } ((\exists a : ShoppingCart;\ b : Item \mid \\
\quad d \in carts \land b.user.usn = usn? \land \\
\quad b.itemid = itemid? \bullet \\
\quad d.items' = d.items \cup \{(b, itemqty?)\} \lor \\
\quad (\exists c : \mathbb{N}_1 \mid (b, c) \in d.items \bullet \\
\quad d.items' = d.items \setminus \{(b, c)\} \cup \{(b, c + itemqty?)\}))) \\
\textbf{else } SearchMoreItem(p.itemid, p.itemname) \land \\
\quad a.itemid \neq p.itemid
\end{array}
$$

After adding items to the shopping cart, the user can edit the shopping cart's contents. If the shopping cart is empty, after being edited by the user, the user logs out and exit the system. However, if there are items in the shopping cart, after being edited by the user, the user checksout the shopping cart, logs in to the secure part of the e-commerce system and makes payment. After checkout, the user can decide to cancel the order. The following schemas *AfterPutItemCart*, *MakePayment*, and *CancelOrder* capture the

scenario.

AfterPutItemCart

$$
\begin{array}{l}
LogoutUser? : ActiveUser;\ LoginUser? : User \\
UserGroup : \mathbb{P}\ User \\
usn, upwd : \text{seq}\ CHAR;\ StatusFlag : BOOLEAN \\
\hline
\forall\, c : Order \\
b : EditCartStatus \bullet EditShoppingCart(a) \land \\
\textbf{if}\ (b = empty) \\
\textbf{then}\ (\exists_1\, b : UserGroup \bullet \\
\qquad b.usn = LogoutUser?.usn \land \\
\qquad b.StatusFlag = false \land UserExit(usn)) \\
\textbf{else}\ (CheckOutCart(usn) \land \\
\qquad \exists_1\, a : UserGroup \bullet a.usn = LoginUser?.usn \land \\
\qquad a.upwd = LoginUser?.upwd \land \\
\qquad LoginUser?.StatusFlag = true)
\end{array}
$$

The *MakePayment* schema checks if a user is a valid user, confirms if there is sufficient balance in the credit card to defray the total cost of item(s) purchased, and debits the total cost of item(s) purchased from the user's credit card.

MakePayment

$$
\begin{array}{l}
usn?, upwd?, storeid? : \text{seq}\ CHAR \\
itemtotcost?, CrCardBal! : REAL \\
a! : BOOLEAN \\
\hline
a! = Valusr(usn?, upwd?, storeid?) \land \\
a! = Valbal(usn?, itemtotcost?) \land \\
CrCardBal! = Drcrcard(usn?, itemtotcost?)
\end{array}
$$

The *CancelOrder* schema cancels an order by crediting a user's credit card and adds the initial quantity of item(s) purchased to the item inventory.

CancelOrder

$$
\begin{array}{l}
usn? : \text{seq}\ CHAR \\
items, items' = \mathbb{P}(Item \times \mathbb{N}_1) \\
itemtotcost?, CrCardBal!, CrCardBal? : REAL \\
\hline
(\exists\, i, j : CrCard \mid i.CrCardNo = j.CrCardNo \bullet \\
j = CrcrCard(usn?, itemtotcost?) \land \\
(\exists\, x, y : Item \mid x.itemid = y.itemid \bullet \\
y = IncrementInv(usn?, itemtotcost?))
\end{array}
$$

After the user makes payment, the system checks if the balance left in the payment instrument is enough to cover the cost of item(s) purchased. If the payment is not satisfactory, the system notifies the user of the problem and requests for other payment mechanisms. However, if the payment is accepted, the system processes the order, updates the inventory, and the item(s) are sent to the user. The schema *AfterPayment* captures the described scenario.

AfterPayment

$$
\begin{array}{l}
usn? : \text{seq } CHAR \\
itemtotcost? : REAL \\
\hline
\forall c : Order \bullet \\
\textbf{if } (Valbal(usn?, itemtotcost?) = true) \\
\textbf{then } (\exists a, b : CrCard \mid a.CrCardNo = b.CrCardNo \bullet \\
\quad b = Drcrcard(usn?, itemtotcost?) \land \\
\quad (\forall p : Item \mid p \in c.orderitems \bullet \\
\quad (\exists x, y : Item \mid x.itemid = y.itemid \bullet \\
\quad y = DecreaseInv(usn?, itemtotcost?)) \land \\
\quad SendItem(usn)) \\
\textbf{else } (NotifyUser(usn) \land AskOtherPayment(usn))
\end{array}
$$

Lastly, the various specified modules are used to compose the *Ordering Process*. The *Ordering Process* can be specified by a sequential composition of six operations including *FindItem*, *AfterFindItem*, *UserPutItemCart*, *AfterPutItemCart*, *MakePayment* and *AfterPayment*. Hence, the *Ordering Process* is composed as follows:

$$
\begin{aligned}
OrderingProcess \mathrel{\widehat=} (&FindItem \fatsemi (AfterFindItem \land UserPutItemCart) \fatsemi \\
&(AfterPutItemCart \land MakePayment \land AfterPayment))
\end{aligned}
$$

The schemas, axiomatic definitions, and signatures of the ordering process specification composed in the Z notation were checked for syntax and type errors using the Z/EVES version 2.1 tool.

4.2.2 Petri nets Specifications

Petri nets, as described earlier in Section 2.2.1, are graphical but mathematical modelling tools that consist of places, transitions, and arcs to link the places with the transitions.

Places are use to model conditions, while transitions are use to model events. Input arcs are use to link places with transitions, while output arcs (originates at a transition and terminates at a place) are use to link transitions with places.

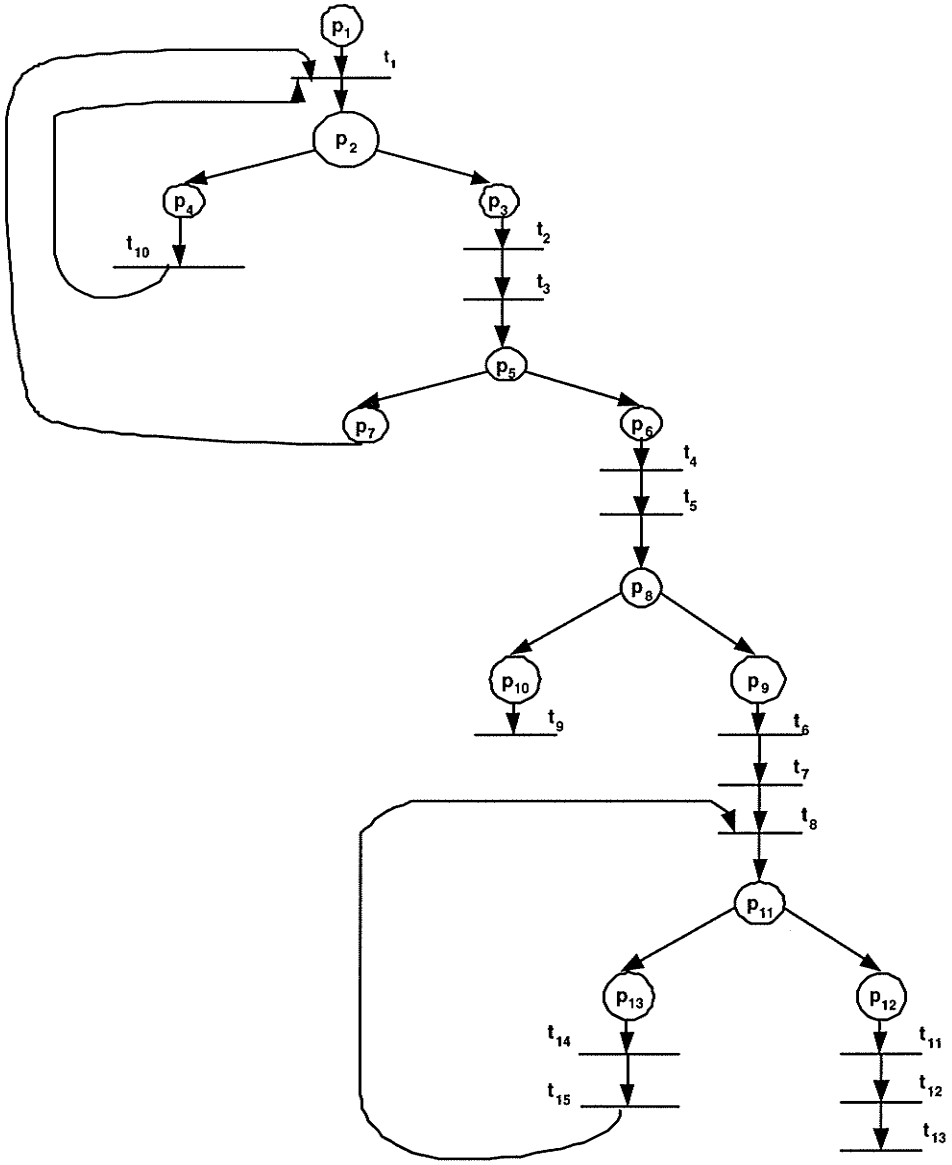

Figure 4.1: Petri nets Model of Ordering Process.

The expressive power of Petri net graphs makes Petri nets an obvious choice to model the *ordering process* in a typical B2C e-commerce scenario. Figure 4.1 shows a Petri nets representation of the *ordering process* described in the pseudocode in Section 4.1, while

Table 4b shows the meanings of the symbols used in the Petri nets representation.

Place Descriptions	Transition Descriptions
p_1: browse e-commerce site	t_1 : search for item in e-commerce site
p_2: is item available?	t_2 : views item description
p_3: item is found	t_3 : check/negotiate item price
p_4: item is not found	t_4 : put item to shopping cart user
p_5: is item price OK?	t_5 : edit shopping cart
p_6: item price is OK	t_6 : checkout cart
p_7: item price is not OK	t_7 : log in to secure e-commerce site
p_8: is shopping cart empty?	t_8 : make payment
p_9: shopping cart is empty	t_9 : exit e-commerce system
p_{10}: shopping cart is not empty	t_{10} : search for other items
p_{11}: is payment valid?	t_{11} : process order
p_{12}: payment is accepted	t_{12} : update database and inventory
p_{13}: payment is declined	t_{13} : send item to user
	t_{14} : notify user of payment problem
	t_{15} : request other payment mode

Table 4b: Places and Transitions Descriptions for Ordering Process.

4.2.3 Statecharts Specifications

Recall Statecharts, as described earlier in Section 2.2.3, is a graphical modelling tool based on the conventional state machine for capturing system behaviours using state-transition diagrams.

Initially, flat Statecharts models (see Figure 4.2a — 4.2k) are use to capture the various steps of the *ordering process* described in the pseudocode in Section 4.1. While, Figure 4.2l is a single flat Statecharts model, which shows all the events, actions, and conditions involved in making and processing an order in a typical B2C e-commerce scenario.

The Statecharts model in Figure 4.2a shows the steps involve in creating a valid user in the sample e-commerce system.

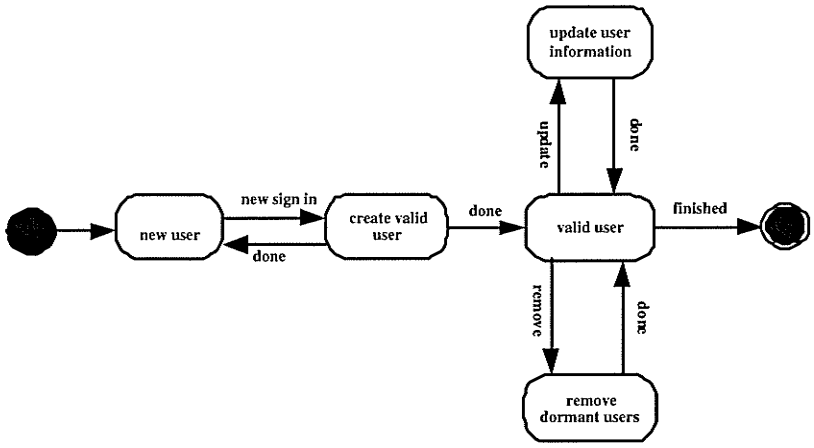

Figure 4.2a: Statecharts Model for Creating Valid Users.

The Statecharts model in Figure 4.2b shows the various operations carried out by the *user*, the some operations carried out on the *user*.

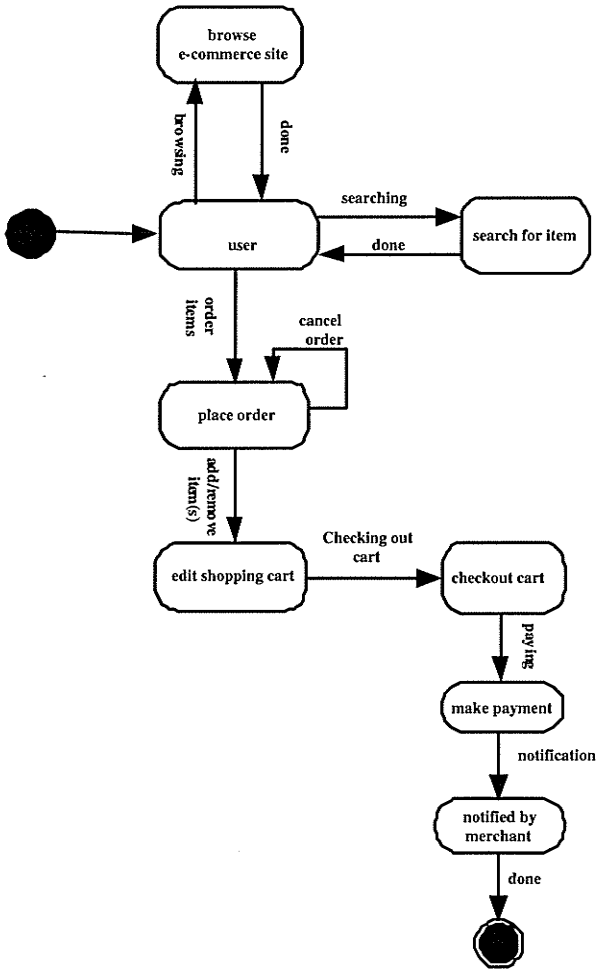

Figure 4.2b: Statecharts Model for User Operations.

Figure 4.2c shows the Statecharts model of the steps involve in searching for an item in the e-commerce system.

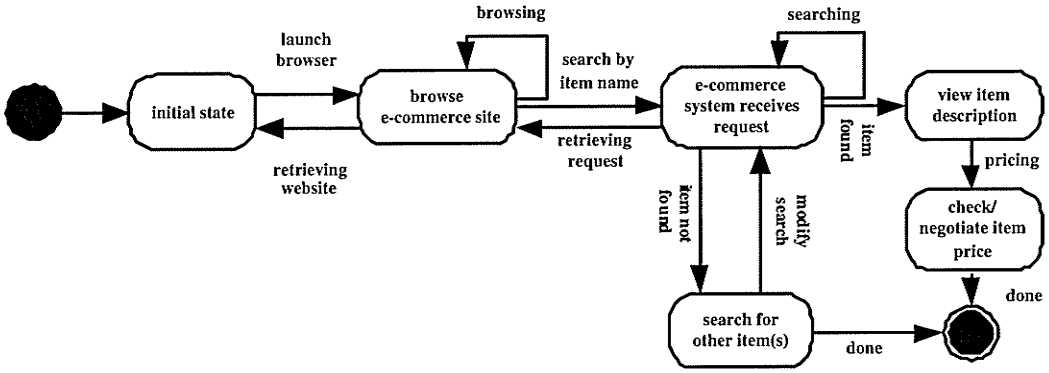

Figure 4.2c: Statecharts Model for Item Search.

The Statecharts model in Figure 4.2d shows the various operations carried out on an *Item* in the e-commerce system.

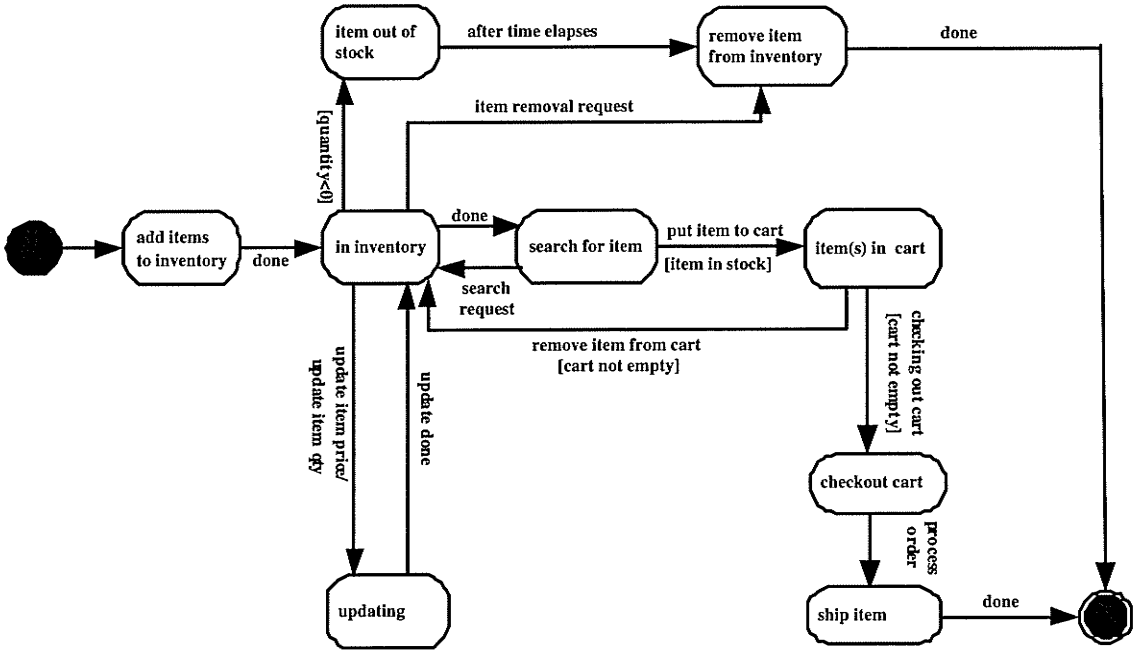

Figure 4.2d: Statecharts Model for Operations on Item.

Figure 4.2e shows a substate in Figure 4.2d, which involves checking/negotiating the price of an item in the sample e-commerce system.

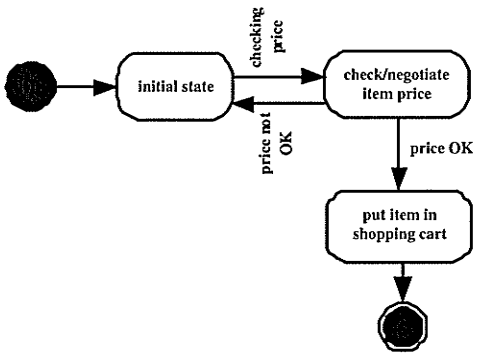

Figure 4.2e: Statecharts Model for Check Item Price.

Figure 4.2f is the Statecharts model, which shows the operations carried out on the shopping cart in a typical B2C e-commerce setting.

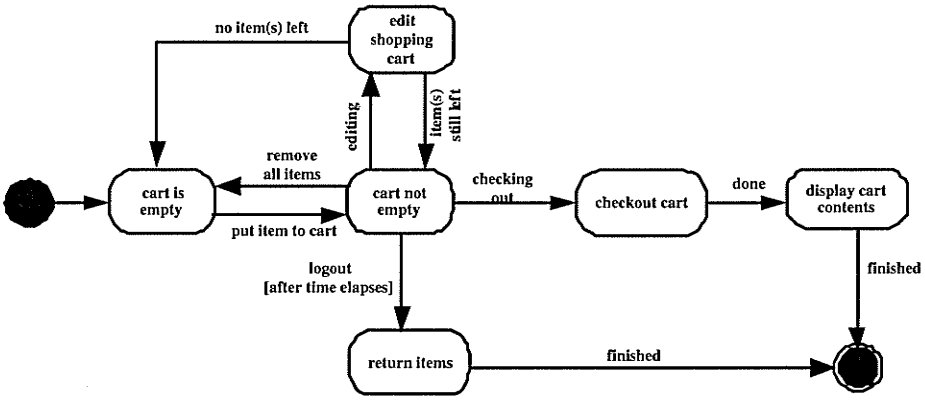

Figure 4.2f: Statecharts Model for Shopping Cart Operations.

Figure 4.2g shows the Statecharts model showing a refinement of the *edit shopping cart* substate in Figure 4.2f.

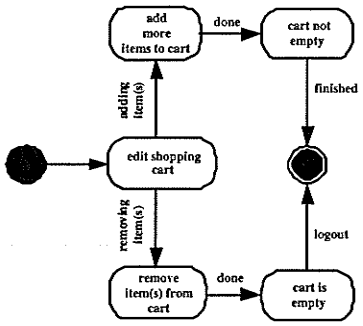

Figure 4.2g: Statecharts Model for Edit Shopping Cart Substate.

Figure 4.2h shows the various operations on the inventory, while Figure 4.2i shows a refinement of the *update inventory* substate 4.2h.

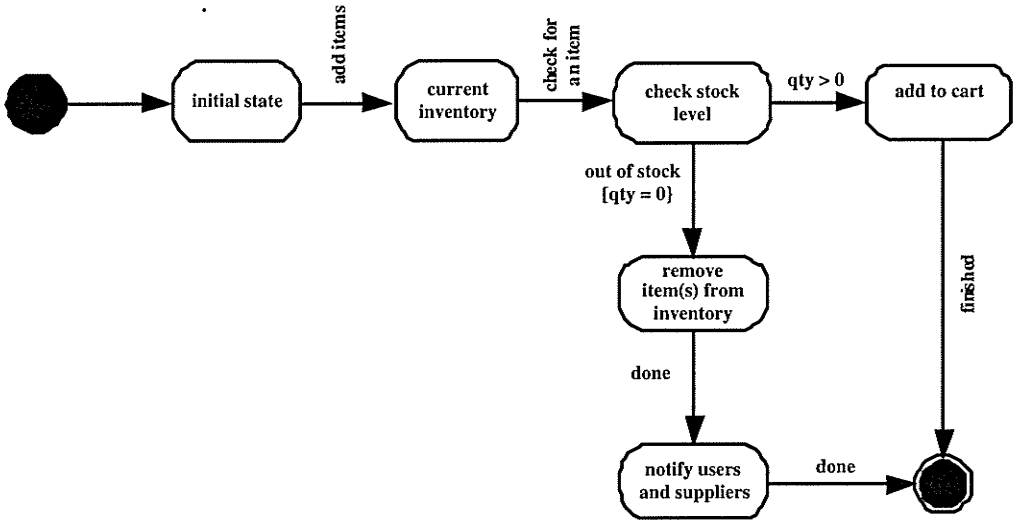

Figure 4.2h: Statecharts Model for Inventory Operations.

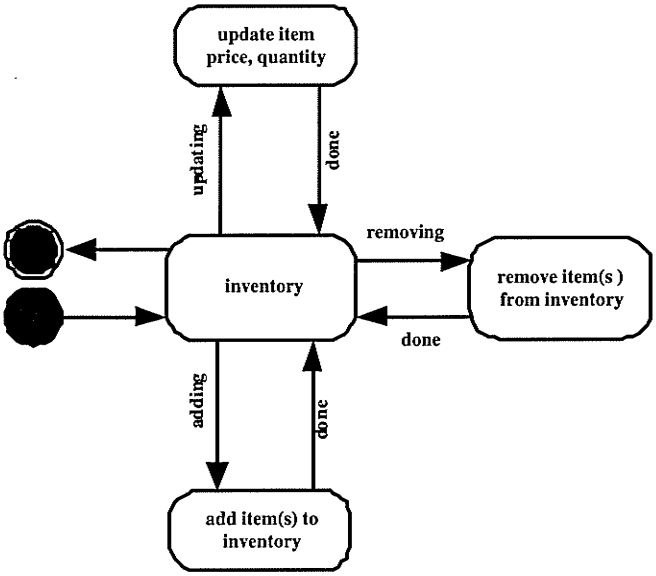

Figure 4.2i: Statecharts Model for Update Inventory Substate. The Statecharts model depicted in Figure 4.2j shows the steps involve in making payment by a user in the sample e-commerce system.

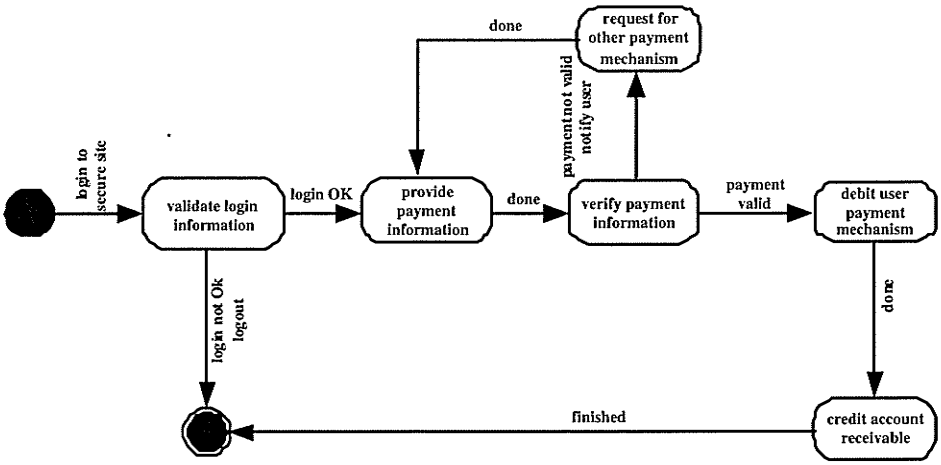

Figure 4.2j: Statecharts Model for Make Payment.

Figure 4.2k is a Statecharts model showing the steps involve in processing an order in the e-commerce system.

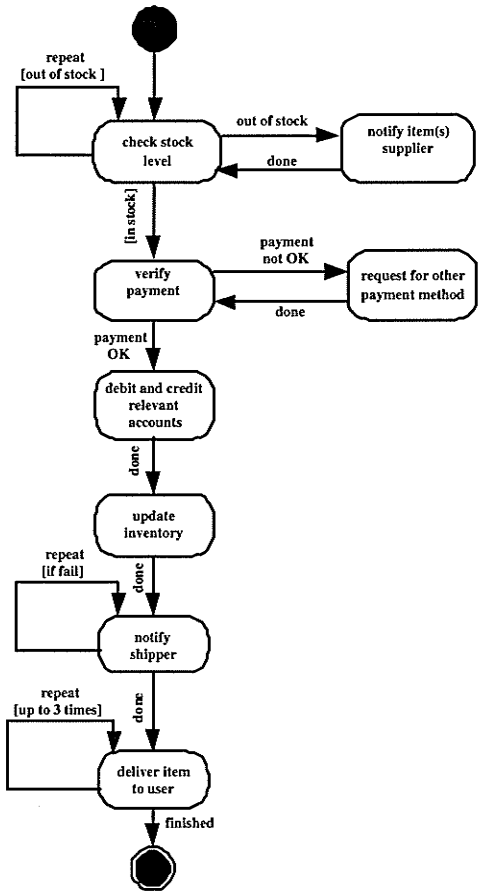

Figure 4.2k: Statecharts Model for Process Order.

The diagram in Figure 4.2l is a single consolidated flat Statecharts model, which shows all the events, actions, and conditions involved in making and processing an order in a typical B2C e-commerce scenario. Figure 4.2l is a high level flat Statecharts model, which combines the the Statecharts diagrams in Figure 4.2a — 4.2k.

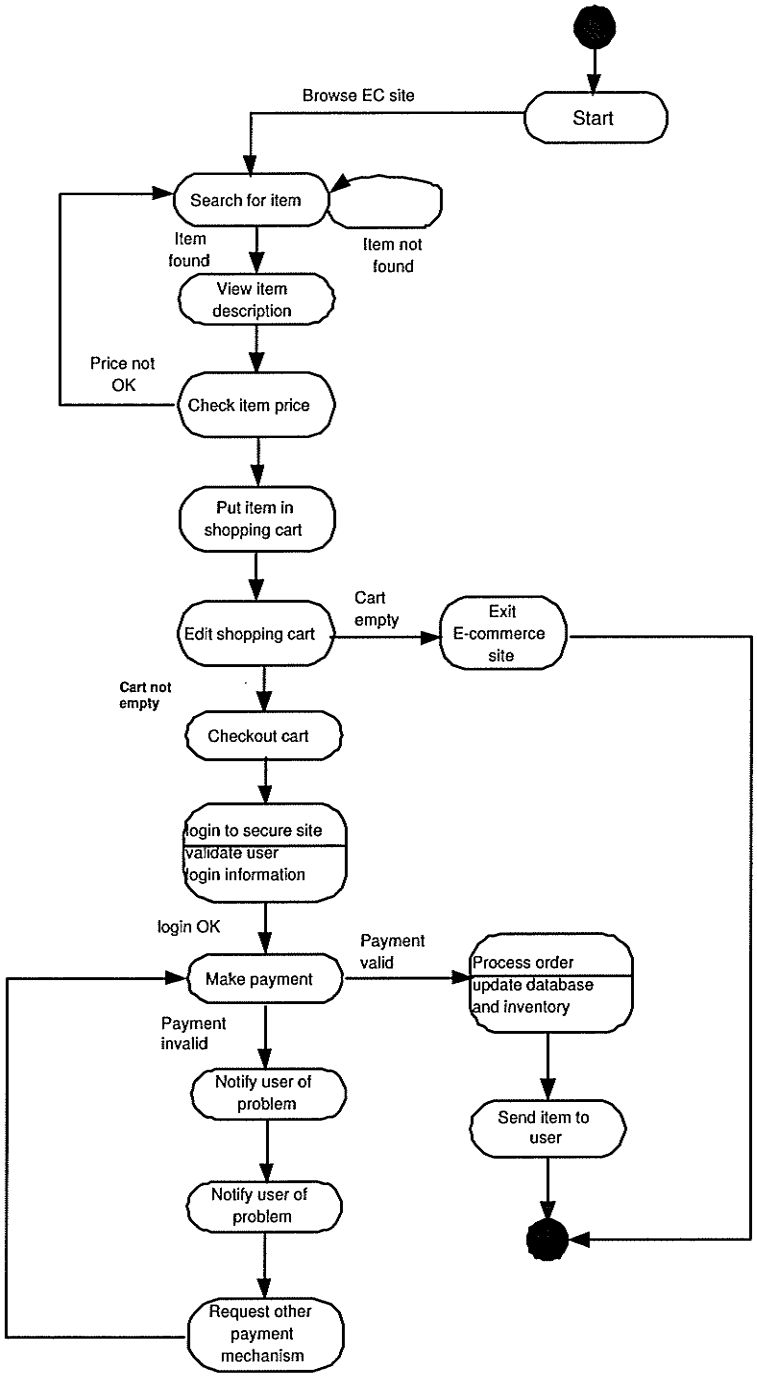

Figure 4.2l: Flat Statecharts Model of Ordering Process.

A hierarchical Statecharts representation of the *ordering process* is shown in Figure 4.3. The hierarchical Statecharts representation uses superstates and substates to show the events, actions, and conditions involved in the *ordering process* in a typical B2C e-commerce scenario. The hierarchical Statecharts drastically reduces the states and transitions explosions inherently associated with flat Statecharts.

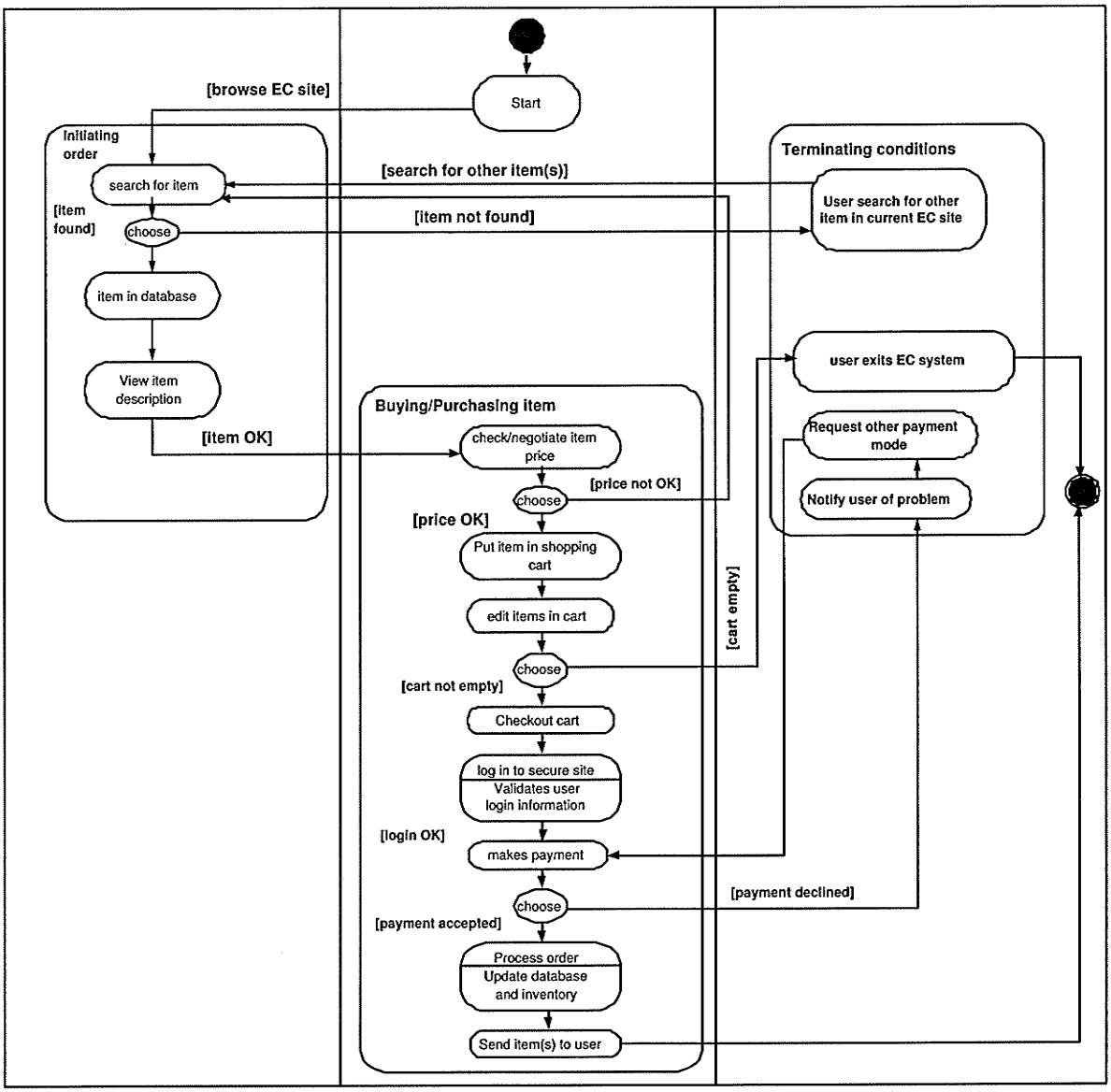

Figure 4.3: Hierarchical Statecharts Model of Ordering Process

A top-level Statecharts representation of the *ordering process* is shown in Figure

4.4. The top-level Statechharts representation is an abridged version of the hierarchical Statecharts representation in Figure 4.4. Figure 4.4 shows mainly the events in making and processing a an order in a typical B2C e-commerce scenario and it abstracts away conditions and transitions.

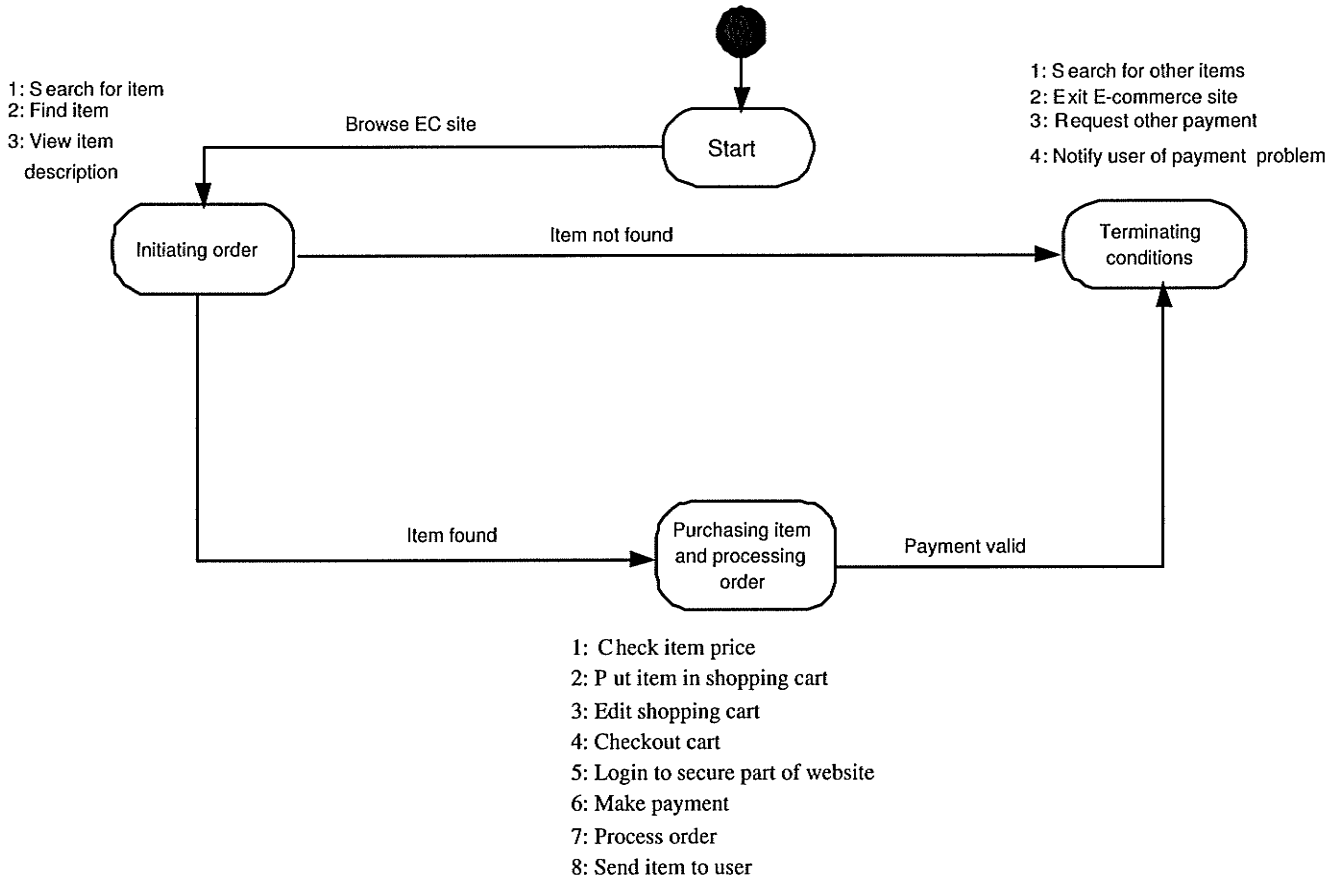

Figure 4.4: Top-level Statecharts Model of Ordering Process.

4.2.4 FSMs Specifications

Finite state machines (FSMs) provide powerful abstractions for describing system behaviours. FSMs can be represented using state transition tables and state transition diagrams.

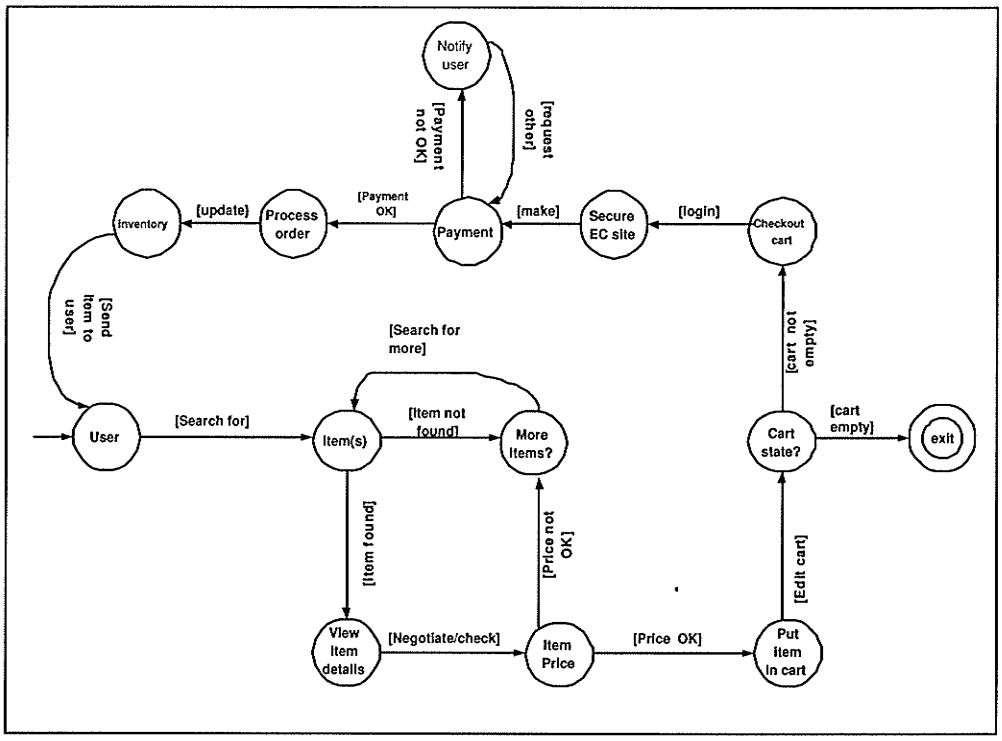

Figure 4.5: FSM representation of Ordering Process.

State transition diagrams are used instead of state transition tables to represent the *ordering process* because of their expressive and visual power. Figure 4.5 shows the non-deterministic state transition diagram representation of the *ordering process* of a typical B2C e-commerce scenario described in the pseudocode in Section 4.1.

4.2.5 UML Specifications

UML uses various diagrams such as class diagram, object diagram, sequence diagram, collaboration diagram, use case diagrams, deployment diagrams, statecharts diagrams, and component diagrams to model system behaviours and properties.

In the specification of the steps involved in making and processing an order in a typical B2C e-commerce scenario, a combination of use case diagrams, sequence diagram, collaboration diagram, and activity diagrams are used to illustrate the modelling power of UML. Sequence diagrams and collaboration diagrams are used to capture the interactions between the components of the system as well as capture the dynamic views of

the system, while activity diagram is use to capture concurrency, synchronization, and parallel behaviour of the system. The other diagrams available in the UML notation are not used because the scope of the case study in this thesis does not entail system implementation and deployment.

The use case description of the main sequence as well as alternative sequence of the *ordering process* are given below.

Use Cases (Main)

Use Case Name: Ordering Process

Summary: This use case describes the the main steps in the process of making order for item(s) by a user and processing the order in a typical B2C e-commerce system.

Actor: E-commerce system user.

Precondition: E-commerce site awaits to be browsed by user.

Description:

i. User browses e-commerce site.

ii. User searches for item in e-commerce site.

iii. If item is available user views item description.

iv. User checks/negotiates item price.

v. If item price is satisfactory, user puts item in shopping cart.

vi. User edits shopping cart.

vii. If shopping cart is not empty after modification, user checkout shopping cart.

viii. User logs in to secure part of e-commerce site.

ix. E-commerce system validates user login information.

x. User makes payment.

xi. E-commerce system processes user's order if payment is accepted.

xii. Update e-commerce system item inventory.

xiii. Send/deliver item to user.

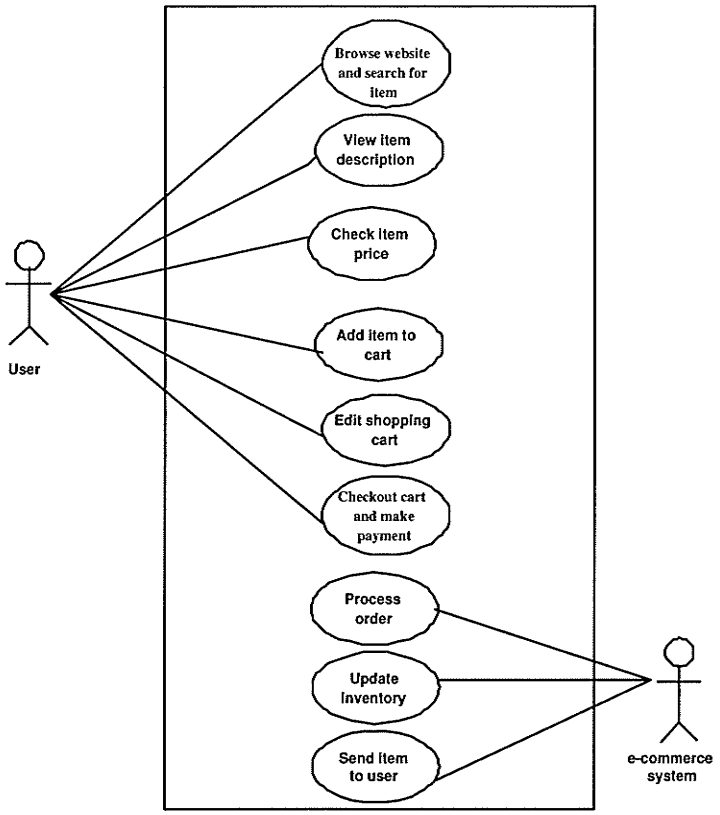

Figure 4.6: Use Case Model of Main Steps of Ordering Process.

Use cases (Alternatives)

Summary: This use case describes the alternative steps in the process of making order for item(s) by a user and processing the order in a typical B2C e-commerce system.

Description:

i. If item is not available, user may search for other item(s).

ii. If item price is not satisfactory, user search for more item(s).

iii. If shopping cart is empty after modification, user exits e-commerce system.

iv. If payment is declined, e-commerce system notifies user of payment problem and requests for other payment method.

Postcondition: E-commerce merchant sends item to user.

Figure 4.6 shows the use case model of the main steps (sequence) in the process of making order for item(s) by a user and processing the order.

The main steps in the use case descriptions above are illustrated using collaboration diagrams and a sequence diagram.

Initially a user searches for an item in an e-commerce system, the systems displays or returns the results of the search to the user. The two possible scenarios are: the item is found or the item is not found. Figure 4.7 captures this requirement.

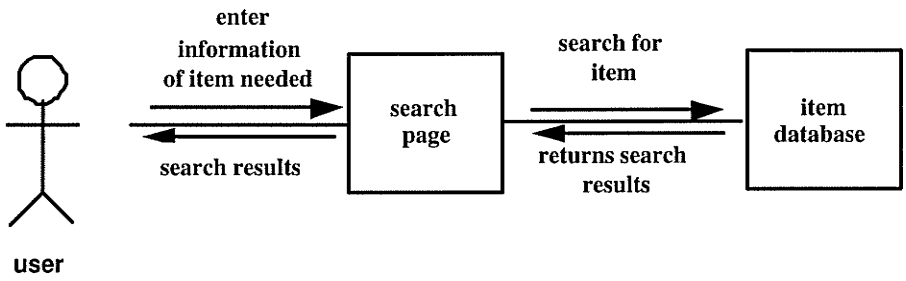

Figure 4.7: Collaboration Diagram of Item Search.

If the search result indicates that the item is available, the user views the detail description of the item (i.e., price of item, image if the item has one, and how many people have bought the item before, etc.). The system displays the description of the item to the user. Figure 4.8 captures this requirement.

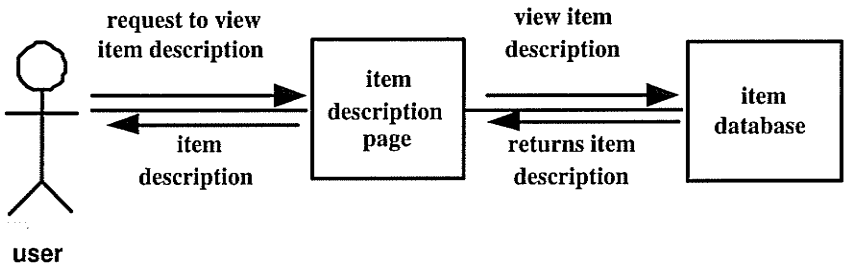

Figure 4.8: Collaboration Diagram of View Item Description.

After viewing the description of the item, the user then checks the price of the item,

then the systems displays the price of the item to the user. Figure 4.9 captures this requirement.

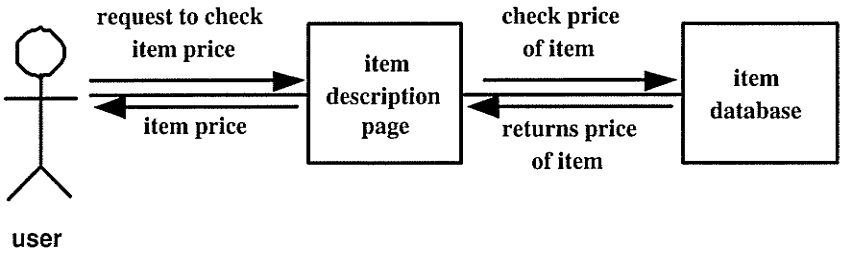

Figure 4.9: Collaboration Diagram of Check Item Price.

If the price of the item is satisfactory to the user, the user puts the item into an electronic shopping cart, and the systems displays the list of items in the shopping cart to the user. Figure 4.10 captures this requirement.

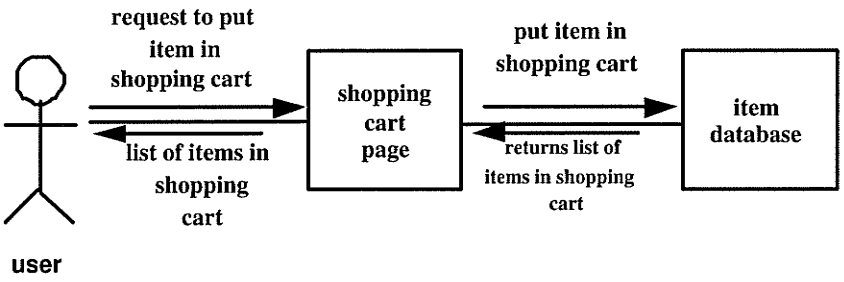

Figure 4.10: Collaboration Diagram of Add Item to Shopping Cart.

A user may edit the content of his/her shopping cart. The system displays the contents of the shopping cart after modification by the user. Figure 4.11 captures this requirement.

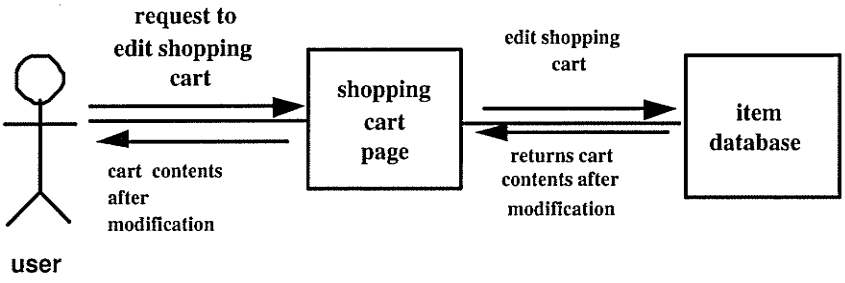

Figure 4.11: Collaboration Diagram of Edit Shopping Cart.

If the content of shopping cart is not empty after modification by the user, the user may checkout the shopping cart, then the system displays the content of the checkout cart to the user. Figure 4.12 captures this requirement.

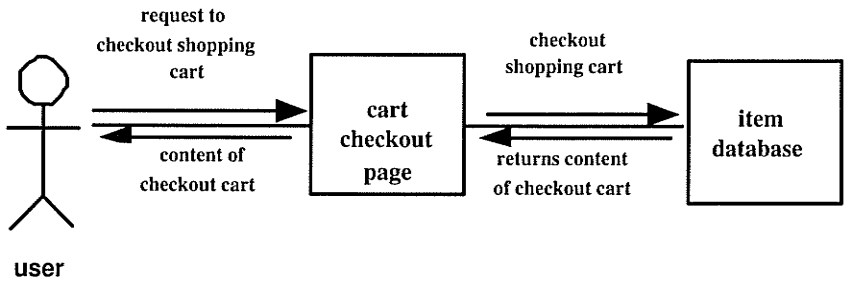

Figure 4.12: Collaboration Diagram of Checkout Cart.

After checking out the shopping cart the user logs in to a secure part of the e-commerce website, then the system validates the user login credentials. Figure 4.13 captures this requirement.

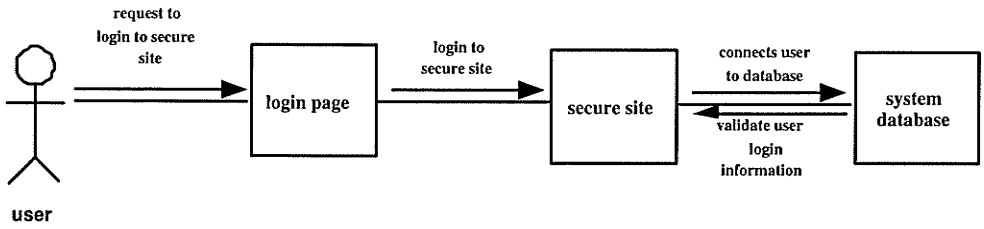

Figure 4.13: Collaboration Diagram of User Login.

After the user logs in to the secure site, the user provides payment information (such as credit card number and credit card expiration date), thus making payment. The system validates the payment information (checks whether the user balance is adequate to defray the cost of items purchased by the user) provided by the user. Figure 4.14 captures this requirement.

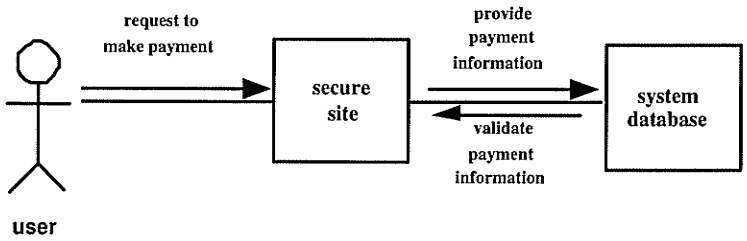

Figure 4.14: Collaboration Diagram of Make Payment.

If the payment information provided by the user is valid and sufficient to defray the cost of the item purchased by the user, the system processes the order, updates the inventory and database and provides logistics information to send the item(s) to the user.

The sequence diagram in Figure 4.15 captures this requirement.

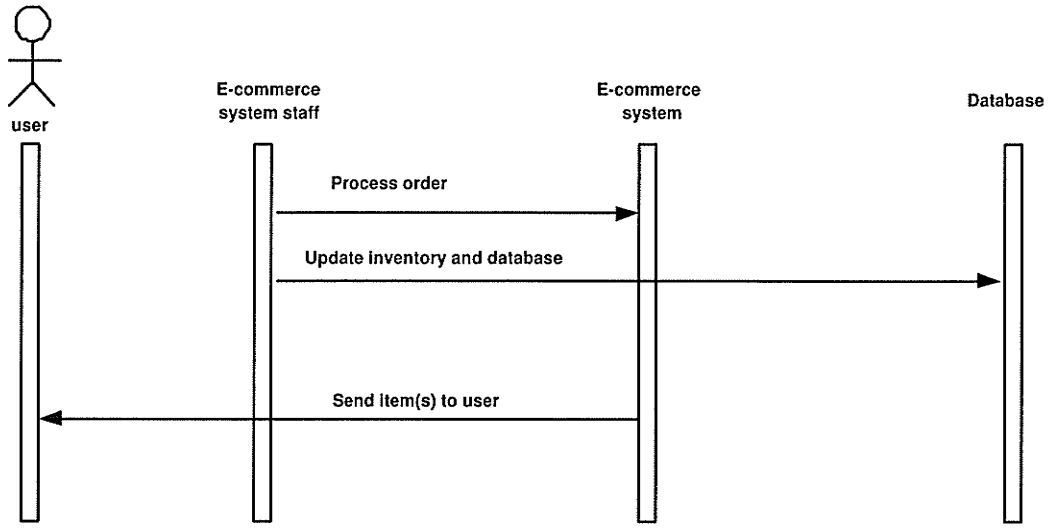

Figure 4.15: Sequence Diagram of Process Order.

Figure 4.16 shows the main steps involve in the *ordering process* in a single collaboration diagram.

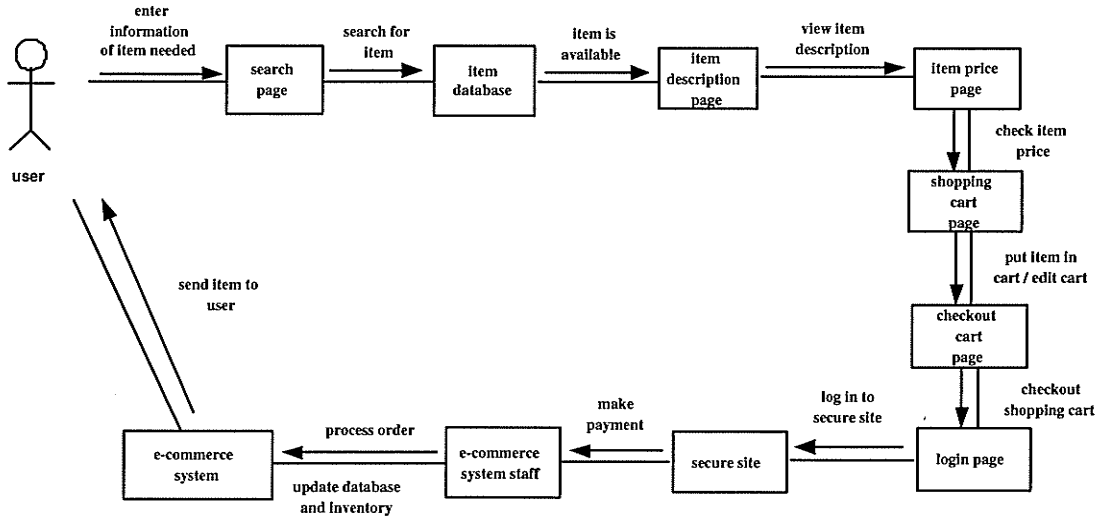

Figure 4.16: Collaboration Diagram of the Main Steps in Ordering Process.

Lastly, activity diagrams are use to show the applicability of UML in modelling concurrency, synchronization, and parallel behaviour of e-commerce systems. The activity diagrams in Figure 4.17a — 4.17e show various steps involve in the *ordering process* described in Section 4.1, while Figure 4.17f is a single consolidated activity diagram showing the of the *ordering process* of a typical B2C e-commerce scenario.

The UML activity diagram in Figure 4.17a shows the steps involve in searching for an item in a sample e-commerce system.

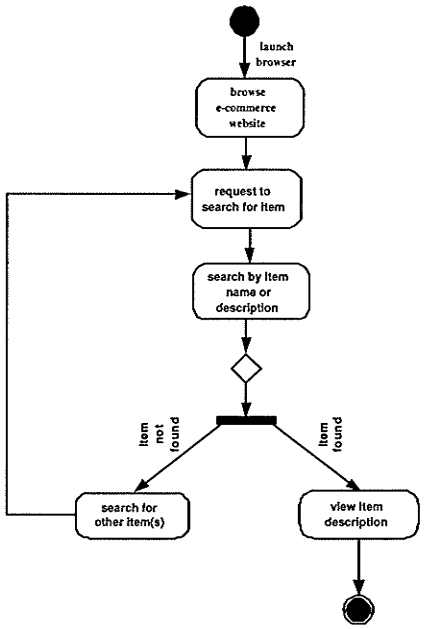

Figure 4.17a: Activity Diagram Model of Item Search Operation.

Figure 4.17b shows an activity diagram showing the steps involve in checking and negotiating the price of an item in a typical B2C e-commerce scenario.

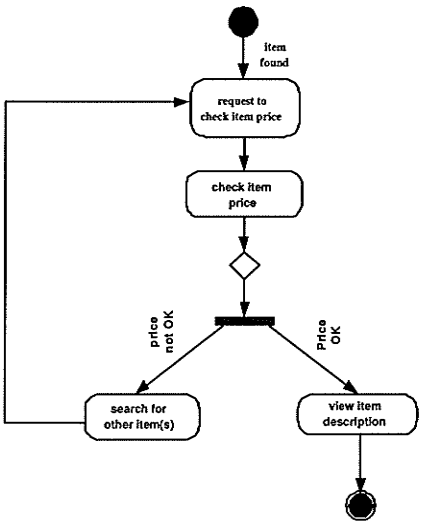

Figure 4.17b: Activity Diagram Model of Check Item Price.

The activity diagram depicted in Figure 4.17c shows the various operations on a shopping cart in the sample e-commerce system.

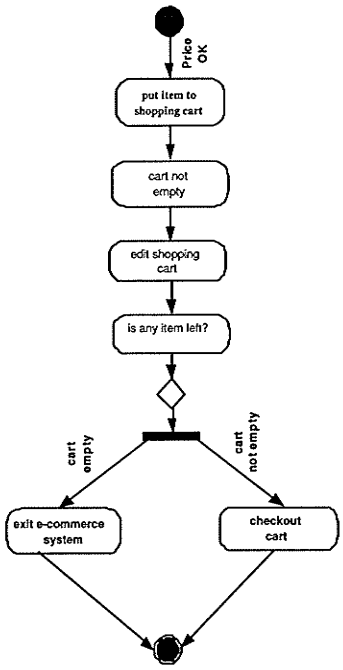

Figure 4.17c: Activity Diagram Model of Shopping Cart Operations.

The activity diagram depicted in Figure 4.17d shows the steps involve in making payment in the sample e-commerce system.

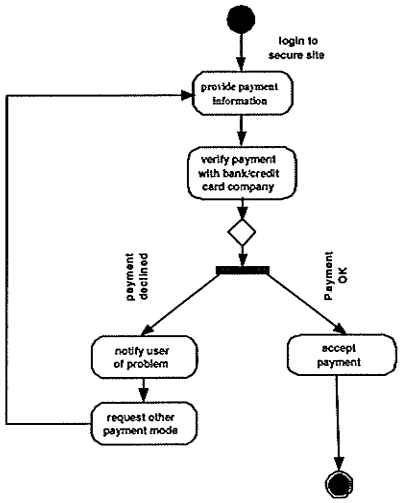

Figure 4.17d: Activity Diagram Model of Make Payment.

The activity diagram depicted in Figure 4.17e shows the steps involve in processing orders in the sample e-commerce system.

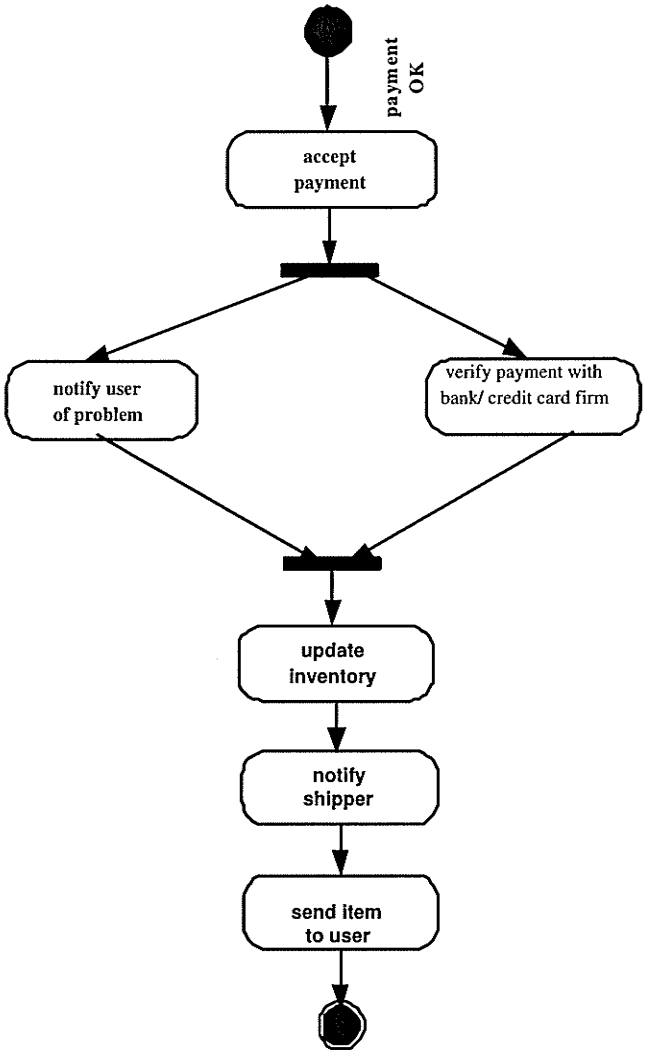

Figure 4.17e: Activity Diagram Model of Process Order.

Figure 4.17f shows a single consolidated activity diagram showing the steps of the *ordering process* of a typical B2C e-commerce scenario. The activity diagram in Figure 4.17f combines the various activity diagrams in Figure 4.17a — 4.17e.

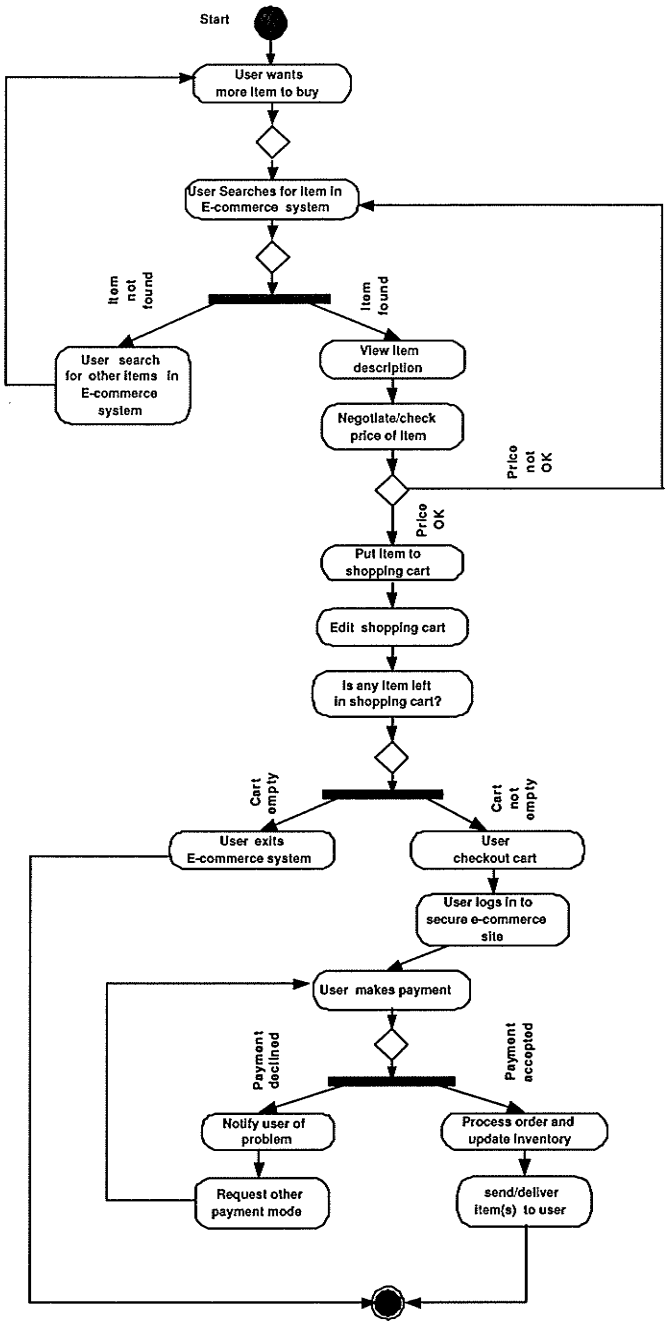

Figure 4.17f: Activity Diagram Model of Ordering Process.

Specifications Consonlidation

An e-commerce system consists of various components such as described in Section 2.1.3. The *ordering process* is a major functional requirement of B2C e-commerce applications. The specifications provided in this thesis describe a sample ordering process of a typical B2C e-commerce system. However, some other requirements of e-commerce

systems not covered in this thesis include:

- Specifications of system reports such as sales reports, account receivable and account payable reports, inventory level reports, etc.
- Detail specifications of security requirements such as authorization of users, authentication of users, encryption of users' information, among other e-commerce system security requirements.
- Specification of logistic requirements relating to shipping of items such as finding best shipping route, shipping costs, checking shipment status, among other shipping requirements are not included in the sample specifications.
- Detail Specifications of requirements relating to various accounting operations such as book keeping entries, account balancing, account reconciliation, account auditing and investigations among other accounting operations.
- Specification of website interfaces, multimedia artefacts, telecommunications/network components among other components of e-commerce systems.

Chapter 5

Evaluation

This chapter presents the evaluation approach adopted in this research, the the criteria for evaluating the specification techniques considered, the fuzzy logic evaluation framework, and the findings of the evaluation and the summary of the evaluation.

5.1 Evaluation Approach

This thesis adopts a systematic, pragmatic, and objective approach in assessing the seven specification techniques based on my practical usage and experience with the specification techniques as well as the theoretical attributes of the techniques.

In developing the evaluation criteria, an exhaustive examination of e-commerce applications and their requirements as well as the identification of some functionalities or attributes that must be present in a specification technique that can be applied to specify all the components of an e-commerce system was carried out. Subsequently a questionnaire (that consists of questions which concern the relevance of attributes of specification techniques such as concurrency, timing mechanism, etc.) on these identified attributes was constructed in such a manner as to prevent subjective responses from respondents. Then, the questionnaires were administered to some e-commerce systems designers and developers as well as researchers with focus on the low-level aspects of e-commerce

systems and capturing these low level aspects. Based on the responses received from the questionnaires, the requirements were refined and classified as critical, essential, and secondary requirements. Subsequently these requirements were evolved into the criteria for the evaluation of formal specification techniques for e-commerce applications.

Thereafter, the requirements for making and processing an order placed by a customer in a typical B2C e-commerce scenario were examined and developed the pseudocode (in Section 4.1). Thereafter, the Z notation, some diagrams in the UML notation (use case, sequence, collaboration, and activity diagrams), various types of Statecharts (top-level, hierarchical, and flat statecharts), Petri net graphs, and Finite state transition diagrams were used to specify the requirements depicted in the pseudocode. Based on the specifications and the theoretical attributes (form and structure) of the specification techniques, an evaluation of the Z notation and two of its popular variants — Concurrent Z and Real-time Z, UML, Statecharts, Petri nets and Finite State Machines was done in conjunction with some specialists using the evaluation criteria and a fuzzy logic framework. Also, the evaluation rankings obtained from the domain specialists were analysed using first order statistics.

5.2 Evaluation Criteria

The criteria for evaluating the specification techniques amenable to specifying e-commerce systems are presented below. The criteria were systematically derived mainly from the requirements (critical and essential requirements) necessary for a specification technique to completely specify all the components of an e-commerce system.

Each of the specification technique considered was examined for the following attributes:

i. *Concurrency support*: Does the specification technique have constructs for modelling concurrent behaviour of e-commerce systems? In assessing this criterion, a specification technique is checked for constructs for capturing concurrency be-

haviour of a system. If a specification technique has constructs for handling concurrency, then determine the extent (relative to other techniques for evaluation) to which the specification technique can capture concurrent behaviour of e-commerce systems.

ii. *Timing support*: Can the specification technique capture time-dependent objects or actions, dynamic, as well as real-time constraints of e-commerce applications? In rating this criterion, a specification technique is checked for constructs for capturing time property and temporal constraints of system behaviour. If a specification technique has attributes for handling time and capturing temporal constraints, then ascertain the relative degree to which a specification technique can capture time property and temporal constraints of an e-commerce system.

iii. *User interfaces support*: Can the specification technique describe non-functional and external interactions such as human-computer interaction and system usability of the e-commerce system? The emphasis here is to see if a specification technique has attributes to handle non-functional system requirements such as user friendliness and usability. If a specification technique has attributes to model these requirements, then an assessment of the degree to which the technique can capture these requirements for e-commerce system is carried out.

iv. *Abstract functionality support*: Can the specification technique describe functional system requirements, such as the data component of e-commerce applications? In this criterion, the emphasis is to first determine if a specification technique has attributes for modelling low-level abstract functionalities (such as data, variables, literals, etc) of a system. If this attribute is present in a specification technique, then assess the degree to which a technique can capture these requirements for e-commerce system.

v. *Nondeterminism*: Does the specification technique have non-deterministic constructs for modelling indirect data access and unrestricted choice from a list of system attributes? In assessing this criterion, check each specification technique

for constructs for capturing non-deterministic behaviour of a system. If a specification technique has constructs for handling non-determinism, then determine the extent (relative to other techniques for evaluation) to which the specification technique can capture non-deterministic behaviour of e-commerce systems.

vi. *Size or Complexity management*: E-commerce systems are generally large and complex. We are interested in ascertaining if a specification technique has constructs for structuring system specifications into manageable sizes. Thus, can the specification technique handle an arbitrarily size specification? Does the technique provide mechanisms to support modularity?

vii. *Verifiability*: Does the specification technique have tools for type checking, verifying, and validating specifications as well as a proof mandate? If the specification technique has tools for validating and verifying specifications, are these tools widely available?

viii. *Executability*: Can specifications derived from a specification technique be tested, executed, and easily translated into the system implementation?

ix. *Modularity*: Does the specification technique have constructs for composing smaller specifications and extending specifications, thereby enabling the derivation of large and complex specifications from smaller specifications?

5.3 Fuzzy Logic Evaluation Framework

Fuzzy set is use to cluster the degree or extent of satisfaction of each evaluation criterion by the evaluated specification techniques. The approach adopts the sliding window mechanism to appropriately place a value on the levels of satisfaction of each evaluation criterion. The fuzzy logic evaluation framework uses trapezoidal membership functions to depict the degree of satisfaction of each evaluation criterion by the evaluated specification techniques. The framework adopts evaluation rating levels, which consist of a fixed

measurement scale in the interval [0..4]. This interval defines the set of numbers use to measure the degree of satisfaction of each evaluation criterion by the various evaluated specification techniques.

The linguistic values used in the framework include: *total absence (TA), marginal presence (MP), average presence (AP), high presence (HP),* and *extensive presence (EP).* The linguistic value TA represents a scenario in which an attribute is totally absent in a specification technique, an evaluation rating level of 0 is assigned to this value. The linguistic value MP depicts a scenario in which a specification technique has minimal (very little or marginal) constructs for describing an attribute, an evaluation rating level of 1 is assigned to this value. The linguistic value AP illustrates a scenario in which a specification technique has just sufficient (about average) constructs for describing an attribute, an evaluation rating level of 2 is assigned to this value. The linguistic value HP represents a scenario in which a specification technique has more than adequate constructs for describing an attribute, but the attribute is not the major strength of the specification technique, an evaluation rating level of 3 is assigned to this value. The linguistic value EP portrays a scenario in which an attribute is the main strength of a specification technique, an evaluation rating level of 4 is assigned to this value.

These linguistic values can be represented as a set:

$C = [TA, MP, AP, HP, EP]$

where the elements in the set are the five linguistic symbols representing the five linguistic values. Table 5 presents a summary of the interpretation of the five linguistic values used in the framework and their corresponding rating levelss.

Levels	Linguistic Value	Interpretation
0	Total Absence (TA)	Attribute is completely deficient in specification technique.
1	Marginal Presence (MP)	Specification technique has minimal constructs for representing attribute.
2	Average Presence (AP)	Specification technique has just adequate constructs for representing attribute.
3	High Presence (HP)	Specification technique has more than sufficient constructs for representing attribute, but attribute is not its key strength.
4	Extensive Presence (EP)	Attribute is a major strength of specification technique.

Table 5: Interpretation of Linguistic Values.

A trapezoidal fuzzy number $A = [a, b, c, d]$ (where a and d represent the values of the lower and upper limits of the larger base of the trapezium, b and c represent the values of the lower and upper limits of the smaller base of the trapezium, $a \leq b$, $b \leq c$, and $c \leq d$) is use to represent the range of evaluation rating levels for the linguistic values. Trapezoidal fuzzy numbers and representations are used in this framework because they offer a finer level of granularity compared to other representations such as triangular representations. Figure 5.1 shows a trapezoidal representation of the various linguistic values, while Table 6 shows the various trapezoidal fuzzy numbers and their associated linguistic value/symbol.

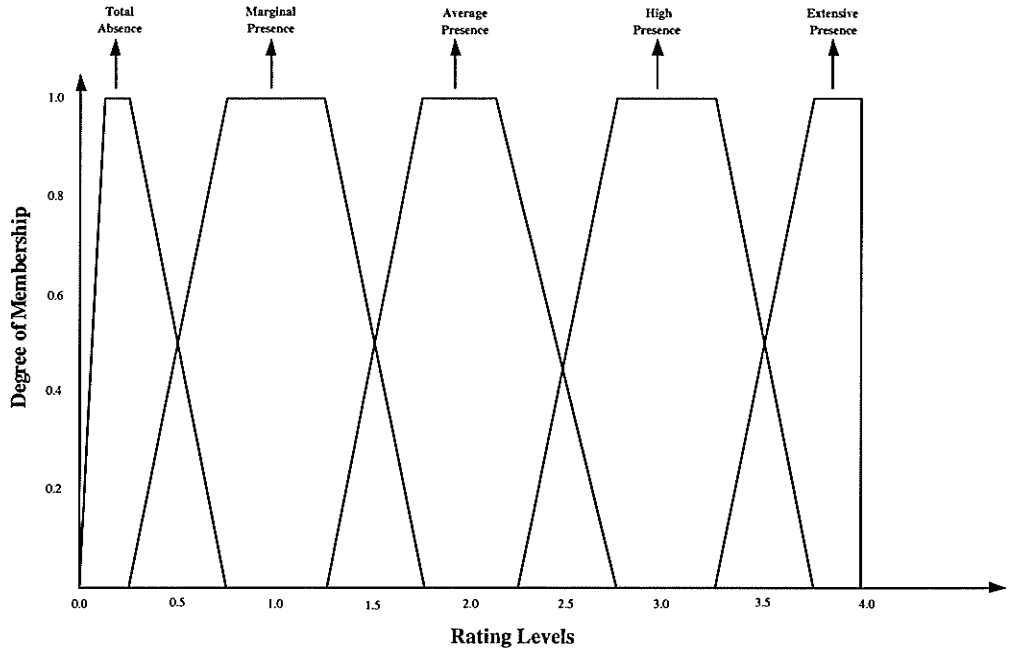

Figure 5.1: Trapezoidal Representation of The Various Linguistic Values.

Lingustic Symbols	Trapezoidal fuzzy numbers
TA	[0.00, 0.125, 0.25, 0.75]
MP	[0.25, 0.75, 1.25, 1.75]
AP	[1.25, 1.75, 2.25, 2.75]
HP	[2.25, 2.75, 3.25, 3.75]
EP	[3.25, 3.75, 4.00, 4.00]

Table 6: Trapezoidal Fuzzy Numbers of Linguistic Values.

Now the membership functions are formulated as a set of linear equations using the trapezoidal fuzzy numbers a, b, c, and d respectively. The membership function f_m for each linguistic value is given by:

$$f_m = \begin{cases} \frac{x-a}{b-a} & \text{for} \quad a \leq x \leq b; \\ 1 & \text{for} \quad b \leq x \leq c; \\ \frac{x-c}{d-c} & \text{for} \quad c \leq x \leq d; \\ 0 & \text{for} \quad x < a;\ x > d \end{cases}$$

The membership functions of the various linguistic values are given below.

$$f_{TA} = \begin{cases} 8x & \text{for} \quad 0.00 \leq x \leq 0.125; \\ 1 & \text{for} \quad 0.125 \leq x \leq 0.25; \\ 2x - 0.5 & \text{for} \quad 0.25 \leq x \leq 0.75; \\ 0 & \text{for} \quad x < 0.00;\ x > 0.75. \end{cases}$$

$$f_{MP} = \begin{cases} 2x - 0.5 & \text{for} \quad 0.25 \leq x \leq 0.75; \\ 1 & \text{for} \quad 0.75 \leq x \leq 1.25; \\ 2x - 2.5 & \text{for} \quad 1.25 \leq x \leq 1.75; \\ 0 & \text{for} \quad x < 0.25;\ x > 1.75. \end{cases}$$

$$f_{AP} = \begin{cases} 2x - 2.5 & \text{for} \quad 1.25 \leq x \leq 1.75; \\ 1 & \text{for} \quad 1.75 \leq x \leq 2.25; \\ 2x - 4.5 & \text{for} \quad 2.25 \leq x \leq 2.75; \\ 0 & \text{for} \quad x < 1.25;\ x > 2.75. \end{cases}$$

$$f_{HP} = \begin{cases} 2x - 4.5 & \text{for} \quad 2.25 \leq x \leq 2.75; \\ 1 & \text{for} \quad 2.75 \leq x \leq 3.25; \\ 2x - 6.5 & \text{for} \quad 3.25 \leq x \leq 3.75; \\ 0 & \text{for} \quad x < 2.25;\ x > 3.75. \end{cases}$$

$$f_{EP} = \begin{cases} 2x - 6.5 & \text{for} \quad 3.25 \leq x \leq 3.75; \\ 1 & \text{for} \quad 3.75 \leq x \leq 4.00; \\ 0 & \text{for} \quad x < 3.25;\ x > 4.00. \end{cases}$$

The evaluation rating presented in this thesis was carried out in conjunction with fifteen other specialists $A_1..A_{16}$, so as to enhance the objectivity and acceptability of the evaluation. The specialists have several years of applying at least three of the specification techniques to specifying various software applications including e-commerce applications. Some of the specialists are academic and industrial researchers, while others are practising e-commerce application developers.

The procedure adopted in arriving at the evaluation results is given below.

Step 1 : The specialists rate each evaluation criterion for each of the seven specification techniques using the rating linguistic values [TA, MP, AP, HP, EP], which conforms to the rating levels [0..4].

Step 2 : The rating of linguistic values obtained from the various specialists are fuzzified using trapezoidal fuzzy numbers (as shown in Table 6) giving adjusted trapezoidal fuzzy numbers. Then for each specification technique, the fuzzy mean (average) of the adjusted trapezoidal fuzzy numbers is computed for each evaluation criteria yielding aggregated trapezoidal fuzzy numbers.

Step 3 : The aggregated trapezoidal fuzzy numbers are defuzzified to appropriate crisp (non fuzzy) values using the formula for the maximizing value $\frac{b+c}{2}$, where b and c are the second and third coordinate values the trapezium.

Step 4 : Lastly, the crisp values obtained from the defuzzification of the aggregated trapezoidal fuzzy numbers are normalized such that the resulting crisp values lie in the interval [0..1] — obtained by dividing the crisp values by 4 (the maximum rating level).

5.4 Evaluation Results

This section presents the ratings of the specialists of each criterion for the seven specification techniques, the analysis of the ratings, and a discussion of the evaluation outcome.

The results obtained from the procedure outlined at the end of Section 5.3 were compared with results obtained using first order statistics.

Tables 7a - 7i show the ratings of each of the evaluation criterion — concurrency, timing constraint, user interfaces, abstract functionality, nondeterminism, complexity management, modularity, verifiability, and executability by the specialists for the seven specification techniques. Tables 8a - 8i show the analysis of the ratings (aggregated trapezoidal fuzzy numbers, defuzzified values, first order statistics values, normalized defuzzified values, and the normalized first order statistics values) of each of the seven specification techniques for the various evaluation criterion. The aggregated trapezoidal fuzzy numbers, defuzzified values, and the normalized defuzzified values (presented in the second, third, and fifth columns of Tables 8a - 8i) were obtained using the procedure outlined towards the end of Section 5.3. While the values presented in the fourth and sixth columns where obtained using first order statistics.

Concurrency	Z	Concurrent Z	Real-Time Z	UML	Petri nets	Statecharts	FSMs
A_1	TA	EP	MP	AP	HP	HP	TA
A_2	MP	HP	TA	HP	AP	MP	MP
A_3	MP	HP	AP	MP	EP	AP	MP
A_4	TA	EP	AP	HP	HP	AP	AP
A_5	TA	EP	AP	HP	HP	AP	TA
A_6	AP	HP	HP	AP	AP	MP	MP
A_7	TA	AP	MP	HP	HP	HP	AP
A_8	MP	EP	AP	HP	HP	EP	MP
A_9	TA	EP	AP	AP	HP	MP	MP
A_{10}	TA	EP	AP	HP	AP	AP	AP
A_{11}	TA	HP	HP	HP	MP	MP	AP
A_{12}	MP	EP	TA	EP	HP	AP	MP
A_{13}	TA	HP	AP	AP	HP	AP	MP
A_{14}	TA	EP	HP	MP	MP	HP	MP
A_{15}	MP	AP	AP	HP	AP	MP	AP
A_{16}	TA	EP	AP	HP	HP	HP	MP

Table 7a: Specialists Rating for Concurrency Criterion

Timing Constraint	Z	Concurrent Z	Real-Time Z	UML	Petri nets	Statecharts	FSMs
A_1	TA	AP	EP	AP	AP	AP	MP
A_2	MP	MP	HP	MP	MP	MP	TA
A_3	TA	AP	EP	AP	HP	AP	MP
A_4	TA	HP	EP	HP	AP	HP	AP
A_5	MP	AP	AP	AP	AP	AP	MP
A_6	TA	AP	EP	AP	HP	AP	MP
A_7	TA	HP	EP	HP	AP	MP	TA
A_8	AP	AP	HP	MP	AP	AP	MP
A_9	MP	AP	EP	AP	HP	MP	TA
A_{10}	TA	MP	EP	AP	AP	AP	MP
A_{11}	TA	AP	HP	HP	AP	HP	AP
A_{12}	TA	AP	EP	AP	AP	AP	MP
A_{13}	TA	AP	EP	HP	HP	HP	AP
A_{14}	MP	HP	EP	MP	AP	MP	TA
A_{15}	TA	AP	HP	AP	HP	AP	MP
A_{16}	AP	HP	EP	AP	AP	AP	MP

Table 7b: Specialists Rating for Timing Constraint Criterion

User Interfaces	Z	Concurrent Z	Real-Time Z	UML	Petri nets	Statecharts	FSMs
A_1	TA	MP	MP	EP	MP	HP	MP
A_2	MP	TA	TA	HP	TA	EP	TA
A_3	TA	TA	TA	EP	AP	HP	AP
A_4	TA	AP	MP	EP	MP	AP	MP
A_5	MP	TA	TA	AP	TA	AP	TA
A_6	TA	TA	TA	EP	MP	HP	MP
A_7	TA	MP	TA	EP	TA	HP	MP
A_8	TA	TA	MP	HP	MP	EP	MP
A_9	MP	TA	TA	EP	TA	HP	AP
A_{10}	TA	MP	TA	EP	MP	HP	MP
A_{11}	TA	TA	TA	HP	TA	EP	TA
A_{12}	MP	TA	MP	EP	MP	HP	MP
A_{13}	TA	MP	TA	EP	TA	HP	MP
A_{14}	MP	TA	TA	EP	MP	EP	TA
A_{15}	TA	MP	MP	HP	TA	HP	MP
A_{16}	TA	TA	TA	EP	TA	HP	MP

Table 7c: Specialists Rating for User Interfaces Criterion

Abstract Functionality	Z	Concurrent Z	Real-Time Z	UML	Petri nets	Statecharts	FSMs
A_1	EP	AP	HP	MP	AP	TA	MP
A_2	HP	HP	AP	TA	HP	MP	TA
A_3	HP	EP	HP	MP	AP	MP	MP
A_4	EP	HP	HP	AP	AP	AP	MP
A_5	EP	AP	AP	TA	MP	MP	AP
A_6	AP	EP	HP	MP	AP	AP	MP
A_7	HP	HP	MP	MP	MP	MP	MP
A_8	EP	HP	HP	AP	TA	MP	TA
A_9	EP	AP	AP	HP	AP	TA	MP
A_{10}	EP	HP	HP	MP	AP	MP	MP
A_{11}	AP	AP	MP	MP	HP	MP	AP
A_{12}	EP	HP	HP	TA	AP	AP	MP
A_{13}	HP	MP	HP	MP	AP	TA	MP
A_{14}	EP	AP	EP	AP	MP	MP	TA
A_{15}	HP	HP	AP	MP	AP	MP	MP
A_{16}	EP	HP	AP	AP	MP	AP	TA

Table 7d: Specialists Rating for Abstract Functionality Criterion

Nondeterminism	Z	Concurrent Z	Real-Time Z	UML	Petri nets	Statecharts	FSMs
A_1	TA	HP	EP	AP	AP	MP	MP
A_2	MP	EP	HP	HP	MP	AP	HP
A_3	TA	HP	HP	AP	HP	MP	AP
A_4	AP	AP	AP	MP	AP	HP	MP
A_5	TA	HP	HP	AP	HP	MP	AP
A_6	TA	HP	HP	AP	MP	HP	HP
A_7	MP	AP	AP	MP	AP	AP	MP
A_8	TA	HP	HP	AP	AP	MP	MP
A_9	MP	HP	EP	AP	HP	AP	AP
A_{10}	TA	EP	HP	HP	AP	MP	MP
A_{11}	AP	HP	HP	AP	MP	MP	HP
A_{12}	TA	HP	HP	AP	HP	AP	AP
A_{13}	AP	HP	AP	AP	HP	MP	AP
A_{14}	MP	AP	HP	MP	MP	AP	HP
A_{15}	TA	HP	HP	AP	HP	MP	AP
A_{16}	TA	AP	HP	MP	AP	AP	MP

Table 7e: Specialists Rating for Nondeterminism Criterion

Complexity Management	Z	Concurrent Z	Real-Time Z	UML	Petri nets	Statecharts	FSMs
A_1	EP	HP	HP	AP	AP	AP	MP
A_2	HP	AP	MP	HP	HP	HP	TA
A_3	EP	MP	AP	AP	MP	AP	MP
A_4	AP	HP	HP	MP	AP	MP	MP
A_5	EP	EP	AP	AP	HP	MP	AP
A_6	HP	AP	EP	MP	MP	AP	MP
A_7	EP	HP	AP	AP	AP	AP	MP
A_8	HP	HP	HP	AP	AP	HP	TA
A_9	EP	AP	MP	MP	HP	HP	MP
A_{10}	EP	MP	HP	AP	AP	AP	MP
A_{11}	AP	HP	AP	AP	AP	AP	AP
A_{12}	HP	HP	AP	HP	MP	AP	MP
A_{13}	EP	AP	HP	AP	MP	HP	MP
A_{14}	HP	HP	EP	HP	AP	AP	TA
A_{15}	EP	EP	HP	AP	MP	MP	MP
A_{16}	EP	HP	AP	MP	AP	AP	TA

Table 7f: Specialists Rating for Complexity Management Criterion

Modularity	Z	Concurrent Z	Real-Time Z	UML	Petri nets	Statecharts	FSMs
A_1	HP	HP	HP	HP	HP	HP	MP
A_2	HP	EP	EP	EP	EP	EP	AP
A_3	EP	AP	AP	AP	AP	AP	MP
A_4	AP	HP	AP	HP	HP	HP	HP
A_5	HP	AP	HP	AP	AP	AP	HP
A_6	HP	EP	HP	HP	HP	HP	MP
A_7	EP	HP	EP	AP	AP	AP	MP
A_8	AP	HP	AP	HP	HP	MP	AP
A_9	HP	AP	HP	EP	AP	HP	HP
A_{10}	HP	HP	HP	AP	EP	HP	MP
A_{11}	EP	HP	EP	HP	HP	EP	MP
A_{12}	EP	EP	AP	AP	AP	AP	AP
A_{13}	HP	HP	AP	HP	HP	HP	MP
A_{14}	AP	AP	HP	AP	AP	AP	AP
A_{15}	HP	HP	AP	HP	HP	HP	HP
A_{16}	HP	AP	HP	HP	AP	HP	AP

Table 7g: Specialists Rating for Modularity Criterion

Verifiability	Z	Concurrent Z	Real-Time Z	UML	Petri nets	Statecharts	FSMs
A_1	EP	TA	TA	AP	AP	AP	TA
A_2	HP	AP	AP	HP	HP	HP	AP
A_3	AP	MP	MP	MP	AP	MP	TA
A_4	EP	TA	AP	AP	MP	AP	TA
A_5	EP	MP	MP	AP	AP	AP	MP
A_6	HP	TA	TA	MP	MP	MP	TA
A_7	EP	TA	TA	AP	AP	AP	TA
A_8	EP	AP	AP	HP	AP	HP	MP
A_9	HP	TA	TA	MP	HP	MP	TA
A_{10}	AP	MP	MP	MP	AP	AP	MP
A_{11}	EP	TA	TA	AP	MP	AP	TA
A_{12}	EP	MP	MP	AP	AP	AP	MP
A_{13}	HP	TA	TA	AP	AP	AP	TA
A_{14}	EP	MP	MP	MP	HP	HP	MP
A_{15}	EP	TA	TA	AP	AP	AP	TA
A_{16}	EP	MP	MP	AP	MP	AP	TA

Table 7h: Specialists Rating for Verifiability Criterion

Executability	Z	Concurrent Z	Real-Time Z	UML	Petri nets	Statecharts	FSMs
A_1	MP	MP	MP	MP	MP	TA	AP
A_2	AP	TA	TA	TA	AP	MP	MP
A_3	TA	MP	MP	MP	TA	AP	AP
A_4	MP	AP	AP	AP	MP	MP	TA
A_5	MP	TA	MP	MP	TA	AP	AP
A_6	MP	MP	AP	AP	MP	MP	MP
A_7	TA	MP	MP	MP	AP	TA	AP
A_8	MP	AP	MP	MP	MP	MP	AP
A_9	AP	MP	TA	TA	MP	TA	MP
A_{10}	MP	TA	MP	MP	TA	MP	AP
A_{11}	TA	MP	AP	AP	MP	MP	TA
A_{12}	MP	AP	MP	TA	AP	MP	AP
A_{13}	MP	MP	TA	MP	MP	AP	MP
A_{14}	TA	TA	MP	AP	TA	MP	TA
A_{15}	MP	MP	TA	MP	MP	AP	MP
A_{16}	MP	AP	MP	AP	TA	MP	TA

Table 7i: Specialists Rating for Executability Criterion

Technique	Aggregate trapezoidal no	Defuzzified values	1st order Statistical values	Normalized defuzzified values	Normalized 1st order values
Z	[0.16, 0.73, 1.00, 1.50]	0.87	0.85	0.22	0.21
Conc. Z	[2.69, 3.19, 3.55, 3.77]	3.37	3.30	0.84	0.83
R.Time Z	[1.16, 1.61, 2.06, 2.56]	1.84	1.85	0.46	0.46
UML	[1.81, 2.31, 2.80, 3.27]	2.56	2.55	0.64	0.64
P. nets	[1.81, 2.31, 2.80, 3.27]	2.56	2.55	0.64	0.64
Scharts	[1.31, 2.13, 2.61, 3.08]	2.36	2.28	0.59	0.57
FSMs	[0.53, 0.98, 1.44, 1.94]	1.21	1.22	0.30	0.31

Table 8a: Defuzzified and Normalized Values for Concurrency Criterion

Technique	Aggregate trapezoidal no	Defuzzified values	1st order Statistical values	Normalized defuzzified values	Normalized 1st order values
Z	[0.22, 0.48, 0.75, 1.25]	0.62	0.68	0.16	0.17
Conc. Z	[1.38, 1.88, 2.38, 2.88]	2.13	2.12	0.53	0.53
R.Time Z	[2.88, 3.38, 3.70, 3.86]	3.54	3.45	0.89	0.86
UML	[1.31, 1.81, 2.31, 2.81]	2.06	2.06	0.52	0.52
P. nets	[1.50, 2.00, 2.50, 3.00]	2.25	2.26	0.57	0.57
Scharts	[1.17, 1.64, 2.11, 2.58]	1.87	1.88	0.47	0.47
FSMs	[0.38, 0.81, 1.19, 1.69]	1.00	1.02	0.25	0.26

Table 8b: Defuzzified and Normalized Values for Timing Constraint Criterion

Technique	Aggregate trapezoidal no	Defuzzified values	1st order Statistical values	Normalized defuzzified values	Normalized 1st order values
Z	[0.08, 0.32, 0.56, 1.06]	0.44	0.51	0.11	0.13
Conc. Z	[0.16, 0.42, 0.69, 1.19]	0.56	0.61	0.14	0.15
R.Time Z	[0.08, 0.32, 0.56, 1.06]	0.44	0.51	0.11	0.13
UML	[2.88, 3.56, 3.70, 3.86]	3.63	3.50	0.91	0.88
P. nets	[0.19, 0.50, 0.81, 1.31]	0.66	0.70	0.17	0.18
Scharts	[2.38, 2.88, 3.31, 3.69]	3.10	3.06	0.78	0.77
FSMs	[0.31, 0.72, 1.13, 1.63]	0.92	0.95	0.23	0.24

Table 8c: Defuzzified and Normalized Values for User Interfaces Criterion

Technique	Aggregate trapezoidal no	Defuzzified values	1st order Statistical values	Normalized defuzzified values	Normalized 1st order values
Z	[2.69, 3.19, 3.55, 3.77]	3.37	3.23	0.93	0.81
Conc. Z	[1.94, 2.44, 2.84, 3.34]	2.64	2.64	0.66	0.66
R.Time Z	[1.75, 2.25, 2.73, 3.20]	2.49	2.48	0.62	0.62
UML	[0.58, 1.01, 1.44, 1.94]	1.22	1.24	0.31	0.31
P. nets	[1.05, 1.52, 2.00, 2.50]	1.76	1.77	0.44	0.44
Scharts	[0.45, 0.88, 1.31, 1.75]	1.10	1.11	0.28	0.28
FSMs	[0.31, 0.72, 1.13, 1.63]	0.92	0.95	0.23	0.4

Table 8d: Defuzzified and Normalized Values for Abstract functionality Criterion

Technique	Aggregate trapezoidal no	Defuzzified values	1st order Statistical values	Normalized defuzzified values	Normalized 1st order values
Z	[0.30, 0.62, 0.88, 1.38]	0.75	0.80	0.19	0.20
Conc. Z	[2.13, 2.63, 3.09, 3.53]	2.86	2.84	0.72	0.71
R.Time Z	[2.19, 2.72, 3.16, 3.59]	2.94	2.91	0.74	0.73
UML	[1.13, 1.63, 2.13, 2.63]	1.88	1.88	0.47	0.47
P. nets	[1.38, 1.88, 2.38, 2.88]	2.13	2.12	0.53	0.53
Scharts	[0.88, 1.38, 1.88, 2.38]	1.63	1.63	0.41	0.41
FSMs	[1.13, 1.63, 2.13, 2.63]	1.88	1.88	0.47	0.47

Table 8e: Defuzzified and Normalized Values for Nondeterminism Criterion

Technique	Aggregate trapezoidal no	Defuzzified values	1st order Statistical values	Normalized defuzzified values	Normalized 1st order values
Z	[2.69, 3.19, 3.56, 3.77]	3.37	3.30	0.84	0.83
Conc. Z	[1.88, 2.38, 2.84, 3.28]	2.61	2.59	0.65	0.65
R.Time Z	[1.75, 2.25, 2.72, 3.16]	2.49	2.47	0.62	0.62
UML	[1.19, 1.69, 2.19, 2.69]	1.94	1.94	0.49	0.49
P. nets	[1.13, 1.63, 2.13, 2.63]	1.88	1.88	0.47	0.47
Scharts	[1.31, 1.81, 2.31, 2.81]	2.06	2.06	0.52	0.52
FSMs	[0.31, 0.72, 1.00, 1.63]	0.86	0.93	0.22	0.23

Table 8f: Defuzzified and Normalized Values for Complexity Management Criterion

Technique	Aggregate trapezoidal no	Defuzzified values	1st order Statistical values	Normalized defuzzified values	Normalized 1st order values
Z	[2.31, 2.84, 3.25, 3.63]	3.01	3.00	0.75	0.75
Conc. Z	[2.13, 2.63, 3.08, 3.48]	2.85	2.83	0.71	0.70
R.Time Z	[2.06, 2.56, 3.02, 3.42]	2.79	2.77	0.70	0.69
UML	[2.00, 2.50, 2.97, 3.41]	2.73	2.72	0.68	0.68
P.nets	[1.94, 2.55, 2.91, 3.34]	2.73	2.68	0.68	0.67
Scharts	[1.95, 2.48, 2.97, 3.41]	2.73	2.70	0.68	0.68
FSMs	[1.06, 1.56, 2.06, 2.56]	1.81	1.81	0.45	0.45

Table 8g: Defuzzified and Normalized Values for Modularity Criterion

Technique	Aggregate trapezoidal no	Defuzzified values	1st order Statistical values	Normalized defuzzified values	Normalized 1st order values
Z	[2.75, 3.25, 3.59, 3.78]	3.42	3.34	0.86	0.84
Conc. Z	[0.25, 0.56, 0.88, 1.38]	0.72	0.77	0.18	0.19
R.Time Z	[0.33, 0.66, 1.00, 1.50]	0.83	0.87	0.21	0.22
UML	[1.06, 1.88, 2.38, 2.88]	2.13	2.05	0.53	0.51
P. nets	[1.19, 1.69, 2.19, 2.69]	1.94	1.94	0.49	0.49
Scharts	[1.25, 1.75, 2.25, 2.75]	2.00	2.00	0.50	0.50
FSMs	[0.16, 0.73, 1.00, 1.50]	0.87	0.85	0.22	0.21

Table 8h: Defuzzified and Normalized Values for Verifiability Criterion

Technique	Aggregate trapezoidal no	Defuzzified values	1st order Statistical values	Normalized defuzzified values	Normalized 1st order values
Z	[0.31, 0.72, 1.13, 1.56]	0.93	0.93	0.23	0.23
Conc. Z	[0.44, 0.84, 1.25, 1.75]	1.05	1.07	0.26	0.27
R.Time Z	[0.38, 0.81, 1.19, 1.69]	1.00	1.02	0.25	0.26
UML	[0.52, 0.95, 1.38, 1.88]	1.16	1.18	0.29	0.30
P.nets	[0.36, 0.77, 1.13, 1.63]	0.95	0.97	0.24	0.24
Scharts	[0.45, 0.88, 1.31, 1.81]	1.10	1.12	0.28	0.28
FSMs	[0.90, 1.25, 1.61, 1.90]	1.23	1.26	0.31	0.32

Table 8i: Defuzzified and Normalized Values for Executability Criterion

The ratings 0.64 (Table 8a) and 0.53 (Table 8e) of Petri nets in the concurrency and nondeterminism criteria respectively, indicate that Petri nets are suitable for describing concurrent, parallel, stochastic, and asynchronous system behaviour. The graphical structures (such as the Petri net graphs used in this thesis) in Petri nets are useful as a visual communication aid. The rating 0.57 (Table 8b) in the timing mechanism criterion shows that Petri nets allow designers to introduce timing constraints into a model of a system, which is achieved by associating a firing delay with a transition. While the ratings 0.17 (Table 8c) and 0.47 (Table 8f) in the user interfaces and complexity management criteria show that Petri nets is deficient in constructs for handling non-functional system requirements, but has constructs/attributes for decomposing specifications into modular components.

The rating 0.91 (Table 8c) of UML in the user interfaces criterion obviously portray that UML is exceptionally amenable to describing, capturing, and visualizing non-

functional and external interactions such as user interfaces, human-computer interaction, and system usability. The ratings 0.64 (Table 8a) and 0.47 (Table 8e) in the concurrency and nondeterminism criteria indicate that UML diagrams are suitable for modelling concurrent and distributed systems as well as capture dynamic properties of system behaviour. However, the variant of UML considered in this thesis has limited capability for describing temporal and real time constraints of system behaviour shown by the rating of 0.52 (Table 8b). Although various extensions to UML now exist for completely describing these temporal and real time constraints of system behaviour. However, a rating of 0.31 (Table 8d) in the abstract functionality criterion show that UML diagrams are deficient in describing low-level abstract functionalities of system behaviour. This deficiency is due to the absence of constructs premised on mathematics such as sets, predicate logic, description logic, etc in the UML notation.

Generally, FSMs are easy, flexible, and simple to use, resulting in ease of development and implementation. The rating 0.32 (Table 8i) and 0.47 (Table 8e) of FSMs in the executability and nondeterminism criteria show that FSMs representations can easily be translated into source code and can moderately capture nondeterministic system behaviour. However, the rating 0.22 (Table 8f) in the complexity management criterion indicates that FSMs are deficient in modelling systems with very many states, this deficiency is attributable to the states explosion problem which occurs when a system becomes too large and complex. The ratings 0.30, 0.23, (Table 8a, Table 8b) and 0.23 (Table 8c) in the concurrency, timing mechanism, and user interfaces criteria respectively, show that FSMs are deficicient in capturing concurrency, real time constraints of system behaviour as well as user interfaces. Also, FSMs are deficient in formal semantics constructs and have little expressive power shown by a rating of 0.23 (Table 8d) in the abstract functionality criterion.

The ratings 0.93, 0.84, (Table 8d, Table 8f) and 0.75 (Table 8g) of Z in the abstract functionality, complexity management, and modularity criteria respectively, indicate that Z is a very powerful tool for capturing abstract functionality of a system, composing modular, large, as well as complex specifications. Also, Z makes a designer to think

clearly about the system and hence gains a deeper understanding of the system being modelled. The combination of Z with informal descriptions in a specification enhances the readability of a system specification. In addition, Z has a well-defined syntax and semantics, structured mechanism via schema, tools for type checking and analyses (shown by the rating 0.86 (Table 8h) in the verifiability criterion), widely available documentation and resources. However, the ratings $0.19, 0.22, 0.11$, (Table 8e, Table 8a, Table 8c) and 0.11 (Table 8c) of Z in the nondeterminism, concurrency, timing mechanism, and user interfaces criteria respectively, indicate that Z is not well suited for describing non-functional requirements such as usability, reliability, user interfaces, size, and performance, as well as concurrency, real-time constraints, and dynamic behaviour of a system. Some extensions to Z, such as Real-Time Z (with a rating of 0.89 (Table 8b) in the timing constraint criterion) is now available to capture real-time constraints and Concurrent Z (with a rating of 0.84 (Table 8a) in the concurrency criterion) is available for capturing concurrency and parallel behaviour of systems. Also, Z does not have primitives for capturing real numbers; hence variables such as money can not be accurately described in Z.

The ratings $0.59, 0.47$, (Table 8a, Table 8b) and 0.41 (Table 8e) of Statecharts in the concurrency, timing mechanism, and nondeterminism criteria respectively, show that the variant of Statecharts considered in this research has limited capabilities for capturing dynamic, concurrent, and real time properties of system behaviour. Statecharts like FSMs are easy, flexible, and simple to use. Statecharts aid visualization and have tools for validating and verifying its specifications shown by the rating 0.50. The rating 0.52 (Table 8f) in the complexity management criterion shows that Statecharts are useful for describing not too large and complex systems, while a rating of 0.28 (Table 8i) in the executability metric indicates that its specifications is executable, this capability is however limited. However, the rating 0.28 (Table 8d) in the abstract functionality criterion clearly portray that Statecharts are deficient in constructs for capturing low-level abstract functionalities of system behaviour because it lacks mathematics-based constructs (such as sets and predicate logic) for representing these low-level functionalities.

5.5 Evaluation Summary

Table 9 shows a tabular summary of the evaluation results (the normalized defuzzified values) of the seven specification techniques for e-commerce systems.

	Z	Concurrent Z	Real-Time Z	UML	Petri nets	Statecharts	FSMs
Concurrency	0.22	0.84	0.46	0.64	0.64	0.59	0.30
Timing constraint	0.16	0.53	0.89	0.52	0.57	0.47	0.25
User interfaces	0.11	0.14	0.11	0.91	0.17	0.78	0.23
Abstract functionality	0.93	0.66	0.62	0.31	0.44	0.28	0.23
Nondeterminism	0.19	0.72	0.74	0.47	0.53	0.41	0.47
Complexity management	0.84	0.65	0.62	0.49	0.47	0.52	0.22
Modularity	0.75	0.71	0.70	0.68	0.68	0.68	0.45
Verifiability	0.86	0.18	0.21	0.53	0.49	0.50	0.26
Executability	0.23	0.26	0.25	0.29	0.24	0.28	0.32

Table 9: Evaluation Summary of The Specification Techniques.

From the evaluation summary, it is obvious that none of the seven specification techniques evaluated scored at least *average* in all the evaluation criteria. This observation underscores the need to develop an integrated specification technique that will possess all the requisite functionalities and attributes necessary to completely model the various components of an e-commerce system.

The use of fuzzy sets in classification or evaluation reduces imprecisions, ambiguity, vagueness, and subjectivity associated with crisp (non-fuzzy) methods. However, the results obtained by using the procedure outlined in this thesis differs marginally from the results obtained using first order statistics.

Chapter 6

Conclusions and Future Work

This chapter presents the conclusion of this thesis, a summary of the contributions made by this research work, and highlights possible research areas open for investigation in the near future.

This thesis presented the characteristics of e-commerce systems that make e-commerce systems complex and amenable to formal specification. In addition, the thesis discussed the various requirements necessary for a specification technique to completely specify all the components of an e-commerce system and classifies these requirements. Also, existing specification techniques that can be applied for specifying e-commerce systems were examined and a taxonomy provided for the techniques.

Furthermore, the five major specification techniques evaluated (the Z notation, Statecharts (top-level, hierarchical, and flat statecharts), UML (use case, sequence, collaboration, and activity diagrams), Petri net graphs, and finite state transition diagrams) were used to specify the *ordering process* (described in the pseudocode in Section 4.1) in a typical B2C e-commerce scenario to demonstrate the pragmatic nature of this thesis. Also, the specification techniques were evaluated using the evaluation criteria and the fuzzy logic evaluation framework developed in this thesis. Lastly, the evaluation results were discussed and a summary of the evaluation presented in tabular form.

The evaluation results indicate that none of the evaluated seven specification tech-

niques has all the attributes required to specify an e-commerce system at the appropriate level.

6.1 Summary of Contributions

In summary, this thesis contributes to knowledge by highlighting the requirements necessary for a specification technique to completely specify all the component parts of an e-commerce system. This highlight enables me to provide a classification for these requirements and provide a taxonomy for e-commerce specification techniques. Also, this thesis provides a fuzzy logic-based evaluation framework and a systematic and pragmatic evaluation of seven specification techniques for e-commerce systems. This evaluation will assist designers of e-commerce specification techniques with the functionalities required in any specification technique they are developing or hope to develop for capturing e-commerce applications.

6.2 Future Work

This thesis can be extended in future by:

i. Adopt a more comprehensive approach to this evaluation by investigating additional specification techniques such as model checking, CSP, RAISE, VDM, and emerging specification techniques that can be used to model e-commerce applications for their suitability or otherwise in specifying e-commerce systems. The inclusion of more specification techniques in a future evaluation will provide a broader perspective to developers of e-commerce applications in their choice of the specification technique they want to use in specifying e-commerce systems.

ii. The evaluation carried out in this thesis applies specifically to the B2C e-commerce domain. Future evaluations could be generic and applicable to other major categories of e-commerce systems such as the C2C and B2B e-commerce applica-

tion domains. This is underscored by the increasing popularity of the C2C e-commerce application doamins (especially various types of online auctions) and B2B e-commerce application domain (especially e-commerce hubs).

iii. The various specifications in this research were not translated into implementation. An implementation of the specifications with either the same or different programming languages might provide further insight into the relative strengths and weaknesses of the specification techniques. Also, implementing the specifications with an identical programming language might provide another perspective to the evaluation. The e-commerce systems derived from the implementations can be tested for scalability, fail-safety, dependability, security, ease of deployment, and correctness of operations.

iv. The simple fuzzy logic evaluation framework developed in this thesis can be extended into a complete generic, robust, and scalable model that can be used for evaluating the effectiveness, adequacy, and suitability of specification techniques for e-commerce systems and other emerging application domains. The proposed model which can be formulated a linear or integer programming optimization problem, could consist of the following variables:

   (a) the rating of each evaluation criterion by domain experts or specialists,

   (b) the weight of each evaluation criterion (these weightings can be determined either by directly assigning weights to each criterion or by comparing each criterion with one another so as to ascertain the relative weightings), and

   (c) the relative weights of each of the specialists based on the proficiency of the specialists and the years of experience in applying specification techniques in specifying e-commerce applications.

   Some of the constraints on the model could be that the sum of the weightings assigned to each evaluation criterion must be equal to one and the sum of the weightings assigned to the evaluation specialists must also be equal to one.

v. Since the evaluation results show that none of the seven techniques scored up to average in each of the evaluation criteria, the development of an integrated specification technique that will possess all the requisite functionalities and attributes necessary to specify all the parts of an e-commerce system is considerable in the future. The specification technique should have attributes for specifying both functional and non-functional requirements of system behaviour. Also, the specification technique should be capable of handling large and complex specifications and the specifications must be executable.

Bibliography

[1] J. R. Abrial, M.K.O. Lee, D. Neilson, P. N. Scharbach, and I. Sorenson. The B-Method. In S. Prehn and W. J. Toetenel, editors, *Formal Software Development, 4th International Symposium of VDM Europe*, volume 552 of *Lecture Notes in Computer Science*, pages 398–405, Noordwijkerhout, Holland, October 1991. Springer-Verlag.

[2] V. S. Alagar and Z. Li. A Rigorous Approach to Modelling and Analyzing E-commerce Architectures. In J. N. Oliveira and P. Zave, editors, *FME 2001*, volume 2021 of *LNCS*, pages 173–196, Berlin, Germany, 2001. Springer-Verlag.

[3] V. S. Alagar and K. Periyasamy. *Specification of Software Systems*. Springer, 2nd edition, 1998.

[4] M. Ardis, J. Chaves, L. J. Jagadeesan, P. Mataga, C. Puchol, M. Staskauskas, and J. von Olnhausen. A Framework for Evaluating Specification Methods for Reactive Systems. In *International Conference on Software Engineering*, pages 159–168, Seattle, USA, April 1995.

[5] M. Ardis, J. Chaves, L. J. Jagadeesan, P. Mataga, C. Puchol, M. Staskauskas, and J. von Olnhausen. A Framework for Evaluating Specification Methods for Reactive Systems. *IEEE Transactions on Software Engineering*, 22(6):378–389, June 1996.

[6] E. Astesiano, M. Broy, and G. Reggio. Algebraic Specification of Concurrent Systems. In E. Astesiano, H. Kreowski, and B. Krieg-Bruckner, editors, *Algebraic Foundations of Systems Specification*, chapter 13, pages 467–520. Springer Verlag, 1999.

[7] G. Barry and G. Gonthier. The ESTEREL synchronous programming language: Design, Semantics, and Implementation. *Science of Computer Programming*, 19:87–152, 1992.

[8] A. Bartelt and W. Lamersdorf. A Multi-criteria Taxonomy of Business Models in Electronic Commerce. In L. Fiege, G. Muhl, and U. G. Wilhelm, editors, *Second International Workshop on Electronic Commerce (WELCOM)*, volume 2232 of *Lecture Notes in Computer Science*, pages 193–205, Heidelberg, Germany, November 2001. Springer-Verlag.

[9] M. Benyoucef and R. K. Keller. An Evaluation of Formalisms for Negotiations in E-commerce. In P. G. Kropf, G. Babin, J. Plaice, and H. Unger, editors, *3rd International Workshop on Distributed Communities on the Web (DCW)*, volume 1830 of *Lecture Notes in Computer Science*, pages 45–54, Quebec, Canada, June 2000. Springer.

[10] D. Bjorner and C. Jones. The Vienna Development Method (VDM): The Meta-Language. In *The Vienna Development Method: The Meta-Language*, volume 61. Springer Verlag, 1978.

[11] D. Bolignano and G. Dyade. Towards the Formal Verification of Electronic Commerce Protocols. In *10th IEEE Computer Security Foundations Workshop*, pages 133–147, Rockport, USA, June 1997.

[12] T. Bolognesi and E. Brinksma. Introduction to the ISO Specification Language LOTOS. *Computer Network*, 14(1):25–59, 1987.

[13] J. Bowen and M. Hinchey. Communicating Sequential Processes. In *High-Integrity System Specification and Design*, chapter 5, pages 303–330. Springer Verlag, 1999.

[14] J. M. Bruel, A. Benzekri, and Y. Raymaud. Z and the Specification of Real-time Systems. In *7th International Conference on: Putting into Practice, Methods and Tools for Information System Design*, Nantes, France, October 1995.

[15] D. Burdett. *Internet Open Trading Protocol (IOTP) Version 1.0.* IETF Trade Working Group, 2000.

[16] M. J. Butler. CSP2B: A Practical Approach to Combining CSP and B. In *World Congress on Formal Methods*, volume 1708 of *LNCS*, pages 490–508, Toulouse, France, September 1999. Springer-Verlag.

[17] M. Cadoli, G. Ianni, L. Palopoli, A. Schaerf, and D. Vasile. NP-SPEC: An Executable Specification Language for Solving all Problems in NP. *Technical Report 13–00*, University of Roma, Italy, September 2001.

[18] D. Carrington, D. Duke, R. Duke, P. King, G.A. Rose, and G. Smith. Object-Z: An Object-Oriented Extension to Z. In S. Vuong, editor, *2nd International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols*, pages 281–296, Vancouver, Canada, December 1989. North-Holland.

[19] A. G. Cass, H. Lee, B. S. Lerner, and L. J. Osterweil. Formally Defining Coordination Processes to Support Contract Negotiation. *Technical Report UM-CS-1999-039*, University of Massachusetts, Amherst, MA, USA, June 1999.

[20] D. Chaum. Security Without Identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, October 1985.

[21] D. Chaum, A. Fiat, and M. Naor. Untraceable Electronic Cash. In *Advances in Cryptology*, pages 200–212. Springer-Verlag, July 1988.

[22] E. Ciapessoni, M. Negri, and D. Pieragostini. NETLAB: A Software Tool for Drawing and Validating Petri Nets. *Gesellschaft fur Informatik (GI) Special Interest Group on Petri Nets and Related System Models (Newsletter)*, 18:4–6, October 1984.

[23] E. Clarke, O. Grumberg, and D. Long. Model checking and abstraction. *ACM Transactions on Programming Languages and Systems*, 16(5):1512–1542, September 1994.

[24] M. Clavel, F. Duran, S. Eker, P. Lincoln, J. Meseguer N. Marti-Oliet, and J. F. Quesada. Maude: Specification and Programming in Rewriting Logic. *Theoretical Computer Science*, June 2001.

[25] R. S. Cost, Y. Chen, T. Finin, Y. Labrou, and Y. Peng. Modelling Agents Conversions with Coloured Petri Nets. In *Workshop on Specifying and Implementing Conversation Policies, Autonomous Agents*, Seattle,Washington, USA, May 1999.

[26] B. Cox, D. Tygar, and M. Sirbu. NetBill Security and Transaction Protocol. In *1st USENIX Workshop on Electronic Commerce*, pages 77–88, New York, USA, July 1995.

[27] D. Dubois and H. Prade. Approximate and Commonsense Reasoning: From Theory to Practice. In Z. W. Ras and M. Michalewicz, editors, *9th International Syposium on Methodologies for Intelligent Systems*, volume 1079 of *Lecture Notes in Computer Science*, pages 19–33, Zakopane, Poland, June 1996. Springer.

[28] R. Duke, G. Rose, and G. Smith. Object-Z: A Specification Language Advocated for the Description of Standards. *Computer Standards and Interfaces*, 17(5–6):511–533, 1995.

[29] S. A. Ehikioya. *Specification of Transaction System Protocols*. PhD thesis, University of Manitoba, Winnipeg, Canada, September 1997.

[30] S. A. Ehikioya. A Formal Design Framework for Electronic Commerce Transactions. In *Intl. Conference on Advances in Infrastructure for Electronic Business, Science and Education on the Internet*, pages 1–13, L'Aquila, Italy, July–August 2000.

[31] S. A. Ehikioya. A Formal Characterization of Electronic Commerce Transactions. *Intl. Journal of Computer and Information Sciences*, 2(3):97–117, September 2001.

[32] S. A. Ehikioya. A Formal Perspective to Modelling Electronic Commerce Transactions. *Colombian Journal of Computation*, 2(2):21–40, December 2001.

[33] S. A. Ehikioya and K. Barker. A Formal Specification Strategy for Electronic Commerce. In *International Database Engineering and Applications Symposium (IDEAS)*, pages 201–210, Montreal, Canada, August 1997. IEEE Computer Society.

[34] S. A. Ehikioya and K. Hiebert. A Formal Model of Electronic Commerce. In *1st Intl. Conference on on Software Engineering, Networking, and Parallel and Distributed Computing*, pages 1–9, Champagne-Ardenne, France, May 2000.

[35] A. Evans. Specifying and Verifying Concurrent Systems Using Z. In *Formal Methods Europe*, pages 366–380, Barcelona, Spain, October 1994.

[36] Failures Divergence Refinement (FDR), User Manual and Tutorial Version 1.4. *Formal Sytems (Europe)*, 1994.

[37] P. Fettke and P. Loos. Classification of reference models: a methodology and its application. *Information Systems and e-business Management*, 1:35–53, 2003.

[38] C. Fischer. CSP-OZ: A Combination of Object-Z and CSP. In Howard Bowman and John Derrick, editors, *2nd IFIP Workshop on Formal Methods for Open Object-Based Distributed Systems*, pages 423–438, Canterbury, UK, 1997. Chapman and Hall.

[39] A. Franz, P. Sties, and S. Vogel. Formal Specification of E-commerce Applications: An Interdisciplinary Approach. In *6th (INFORMS) Conference on Information Systems and Technology*, pages 314–332, Miami, USA, November 2001.

[40] X. Fu, T. Bultan, and J. Su. Formal Verification of E-Services and Workflows. In *Workshop on Web Services, E-business, and the Semantic Web*, volume 2512 of *LNCS*, pages 188–202, Toronto, Canada, May 2002.

[41] N.E. Fuchs. Hoare Logic, Executable Specifications and Logic Programs. *Structured Programming*, 13(3):129–135, 1992.

[42] GDPro Reference Manual: Windows Platform Version 5.0. *Advanced Software Technologies, Inc., USA*, 2000.

[43] J. Goguen, T. Winkler, J. Meseguer, K. Futatsugi, and J. Jouannaud. Introducing OBJ. In Joseph Goguen, editor, *Applications of Algebraic Specification using OBJ*. Cambridge Press, 1993.

[44] H. Gomaa. *Designing Concurrent, Distributed, and Real-Time Applications with UML*. Addison-Wesley, 2nd edition, 2000.

[45] The RAISE Language Group. *Rigorous Approach to Industrial Software Engineering (RAISE) Specification Language*. Prentice-Hall International (UK), 1992.

[46] J. Guttag and J. Horning, editors. *LARCH: Languages and Tools for Formal Specification*. Springer Verlag, 1993.

[47] D. Harel. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, 8(3):231–274, June 1987.

[48] D. Harel, H. Lachover, A. Naamad, A. Pnueli, M. Politi, R. Sherman, A. Shtull-Trauring, and M. B. Trakhtenbrot. STATEMATE: A Working Environment for the Development of Complex Reactive Systems. *Software Engineering*, 16(4):403–414, 1990.

[49] N. Heintze, D. Tygar, J. M.Wing, and C. Wong. Model Checking Electronic Commerce Protocols. In *2nd USENIX Workshop on Electronic Commerce*, pages 147–164, Oakland, USA, November 1996.

[50] A. Hoare. Communicating Sequential Processes. *Comunications of the ACM*, 21(8):666–677, 1978.

[51] A. Hull. What Does Industry Need from Formal Specification? In *2nd IEEE Workshop on Industrial-Strength Formal Specification Techniques*, pages 2–9, Boca Raton, USA, October 1998.

[52] F. Jahanian and A. Mok. Modechart: A Specification Language for Real-Time Systems. *IEEE Transactions On Software Engineering*, 20(12):933–947, December 1994.

[53] S. Kaplan and M. Sawhney. B2B E-commerce Hubs: Towards a Taxonomy of Business Models. *Harvard Business Review*, December 1999.

[54] S. Kaplan and M. Sawhney. Let's Get Vertical. *Business*, 2, September 1999.

[55] G. J. Klir. Fuzzy Logic: Unearthing its meaning and significance. *IEEE Potentials*, 14(4):10–15, October/November 1995.

[56] J. C. Knight, C. L. DeJong, M. S. Gibble, and L. G. Nakano. Why Are Formal Methods Not Used More Widely? In *Fourth National Aeronautics and Space Administration (NASA) Formal Methods Workshop*, Hampton, Virginia, USA, September 1997.

[57] L. M. Kristensen, S. Christensen, and K. Jensen. The Practitioner's Guide to Coloured Petri Nets. *International Journal on Software Tools for Technology Transfer*, 2(2):98–132, 1998.

[58] M. Kumar and S. I. Feldman. Business Negotiations on the Internet. In *Internet Summit (INET) '98*, Geneva, Switzerland, July 1998.

[59] K. C. Laudon and C. G. Traver. *E-commerce: Business, Technology and Society.* Addison-Wesley, 2002.

[60] X. Li, Z. Liu, and Z. Guo. A Formal Design of Online Ticketing System in UML. In *8th Asia-Pacific Software Engineering Conference*, pages 259–266, Macau, China, December 2001.

[61] P. Linz. *An Introduction to Formal Languages and Automata.* Jones and Bartlett, 3rd edition, 2001.

[62] LOTOS: A formal description technique based on the temporal ordering of observational behaviour. *International Standard (ISO/IEC) 8807*, September 1988.

[63] G. Lowe. Towards a Completeness Result for Model Checking of Security Protocols. *Computer Security*, 7(2–3):89–146, 1999.

[64] B. Mahony and J. S. Dong. Timed Communicating Object Z. *IEEE Transactions on Software Engineering*, 26(2):150–177, February 2000.

[65] S. Mellor and P. Ward. *Structured Development for Real-Time Systems.* Prentice-Hall, 1985.

[66] K. Mills. An Experimental Evaluation of Specification Techniques for Improving Functional Testing. *Systems and Software*, 32(1):83–95, February 1996.

[67] V. B. Misic and J. L. Zhao. Reference Models for Electronic Commerce. In *9th Hong Kong Computer Society Database Conference - Database and Electronic Commerce*, pages 199–209, Hong Kong, May 1999.

[68] B. Oestereich. *Developing Software with UML: Object Oriented Analysis and Design in Practice.* Addison-Wesley, 2nd edition, 2002.

[69] OMG Negotiation Facility. *Object Management Group, Inc.*, March 1999.

[70] C. Ouyang, L. M. Kristensen, and J. Billington. A Formal and Executable Specification of the Internet Open Trading Protocol. In K. Bauknecht, A.M. Tjoa, and G. Quirchmayr, editors, *3rd International Conference on E-commerce and Web Technology*, volume 2455 of *LNCS*, pages 377–387, Aix-en-Provence, France, September 2002. Springer-Verlag.

[71] C. Ouyang, L. M. Kristensen, and J. Billington. A Formal Service Specification for the Internet Open Trading Protocol. In J. Esparza and C. Lakos, editors, *23rd International Conference on Application and Theory of Petri Nets*, volume 2360 of *LNCS*, pages 352–373, Adelaide, Australia, June 2002. Springer-Verlag.

[72] S. Owre, J. M. Rushby, and N. Shankar. PVS: A Prototype Verification System. In D. Kapur, editor, *11th International Conference on Automated Deduction (CADE)*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 748–752, Saratoga, New York, USA, June 1992. Springer-Verlag.

[73] M. Papa, O. Bremer, J. Hale, and S. Shenoi. Formal Analysis of E-commerce Protocols. In *5th International Symposium on Autonomous Decentralized Systems*, pages 19–28, Dallas, Texas, USA, March 2001. IEEE Computer Society.

[74] O. Pastor, S. M. Abrahão, and J. J. Fons. Building E-commerce Applications from Object-Oriented Conceptual Models. *ACM SIGecom Exchanges (Newsletter)*, 2.2(1):28–36, April–May 2001.

[75] M. R. Patra and R. Moore. A Formal Model of an Agent-mediated Electronic Market. In *Design of Information Infrastructures Systems for Manufacturing (DIISM) Conference*, pages 15–17, Melbourne, Australia, November 2000.

[76] A. Pereira, M. Song, G. Gorgulho, W. Meira Jr., and S. Campos. A Formal Methodology to Specify E-commerce Systems. In C. George and H. Miao, editors, *4th International Conference on Formal Engineering Methods*, volume 2495 of *LNCS*, pages 180–191, Shanghai, China, October 2002. Springer-Verlag.

[77] A. Pereira, M. Song, G. Gorgulho, W. Meira Jr., and S. Campos. FormalCAFE: An E-commerce System's Case Study. In *5th International Conference on Eletronic Commerce Research*, Montreal, Canada, October 2002.

[78] K. Periyasamy and V.S. Alagar. Adding Real-Time Filters to Object-Oriented Specification of Time Critical Systems. In *2nd IEEE Workshop on Industrial-Strength Formal Specification Techniques*, pages 28–39, Boca Raton, Florida, USA, October 1998. IEEE Computer Society.

[79] J. L. Peterson. *Petri Net Theory and the Modelling of Systems*. Prentice-Hall, 1981.

[80] W. E. Rajput. *E-commerce Systems Architecture and Applications*. Artech House, 2000.

[81] B. Randell. Dependability — A Unifying Concept. In *1998 Computer Security, Dependability and Assurance Conference*, pages 16–25, York, England, July 1998.

[82] I. Ray and I. Ray. Failure Analysis of an E-commerce Protocol using Model Checking. In *2nd International Workshop on Advanced Issues of E-commerce and Web-based Information Systems*, pages 176–183, Milipitas, California, USA, June 2000. IEEE Computer Society.

[83] W. Reisig. *Petri Nets: An Introduction*. Springer, 1985.

[84] A. W. Rohm, G. Pernul, and G. Herrmann. Modelling Secure and Fair Electronic Commerce. In *14th Annual Computer Security Applications Conference*, pages 155–166, Scottsdale, Arizona, USA, December 1998. IEEE Computer Society.

[85] M. Saaltink. *The Z/EVES User's Guide*. ORA Canada, 1997.

[86] S. Schneider. An Operational Semantics for Timed CSP. *Information and Computation*, 116(2):193–213, 1995.

[87] M. Sirbu and D. Tygar. NetBill: An Internet Commerce System Optimized for Network Delivered Services. *IEEE Personal Communications*, pages 34–39, August 1995.

[88] G. Smith and I. Hayes. Towards Real-Time Object-Z. In Keijiro Araki, Andy Galloway, and Kenji Taguchi, editors, *1st International Conference on Integrated Formal Methods*, pages 49–65, York, UK, June 1999.

[89] G. Smith and I. Hayes. Structuring Real-Time Object-Z. In W. Grieskamp, T. Santen, and B. Stoddart, editors, *2nd International Conference on Integrated Formal Methods*, pages 97–115, Dagstuhl Castle, Germany, November 2000.

[90] C. Snook and M. Butler. Using a Graphical Design Tool for Formal Specification. In *13th Workshop of the Psychology of Programming Interest Group*, pages 311–321, Bournemouth, UK, April 2001.

[91] I. Sommerville. *Software Engineering*. Pearson Education, 6th edition, 2001.

[92] M. Song and S. Campos. A Framework to Design and Verify E-Commerce Systems. Available at http://www.dcc.ufmg.br, 2002.

[93] M. Song and S. Campos. UML-CAFE: A Process to Specify and Verify E-Commerce Systems. In *Sixth International Conference on Electronic Commerce Research - (ICECR-6)*, Dallas, Texas, USA, October 2003.

[94] M. Song, A. Peirara, G. Gorgulho, W. Meira Jr., and S. Campos. Model Checking Patterns for E-commerce Systems. In *1st Seminar on Advanced Research In Elecronic Business*, LNCS, Rio de Janeiro, RJ, Brazil, November 2002.

[95] M. Song, A. Pereira, F. Lima, G. Gorgulho, W. Meira Jr., and S. Campos. A Software Engineering Process to Specify and Verify E-Commerce Systems. In B. Al-Ani, H. R. Arabnia, and Y. Mun, editors, *International Conference on Software Engineering Research and Practice*, volume 1, pages 419–425, Las Vegas, Nevada, USA, June 2003. CSREA Press.

[96] Specification and Description Language (SDL). *Intl. Telecommunications Union – Telecommunications (ITU-T) Recommendation Z.100*, 1993.

[97] J. M. Spivey. *The Z notation: A reference manual.* Prentice Hall International, 2nd edition, 1992.

[98] C. Sühl. *An Integration of Z and timed CSP for Specifying Real-Time Embedded Systems*. PhD thesis, Technical University of Berlin, Berlin, Germany, December 2002.

[99] D. Tygar. Atomicity in Electronic Commerce. In *15th ACM Symposium on Principles of Distributed Computing*, pages 8–26, Philadelphia, USA, May 1996.

[100] Unified Modelling Language (UML) Specification, Version 1.5. *Object Management Group, Inc.*, March 2003. Available at http://www.omg.org/uml/.

[101] F. van Harmelen, M. Aben, F. Ruiz, and J. van de Plassche. Evaluating a Formal KBS Specification Language. *IEEE Expert*, 11(1):56–62, February 1996.

[102] F. van Harmelen and J. Balder. $(ML)^2$ — A formal language for KADS models of expertise. *Knowledge Acquisition Journal*, 4(1):127–161, 1992.

[103] G.V. Vijayaraghavan. A Taxonomy of E-commerce Risks and Failures. Master's thesis, Florida Institute of Technology, Melbourne, FL, USA, May 2003.

[104] F. Wagner. VFSM executable specification. In *IEEE International Conference on Computer System and Software Engineering*, pages 226–231, The Hague, Netherlands, 1992.

[105] W. Wang, Z. Hidvégi, A. D. Bailey, and A. B. Whinston. E-Process Design and Assurance Using Model Checking. *IEEE Computer Society*, 33(10):48–53, October 2000.

[106] R. Wieringa. A Survey of Structured and Object-oriented Software Specification Methods and Techniques. *ACM Computing Surveys*, 30(4):459–527, December 1998.

[107] J. M. Wing. A Specifier's Introduction to Formal Methods. *IEEE Computer*, 23(9):8–24, September 1990.

[108] J. Woodcock and J. Davies. *Using Z: Specification, Refinement and Proof*. Prentice Hall, 1996.

[109] P. Yolum and M. P. Singh. Commitment-based Enhancement of E-commerce Protocols. In *9th IEEE Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 278–283, Gaithersburg, Maryland, USA, March 2000. IEEE Computer Society.

[110] L. A. Zadeh. Fuzzy logic. *IEEE Computer*, 21(4):83–93, April 1988.

[111] A. Zamulin. Specification of Dynamic Systems by Typed Gurevich Machines. In Z. Bubnicki and A. Grzech, editors, *13th International Conference on System Science*, pages 160–167, Wroclaw, Poland, September 1998.

APPENDIX

Questionnaires for e-commerce service providers

1. Which e-commerce application domain does your activities fall into?
   a) Business-to-Business
   b) Business-to-Consumer
   c) Consumer-to-Consumer
   d) Online Auction
   e) Others (please specify)
2. How is your e-commerce applications developed ?
   a) In-house by Software Engineers
   b) Outsourced to External Consultants
   c) Combination of a) and b)
   d) Bought off the shelf
   e) Others (please specify)
3. If your answer to question 2 is a), please answer this question, what approach do your software engineers adopt in developing e-commerce applications ?
   a) Informal Approach
   b) Formal Approach
   c) Combination of Formal and Informal Approach
4. If your answer to question 3 is a), has any of your users/customers reported any error(s) or fault(s) to you ?
   a) Yes
   b) No
   c) Can't say specifically
5. If your answer to question 4 is *Yes*, what do you think was the cause(s) of the error(s) ?
   a) E-commerce application not exhaustively tested
   b) Inadequate post-implementation support
   c) Not using formal techniques in developing the e-commerce application
   d) Others (please specify)
6. If your answer to question 3 is b), which of these formal approach did you adopt ?
   a) Formal specification
   b) Formal verification

c) Combination of a) and b)

d) Others (please specify)

7. If your answer to question 6 is a), which specification technique did you employ ?

   a) Natural Language

   b) Pseudocodes

   c) Model-based formal specification techniques

   d) Algebraic-based formal specification techniques

   e) Graphical-based formal specication techniques

   f) Objected-oriented formal specification techniques

   g) If any of these techniques were combined, specify the combination

   h) Others (please specify)

8. If a natural language was used, state the natural language employed.

9. If pseudocodes were used, state the language in which the pseudocodes were constructed.

10. If model-based formal specification techniques were used, state the model-based formal specification language used.

11. If algebraic-based formal specification techniques were used, state the algebraic-based specification language used.

12. If graphical-based formal specication techniques were used, state the graphical-based specification language used.

13. If object-oriented formal specification techniques were used, state the object-oriented formal specification language used.

14. If a combination of any of the techniques in 7 a) to 7 g) was used, state the specification language used.

15. State the actual formal specification language used in formally specifying your e-commerce application.

16. Do you think that the application of formal specification in developing your e-commerce application did enhance the reliability and dependability of your e-commerce application ?

   a) Yes

   b) No

   c) Can't say exactly

17. What do you think are the major strength(s) of the formal specification language used in formally specifying your e-commerce application. Tick all the applicable strength(s).

   a) Supports concurrency of the e-commerce application

   b) Supports timing constraint and real-time properties

c) Can model non-functional requirements, such as user interfaces and usability of the e-commerce application
d) Easy to learn
e) Easy to use
f) Sufficient documentation available
g) Can model abstract functionalities of the e-commerce application
h) Supports large and complex systems
i) Specifications can easily be translated to source code
j) Other strength(s) (please specify)

18. What do you think are the major weakness(es) of the formal specification language used in formally specifying your e-commerce application. Tick all the applicable weakness (es).
a) Difficult to use
b) Difficult to learn
c) Insufficient documentation
d) Deficient in constructs to model non-functional requirements, such as user interfaces and usability of the e-commerce application
e) Deficient in constructs to model abstract functionalities of the e-commerce application
f) Deficient in constructs to model concurrency of the e-commerce application
g) Deficient in constructs to model real time behaviour of the system
h) Specifications can't be translated to source code
i) Other weakness(es) (please specify)

19. Do you have any experience using any other formal specification language ?
a) Yes
b) No

20. If your answer to question 19 is Yes, state the formal specification language, identify the strength and weaknesses of this language using the factors in Questions 17 and 18.

21. Based on the strength(es) and weakness(es) identified in Questions 17 and 18 and other information at your disposal, list the attributes you expect a formal specification language for formally specifying e-commerce applications should have.

Questionnaires for Industrial and Academic Researchers

1. What level of education do you have in Computer Science? Choose all applicable qualifications and current programme enrolled in.

a) BSc or equivalent qualification in Computer Science or a closely related field

b) BSc or equivalent qualification in other fields

c) MSc or equivalent qualification in Computer Science or a closely related field

d) MSc or equivalent qualification in other fields

e) PhD or equivalent qualification in Computer Science or a closely related field

f) PhD or equivalent qualification other fields

g) Other qualifications (please specify)

2. How long have you been in software engineering ? Tick the applicable

a) 0 – 2 years

b) 2 – 5 years

c) 5 – 10 years

d) 10 – 15 years

e) 15 years and above

3. How long you have you been involved in developing e-commerce applications ?

a) less than 1 year

b) between 1 year and 2 years

c) between 2 years and 4 years

d) over 4 years

4. How long have you being involved in formal specification ?

a) 0 – 2 years

b) 2 – 5 years

c) 5 – 10 years

d) 10 – 15 years

e) 15 years and above

5. How many formal specification methods are you very proficient in ?

a) Between 1 and 2

b) Between 2 and 5

c) Between 5 and 10

d) 10 and above

6. State the formal specification method(s) you have used before. Answer

questions 7 – 21 for each method stated.

7. Is there formal syntax and semantics for the formal specification method(s) stated in question 6 ?
   a) Yes
   b) No
8. Does the structure of the formal specification method(s) stated in question 6 support different levels of abstraction ?
   a) Yes
   b) No
9. Do sample formal specifications of e-commerce applications exist in published literature ?
   a) Yes
   b) No
10. Do sample formal verifications based on the specifications in Question 9 exist in published literature ?
   a) Yes
   b) No
11. Can the specification method(s) represent numeric and literal constants ?
   a) Yes
   b) No
12. Can the specification method(s) represent time ?
   a) Yes
   b) No
13. Do the specification method(s) support concurrency ?
   a) Yes
   b) No
14. Do the specification method(s) support timing constraint and real-time properties?
   a) Yes
   b) No
15. Can the specification method(s) model abstract functionalities of an e-commerce application?
   a) Yes
   b) No
16. Can the specification method(s) model very large and complex e-commerce applications?
   a) Yes
   b) No
17. Can specifications in the formal specification method(s) be easily translated into source code?

a) Yes

b) No

18. Do the specification method(s) have tools for verifying and validating their specifications?

a) Yes

b) No

19. Do the specification method(s) have constructs modelling non-functional requirements such as user interfaces and usability?

a) Yes

b) No

20. State the major strength(s) of the formal specification method(s).

21. State the major weakness (es) of the formal specification method(s).

22. Have you combined more than one specification method to model an e-commerce transaction before?

a) Yes

b) No

23. If your answer to question 22 is Yes, did you encounter any difficulties or complications ?

a) Yes

b) No

24. State the difficulties or complications encountered in Question 2 .

25. List any other attribute not mentioned in this questionnaire you expect a formal specification language for formally specifying e-commerce applications should have.

26. Do you believe that using an integrated formal specification method to model all components e-commerce transactions can increase the reliability and dependability of e-commerce applications?

a) Yes

b) No