

Domain Analysis of E-Commerce Systems Using Feature-Based Model Templates

by

Sean Quan Lau

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2006

© Sean Quan Lau 2006

I hereby declare that I am the sole author of this thesis.

I authorize the University of Waterloo to lend this thesis to other institutions or individuals for the purpose of scholarly research

Signature

I further authorize the University of Waterloo to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Signature

Abstract

The pervasiveness and criticality of software applications in modern times have resulted in the demand for highly customized, high quality products in a timely and cost-efficient manner. Model-Driven Software Product Lines (MDSPL) is an approach to software development which allows developers to automatically build such products based on configuration knowledge and reusable assets. A product line is a group of related products that can be built from a common set of assets. The approach relies on two activities: 1) Domain Engineering, where features in the product line are scoped and reusable assets are built, and 2) Application Engineering, where individual products are built from the reusable assets. The MDSPL approach is supported by feature models and feature-based model templates. Feature models are a modeling notation used to represent the variability in a system family and describe all valid configurations. Feature-based model templates describe models for all valid products in a product line and are parameterized with feature configurations.

In this work, we develop an example, inspired by a realistic application, of a product line of Business-to-Consumer systems, which is used to demonstrate the viability of the approach on a real world scenario. In addition, we analyze our experience with the approach in order to produce guidelines for users of the modeling approach and recommendations for future improvements of the approach and the tools. The key recommendations include proposals to 1) investigate the representation of ordering in feature models and activity diagram model templates, 2) extend the existing work on the semantics of class and activity diagram model templates, and 3) add mechanisms to address annotation consistency issues.

Acknowledgements

First and foremost, I would like to thank my supervisor, Dr. Krzysztof Czarnecki, for his guidance and support during these past two years. I am extremely grateful for the opportunity he has given me to work with him in the Generative Software Development lab at the University of Waterloo. I would also like to thank my thesis readers, Dr. Kostas Kontogiannis and Dr. Paul Ward, whose invaluable comments and advice have helped improve the quality of my thesis.

I must further express my deep gratitude to Barry Pekilis, whose immense support throughout the past year has been instrumental in the completion of this thesis. As well, I would like to acknowledge the numerous discussions with Michał Antkiewicz in evaluating the models contained in this thesis. I would also like to acknowledge the support from of the rest of my colleagues in the Software Engineering group.

It is impossible for me to fully express the extent of my appreciation and the gratitude I owe to my mom, dad, grandmother, brother, aunts & uncles, and Jacquelyn, whose support and belief in me helped me throughout the trials and tribulations of this MASc thesis.

Finally, I would like to thank our department graduate secretary, Wendy Boles, who remained patient through my barrage of questions and paperwork.

I gratefully acknowledge the financial support provided for this research effort by the University of Waterloo's Faculty of Engineering.

Table of Contents

Chapter 1: Introduction	1
1.1 Generative Software Development	2
1.1.1 System Families	2
1.1.2 Development Methodology: Domain and Application Engineering	2
1.1.3 Reusable Assets.....	3
1.2 Model-Driven Software Product Lines.....	4
1.2.1 Model-Driven Software Development	4
1.2.2 Feature Models	4
1.2.3 Feature-Based Model Templates.....	5
1.3 E-Commerce Systems.....	5
1.4 Research Overview	6
1.5 Research Contribution.....	6
1.6 Thesis Organization	7
1.7 Summary	7
Chapter 2: Background.....	8
2.1 Unified Modeling Language	8
2.1.1 Class Diagrams	9
2.1.2 Activity Diagrams.....	9
2.2 Feature Models	11
2.2.1 Configuration.....	12
2.2.2 Binding Time.....	13
2.2.3 Tool Support	14
2.3 Feature-Based Model Templates.....	15
2.3.1 Tool Support	17
2.4 Summary	17
Chapter 3: Research Methodology.....	18
3.1 Domain Analysis of E-Commerce Systems	18
3.2 Applying the MDSPL Approach	20
3.3 Evaluation of the Approach.....	21
3.4 Summary	22
Chapter 4: E-Commerce Domain Analysis: Store Front.....	23
4.1 Home Page	23
4.1.1 Static Content	24
4.1.2 Dynamic Content	24
4.2 Registration	25
4.2.1 Registration Enforcement.....	26
4.2.2 Registration Information	27
4.2.3 User Behaviour Tracking Information.....	29
4.3 Catalog.....	29
4.3.1 Product Information	30

4.3.1.1	Product Types	30
4.3.1.2	Basic Information	31
4.3.1.3	Detailed Description	31
4.3.1.4	Warranty Information	32
4.3.1.5	Customer Reviews	32
4.3.1.6	Associated Assets	32
4.3.1.7	Product Variants	34
4.3.1.8	Size	34
4.3.1.9	Weight	34
4.3.1.10	Availability	35
4.3.1.11	Custom Fields	35
4.3.2	Categories	35
4.3.3	Multiple Catalogs	36
4.3.4	Searching	36
4.3.5	Browsing	37
4.3.6	Custom Views	38
4.4	Wish Lists	38
4.5	Buy Path	40
4.5.1	Shopping Cart	41
4.5.2	Checkout	42
4.5.2.1	Checkout Type	43
4.5.2.2	Shipping Options	44
4.5.2.3	Taxation Options	45
4.5.2.4	Payment Options	48
4.5.3	Order Confirmation	49
4.6	Customer Service	50
4.6.1	Question and Feedback Forms	50
4.6.2	Product Returns	51
4.6.3	Order Status Viewing	52
4.6.4	Shipment Status Tracking	52
4.7	User Behaviour Tracking	53
4.7.1	Behaviour Tracked	53
4.8	Summary	54

Chapter 5: E-Commerce Domain Analysis: Business Management ...55

5.1	Order Management	55
5.1.1	Physical Goods Fulfillment	57
5.1.1.1	Warehouse Management	57
5.1.1.2	Shipping	58
5.1.2	Electronic Goods Fulfillment	59
5.1.2.1	File Repository	60
5.1.2.2	License Management	60
5.1.3	Services Fulfillment	60
5.2	Targeting	61
5.2.1	Targeting Criteria	62
5.2.2	Targeting Mechanisms	64
5.2.2.1	Advertisements	65
5.2.2.2	Discounts	67
5.2.2.3	Sell Strategies	69

5.2.2.4	Display and Notification	71
5.2.2.5	Campaigns.....	72
5.3	Affiliates	73
5.4	Inventory Tracking	73
5.5	Procurement.....	75
5.6	Reporting and Analysis.....	76
5.7	External Systems Integration	77
5.8	Administration.....	79
5.8.1	Content Management.....	79
5.8.2	Store Administration.....	80
5.9	Summary	81

Chapter 6: Model Template Descriptions82

6.1	Class Diagram Model Templates	82
6.1.1	Store Front Entity Model	82
6.1.2	Services Model	88
6.2	Activity Diagram Model Templates.....	89
6.2.1	StoreFront Activity.....	90
6.2.2	FindProduct Activity.....	91
6.2.3	SearchProduct Activity	92
6.2.4	SelectProductFromCatalog Activity	94
6.2.5	SelectProductFromWishlist Activity	95
6.2.6	CheckoutItems Activity	97
6.2.7	CreateOrder Activity	101
6.2.8	OrderProducts Activity	102
6.2.9	ProcessOrder Activity.....	103
6.2.10	CheckOrderStatus Activity	105
6.2.11	RefundOrder Activity	106
6.2.12	RegisterWithTheStore Activity.....	107
6.2.13	UpdatePersonalProfile Activity.....	108
6.2.14	CreateQuickCheckoutProfile Activity.....	110
6.2.15	ResetPassword Activity.....	111
6.2.16	CreateWishList Activity	112
6.2.17	SendWishList Activity	113
6.3	Summary	114

Chapter 7: Evaluation of the Feature-Based Model

Template Approach 115

7.1	Analysis of Variability Modeling in Feature Models	115
7.2	Analysis of Annotations in Model Templates	118
7.2.1	Class Diagram Model Templates	119
7.2.1.1	Store Front Entity Model.....	119
7.2.1.2	Service Model.....	121
7.2.2	Activity Diagram Model Templates.....	121
7.3	Candidate Modeling Guidelines.....	124
7.3.1	Applying Binding Time Analysis to Feature Modeling.....	124
7.3.2	Modeling Feature Groups in Model Templates.....	125
7.3.2.1	Modeling Feature Groups in Class Diagram Model Templates.....	126

7.3.2.2	Modeling Feature Groups in Activity Diagram Model Templates	127
7.3.3	Applying MetaExpressions to Increase Conciseness	128
7.4	Recommendations	130
7.4.1	Variability in the Ordering of Features and Model Elements.....	130
7.4.2	Additional Semantics for UML Class Diagram Model Templates	131
7.4.3	Additional Semantics for UML Activity Diagram Model Templates	132
7.4.3.1	IPCs for Data Stores and Generic Data Nodes	133
7.4.3.2	Branch Annotations Semantics	133
7.4.3.3	Automatic Flow Type Correction	134
7.4.4	Annotation Consistency Issues	135
7.5	Summary	139
Chapter 8: Conclusion.....		140
8.1	Summary	140
8.2	Future Work	143
8.3	Closing Remarks.....	144
Appendix A: Additional Constraints in the E-Shop Feature Model.....		145
Appendix B: Annotation Analysis Data.....		151
References.....		155

List of Tables

Table 2.1: Feature Modeling Notation in fmp	14
Table 4.1: Behaviour Analysis of Cart Saved After Session Feature.....	42
Table 6.1: Activity Categories	90
Table A.1: Additional Constraints	146
Table B.1: Class Diagram (Store Front Entity Model) Annotation Analysis	151
Table B.2: Class Diagram (Service Diagram) Mapping Analysis	152
Table B.3: Activity Diagram Annotation Analysis	153

List of Illustrations

Figure 1.1: The Problem and Solution Space Pattern.....	3
Figure 2.1: Labeled Class Diagram Rendered in RSM.....	9
Figure 2.2: Class Diagram with Stereotypes Rendered in RSM	9
Figure 2.3: Labeled Activity Diagram Rendered in RSM	10
Figure 4.1: Store Front Feature	23
Figure 4.2: Home Page Feature.....	24
Figure 4.3: Registration Feature.....	26
Figure 4.4: Registration Enforcement Feature	26
Figure 4.5: Registration Information Feature	27
Figure 4.6: Catalog Feature.....	30
Figure 4.7: Product Information Feature	31
Figure 4.8: Associated Assets Features.....	33
Figure 4.9: Categories Feature	35
Figure 4.10: Searching Feature.....	36
Figure 4.11: Browsing Feature.....	37
Figure 4.12: Custom Views Feature.....	38
Figure 4.13: Wishlists Feature.....	39
Figure 4.14: Buy Path and Shopping Cart Features	40
Figure 4.15: Checkout and Checkout Type Features.....	43
Figure 4.16: Shipping Options Feature	44
Figure 4.17: Taxation Options Feature	46
Figure 4.18: Payment Options Feature	48
Figure 4.19: Order Confirmation Feature	50
Figure 4.20: Customer Service Feature.....	51
Figure 4.21: User Behaviour Tracking Feature.....	53
Figure 5.1: Business Management Feature	55
Figure 5.2: Order Management and Fulfillment Features	56
Figure 5.3: Physical Goods Fulfillment Feature	57
Figure 5.4: Targeting Feature	62
Figure 5.5: Targeting Criteria Feature.....	63
Figure 5.6: Targeting Mechanisms Feature	65
Figure 5.7: Advertisements Feature and Subfeatures.....	65
Figure 5.8: Discounts Feature	67
Figure 5.9: Sell Strategies Feature	70
Figure 5.10: Display and Notification Feature.....	71
Figure 5.11: Affiliates Feature.....	73
Figure 5.12: Inventory Tracking Feature	74
Figure 5.13: Procurement Feature.....	75
Figure 5.14: Reporting and Analysis Feature	76
Figure 5.15: External Systems Integration Feature	78
Figure 5.16: Administration Feature.....	79
Figure 6.1: EShopArtifact, Catalog, Category, Product and Money Classes	83
Figure 6.2: EShopArtifact, Product, and Asset Classes	84
Figure 6.3: Customer and BillingInformation Classes.....	85
Figure 6.4: Order, ShoppingCart and Wishlist Classes.....	86
Figure 6.5: Tax Rule Classes	87
Figure 6.6: EShop Interactions with Warehouse, Payment Gateway & Order Processor	88
Figure 6.7: EShop Interactions with Tax and Shipping Services.....	89

Figure 6.8: StoreFront Activity	90
Figure 6.9: FindProduct Activity	92
Figure 6.10: SearchProduct Activity	93
Figure 6.11: SelectProductFromCatalog Activity	95
Figure 6.12: SelectProductFromWishlist Activity.....	95
Figure 6.13: CheckoutItems Activity – Phases One and Two	98
Figure 6.14: CheckoutItems Activity – Phase Three.....	99
Figure 6.15: CreateOrder Activity.....	101
Figure 6.16: OrderProducts Activity.....	103
Figure 6.17: ProcessOrder Activity	104
Figure 6.18: CheckOrderStatus Activity.....	105
Figure 6.19: RefundOrder Activity.....	107
Figure 6.20: RegisterWithTheStore Activity	108
Figure 6.21: UpdatePersonalProfile Activity	109
Figure 6.22: CreateQuickCheckoutProfile Activity.....	110
Figure 6.23: ResetPassword Activity	111
Figure 6.24: CreateWishList Activity.....	113
Figure 6.25: SendWishList Activity	114
Figure 7.1: Adding Wish Lists Subfeature for Modeling Requirement	117
Figure 7.2: Adding Policy Subfeatures for Modeling Requirement	118
Figure 7.3: Using Multiple Associations to Model Multiplicity Variability	129
Figure 7.4: Using Multiplicity MetaExpression to Model Multiplicity Variability	130

Chapter 1 Introduction

In 1968, a growing concern about the gap between “what was hoped for from a complex software system, and what was typically achieved” [Nat68] began to surface among the attendees of the North Atlantic Treaty Organisation (NATO) Software Engineering Conference. The term *software crisis* was coined to describe this concern. The attendees discussed problem areas and underlying causes to the software crisis, but perhaps Booch summarized it best in 1991 when he stated, “our failure to master the complexity of software results in projects that are late, over budget, and deficient in their stated requirements.” [Boo91] The crisis resulted from the need to build *industrial-strength software* with rich functionality and a long production lifecycle.

Today, software is a pervasive entity which is critical to many areas. The applications range from scheduling a coffeemaker to handling ignition timing in the automobile engine control unit to managing supply chain systems for multinational corporations. The need for industrial-strength software has increased, along with the demand for highly customized, high quality software in different domains. These demands increase the software complexity factors defined by Booch¹.

Despite the advancement in software engineering technologies, studies show that the software crisis continues to persist. In 1995, The Chaos Report [Sta94] by the Standish Group predicted that almost half of all software projects would cost almost twice the amount they were budgeted for and almost a third would be canceled. In 2005, the Cutter Group released preliminary results from a study consisting of several hundred software projects which showed nearly identical results [Cut05]. Clearly, the software development processes and tools available now are not meeting the demand.

Generative Software Development (GSD) is an approach that allows developers to manage some of the software complexity factors. In this thesis, an example, inspired by a realistic application, of a Business-to-Consumer system is developed. The example is modeled using a GSD approach and the modeling experience is analyzed for areas in which the approach can be improved.

¹ Booch defines the four software complexity factors as: “1) the complexity of the problem domain, 2) the difficulty of managing of the development process, 3) the flexibility possible through software, and 4) the problems of characterizing the behaviour of a discrete system.” [Booch 91].

1.1 Generative Software Development

GSD is an approach that automates product development based on configuration knowledge and a set of reusable assets [Cza05a]. It provides a practical framework for reuse, enables a better understanding and expression of the application domain, and improves the quality of software produced. The key concepts in GSD are system families, development methodology, and reusable assets.

1.1.1 System Families

A system family is “a group of products that can be built from a common set of assets” [Wit96]. The term product line is sometimes used interchangeably, but product line is a more specific concept because the products also share “a common, managed set of features that satisfy the specific needs of a market” [Wit96].

The motivation for developing system families as opposed to individual products in the family is to encourage systematic reuse of assets within a specific domain. Rapid development and reduced development costs can be realized for products with multiple platform implementations or target markets. The focus on building reconfigurable assets allows for components to be assembled in combinations to satisfy customer requirements for a highly customized, unique product without expending the development effort of a custom development project for each individual product.

1.1.2 Development Methodology: Domain and Application Engineering

The development methodology for GSD requires two separate activities – Domain Engineering (DE), which is developing *for* reuse, and Application Engineering (AE) developing *with* reuse. DE is the process of understanding and scoping the system family through domain analysis, and developing the set of reusable, configurable assets. Domain analysis involves research to gain an understanding of the domain from the perspective of the domain expert; it typically consists of consultations with domain experts, examining documentation, and performing site surveys. An important aspect of domain analysis is to obtain a clear definition of the terminology and processes, which helps reduce the impedance mismatch between the requirements and the design [Boo91]. The construction of reusable assets involves developing assets, such as components or patterns, to be used for creating concrete products.

AE is the process of specifying and configuring individual products based on the configuration information provided by the application engineer and the reusable assets provided through DE. The process can be either manual or automatic depending on the framework that is provided; GSD provides a mechanism for automatic product generation. New requirements may be discovered during AE which can be fed back to DE to extend the scope of the product line and define new reusable assets, thus allowing new products to be generated. Therefore, DE and AE are complementary processes. The initial cost of DE can be large, but it may be amortized over the individual products generated and the improved quality of the software produced.

1.1.3 Reusable Assets

Reusable assets describe any artifact which is part of the reuse process. There are many forms and granularities for reusable assets, including object-oriented design patterns [Gam95], individual classes and mixins, components [Szy03], domain specific languages (DSLs) [Cza05a], domain specific business patterns such as Enterprise Architecture Patterns [Fow03a], or frameworks which implement common architectural styles [Sha96]. These assets can be classified as part of the problem space and solution space. The problem / solution space relationship is shown in figure 1.1.

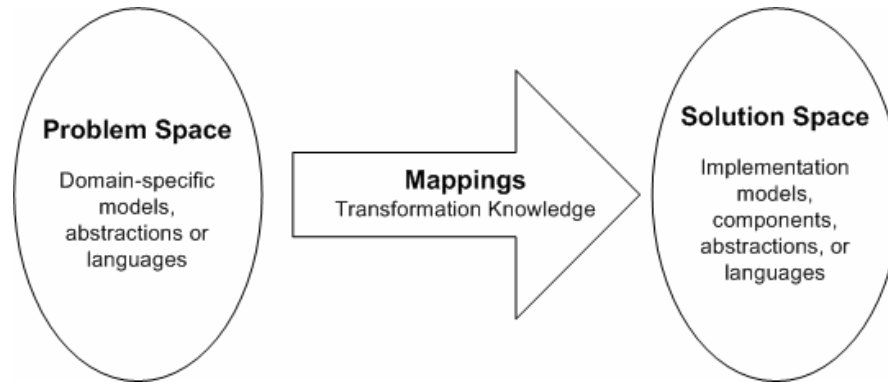


Figure 1.1: The Problem and Solution Space Pattern [Cza05a]²

The problem space describes the set of artifacts that represents the domain knowledge, such as requirements, glossaries, DSLs and models of the domain. The solution space describes the set of

² This is a modified version of the diagram which concatenates information from multiple diagrams in the source.

implementation artifacts used to generate a product, such as algorithms, concrete components and code templates. The two spaces are related through the transformation knowledge, which describes the mappings from the product configuration to the implementation. Mappings between the two are not necessarily one-to-one. For example, multiple DSLs can be mapped to the same implementation, where different DSLs are available for users with different proficiencies.

1.2 Model-Driven Software Product Lines

Model-Driven Software Product Lines (MDSPL) is a particular implementation of the GSD approach [Cza05b]. It combines two emerging practices in software engineering: Software Product Line Engineering (SPLE) and Model-Driven Software Development (MDSD). SPLE is a synonymous with the system family approach, which focuses on building system families. One way of realizing MDSPL is to use feature models and feature-based model templates.

1.2.1 Model-Driven Software Development

MDSD is a development process which uses models, as opposed to source code, as the primary implementation artifact. Models tend to be better at capturing the intent of requirements because they are closer to the requirements than source code. Modeling languages, such as the Unified Modeling Language (UML), provide semantics which allow the user to be more precise in stating requirements, rather than worrying about implementation language details which can lead to implementation bias. It is important to note that having models as primary development artifacts means that the models should be treated the same way as source code in traditional programming.

1.2.2 Feature Models

Feature modeling was proposed by [Kan90] as a mechanism to manage variability in a system family. It was done as a part of the Feature-Oriented Domain Analysis (FODA). FODA defined some of the basic constructs of feature models, such as mandatory and optional features, and the relationships between sets of features, such as exclusive-or groups of features. The original FODA notation has been extended through cardinality-based feature modeling [Cza05c], feature attributes, references to feature models and reference attributes, multi-staged and multi-level configurations [Cza05d], external constraints, and grouped feature cardinality [Cza05e]. These extensions allow a greater degree of expression and unlock a greater potential for its application

in GSD. The process of removing variability from a feature model through the selection or elimination of features is referred to as configuration. Feature modeling is described in greater detail in section 2.2.

1.2.3 Feature-Based Model Templates

Feature-based model templates, proposed by [Cza05f], allow semantics to be associated to features through the annotation of presence conditions (PCs) or metaexpressions (MEs) on model elements. The PCs and MEs are based on the features; their effect is dependent on whether or not the feature is selected during configuration. The novel aspect of this approach is that all variability is superimposed in the model template, so that template instantiations only involve the removal of elements. A model template can be instantiated based on a configuration to generate a model which represents a concrete product. Feature model templates are described in greater detail in section 2.3.

1.3 E-Commerce Systems

E-Commerce Systems describe software systems which enable companies to do business over the Internet. They must be able to deal with a large number of visitors and transactions, and coordinate multiple stakeholders to deliver the product or service to the customers. Three common transaction patterns are Business-to-Consumer (B2C), Business-to-Business (B2B), and Consumer-to-Consumer (C2C) [Raj00]. B2C sites, like amazon.com, enable retail transactions where a company sells goods or services to an individual. This is done through an e-shop web site, which is sometimes referred to as a shopping cart solution. B2B sites are meant for the exchange of products, services or information between multiple companies. They include company web pages, product supply and procurement exchanges, information sites, and brokerage services. C2C site allows individuals to sell to other individuals. Sites can be run by an intermediary business, such as E-Bay or the Amazon Marketplace, or by consumers themselves.

1.4 Research Overview

The primary objectives of this research are:

- to demonstrate that the MDSPL approach is viable for software systems inspired by realistic applications, as opposed to just toy examples;
- to identify areas which impede usability and may discourage developers from adopting the tool;
- to make recommendations on the identified problem areas.

Due to the immense size of the e-commerce domain, the scope of the research was limited to B2C solutions with fixed purchase prices.

Over the course of the research period, the following tasks were performed:

- domain analysis of e-commerce systems,
- application of the feature-based model template development approach to constructing an online B2C system,
- analysis of the development approach.

The tasks are explained in detail within the context of the research methodology in chapter 3.

1.5 Research Contribution

The following contributions were made throughout the course of the research:

- application of the feature-based model template development process to develop and model an e-commerce system of moderate complexity,
- evaluation of the development process based on the experience with the e-commerce model; the evaluation was used to derive guidelines for feature modeling and model template creation, as well as recommendations for process and tool improvement.

Both are unique contributions; the first contribution is the only example known to the author of a moderately complex system being modeled using the feature-based model template approach. Previous models have been smaller in scope and used for examples in papers [Cza05f]. The

second contribution is the first evaluation performed on the process after the initial tool release. It is also unique because the analysis is based on a larger example than in previous papers and performed by a member of the research team who was not directly involved with the original design of the approach and implementation of the tool. The size of the example helps support the findings in this work.

1.6 Thesis Organization

The remainder of the thesis is organized as follows: Chapter 2 describes background material for concepts used in the research. Chapter 3 presents the research methodology in detail, based on the tasks described in section 1.4. Chapters 4 and 5 document the domain analysis through the feature model, based on the store front and business management features respectively. Chapter 6 describes the design of the feature-based model templates. Chapter 7 evaluates the feature-based model templates approach based on experiences with the e-commerce model, and presents a set of recommendations to improve the process and implementation tool. Chapter 8 concludes the thesis and discusses areas for future work. Two appendices are included: Appendix A contains a table which summarizes the additional constraints in the e-shop feature model, and Appendix B contains the raw data that was obtained during the annotation analysis.

1.7 Summary

In summary, GSD is an approach to software development that is touted to address some of the issues raised in Booch's factors of complexity. The purpose of this research is to perform an analysis on the e-commerce domain to develop an example that can be used in the MDSPL approach. This research will result in an example that can be used to demonstrate the viability of this approach and a set of recommendations that can be used to improve the process.

In this chapter, the research problem and the motivation for the work were introduced. This chapter included a brief overview of GSD and related concepts, a high-level overview of the e-commerce domain, a summary of the research methodology and contributions, and a description of the thesis organization.

Chapter 2 Background

This chapter describes key concepts which are relevant to the remainder of the thesis. It covers aspects of the Unified Modeling Language (UML) and the diagrams presented in the thesis, feature models and feature-based model templates. In each section, the theory is presented first, followed by any tool-specific details or notations which are used in the models.

2.1 Unified Modeling Language

The UML is “the standard language for specifying, visualizing, constructing, and documenting all artifacts of a software system.” [Qua03] The standard is owned and maintained by the Object Management Group (OMG). It is meant to be used as a notation for modeling object-oriented systems. The standard defines a series of models that can be used to capture the structure, behaviour or organization of a piece of software. All UML models presented in the thesis conform to UML 2.0, the most recent version of the standard.

The UML metamodel is defined by the Meta Object Framework (MOF) model, another OMG standard that is used to define modeling languages. MOF is meant to be platform-independent and can be serialized into the XML Metadata Interchange (XMI) format for portability between applications which support MOF models [Omg05a].

The models defined by UML are very general; domain specificity can be added by extending and / or restricting the language through profiles. OMG defines profiles as a package which “contains mechanisms that allow metaclasses from existing metamodels to be extended to adapt them for different purposes.” [Omg05b] These mechanisms include stereotypes, tagged values and constraints. Stereotypes are tags which can be used to add or restrict semantics of a metaclass. A stereotype can be specified by constraints, attributes or icons.

The two types of UML models which are used in the research are class diagrams and activity diagrams; both are discussed in the following subsections.

2.1.1 Class Diagrams

Class diagrams are used to depict the structure of the software by describing “the types of objects in the system and the various kinds of static relationships among them”, as well as “the properties and operations of a class and the constraints that apply to the way objects are connected” [Fow03b]. An example of a class diagram rendered with IBM Rational Software Modeler (RSM) is shown in figure 2.1; different elements in the diagram are labeled. The formal definition for each labeled element is given in the UML2 Super Structure document [Omg05b].

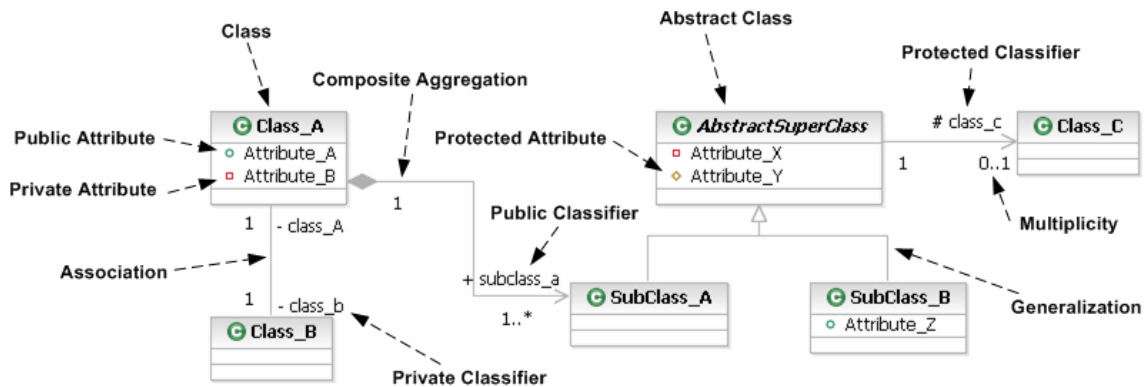


Figure 2.1: Labeled Class Diagram Rendered in RSM

Figure 2.2 is an example of a class diagram which uses stereotypes to denote components and services.

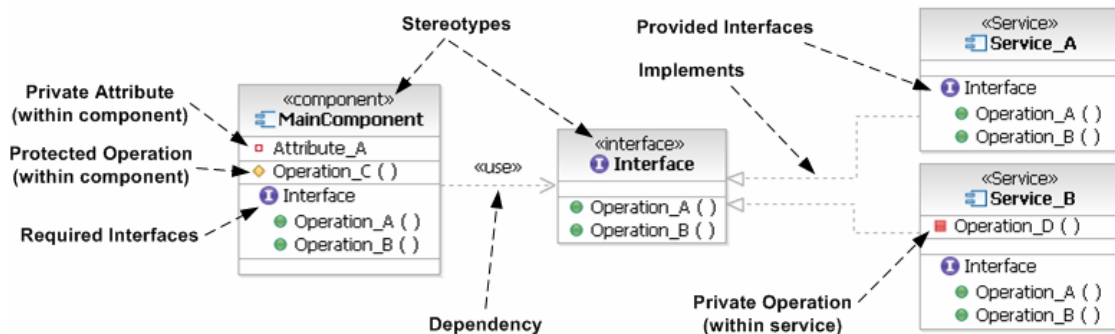


Figure 2.2: Class Diagram with Stereotypes Rendered in RSM

2.1.2 Activity Diagrams

Activity diagrams are used to model behaviour in the software through series of actions. They are similar to flowcharts with the main difference being that activity diagram support parallelism in

the workflow [Fow03b]. An example of an activity diagram rendered with RSM is shown in figure 2.3; different model elements in the diagram are labeled. In addition, the Action and CallBehaviourAction nodes are named after their respective element types. The formal definition for each labeled element, except for those in quotations, and some of the named nodes are described in the UML2 Super Structure document [Omg05b].

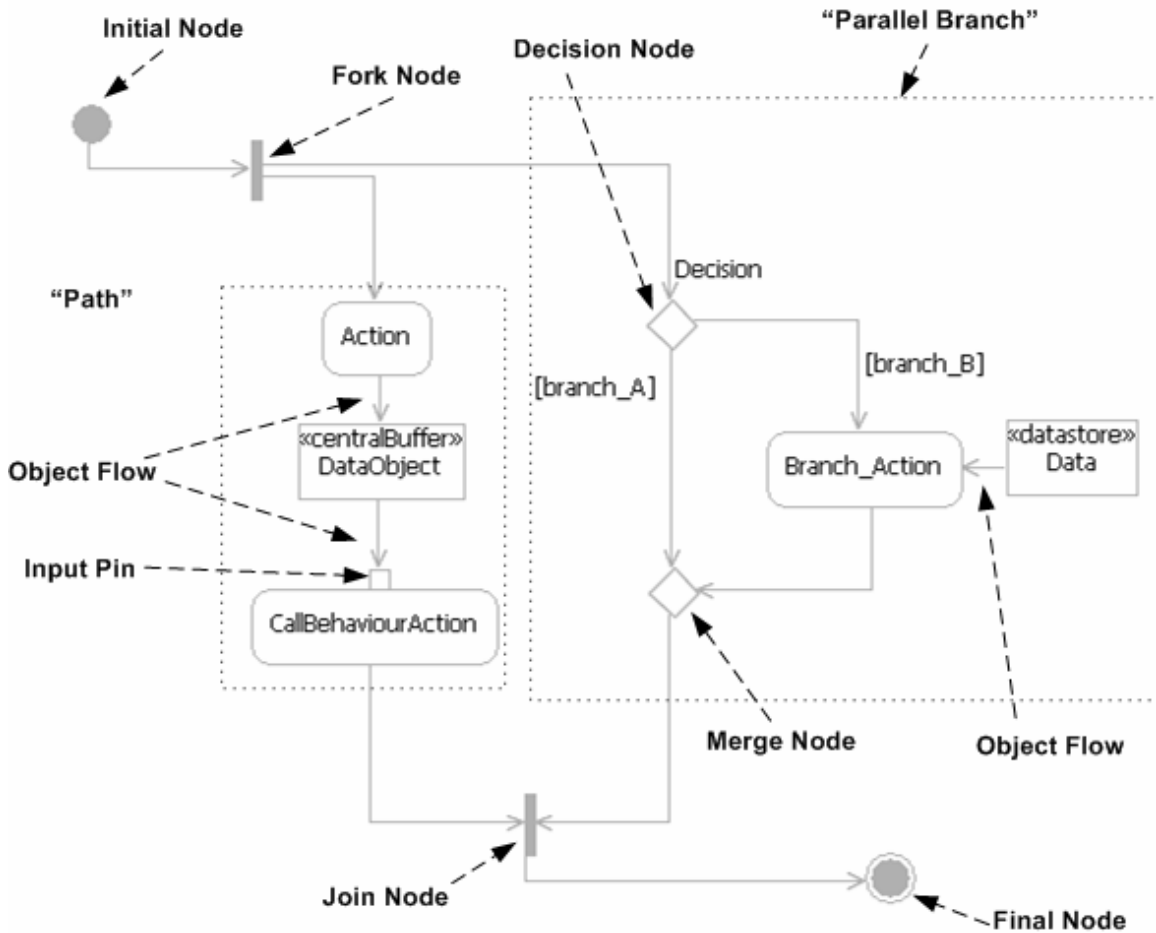


Figure 2.3: Labeled Activity Diagram Rendered in RSM

It is useful to define three terms to describe regions in an activity diagram. The first term is *path*, which describes a consecutive sequence of nodes and flows. The second term is *branch*, which describes a set of model elements between a pair of decision and merge nodes. A branch is identified by its guard. The third term is *parallel branch*, which consists of a set of model elements along a single path, starting at a fork node and ending at a corresponding join node. A parallel branch is identified by its ordinal position.

In the example, two parallel branches execute in the activity. The first parallel branch executes Action, which passes an object of type DataObject to CallBehaviourAction. The CallBehaviourAction receives the object through its input pin and invokes the target activity. Once the target activity has executed, control returns to this activity and waits at the join node for the other parallel branch to complete. The second parallel branch begins with a decision. If the outcome is branch_B, it will execute the Branch_Action action, which requires an object of type Data from the data store, and proceed to the merge node. If the outcome is branch_A, no actions are executed and it proceeds directly to the merge node; this single flow is often referred to as a *bypass branch*. Once both parallel branches have completed execution, the activity reaches the final node and the execution of the activity is complete. Unless otherwise indicated, all flows in the activity diagram are control flows.

2.2 Feature Models

Feature models represent the variability perspective of a product line through the definition of features and their relationships. They describe the set of all valid products in a product line. The instantiation of a concrete product from a feature model requires the removal of variability, which can be achieved through the selection or elimination of features; this process is called configuration and is discussed in section 2.2.1.

A feature is a characteristic of a system which is of interest to a stakeholder and each feature represents a common or variable aspect of a product [Cza00]. Features can take the form of functional or non-functional requirements. In the example studied later, it is often found that software requirements can be described directly as features. Features may model parts of a system which correspond to entities, entity attributes, processes, or non-functional properties.

Features can be related through relationship types, which include relation through hierarchy, feature cardinality, group cardinality and additional constraints. Hierarchy structures the feature model through subfeatures. There are two main purposes for using subfeatures. The first is for modularization purposes; subfeatures allow a feature to be decomposed into greater detail. The second is for implication purposes; the selection of a feature implies the selection of all of its ancestors. There are two common types of features: a *solitary feature*, which is a feature that is made a subfeature directly under another feature, and a *grouped feature*, which is a feature that is placed directly under a feature group.

Feature cardinality is “an interval denoting how often a feature with subfeatures can be cloned as a child of its parent when specifying a concrete system.” [Cza05f] Feature cardinalities are placed on solitary features and denoted using square brackets; for example, a feature cardinality of $[1..k]$ indicates that at least one and at most k clones must be present in a concrete product. Two common feature cardinalities are $[1..1]$ and $[0..1]$, which denotes a *mandatory feature* and an *optional feature* respectively. A mandatory feature must be present exactly once in a concrete system, and an optional feature may or may not be present in a concrete system.

Group cardinality is an interval placed on a feature group that denotes how many grouped features can be selected from the feature group in a concrete system. Group cardinalities are denoted using angle brackets; for example, a group cardinality of $\langle 1..k \rangle$ indicates that at least one and at most k features can be selected from the feature group, and that k is the total number of grouped features in the feature group.

Additional constraints specify additional relationships in the feature model. There are two types of additional constraints: a *strong* constraint and a *weak* constraint. A strong constraint takes the form of *requires* or *excludes*, whereas a weak constraint takes the form of *recommends* or *discourages*. Additional constraints are described in greater detail in appendix A.

Relating the use of feature models back to GSD, the feature models themselves are reusable assets and their development, through domain and variability analysis, is part of DE. Reuse occurs when a concrete product is created during AE.

2.2.1 Configuration

Configuration is the process of eliminating variability in a feature model through the selection or elimination of features; the output of the process is either a specialization, which is a feature model with some variability remaining, or a feature configuration, which is a feature model that represents a concrete product because all of the variability has been eliminated.

A specialization is created when it is undesirable to remove all of the variability in the product line in a single iteration; therefore, instantiating a concrete product occurs over multiple iterations. This process is called multi-staged configuration. A specialization in any given iteration will contain the same or less variability than in the previous iteration. Any decisions are

propagated to all subsequent iterations. Furthermore, cardinality can also be configured through refinement. Refinement is the process of narrowing the bounds of the cardinality such that the set of possible values in the refined cardinality is a subset of the set of possible values in the original cardinality.

Another form of configuration is multi-level configuration [Cza05d]. In this type of configuration, there are multiple levels, each represented by a different feature model. During configuration, a feature model at one level is manually configured. This results in the automatic specialization of all feature models in subsequent levels through some specified configuration knowledge. The intention is to have feature models at different levels represent different levels of abstraction; thus, multi-level configuration allows higher-level choices from the requirements to configure lower level details in an implementation.

2.2.2 Binding Time

Binding time refers to the time at which the decision regarding a feature's selection or elimination is made. The definition for binding times is strongly influenced by workflows, such as the development and deployment process. In this thesis, it is convenient to define two binding times: *build-time* and *run-time*. Build-time refers to the period of time up to the configuration of the product during AE. Decisions made during build-time often result in the inclusion or exclusion of a feature in the concrete product. These decisions are usually based on product requirements or specifications. Run-time refers to the period of time after the concrete product has been configured. This encompasses the deployment, daily operation, and use of the product. Decisions made during run-time often deal with the configuration of the product options.

In practice, there are cases where it can be difficult to qualify when build-time ends and when run-time begins. For example, during multi-staged or multi-level configuration, one user's run-time can be another user's build-time. In another example, the distinction between binding times can be blurred by the use of dynamic plug-ins. Depending on the process, there can be different qualifications of what can be defined as build-time and run-time.

Furthermore, there are many granularities of binding times to consider. For example, a product may be able to support multiple features which are mutually exclusive. It will continue to do so until it reaches a point where only one feature can be selected. That point in time may be at











deployment, at the definition of a rule, or during a user session. In such a case, it is clear that there are several sub-stages within run-time, such as the decisions made per installation, per policy or per user.

For simplicity, discussions about binding times will be based on the definitions of build-time and run-time presented in this section. There may be some cases which involve a more specific or finer-grained binding time. In those cases, the discussion will describe the binding times in relation to build-time or run-time.

2.2.3 Tool Support

Feature modeling is supported through *fmp*, an Eclipse plug-in which was developed by the research group [Cza05g]. *fmp* is supplied with a default metamodel which defines the structure of the feature model, along with the feature types described in the parent section. It includes definitions for feature properties, which include a name, id and description for every feature. It also supports metamodel editing to allow the user to define additional feature attributes. Table 2.1 displays the icons used to denote the commonly used feature modeling concepts.

Table 2.1: Feature Modeling Notation in *fmp* [Kim05]

Icon	Explanation
 F	Solitary feature <i>F</i> with feature cardinality [1..1] (i.e. mandatory feature)
 G	Solitary feature <i>G</i> with feature cardinality [0..1] (i.e. optional feature)
 [0..m] H	Solitary feature <i>H</i> with feature cardinality [0..m], $m > 1$ (i.e. optional cloneable feature)
 [m..n] J	Solitary feature <i>J</i> with feature cardinality [m..n], $m > 0 \wedge n > 1$ (i.e. mandatory cloneable feature)
 K	Grouped feature <i>K</i> with feature group cardinality [0..1]
 L	Grouped feature <i>L</i> with feature group cardinality [1..1]
 M ('value' : T)	Feature <i>M</i> with attribute of type <i>T</i> and value of <i>value</i>
	Feature group with group cardinality <1-1> (i.e. exclusive-or group)
	Feature group with group cardinality <1-k>, $k = \text{group size}$ (i.e. inclusive-or group)
 <i-j>	Feature group with group cardinality <i-j>

The tool also provides support for configuring a feature model through a checkbox interface. A checkbox can have three states: *empty*, denoting that the feature state is undecided and that no decision has been made yet; *checked*, denoting that the feature has been selected and will be included in the concrete product; and *crossed*, denoting that the feature has been eliminated and

will be excluded in the concrete product. In addition, the tool also provides support for automatic constraint propagation, which enables the system to automatically select or eliminate features based on the relationships between the features.

A research prototype version of the tool, *fmp* 0.6.1, was used to create the models presented in this thesis.

2.3 Feature-Based Model Templates

Although a feature is given context within the feature model, there is no satisfactory way of defining a feature's semantics within the model so that it can be processed and transformed into an implementation. Feature-based model templates, which are later referred to as *model templates* for simplicity, allow the modeler to define structure and behavioural models in a MOF-compliant language, and to provide semantics for features by associating them to model elements through annotations. In addition, models for concrete products can be instantiated based on a configuration.

The novel aspects of feature-based model templates are that they separate the representation of the variability into the feature model and that all of variability is represented through superimposed variants. Like the feature model, the model template represents well-formed models for all valid products in a product line; however, the model template itself does not have to be well-formed with respect to the target modeling language due to the need to superimpose variability. The benefit of the superimposed variability is that models can be automatically instantiated through the removal of model elements and the application of patching transformations to ensure well-formedness.

There are two types of annotations which can be used to associate features to model elements: metaexpressions (MEs) and presence conditions (PCs). MEs allow the modeler to calculate a value based on the presence of features in a configuration through the specification of an expression. PCs are Boolean expressions which determine the presence of the annotated model element based on the selection or elimination of the features in the configuration. In addition to PCs, there are also implicit presence conditions (IPCs). IPCs are defined for model elements to determine the presence of a model element in the absence of an explicit annotation. IPCs are

based on the modeling language syntax and are used to reduce the annotation effort required and ensure the well-formedness of the template instance.

During template instantiation, model templates undergo additional processing steps, which include patching transformations and simplification. Patching is a transformation which corrects certain errors that may result from the removal of elements [Cza05f]. An example is flow closure in an activity diagram, which is required to connect a broken flow as a result of removing an action. Simplification is a transformation which removes any redundant model elements that result after the removal of elements. An example is the removal of a merge node which has only one incoming and one outgoing flow.

There are four steps in the template instantiation process:

- 1) the MEs and PCs are evaluated. The evaluation of PCs is performed depth-first on the containment hierarchy of the elements in the models to prevent unnecessary processing of nested elements, e.g. if the containing model element is removed;
- 2) removal analysis is performed to calculate the values of the IPCs. In addition, removal analysis is also used to determine where patching transformations will be necessary;
- 3) the elements with false PCs or IPCs are removed. Any patch transformations are also applied at this time;
- 4) simplification transformations are applied.

Due to the way PCs are evaluated, the presence of all model elements that are contained depends on the annotation on the container elements. In this situation, it is convenient to define the term *implicit annotation*, which denotes the transitive effect of a parent element's annotation on a child element. For example, when an activity is annotated with feature F , one can state that all of the nodes and flows contained by the activity are implicitly annotated with feature F .

Relating the use of model templates back to GSD, the model templates themselves are reusable assets and their development, through the template design and annotation, is part of DE. Reuse occurs when the model is generated through template instantiation, which is a part of AE.

2.3.1 Tool Support

Model templates are supported through *fmp2rsm*, a plug-in for IBM Rational Software Modeler (RSM) or Rational Software Architect (RSA). The plug-in was developed by the research group [Cza05g]. *fmp2rsm* depends on *fmp* for the feature modeling and it depends on RSM for the UML diagrams. PC annotations are created through a context menu in the feature model editor, where a PC in the form of a conjunction of selected features is created. The tool represents the annotations as stereotypes, which are stored in a variability profile. The model files are associated with the variability profile and a model element is annotated by applying stereotypes from the profile.

A research prototype version of the tool, *fmp2rsm* 0.0.4, was used to create the models presented in the thesis. It works for all UML models; however, IPCs and additional processing steps are only implemented for activity diagrams.

2.4 Summary

In this chapter, an overview of the background concepts was presented. The key concepts applied in this thesis are UML class and activity diagrams, feature models and feature-based model templates.

Chapter 3 Research Methodology

This chapter describes, in detail, the steps taken throughout the research process. As mentioned in section 1.4, the research consisted of three main steps:

- domain analysis of e-commerce systems,
- application of the feature-based model template development approach to constructing an online B2C system,
- evaluation of the development approach.

Each step is expanded upon in the following subsections. Please note that, unless otherwise noted, all of the work described in this chapter was performed by the author.

3.1 Domain Analysis of E-Commerce Systems

The motivation for choosing e-commerce systems as the domain was the need to find a relatively large, well-understood domain with a large degree of variability on which a serious evaluation of the approach could be performed. In addition, e-commerce systems form an important part of many business strategies as companies expand their presence to the Internet; this guaranteed applicability of the research, which was another reason why the domain was an attractive choice.

Initial research indicated there were different types of e-commerce sites, as described in section 1.3. It also suggested that the domain was too large to model in its entirety given the research time frame; therefore, the research project was scoped to focus on a B2C e-shop solution with fixed-price purchasing only. The initial research suggested that there was a significant amount of material available, including the ability to do field research on real e-shops. Several sources were used when performing the domain analysis for the e-shop model; some of the information gathered was integrated into the model designs. These sources included:

- *Literature Research.* Much has been written and published on the subject of e-commerce systems. One set of books deals with the implementation of e-commerce solutions; however, they tend to be skewed to a particular platform. Two platform-specific books were consulted for domain information and examples: 1) [Alu01, Sun02] describes an online pet store example implemented using the Java 2 Enterprise Edition (J2EE) Software Development Kit

(SDK), and 2) [Ped03] describes how to build an e-shop solution using Microsoft Commerce Server 2002 and the .net framework. The other set of books deals with the domain itself, although they vary in their focus and the amount of detail. *Software Factories* [Gre04] contains a chapter which discusses modeling the e-commerce domain and presents a high-level feature model of the domain. [Raj00] provides an overview of e-commerce systems, both from the technical perspective and the marketing perspective; it was useful for clarifying some domain concepts. [Fow03a] focuses on enterprise systems³ of which an e-commerce system is an example. The concepts are presented through a set of patterns which provided some inspiration for the model templates. For example, the modeling for the price attributes in the Product class was taken from Fowler's Money class.

- *Web Research.* Various sites were browsed for information, but one web site in particular [Hos05] provided a concise summary of the features in shopping cart solutions, as well as a directory of online B2C solutions.
- *IBM WebSphere Commerce Documentation* [Ibm05]. IBM provides detailed information about the workflows that can be configured and incorporated into an e-commerce product which is built using IBM's WebSphere Commerce tools. The ConsumerDirect documentation describes a simple e-shop with many process definitions, such as ordering products or maintaining registration information. The activity diagram model templates were based on a subset of the models presented in the documentation.
- *Existing Implementation Study.* A pre-existing shopping cart solution, AbleCommerce [Abl06], was deployed and studied for the different types of configuration options which were available. AbleCommerce stores all of the store data and configuration in an eXtensible Modeling Language (XML) file; this file was reverse-engineered to derive the XML schema. The schema highlighted different features which were not readily apparent through the web site itself.
- *Field Research.* A few large e-commerce sites were studied based on their web sites. American e-shops, such as amazon.com, barnesandnoble.com and circuitcity.com, were studied based on the interface without making any purchases; therefore, they contributed primarily to the analysis of store front features. Canadian e-shops, such as bestbuy.ca, chapters.indigo.ca, futureshop.ca, and staples.ca, were studied based on both the browsing interface and through the shopping experience when products were ordered; therefore, they

³ Fowler does not provide a definition for enterprise systems. Instead, he describes four characteristics of enterprise systems: 1) they deal with large quantities of complex, persistent data, 2) the data must be concurrently accessible by a large number of users, 3) there are a large number of user interface screens, and 4) enterprise systems require integration with other enterprise applications [Fowler 99].

contributed to the analysis of the store front and business management⁴ features from the customer's point of view. For example, registration profiles, payment options, and checkout processes were studied in these e-shops and incorporated into the descriptions and the models. The incorporation of field research helped justify the inclusion or exclusion of certain features in the model based on their frequency of occurrence.

3.2 Applying the MDSPL Approach

The research performed during the domain analysis helped build an understanding of the domain, which was translated into a series of models through the MDSPL approach. The first model was the feature model. The first version of the feature model was created by Dr. Simon Helsen. It consisted of approximately thirty features, which modeled the catalog structure, payment options, and some business management features. This model was extended by Dr. Krzysztof Czarnecki, who added approximately two hundred features detailing many aspects of the e-shop, such as targeting mechanisms and buy path elements. The thesis presents the extended version of the feature model. The extended model contains some additional refinements to certain features, such as the subfeatures of the wish list, and some reorganization in the model, such as promoting the user behaviour tracking feature to the store front level. The extended version of the model has never been documented; therefore, a major contribution of this thesis is the definition of each feature in terms of domain concepts and specific examples. Some of the examples suggest further decomposition of features; however, these additions were not incorporated into the thesis version of the feature model due to time constraints.

The model templates were the second set of models developed. The model templates focused on the structure and behaviour of the store front. The primary goal was to capture, at minimum, the entities and workflows needed to allow a customer to visit the e-shop and make a purchase. There are two types of model templates: class diagrams and activity diagrams.

The class diagrams consist of the store front entity model and the service model, both of which were heavily influenced by the feature model; the classes, services and operations were all created directly from features. The store front entity model incorporates some earlier work that

⁴ Business management features deal with back-office operations in an e-shop; however, these features are presented from the customer's perspective because no specific information was available from the business perspective.

was presented in an earlier paper [Cza05f]. The earlier work consisted of model fragments with annotations, such as the Product and Catalog classes, which were used to demonstrate key points in the paper. The version of the store front entity model presented in this thesis adds much more detail to reflect the store front features in the feature model, such as classes for wish lists, shopping carts, orders, and taxation rules. The service model is an original creation that was designed to depict the e-shop from a higher-level of abstraction, as well as to illustrate the actors involved.

The activity diagrams consist of activities which support site browsing, registration, customer administration tasks, wishlist management, item purchase, and order processing. Most of the activities were heavily influenced by the ConsumerDirect workflows [Ibm05]; some of the activities are a direct translation from the IBM workflow notation to the UML activity diagram notation. Some modifications, such as adding paths or actions, were required to model the variability in the model template.

After the models were created, all of the model templates were annotated with PCs and MEs. Some of the model templates went through a few revisions due to experimentation with different modeling styles and annotations. Although new features were discovered through modeling the activities, the feature model was not updated. For example, none of the elements in the ResetPassword activity, as described in section 6.2.15, are annotated; however, one could derive several optional features based on the workflow, such as the use of challenge questions and temporary passwords. It is important to note that most elements in an activity diagram model template are not annotated because they represent common aspects in a product; the scenario described for the ResetPassword activity is a special case because it represents a work in progress.

3.3 Evaluation of the Approach

The evaluation of the approach consisted of analyzing the experience during the development process. The experience led to some discussion of various aspects in feature modeling, such as binding time analysis and model template design. All of the annotations in the models were studied to discover any frequently occurring annotation patterns. The iterations in experimenting with different modeling styles and annotations were documented in the form of modeling

guidelines. In general, the documentation of observations during the process took the form of modeling guidelines or recommendations.

3.4 Summary

In this chapter, the research methodology was presented. The research was divided into three major activities: the domain analysis of the e-commerce system, the application of the MDSPL approach using feature models and feature-based model templates, and the evaluation of the approach. The domain analysis consisted of consulting various literature resources and field research. The application of the approach consisted of developing the feature-based model templates and documenting the feature models. The evaluation of the approach consisted of an analysis of the modeling experience and the models produced; candidates for modeling guidelines and recommendations were proposed.

Chapter 4 E-Commerce Domain Analysis: Store Front⁵

The store front is the interface that the customer uses to access the e-shop. The features described in this chapter are related to the interface; many are directly visible to the customer and impact their experience at the e-shop. Figure 4.1 illustrates the store front feature and its subfeatures.

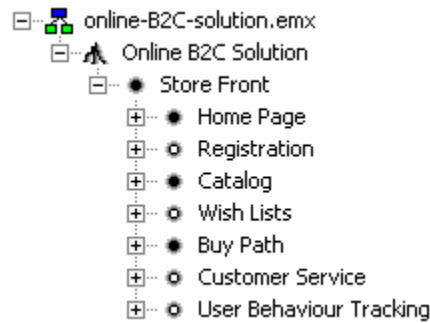


Figure 4.1: Store Front Feature

The required store front feature consists of a set of functional features representing the home page, catalog, and buy path. The store front may also include registration, wish lists, customer service and / or user behaviour tracking capabilities.

4.1 Home Page

Every e-shop has a home page which serves as a welcome page. It is the first page that a customer will see when they enter the site. A customer will be directed to the home page when they enter the top-level Uniform Resource Locator (URL), such as “www.amazon.com”. The e-shop may also be configured to redirect the customer to the home page if the URL points to an expired session, an invalid product page, or a restricted page. Figure 4.2 shows the home page feature and its subfeatures. The primary content of the home page consists of a welcome message and featured products. The content can be generated statically or dynamically. A page is classified as dynamically generated if any page element, such as a frame, is dynamically generated; otherwise, the page is classified as statically generated. At build time, the choice is inclusive-or since the e-shop product can have the capability of doing both; however, during the creation of individual pages at run-time, the choice is exclusive-or since the page is either statically or dynamically generated.

⁵ Additional constraints in the model are described in table A.1 in appendix A.

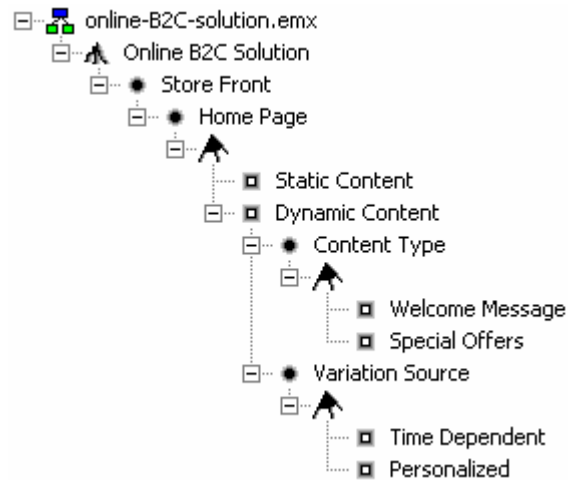


Figure 4.2: Home Page Feature

An alternate way of interpreting the static and dynamic content features is to scope them for individual elements on the page as opposed to entire pages. Using this interpretation, a page can contain both static and dynamic content, meaning that the feature group is inclusive-or for all binding times.

4.1.1 Static Content

Static content is ideal for content which changes infrequently. In a typical implementation with static content, customers see identical information on the home page. The page is created once, stored on the server, and served to every client that makes a request for the page. Any changes to the page will require a new page to be generated and uploaded to the server.

4.1.2 Dynamic Content

Dynamic content is ideal for content which changes frequently. In a typical implementation with dynamic content, the content is generated on demand and every customer receives a customized home page for each session. There are two required parameters for customization: the content type and the variation source.

- *Content Type*. Content type describes which elements can be dynamically generated. Two common content types are the welcome message and special offers. A welcome message is a greeting which is usually rendered as text. Special offers are promotions for customer, which

may include sales on products or discounts on orders. Support for special offers requires the selection of the discounts feature; however, the selection of the discounts feature has no impact on the special offers feature.

- *Variation Source*. Variation sources provide information that is used to generate the content. Two common variation sources are time dependence and personalization. Time dependence generates content based on the time at which the customer enters the e-shop. Time can refer to the time of day, such as morning or evening, or the time of year, such as summer or Christmas time. Personalization generates content based on customer information or inferred information. Customer information is data that is stored in the registration profile. Inferred information is data that is not provided explicitly, such as the country which a guest is visiting from⁶. Personalization is strongly related to targeting because targeting mechanisms are often used to generate customized advertisements and content for customers.

Based on the two parameters, different types of dynamic content can be generated. An example of simple dynamic content is a personalized greeting based on the time of the day. An example of complex dynamic content is a set of personalized special offers based on the customer's personal information. This can be extended by generating special promotions, such as a discount, for a customer each time they visit. Amazon.com's "Gold Box" is an example of this feature.

4.2 Registration

An e-shop may enable registration, which allows a customer's information to be solicited, persisted and reused. This is a convenient feature for customers because they do not have to re-enter their information every time they make a purchase. In addition, this information may also be useful for creating targeting strategies. Figure 4.3 shows the registration feature and its subfeatures. Registration requires decisions to be made about enforcement and the information that is collected. In addition, user-behaviour tracking information can be optionally associated with a profile.

⁶ This information can be deduced by examining the HyperText Transfer Protocol (HTTP) request and performing an Internet Protocol (IP) lookup.

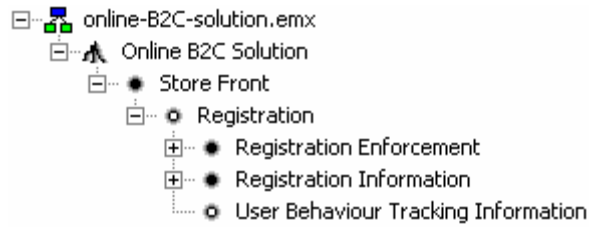


Figure 4.3: Registration Feature

4.2.1 Registration Enforcement

If registration is enabled, there must be a policy to determine which actions in the e-shop are restricted to registered users only. Figure 4.4 shows three policies: register to browse, register to buy and none. An e-shop product can be configured to support any combination of the three at build-time, but only one policy can be in effect at any given time after the e-shop is deployed at run-time.

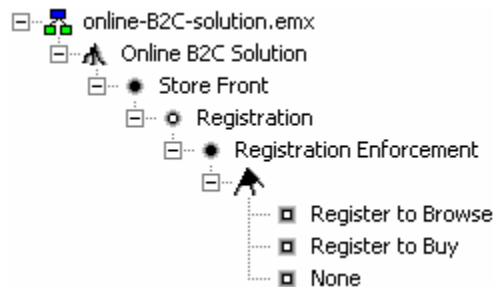


Figure 4.4: Registration Enforcement Feature

- *Register to Browse.* This policy restricts browsing to registered customers; it is the most restrictive policy. There are many ways to define browsing permissions. A fine-grained policy would define permissions on certain products or specific details about products, whereas a coarse-grained policy would define permissions on a page type, such as product or search pages. One implementation of this policy restricts access to all product pages to registered customers, but allows guests to see lists of products.
- *Register to Buy.* This policy requires customers to register before they can make a purchase. This can be implemented by requiring customers to log in before they can add an item to their shopping cart or start the checkout process.
- *None.* This is an unrestricted policy. Any visitor can freely browse and purchase items in the e-shop without going through the registration process.

4.2.2 Registration Information

Registration requires that the customer provide information about themselves. This information is stored in a customer profile. Figure 4.5 shows the fields which can be contained by a customer profile; most of the fields are optional.

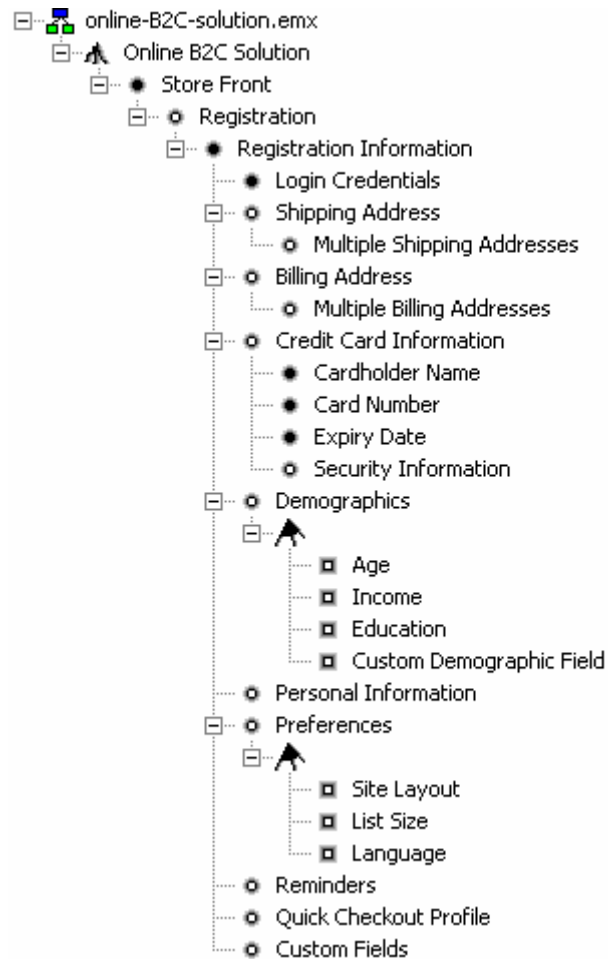


Figure 4.5: Registration Information Feature

The only mandatory field in a profile is the login credentials, which allow a customer to identify themselves when they log in. The credentials include a unique identifier, such as an e-mail address, and a password. The remaining profile fields, discussed below, are optional.

- *Shipping Address*. A shipping address specifies where to send the order. Storing multiple shipping addresses may also be supported. This may require the selection of a default

shipping address, either by the customer or the system. There are two possible ways of modeling the multiple shipping addresses feature: 1) making it an optional subfeature, or 2) making the shipping address feature a cloneable feature. The former was chosen because it provides a better level of abstraction. Furthermore, the complexity between building a system that supports a single address and multiple addresses is more significant than the complexity between building a system that supports n and $n+1$ addresses, where $n > 1$. If the shipping address is stored in the profile, it is recommended that the shipping feature⁷ be supported; however, support for shipping has no impact on the storage of the shipping address in the profile.

- *Billing Address.* A billing address specifies where to send the invoice. Storing multiple billing addresses may also be supported; the same points presented about multiple shipping addresses apply here as well. An application of the billing address is to validate credit card information; many e-shops require that the billing address matches the address registered with the credit card company.
- *Credit Card Information.* Credit card information consists of the information that is needed to validate the card and process the payment. This information includes the name of the cardholder, card number, expiry date and, optionally, any other additional security information on the card. If the profile supports storing credit card information, the acceptance of credit cards as a form of payment is recommended; however, the acceptance of credit cards has no impact on whether or not the profile can store credit card information.
- *Demographics.* Demographics include information, such as age, income and education, about the customer [Wik06a]. A custom demographic field allows the e-shop to specify new demographic data fields at run-time. Demographics are used primarily for business intelligence activities. One application is to use previous purchase data for a customer to recommend products to other customers who have similar information in their demographic profile. Another application is to group customers with similar information into a consumer group in order to study and predict trends in consumer behaviour for future marketing efforts.
- *Personal Information.* Personal information includes any data that can be used to better understand the needs of the customer, excluding any information that is covered by the demographics feature. Examples of personal information are the customer's hobbies or interests.

⁷ Shipping, described in section 5.1.1.2, is a subfeature of the physical goods fulfillment feature.

- *Preferences.* Preferences are options that allow a customer to customize their e-shop interface. They can include site options, such as the site layout, how many items to display in a product list, and the preferred language for rendering the site and any other correspondence.
- *Reminders.* Reminders are customer-requested notifications for pre-defined events. Notification events include informing a customer when a product becomes available or when the price of the product changes. When the event occurs, the customer will receive a notification through a communications channel, such as an e-mail or an on-screen reminder while browsing the e-shop. A customer can create, edit and delete their reminders.
- *Quick Checkout Profile.* The quick checkout profile is stored in the customer profile and contains default information that is used when placing an order. The information includes the payment information and, if necessary, the shipping information. If there is support for a quick checkout profile, it is recommended that the quick checkout type also be selected; however, if the quick checkout type is selected, the quick checkout profile is required.
- *Custom Fields.* The custom fields allow the e-shop to define additional information to be stored in the registration profile after the e-shop is deployed at run-time. Enabling custom fields requires a mechanism for the e-shop staff to define these fields in terms of their representation, such as the data type, value range, and semantics. The applicability of the custom fields in other e-shop workflows, such as business intelligence, depends on the ability to modify the workflows within the e-shop.

4.2.3 User Behaviour Tracking Information

User behaviour tracking information allows the e-shop to associate data that it collects on user actions to a registration profile. The additional information can be used to interpret the data from a marketing perspective. For example, it can combine the information sources to determine what types of products are browsed by high income visitors or what other sites younger customers tend to visit. This feature requires the selection of the user behaviour tracking feature; however, there is no implication on this feature if the user behaviour tracking feature is selected.

4.3 Catalog

A catalog contains the goods and / or services that an e-shop offers; therefore, an e-shop must support the use of a catalog. The catalog provides a framework to organize the information for goods and services, which can significantly influence the navigability and usability of the e-shop.

The catalog feature is shown in figure 4.6. Since every e-shop offers products, it must define a format for the product information. The remainder of the features, which include categories, multiple catalogs, searching, browsing, and custom views, are optional.

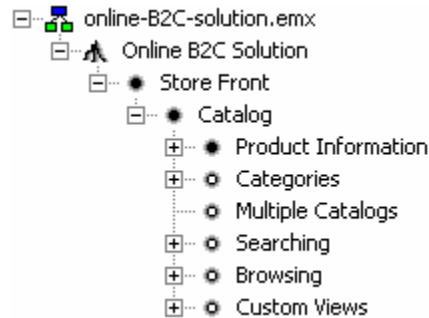


Figure 4.6: Catalog Feature

4.3.1 Product Information

Product information, shown in figure 4.7, describes all of the attributes that are recorded for a product. Every product shares two common features: the product type and basic information about the product. The remaining features are optional.

4.3.1.1 Product Types

Every product can be categorized as one of three types: electronic goods, physical goods or services. Electronic goods exist in digital form only. Examples of electronic goods include e-books or e-certificates⁸. Physical goods have a physical manifestation and are traditionally sold through brick and mortar⁹ retail locations. Examples of physical goods include books, electronic equipment or hardware. A service is the “non-material equivalent of a good” [Wik06b] in which the application of specialized resources, such as an expert or a computing resource, is used to satisfy a customer’s need. Traditional services tend to be installation, repair and maintenance jobs, but services can also include things like consulting and data processing.

An e-shop can offer any combination of product types, but the selection of each type imposes additional requirements. Electronic goods require a mechanism to manage the digital rights and a

⁸ An e-certificate is an electronic gift certificate which can be redeemed at the e-shop.

⁹ This term is commonly used to describe a physical store.

secure distribution channel. Physical goods require shipping support in order to deliver the product to the customer. Services require a scheduling facility for appointments and resource management. These additional requirements correspond to the fulfillment features. Each product type has a corresponding fulfillment feature which is required; however, if a fulfillment feature is selected, it is recommended that the corresponding product type be selected also. Finally, since some product type specific features, such as weight, are not specified as subfeatures of the product type, additional constraints are required to express these relationships.

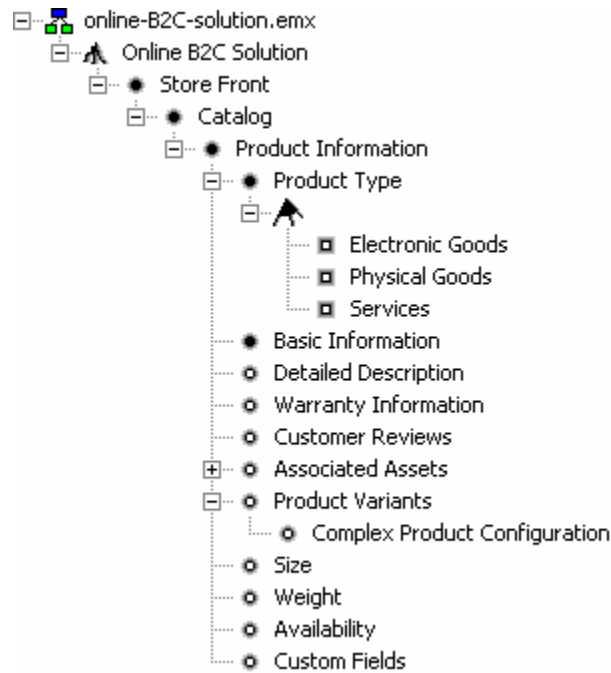


Figure 4.7: Product Information Feature

4.3.1.2 Basic Information

Basic information describes a product. The minimal set of information is the product name and a unique identifier for the product, such as a company-assigned product ID, a Stock Keeping Unit (SKU), or a Universal Product Code (UPC).

4.3.1.3 Detailed Description

The detailed description provides additional information about the product's features, as well as any other details that the e-shop or supplier feels are relevant for the customer. It does not have

any specific structure; it is usually represented as freeform text. The detailed description is displayed on the product page.

4.3.1.4 Warranty Information

Warranty information provides details about how long the manufacturer, supplier or retailer will guarantee the functionality of a product or the work produced by a service. It can also provide information about the policy for defects. A warranty can be broken down for different components, such as different part types or different defect types, and each component can be covered differently. The key information is usually the period of time for the warranty coverage. If there are additional terms or conditions, they can also be specified. Depending on the warranty information required, it can be rendered in a tabular format or as freeform text.

4.3.1.5 Customer Reviews

Customer reviews allow customers to share their opinion of a product by rating the product and / or posting comments. This information is displayed on the product page. Ratings tend to be a numerical value on a pre-defined scale and comments tend to be freeform text. Due to the interactive nature of this feature and the potential for libel, the ability to post reviews requires the customer to be registered. In addition, a moderator may review the comments before making them visible publicly.

4.3.1.6 Associated Assets

Associated assets are a set of files that describe or illustrate a product. They can be used to preview electronic goods, present physical goods, or demonstrate a service. The structure of the associated assets feature is shown in figure 4.8. The two file types are documents and media files. Document files consist of product brochures, installation manuals and supplemental information about the product; they can be implemented as a separate file, such as a Portable Document File (PDF), or as another web page. Media files consist of image, sound and video files. Sound and video files do not exhibit a high degree of variability with the exception of the encoding formats that are supported. Choices for these formats are technical in nature and, therefore, too low level for this model. In addition, there are many encoding formats for images which are omitted for the

same reasons, but there are also many image types that can be supported by the e-shop. The image types are described next.

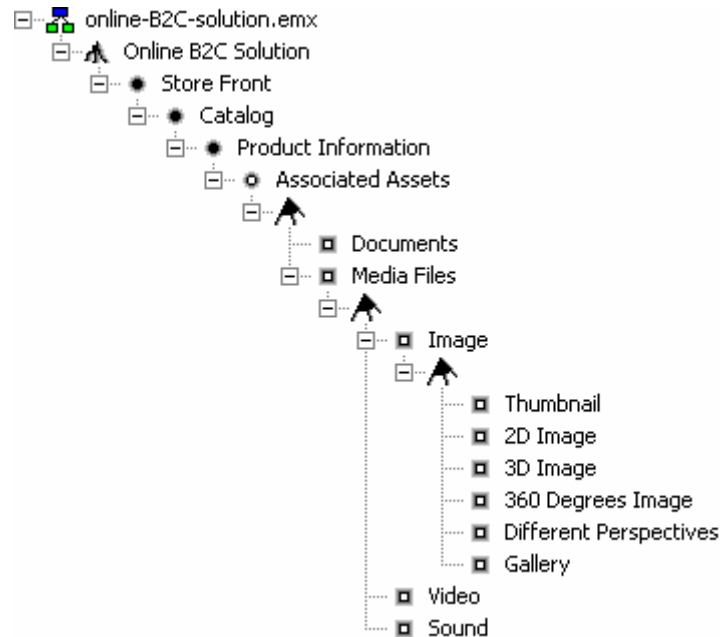


Figure 4.8: Associated Assets Features

- *Thumbnail*. Thumbnails are relatively small, preview-sized image files that are usually linked to a full-sized image. They are used in product lists and product pages to conserve space on the page, thus allowing more information to be displayed. The smaller file size also allows the page to load faster.
- *2D Image*. 2D images are regular pictures, such as photographs.
- *3D Image*. 3D images are pictures which appear three-dimensional. They may require additional technology, such as 3D glasses, to view the three-dimensional effect.
- *360 Degree Image*. 360 degree images provide images that encircle the subjects on a plane. These images are popular when documenting locations. In such images, the camera is rotated on a single spot and images are taken from different angles. The images are then combined into a single image. An example of a 360 degree image is a panoramic image.
- *Different Perspectives*. Different perspectives provide multiple images which show the product from different angles. It may also be an interactive image which allows the viewer to manipulate the viewing angle. Common perspectives are front, rear, side, overhead and angled shots on the subject. For example, books are usually shown from two perspectives: the front cover and the back cover. Different perspectives can also mean different situations, such

as a low-lighting environment for products that glow or time elapsed perspectives for products that grow.

- *Gallery*. A gallery contains a set of product images.

4.3.1.7 Product Variants

Product variants are a set of related products where individual products are uniquely identifiable by some variability criteria. An example is a shirt that is available in four sizes and three colours. Although the base product, the shirt, is the same, every combination of size and colour forms a unique product and the set of twelve unique products form the product variants¹⁰. Optionally, the e-shop product can support complex product configuration, which allows customers to create their own configurations of a product for purchase. This feature is ideal for products that require heavy customization, such as computer systems. For example, Dell has a site where customers can specify a computer system by selecting options for the processor, amount of memory, video card and other components. Due to the number of possible configurations, customized products usually have to be built before they can be shipped to the customer.

4.3.1.8 Size

Size is a measurement of the physical dimensions of a product. For physical goods, it is a measure of the length, width and height of the product. This information may play a role in determining shipping options and cost. For electronic goods, size can be interpreted as the file size, which is provided as supplemental information.

4.3.1.9 Weight

Weight specifies the mass of a physical product, which can factor heavily into shipping decisions. Weight is not required for any other product type.

¹⁰ It should be noted that each variant is usually tracked separately using its own SKU [Wik06c]. The current set of models (feature model and store front entity model) does not take this into consideration.

4.3.1.10 Availability

Availability indicates if a product is in-stock in the warehouse and, if not, how long it will take to restock the product. It can also display the number of items remaining for purchase. Supporting availability requires the selection of the inventory tracking feature in order to obtain the necessary data; however, if the inventory tracking feature is selected, selection of the availability feature is recommended.

4.3.1.11 Custom Fields

Custom fields for product information are similar to the custom fields for registration information described in section 4.2.2. It allows the e-shop to define additional data fields to be stored with the product information after the e-shop has been deployed at runtime. The discussion about the limitations and applicability of custom registration information fields applies here as well.

4.3.2 Categories

Categories are grouping mechanisms which allow products to be grouped together based on product characteristics or attributes. A product's category can be assigned manually by an operator or inferred automatically based on a set of attribute values. In a simple category system, the catalog contains multiple categories and every product is contained by a single category. A more complex category system can be created by selecting the optional multi-level and / or multiple classification features. The categories feature and its subfeatures are shown in figure 4.9.

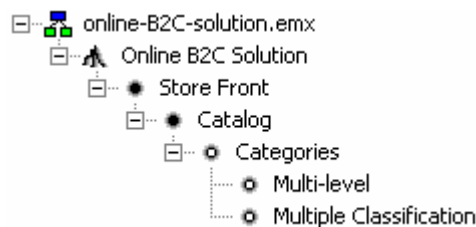


Figure 4.9: Categories Feature

- *Multi-level.* Multi-level categories allow categories to be nested. Nested categories allow categories to be decomposed into finer levels of detail. This can help improve site

navigability by reducing the scope of a category page. Products can be contained at any category level.

- *Multiple Classification.* Multiple classification categories allow a product to be classified under more than one category. For example, an electrical shaver can be classified as both an appliance and a lifestyle product.

4.3.3 Multiple Catalogs

Multiple catalogs provide another method to organize the products in the e-shop. This is useful when an e-shop deals with a large variety of goods. If the products belong to different thematic categories, such as clothing and electronics, these different product categories can be placed in different catalogs. Multiple stores can be setup within the e-shop, where each store sells a particular type of good and is sourced by a different catalog. Another possibility is to use multiple catalogs for back-end organization, but to provide a unified view of the catalogs in the store front.

4.3.4 Searching

The searching feature allows the customer to query the catalog and display the results. The query tools are dependent on the search type, but the results are always in the form of a product list. The search feature is shown in figure 4.10. There are two search types: basic search and advanced search.

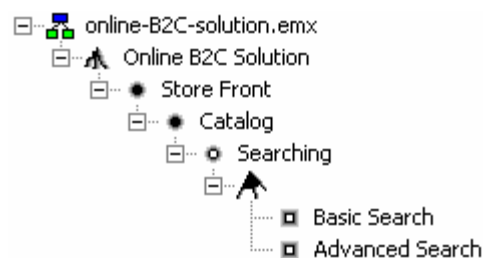


Figure 4.10: Searching Feature

- *Basic Search.* A basic search is based on a few common attributes, such as the product name or keywords. The interface usually consists of a single text box that allows for the search term to be entered. The search term can include common search syntax, such as phrase searching using quotations, and basic Boolean operators, such as AND, OR and NOT. If the

categories feature is selected, the interface may also include a combo box that allows a category to be selected to narrow the scope of the search.

- *Advanced Search.* An advanced search allows a larger set of attributes to be used in the query, such as the product identifier and manufacturer name. It may also search through product-specific attributes. For example, a book has a publisher, but a movie has a production studio and a distributor. The interface must provide the ability to refer to these attributes, associate search values to them, and apply Boolean conditions to form an advanced query.

4.3.5 Browsing

The browsing feature, shown in figure 4.11, allows the catalog to be rendered and viewed by customers. If the feature is selected, the e-shop must support product pages, but category page and index page support is optional.

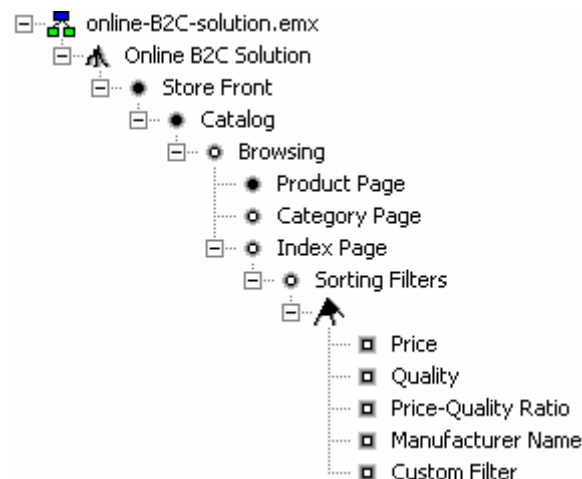


Figure 4.11: Browsing Feature

- *Product Page.* A product page is a web page which describes a single product. The information is usually rendered in the form of text, associated assets or links. It also contains a link to add the item to the shopping cart or wish list. Every product has its own product page.
- *Category Page.* A category page presents a view of the catalog filtered by a category. It can take the form of a product listing of all products in a category, a specially formatted page designed to showcase featured products within a category, a listing of all categories within a

container¹¹, or a combination of the three forms. Value-added information pertinent to the category can also be provided on the category page. One example is a book category page, which can feature new books, a future release schedule, and links to different book categories, such as science-fiction and non-fiction. The category page requires the categories feature to be selected; if the categories feature is selected, it is recommended that this feature is selected also.

- *Index Page.* An index page can be a master list of all products available in the e-shop or a list of products belonging to the same category¹². Due to the potential size of such lists, the optional sorting filters feature allows sorting criteria to be applied to the list. Filters can include sorting by the product price, quality rating, price-to-quality ratio, the manufacturer's name or by some custom filter criteria that is defined at run-time. If the sorting filter is not selected, the products are presented in some default order, such as the order of the data entry.

4.3.6 Custom Views

A custom view defines filters on one or more catalogs in order to create a specialized store or highlight certain products. The filters are based on product attributes or, if available, categories. It may be possible to define the view and manually add products to it, but e-shops with a large number of products are unlikely to use such an approach. Similar to database views, custom views on the catalog do not affect the underlying structure of the catalog or products. Two potential applications of a custom view, as shown in figure 4.12, are to create a seasonal product view or a personalized view for customer groups or distributors.

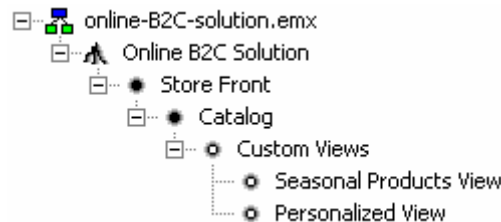


Figure 4.12: Custom Views Feature

¹¹ A category can be directly contained by the catalog or by another category if the multi-level feature is selected.

¹² Although multiple definitions are presented for the category and index page features, it is up to the reader to choose a set of definitions which do not conflict with each other.

4.4 Wish Lists

A wish list is a view of the catalog defined by the customer. It allows them to keep track of products in the e-shop that they would like to purchase or receive as gifts. Wish lists can also be used by customers to keep track of products and their prices or by the marketing department to gather business intelligence data for targeting consumers. A wish list is stored with the registered customer's profile; however, guests can keep wish lists if the wish list saved after session subfeature is selected. Therefore, the wish list feature requires the selection of the registration feature and / or the wish list saved after session subfeature.

Figure 4.13 shows that wish lists can be augmented by optional features, such as saving them after a session, enabling them to be sent to others, allowing a customer to have multiple wish lists, allowing permissions to be set on them, and updating them automatically.

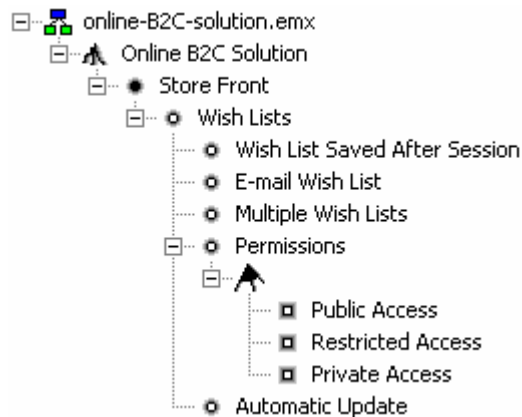


Figure 4.13: Wishlists Feature

- *Wish List Saved After Session.* This feature is required to allow guests to keep a wish list. The guest's wish list is stored locally on the guest's computer. The saving mechanism is implicit since it is assumed that a guest who puts the effort into creating a wish list intends to access it later. This feature does not affect registered users since their wish list is stored automatically; however, this feature is required if the wish lists feature is selected and the registration feature is eliminated because a wish list must be accessible to guests if registered customers are not supported. If this feature is selected, only the wish lists feature is required due to the subfeature relationship; there is no implication on the registration feature.

- *E-mail Wish List.* The e-mail wish list feature allows a customer to send their wish list to friends, family members and other potential gift-givers. The e-mail will contain either a link to the wish list or a list of products on the wish list with links to the product pages. This feature is only available to registered users.
- *Multiple Wish Lists.* Multiple wish lists allow customers to create and maintain more than one wish list. Each wish list must be named for identification purposes. A practical application for multiple wish lists is to use them to generate a unique list for each gift giver to prevent the receipt of duplicate items.
- *Permissions.* Permissions allow the customer to set the visibility of a wish list for other visitors. When selected, this feature allows potential gift givers to search the site for the customer's wish list. Permissions can include making the list publicly accessible, restricting access by password or e-mail invitation, or restricting access completely. If there are multiple lists, permissions may be coarse-grained, meaning the same permission applies to all lists, or fine-grained, meaning each list can have different permissions. This feature requires the registration feature since lists can only be accessible to other visitors if they are stored on the server; however, there is no implication on the registration feature if this feature is selected.
- *Automatic Updating.* Automatic updating modifies the wish list by removing items as they are purchased, either by the customer or a gift giver. One way to implement this is to update the wish list only if the item is purchased through the link on the wish list page.

4.5 Buy Path

Buy path is a grouping of features relating to the customer purchases workflow. It starts with the checkout process and ends with order placement [Ped03]. This can include actions like displaying items in a shopping cart and entering order information. Buy path, shown in figure 4.14, consists of three required features: shopping cart, checkout and order confirmation.

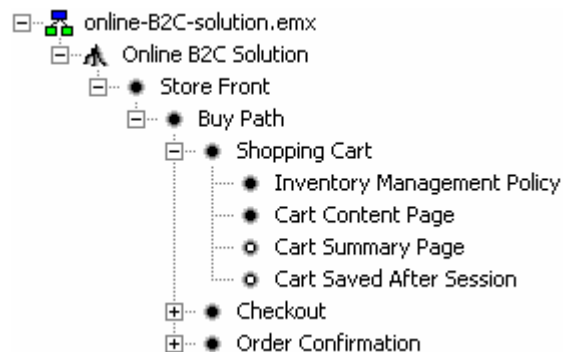


Figure 4.14: Buy Path and Shopping Cart Features

4.5.1 Shopping Cart

The shopping cart allows a customer to keep track of the items that they wish to purchase during their shopping session. The cart contains a list of products and each product is associated with the quantity that the customer would like to purchase. Placing an item in the shopping cart implies the intent to purchase, but there is no obligation for the customer to complete the transaction. Selection of the shopping cart feature also requires the inventory management policy and cart content page features to be selected. The shopping cart can be further enhanced by the optional cart summary page and cart saved after session features.

- *Inventory Management Policy.* This feature allows the e-shop to specify how actions on the shopping cart affect the inventory systems. When an item is placed into a cart, the e-shop may reserve an item from the inventory. This is good in the sense that it ensures that the customer is guaranteed the item when they order it, but it becomes problematic if there are many customers who are placing the item into the cart but not purchasing them. Therefore, it may make sense to release the inventory when the customer's session ends. When an e-shop is selling distinct items, such as tickets for events with reserved seating, items can be sold out very quickly. Therefore, the e-shop must associate a timer for each item once it is presented to the customer. If the item is not ordered before the timer expires, the item will be released. Another policy is to delay item reservation until later in the checkout process; however, a customer may be irritated if they learn that an order cannot be fulfilled after they have entered all of their order information. Depending on the policy selected, this feature may require the inventory management feature.
- *Cart Content Page.* The cart content page allows a customer to view all of the items that have been placed in the cart. This page also allows the customer to edit the quantity of an item or remove an item from the shopping cart. Each product is listed along with the desired quantity and the subtotal; the cart total may also be included. Tax and shipping costs are usually excluded since they require additional information before they can be calculated. The page also contains a link that is used to start the checkout process.
- *Cart Summary Page.* The cart summary page contains information that is similar to the information found in the cart content page; however, the information may be condensed. The cart summary page does not provide the ability to edit the cart contents directly, but it will contain a link to the cart content page. It can be used to confirm the addition of an item into

the cart, which is what Amazon.com does, or to display a summary of the items for confirmation purposes before the order is placed.

- *Cart Saved After Session.* This feature allows customers to save their cart contents for their next visit, which is useful if they leave the store unexpectedly (e.g. due to accidental browser closure, browser failure or network problems) or want to complete the purchase at a later time. There are two factors which affect this feature. The first factor is the types of customers to which this feature applies; it can apply to registered customers, guests or both. The second factor is where the cart data is saved; it can be saved locally on the visitor's machine or remotely on the e-shop's servers. Table 4.1 defines the issues that arise depending on the combination of factors. Regardless of the factors, cart synchronization must be addressed when the product's price or availability changes while the product is sitting in a saved cart. The synchronization strategy must be one that maintains data consistency while ensuring that the customer is informed and their inconvenience is minimized.

Table 4.1: Behaviour Analysis of Cart Saved After Session Feature

<i>Data Storage</i>	<i>Applicable Groups</i>	
	Registered Customers	Guests
Stored Locally	This requires the use of a local file cache and implies that the cart data is not portable. It is possible to associate the cache to a registered customer so that other users who visit the e-shop from the computer will not have access to the cart. This technique can be combined with storing the data on the server to improve performance for the customer.	This requires the use of a local file cache. Since there is no identification to distinguish different guests, anyone who visits the e-shop as a guest from the computer will have full access to the cart.
Stored on Server	This applies if the cart data is stored in the profile; it implies that the cart data is portable and loaded every time the customer logs in. If the data is not stored with the profile, then the same method for storing guest data on the server can be used.	This requires a method to allow the guest to retrieve the cart, such as a parameterized URL. An expiry date is needed to prevent server storage space from being depleted by orphaned cart data.

4.5.2 Checkout

The checkout feature, shown in figure 4.15, encapsulates the features related to the checkout process. In the process, the customer reviews the items they have added to their shopping cart, enters their payment and shipping information, selects any shipping and gift options, and confirms the order. The process begins when the customer has finished selecting their items and ends with the submission of the order to the order processor. Checkout requires that the checkout

type, taxation options and payment options features be selected; the shipping options feature may also be selected if it is necessary.

4.5.2.1 Checkout Type

Figure 4.15 shows the details of the checkout type feature. There are two checkout types: registered and guest. An e-shop can support both types simultaneously, but the customer will select which checkout type to use during their session at run-time.

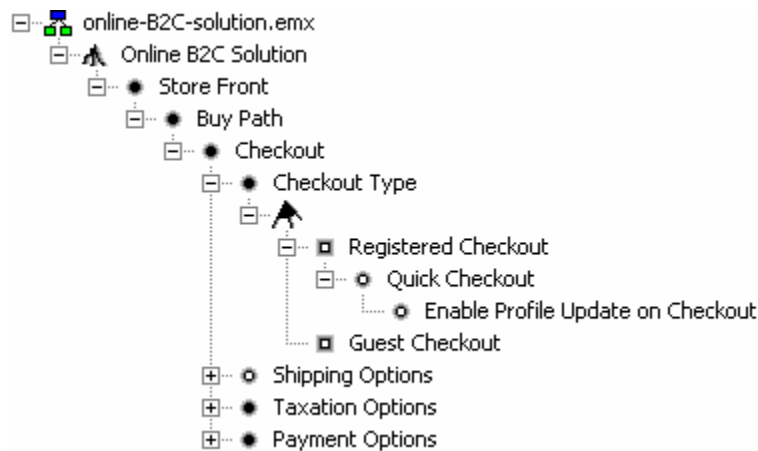


Figure 4.15: Checkout and Checkout Type Features

- Registered Checkout.** The registered checkout requires customers to log in before they can start the checkout process. During the checkout, customers must enter or select their shipping and payment information for the order. Support for a registered checkout requires the selection of the register to buy policy in the registration enforcement feature; if the register to buy feature is selected, the selection of this feature is recommended. The quick checkout is an optional feature that allows customers to place an order for the items in their shopping cart by using a default set of information from their profile. This feature requires the quick checkout profile feature in order to store the default shipping and payment information. Amazon.com's "1-Click Ordering" is an implementation of such a feature except that it applies to a single item instead of the cart contents. The optional enable profile update on checkout feature allows customers to automatically propagate any profile information or default selection changes to the quick checkout profile when they are using the regular checkout.
- Guest Checkout.** The guest checkout allows a guest to purchase products from the e-shop. Guests have to enter their personal information to place an order. The information will be

stored to fulfill the order and for regulatory reasons, but it will not be available for reuse in a future purchase.

4.5.2.2 Shipping Options

The shipping options feature describes the options pertaining to shipping which the customer has control over while going through the checkout. Shipping options require the selection of the shipping feature¹³; if the shipping options feature is selected, there are no implications on the shipping feature since the shipping options only determine the degree of control the customer has on the shipping process. Figure 4.16 shows the details of the shipping options feature. There are many optional features in shipping options, including quality of service selection, carrier selection, gift options and multiple shipments; however, the shipping cost calculation feature is mandatory since it is needed to generate the total cost of the order.

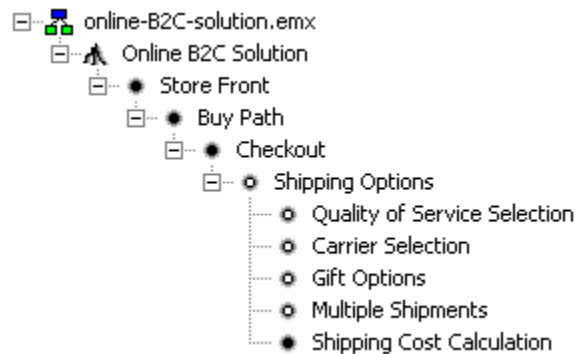


Figure 4.16: Shipping Options Feature

- *Quality of Service Selection.* The quality of service selection feature requires the customer to specify the level of service they want. It is usually expressed in terms of the number of days it takes for the shipment to arrive. Some common options for quality of service include “express”, “air”, “priority” and “next-day shipping”. If this feature is eliminated, the quality of service is determined by the e-shop.
- *Carrier Selection.* The carrier selection feature requires the customer to choose the company to perform the delivery service. There may be constraints between the carrier and quality of services available; however, those constraints are dependent on any business arrangements

¹³ Shipping, described in section 5.1.1.2, is a subfeature of the physical goods fulfillment feature.

made between the e-shop and the carrier. This option is very rarely used in practice. If this feature is eliminated, the carrier is chosen by the e-shop.

- *Gift Options.* The gift options feature allows the customer to designate an order or part of an order as a gift. This means that the gift portion of the order may be sent to a different recipient. Some e-shops cannot process multiple shipping addresses for a single order, so placing multiple orders may be necessary. In addition, a gift receipt should be issued and gift wrapping options must be performed before the gift is shipped.
- *Multiple Shipments.* The multiple shipments feature allows the customer to partition their order into multiple deliveries. Each delivery can be configured individually with its own shipping options. This is a useful feature when the customer wants to expedite the delivery of some items in an order. It is also useful when items have different availabilities; this prevents the customer from having to wait for all the items to become available before receiving their order. The most common way to charge for multiple shipments is to perform the shipping cost calculation as if each individual shipment comes from a separate order. If this feature is eliminated, the customer will only have the option to receive their order when all of the items are available. It is important to note that this feature does not affect the e-shop's ability to partition shipments and adjust the shipping costs manually.
- *Shipping Cost Calculation.* The shipping cost calculation feature allows the e-shop to assess shipping costs. Many implementations use base shipping rates, which are pre-calculated for different qualities of service and carriers. The base shipping rates are applied against various factors, such as the number of items ordered, the type of items ordered, the size or weight of the items, or the total order cost. For example, an e-shop may have a set of rates for books which state that the first book costs \$x and each additional book costs \$y when the books are shipped using the express service. Another example is when a company offers free shipping if the total order amount exceeds a threshold value. Each e-shop must define and implement its own policies for shipping cost calculation.

4.5.2.3 Taxation Options

The taxation options feature describes all of the options available for applying the tax laws and calculating the amount of tax to be charged on an order. Taxation can be affected by many factors, including the location of the purchaser, the location of the e-shop's company, and the items purchased. Exemptions may also be applicable to certain orders depending on the purchaser or the intended use; special information, such as exemption codes and tax numbers, needs to be

taken into consideration in these cases. In addition, tax laws can vary from region to region. Taxation can be handled through custom taxation rules, a tax gateway or both. The details of the taxation options feature is shown in figure 4.17.

The inclusive-or feature groups under the taxation options and tax gateways features apply to all binding times since an e-shop may want to support custom tax rules and multiple tax gateways simultaneously. For example, an e-shop may specify a set of custom tax rules to handle special situations and refer everything else to a tax gateway. Multiple tax gateways may also be required during a tax calculation depending on the circumstances surrounding the order and the capabilities of each tax gateway. Therefore, a single order submitted during a session at runtime may require a combination of these features.

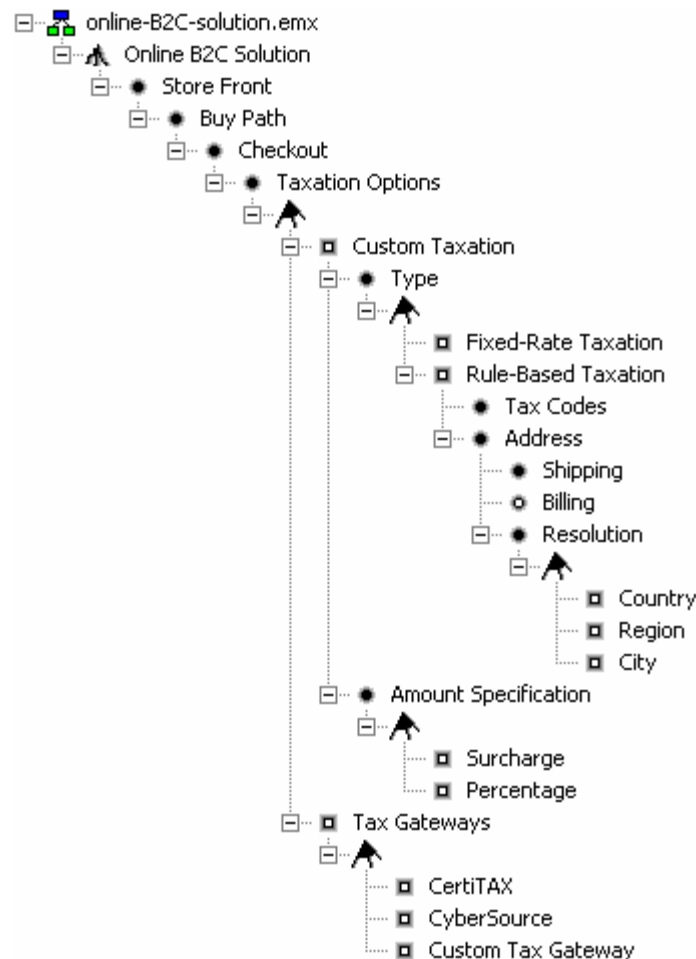


Figure 4.17: Taxation Options Feature

- *Custom Taxation.* The custom taxation feature allows the e-shop to define tax calculation strategies. There are two types: fixed-rate taxation and rule-based taxation. In addition, the amount specification feature is needed to determine how to express the tax rate. Both surcharge and percentage methods can be supported, although a tax rate is only specified with one.
 - *Fixed-Rate Taxation.* Fixed-rate taxation charges a fixed percentage or amount on every order, regardless of the circumstances. This is the simplest way that tax can be calculated. It is rarely sufficient to apply fixed-rate taxation only since tax laws tend to contain many provisions which must be observed.
 - *Rule-Based Taxation.* Rule-based taxation allows the e-shop to define its own tax rules. The tax rules are defined using tax codes and addresses. A tax code defines a category that is associated with a tax; a product is assigned one or more tax codes so that the appropriate taxes can be charged. For example, some regions have an environmental surcharge on electronic products because the products contain materials which are harmful to the environment. Therefore, a tax code can be defined for this environmental surcharge and associated with electronic products sold at the e-shop.
 The address allows the location to be taken into consideration, which is necessary since tax codes define rules for products only. Current Canadian tax laws for Internet purchases state that customers are only responsible for any taxes that are charged within their own province; customers do not have to pay provincial taxes for the province in which the e-shop has a physical presence. Both the shipping and billing addresses can affect the tax calculation. The customer's location is often used to determine which taxes to apply. Since customers usually reside at the shipping address, the e-shop must support it. The billing address may be supported since it is required for tax calculations in some jurisdictions; furthermore, it may also be useful if the shipping address and billing address differ. Finally, the resolution feature is required to determine the granularity of the address. Different tax laws may be applicable at different levels of government, so the address must be resolvable to the country, region, or city. The main problem with using rule-based taxation is that there are a large number of tax rules which are liable to change periodically. This can make it difficult for the e-shop to keep all of the tax rules in compliance with the tax laws and it can cause problems over the long run.
- *Tax Gateways.* Due to the complexity of tax laws, an e-shop may want to outsource the tax calculations. Tax gateways are third parties who provide tax calculation services; most gateways operate as a web service and can be integrated into the e-shop's checkout process.

Tax gateways can include CertiTAX and Cybersource, but the e-shop can also support the run-time addition of custom tax gateways.

4.5.2.4 Payment Options

The payment options feature describes details pertaining to purchase payments from the customer. The payment types feature must be selected since the e-shop cannot process any payments without it. In addition, payment options can be further supported by two optional features: fraud detection and payment gateways. Figure 4.18 displays the details of the payment options feature.

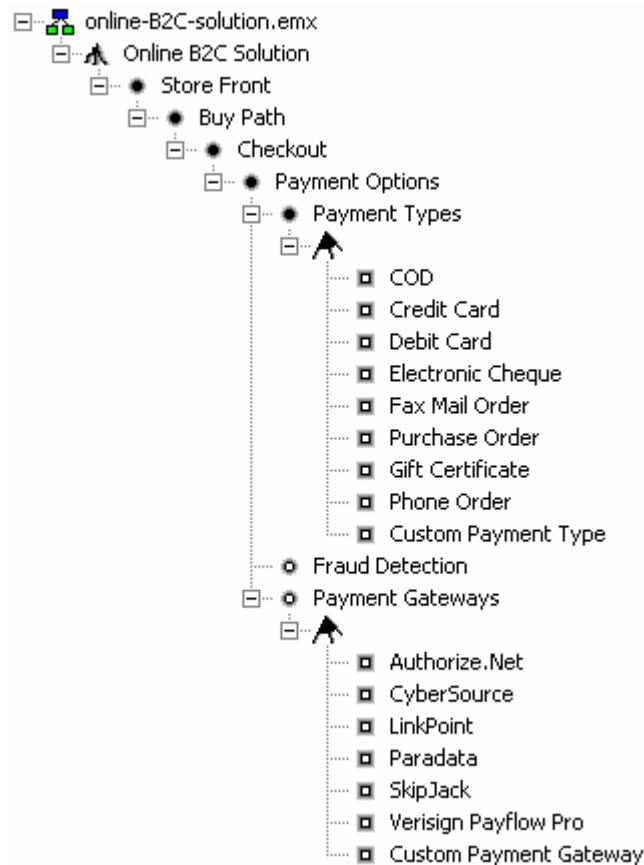


Figure 4.18: Payment Options Feature

- *Payment Types*. The payment types feature denotes the forms of payment that can be handled by the e-shop. Payment types can include Cash On Delivery (COD), credit cards, debit cards, electronic cheques, fax mail orders, purchase orders, gift certificates, phone orders, and a

custom payment type. The feature group remains inclusive-or for all binding times because the e-shop may be required to accept multiple forms of payment for a single order. For example, a customer may make a purchase which exceeds the value of a gift certificate; therefore, a second payment type is needed to cover the balance. Depending on the e-shop, some payment types also have specific information that must be collected during checkout.

- *Fraud Detection.* The fraud detection feature performs checks on the payment information to verify its authenticity. This can be accomplished through card authorization and verification services. Fraud detection also makes use of purchase data, neural networks and rule-based systems to generate a risk score for the e-shop [Vis05]. This feature requires expertise that is usually available through an external service, such as a payment gateway. The specific fraud detection capabilities are dependent on the external service chosen; the actual details are service-specific and beyond the scope of this model.
- *Payment Gateways.* The payment gateways feature allows the e-shop to outsource payment services. Payment gateways are third parties that can handle the verification of the payment information, fraud detection, and payment arrangements with financial institutions. Furthermore, different payment gateways can handle different payment types, so the payment types selected can limit the payment gateways which are available and vice-versa. The feature group remains inclusive-or for all binding times since each gateway has a distinct set of capabilities and processing an order payment may require access to capabilities from multiple gateways. Payment gateways can include Authorize.Net, CyberSource, LinkPoint, Paradata, SkipJack, and VerisignPayflowPro. If the custom payment gateway feature is selected, a payment gateway can be defined and configured after the e-shop is deployed at run-time.

4.5.3 Order Confirmation

The order confirmation feature, shown in figure 4.19, provides an acknowledgement to the customer that the order was received by the order processor and placed successfully. An order number is usually provided to the customer for future reference. This feature is mandatory since customers require feedback after placing an order; otherwise, they may believe that the order submission was unsuccessful and place another order. Order confirmation can be provided through the following communication channels: electronic page, e-mail, phone or mail. Multiple channels may be used to achieve a higher-level of service; therefore, the feature group is inclusive-or for all binding times.

- *Electronic Page.* An electronic page is a web page that is displayed after the checkout process is complete for immediate confirmation. In some implementations, an electronic page can serve as a preliminary acknowledgment for the order. The official confirmation, indicating that the order has been received by the order processor, arrives through another channel at a later time.
- *E-mail Confirmation.* An e-mail confirmation can be sent immediately after the order is placed, after the order processor receives the order, or at both times.
- *Phone.* A phone call can be made to confirm the order if the customer does not provide an e-mail address. It is especially useful for high-risk transactions, such as those involving an expensive purchase.
- *Mail.* A hard copy of the order confirmation from the e-shop can serve as an official document for the customer's records.

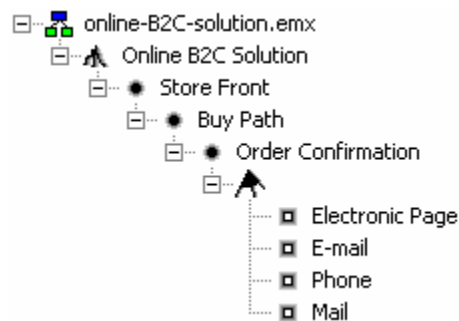


Figure 4.19: Order Confirmation Feature

4.6 Customer Service

The customer service feature, shown in figure 4.20, contains subfeatures that enhance a customer's shopping experience. The optional subfeatures include question and feedback forms, product returns, order status viewing, and shipment status tracking. These subfeatures are usually implemented through additional screens and workflows.

4.6.1 Question and Feedback Forms

Question and feedback forms allow a customer to submit a question or comment to the customer service department directly from the web site. The form can solicit contact information in order to

provide a response to the visitor. There is also the option to track submissions through the question and feedback tracking feature, which allows the e-shop to aggregate the submissions and use the information to identify areas requiring improvement. For example, several similar questions about searching for products may indicate a poor design for the search interface or a lack of documentation for the search functionality. Tracking is done primarily to aggregate the content as opposed to tracing the source of the information; therefore, customers do not have to be logged in to use these forms. This allows visitors to easily pose questions before they commit to sharing their personal information through registration.

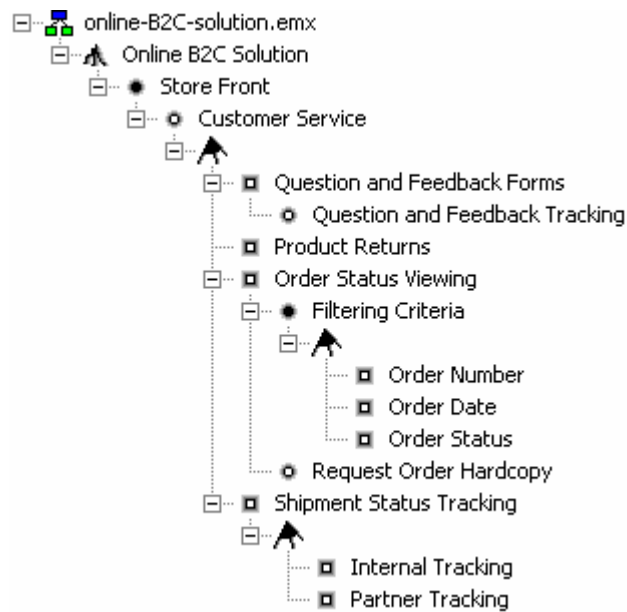


Figure 4.20: Customer Service Feature

4.6.2 Product Returns

Product returns allow customers who are dissatisfied with their purchases to return the items and receive a refund. This feature is only used to accept returns for physical products. The return process for an e-shop is more complicated than the return process for a brick and mortar store because an e-shop requires several actors to be coordinated throughout the process. First, the e-shop must have an interface which allows customers to submit a return request. If the request is approved, a return merchandise authorization (RMA) number, which is used to authorize and track the return, and a shipping label for the return package are generated for the customer. The return package is sent to the warehouse, where personnel inspect the item, confirm that the item's condition is acceptable, and process the item. Once the warehouse personnel have verified the

condition of the item, the original payment is refunded to the customer. The e-shop is also responsible for tracking the state of the transaction throughout the return process.

4.6.3 Order Status Viewing

Order status viewing allows a customer to track orders after they are placed. Customers can retrieve a list of their orders, which can be sorted or filtered by the order number, date, or status. A selection from an order list brings up the order status page. The order status page displays the order number, payment information, shipping information for each delivery, items ordered, costs, discounts, item statuses, and expected arrival dates. The order status viewing feature requires the registration feature; however, the selection of the order status viewing feature has no implication on the registration feature. There is also the request order hardcopy feature, which is an optional subfeature that allows the customer to request a hardcopy of the order page to be sent by mail. The hardcopy can be used as an official proof-of-purchase. The order page is distinct from the packing slip which is included with the product shipment. The difference is that the packing slip is an itemized list of the package contents, whereas the order page includes the full payment details, such as the methods of payment and the total order amount, and is usually sent to the billing address.

4.6.4 Shipment Status Tracking

Shipment status tracking allows a customer to query the state and location of their order once it leaves the warehouse. The information is provided by the shipping company's systems. This feature requires the shipping feature¹⁴ to be selected; however, the selection of this feature makes no implication on the selection of the shipping feature. The two ways of tracking the shipment are internal tracking and partner tracking.

- *Internal Tracking.* The internal tracking feature allows the shipping information to be retrieved from the shipping company's system and displayed on the order status page. Most sites that use internal tracking only display the current shipment status due to the limited screen space on the order status page and / or the limited amount of information provided by the shipping company.

¹⁴ This refers to the subfeature of the physical fulfillment feature. It is described in section 5.1.1.2.

- *Partner Tracking.* The partner tracking feature redirects customers to the shipping company's website so that they can obtain the shipment status. The e-shop's order status page will provide a tracking number and / or a link to the shipping company's website. The link may be parameterized with the tracking number or customers may have to enter the tracking number manually. Since the information is being displayed on the shipping company's website, the shipping company usually provides more detailed information, such as the shipment history, which indicates where the package has been and what time it was there, and the current shipment status.

4.7 User Behaviour Tracking

The user behaviour tracking feature, shown in figure 4.21, allows the e-shop to monitor and record a customer's actions while browsing and shopping. This data can be associated with data from the customer's profile to study trends and consumer behaviour; however, this requires the selection of the user behaviour tracking information feature. The user behaviour tracking feature requires the e-shop to specify which types of behaviour are tracked.

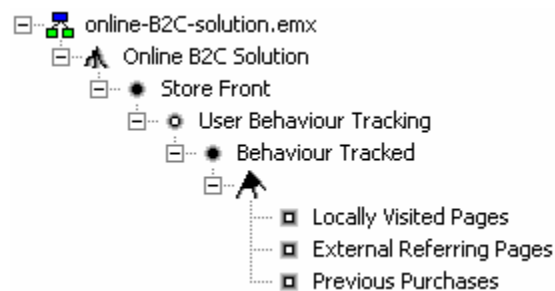


Figure 4.21: User Behaviour Tracking Feature

An e-shop can track users anonymously by session. A session begins when the visitor enters the site and ends when the visitor leaves the site. The data from a session constitutes the behaviour of a single anonymous user.

4.7.1 Behaviour Tracked

Several types of behaviour can be tracked, including locally visited pages, external referring pages and previous purchases.

- *Locally visited pages.* Locally visited pages refer to pages in the e-shop which are visited by the user. Information can be recorded about which internal pages are visited, the order of page traversal, the duration spent on browsing each page, entry and exit points for the e-shop, and whether a page led to a sale. This type of information is helpful when assessing the performance and effectiveness of the e-shop. In addition, this information can also be used for targeting efforts, such as developing personalized advertisements based on the pages visited.
- *External referring pages.* External referring pages refer to tracking entries into the e-shop which originate from links on external sites. There are many applications for this type of information; three examples are 1) to help determine the effectiveness of any external marketing campaigns and the exposure of the e-shop on the internet; 2) to gauge how visible the e-shop is in search engines, since external links are considered in some page ranking algorithms; and 3) to calculate the commission for affiliates.
- *Previous purchases.* Previous purchases refer to tracking purchases made by customers. This can involve keeping a list of items and the quantities that have been purchased or storing all previous order data. This type of information can be used to understand individual and global trends. An example of an individual trend would be studying a consumer's purchasing patterns to determine what and when an individual is most likely to make a purchase. An example of a global trend would be correlating the purchase information with other data to assess the effectiveness of the web site. For example, a low purchase rate on a product may be correlated with a low hit rate on the corresponding product page; this may indicate a navigability problem to that particular product page.

4.8 Summary

In this chapter, the domain analysis for the store front features in the e-shop was presented. The store front consists of features which affect the interface and affect the customer's shopping experience. The key features in the store front include: registration, which allows customers to enter their information so that it can be reused for future purchases; catalog, which defines the structure of the product information and determines which types of products are offered by the e-shop; wish lists, which allow customers to maintain a list of items they would like to purchase; buy path, which describes the features relating to the purchase of an item; and user behaviour tracking, which allows the e-shop to record a visitor's actions for future study and marketing purposes.

Chapter 5 E-Commerce Domain Analysis: Business Management¹⁵

Business management deals with aspects pertaining to the e-shop's operation. Most of these aspects are back-office concerns, such as product management, order processing and marketing, which are handled by the e-shop staff. Business management features can involve different stakeholders from both inside and outside the company. An internal stakeholder is the e-shop management; external stakeholders are suppliers and third parties providing gateway services. Figure 5.1 illustrates the top-level details of the business management feature. Business management requires the order management and administration features, which enable order processing and general e-shop management capabilities respectively. Business management capabilities can be augmented by selecting optional features, such as targeting, affiliates, inventory tracking, procurement, reporting and analysis, and external systems integration¹⁶.

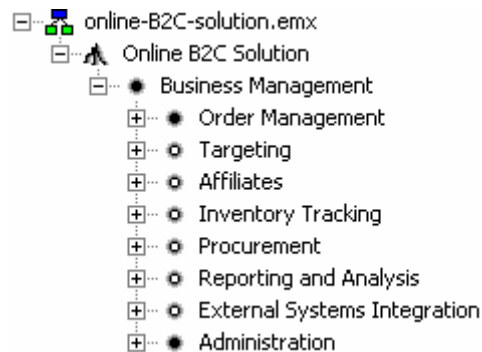


Figure 5.1: Business Management Feature

5.1 Order Management

The order management feature, illustrated in detail in figure 5.2, deals with the workflow that describes the complete lifecycle of an order. The order workflow combines *order handling* and *order processing* activities to complete the business transaction. Order handling is responsible for “paperwork” activities; it performs four tasks: 1) creates the order when a request is submitted by the customer, 2) stores the order in the database for future processing and viewing, 3) updates the order status as it is being processed, and 4) closes the order once it has been fulfilled. Order processing carries out the steps needed to fulfill the order, which requires communication

¹⁵ The additional constraints are described in table A.1 in appendix A.

¹⁶ The level at which a section is nested may not correspond to the depth of the feature because some branches contain features which are used to organize subfeatures only; however, the bullet points always represents the relative depth within each branch.

between stakeholders. Order processing performs four tasks: 1) checks for product availability with the fulfillment centre, 2) verifies the payment information, 3) sends a request to the fulfillment centre to dispense the goods or have the service performed, and 4) makes arrangements for the payment. The order workflow is executed by the order processor. The degree of automation in the workflow depends on the e-shop's policies. For example, an e-shop may require orders over a certain value to be manually released for shipping to prevent fraud. Most e-shops have a fully automated workflow, especially large e-shops with a high volume of transactions, because manual operations would increase costs significantly. Certain anomalies or error conditions may require an order to be flagged for inspection by e-shop staff, but the order processor can still manage the order status and customer notification automatically. Some e-shops use a system where customers are notified if a problem occurs with their order; it is the customers' responsibility to contact the e-shop or they risk having their order cancelled.

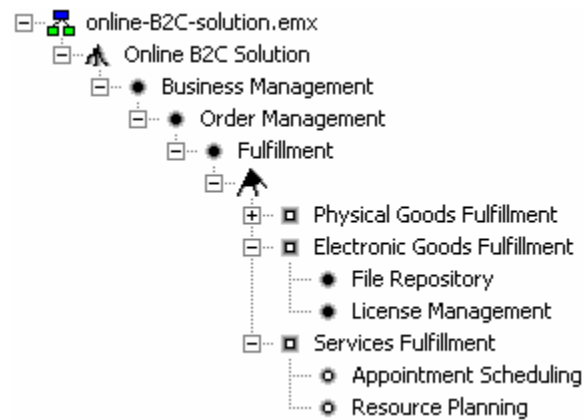


Figure 5.2: Order Management and Fulfillment Features

The fulfillment feature is mandatory¹⁷ and configures options for two tasks: 1) storing inventory in the fulfillment centre until it is sold to customers, and 2) delivering the goods or services to the customer. There are three types of fulfillment: physical goods fulfillment, electronic goods fulfillment, and services fulfillment. Like the product types feature in the store front, the feature group is inclusive-or for all binding times. There is an implication between each product type and its respective fulfillment type; however, if the fulfillment type is selected, it is recommended that the corresponding product type be selected also.

¹⁷ Although fulfillment is the only subfeature of order management, order management can consist of much more than just fulfillment; however, those other features are beyond the scope of this model.

5.1.1 Physical Goods Fulfillment

Figure 5.3 describes the details of the physical goods fulfillment feature. Fulfillment of physical goods requires a real-world fulfillment centre, such as a warehouse. The operations of the warehouse are handled by the warehouse management feature. Product delivery is achieved through the shipping feature, which provides back-office shipping options.

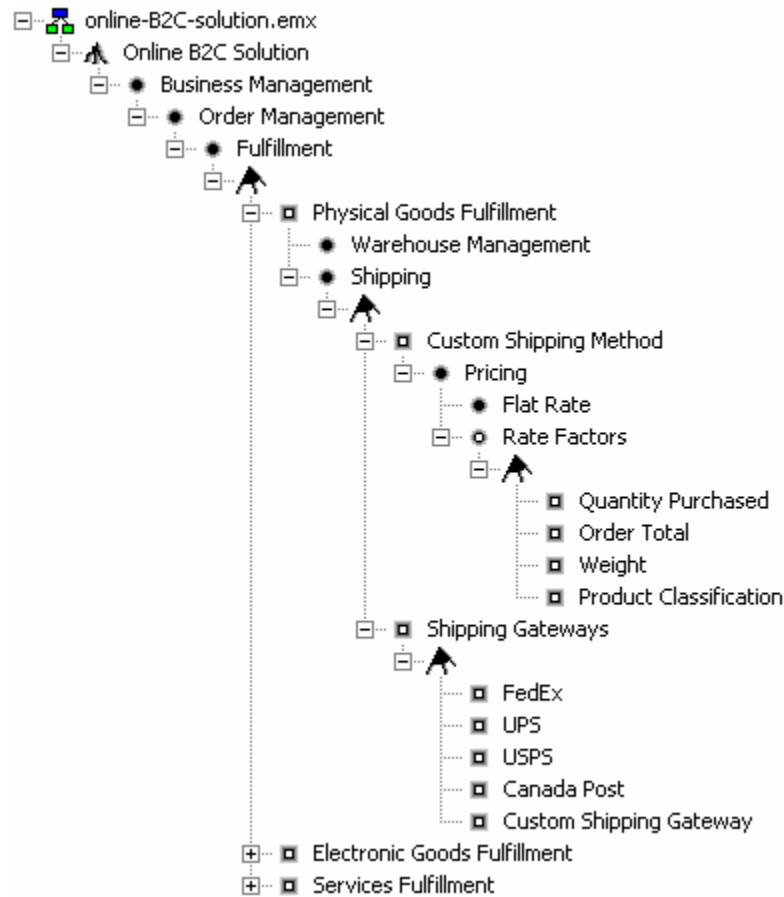


Figure 5.3: Physical Goods Fulfillment Feature

5.1.1.1 Warehouse Management

The warehouse management feature provides tools that allow the warehouse staff to “control the movement and storage of materials within an operation and process the associated transactions” [Inv03]. Since all physical inventories reside in the warehouse at some point, the system must be capable of organizing the flow of products into and out of the warehouse. To support this functionality, the system enables the staff to perform various tasks, including 1) providing

directives to workers or machines on how to sort and where to direct incoming shipments from suppliers, 2) querying the location and quantity of inventory in the warehouse, 3) ordering inventory to be packaged for delivery and arranging a pick-up with the shipping company, and 4) receiving items meant for return processing.

To accomplish these tasks, the warehouse needs to have the technology to link data acquisition devices to the warehouse database. In addition, a communications infrastructure with other units, such as the order processor, is required. Warehouse management systems are separate from the rest of the e-shop; they can be built in-house or purchased as a third-party solution.

5.1.1.2 Shipping

The shipping feature allows the e-shop to define options for the delivery of physical goods. There are two ways in which shipping can be handled: a custom shipping method and shipping gateways. The feature group is inclusive-or for all binding times up to the placement of an individual shipment at run-time because an order may require multiple shipping methods to satisfy different quality of service requirements. For example, if an order has multiple shipments, the e-shop can employ a driver for rush items and a third party for the rest of the order. For an individual shipment, the feature group becomes exclusive-or since only one shipping method will be used per shipment. Finally, the selected subfeatures determine the shipping options which are available to the customer when they are placing an order at run-time.

- *Custom Shipping Method.* The custom shipping method allows the e-shop to define a shipping method. It can be used as a standalone shipping method, such as hiring a driver for a special delivery, or it can be used to implement special shipping policies, such as offering free shipping on certain orders. The pricing subfeature describes ways of calculating the shipping cost. The simplest way is to use the flat rate feature, which assigns the same shipping cost to every order. It is a mandatory feature because it is a basic accounting feature for shipping. The base case, which is no charge for shipping, can be implemented by setting the fixed rate to zero. A more realistic implementation involves taking the order details into consideration. The rate factors feature allows four different criteria to be used in the calculation of shipping costs.
 - *Quantity Purchased.* Quantity purchased refers to the number of items in an order. More items in an order usually leads to higher shipping costs; however, many companies

- charge a higher shipping rate on the first item to cover initial costs and a reduced rate on additional items. The initial costs include the packing materials and labour.
- *Order Total*. Order total refers to the total amount of the order. This can be used to set a threshold amount for free shipping or it may use different value ranges to determine the shipping cost.
 - *Weight*. Some shipping services charge for deliveries by weight. There can be a graduated pricing scale based on the range that the weight falls in or a fixed rate per unit of measure.
 - *Product Classification*. Product classification refers to different classifications of physical goods, such as books, clothing, or electronics. These goods may have similar physical characteristics, so the shipping requirements may also be similar; therefore, a rate can be defined for the class of products.
 - *Shipping Gateways*. Shipping gateways are third parties who provide the shipping service. Shipping gateways allow shipping costs to be calculated and deliveries to be scheduled. The former is handled during the checkout process, although the e-shop may perform some pre-processing to apply its own shipping policies; the latter occurs when the product is ready to be shipped from the warehouse. The shipping gateways consist of major shipping companies, including FedEx, UPS, United States Postal Service (USPS), and Canada Post. The custom shipping gateway feature allows the e-shop to define and configure different shipping gateways after the e-shop has been deployed at runtime. The feature group for the shipping gateways feature varies according to the binding time in the same way as the feature group for the shipping feature due to the same line of reasoning that was presented for the latter feature group.

5.1.2 Electronic Goods Fulfillment

Figure 5.2 describes the details of the electronic goods fulfillment feature. For electronic goods, the role of the fulfillment centre is handled by the file repository feature. Product delivery is usually handled through a file transfer or an e-mail. The license management feature is also required to prevent unauthorized duplication of copyrighted content. This is usually achieved using some form of Digital Rights Management (DRM).

5.1.2.1 File Repository

Electronic goods have no physical presence but they exist in various forms. Digital content, such as music or e-book files, can exist in the form of a source file. When customers purchase digital content, they receive a licensed copy of the file which is encoded with copy protection mechanisms. Alternatively, an electronic goods product like an e-certificate may exist in the form of a unique key; these keys may be automatically generated on demand or they may be pre-defined. In both cases, the keys must be stored for future redemption. Since both forms of electronic goods require storage, the role of the fulfillment centre is handled by the file repository feature, which securely stores the content for distribution or retrieval. The inventory for electronic goods is handled differently than the inventory for physical goods because the digital product is usually replicated on demand; therefore, only one copy of the product needs to be stored and the inventory will never be depleted. Furthermore, the capacity of an electronic fulfillment centre is only limited by hard drive space, which is cheaper and easier to expand than physical space.

5.1.2.2 License Management

License management protects digital content from unauthorized use and duplication. It allows the e-shop to track the content, set access policies, and encode the product with access rights. Access rights include the ability to control the number of times the content can be viewed, the content expiry date, and the copy privileges, such as the ability to transfer the content to another device. Enforcement of the digital rights can occur through the use of a proprietary content viewer, such as Apple's iTunes; the online verification of the content each time the content is viewed, such as the Digital Video Express (DIVX) movie format [Wik06d]; or the use of a special type of media, such as disposable Digital Video Discs (DVDs) which can only be read for a set time limit. Many large software companies and media companies have their own encoding schemes and standards for DRM. The license must be encoded into the product after it is replicated from the file repository, but before it is served to the customer.

5.1.3 Services Fulfillment

Figure 5.2 describes the details of the services fulfillment feature. Fulfillment of services is different from the other two fulfillment types because there are no goods to store in inventory or deliver to customers. To fulfill a service, a company requires resources, such as service personnel,

facilities and equipment. These resources are managed by tools which are defined by the optional appointment scheduling and resource planning features.

- *Appointment Scheduling.* The appointment scheduling feature allows the e-shop to define and manage appointments. Appointments are constrained by the availability of the customer and the company's resources. Scheduling can be performed by the e-shop staff or by the customer. In either case, the e-shop usually defines a set of appointment slots and the customers pick a slot. A basic implementation of this feature involves managing appointments only. Additional features, such as automatic notifications to service personnel when an appointment is scheduled or reminders to customers for upcoming appointments can help streamline the fulfillment process. Appointment scheduling is required for all services except those that operate on a first-come, first-serve basis. For example, an online service that converts documents into PDF files can provide an immediate turn-around or, if it is heavily loaded, service in the order of arrival; there is no need to schedule an appointment to have this service performed.
- *Resource Planning.* The resource planning feature allows the e-shop staff to allocate resources to projects and assess the e-shop's capacity for providing services. This can be used for both long-term planning, such as determining how many more service personnel and how much more equipment would be required to increase the service capacity by x%, and short-term planning, such as determining how much supplies are required for a job and whether or not they have to be restocked. This feature is optional since resource planning can be performed offline also.

5.2 Targeting

Targeting refers to marketing efforts which focus on meeting the needs and preferences of individual users [Ped03]. The objective is to improve business by attracting new customers, increasing store traffic and increasing overall sales. Targeting can be pre-emptive, which implies the execution of a marketing strategy based on an analysis of the collected data, or reactive, which implies a dynamic response based on a customer's actions.

Targeting impacts all areas of the e-shop, from creating additional promotional web pages to modifying the checkout workflow to accept coupon codes. Targeting is defined by three mandatory features: targeting criteria, targeting mechanisms, and display and notification. In

addition, a cohesive targeting strategy can be defined, implemented and executed through the optional campaigns feature. The targeting feature is shown in figure 5.4.

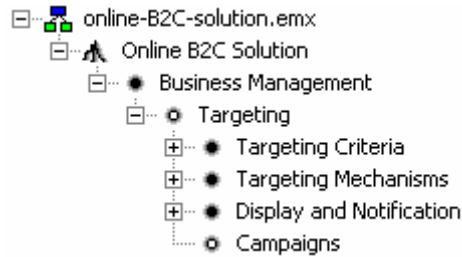


Figure 5.4: Targeting Feature

5.2.1 Targeting Criteria

The targeting criteria feature, shown in figure 5.5, defines a set of features for selecting a group of customers, also referred to as a target audience. The commonality between the customers will make it easier to design an effective marketing strategy. The feature group is inclusive-or for all binding times. Many of the criteria are the same as the fields defined in the registration profile; in those cases, selection of the grouped feature here requires selection of the corresponding feature in the profile. However, there are no implications on the grouped features if the corresponding feature is selected in the registration profile.

- *Customer Preferences.* Customer preferences relate to the preferences feature defined in the registration profile. It can be used to group customers based on their language preferences, which is useful for e-shops that sell multilingual products.
- *Personal Information.* Personal information can be used to select products that relate to the customer's interests and a promotion can be created based on the selected products. The information can be obtained directly from the profile or inferred from other pieces of data, such as the customer's newsletter subscriptions.
- *Demographics.* Demographics provide objective attributes for comparing customers and defining customer groups because demographic information, such as age, income and education, can be clearly defined and the values require no subjective interpretation. Demographics can also be used to predict consumer behaviour; for example, customers with higher incomes are more likely to purchase high-end products.

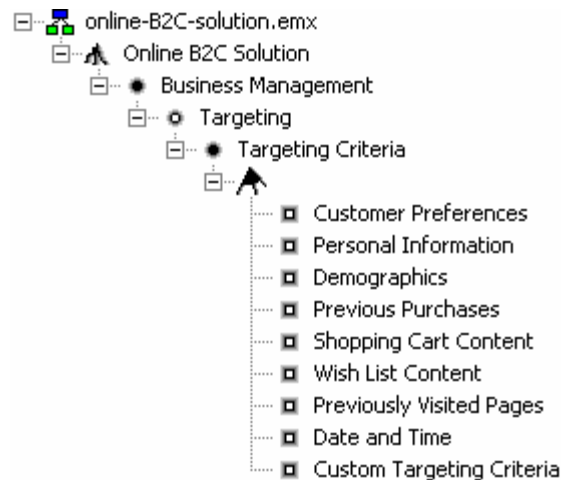


Figure 5.5: Targeting Criteria Feature

- *Previous Purchases.* Previous purchase data is analyzed for trends in consumer purchasing patterns. This information can be used to devise product recommendations in the form of "customers who bought this product also bought these products". Since previous purchase data is implicitly recorded in the orders, this information is always available unless it is explicitly deleted.
- *Shopping Cart Content.* Since placing an item in the cart signals the intent to purchase, an examination of the shopping cart contents can be used to understand consumer behaviour and relationships between products. For example, an item that is left in the cart after a session ends may indicate that the customer is unsure about the item. If this occurs frequently among many customers, it may indicate that the item is overpriced and may benefit from a discount. In another example, items that are placed together in a cart may indicate a relationship between the items. Like previous purchases, this information may be used to devise product recommendations. Since the shopping cart is a mandatory feature and this data can be implicitly recorded based on the cart actions, there are no additional constraints.
- *Wish List Content.* The motivation for examining the items in a wish list and the resulting analysis is similar to the information presented for the shopping cart content feature. The only difference is that adding an item to a wish list signals a weaker intent to purchase than adding an item to a shopping cart; therefore, a more substantial trend must exist before the same conclusions can be drawn. Support for this feature requires the selection of the wish list feature in the store front; however, the selection of the wish list feature has no implication on this feature.

- *Previously Visited Pages.* This feature involves an examination of the pages visited by a user. This information can be used to assess the popularity of certain products, the navigability of the site, the process of comparison shopping or product research, the effectiveness of external links, and other e-shop characteristics. Conclusions can be used to generate promotions or improve the site. In the latter case, the data may be used to restructure the site to enable faster or easier access to popular products. This feature requires the selection of the locally visited pages feature or the external referring pages feature¹⁸; however, the selection of the locally visited pages feature or the external referring pages feature has no implication on this feature.
- *Date and Time.* There are two interpretations for this feature. In the first interpretation, the targeting efforts are based on the date or time at which customers visit the site. For example, date-specific or time-specific promotions are offered to increase the number of visitors during periods with low site traffic. In the second interpretation, targeting efforts are based solely on the date or time. For example, promotions for snow blowers products are usually featured prominently during the winter months.
- *Custom Targeting Criteria.* The custom targeting criteria feature allows other targeting criteria to be defined after the e-shop is deployed at run-time. This is useful if the registration profile is extended with custom fields which contain marketing information. This feature requires the e-shop to be extensible so that analysis algorithms can be specified to make use of the new information.

5.2.2 Targeting Mechanisms

The targeting mechanism feature, shown in figure 5.6, provides methods for implementing a marketing effort. Once the target audience is selected, marketing staff must choose the appropriate methods for targeting the customers. The mechanisms can be divided into three categories: advertisements, discounts, and sell strategies. The inclusive-or feature group applies during all binding times because an effective marketing strategy typically involves a combination of mechanisms, such as advertising a discount on a product.

¹⁸ Locally visited pages and external referring pages are grouped features under the behaviour tracked feature in the store front. The features are described in section 4.7.1.

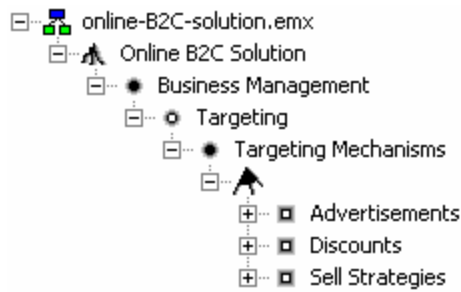


Figure 5.6: Targeting Mechanisms Feature

5.2.2.1 Advertisements

Advertisements, also referred to as ads, are promotions which are displayed to a customer while they are visiting the e-shop. Each advertisement is defined by two mandatory features: advertisement types and advertisement sources. The effectiveness of the advertisements can be improved by selecting the advertisement response tracking feature and / or the context sensitive ads feature. Figure 5.7 describes the details of the advertisement feature.

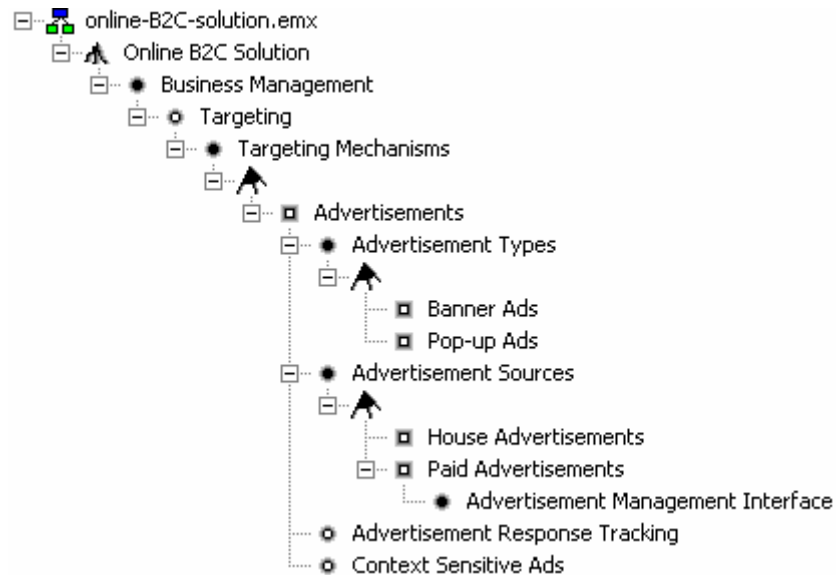


Figure 5.7: Advertisements Feature and Subfeatures

- *Advertisement Types*. There are two advertisement types which can be served on a web page: banner ads and pop-up ads. Banner ads are rectangular boxes that are embedded into the web page. These ads usually contain images, videos or text; they also link to whatever is featured in the ad. Banner ads are used primarily to promote other products, services or websites. Pop-

up ads open a separate window or overlay a window on top of the current page. The window typically contains a stand-alone web page. Pop-up ads can attract more attention because they can take the focus off the original page and redirect it to the ad. There is greater flexibility for the content in a pop-up ad because it resides in its own frame. The inclusive-or feature group applies to all binding times up to the definition of an ad during run-time since a web page can serve both advertisement types simultaneously; however, when an ad is defined, the feature group becomes exclusive-or because an ad must be one of the two types.

- *Advertisement Sources.* Advertisements have one of two purposes: 1) to promote products within the e-shop, or 2) to generate revenue by promoting things from external sites. This leads to two possible sources for advertisements: house advertisements and paid advertisements. The inclusive-or feature group applies to all binding times up to the definition of an ad during run-time since a web page can serve advertisements from both sources simultaneously; however, when an ad is defined, the feature group becomes exclusive-or because an ad is either a house advertisement or a paid advertisement.
 - *House Advertisements.* House advertisements are internal ads; they advertise in-store promotions or featured products. Banner ads which link to promotional pages are usually preferred because they are less intrusive and integrate better into the site; however, pop-up ads can also be effective for special promotions. For example, Amazon.ca uses a pop-up ad to advertise special discounts for new customers when it detects a new visitor.
 - *Paid Advertisements.* Paid advertisements are ads for a third party client. The differences between house advertisements and paid advertisements are the content is defined by the client, the links lead to external sites, and the client must pay each time the ad is served and / or clicked. To satisfy these differences, the Advertisement Management Interface (AMI) feature is required. The AMI provides interfaces for both the client and the e-shop staff. The client interface allows the client to upload ads and view the ads' serving statistics. The e-shop staff interface allows the staff to schedule ads and track the number of times an ad has been served. The AMI is general enough to handle both types of ads and includes facilities for banner management and website frame rentals.
- *Advertisement Response Tracking.* Advertisement response tracking allows the e-shop to assess the effectiveness of an ad by recording how many visitors see the ad and how many click on it. The closer the ratio between the number of views and the number of clicks is to 1:1, the more effective the advertisement.
- *Context Sensitive Ads.* Context sensitive ads are an example of reactive targeting. The ad is selected based on the information that is present on the page. An example of context sensitive

ads is "Ads by Google". "Ads by Google" uses banner ads which list sponsored links; the links are determined based on the keywords on the page and the Google search engine.

5.2.2.2 Discounts

The discounts feature provides mechanisms for defining and managing temporary price adjustments on products or orders. For example, it may be convenient to have a mechanism which allows the e-shop to set a temporary discounted price for a product, as opposed to manually changing the price field at the beginning and end of the discount period. Discounts are defined using the following mandatory subfeatures: discount conditions, award, eligibility requirements and graduation. In addition, there is an optional feature for accepting coupons and a mandatory feature for handling multiple discounts. The details of the targeting feature and all of its subfeatures are shown in figure 5.8.

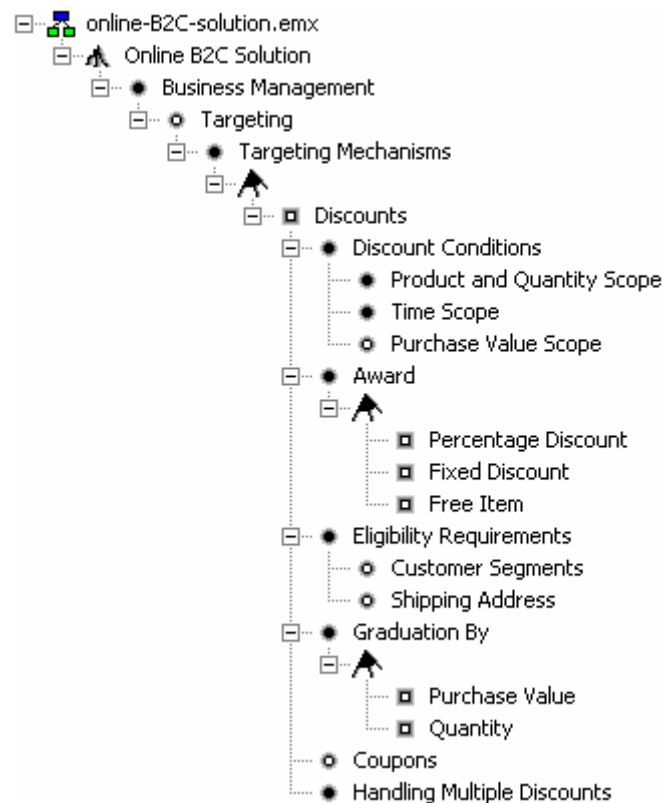


Figure 5.8: Discounts Feature

- *Discount Conditions.* Discount conditions define the conditions that must be satisfied in order to apply the discount. The simplest discount condition is when a customer receives a discount

for purchasing a specific product during the discount period. A more complex discount condition would depend on the order amount and certain products being purchased. A discount condition is defined by the product and quantity scope, time scope and, optionally, purchase value scope.

- *Product and Quantity Scope.* The product and quantity scope feature allows discounts to be assigned to a limited quantity of a given product. The quantity scope is used to limit the number of units that are available at the discounted price; the limit may apply to individual customers or the entire e-shop. An individual customer limit is useful for restricting the number of units which a customer can purchase in a single order or in total. An e-shop limit is particularly useful for special promotions. For example, “door crasher” specials require a limited number of units to be sold at a large discount. An e-shop limit is also useful for regular sales; the quantity scope can be set to the available inventory or no limit. In the former case, the discount applies to everyone who orders before the product goes out of stock; in the latter case, the discount applies to everyone who places an order. The equivalent concepts in brick and mortar stores are “while quantities last” and raincheques respectively. Raincheques allow customers to purchase an out-of-stock item at the discounted price once the item becomes available; they are unnecessary for an e-shop since orders can be held indefinitely in the order processor.
- *Time Scope.* The time scope feature allows a time period to be associated with the discount. This enables price adjustments to be made automatically.
- *Purchase Value Scope.* The purchase value scope feature allows a discount to be applied against an order. The discount can be applied against individual items or the entire order.
- *Award.* The award feature defines the discount amount. The discount amount can be specified as a percentage discount, a fixed discount or a discount in the form of a free item. The feature group relationship is inclusive-or for all binding times up until the definition of a discount rule during run-time. At that time, the feature group becomes exclusive-or because a discount rule can be specified with one discount amount only.
- *Eligibility Requirements.* Eligibility requirements allow a discount to be restricted to a subset of customers; by default, a discount is available to all customers. The optional subfeatures, customer segments and shipping address¹⁹, are pre-defined eligibility requirements. Customer segments restrict the discount to members of a target audience. Shipping address restricts the discount to customers residing in a geographical region; this is determined by the shipping address which is entered during the checkout. The shipping address feature requires the

¹⁹ This shipping address feature is distinct from the shipping address feature described in section 4.2.2.

selection of the shipping feature²⁰; however, the selection of the shipping feature has no implication on the shipping address feature.

- *Graduation By*. This feature allows discounts to vary depending on the total amount of the order or the number of items which were purchased. The discounts are graduated; for example, a discount involving the total amount could be expressed as "receive \$10 off a \$50 order, receive \$20 off a \$100 order, and receive \$50 off a \$250 order". The discount amounts are usually proportional to the discount factor.
- *Coupons*. Coupons are certificates issued by the e-shop which can be redeemed for a discount. In an e-shop, coupons are usually implemented as a code that the customer inputs during the checkout process. Coupon codes can be single use, meaning that the redemption code is unique and must be invalidated after it is redeemed, or general use, meaning that the redemption code can be used by any customer who obtains it.
- *Handle Multiple Discounts*. The handle multiple discounts feature defines a policy for dealing with multiple discounts on an order. This may result from multiple discount conditions being satisfied, multiple coupon codes being redeemed, or a combination of both. The e-shop may define restrictions on combining promotions. Another issue that arises from allowing multiple discounts is to ensure that the end result is consistent. For example, if an x% discount and a \$y discount are applied on an order, the amount of the discount will depend on the sequence in which the discounts are applied. Some strategies to deal with this are to apply coupons in whatever sequence the customer inputs them, apply the best or worst discount possible based on the discounts present, or define a global priority for all discounts to create a deterministic order of application.

5.2.2.3 Sell Strategies

The sell strategies feature, shown in figure 5.9, provides mechanisms for increasing sales by promoting related or better products to customers while they are shopping. The sell strategies, which include product kitting, up-selling and cross-selling, are reactive, context-sensitive product recommendations. The feature group is inclusive-or for all binding times up until the definition of a concrete sell strategy at run-time. At that time, the feature group becomes exclusive-or because a concrete sell strategy must be defined based on one of the three strategies.

²⁰ This refers to the subfeature of the physical fulfillment feature. It is described in section 5.1.1.2.

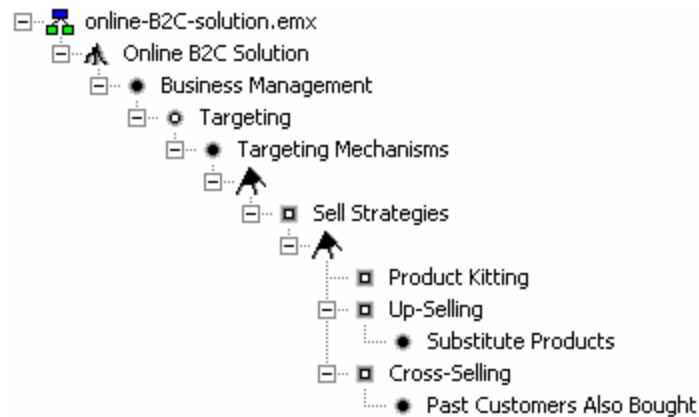


Figure 5.9: Sell Strategies Feature

- *Product Kitting.* Product kitting is a sell strategy where multiple related products are bundled together and the bundle is sold as a single unit. The products may be part of a series or have a related theme. Product kitting is a useful technique for highlighting related products and grouping products which are usually purchased together into a single item. Additional discounts may be offered on the bundle, such that the price of the bundle is less than the sum of the individual item prices. In that case, the bundle can be used to pair a more popular product with a less popular product to increase sales on the latter. Product kits are listed and displayed in a product page like regular products. An example of a product bundle would be a home theater package which includes a TV, a sound system and a DVD player; the package is sold as a single product as opposed to three separate products.
- *Up-Selling.* Up-selling is a sell strategy where customers are encouraged to consider products which are better quality than the one they are considering. This can assist customers in their product research because it allows them to explore related products, learn about additional product features, and compare the price and value of better products. Discounts may have the unintended effect of making the higher quality product cheaper than the lower quality product, which would defeat the purpose of the up-sell from the perspective of the e-shop; a policy may be required to deal with this scenario. Up-selling is implemented through the substitute products subfeature, which provides recommendations for better products. The recommendations are determined through product analysis or customer reviews. Substitutes can also be offered when the desired product is out of stock. The substitutes are often presented as a series of product links on the product page; it is meant to influence the customer's decision before the product is added to the cart.

- *Cross-Selling*. Cross-selling is a sell strategy where customers are encouraged to consider additional, related products. This can help customers discover other products which may interest them. Cross-selling is implemented through the past customers also bought subfeature, which provides recommendations for related products. The recommendations are determined through previous purchase data, browse data or recommendations from customers. The related products can be presented while the customer is browsing the product page, after the product is added to the cart, or after the order is completed through a follow-up e-mail; it is meant to generate additional sales either through the current order or a future order.

5.2.2.4 Display and Notification

The display and notification feature, shown in figure 5.10, describes the techniques which can be used to communicate marketing efforts to the customer. The three techniques are assignment to page types for display, product flagging and e-mails. For all binding times up until a concrete technique is selected at run-time, the feature group is inclusive-or since multiple techniques can be supported and used to communicate the promotion to the customer; however, when a concrete technique is selected, the feature group becomes exclusive-or since a concrete technique must be defined as one of the three techniques.

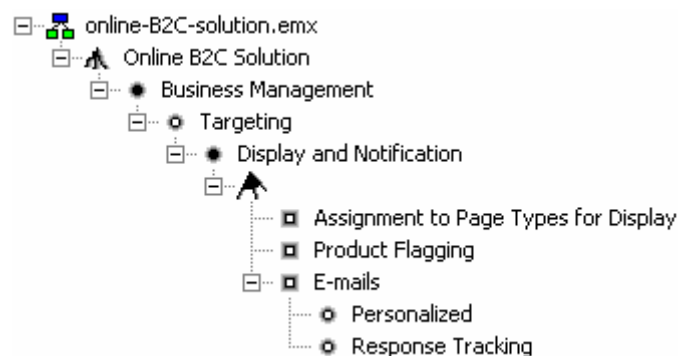


Figure 5.10: Display and Notification Feature

- *Assignment to Page Types for Display*. This feature allows the e-shop to associate a targeting mechanism with a page type for display purposes. A page type refers to a set of related web pages, such as checkout pages or product pages. For example, if cross-sells are associated with product pages, all product pages will include related products automatically. With this

feature, the e-shop staff can avoid mapping targeting mechanisms to individual pages manually.

- *Product Flagging.* This feature allows products that are on promotion to be highlighted in a product list and on the product page. The highlighting distinguishes the product from non-promotional products in the list and provides details about the discount.
- *E-mails.* This feature allows promotions to be communicated to customers outside of their shopping sessions. A promotion can be communicated through e-mail by including a list of links to discounted products, a link to the promotion page, a coupon code for the recipient, or a combination of the three. The use of e-mails may be enhanced through the selection of the personalized feature and / or the response tracking feature.
 - *Personalized.* This feature allows e-mails to be customized for registered customers. The customization can use information from the registration profile to generate an appropriate message and / or select the most relevant products to be included in the e-mail. Personalization may also entail generating a unique coupon code for the customer. This feature requires the registration feature since it uses information from the profile; however, the selection of the registration feature has no implication on this feature.
 - *Response Tracking.* This feature allows the e-shop to evaluate the effectiveness of the e-mails by determining what percentage of recipients click on a link in the e-mail. This is implemented by embedding links which can be tracked by the server. For example, a product link can be encoded with parameters to allow the server to record who clicked the link, when they clicked it and if they ended up purchasing the product.

5.2.2.5 Campaigns

The campaigns feature allows the e-shop to run and manage campaigns. A campaign implements a marketing effort by defining the target audience via the targeting criteria, selecting the appropriate targeting mechanisms to attract the customer, determining the sell strategies to help increase sales, and choosing the display and notification techniques to inform the customer of the promotion. Each element in the marketing effort is referred to as a campaign item. Campaign items can be combined to create a highly focused marketing effort. The effectiveness of a campaign can be assessed by measuring the promotion response rate and the conversion of responses into purchases.

5.3 Affiliates

Affiliates are business partners who collaborate with the e-shop by referring other customers or by providing links to the e-shop and its products on external sites. These collaborations are meant to increase traffic and sales through word-of-mouth or external advertisement. If the e-shop supports affiliate relationships, it must support affiliate registration and commission tracking also. The affiliates feature is shown in figure 5.11.



Figure 5.11: Affiliates Feature

- *Affiliate Registration.* Affiliate registration provides an interface for business partners to register. They must provide personal information for contact and compensation purposes, as well as information about their web site if they are providing links. The affiliate registration page also describes how to generate affiliate links, which are site or product links encoded with the affiliate's information. Many e-shops that offer affiliate relationships allow almost anyone to become an affiliate; in that case, the affiliate registration page must be publicly accessible.
- *Commission Tracking.* Commission tracking manages the compensation that is owed to the affiliate. If the affiliate provides links, the server keeps track of when an affiliate link is clicked so that the commission can be calculated and paid out. Commissions can be earned for referrals to the site or it may require that the referred visitors make purchases. If the affiliate refers customers, the customer must enter a referral code when making a purchase. The referral code is used to track the commission.

5.4 Inventory Tracking

Inventory tracking deals with tracking and controlling the inventory to ensure that there is an adequate supply to meet consumer demand. In the definition of this feature, it is assumed that only inventories of physical goods are being tracked. The inventory tracking feature, shown in figure 5.12, provides tool support that allows the e-shop staff to track the amount of inventory

sold, on-hand and on-order. The availability of an item can be determined using this information. If the inventory on-hand exceeds zero, the item will be available for ordering. Otherwise, it is considered to be out-of-stock. The amount of inventory can also be displayed on the product page as the number of items remaining, which may be useful for generating a sense of urgency on popular or promotional items. The information can also be used to determine if additional stock must be acquired from the supplier.

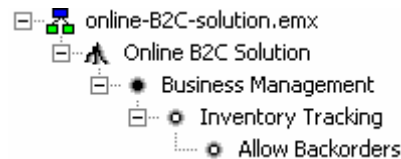


Figure 5.12: Inventory Tracking Feature

Although the inventory tracking feature can operate as a standalone feature, this limits its effectiveness. For example, the availability data may not be accurate since the data is only based on orders and sales; it does not reflect the actual state of the stock. Some of the stock may have been lost in transit, but this data would not be reflected until an operator updates it manually. In a practical scenario, the inventory feature requires the selection of the warehouse management feature and the fulfillment system feature²¹. When these features are selected, it enables the e-shop to monitor several characteristics of the inventory, including how much stock is remaining, the state of the stock, the location of the stock and if any additional stock has been ordered from the supplier. This provides up-to-date inventory data from the warehouse for customers who are browsing the e-shop.

- *Allow Backorders*. Backorders are orders which are placed against an item that has no available stock; they are filled as the stock is replenished. The strict interpretation of this feature is that a customer may order out-of-stock items regardless of the circumstances; however, it may be reasonable to impose a restriction where a backorder may be placed if and only if the number of items on order from the supplier is greater than the number of backorders already placed. There is also the possibility that a supplier may not be able to replenish the stock. In this situation, the e-shop must handle all of the outstanding backorders by notifying customers and, optionally, offering the customers a substitute product. If the

²¹ This is a subfeature of the external systems integration feature. Both features are described in section 5.7.

allow backorders feature is eliminated, customers will be unable to place an out-of-stock item into their shopping cart. In addition, changes to the availability of items that are left in a saved shopping cart must be synchronized.

5.5 Procurement

The procurement process deals with the acquisition of goods or services between businesses, such as the e-shop and its suppliers. The suppliers provide the inventory, in the form of goods for end-users or resources needed to perform services, for the e-shop to sell. The selection of the procurement feature, shown in figure 5.13, means that there is tool support, such as a B2B system, for handling the acquisition transactions. The stock replenishment feature allows the e-shop to replenish its inventory through the tools provided by the procurement feature. Stock replenishment can be performed manually or automatically, but the manual process is mandatory since it is required to maintain the continuity of business operations if the automatic process fails. The tool support implies that the procurement system feature²² is required; however, if the procurement system feature is selected, then the selection of this feature is also recommended.

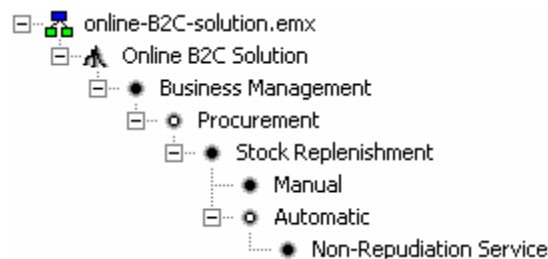


Figure 5.13: Procurement Feature

- *Manual.* The manual feature requires stock replenishment to be performed by human operators. The operators are responsible for monitoring the inventory levels, making the decision to replenish the inventory, and placing an order with the supplier. If a mistake is made, it may result in overstock in the warehouse or disruptions to sales due to a lack of inventory. The operator must also take into consideration how much additional inventory to procure, which can be influenced by the sales history of the item. The amount of tool support that the operator has in making these decisions depends on the selection of other features, such as inventory management, external systems integration, and reports and analysis.

²² This is a subfeature of the external systems integration feature. Both features are described in section 5.7.

- *Automatic.* The automatic feature allows stock replenishment to occur based on a set of predefined rules with no or minimal human intervention. Automatic stock replenishment requires the system to monitor several conditions, such as inventory stock and sales, and make the same decisions that operators make in the manual process. For example, the system may be programmed to submit an order for an item when the inventory remaining reaches a threshold value. Some manual interaction may be added to the process as a check and balance for the system's decisions. The automatic procurement process requires the selection of the inventory tracking feature in order for the system to acquire the necessary data for making the ordering decisions; however, the selection of the inventory tracking feature has no implication on this feature. The automatic process requires a non-repudiation service, which is a service that prevents either the e-shop or the supplier from denying the existence or content of an order transaction. This is especially important when order submissions are made by the system automatically because it maintains an irrefutable record of the transaction and holds both parties responsible for the order.

5.6 Reporting and Analysis

The reporting and analysis feature, shown in figure 5.14, provides tools which allow the e-shop to interpret collected data and format the interpreted data into a report. The information is used to assess the e-shop's performance and effectiveness. An e-shop can record a great deal of data about how customers interact with the site, such as the pages viewed, links clicked, and items purchased. Essentially, any action that requires a mouse click can be recorded for analysis. The data collection required by this feature is supported by the user behaviour tracking feature; however, the selection of the user behaviour tracking feature has no implication on the selection of this feature.



Figure 5.14: Reporting and Analysis Feature

Reports are defined by three characteristics: report types, report formats, and level of detail. The report type describes the focus of the report, such as daily operations, customer satisfaction or transactions. Each type represents a viewpoint for a subset of the underlying data and pertains to some operational aspect of the e-shop. The report format describes how the report is organized and how the data is presented. For example, data presentation can take the form of a graphical representation, which can be used to visualize trends or outlier data, or a tabular representation, which can be used for further data processing. The level of detail describes the amount of information that is presented in the report, which is dependent on the intended audience. For example, an executive report may contain key points and store-wide trends, while an engineering report may contain detailed descriptions and analyses for each incident that is found in the data. The combination of these three characteristics allows the reporting tools to create flexible, useful reports for different stakeholders.

The reports are used to make business decisions or to perform further analysis. For example, the reports can be used to identify ways of improving the business, such as focusing on customer service, or identify any problems with the site, such as navigability issues. In some cases, the direction of the analysis drives the data interpretation in the report; in other cases, the report summarizes the findings of the analysis.

5.7 External Systems Integration

External systems integration allows the e-shop to work with other systems, such as suppliers, warehouses, order processors and business management systems. These systems can be integrated on many levels, including the User Interface (UI), workflow and data levels. With UI integration, screens or UI widgets from external systems can be merged into the e-shop's interface such that it appears to be native to the e-shop interface. For example, a supplier may provide a button for the e-shop product management screen which allows an e-shop operator to replenish the inventory with a single click. Workflow integration allows the e-shop to define workflows which span multiple systems. The RefundOrder activity is a good example of workflow integration between the e-shop, the order processor and the warehouse. Data integration implies that data is shared between the systems. An example of data integration is the order database, which is used by both the order processor and the e-shop. There are four external systems which can be integrated with the e-shop: fulfillment system, inventory management system, procurement system, and external distributor system. The external systems integration feature is shown in figure 5.15. The feature

group is inclusive-or for all binding times since multiple external systems can be integrated simultaneously and multiple sources may be required to fulfill a particular request in the e-shop.

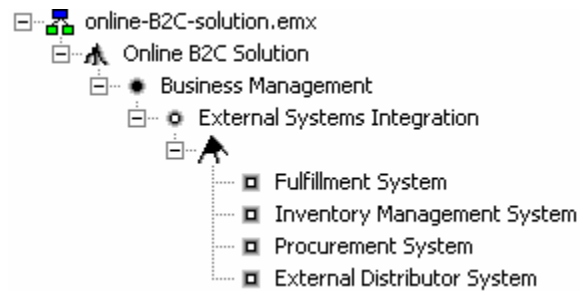


Figure 5.15: External Systems Integration Feature

- *Fulfillment System.* The fulfillment system refers to systems which are involved in order fulfillment, such as the order processor and the warehouse management system. Integrating the system allows order information, such as the order status, to be queried by the customer and displayed in the e-shop. It also makes up-to-date inventory information from the warehouse available to both the e-shop and its customers.
- *Inventory Management System.* The inventory management system refers any systems which are involved with the inventory replenishment and managing inventory availability. This is only required when the inventory tracking system is external to the e-shop. Integrating the system provides the same information which is available through the inventory tracking feature.
- *Procurement System.* The procurement system refers to supplier systems. Integrating the system provides information about the amount of inventory a supplier has and if the supplier intends to continue to carry a product. This information may be required for the backordering policy in the e-shop and can also be used by the e-shop for planning promotions, such as putting discontinued items on clearance.

External Distributor System. The external distributor system refers to the systems of parties that are involved with the distribution of the products to the customer, such as the shipping companies. Integrating their systems allows shipping information, such as the current package location, to be displayed through the store front.

5.8 Administration

The administration feature, shown in figure 5.16, provides tools for operating and maintaining the e-shop. These tools configure both offline and online options. Offline options are parameters which are read and applied during load time. The server must be restarted before it takes effect. Changes to static configuration files and some global options fall into this category. Online options are parameters which can be changed while the system is running and takes effect immediately. Options can be configured through a configuration file, a specialized program, or a secure web page on the site. The administration tools fall into two main categories: content management and store administration. Both categories are required to keep the e-shop running.

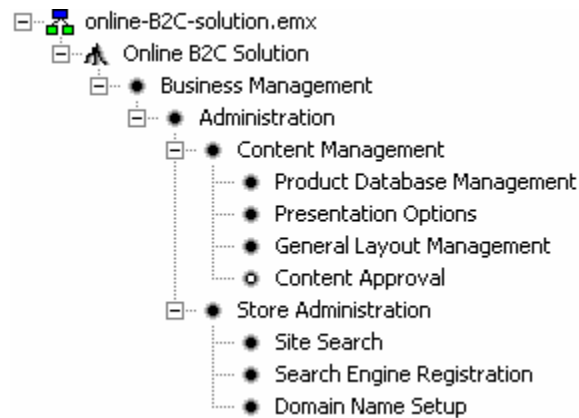


Figure 5.16: Administration Feature

5.8.1 Content Management

Content management deals with the information in the e-shop. The most crucial information is the product data. Content management tools perform two specific tasks: 1) manage product information, and 2) define the manner in which product information is presented. Managing product information is achieved through the product database management feature. This entails adding or updating product attributes, such as the product name, description, and any associated multimedia files. It may also include defining categories and assigning products to categories. Defining the presentation style is achieved through the presentation options and general layout management features. Presentation options affect the look and usability of the e-shop web pages, such as the default number of items to display in a product list page. General layout management specifies page layouts through the arrangement of elements, such as text boxes, banner ads, images and links, on a page. Page layout has a significant impact on the navigability and

aesthetics of the e-shop, which can affect its performance. The selection of the content approval feature adds a check-and-balance to the process. This feature prevents any changes from being published until someone other than the change author reviews and approves the change. This can help filter out mistakes which may cause the e-shop to lose money, such as having to honour pricing errors caused by typos, or customer inconvenience, such as having a customer return an item because the wrong product image was displayed.

5.8.2 Store Administration

Store administration deals with the e-shop's operational aspects, which are comprised of three features: site search, search engine registration and domain name setup.

- *Site Search.* The site search feature provides an interface for e-shop staff to perform queries on the site itself. It can search both the content and the meta-aspects of the site. For example, a query can be executed to find all products which do not have a product image. The purpose of the site search is to facilitate e-shop maintenance by allowing queries to turn up anomalous formatting or content on the site. The main difference between this site search feature and the search functionality in the store front is the ability to use meta-aspects as query parameters; the customer is only interested in searching through the product data. The two features are completely independent.
- *Search Engine Registration.* The search engine registration feature allows the e-shop site to interface with search engines. The feature's primary function allows the e-shop to register its top level domain name with different search engines and, if possible, specify keywords to be used in the indexing process. Registration makes the search engine crawlers aware of the e-shop's site. In addition, this feature can also make site pages accessible or inaccessible to the search engine. First, this feature allows the e-shop to specify which pages should not be indexed, such as checkout pages. Secondly, link rewriting can be performed. Link rewriting is the process of converting complex URLs, such as URLs with several parameters, into a simpler URL which is recognizable to the search engine crawler. This is useful since many crawlers impose a limit on the number of parameters; if the limit is exceeded, the crawler ignores the page. Finally, this feature can also store static versions of dynamic e-shop pages. This is necessary to allow search engines crawlers to reach these pages.
- *Domain Name Setup.* The domain name setup feature configures the e-shop so that it is accessible through a friendly URL, such as the top level domain name. This type of option

can be set through the web server, such as with Apache's redirect command, or by using a proxy server to redirect the traffic. There are two applications for server redirection: 1) sending the request to the least busy server in the server pool for load balancing, and 2) sending the request to an appropriate handling server. An example of the latter is to direct a request to the regional server, depending on the origin of the request. This feature allows the configuration to be performed manually or through a wizard interface.

5.9 Summary

In this chapter, the domain analysis for the business management features in the e-shop was presented. In contrast to the store front, the business management features deal with back-office operations. The key features in business management include: order management, which enables the fulfillment of orders; targeting, which enables the e-shop to launch marketing efforts; inventory tracking, which determines how item availabilities are dealt with; external systems integration, which determine how different services can be leveraged by the e-shop; and administration, which provides tools which manage the e-shop.

Chapter 6 Model Template Descriptions

This chapter describes the set of feature-based model templates used to specify the e-shop. The primary focus of the model templates is to represent the store front and a full workflow that allows a customer to visit the e-shop and make a purchase. The chapter is divided into two main sections: the first describes the class diagram model templates and the second describes the activity diagram model templates. The design of each model template and highlights of their annotations are presented.

6.1 Class Diagram Model Templates

There are two class diagram model templates; the first deals with the store front entity model and the second deals with the service model. In the class diagrams, three stereotypes, which are not annotations denoting PCs, are used: id, entity and service. The id stereotype denotes an attribute with a unique value which is used to identify instances of the class. The entity stereotype denotes a class whose instances require persistence. The service stereotype denotes an active component.

6.1.1 Store Front Entity Model

The store front entity depicts the entities in the store front. The model is composed of regular class diagram model elements, such as classes, attributes, associations, composite aggregations and enumerations. There are two major parts to the model. The first part, shown in figures 6.1 and 6.2, describes information about the products offered by the e-shop. The second part, shown in figures 6.3, 6.4 and 6.5, describes information about the customer and entities used to facilitate their shopping experience. The store front entity model is fairly large, so the description will only cover key entities with notable annotations; elements which are annotated with features of the same name will not be covered in much detail.

The first part is centered on the EShopArtifact class. EShopArtifact is an abstract class used to describe any unit of information relating to the structure of products in the e-shop. The class contains common attributes needed for rendering and storing the information, such as an id field and a description. EShopArtifact is subclassed by the Catalog, Category, Product and Asset classes.

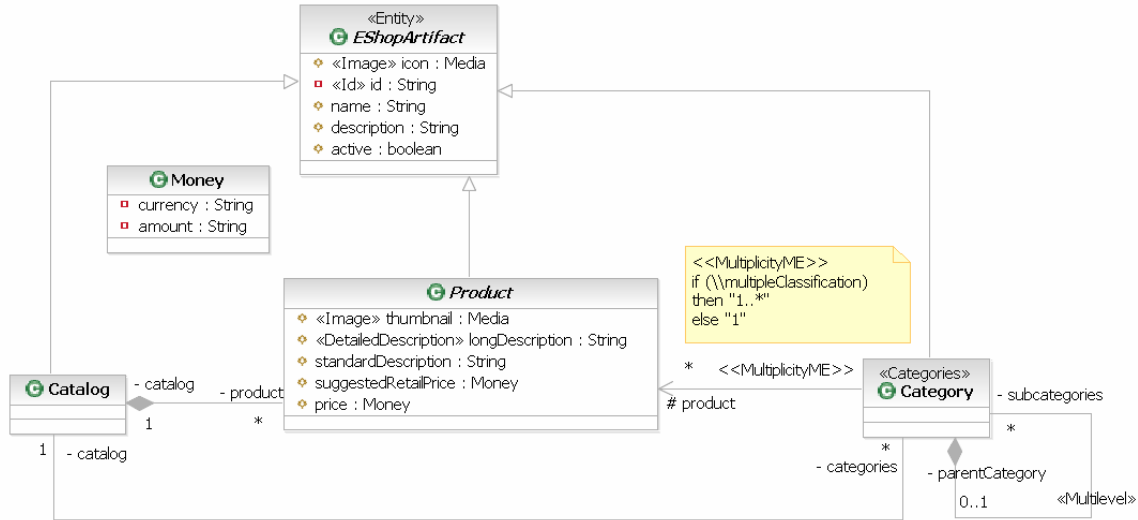


Figure 6.1: EShopArtifact, Catalog, Category, Product and Money Classes

The key features of the classes are as follows:

- The Catalog class, shown in figure 6.1, contains a set of products; every e-shop contains at least one catalog. There are no annotations since the catalog feature is mandatory.
- The Category class, shown in figure 6.1, can be used to organize products. Instead of containing products, a category is associated with products so that it can be deleted without deleting all of the products associated with it. This also simplifies the representation of the multiple classification feature because multiple classification can be modeled as a multiplicity. This would not be possible if containment was used because a product cannot be contained by multiple categories. The MultiplicityME changes the multiplicity on the category side to “1..*” based on the presence of the multiple classifications feature to accommodate multiple associations between a product and a category. The composite aggregation which is used to represent the multilevel feature is also of interest. Containment is used since the multilevel feature allows categories to be nested. It is important to note that if the categories feature is eliminated, the association and composite aggregation, which are annotated with subfeatures of categories, will also be removed.
- The Product class, shown in figure 6.1 and 6.2²³, contains several attributes. One of particular interest is the thumbnail attribute because it demonstrates the simplicity of expressing variability by focusing on the concept instead of the structure. The attribute is annotated with

²³ The associations and generalizations between the Product class and other classes have been divided into two diagrams (figures 6.1 and 6.2) for clarity.

the image feature. No additional annotations are required to depict the selection of the image's superfeatures to include this attribute. For example, if the associated assets feature is eliminated, the thumbnail attribute will be automatically removed because the image feature is a subfeature of the associated assets feature.

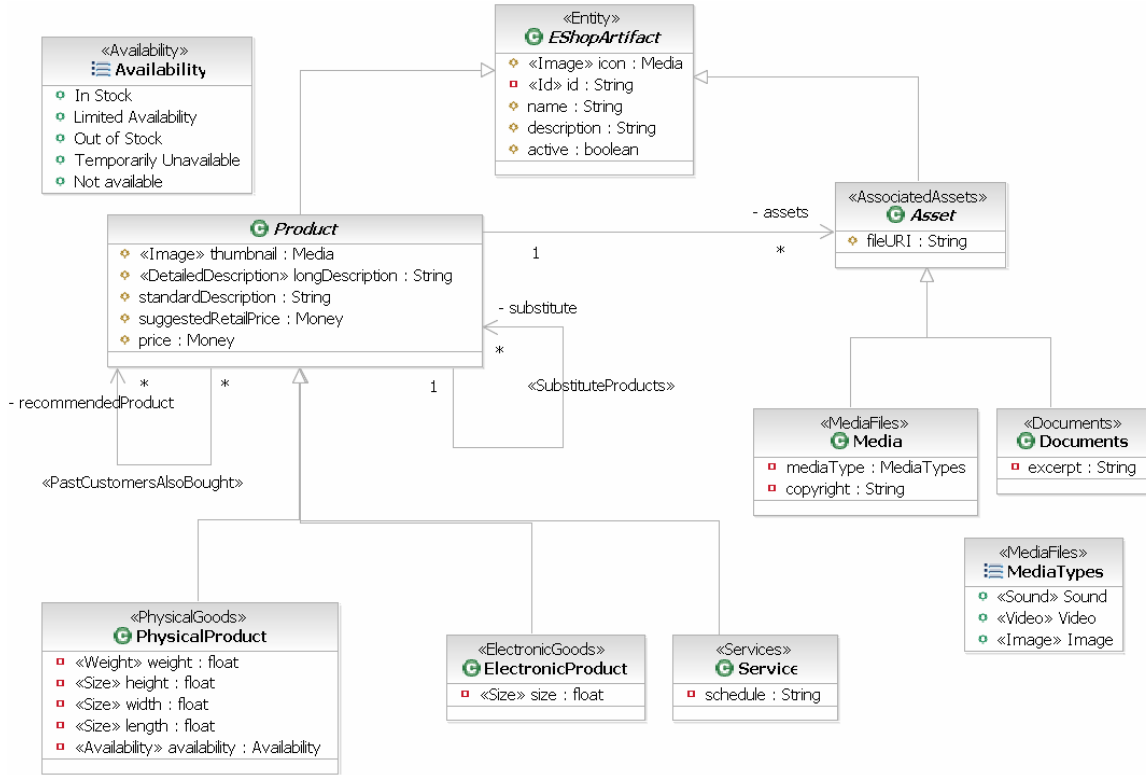


Figure 6.2: EShopArtifact, Product, and Asset Classes

The Asset class, shown in figure 6.2, is an abstract class used to represent files that are associated with the product. The most interesting annotation is in the media types enumeration. The respective enumeration values are annotated with the image, sound and video features. In addition, the enumeration itself is annotated with the media files feature. The feature model depicts image, sound and video as grouped features of an inclusive-or feature group under the media files feature; therefore, the selection of media files implies that one of the grouped features will be selected. Although the media files annotation seems redundant, its purpose is to ensure that the empty enumeration is removed if the media files feature is eliminated.

The second part deals with several entities; the simplest way to describe it is to focus on five classes: Customer, ShoppingCart, Wishlist, Order and TaxRule.

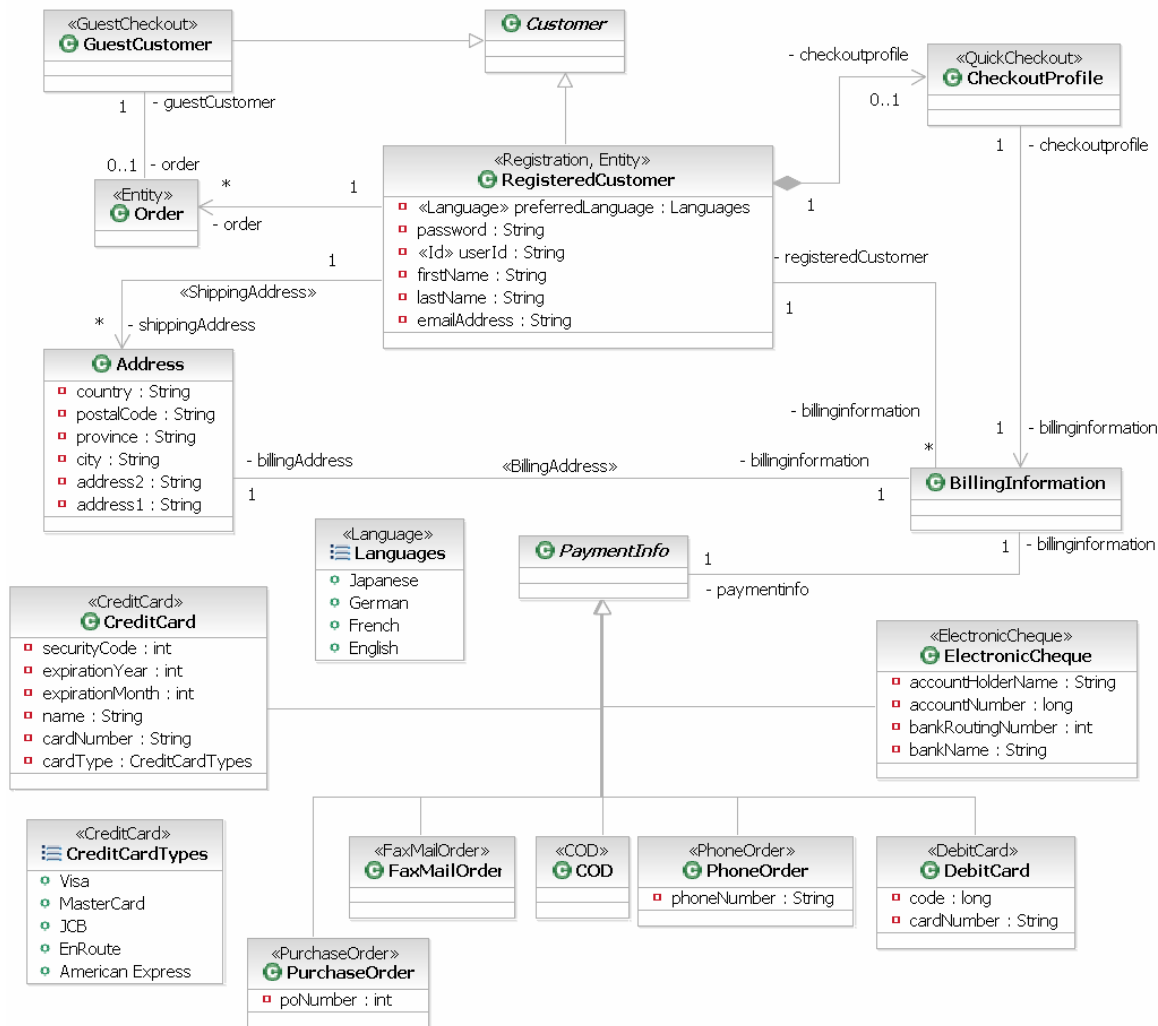


Figure 6.3: Customer and BillingInformation Classes²⁴

The Customer class is an abstract class which relates the customer types, as shown in figure 6.3, and associates the customer with a shopping cart and a wish list, as shown in figure 6.4. It contains no common attributes because the GuestCustomer class does not contain any identifiable information. The RegisteredCustomer class defines a registration profile. It can contain information about the customer, their preferences, and their billing information. Information about the customer and their preferences is represented by attributes in the class. Figure 6.3 shows that the billing information is represented by an association to one or more BillingInformation classes, which encapsulates the payment information and billing addresses. The registration profile also contains a CheckoutProfile class, which references a set of billing

²⁴ The attributes for the Order class are not shown in this diagram; full details are given in figure 6.4.

information and shipping instructions stored in the profile. The class is used by the quick checkout feature. Finally, customers can be associated with orders, as shown in figure 6.4. The association between GuestCustomer and Order is meant to allow a guest to place an order, whereas the association between RegisteredCustomer and Order is meant to allow orders to be retrieved by customers. All of the required information for an order is stored in the Order class, which is depicted in figure 6.4.

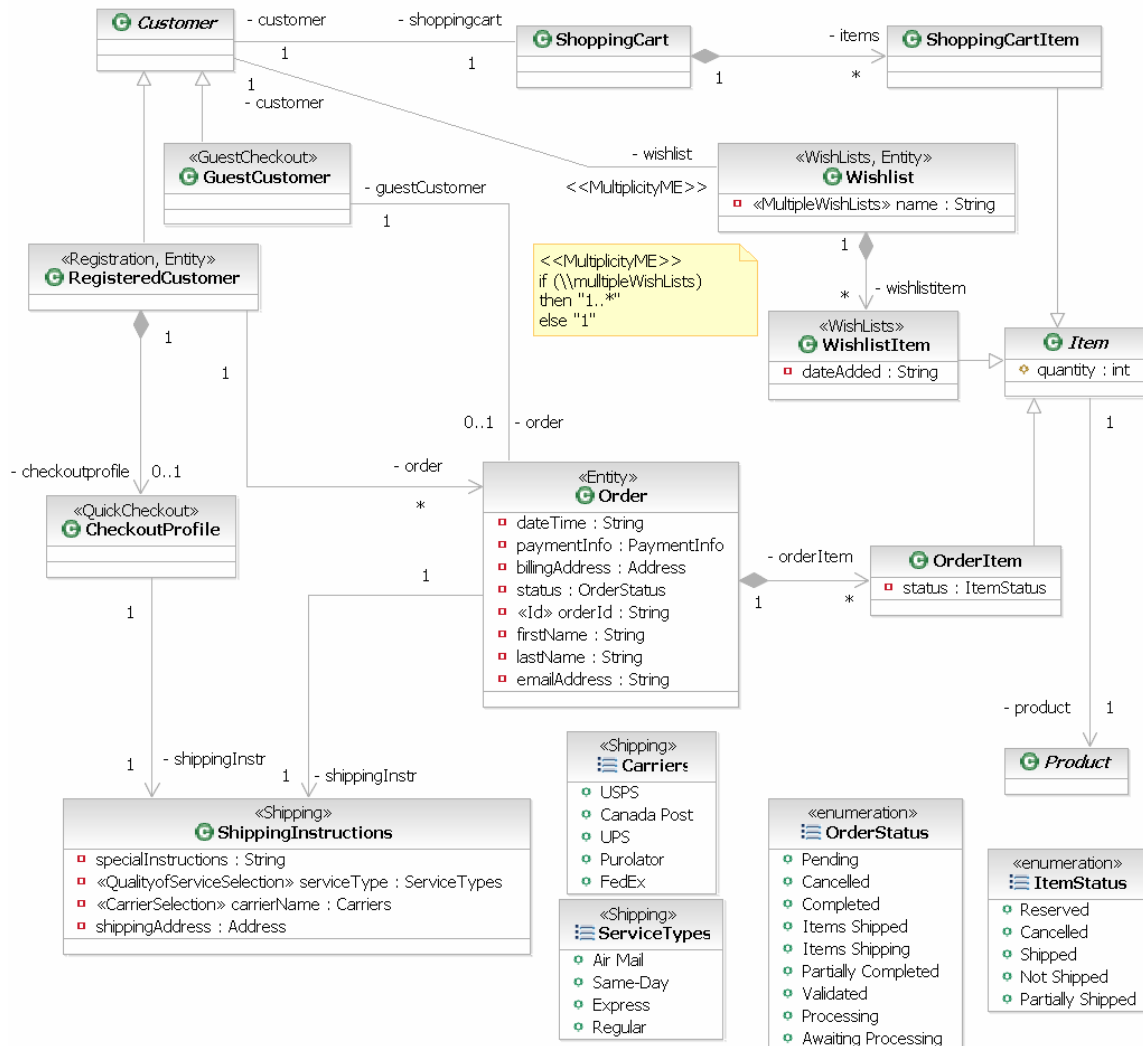


Figure 6.4: Order, ShoppingCart and Wishlist Classes²⁵

The ShoppingCart class, shown in figure 6.4, represents the list of items that a customer intends to purchase. The cart contains instances of the ShoppingCartItem class, which references a

²⁵ The attributes for the Product and RegisteredCustomer classes are not shown in this diagram.

product and associates it with the quantity to be purchased. To reduce diagram clutter, the ShoppingCart class is not annotated because the shopping cart is a mandatory feature; however, it is possible that one may wish to annotate the class for traceability reasons.

Wish lists and shopping carts share a similar structure; however, there are two main differences: 1) a single customer may be associated with multiple wish lists if the corresponding feature is selected, and 2) wish list items also keep track of the date the item was added. It is interesting to note that the WishList class is annotated with a feature, wish lists, and an attribute is annotated with a subfeature, multiple wish lists. This is a simple way of incrementally adding features into a class.

The Order class, shown in figure 6.4, contains all of the attributes needed to store order information. Some of the attributes, such as firstName and lastName, seem redundant; however, the duplication of information is required to store information for guest purchases. Furthermore, it keeps the order information consistent if changes are made to the profile information after the order is submitted. The order contains instances of the OrderItem class, which tracks the product, the quantity ordered, and the item status. The order is also associated with the ShippingInstructions class, which provides details on what options a customer has control over for their shipment. It is interesting to note that the annotations on the ShippingInstructions class and its attributes represent a partial view of the variability in the entity.

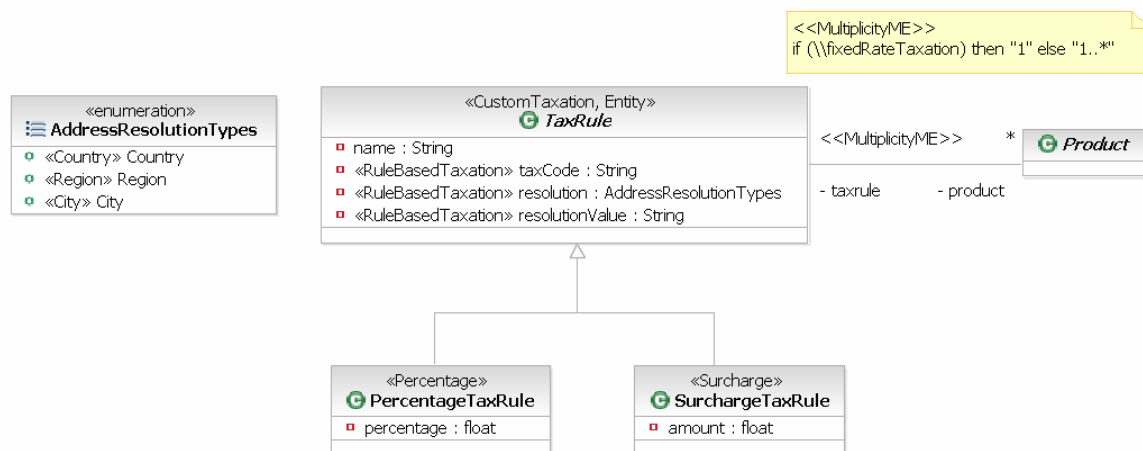


Figure 6.5: Tax Rule Classes²⁶

²⁶ The attributes for the Product class are not shown in this diagram.

The TaxRule class, shown in figure 6.5, defines the tax calculation rules for each product. Every product can be associated with one or more tax rules. Annotations are used in a similar manner as in previous classes.

6.1.2 Services Model

The services model depicts the actors in the e-commerce solution as individual, high-level services. Modeling the system in this fashion allows for greater modularity and abstraction of underlying details. A service can be implemented as a wrapper to a local implementation or an invocation to a web service provided by a third-party gateway. The service model is shown in figures 6.6 and 6.7.



Figure 6.6: E-Shop Interactions with Warehouse, Payment Gateway and Order Processor

The model is centered on the EShop component which represents the store front. The EShop component depends on a set of services which are accessed through a set of interfaces. The services are provided by the warehouse, tax gateway, tax calculator, shipping gateway, in-house

shipping, payment gateway and order processor. The interactions with the payment, warehouse and order processor services are shown in figure 6.6. The interfaces for some services are divided into front-end and back-end operations, which are accessed by the e-shop component and the order processor services respectively.

Figure 6.7 depicts the tax and shipping services. Each interface is supported by two different services. For example, the TaxGateway and TaxCalculator services correspond to the tax gateways and custom taxation grouped features respectively.

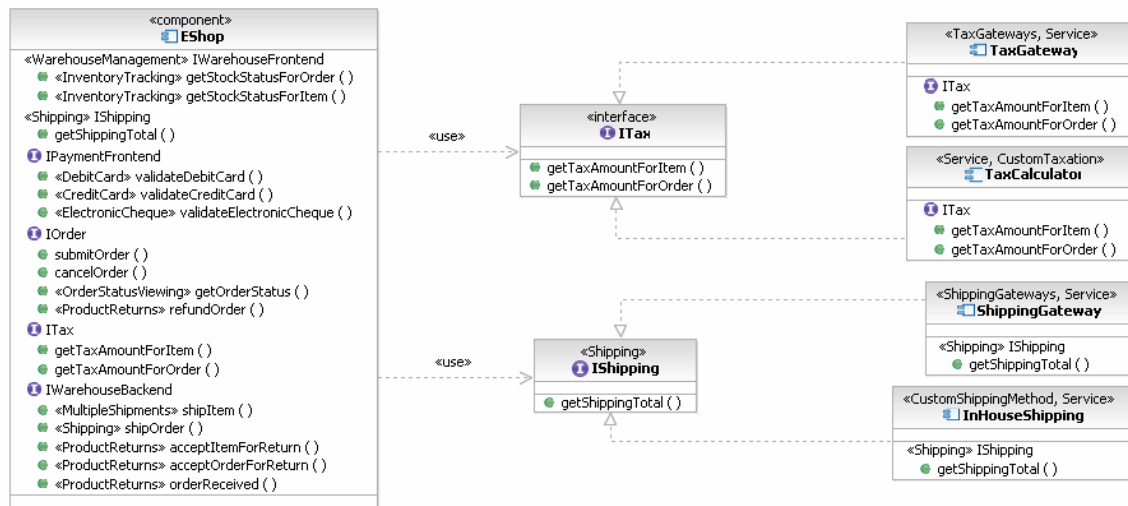


Figure 6.7: E-Shop Interactions with Tax and Shipping Services

6.2 Activity Diagram Model Templates

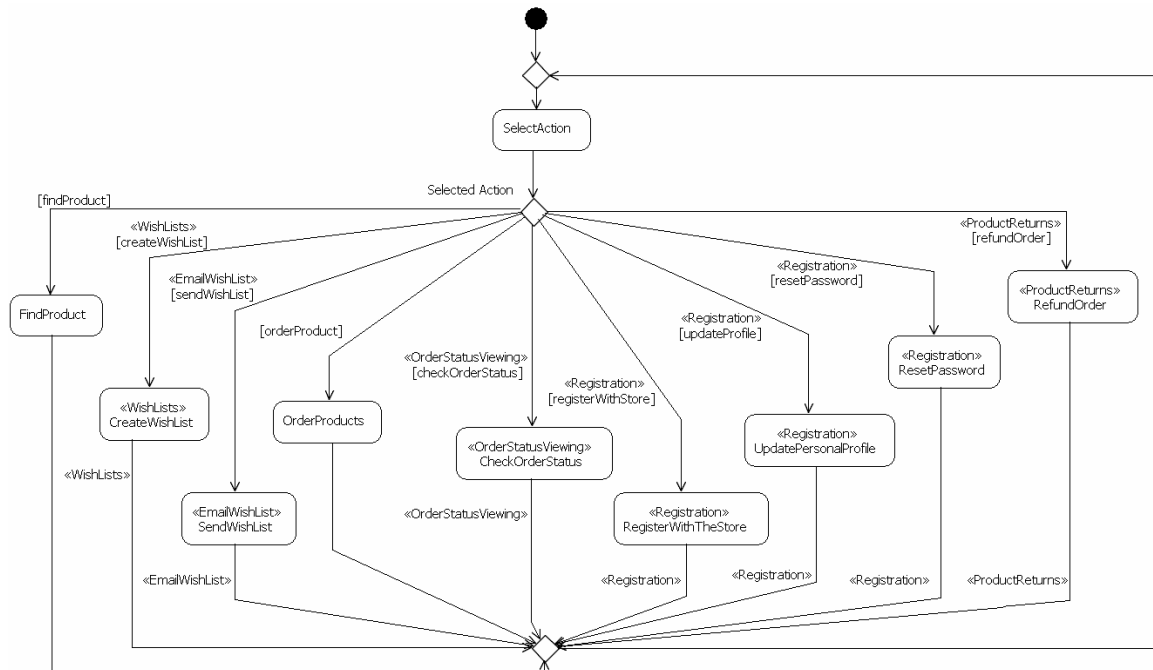
There are seventeen activity diagram model templates which deal primarily with workflows in the store front. The activities can be divided into six categories, as shown in table 6.1. The model templates are presented in the order shown in the table.

Table 6.1: Activity Categories

Category	Related Activities
Top Level	StoreFront
Shopping	FindProduct, SearchProduct, SelectProductFromCatalog, SelectProductFromWishlist
Customer Purchases	CheckoutItems
Order Related	CreateOrder, OrderProducts, ProcessOrder, CheckOrderStatus, RefundOrder
Customer Administration	RegisterWithTheStore, UpdatePersonalProfile, CreateQuickCheckoutProfile, ResetPassword
Wish List	CreateWishList, SendWishList

6.2.1 StoreFront Activity

The StoreFront activity, shown in figure 6.8, is a top-level representation of the customer's choices while the customer visits the e-shop. Every choice leads to a call behaviour action, which invokes the target activity. There are no implicit annotations since this is a top-level activity. Unless the call behaviour represents the semantics of a mandatory feature, each call behaviour is explicitly annotated with the feature which enables it.

**Figure 6.8: StoreFront Activity**

6.2.2 FindProduct Activity

The FindProduct activity, shown in figure 6.9, deals with the selection of a product from one of several sources and returns the product. Products can be selected from a promotional e-mail, the e-shop catalog, the product search facility or a customer's wish list. The selection process is represented by the respective call behaviour or action after the "Product Source" decision node. In the first three sources, the selected products can be further configured using the set of actions following the merge node. In the wish list source, the customer is selecting a configured product since a product must be configured before it can be added to a wish list or shopping cart. The FindProduct activity is used as an intermediate step.

This model template describes a great deal of variability. There are no implicit annotations on any of the elements since the presence of the FindProduct activity is independent of any feature. The Select* call behaviours and actions are all annotated with their respective features, except for the catalog since it is a mandatory. Although the catalog call behaviour could be annotated for traceability, the annotation would make it more difficult to distinguish the commonality in the product line. The central buffer following the merge node after the three Select* elements is annotated with a TypeME; the ME is based on the product variants feature. If the feature is selected, then the element type should be an unconfigured product, which is denoted as "General Product"; otherwise, it is denoted as a regular "product". This is particularly important due to the path that follows the buffer. The path deals with the configuration of a product, which is required when there are product variants; therefore, the three actions, two central buffers and four flows in the path are all annotated with the feature. If the feature is eliminated, the entire path is removed and flow closure is performed, meaning that the customer will not have to perform any product configuration when selecting a product through an e-mail, the catalog or the search function.

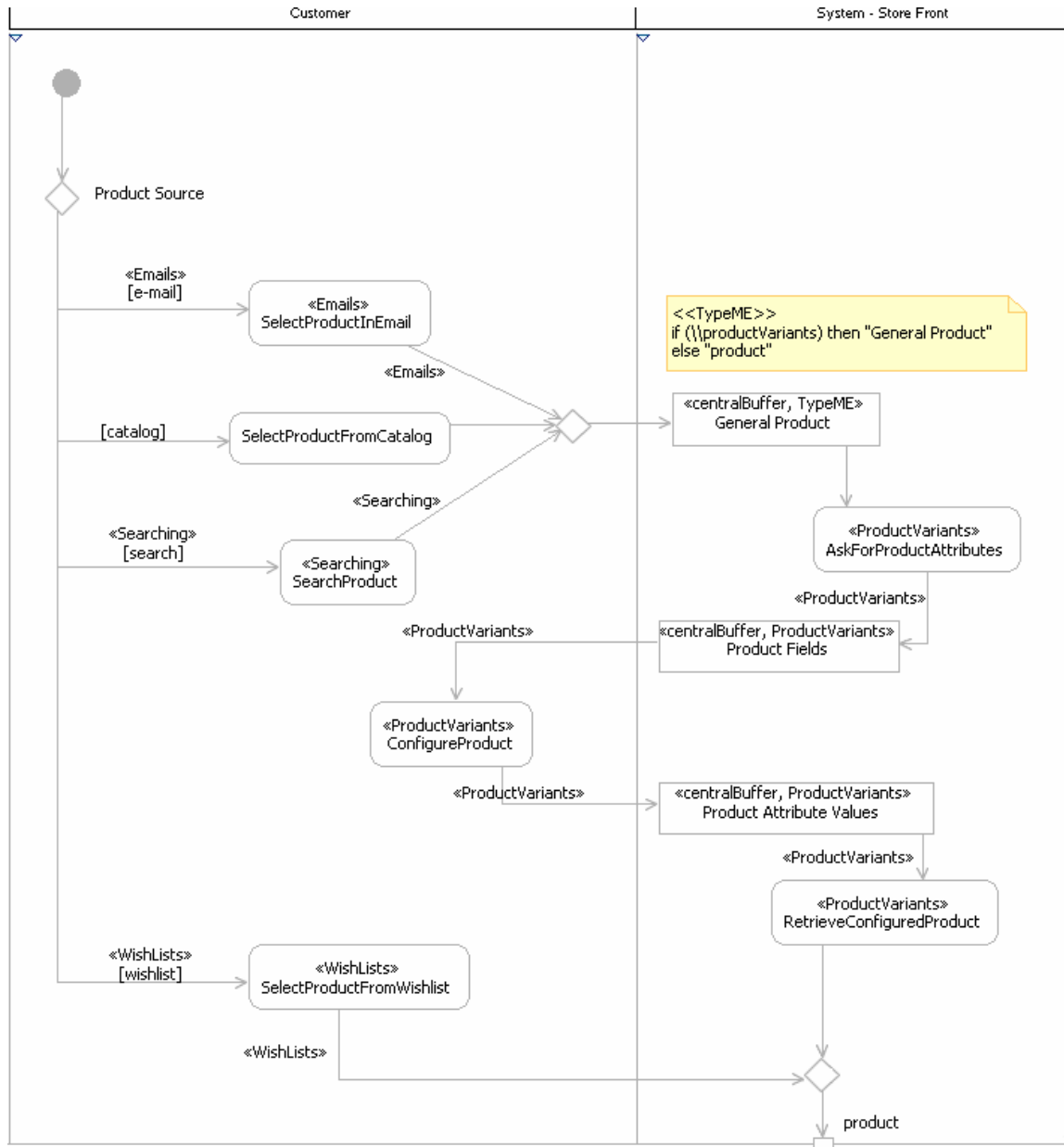


Figure 6.9: FindProduct Activity

6.2.3 SearchProduct Activity

The SearchProduct activity, shown in figure 6.10, describes the workflow for invoking the search functionality. The process begins with the store front soliciting the search parameters from the customer. For a basic search, the search interface typically consists of a text box with a search button and is accessible from almost any page in the e-shop. For an advanced search, the additional parameters require an advanced search form to be displayed. Once the customer enters

the search parameters and invokes the search, the store front executes the query on the products database and displays the results in the form of a product list. The customer selects a product from the product list and the product is returned by the activity through the activity parameter. The SearchProduct activity is used by the FindProduct activity as an intermediate step to retrieve a product.

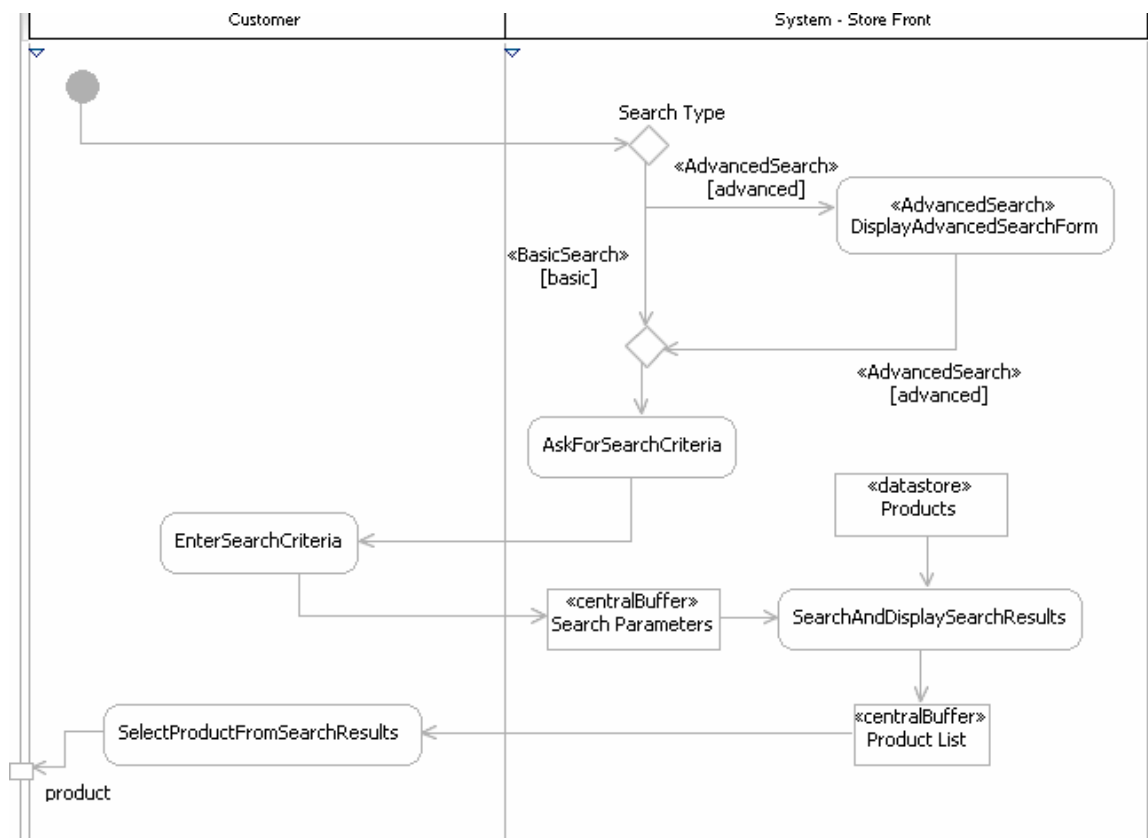


Figure 6.10: SearchProduct Activity

All elements are implicitly annotated with a conjunction with the searching feature, due to the annotation of the searching feature on the activity²⁷. There are two sets of annotations in this model template. The first set pertains to the advanced search functionality. It consists of the "advanced" branch following the "Search Type" decision node, which includes the DisplayAdvancedSearchForm action and the adjacent flows. The second set pertains to the basic search functionality. It consists of the "basic" branch of the "Search Type" decision node, which is a single control flow between the decision and merge node. The annotations are set up to allow

²⁷ Annotations on activities are not visible through the figures.

either one or both branches to be present. In the event that only one of the two features is selected, flow closure and simplification will reduce the region into a sequential path.

6.2.4 SelectProductFromCatalog Activity

The SelectProductFromCatalog activity, shown in figure 6.11, describes the workflow for browsing the catalog and selecting a product from it. The process begins when the store front retrieves the catalog and passes it to the data extraction branch. The data extraction branch includes all of the elements starting from the outgoing flow of the first merge node and ending with the DisplayInformation action. The branch is responsible for extracting categories and products. The GetCategoriesAtContainerLevel and GetProductsAtContainerLevel actions are generic routines which extract information from the container that is passed to them. A container can be a catalog, category or sub-category; this design allows a container to contain products as well as other categories. The two actions generate an element list which is rendered on the page; the rendering is dependent on which element types are present. The elements in the element list correspond to the EShopArtifact class in the entity model. When control is returned to the customer, they select an element. The store front checks the type of the selection. If the selection is a product, it is returned and the activity ends; otherwise, the selection is a container and the store front checks if there are any more subcategories to determine which part of the data extraction branch needs to be run. The SelectProductFromCatalog activity is used by the FindProduct activity as an intermediate step to retrieve a product.

There are two sets of PCs and one TypeME used in this model template. The first set of PCs references the categories feature and is used to annotate a part of the data extraction branch and a part of the decision process after the customer selects the element. The second set of PCs references the multi-level feature and is used to annotate the part of the decision process that checks for subcategories. It is important to note that the “no” branch of the “More Subcategories” decision node is annotated with the categories feature instead of the multi-level feature to ensure that flow closure will preserve that branch even if the multi-level feature is eliminated. The TypeME changes the type of the central buffer following the GetProductsAtContainerLevel action so that the DisplayInformation action can render the page correctly. There are no implicit annotations on the model template elements.

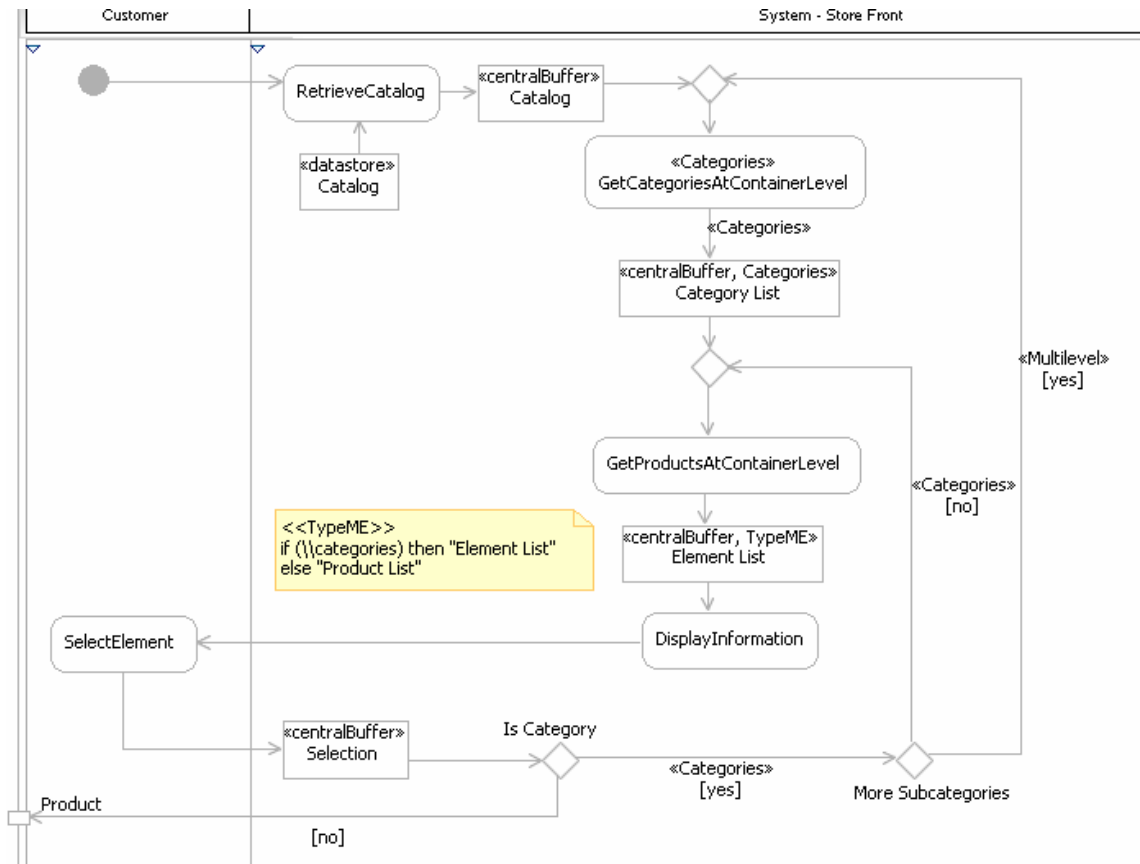


Figure 6.11: SelectProductFromCatalog Activity

6.2.5 SelectProductFromWishlist Activity

The **SelectProductFromWishlist** activity, shown in figure 6.12, describes the workflow for selecting a product from a wish list with the intent of adding the product to the shopping cart or browsing its product page. The process begins with the customer making a request for the wish list; if the customer has multiple wish lists, they must also specify the wish list identifier. Depending on the customer type, the wish list is retrieved from the server or client side. The wish list is returned in the form of a product list. The customer selects a product from the list and the product is returned through the activity parameter. The **SelectProductFromWishlist** activity is used by the **FindProduct** activity as an intermediate step to retrieve a product.

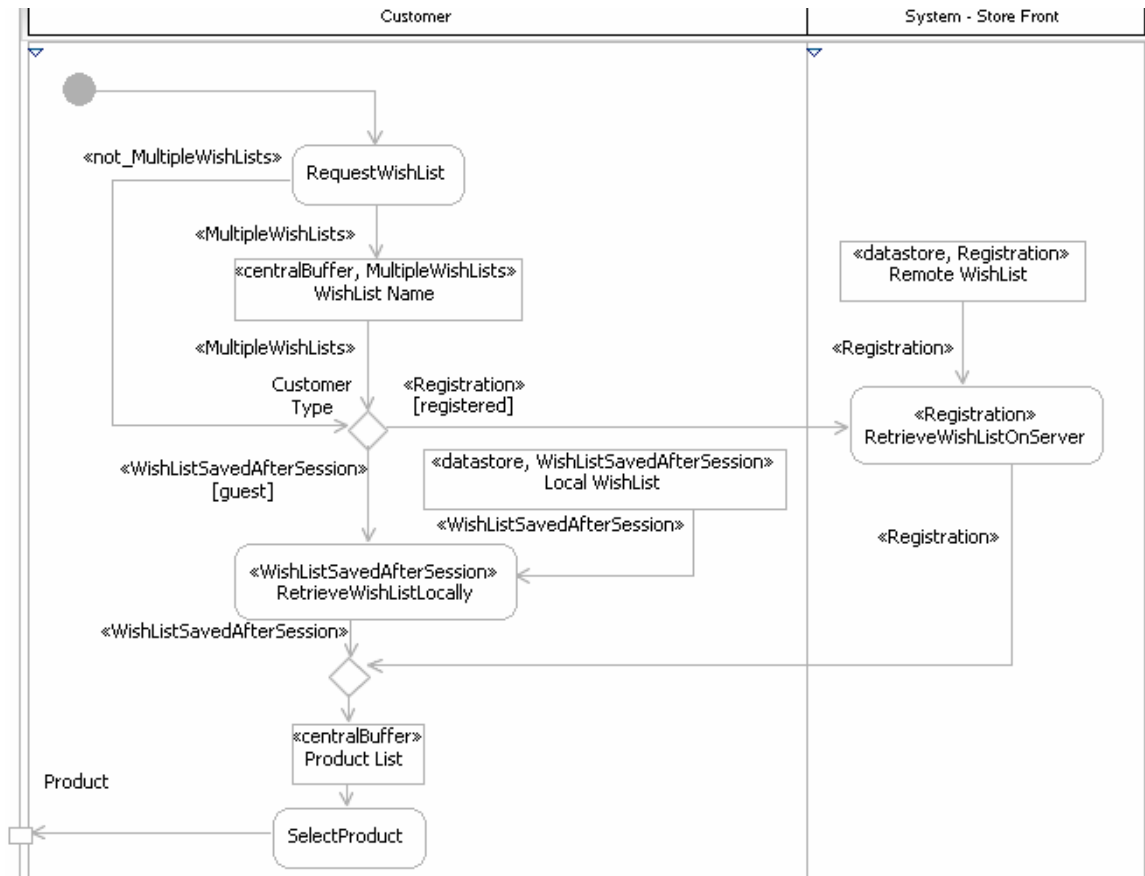


Figure 6.12: SelectProductFromWishlist Activity

All elements are implicitly annotated with a conjunction with the wish list feature, due to the annotation of the wish list feature on the activity. In addition, there are three sets of annotations in this model template. The first set consists of annotations on the first central buffer and the adjacent flows, which are annotated with the multiple wish lists feature. The information is required only when the system needs a parameter to determine which wish list to retrieve. There is an additional control flow which is present if the multiple wish lists feature is eliminated. This additional flow is necessary because it introduces the correct flow type to connect the actions when the central buffer is removed.

The second set of annotations reference the registration feature. All of the elements in the “registered” branch of the “Customer Type” decision node, as well as the data store used by the RetrieveWishListOnServer action, are annotated. This branch is removed if the e-shop does not support registered customers.

The third set of annotations involves the wish list saved after session feature, which allows guests to maintain wish lists. All of the elements in the “guest” branch of the “Customer Type” decision node, as well as the data store used by the RetrieveWishListLocally action, are annotated. This branch is removed if the e-shop does not allow wish lists to be persisted for guest customers.

6.2.6 CheckoutItems Activity

The CheckoutItems activity describes the checkout workflow, which includes the process of collecting information from the customer, performing calculations on the order and cleaning up following the submission of the order. The entire process can be dividing into three phases, which are described as follows.

The first phase, shown in figure 6.13, deals with selecting the checkout type and, if necessary, entering the shipping information. It consists of all the elements from the initial activity node to the merge node which precedes the HandleItemAvailability action. The process begins with determining the customer type and is followed by selecting the checkout type. Registered customers may have the option of selecting between a quick checkout and a registered checkout, while guests can only make purchases through the guest checkout. A quick checkout involves retrieving the quick checkout profile and bypassing the shipping information entry, which leads to the end of the phase. The registered and guest checkouts²⁸ are functionality identical and require the customer to select their shipping destination and options. Once that is done, the phase is complete.

The second phase, also shown in figure 6.13, is a linear sequence of actions which handles calculations. It contains all of the elements from the HandleItemAvailability action to the outgoing flow of the DisplayOrderSummary action. The HandleItemAvailability action reads in the shopping cart contents and retrieves information about the availability of the items. This gives the customer up-to-date information about when they can expect their order to arrive. The ApplyShippingCosts, ApplyDiscounts, and CalculateTaxes actions perform shipping cost, discount and tax calculations which are applied against the order total respectively. These actions are implementation independent; the calculations can be performed in-house through a local function call or by a third party gateway through a web service invocation. The phase ends with DisplayOrderSummary, which displays the order summary with the final order total on the page.

²⁸ These two checkout types are also referred to as a regular checkout.

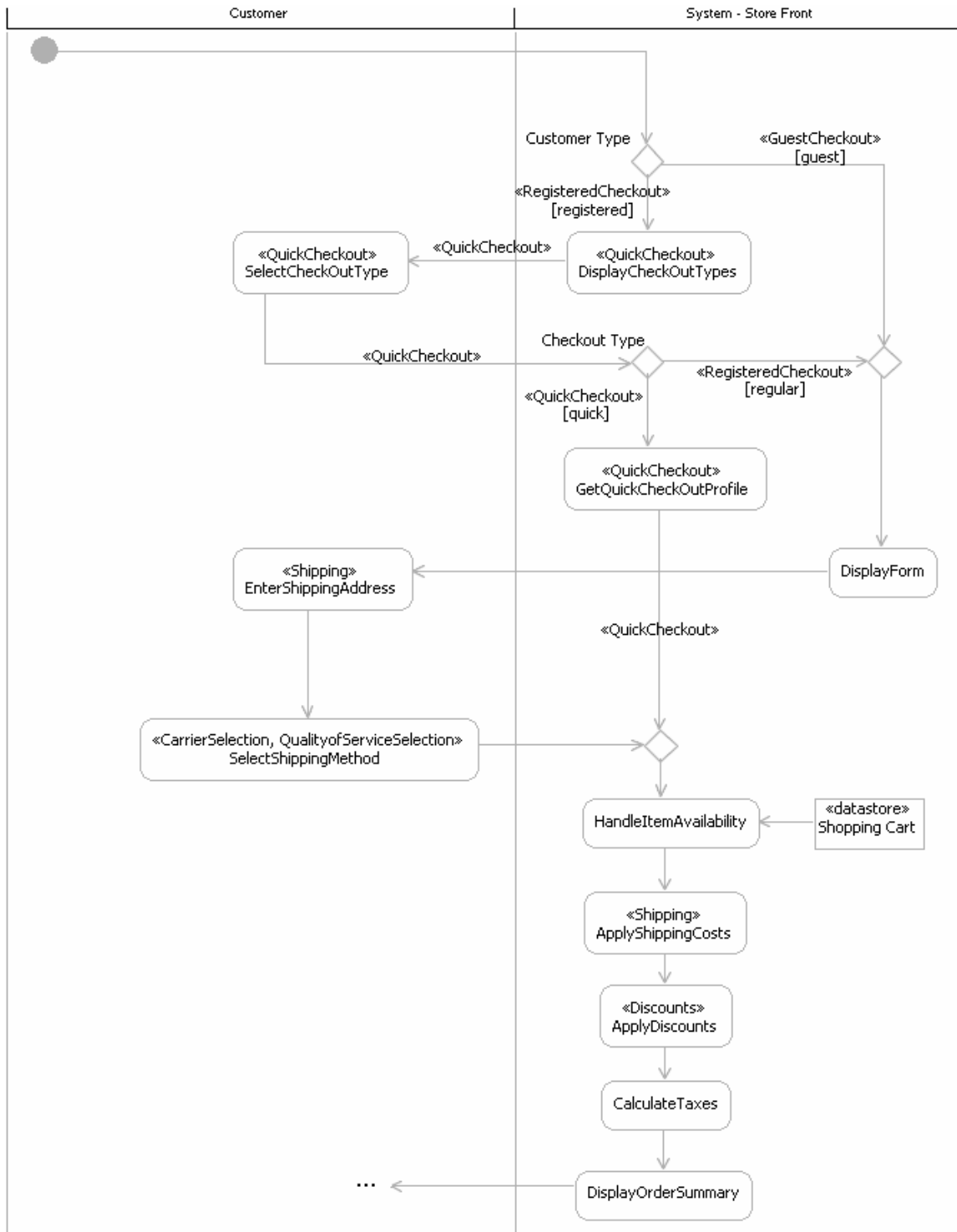


Figure 6.13: CheckoutItems Activity – Phases One and Two

The third phase, shown in figure 6.14, completes the process. It contains all of the elements from the merge node before the EnterPromotionCode action to the final activity node. The region

preceding the CreateOrder call behaviour deals with entering payment data. Promotion codes are entered first, followed by payment information if a regular checkout is being used. The customer then confirms the order in the SubmitInformation action. This causes the CreateOrder activity to be invoked. If the order creation fails either due to an invalid coupon code or payment information, an error message is displayed and the customer is allowed to re-enter the information; otherwise, the store front will clean-up the wish list and the shopping cart. The store front may also display a confirmation to the customer that the order has been submitted successfully.

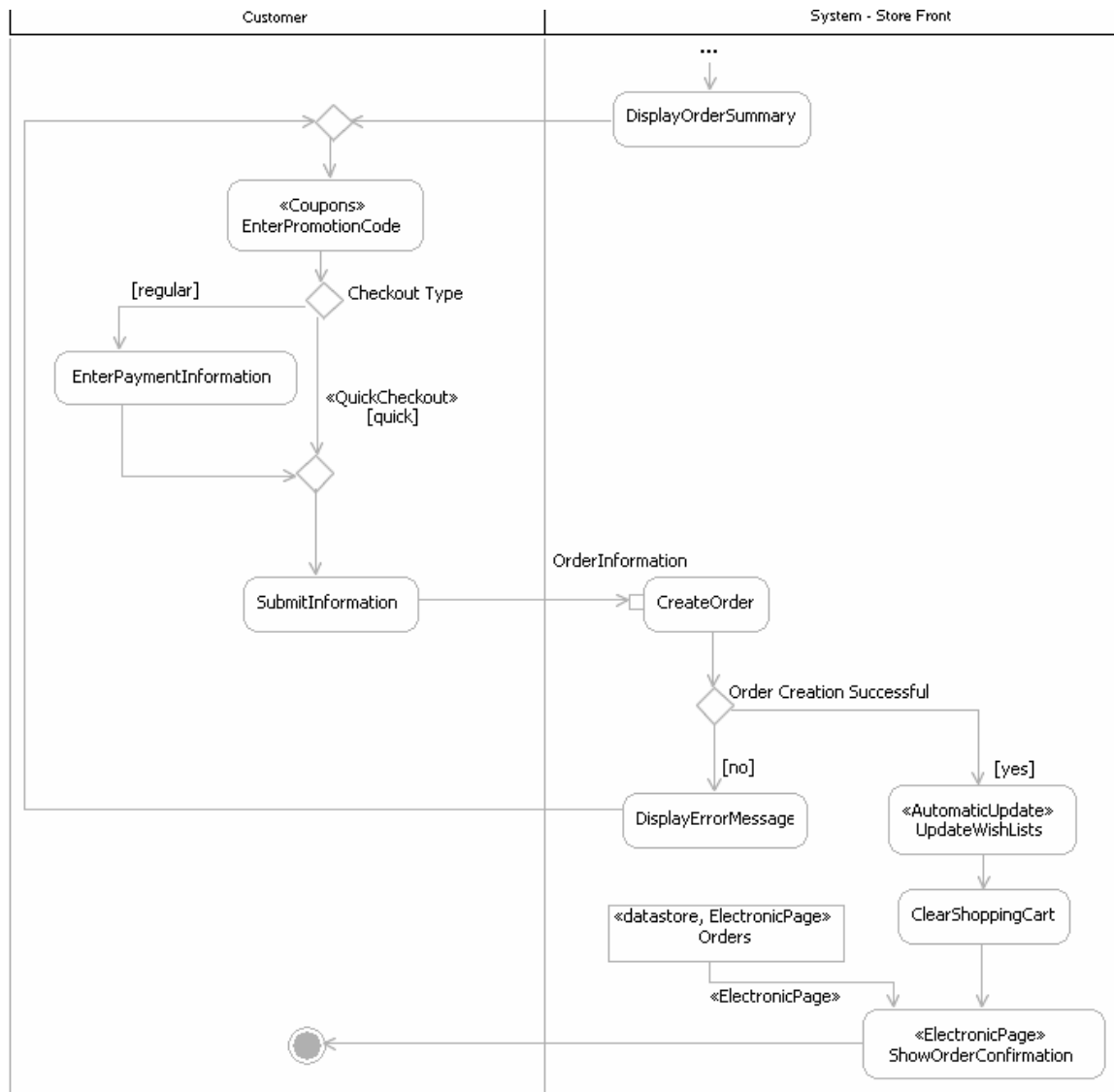


Figure 6.14: CheckoutItems Activity – Phase Three

The checkout process contains many points of variability. These points will be described with respect to the phases established previously. In the first phase, the grouped features of the checkout type are referenced by PCs which are used to annotate the different decision branches. The flow that bypasses the checkout type selection is annotated with the guest checkout feature. The quick checkout PC is used to annotate the elements starting from the DisplayCheckoutTypes action to the outgoing flow of the GetQuickCheckoutProfile action; this region corresponds to the semantics of the quick checkout feature. The registered checkout feature is used to annotate two flows: the “registered” branch of the “Customer Type” decision node and the “regular” branch of the “Checkout Type” decision node. The annotations were strategically placed to maximize the benefits of flow closure and simplification. In the case where the registered checkout feature is selected and the quick checkout feature is eliminated, the quick checkout region will be removed and the two remaining flows will be merged. This will make it identical to the guest checkout branch. In addition to these annotations, a PC referencing the shipping feature is used to annotate the EnterShippingAddress action and a PC referencing to the disjunction of the QualityofServiceSelection and CarrierSelection features is used to annotate the SelectShippingMethod action.

In the second phase, only two actions are annotated: the ApplyShippingCosts action is annotated with the shipping feature and the ApplyDiscounts action is annotated with the discounts feature. This allows the actions to be removed from the path if necessary.

In the third phase, four elements are annotated with different features. The first element is the EnterPromotionCodes action, which is annotated with the coupons feature since codes are the most common way to implement coupons in an e-commerce environment. The second element is the “quick” branch from the “Checkout Type” decision node, which bypasses the EnterPaymentInformation action if the quick checkout is selected. If the feature is eliminated, it results in the removal of the flow, thus requiring the customer to enter their payment information each time the customer makes a purchase. The third element is the UpdateWishLists action, which is annotated with the automatic update feature. If the feature is eliminated, the action is removed from the path and the wish list is not modified at all. The last element is the ShowOrderConfirmation action, whose presence is controlled by the electronic page feature. If the feature is eliminated, the workflow ends after the post-order clean-up, meaning that the customer is immediately returned to the home page after the order is created successfully.

6.2.7 CreateOrder Activity

The CreateOrder activity, shown in figure 6.15, describes the process of using the submitted information to create an order. The activity takes place within the order processor, where it is invoked by the CheckoutItems activity. It is important to note that the store front requires a response before the customer's session can continue in the CheckoutItems activity.

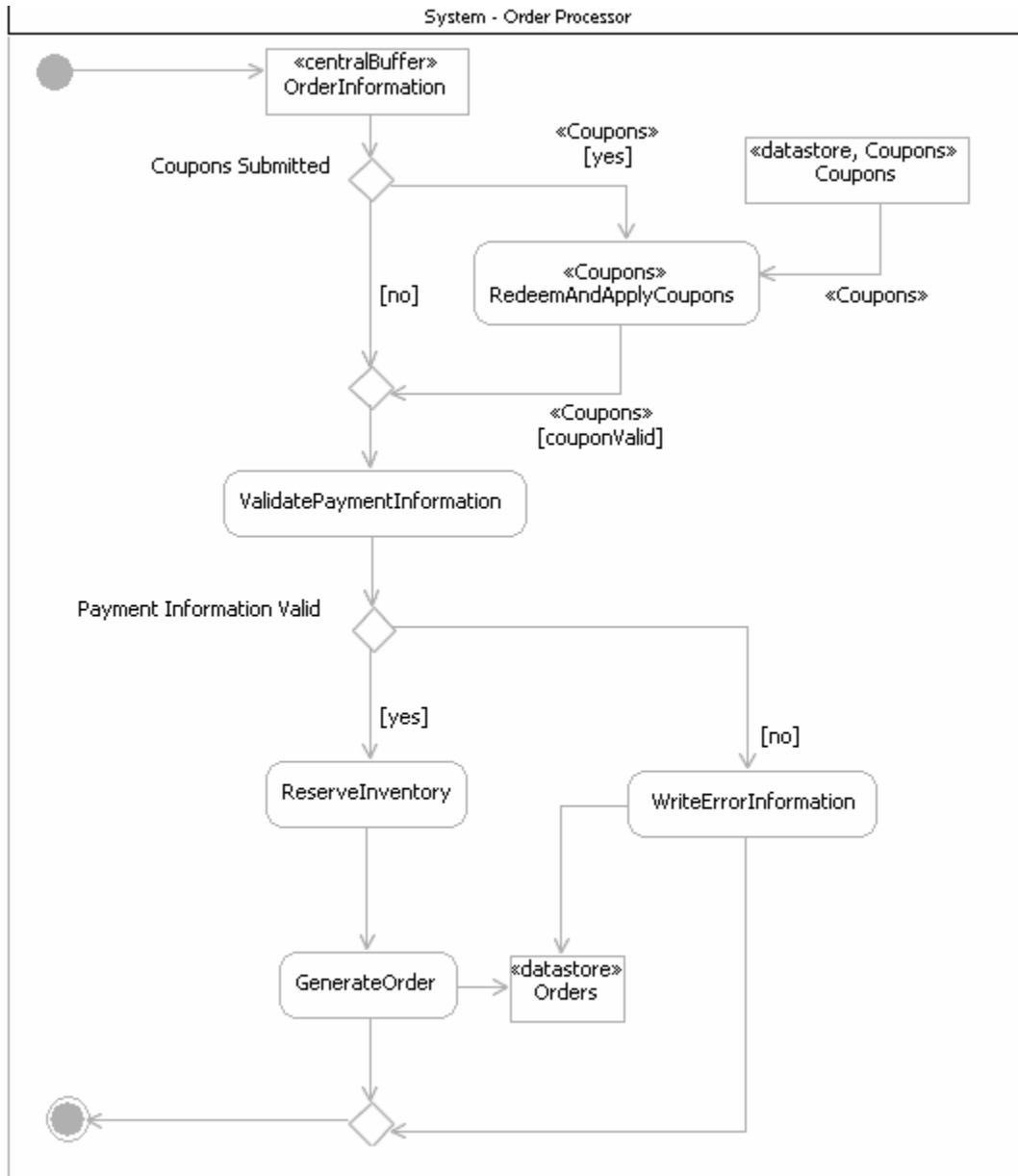


Figure 6.15: CreateOrder Activity

The process begins after the order information is submitted to the order processor. Coupons, if present, are redeemed and processed first. Although the model only depicts the RedeemAndApplyCoupons action pulling information from the coupons database to validate the coupon, the action encapsulates all of the details which are required to handle both unique and reusable coupon codes. The couponValid guard on the outgoing flow ends the activity if the coupon code is invalid, resulting in an unsuccessful order creation. Once the coupons have been redeemed, the payment information is checked. If the information is valid, the order processor will make a request to reserve inventory in the fulfillment centre. Afterwards, it will generate and store the order in the orders database. If the information is invalid, the order processor will generate error information for the order.

This model template only contains a set of annotations pertaining to the coupons feature. Only the “yes” branch of the “Coupon” decision node, along with the required data store, is annotated with the coupons feature. If the feature is eliminated, all of the processing associated with coupons is removed. There are no implicit annotations on the model elements.

6.2.8 OrderProducts Activity

The OrderProducts activity, shown in figure 6.16, describes the ordering process from the customer’s perspective. It assumes that the ordering process begins after the last item is added to the cart; the addition of the final item is modeled in the activity. The last item can be selected and added to the shopping cart either through the FindProduct activity or the entry of the product SKU. The latter method is ideal for gaining quick access to a product if the customer had obtained the product SKU previously. The Add* actions are responsible for retrieving the cart and adding the item to it. Once the product has been added to the cart, the cart must be stored to prevent the loss of the cart contents before the transaction is completed. Both remote and local storage of the shopping cart are taken into consideration. Finally, the call behaviour invokes the CheckoutItems activity. The process ends upon the completion of the activity. There are no implicit or explicit annotations in this model template.

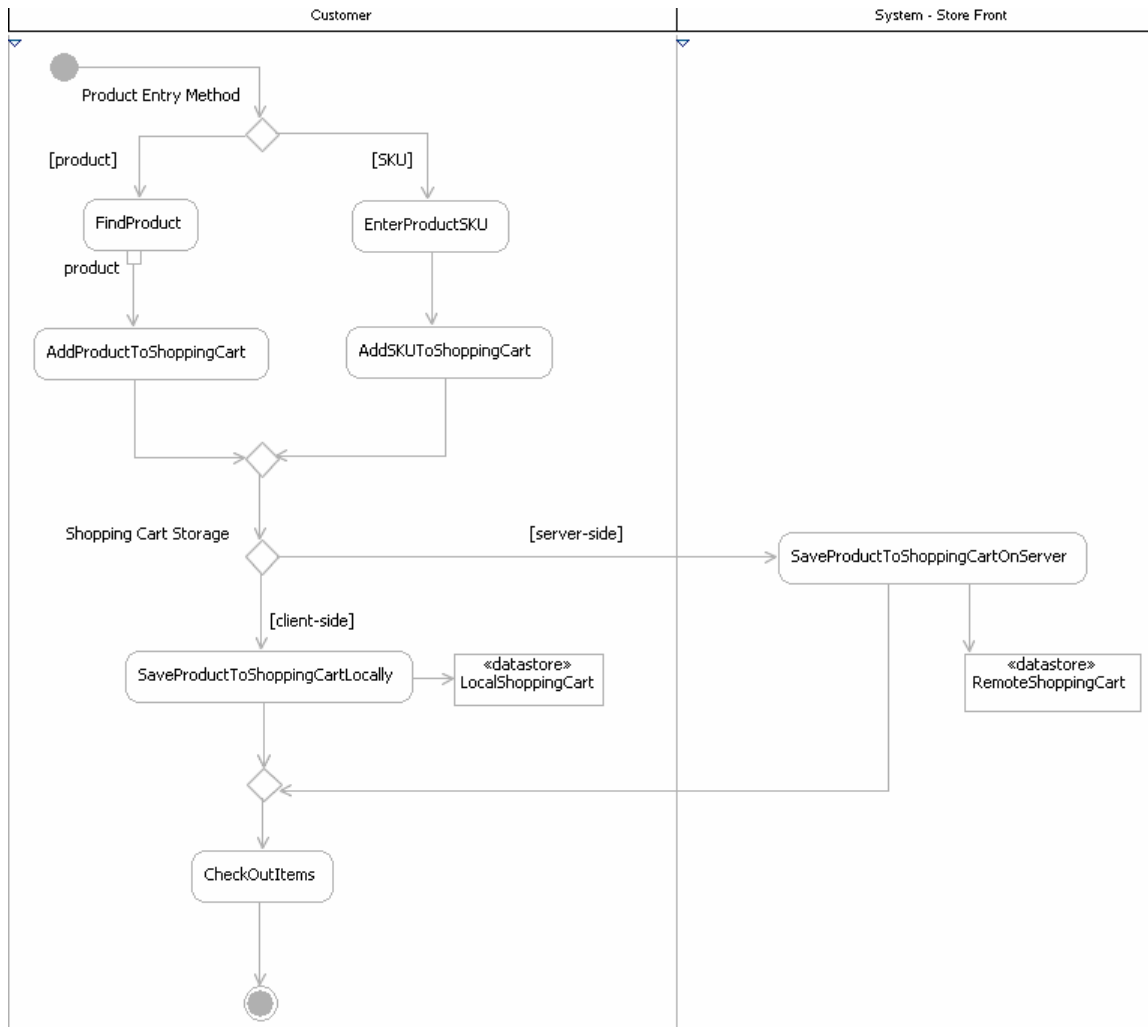


Figure 6.16: OrderProducts Activity

6.2.9 ProcessOrder Activity

The ProcessOrder activity, shown in figure 6.17, describes the workflow involved with managing an order after it is created. It is run within the order processor, which goes through the orders and attempts to fulfill and close each of them. It is based on the IBM Websphere Commerce model for order processing [Ibm05], but it is simplified to focus on aspects that directly relate to the variability presented in the feature model. The activity deals with the workflow for a single order, but the workflow is carried out in the same manner for each submitted order. Different orders can be processed simultaneously as long as concurrency mechanisms are in place in the RetrieveOrder action.

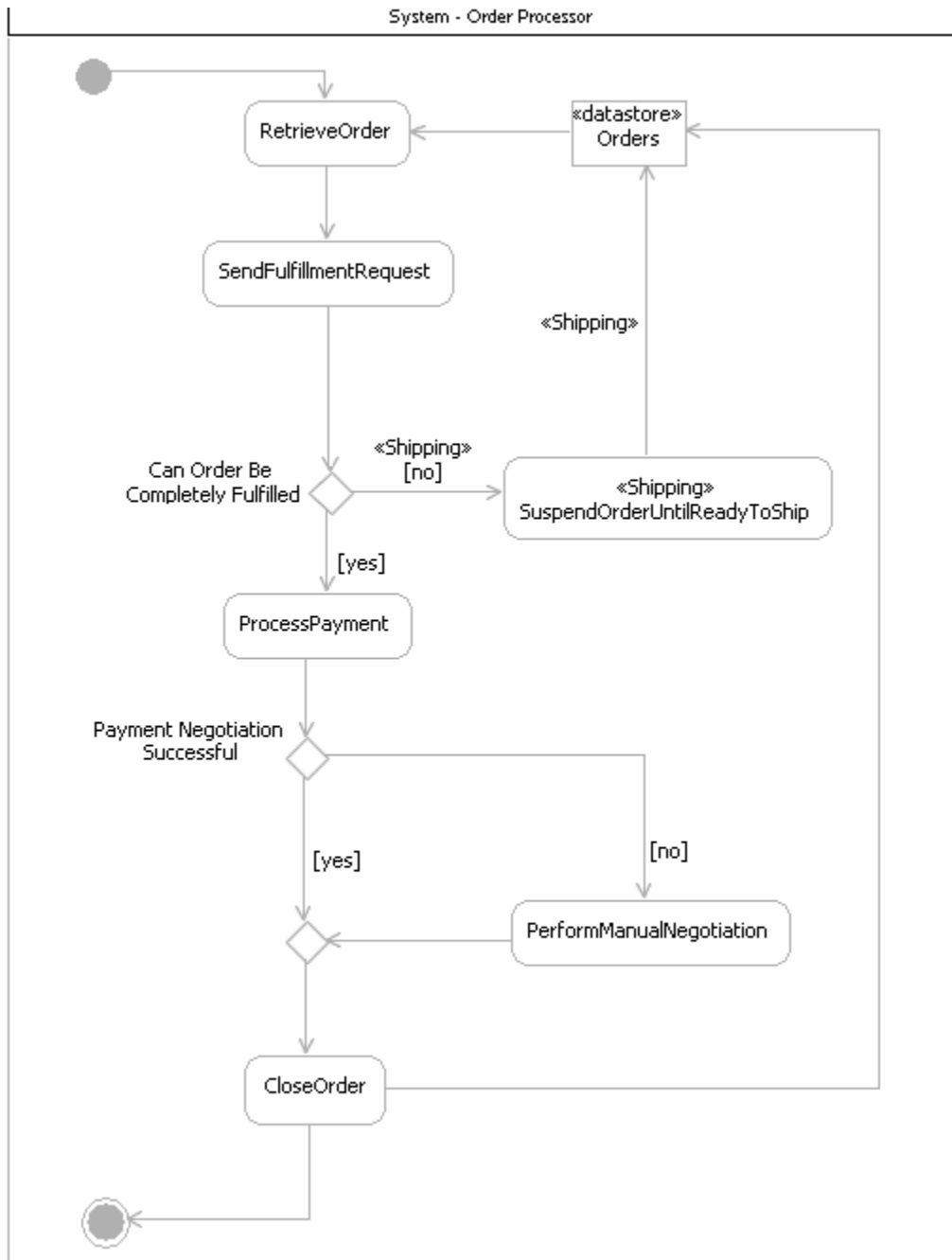


Figure 6.17: ProcessOrder Activity

The activity begins with the **RetrieveOrder** action, which retrieves a single, open order from the database. It sends a fulfillment request to the fulfillment centre; this has been kept general enough to accommodate all product types since the fulfillment feature is a common to all product types. If the order cannot be completed, it is placed in a suspended state and the order status is updated. The order status will have to be updated by the fulfillment centre before the order can be retrieved

and processed again. Once the order is fulfilled, the payment is processed either automatically or manually, depending on if the automated process is successful. Once payment is received, the transaction is complete, the order is closed and the status is updated. For simplicity, this model does not take the multiple shipments feature into consideration. An order is considered atomic, so all of the order items must be available or the order will be suspended.

The only annotations in this model template are the PCs which reference the shipping feature on the “no” branch of the “Can Order Be Completely Fulfilled” decision node. A fulfillment request is defined such that it can fail only if there is no inventory to send to the customer. This is only possible when shipping is required for a physical product.

6.2.10 CheckOrderStatus Activity

The CheckOrderStatus activity, shown in figure 6.18, describes the workflow that allows customers to check their orders through the store front.

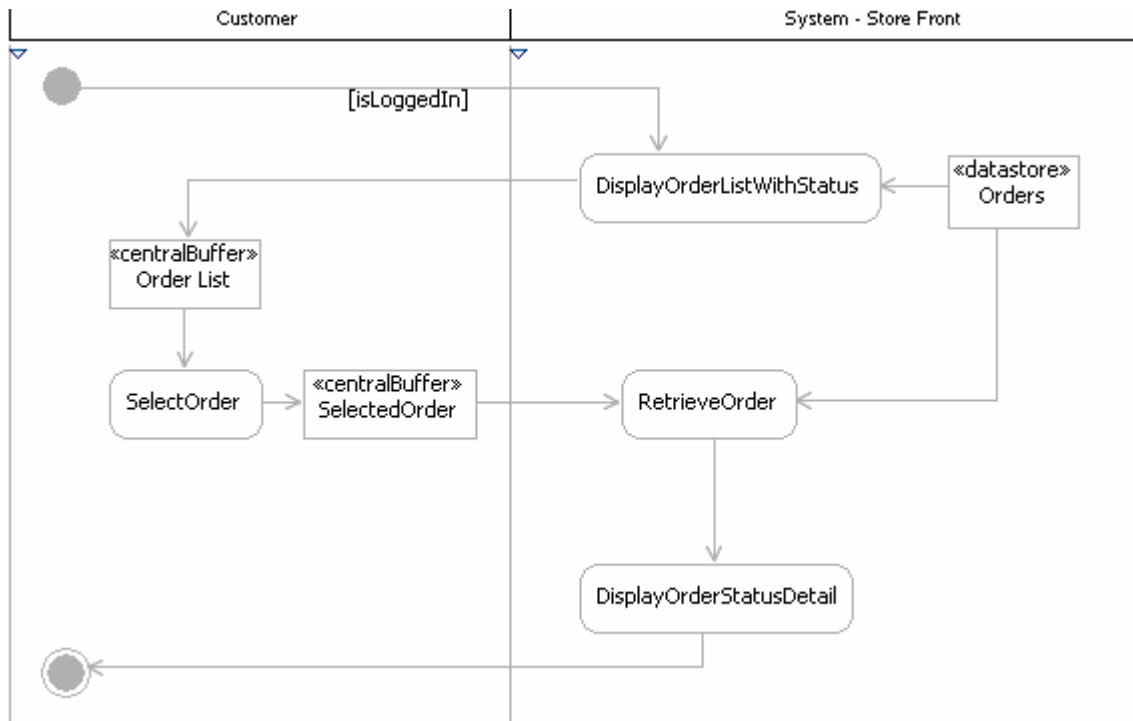


Figure 6.18: CheckOrderStatus Activity

The process begins when the customer makes the request. The `isLoggedIn` guard ensures that the customer is logged in, which allows the store front to determine which customer's orders to display with the `DisplayOrderListWithStatus` action. The customer selects one of the orders, which is retrieved and displayed by the store front. The `DisplayOrderStatusDetail` action opens the order status page, which is used to display all of the order details.

In this model template, the activity is annotated with the order status viewing feature. Therefore, all of the model elements in this activity are also implicitly annotated with the order status viewing feature.

6.2.11 RefundOrder Activity

The `RefundOrder` activity, shown in figure 6.19, describes the workflow for returning an order at the e-shop due to problems with the merchandise or customer dissatisfaction. It requires communication between several different actors. In this model, the activity focuses on returning physical goods. In addition, many of the details have been simplified to keep the model size manageable.

The process begins with the customer making a request to return the item. The request is passed through the store front to the order processor, which makes a decision about whether or not to accept the return in the `ObtainApprovalForReturn` action. The approval of the request leads to the generation of a RMA number, which is stored in the returns database and sent to the store front for return label generation. The customer uses the label to ship the item back to the warehouse, which is responsible for processing the item. The `isMatched` guard indicates that the item must match an approved return file in order for the process to continue; otherwise, the workflow ends with an error condition, which is not specified in this model. The returned goods are also inspected for its condition and the warehouse will notify the order processor of the status. Depending on the condition of the goods, the warehouse will notify the order processor to issue a refund or to contact the customer for an explanation.

The `RefundOrder` activity does not model any variability in the return process; however, the activity itself is annotated with the product returns feature. Therefore, all model elements contained in this activity are implicitly annotated with the product returns feature.

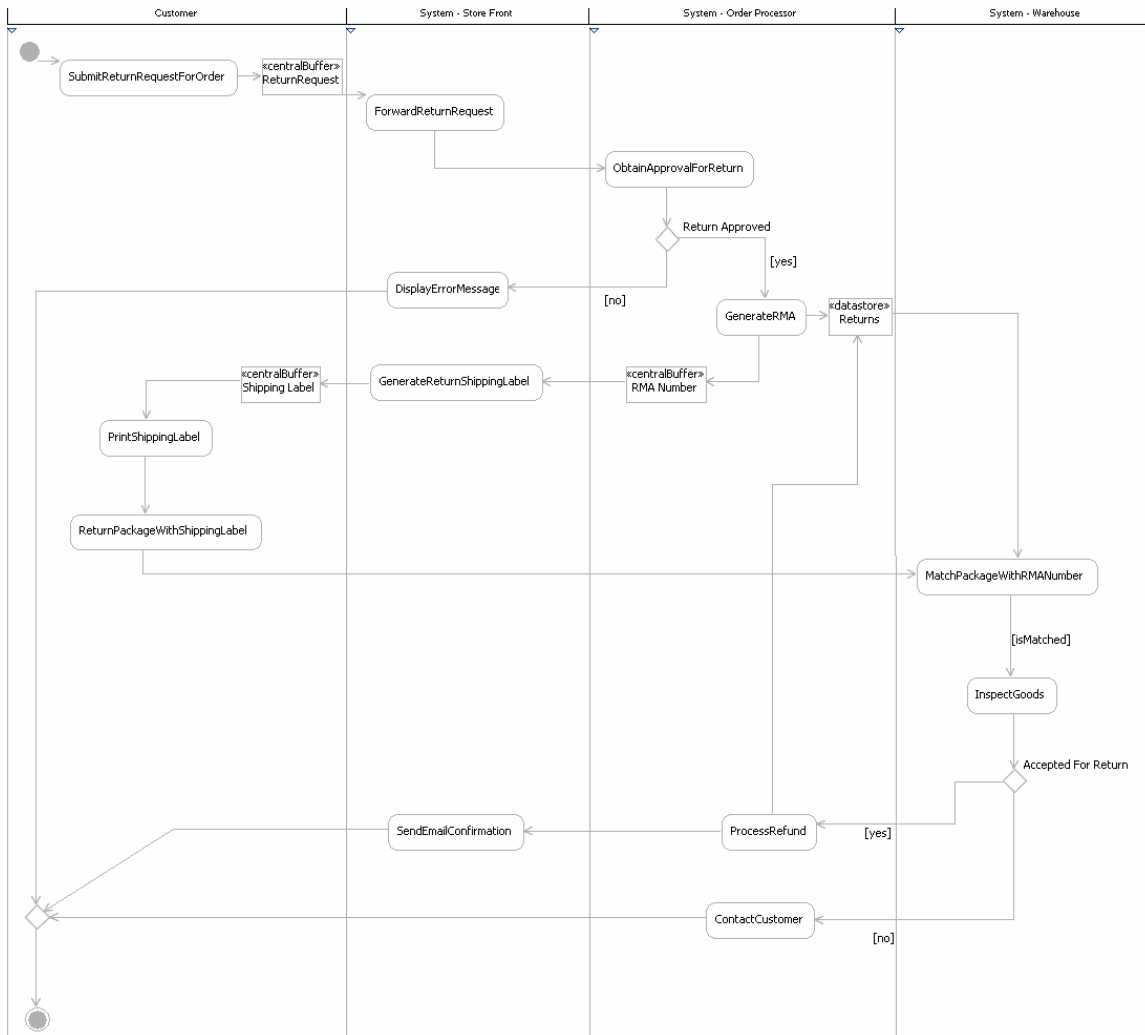


Figure 6.19: RefundOrder Activity

6.2.12 RegisterWithTheStore Activity

The RegisterWithTheStore activity, shown in figure 6.20, describes the workflow for the registration process in the e-shop. The process begins with the store front soliciting the customer for registration information. In the EnterRegistrationInformation action, the customer enters information for different profile fields, such as login credentials. The information is validated to ensure well-formedness and conformance to any policies, such as ensuring that the password follows a specific format. If the validation fails, an error message is displayed and the customer can re-enter the information; otherwise, the store front will create the profile and send an e-mail confirmation. As a security precaution against bots and spammers, the profile remains inactive until the customer clicks the link in the e-mail to activate the account. The link must be clicked

before the confirmation period expires; otherwise, the profile expires and the registration process must be restarted. There are no explicit annotations, but all model elements are implicitly annotated with the registration feature due to the annotation of the registration feature on the activity.

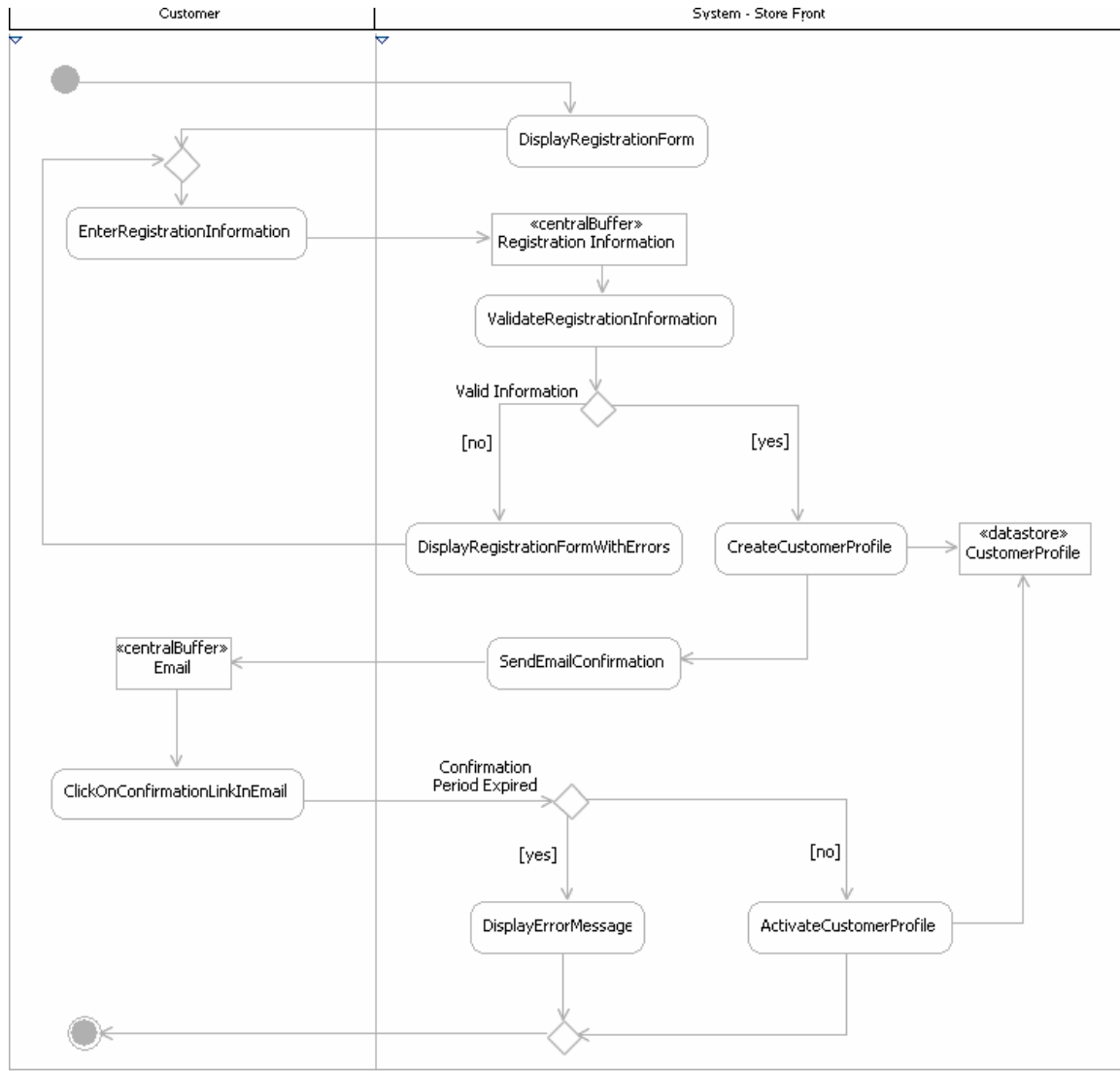


Figure 6.20: RegisterWithTheStore Activity

6.2.13 UpdatePersonalProfile Activity

The UpdatePersonalProfile activity, shown in figure 6.21, allows a registered customer to make changes to their profile information. A pre-condition is defined by the isLoggedIn guard, which checks that a registered customer is logged in before allowing the process to continue; otherwise,

it will display the log in page and end the workflow. The process begins when the customer selects the profile that they want to update. The workflow for both profile types is similar and consists of two steps: 1) the profile information is retrieved and displayed, and 2) the customer is able to make modifications to the data, which is sent back to the store front to update the profile. The only difference lies in the profile retrieval; when the store front executes the GetQuickCheckoutProfile action, it determines if a quick checkout profile exists. If the profile exists, the workflow proceeds normally; otherwise, the CreateQuickCheckoutProfile activity is invoked.

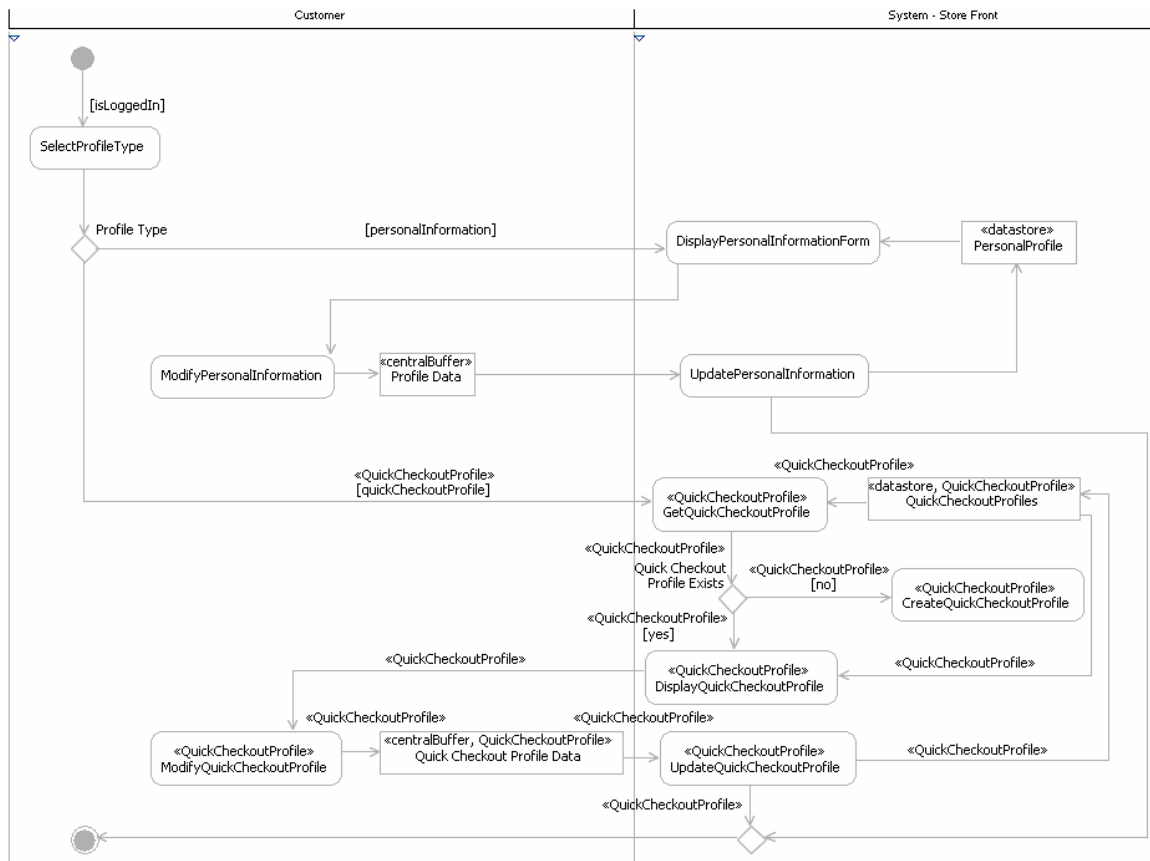


Figure 6.21: UpdatePersonalProfile Activity

There is only one set of explicit annotations in this model template; they pertain to the quick checkout profile feature. All elements in the “quickCheckoutProfile” branch of the “Profile Type” decision node are annotated with the feature since the entire region deals specifically with the quick checkout profile. If the feature is eliminated, only the branch for updating the registration profile will remain. All model elements are also implicitly annotated with the registration feature

since the activity is annotated with the registration feature; this is because profiles are stored for registered users only.

6.2.14 CreateQuickCheckoutProfile Activity

The CreateQuickCheckoutProfile activity, shown in figure 6.22, allows a customer to define a quick checkout profile. The process begins with the store front displaying a form which allows the information to be input. There are three components to the quick checkout profile: the shipping address, the shipping options, and the payment information. The EnterShippingAddress and EnterPaymentInformation actions allow the customer to select pre-existing data in their profile or to enter a new set of information. The SelectShippingMethod action is identical to the one in the CheckoutItems activity. Once the information is fully specified, the store front creates the profile and stores it in the database. The CreateQuickCheckoutProfile activity is used by the UpdatePersonalProfile activity when the UpdatePersonalProfile activity is unable to find a pre-existing quick checkout profile.

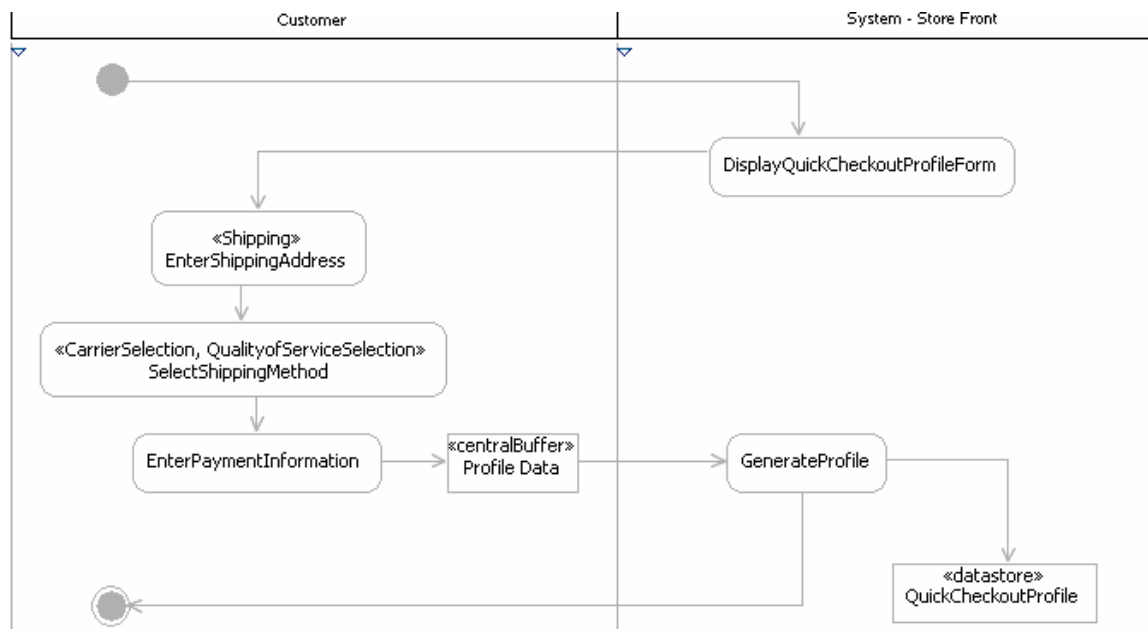


Figure 6.22: CreateQuickCheckoutProfile Activity

The annotations on the EnterShippingAddress and SelectShippingMethod actions are identical to the ones in the CheckoutItems activity. All model elements are implicitly annotated with the

quick checkout profile feature due to the annotation of the quick checkout profile feature on the activity.

6.2.15 ResetPassword Activity

The ResetPassword activity, shown in figure 6.23, describes the process of having a customer reset their password because s/he has forgotten it.

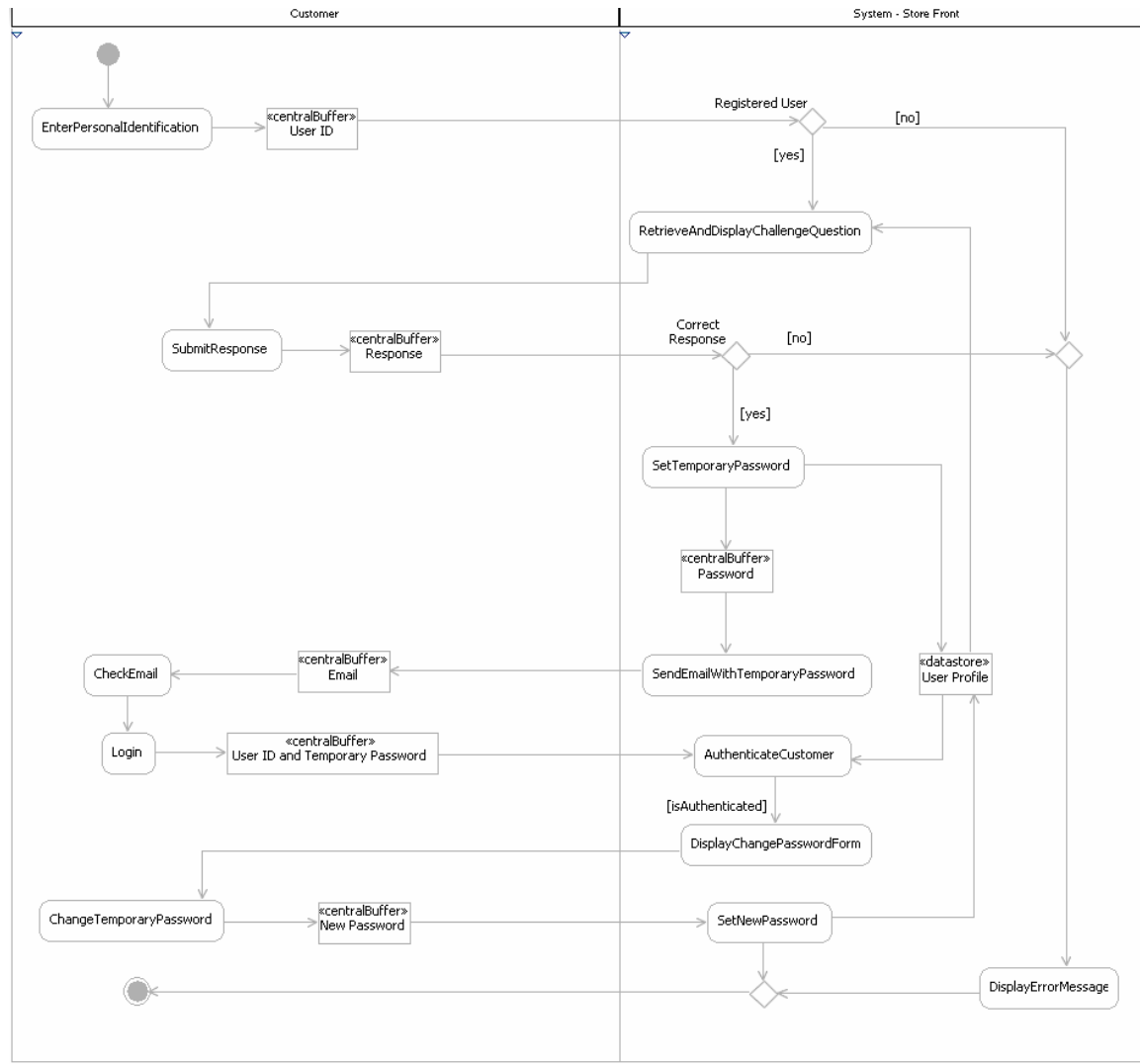


Figure 6.23: ResetPassword Activity

The process begins with the customer entering their account id, which is used to confirm if the user is registered with the e-shop and retrieve the challenge question. The challenge question prevents unauthorized users from resetting a customer's password. If the customer answers the

challenge question correctly, the password is reset to a temporary password and the temporary password is sent to the customer by e-mail. When the customer logs in the next time, the store front will authenticate them using the temporary password and display the change password form. The customer must set a new password at that time. If any part of the process fails, an error message is displayed and the activity ends. Also, the `isAuthenticated` guard following the `AuthenticateCustomer` action simply ensures that the log in has completed successfully; otherwise, it will display an error message and allow the customer to try again. There are no explicit annotations, but all of the elements are implicitly annotated with the registration feature due to the annotation of the registration feature on the activity. The activity is annotated because a customer will have an e-shop password only after they register.

6.2.16 CreateWishList Activity

The `CreateWishList` activity, shown in figure 6.24, allows a customer to create a wish list. This activity does not deal with naming the new wish list or creating multiple wish lists since those tasks fall under wish list management, which is beyond the scope of this set of models. The process begins when the customer adds the first product to the wish list, assuming that there are no other existing wish lists. The `AddToWishList` action initializes the wish list and adds the product selected by the `FindProduct` activity. Depending on the customer type, the wish list is stored locally or remotely. Since registered customers have a profile, their lists are stored with the profile on the server side. Guest customers will require local storage; the store front serializes the wish list in the e-shop format and sends it to the client side for storage.

There are two sets of explicit annotations in this model template. The first set pertains to the registration feature, which is used to annotate elements in the “registered” branch of the “Customer Type” decision node. The second set pertains to the wish list saved after sessions feature, which is used to annotate elements in the “guest” branch of the “Customer Type” decision node. Both sets of annotations are based on the storage method, similar to the way the annotations were used in the `SelectProductFromWishlist` activity. Finally, there is an implicit annotation of the wish lists feature on all model elements, which is caused by the annotation of the wish lists feature on the activity.

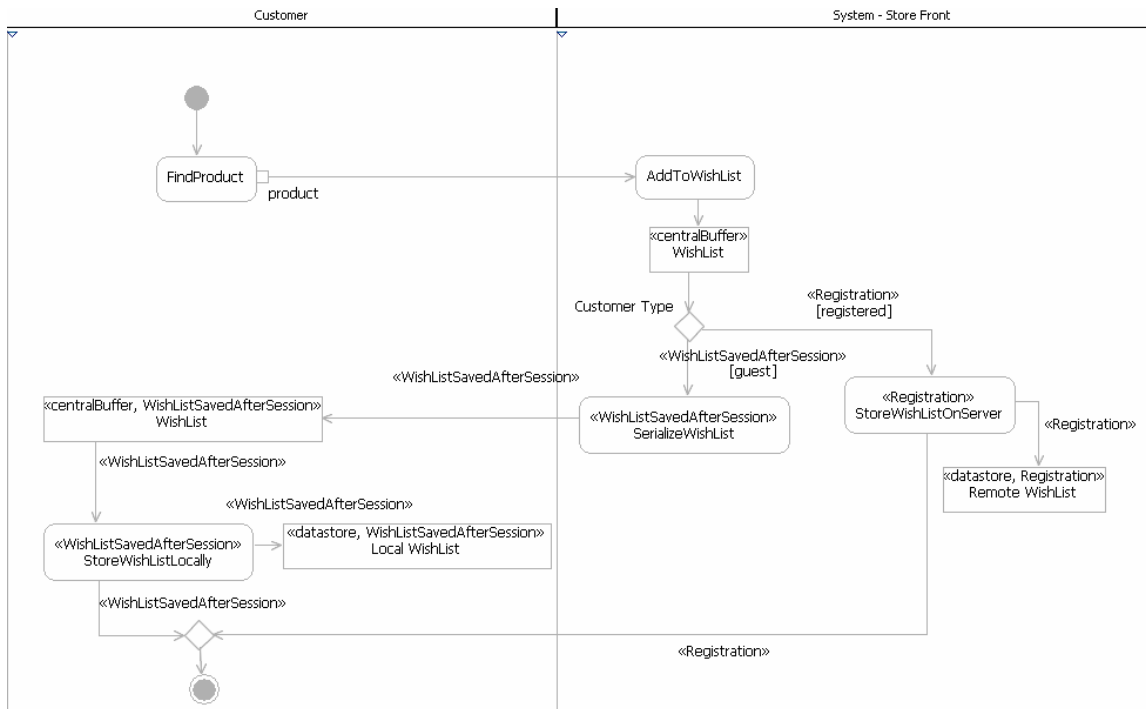


Figure 6.24: CreateWishList Activity

6.2.17 SendWishList Activity

The SendWishList activity, shown in figure 6.25, allows a customer to send their wish list to a potential gift-giver by e-mail. The process begins with the store front displaying a list of the customer's wish lists. The customer selects a wish list, which is retrieved by the store front²⁹ and displayed along with a form to enter the recipient information. The customer enters the recipient's name and e-mail address, which is used to generate the e-mail with the wish list. The e-mail can take the form of a greeting message combined with a product list containing all of the items in the wish list or a link to view the wish list at the e-shop. After the e-mail is composed, the store front sends the e-mail to the recipient.

There is one set of explicit annotations on the elements in this model template. A path of elements, starting from the DisplayCustomerWishlists action and ending with the "WishList Name" central buffer, is annotated with the multiple wish lists features. In addition, there is an additional control flow between the SelectWishListFromList action and the RetrieveWishList action which is present if the multiple wish lists feature is eliminated. This additional flow is

²⁹ It should be noted that the additional constraints state that this feature is only available to registered customers; therefore, routines for retrieving a local copy of the wish list are not included.

needed to perform flow closure between the initial activity node and the RetrieveWishList action because it introduces the correct flow type to connect the two elements. In addition, all model elements are also implicitly annotated with the e-mail wish list feature due to the annotation of the feature on the activity.

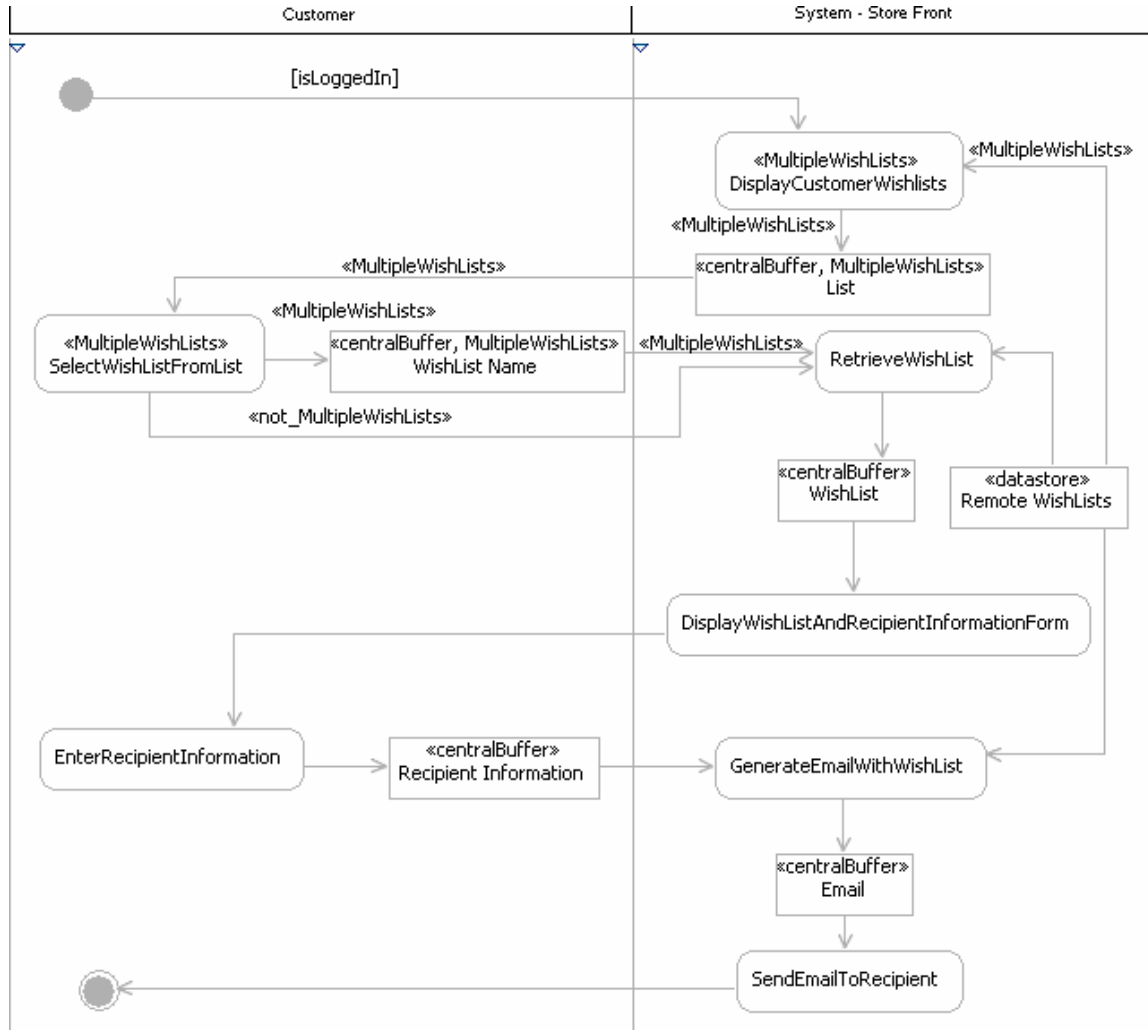


Figure 6.25: SendWishList Activity

6.3 Summary

In this chapter, the design of the model templates and notable annotations were presented. The e-shop model templates consist of the store front entity model, the service model and the activity diagrams. The store front entity model describes the structure of entities, such as products, customers, orders, shopping cards and wish lists. The service model describes how the e-shop can leverage services. The activity diagrams consist of seventeen models; through the combination of workflows presented, one can see how a customer could browse and purchase products at the e-shop.

7 Evaluation of the Feature-Based Model Template Approach

This chapter is devoted to an evaluation of the feature-based model template approach through the development of the e-commerce models. Feature model and model template development was performed using *fmp2rsm* [Cza05g], hereafter referred to as the development tool. Key points are supported by examples that were observed throughout the development process. Unless otherwise indicated, examples are taken from the final version of the models.

There are five sections in this chapter. The first two sections describe the analysis of specific aspects of feature modeling and model template development respectively. The third section builds upon the analysis and describes candidates for modeling guidelines. The fourth section presents a set of recommendations that are based on observations or key challenges encountered throughout the development process; the recommendations suggest improvements for both the process and the development tool. The final section summarizes the chapter.

7.1 Analysis of Variability Modeling in Feature Models

A common part of the feature modeling process is to express requirements in terms of features and to model the relationship between the features. In some cases, the requirements provide enough information to express domain entities or entity operations as features directly; however, there are some cases where further analysis is needed to discover binding time and variability issues.

In this section, an example of integrating a new requirement into the existing e-commerce feature model is presented. The entire process is examined in detail and several candidate models are proposed and analyzed. The discussion is concluded with an examination of how binding time plays a role in modeling variability. The requirement to be used in this example is defined as follows:

An e-shop may have the option to restrict access to and use of wish lists to registered customers only.

The three key points in this requirement are 1) the option to restrict wish list access to registered customers implies that there are two possible outcomes: allow only registered customers to access

wish lists or allow both registered customers and guests to access wish lists; 2) the ability to restrict wish list access may or may not be available in an e-shop; and 3) the decision to restrict wish list access is a policy that is decided by the e-shop at run-time³⁰. As a side note, there is no mention about the validity of a scenario where wish list access is restricted to guest customers only. This scenario was never observed during domain analysis, so it was decided that it would not be included in the model.

An examination of the feature model indicates that the wish list and registration features are orthogonal. Their common ancestor feature is the store front, meaning that the only commonality between them is that they are both features in the store front. Two different techniques for modeling this requirement are proposed, followed by an analysis of the trade-offs for each technique.

The first technique models the requirement by using a subfeature relationship. An optional feature, “requires registration”, is introduced and made a subfeature of wish list as shown in figure 7.1. Selection of the “requires registration” feature implies that a customer must be registered before being able to access the wish list³¹. This representation of the requirement fails to recognize the binding time property of the feature. Assuming that the instantiation of a concrete e-shop occurs only when a feature configuration is reached and the configuration process occurs over a single level, the “requires registration” feature must be selected or eliminated before the e-shop product can be deployed at run-time. Therefore, the choice is made at a binding time which prevents the e-shop administrator from changing the policy; this violates the third key point.

The problem with this technique is that it does not address the fact that there are two decisions that must be made at different binding times. The decisions are:

- 1) if wish list access should have any dependency on registration. If it does not, the default behaviour is to allow wish lists to be accessed by both guests and registered users. If it does,

³⁰ To be more precise, this decision is made at a later binding time than the decision in the second point, meaning that this decision could be made during build-time; however, the intent of the requirement is to allow the decision to be made for each e-shop individually at run-time..

³¹ In this technique, an additional constraint, (requiresRegistration) requires (registration) must be added; however, the presence of the constraint does not impact the rest of the example.

- the second decision, specified below, must be made. This decision is the essence of the second key point and is used to scope the feature for the e-shop product line at build-time,
- 2) the run-time policy for dealing with whether or not registration is required for wish list access. This is the essence of the first and third key points with the two possible outcomes being described in the first key point. This decision is used to configure the e-shop at run-time after it is deployed.

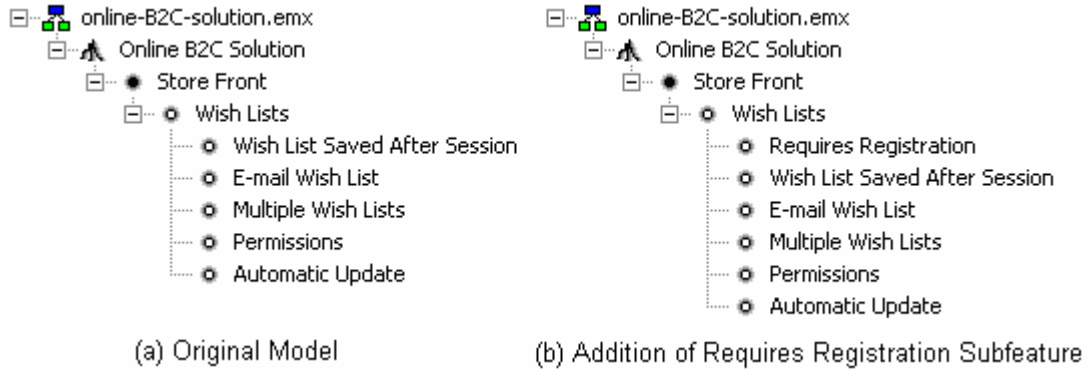


Figure 7.1: Adding Wish Lists Subfeature for Modeling Requirement

The second technique models the decision structure directly, as shown in figure 7.2. An optional feature, “dependent on registration”, is introduced and made a subfeature of wish lists³². The inclusive-or feature group is added to represent the two policies in the form of the grouped features “requires registration” and “does not require registration”. Although allowing both policies to be selected simultaneously seems contradictory, this model allows the concrete e-shop to support one or both policies. This feature model can be specialized over multiple stages also. There are no restrictions on when the decisions are made except that the dependency on registration feature must be decided upon before the policy selection. Therefore, this technique produces a model that best represents the requirement because it takes binding time issues and the decision making process into consideration³³.

³² In this technique, an additional constraint, (dependentonRegistration) requires (registration), must be added; the subfeature relationship covers the requires registration feature.

³³ The same process was used to model the registration enforcement feature described in section 4.2.1.

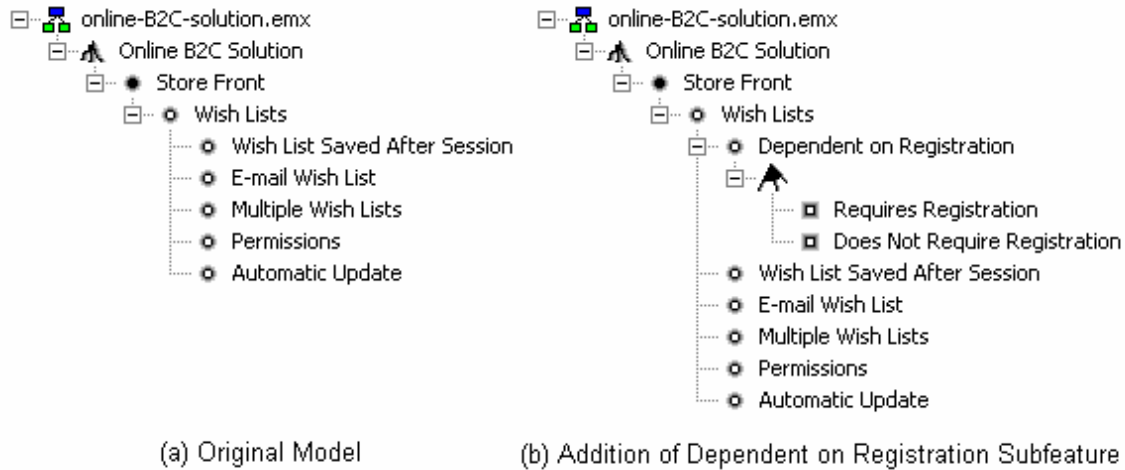


Figure 7.2: Adding Policy Subfeatures for Modeling Requirement

From a configuration point of view, it is interesting to note that the model produced by the first technique accurately describes the system from a run-time perspective because the choice of requiring registration for wish list access can be reduced to an optional feature, as shown in figure 7.1(b). This indicates that the first technique is appropriate if the requirement focuses on the run-time configuration only. Another interesting point about the configuration is that if the “dependent on registration” feature is eliminated, it is functionally equivalent to selecting the “does not require registration” feature because both will allow registered customers and guests to access the wish list; however, the differences lie in who makes the decision and the actual configuration of the e-shop product due to the binding time difference.

7.2 Analysis of Annotations in Model Templates

The annotation analysis focuses on examining how feature annotations are applied to model elements. The objective of the analysis is to determine if there are any frequently occurring patterns for annotations which can be leveraged in the model template development process. The analysis involves the examination of each model template and the categorization of the sets of adjacent model elements which are annotated with the same feature³⁴. This section summarizes the findings of the analysis and presents the annotation categories which were observed in the model templates. The results of the analysis are provided for class diagrams and activity diagrams in separate sections. The complete set of the analysis results is presented in appendix B.

³⁴ From this point forward, these categorizations will be referred to as “annotation categories” for brevity.

7.2.1 Class Diagram Model Templates

The two class diagrams in the e-commerce model were of different types; one used regular classes to depict entities in the store front, whereas the other used stereotypes to depict services which interacted with the store front. The annotation categories were different for the two class diagrams; therefore, the diagrams are described in separate subsections.

7.2.1.1 Store Front Entity Model

Adjacent model elements in the store front entity model tend to be annotated with different features. Furthermore, the annotations in this model only referred to a single feature. The results of the annotation analysis for the store front entity model are presented in table B.1 in appendix B. The following is a summary of the annotation categories which were observed in the model. Most of the annotation categories, which have been bolded, correspond to individual elements in a class diagram.

Class denotes any concrete class in the model that is annotated with a single set of features. The class may be involved in generalization, association or dependency relationships, but this classification implies that none of the model elements pertaining to those relationships are annotated with the same set of features as the concrete class. The context in the model element column of table B.1 describes two things: 1) the corresponding class in the model, and 2) indicates if the class is a subclass and, if so, what the superclass is. Classes are typically annotated with features that represent key concepts, such as a wish list or a registered customer, whereas subclasses are usually annotated with types, such as physical and electronic goods.

Abstract class denotes the annotation of an abstract class. Like the class annotation category, this classification applies when the abstract class is the only annotated element. The context in the model element column of table B.1 describes the corresponding class in the model. Abstract classes are often used as a grouping mechanism, as seen in the AssociatedAssets class.

Attributes denote any member variable that is explicitly annotated. Implicit annotations that are a result of the containing class or abstract class are not considered in this category. The context in the model element column of table B.1 is used to denote the class which the annotated attribute

belongs to. An example of the annotation category is the quality of service selection feature on the serviceType attribute in the ShippingInstructions class.

Enumeration denotes the annotation of an enumeration. The context in the model element column of table B.1 is used to denote the name of the annotated enumeration. Enumerations are often annotated with a feature that describes a type, e.g. an abstraction used to describe type values. An example is the MediaTypes enumeration.

Enumeration value denotes the annotation of an enumeration value. The context in the model element column of table B.1 is used to denote the enumeration which contains the enumeration value. The features used in these annotations typically represent type values. The annotations on the sound, video and image enumeration values are examples of this annotation category.

Composite denotes the annotation of a composite aggregation. The context in the model element column of table B.1 describes the relationship defined by the composite aggregation. It only occurs once in the entity model where the subcategories relationship in the Category class is annotated with the multilevel feature. In this case, the feature in the annotation denotes the nesting of categories, which is best represented by composition.

Association denotes the annotation of a non-containment relationship between classes. The context in the model element column of table B.1 denotes the classes which are involved in the association. Associations are applied in two ways: 1) a directed association that begins and ends with the same class, and 2) a regular association. An example of the former is the substitute product feature in the Product class; an example of the latter is the shipping address feature for the RegisteredCustomer class. The number of associations without annotations greatly exceeds the number with annotations, which is a direct result of the IPC for associations.

MultiplicityME denotes the annotation used for calculating the value of a multiplicity on an association end. This is used in several places, such as calculating the multiplicity for Category based on the multiple classification feature.

7.2.1.2 Service Model

Adjacent model elements in the service model also tend to be annotated with different features. Furthermore, the annotation in this model is referred to a single feature. Only three types of model elements are annotated: operations, services and interfaces. These types make up the three annotation categories observed in this model. The results of the annotation analysis for the service model are presented in table B.2 in appendix B. The following is a summary of the annotation categories which were observed in the model; the categories have been bolded.

Operation denotes the annotation of service or interface operations. The context in the model element column of table B.2 denotes the name of the operation container. An example of this annotation category is the product returns feature for the processRefund operation in the IOrder interface. It should be noted that operations which are contained within an interface or service are implicitly annotated with the container's annotation; the implicit annotation is not considered as an instance of this annotation category. This is the most common form of annotation in service models, although it should be noted that operations are also the most frequently occurring type of model element in this model.

Interface denotes the annotation of an interface. The context in the model element column of table B.2 denotes the name of the annotated interface. An example of this annotation category is the warehouse management feature on the IWarehouseBackend and IWarehouseFrontend interfaces. The annotation serves to enable a set of functionality for the Store Front component.

Service denotes the annotation of a stereotyped class that implements an interface. The context in the model element column of table B.2 denotes the name of the annotated service. An example of this annotation category is the shipping gateways feature on the Shipping Gateway service. The annotation of a service is independent of the annotation on the service's interface.

7.2.2 Activity Diagram Model Templates

There are a significant number of cases in which adjacent model elements in the activity diagram model templates are annotated with the same feature. This resulted in a large number of annotation categories. In addition, there are two instances in which an annotation refers to a disjunction of features; these are the only occurrences of an annotation referencing multiple

features in the entire model. The results of the annotation analysis for activity diagrams are presented in table B.3 in appendix B. The descriptions used in the model elements annotated column of the table are described in detail in appendix B. The following is a summary of the annotation categories which were observed in the model. Half of the categories correspond to individual elements while the other half corresponds to regions in the activity diagram; the annotation categories have been bolded.

Action denotes the annotation of an action node. This classification applies when only the action is annotated; there are other classifications when adjacent flows or actions are annotated with the same feature. This annotation category occurs frequently, although the feature used in the annotation usually describes an entity which is related to the action. For example, the RedeemAndApplyCoupon action in the CreateOrder activity is annotated with the coupons feature because there is no coupon redemption feature in the feature model; the ability to redeem coupons is assumed when the coupon feature is selected.

Activity denotes the annotation of an activity. The purpose is to control the presence of all elements contained within the activity through a single annotation. It can be thought of as an enabling condition on the activity. For example, the CreateWishList activity is annotated with the wish list feature since one cannot create a wish list if it is not supported in the first place.

Call Behaviour denotes the annotation of an action that invokes another activity. It is rare to find a call behaviour that is annotated without some adjacent elements being annotated with the same features also. Call behaviours are annotated to control the invocation of an activity within a workflow. Thus, it is used to prevent an activity from being called when it is not appropriate for the workflow; however, most of the annotations on the call behaviours are used to control the presence of the activity, which is supposed to be done through the annotation on the activity. This is due to the fact that the IPC on call behaviour is not implemented in the tool at this time; once it is, the number of annotated call behaviours is likely to decrease.

TypeME denotes the annotation used for calculating the type of an element on a data node, data pin or activity parameter. This occurs twice in the model. The first occurrence is when a central buffer node in the FindProduct activity is annotated with a TypeME that is used to calculate the type of the central buffer based on the product variants feature. The second occurrence involves

the same types of elements, but it is in the `SelectProductFromCatalog` activity, where the `TypeME` is used to calculate the type of a central buffer based on the categories feature.

Decision Branch denotes the annotations on a set of adjacent model elements along a path, starting from the outgoing flow of the decision node and ending at the incoming flow of the first merge or decision node encountered. In this situation, the feature represents a decision outcome that is tied to a series of actions; selection of the feature makes the decision outcome available. This annotation category was observed several times, such as with the advanced search feature in the `SearchProduct` activity and the coupons feature in the `CheckoutItems` activity.

Decision Branch Bypass is a special case of a decision branch where the set of adjacent model elements consists of a single element only – the outgoing flow from a decision node³⁵. Like the decision branch, the annotation captures the feature’s implication of the decision outcome. The purpose of creating a separate classification is to capture situations where a feature corresponds to no additional actions with respect to the other decision outcomes. There are a few occurrences of this annotation category in the model, such as the basic search feature in the `SearchProduct` activity.

Sequential Path³⁶ is a more general case of a decision branch. A sequential path consists of a contiguous sequence of model elements that is annotated with a common PC. The conditions on the start and end points of the sequence are more relaxed than those in a decision node branch; the points can be any two distinct model elements. The sequential paths found in the model were either a subset of model elements within a branch, such as the product variants feature in the `FindProducts` activity, or a path that includes a decision or merge node within it, such as the quick checkout feature in the `CheckoutItems` activity. None of the sequential paths in the models were annotated with features that modeled entities.

The characteristics of the decision branches, decision branch bypasses and sequential paths are almost identical; however, the limited number of occurrences of the latter two prevents any definite conclusions about the usefulness of those annotation categories from being made. Therefore, these annotation categories will be kept for future study.

³⁵ This implies that the flow is also the incoming flow to the first decision or merge node encountered.

³⁶ Sequential path is also referred to as “sequence” for brevity in certain situations.

Another interesting observation is that there are many types of model elements in activity diagrams which are rarely annotated. They include decision and merge nodes, parallel branches, and fork and join nodes. This is most likely due to the fact that they are often adequately covered by their IPCs.

7.3 Candidates for Modeling Guidelines

This section focuses on deriving some candidates for modeling guidelines³⁷. Guidelines are derived for both feature modeling and model template design. These guidelines are based on the experiences during modeling, as well as observations from the previous sections; therefore, some of the guidelines make reference to earlier versions of models. In some cases, multiple modeling approaches are presented and the trade-offs for each approach are discussed.

7.3.1 Applying Binding Time Analysis to Feature Modeling

As mentioned in section 7.1, an understanding of the binding times of individual features and the decision making process are essential for accurately capturing a set of requirements in a feature model. The structure of the feature model can be significantly impacted by the intent of when a feature is meant to be configured and the ability to use multi-staged configuration. The best strategy for modeling the decision structure is to apply a subfeature relationship for each subsequent set of decisions. In other words, features with later binding times can be placed at a lower depth in the model. There are three reasons for this: 1) it clearly separates the levels of decision making; 2) it defines an order in which decisions should be made without forcing it upon the user³⁸; and 3) it allows decisions with later binding times to be hidden by preventing the expansion of a node, thus allowing the modeler to decide if certain decisions should be made unavailable until a later point in the configuration. In addition, this structure, coupled with the branch hiding capability, can allow different levels of feature models to be generated systematically by interpreting a binding time attribute³⁹ and pruning branches as necessary. In contrast, a non-hierarchical structure would most likely require a manual definition for each level or a model transformation.

³⁷ From this point forward, “candidates for modeling guidelines” are referred to as “guidelines” for brevity.

³⁸ Due to constraint propagation in a configuration, selection of a late-binding decision will automatically cause all previous decisions to be selected.

³⁹ This would require a metamodel extension to add a binding time attribute to all nodes. The attribute could be an integer value that is interpreted relative to other features in the same branch or across the entire feature model as the level in which a feature would appear in the model.

In addition, one should also take into consideration whether the binding time for a decision is build-time or run-time. Run-time policies, in particular, often provide a set of options that have mutually exclusive meanings. These policies tend to be captured in feature groups as individual grouped features. Some of the grouped features can represent opposing policies. For example, the registration dependency feature shown in figure 7.2(b) contains a “requires registration” policy and a “does not require registration” policy. These policies are mutually exclusive when the e-shop administrator selects a policy at run-time, either during or after deployment of the e-shop; however, prior to that, the inclusive-or relationship is necessary because the features are being scoped for the e-shop.

Finally, the modeling of feature groups also varies depending on the binding time. Experience suggests that feature models which are intended for use in multi-staged configuration should strive to capture the widest variability possible at an early binding time; therefore, feature groups should be modeled with the relationship which can select the largest number of grouped features from the set, meaning an inclusive-or relationship. This is ideal because inclusive-or expresses the largest amount of variability. The feature group can be refined in subsequent stages to select a smaller group of features⁴⁰, thus constraining the variability in future stages. At later binding times, the feature group relationship depends on the domain. In some cases, the feature group is refined into an exclusive-or relationship, which is the case for the award feature described in section 5.2.2.2. On the other hand, some feature groups retain an inclusive-or relationship for all binding times, which is the case for the taxation rules feature described in section 4.5.2.3.

7.3.2 Modeling Feature Groups in Model Templates

In the e-commerce model, feature groups are often used to model types, such as payment or product types. This section describes the methods for translating a feature group into a set of elements and annotations in a model template.

Since feature groups cannot be named directly⁴¹, it is a good practice to introduce a solitary feature to contain the feature group and use the solitary feature for describing the relationship

⁴⁰ This can be achieved by refining the group cardinality or by selecting and eliminating grouped features.

⁴¹ The feature id is not displayed directly on the model; it must be accessed via the properties. Furthermore, it is a unique identifier over the scope of the entire feature model, so two types which are the same but under different features would have different identifiers.

between the grouped features. Under this organizational structure, the solitary feature is the group type and the grouped features are the type values. Feature groups are modeled differently depending on the model type and the way in which the feature group is used.

7.3.2.1 Modeling Feature Groups in Class Diagram Model Templates

In a class diagram, feature groups can be modeled through an inheritance hierarchy or an enumeration. An inheritance hierarchy uses generalization relationships to express the grouping construct. In a strict form of an inheritance hierarchy, the type is represented by the superclass and the type values are represented by the subclasses. Following the convention described above, the superclass corresponds with the solitary feature and the subclasses correspond to the grouped features. The superclass can be abstract or concrete depending on the implementation requirements, but they are usually abstract. Examples include the Product and PaymentInformation classes in the store front entity model. In a relaxed form of an inheritance hierarchy, only a subset of the elements from the strict form is required. A relaxed form is used when the modeler decides that it does not make sense to represent every grouped feature in the feature group as a subclass. One example of this would be when there are two grouped features and only one has a concrete class representation; if the solitary feature is only used for grouping purposes, there is no point of modeling it as a superclass⁴². An example would be the taxation feature, which has two grouped features: custom taxation and tax gateway. In the entity model, only the custom taxation feature is represented as a subclass; the tax gateway is modeled as a service. Therefore, there is no concrete superclass representation of a taxation entity either. In both forms of an inheritance hierarchy, the solitary feature and the grouped features can each be traced to their respective class, provided that there is a concrete representation of the feature.

With the enumeration representation, the type is represented by the enumeration and the type values are represented by enumeration values. Following the convention described above, the enumeration corresponds to the solitary feature and the enumeration values correspond to the grouped features. Each enumeration value is annotated with its corresponding grouped feature and the enumeration can be annotated with the solitary feature to ensure that there are no empty

⁴² Although one can argue that the result of this relaxed form is a seemingly ad-hoc solution, the inheritance hierarchy provides a framework for constructing the solution. Furthermore, in a scenario with multiple levels of feature groups, some parts may use a strict form and others may use a relaxed form of the inheritance hierarchy. This gives the modeler the ability to choose the most succinct representation rather than enforcing a scheme that would add unnecessary elements to the model.

enumerations generated in the template instance. In addition, attributes of the enumeration type should also be annotated with the solitary feature to ensure that the attributes are removed if the enumeration is removed. The MediaTypes enumeration in the store front entity model is a good example of this. The enumeration is annotated with the media files feature and the enumeration types are annotated with the corresponding media file types, thus mirroring the structure of the media files feature.

The key difference between the two types of representation is that the inheritance hierarchy is useful for capturing additional attributes or semantics which are associated with the type or type values, whereas the enumeration is better suited if the type value is used primarily to influence processing behaviour. Other considerations for choosing between the two include the readability of the model template, preference of the modeler, and implementation concerns.

7.3.2.2 Modeling Feature Groups in Activity Diagram Model Templates

In an activity diagram, a feature group can be modeled using a decision structure, a set of parallel branches, or a sequential set of actions. The choice of the modeling technique depends on the semantics of the feature group. The most common modeling technique is the decision structure, which is used to model a feature group when it allows for the selection of multiple grouped features at build time, but forms a set of mutually exclusive alternatives at run-time⁴³. The decision node's outgoing flows have guards which represent the different type values. In this technique, a branch represents a set of actions which correspond to the type value. The decision node can be annotated with the solitary feature and all of the elements in a branch are annotated with the corresponding grouped feature. If the type value is eliminated, it results in the removal of any model elements which provide a handling mechanism for the type value. Theoretically, the type value should never appear at run-time, but if it did, the activity would immediately exit at the decision node because none of the guard conditions would be satisfied. One must be careful to annotate all of the model elements in the branch; if the flows are not annotated, the removal of the nodes elements could result in flow closure and create a bypass flow. This means that if the type value is encountered, the activity will continue to execute but there will be no actions associated with it. This may result in erratic behaviour that may be difficult to debug. One example of

⁴³ This is not a strict condition. If multiple alternatives can be applied in parallel, then parallel branches is the most appropriate technique. If multiple alternatives from the set can be applied sequentially, a loop to the decision node can be added to allow for multiple iterations; however, the scenario described in the main body text is the most common application.

applying the decision structure modeling technique can be seen in the SearchProduct activity. The first decision node in the activity assesses the search type and there are separate branches for dealing with basic and advanced searches.

A set of parallel branches can be used to model a feature group when multiple grouped features can be selected at run-time and there is no sequential order imposed on them. In this technique, each parallel branch is annotated with a feature that represents a type value; therefore, all model elements in a branch are annotated with a corresponding grouped feature. The same precautions for annotations, as described for the decision structure, should be observed; however, in the case of the parallel branches, ensuring that the annotation is made on every element in the branch is a matter of good form. This is because a bypass flow in a parallel flow would have no side effects unless it is the only flow. There are no occurrences of this in the e-commerce model because parallel flows are not used in any of the models templates.

A sequential set of actions can be used to model a feature group when more than one grouped feature can be selected at run-time and their order of execution matters. In this technique, each action corresponds to a grouped feature and is annotated as such. Therefore, any grouped feature that is eliminated will simply remove the corresponding action from the sequence. Furthermore, it does not matter how many features are selected since the resulting sequence will always be valid due to flow closure. One can distinguish between a strict interpretation, in which every grouped feature must correspond to a single action in the sequence, and a relaxed interpretation, in which a subset of the grouped features correspond to a subset of actions in the sequence. Although there are sequences of actions which are annotated with different features in the model, there are no examples of this particular technique. An example which comes close can be seen in the second phase of the CheckoutItems activity. There is a sequence of actions corresponding to the calculation of the shipping costs, discounts, and tax. These actions form a sequential path and the order in which they execute affects the outcome; however, the three corresponding features, shipping, discounts and taxation options, do not share a common feature group.

7.3.3 Applying MetaExpressions to Increase Conciseness

The ME annotations are a useful mechanism for increasing conciseness in model templates. Many situations which would normally require multiple, mutually exclusive model elements in order to express the variability can be simplified. Simplification is achieved by annotating the element

with a ME and specifying an expression to calculate the required value based on a set of features. This section illustrates the application of the MultiplicityME to simplify a class diagram model template.

In class diagram model templates, there are many examples of where multiplicity depends on the presence of a feature. One example can be seen in the association between the Category and Product classes with respect to the multiple classification feature. In this example, selection of the multiple classification feature changes the multiplicity on the association at the category end from “1” to “1..*”. In an earlier version of the model, as shown in figure 7.3, this was modeled as two associations between the classes, each with a different multiplicity on the category end. These two associations were annotated with mutually exclusive PCs; the association with “1” was annotated with “!MultipleClassification” and the association with “1..*” was annotated with “MultipleClassification”. Aside from that, everything else was identical⁴⁴. The problem with this approach is that the extra elements used to model alternatives tend to add clutter to the model template, thus making the template difficult to read or maintain.



Figure 7.3: Using Multiple Associations to Model Multiplicity Variability⁴⁵

This problem can be addressed with the MultiplicityME, which is used to calculate the multiplicity on an association end. In the thesis version of the model shown in figure 7.4, only one association is required. The variable multiplicity is annotated with the MultiplicityME, which

⁴⁴ The association classifiers are actually different, but that is done by the tool to prevent ambiguity when navigating the model. This difference is mainly for the tool because it does not interpret the associations as mutually exclusive elements. The use of the MultiplicityME solves this problem implicitly.

⁴⁵ This figure has been modified to highlight the associations of interest only.

is given the following value: *if* ($\backslash\backslash multipleClassification$) *then* "1..*" *else* "1". $multipleClassification$ is the identifier for the multiple classification feature and serves as a Boolean variable in the expression. This expression is evaluated and the corresponding value is replaced during template instantiation.

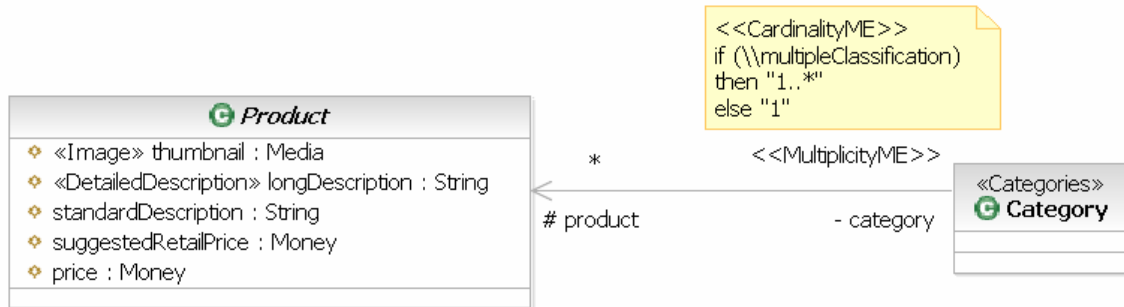


Figure 7.4: Using Multiplicity MetaExpression to Model Multiplicity Variability⁴⁶

7.4 Recommendations

This section focuses on presenting a set of recommendations to improve the process and development tool implementation. The recommendations focus on usability issues, primarily to reduce the amount of effort required for template development and the occurrence of errors during annotation.

7.4.1 Variability in the Ordering of Features and Model Elements

One area that has not been addressed in model templates is accounting for variability in the ordering of nodes in activity diagram model templates. An ordering defines a sequence of model elements; variability in ordering arises when one requires actions to be run in different orders. An example in the model where ordering is important is in the second phase of the CheckoutItems activity. In that phase, it can be seen that shipping costs are calculated before applying discounts and assessing taxes. This implies that the shipping costs calculations cannot take discounts and taxes into consideration. An e-shop may want to create a policy which applies the discounts before calculating the shipping costs, especially if the shipping costs vary based on how much the customer spends. This would require the order of the actions to be changed. In addition, one must consider that not every single permutation of the ordering may be valid.

⁴⁶ The stereotype on the association end is used for visualization purposes only because the tool does not support the display of metaexpressions on the diagrams directly. In addition, this figure has been modified to highlight the association of interest only.

There are two major obstacles for dealing with ordering in model templates. They lie with the representation of order in feature models and activity diagrams respectively. There is no pre-defined attribute or model element to represent ordering in the feature model metamodel supplied by the development tool. The modeler can extend the metamodel to include an attribute to represent ordering; however, it is unclear what value can be assigned to capture the ordering between features in general. Features that require ordering do not necessarily reside in the same feature group; therefore, any ordering mechanism must be able to relate features globally within the feature model. Furthermore, there must be some way to define a set of constraints on the ordering. In activity diagrams, there are no existing annotations or constructs which can be used to model the variability in ordering either.

Although there is only one example of a sequential set of elements in the model, many problems, such as multiple discount handling or policy enforcement, could benefit from a solution to the ordering problem. The problems and obstacles associated with ordering are non-trivial. Insofar, no method has been found to satisfactorily and intuitively represent the ordering of features; however, such a method would be useful for both feature models and model templates. Therefore, it is recommended that further investigation on the ordering problem, in the context of feature models and model templates, be pursued.

7.4.2 Additional Semantics for UML Class Diagram Model Templates

[Cza05f] already specifies a set of semantics for class diagrams which include IPCs for generalization, dependency association, classes, and simplification rules for generalization chain closure and containment chain closure⁴⁷. In this section, additional semantics are proposed for class diagrams.

The technique for modeling a feature group with enumerations is described in section 7.3.2. The description indicates that the enumeration should be annotated with the solitary feature in order to remove an empty enumeration; however, if the guidelines are followed, the enumeration will only contain values which are annotated with group features. Therefore, if the solitary feature is

⁴⁷ These rules are analogous to flow closure in activity diagrams.

selected, at least one of the grouped features must also be selected⁴⁸, which makes the annotation on the enumeration redundant. This situation can be handled by defining a simplification rule for enumerations; an empty enumeration can be determined by calculating the disjunction of the PCs for all contained enumeration values. A stronger condition can be added such that the presence of an enumeration is also dependent on whether or not any attributes are of the enumeration type. This can be achieved by modifying the simplification rule so that it calculates a conjunction of the previously defined rule with a disjunction of the PCs of attributes of the enumeration type over the entire diagram. This is essentially performing garbage collection on unreferenced enumerations.

Abstract classes are only useful if there is a concrete class which inherits from the abstract class or if it contains additional attributes or relationships with other elements. The simplification rule for an abstract class can be defined as a disjunction of two expressions: 1) a disjunction of the PCs of all specializations, where the abstract class is the general classifier, and 2) a disjunction of the PCs of all elements associated with or contained by the abstract class⁴⁹. It may be argued that having a non-empty abstract class without any implementation is useless; however, it is assumed that implementation classes can be dynamically linked at run-time. This means that the abstract class can still exist in the template instance even without a concrete class. The same line of reasoning applies to interfaces, as seen through the service diagram; therefore, an analogous simplification rule can be defined for interfaces.

7.4.3 Additional Semantics for UML Activity Diagram Model Templates

Since the focus of [Cza05f] was to develop semantics for activity diagrams, the set of IPCs, patch transformations and simplifications are already fairly extensive; however, the analysis in section 7.2.2 indicates that there are some common annotation categories which may be leveraged to reduce the annotation effort and / or improve the model template. Due to the length of some of these proposed additions, each recommendation is accorded a separate subsection.

⁴⁸ This holds as long as the group cardinality has a lower bound of 1, which is the default; however, a scenario where the group cardinality has a lower bound of 0 can be normalized such that the solitary feature which contains the feature group is made optional and the group cardinality lower bound is set to 1. Both produce equivalent outcomes during configuration.

⁴⁹ The model elements would include member variables, associations, and composite aggregations.

7.4.3.1 IPCs for Data Stores and Generic Data Nodes

Data stores are a stereotyped data node in which data can be retrieved from or stored to; however, in the case where all adjacent flows are eliminated, the data store becomes redundant. Therefore, an IPC for data stores can be defined as the disjunction of the PCs of all incoming and outgoing flows. The difference between the data store and the central buffer is that a data store is required as long as one flow is present, whereas a central buffer requires both flows since it serves as an intermediary for actions with object flows. Data stores and central buffers are just stereotypes on a generic data node, which is available as a model element in the development tool; however, it is difficult to define an IPC for the generic data node since its semantics are not well-defined.

7.4.3.2 Branch Annotations Semantics

The analysis of the activity diagram annotations indicates that there are a significant number of features whose semantics require an entire branch of a decision node to be annotated. The branch is inclusive of all model elements from the outgoing flow of the decision node to the incoming flow of the merge node. In many cases, these branches contain a single entry point and a single exit point. There are two approaches which can be used to model this. In the first approach, all of the model elements in the branch are annotated with the set of features. For longer branches, this causes a significant increase in the number of annotations required. Furthermore, the larger number of annotations increases the likelihood of an annotation error, either due to missing an element or forgetting to annotate the first or the last flow in the branch. In the second approach, all of the model elements in the branch are extracted into a separate activity and the branch is replaced with a call behaviour action. The activity is annotated with the set of features, thus causing an implicit annotation on all of the model elements in the activity. This approach will require fewer annotations, but the re-factoring of the actions may not be desirable due to the fact that the information will be hidden.

These findings suggest that a special type of annotation, based on the annotation categories described in section 7.2.2, may be useful when working with activity diagrams. One possible solution is to define a special stereotype for decision branches. Like regular annotations, the stereotype is created based on a set of features. It is used to annotate the first element in the branch; the last element in the branch, which is the incoming flow to the first decision or merge node that is encountered, is found automatically. Any model elements between the first and last

elements would be automatically annotated with the set of features. For cases with simple branching, such as the advanced search feature in the SearchProduct activity, this would be a sufficient mechanism.

A more general solution would be to define a special stereotype for sequential paths. The idea here is the same as the idea in the previous solution, except that there is a pair of annotations for the first and last elements in the path. This solution would be able to handle a more general case, such as the quick checkout feature whose path includes a decision node in the CheckoutItems activity. To simplify the semantics of the annotation, every element along every valid path between the first and last elements will be automatically annotated with the set of features.

Both solutions reduce the total number of annotations required, but the addition of another form of stereotype adds complexity to the annotation language; however, experience with the e-commerce model suggests that it would be a valuable mechanism for the annotator.

7.4.3.3 Automatic Flow Type Correction

A patching transformation which is missing from the activity diagram model templates is automatic flow type correction, which is useful when flow closures fail due to mismatched flow types. The problem occurs when regions have a mixture of control and object flows, where the entry flow is a control flow and the exit flow is an object flow or vice-versa. If the entry and exit flow types differ, the flow will not be closed. An example of this can be seen in the SendWishList activity. The region with the multiple wish list annotations has a control flow as the entry point and an object flow as the exit point⁵⁰. If the multiple wish list feature is eliminated, the flow cannot be closed. The current workaround is to add a control flow between the SelectWishListFromList and RetrieveWishList actions. This control flow is present if the multiple wish list feature is eliminated, thus allowing the flow to be closed.

The proposed solution is to create a patch transformation which checks for all of the conditions on each flow to determine its correct type after the rest of the patching transformations have been run. The time at which it is run is important since the template instance must be valid before

⁵⁰ The entry flow starts from the initial node as a control flow and enters the region. The exit flow is the incoming object flow to the RetrieveWishList activity. Since the RetrieveWishList activity will expect an object flow, there is a mismatch.

analyzing the flow type; otherwise, the development tool will have to determine the flow types while handling disconnected flows and other anomalies. Once the flow type is corrected, the effect must be propagated to adjoining elements. For example, an adjacent call behaviour action may require the removal of its data pin⁵¹ to join a control flow to the call behaviour action.

7.4.4 Annotation Consistency Issues

Maintaining annotation consistency is one of the most difficult aspects of the feature-based model template development process. The annotator must have a very thorough understanding of both the feature model and the model templates, as well as an understanding of the potential impact of each annotation. Annotation inconsistencies are caused by a set of annotations on a model template that, when instantiated with a valid configuration, results in a semantically-incorrect or ill-formed template instance. A semantically-incorrect template instance violates semantic properties defined by the modeler or exhibits semantic errors with respect to the modeler's original intent. An ill-formed template instance exhibits syntax errors with respect to the modeling language. Aside from a visual inspection of both the feature model and model templates, the only way to detect these errors is to use the fmp2rsmVerifier tool [Cza05g].

There are three major factors which contribute to annotation inconsistencies: annotation errors, hidden annotation effects, and unawareness of additional processing effects. Annotations errors are mistakes which are made by the annotator due to carelessness or a misunderstanding; it is analogous to making a syntax error with respect to the modeling language. For example, if the annotator is annotating a decision node branch and forgets to annotate a flow at the end of the branch, the omission is considered to be an annotation error. This omission in particular will result in an ill-formed template since there will be a dangling flow. In general, such an error can result in either type of inconsistency.

Hidden annotation effects are inconsistencies which arise when the effect of an implicit annotation overshadows the effect of an explicit annotation. The primary cause is annotations on

⁵¹ Two assumptions are made: 1) the removal of the central buffer is intended and the remainder of the transformations follow because of it. If the situation was reversed and the annotation was on the data pin, the analysis would still hold. If there is a conflict between the annotations on the central buffer and data pin(s), a mechanism would be needed to resolve it, 2) the IPC on the data pins respect the properties of the actions to which they are attached to. For example, if an input pin is required as an input parameter on a call behaviour and the adjacent central buffer is removed by a PC, the patching transformation will handle the situation correctly, e.g. signal an error has occurred.

container elements. Container elements can cause hidden annotation effects because the annotations are implicitly applied as a conjunction with all of the model elements inside the container. The problem with this is that the implicit annotations on the elements are not clearly visible in the model; therefore, it is possible that the annotator may be unaware when s/he is annotating an element. There are two potential problems with the annotation on the contained element:

- 1) it may conflict with the annotation on the container element. In a conflict, the two annotations have mutually exclusive PCs and there are two possible outcomes: a) if the container element is present, then the contained element is not, or b) if the container element is removed, the PC of the contained element will never be evaluated so it does not matter. Either way, it means that the contained element will never be present, which means that the model is semantically incorrect⁵².
- 2) the annotation is redundant. There are two possible scenarios: a) the contained element is annotated with a PC which is inferred by the PC of the container element, or b) the container element is annotated with a PC which is inferred by the PC of the contained element. In the first scenario, the contained element will always be present. In the second scenario, the annotation means that it is impossible to have the contained element without the container element; however, this is always true by the containment property. In both cases, the annotator may not expect this result if s/he believes that the presence of the model element is being controlled solely by the selection of the feature. The overriding effect of the container element may lead to a semantic error if the annotator is unaware of the container element annotation or its relationship to the annotation on the contained element. Even if there are no semantic errors, it still results in additional annotations.

In general, the best solution to the hidden annotation effects problem is to enable tool support for propagating and visualizing implicit annotations on all model elements. The implicit annotations would be based on what is annotated on higher-level elements; any constraints from the feature model will still remain hidden and must still be taken into account by the annotator.

Additional processing can result in annotation inconsistencies if the annotator is unaware of the effects of automatic transformations, such as simplifications. Current additional processing steps

⁵² The inclusion of an element in the model indicates that the modeler expects it will be present under some condition.

only handle unambiguous redundancy, such as removing decision nodes which have a single outflow. There are cases where the redundancy is not clear, such as removing empty or unreferenced enumerations as recommended in section 7.4.2. These cases can be handled by explicit annotations, but their common occurrence suggests that adding additional processing could improve the usability of the development tool. The problem occurs when the annotator is unsure about which additional processing rules are available and makes assumptions about them. Such assumptions can possibly lead to redundant or omitted annotations. For example, if a class is contained by another class through composite aggregation, it is unclear what the effect on the contained class should be if the container class is removed. Run-time semantics state that the lifetime of the contained object is dependent on the container object; however, there is no such restriction at build-time. The correct outcome is determined by the intent of the modeler; is the composite aggregation intended to denote variability in the structure or is the modeler attempting to describe run-time semantics? Without any explicit annotations, the removal of the container class will cause the composite aggregation to be removed, but the contained class to remain. Therefore, the tool supports build-time semantics by default. In order to have the contained class removed also, the contained class must be annotated with the same PC as the container class. A few examples in the model suggest that the run-time semantics may be more common. For example, the Wishlist class contains the WishlistItem class. Since the wish list is composed of several wish list items, the contained class should be removed if the container class is removed; however, there are an insufficient number of occurrences to determine if this is the definitive case. On the other hand, it should also be noted that no cases of the build-time semantics were observed in the models. It is these unclear instances of redundancy which can cause the annotator to make mistakes which lead to inconsistent results during model template instantiation. There are a few possible solutions to this problem:

- 1) Restrict simplifications to unambiguous redundancies only; the unclear cases can be handled by explicit annotations. This is a safe approach, but the additional annotations can reduce the usability of the tool under certain circumstances.
- 2) Implement simplifications based on frequently occurring scenarios. This approach results in a reduction of effort by essentially automating processing on the most common cases. The other cases, however, will require explicit annotations⁵³. The assumption is that the

⁵³ The explicit annotations may be non-intuitive. For example, if contained elements were automatically removed based on the container element, the only way to override this behaviour would be to annotate the container class and the contained class with mutually exclusive PCs.

occurrence of the most common case will greatly exceed the other cases, leading to a net reduction in annotation effort.

- 3) Implement a set of additional processing rules which can be selected and configured by the annotator. For example, the annotator can choose if they want containment chain closure to be applied during template instantiation. This may be implemented as a special stereotype to be annotated on certain elements or as a set of global options that control the application of additional processing steps during template instantiation. This approach is a compromise since it leaves the decision up to the annotator; it also implicitly provides a documentation set for the additional processing steps.
- 4) Provide better documentation for the additional processing steps to allow the annotator to understand what their effects and limitations are.

Finally, two more general solutions are recommended to help reduce the problems with annotation consistency. The first solution is to provide more computer-aided assistance in the form of wizards, pattern applications, and auto-completion mechanisms for annotations. These tools can capture best practices and common scenarios so that the user can focus on using the tool to model the domain, as opposed to learning all of the tool subtleties. The second solution is to provide quick previews of template instances. It would consist of a configuration view placed adjacent to a model template view. It allows the user to experiment with different configurations to observe the effects of the annotations. Ideally, the preview is updated in real-time as the configuration is modified and the corresponding effects on multiple model templates can be viewed simultaneously.

The annotation consistency problem is a good representation of one of the key challenges in the design of the feature-based model template approach. The challenge is to decide how much automation to implement in the tool for additional processing. There are two extremes: 1) minimize the amount of explicit annotations and allow the additional processing to handle the rest, or 2) explicitly annotate everything. In the first case, the main benefit is that fewer explicit annotations lead to reduced effort for the modeler and a lower probability for annotation error. The IPCs tend to be adequate for guaranteeing the well-formedness of a template instance in the absence of annotation errors, hence the chance for syntax problems is greatly reduced. On the other hand, the reliance on the additional processing steps can introduce unintended side-effects, which may lead to semantic errors in the model. The second case is the exact opposite due to the increased number of explicit annotations. Syntax errors will become more likely since the

annotator will have to take into account all valid configurations; on the other hand, the explicit annotations are a clear declaration of intent by the annotator, so semantic errors are less likely. In the end, it is the position adopted by the users which factors heavily into deciding what additional processing steps should be supported by the development tool and how the support should be implemented.

7.5 Summary

In this chapter, an analysis and evaluation of the feature-based model templates approach was presented. The chapter began with an analysis of variability modeling in feature models, which demonstrated the importance of binding time analysis when translating requirements into feature models. This was followed by an analysis of the annotations in the model templates and the discovery of annotation categories; the annotation categories illustrate certain patterns which may be leveraged in the future to improve the development of model templates. Based on the experiences from the e-commerce model and the analysis in this chapter, several candidates for modeling guidelines and recommendations for process and tool improvement were proposed.

8 Conclusion

The pervasiveness and criticality of software applications in modern times has increased the demand for highly customized, high quality, industrial-strength software with great complexity. The large number of software project failures, as reported by various studies, suggests that traditional approaches to software development lack the capability needed to deal with this software complexity. The MDSPL approach is touted to address some of the issues raised by software complexity, such as reducing impedance mismatch through domain analysis, using models to better characterize discrete states, and enabling behaviour tracing and validation through model simulation [Bru95]. The initial cost of building reusable assets during DE may be amortized over the increased speed at which individual products may be created and the higher quality of product that may be produced in AE. The novelty of feature-based model templates is the management of the variability through a separate feature model, coupled with the super-imposed variants which allow template instantiation to be achieved by removing elements and post-processing transformations.

This thesis presented an example of an e-commerce solution which was used to evaluate the MDSPL approach, as well as an analysis of the development process. The goal was to demonstrate the viability of the approach on an example inspired by a realistic application, as well as to use the workflows in the example to determine characteristics of the process and the tool which could be improved. This chapter presents a summary of the research findings, a discussion on areas for future work, and a concluding remark.

8.1 Summary

The example that was created and described in the thesis is based on a subset of the e-commerce domain, specifically B2C e-shops which sell goods or services using a fixed purchase price. The domain analysis resulted in a feature model and a set of model templates.

The feature model consisted of over 200 features which were divided into two main categories: store front and business management. Store front features dealt with web site features, many of which were accessible by the customer and impacted their shopping experience directly. Business management features dealt with back-office operations and business policies.

The model templates consisted of two class diagrams and seventeen activity diagrams. The two class diagrams were the store front entity model and the service model. The store front entity model consisted of regular class diagram elements, such as classes, attributes, enumerations, associations and composite aggregations, and contained entities which represented the products offered in the e-shop and related to a customer's shopping experience. The service model depicted the system from a higher-level of abstraction, using stereotypes to represent components and services. The model described the relationship between the store front and the other stakeholders.

The activity diagrams depicted workflows which related primarily to the store front area of the e-shop, including site browsing, registration, customer administration tasks, wish list management, item purchase, and order processing. The set of activities included in the model is sufficient to allow a customer to shop and make a purchase at the e-shop, but it should not be considered a complete or definitive set of all possible workflows in an e-shop. The amount of annotations varied depending on the activity and the process of building the model templates resulted in some new features being discovered.

The evaluation of the development process using the e-commerce example was divided up into multiple sections. The first section described an example of extending the feature model and illustrating the importance of taking binding time and the decision making process into consideration when modeling a feature.

The second section described the results of the annotation analysis, which examined all of the annotations in the model templates to discover common annotation patterns specific to the model type. For class diagrams, adjacent model elements tend to be annotated with different features. For the entity model, annotations were made on almost all types of model elements, such as classes, attributes, enumerations and associations. Conversely for the service model, annotations were made on the stereotyped components, services and interfaces only; dependency and implementation relationships were never annotated. For activity diagrams, a combination of individual elements, such as actions and data nodes, and sets of elements, such as decision branches and sequential paths, were annotated.

The third section presented a series of guidelines for the development process based on the experience with the e-commerce example. The guidelines are summarized as follows:

- Subsequent sets of decisions should be modeled using the subfeature relationship, where features with later binding times should be placed at lower depths. In the general case, the subfeature relationship is used to decompose features into greater detail.
- The widest group relationship, inclusive-or, should be used at an early binding time unless it is restricted by the domain.
- The modeling of feature groups is dependent on the context. In class diagrams, feature groups can be modeled as an inheritance hierarchy, if additional information needs to be attached the type value, or enumeration values, if the type value is used solely for controlling process behaviour. In activity diagrams, feature groups depend on the number of features to be selected at build time and run time, as well as if the features have to be ordered.
 - A decision node can be used if there are multiple features selected at build time but only a single feature selected at run-time.
 - A parallel branch can be used if there are multiple features selected at run-time but there is no ordering.
 - A sequential set of actions can be used if multiple features are selected at run-time and the features are ordered.
- Metaexpressions can be used to express variability with a single model element instead of using multiple model elements with multiple presence conditions.

The fourth section presented a series of recommendations, which can be summarized as follows:

- The ability to express ordering in feature models and activity diagram model templates would be a useful feature; it is recommended that this be investigated further.
- Adding model template semantics for class diagrams which include simplification rules for enumerations, abstract classes and interfaces.
- Adding model template semantics for activity diagrams which include an IPC definition for data stores, additional stereotypes for the frequently occurring decision branch and sequential path annotation patterns.
- Handling hidden knowledge issues by enabling the propagation and visualization of implicit annotations.
- Handling unexpected behaviour resulting from additional processing steps by either doing nothing, implementing additional processing steps for the most frequently occurring

annotation patterns only, allowing the user to configure the set of additional processing steps to apply, or improving the documentation for additional processing steps.

- Addition of a computer-assisted annotation feature and a quick preview mechanism for testing annotated model templates.

8.2 Future Work

There are many opportunities to build upon the work described in the thesis. An immediate follow-up to this work is to implement the recommendations presented, re-annotate the model templates, and perform the annotation analysis again. The evaluation can be used to gauge the effectiveness of the recommendations.

In terms of the example and the analysis, there are many dimensions which it can be extended along. One possibility is to extend the features of the e-shop; the activity diagram model templates suggest a series of features which can be used to extend the feature model. Afterwards, new areas of the e-shop can be explored, such as the warehouse or supply chain capabilities. Furthermore, non-functional requirements, such as performance and scalability can also be considered. A second possibility is to pick another type of e-commerce site, such as a B2B site or an auction-based service, perform domain analysis, and extend the feature model to include related features. The interest will be in the configuration process, which will most likely to be multi-level since the first level would configure a type of site and the second level would be used to configure a specific e-commerce product. This would be a useful example to test the effectiveness of multi-level configuration on a realistic example. A third possibility is to create an extensive example for another domain, such as Voice over Internet Protocol (VoIP) telephony systems. The same analysis would be performed and the results could be compared to the e-commerce example. Finally, one could also extend the analysis methodology. In the annotation analysis, annotation categories are only studied from the perspective of the model elements. In addition, there was little focus on classifying the features in the analysis; a more detailed classification for the features used by the annotations could be useful in gaining a better understanding of the annotation categories.

In terms of GSD, there are many other areas to be explored. One direction is to examine the effect of a multi-stakeholder scenario, where specific decisions can be made by different stakeholders at different times. This is an extremely complex process since it involves categorizing features

based on stakeholders, designing a framework to describe the role of stakeholders, determining how one stakeholders' feature can affect another and how to resolve conflicts should they occur during the configuration process. The complexity of the problem increases even further if it is assumed that any stakeholder can configure the feature model at anytime and that the process occurs in a distributed environment. This scenario is referred to as a distributed configuration.

Another direction is based on the current trend in the research group, which is to adopt ontologies for modeling the relationships in a domain. The domain analysis performed in the example can be applied to the creation of an ontology. One possibility is to use ontologies as the primary representation of a domain and to generate feature models as views on the ontology. This idea is currently being explored by the research group.

The final direction proposed for future work is to focus on the code generation aspect of the process. The template instances represent valid models for the concrete product; however, a mapping between the models and the implementation code is still needed. The implementation will need to be in the form of a reusable asset, like a code template. This particular feature may allow the full scope of the MDSPL approach to be realized. Automated implementation generation tends to be one of the most requested features for the tool.

8.3 Closing Remarks

History has shown that the most successful paradigms in software engineering, such as object-oriented design, can influence all aspects of the field and change the way we think about software. GSD is a new paradigm in the field of software engineering. The success of this research can be gauged by the influence it has on other software researchers and practitioners, and their willingness to try it out for themselves in academia and industry.

Appendix A Additional Constraints in the E-Shop Feature Model

This appendix describes the additional constraints that are present in the domain. These additional constraints are based on the domain analysis from chapters 4 and 5. Some of the relationships are explicitly stated in the body, while others were implied.

Additional constraints define relationships between features which are not expressed by the structure of the model. Constraints typically take the form of an implication. There are two types of additional constraints: *strong* constraints and *weak* constraints. It is interesting to note that very few feature relationships in the e-shop model are bidirectional. In most cases, the relationship was different depending on the direction.

A strong constraint takes the form of A *requires* B or A *excludes* B. In the requires case, the selection of A implies that B must also be selected. In the excludes case, the selection of A implies that B must be eliminated. Strong constraints must be obeyed because one feature depends on the other feature for its definition or operation. In other words, one feature will not work without the other.

A weak constraint takes the form of A *recommends* B or A *discourages* B. In the recommends case, if A is selected, then B should also be selected because the only purpose of selecting A is to facilitate B; otherwise, A is meaningless. In the discourages case, if A is selected, then B should be eliminated because there is no purpose of having B if A is selected.

Another interesting relationship is *independence*, which takes the form A *is independent of* B. It states that the selection of A has no implication on the selection of B. There are many cases in the model where there is a strong relationship one way, but an independence relationship the other way. In these cases, it usually means that one of the features has a much wider scope of application than the other; for that reason, many features may depend on the feature with the wider scope.

Table A.1 describes the additional constraints in the e-shop feature model.

Table A.1: Additional Constraints

Feature(s) A	Feature(s) B	Relationship	Context
SpecialOffers (4.1.2)	Discounts (5.2.2.2)	A requires B	Special offers consist of a series of discounts.
		B is independent of A	Supporting discounts has no implication on if the e-shop decides to display special offers on the home page.
!RegistrationEnforcement (4.2.1) OR !RegisterToBuy (4.2.1)	RegisteredCheckout (4.5.2.1)	A excludes B	If either the registration enforcement feature or the registration to buy feature is eliminated, there is no way to force a customer to login to checkout; therefore, a registered checkout cannot be supported.
		B excludes A	If the registered checkout feature is selected, then both features in A must be selected to enable support for enforcing the login; the selection of register to buy implies the selection of registration enforcement through the subfeature relationship.
ShippingAddress (4.2.2)	Shipping (5.1.1.2)	A recommends B	The only purpose of storing a shipping address is if the e-shop offers shipping on its products; however, the shipping process does not require the address to be stored in the profile since it can be manually entered every time.
		B is independent of A	Supporting shipping has no implication on if the e-shop decides to store this information in the profile.
CreditCardInformation (4.2.2)	CreditCard (4.5.2.4)	A recommends B	The only purpose of storing credit card information in a profile is if the e-shop accepts credit card as a form of payment.
		B is independent of A	The decision to store credit card information in a profile is independent of accepting credit card as a payment method since the credit card information can be solicited during checkout.
SecurityInfo (4.2.2)	FraudDetection (4.5.2.4)	A recommends B	The only purpose of the security info is to be used in fraud detection and verification of the authenticity of payment.
		B is independent of A	Many fraud detection schemes require the security info on the credit card, but it does not have to be stored in the profile. In fact, storing it in the profile might be a bad idea.
Demographics (4.2.2)	Demographics (5.2.1)	A recommends B	The only purpose of the demographic data is to use it for promotional considerations; however, it can be collected without being used in the targeting purposes.
		B requires A	Using demographics information for targeting requires the information to be available; demographics is only stored in a customer profile.

Table A.1: Additional Constraints (continued)

Feature(s) A	Feature(s) B	Relationship	Context
PersonalInformation (4.2.2)	PersonalInformation (5.2.1)	A recommends B	The only purpose of the personal information is to use it for promotional considerations; however, it can be collected without being used in the targeting purposes.
		B requires A	Using personal information for targeting requires the information to be available; personal information is only stored in a customer profile.
Preferences (4.2.2)	CustomerPreferences (5.2.1)	A is independent of B	Preferences are used primarily by the customer to customize the interface. The decision to use the information for targeting is completely independent.
		B requires A	Using customer preferences for targeting requires the information to be available; preferences are only stored in a customer profile.
QuickCheckoutProfile (4.2.2)	QuickCheckout (4.5.2.1)	A recommends B	Although it is possible to store a quick checkout profile without offering a quick checkout option, the quick check out profile would serve no other purpose.
		B requires A	The profile is required during the quick checkout process.
UserBehaviourTracking-Information (4.2.3)	UserBehaviourTracking (4.7)	A requires B	Associating registration profile information to behavioural data only makes sense if the latter is collected.
		B is independent of A	The collection of customer behavioural data has no implication on whether or not it is associated to profile information.
ElectronicGoods (4.3.1.1)	ElectronicGoodsFulfillment (5.1.2)	A requires B	In order to offer and sell electronic goods, the corresponding fulfillment method is required.
		B recommends A	It is possible to support the infrastructure for carrying electronic goods without offering them, but there is no other purpose for the fulfillment feature.
PhysicalGoods (4.3.1.1)	PhysicalGoodsFulfillment (5.1.1)	A requires B	In order to offer and sell physical goods, the corresponding fulfillment method is required.
		B recommends A	It is possible to support the infrastructure for carrying physical goods without offering them, but there is no other purpose for the fulfillment feature.

Table A.1: Additional Constraints (continued)

Feature(s) A	Feature(s) B	Relationship	Context
Services (4.3.1.1)	ServicesFulfillment (5.1.3)	A requires B	In order to offer and sell services, the corresponding fulfillment method is required.
		B recommends A	It is possible to support the infrastructure for services without offering them, but there is no other purpose for the fulfillment feature.
PhysicalGoods (4.3.1.1) OR ElectronicGoods (4.3.1.1)	Size (4.3.1.8)	A requires B	Physical and electronic products require the size attribute as part of their description.
		B recommends A	Size is used to describe physical and electronic goods only.
PhysicalGoods (4.3.1.1)	Weight (4.3.1.9)	A requires B	Physical products require the weight attribute as part of the description for shipping purposes.
		B recommends A	Weight is used to describe physical goods only.
Availability (4.3.1.10)	InventoryTracking (5.4)	A requires B	Availability information is determined through the information provided by inventory tracking.
		B recommends A	One of the primary purposes of inventory tracking is to provide this data.
CategoryPage (4.3.5)	Categories (4.3.2)	A requires B	In order to support a page that lists categories, the e-shop must support categories
		B recommends A	It is very useful to have some mechanism for rendering the categories, but it is not absolutely essential.
WishList (4.4) OR !Registration (4.2)	WishListSavedAfterSession (4.4)	A requires B	The wish list must support either guests or registered users, although this is an indirect way of expressing this.
		B is partially independent of A (see relationship context)	Wish list saved after session implies the wish list feature because of subfeature relationship, but it is independent of registration.
EmailWishList (4.4)	Registration (4.2)	A requires B	E-mailing a wish list is a feature which is only available to registered users.
		B independent of A	The registration feature has no implication on whether or not customers can e-mail their wish list.
Permissions (4.4)	Registration (4.2)	A requires B	Since visitors can only view wish lists on the server and only registered users wish lists stored on servers, setting permissions on wish lists requires registered users with wish lists.
		B is independent of A	The registration feature has no implication on whether or not customers can set permissions on their wish list.

Table A.1: Additional Constraints (continued)

Feature(s) A	Feature(s) B	Relationship	Context
ShippingOptions (4.5.2.2)	Shipping (5.1.1.2)	A requires B	The selection of the shipping feature allows other options to be selected which are used for the shipping options.
		B is independent of A	Support for shipping is independent of allowing customers to make choices about the shipping options.
ProductReturns (4.6.2)	PhysicalGoods (4.3.1.1)	A requires B	The product returns feature only supports physical goods. In order to have the return functionality, there is also a dependency on a warehouse management system, which is implicitly provided by the physical goods feature.
		B is independent of A	An e-shop which offers physical products is not obligated to accept returns from customers.
OrderStatusViewing (4.6.3)	Registration (4.2)	A requires B	In order to retrieve orders for a customer, they must be registered and logged in.
		B is independent of A	The registration feature has no implication on whether or not customers can view their previous orders.
ShipmentStatusTracking (4.6.4)	Shipping (5.1.1.2)	A requires B	Tracking a shipment requires that the e-shop support shipping products.
		B is independent of A	The shipping feature has no implication on whether or not a customer can track their shipments.
WishListContent (5.2.1)	WishList (4.4)	A requires B	The use of wish list data for targeting purposes requires the e-shop to support wish lists.
		B is independent of A	The decision to support wish lists is made independently of using it for targeting.
PreviouslyVisitedPages (5.2.1)	LocallyVisitedPages (4.7.1) OR ExternalReferringPages (4.7.1)	A requires B	The use of previously visited pages for targeting purposes requires the e-shop to collect data about page traversals.
		B is independent of A	Collecting the page traversal data has no implication on how the data is used.
ShippingAddress (5.2.2.2)	Shipping (5.1.1.2)	A requires B	If the shipping address is used as an eligibility requirement for a discount, the system must support shipping.
		B is independent of A	Shipping products has no implication on whether or not it is used to determine discount eligibility.

Table A.1: Additional Constraints (continued)

Feature(s) A	Feature(s) B	Relationship	Context
Personalized (5.2.2.4)	Registration (4.2)	A requires B	Personalized e-mails require information from the registration profile.
		B is independent of A	Registration has no implication on if the profile data is used to personalize an e-mail.
InventoryTracking (5.4)	WarehouseManagement (5.1.1.1) AND FulfillmentSystem (5.7)	A requires B	The warehouse management system and fulfillment system provides the data that is required by inventory tracking.
		B is independent of A	Supporting the features has no implication on whether or not the store should perform inventory tracking.
Procurement (5.5)	ProcurementSystem (5.7)	A requires B	The information from the supplier is needed to make stock replenishment decisions.
		B recommends A	There is no reason to link to the suppliers' systems if it is not being used for re-ordering purposes.
Automatic (5.5)	InventoryTracking (5.4)	A requires B	The information provided by inventory tracking is the most up to date and linked with the warehouse, which the system requires if it has to make ordering decisions automatically.
		B is independent of A	The use of inventory tracking has no implication on whether or not stock is replenished automatically.
ReportingAndAnalysis (5.6)	UserBehaviourTracking (4.7)	A requires B	The reporting and analysis depends on the data collected through user behaviour tracking.
		B is independent of A	Tracking user behaviour has no implication on how the tracked data is used. (although could be recommended – used for: this, targeting criteria)

Appendix B Annotation Analysis Data

This appendix includes all of the raw data collected for the annotation analysis. The analysis consisted of assessing each annotation and determining how it was used. This data is analyzed and interpreted in section 7.2.

Tables B.1 and B.2 describe the annotations in the class diagrams. Adjacent model elements tend to have different annotations; therefore, annotations are identified by the individual model elements names. The context defines where the annotation can be found or how it is used.

Table B.1: Class Diagram (Store Front Entity Model) Annotation Analysis

Feature	Model Element (context)
AssociatedAssets	abstract class (Asset)
Availability	enumeration (Availability), attribute (in PhysicalGoods)
BillingAddress	association (PaymentInfo to Address)
CarrierSelection	attribute (in ShippingInstructions)
Categories	class (Category)
City	enumeration value (AddressResolutionTypes)
COD	class (subclass of PaymentInfo)
Country	enumeration value (AddressResolutionTypes)
CreditCard	enumeration (CreditCardTypes), class (PaymentInfo)
CustomTaxation	class (TaxRule)
DebitCard	class (subclass of PaymentInfo)
DetailedDescription	attribute (in Product)
Documents	class (Documents)
ElectronicCheque	class (subclass of PaymentInfo)
ElectronicGoods	class (ElectronicProduct, subclass of Product)
FaxMailOrder	class (subclass of PaymentInfo)
FixedRateTaxation	MultiplicityME
GuestCheckout	class (GuestCustomer)
Image	enumeration value (MediaTypes), attribute (in Product), attribute (in EShopArtifact)
Language	enumeration (Languages), attribute (in RegisteredCustomer)
MediaFiles	class (Media), enumeration (MediaTypes)
MultipleClassification	MultiplicityME
MultiLevel	composite (to Category – self-containment)
MultipleWishLists	MultiplicityME, attribute (in Wishlist)
PastCustomersAlsoBought	association (self-directed to Product)
Percentage	class (PercentageTaxRule)
PhoneOrder	class (subclass of PaymentInfo)
PhysicalGoods	class (PhysicalProduct, subclass of Product)
PurchaseOrder	class (subclass of PaymentInfo)
QualityofServiceSelection	attribute (in ShippingInstructions)
QuickCheckout	class (CheckoutProfile)
Region	enumeration value (AddressResolutionTypes)
Registration	class (RegisteredCustomer)
RuleBasedTaxation	3x attributes (in TaxRule), enumeration (AddressResolutionTypes)
Services	class (Service, subclass of Product)
Shipping	class (ShippingInstructions), enumeration (Carriers), enumeration (ServiceTypes)
ShippingAddress	association (RegisteredCustomer to Address)

Table B.1: Class Diagram (Store Front Entity Model) Annotation Analysis (continued)

Feature	Model Element (context)
Size	3x attributes (in PhysicalProduct), 1 attribute (in ElectricalProduct)
Sound	enumeration value (MediaTypes)
SubstituteProduct	association (self-directed to Product)
Surcharge	class (SurchargeTaxRule)
Video	enumeration value (MediaTypes)
Weight	1 attribute (in PhysicalProduct)
WishLists	2x classes (WishList, WishListItem), MultiplicityME

Table B.2: Class Diagram (Service Diagram) Mapping Analysis

Feature	Model Element (context)
CreditCard	operation (in IPaymentFrontend)
CustomShippingMethod	service (InHouseShipping)
CustomTaxation	service (TaxCalculator)
DebitCard	operation (in IPaymentFrontend)
ElectronicCheque	operation (in IPaymentFrontend)
InventoryTracking	2x operations (in IWarehouseFrontend)
MultipleShipments	operation (in IWarehouseBackend)
OrderStatusViewing	operation (in IOrder)
ProductReturns	3x operations (in IWarehouseBackend), 2x operations (in IPaymentBackend, in IOrder)
Shipping	interface (IShipping), operation (in IWarehouseBackend)
ShippingGateways	service (ShippingGateway)
TaxGateways	service (TaxGateway)
WarehouseManagement	service (Warehouse), 2x interfaces (IWarehouseBackEnd, IWarehouseFrontEnd)

Many adjacent model elements in activity diagrams are annotated with the same feature; therefore, it was convenient to define some descriptive notation for these situations. Under the mode elements annotated column in table B.3, the following descriptions were used:

- "[*branchLabel*] branch of *DecisionNodeName* decision node" is used to describe a set of model elements which correspond to a branch off a decision node. The set consists of an action or call behaviour and the adjacent flows. If the term "bypass branch" is used, the set consists of a single flow from the decision node to the next decision or merge node.
- "action (*context*)" is used to describe a single action being annotated. The context is defined as either *disjunction*, which means that the PC is a disjunction of multiple features, or *sequential path*, meaning that the action is part of a chain of actions;
- "sequence (*elements*)" is used to describe a set of nodes or flows that occur sequentially. The *elements* define the model elements which are included in the sequence.
- "activity (*activityName*)" is used to describe the annotation of an activity.
- "control flow" is used to indicate that the annotation applies to a single flow. This is used primarily for flow type correction.

Table B.3: Activity Diagram Annotation Analysis

Feature	Diagram	Model Elements Annotated
AdvancedSearch	SearchProduct	[advanced] branch of Search Type decision node
AutomaticUpdates	CheckoutItems	action (sequential path)
BasicSearch	SearchProduct	[basic] bypass flow of Search Type decision node
CarrierSelection	CheckoutItems	action (disjunction with QualityofServiceSelection)
CarrierSelection	CreateQuickCheckoutProfile	action (disjunction with QualityofServiceSelection)
Categories	SelectProductFromCatalog	sequence (action, flow, central buffer), TypeME on central buffer, [yes] branch of Is Category decision node, [no] branch of More Subcategories node
Coupons	CheckoutItems	action (sequential path)
Coupons	CreateOrder	[yes] branch of Coupons Submitted decision node, data store + adjacent flow
Discounts	CheckoutItems	action (sequential path)
ElectronicPage	CheckoutItems	action (sequential path), data store + adjacent flow
Emails	FindProduct	[e-mail] branch of Product Source decision node
EmailWishList	SendWishList	activity (SendWishList)
EmailWishList	StoreFront	[sendWishList] branch of Select Action node
GuestCheckout	CheckoutItems	[guest] bypass flow of Customer Type decision node
Multilevel	SelectProductFromCatalog	[yes] branch of More Subcategories node
MultipleWishLists	SelectProductFromWishList	sequence (control flow, central buffer, control flow)
!MultipleWishLists	SelectProductFromWishList	control flow
MultipleWishLists	SendWishList	sequence (2x actions, 2x central buffers, 4x flows), incoming flow from data store
!MultipleWishLists	SendWishList	control flow
OrderStatusViewing	CheckOrderStatus	activity (CheckOrderStatus)
OrderStatusViewing	StoreFront	[checkOrderStatus] branch of Select Action node
ProductReturns	RefundOrder	activity (RefundOrder)
ProductReturns	StoreFront	[refundOrder] branch of Select Action node
ProductVariants	FindProduct	TypeME, sequence (3x actions, 2x central buffers, adjoining flows)
QuickCheckout	CheckoutItems	sequence (2x actions, 2x flows), sequence (2x flows, 1x actions), [quick] bypass flow of Checkout Type decision node
QuickCheckoutProfile	CreateQuickCheckoutProfile	activity (CreateQuickCheckoutProfile)

Table B.3: Activity Diagram Annotation Analysis (continued)

Feature	Diagram	Model Elements Annotated
QuickCheckoutProfile	UpdatePersonalProfile	[quickCheckoutProfile] branch of Profile Type decision node (consists of the entire region up to the final merge node)
QualityofServiceSelection	CheckoutItems	action (disjunction with CarrierSelection)
QualityofServiceSelection	CreateQuickCheckoutProfile	action (disjunction with CarrierSelection)
RegisteredCheckout	CheckoutItems	[registered] out flow of Customer Type decision node, [regular] out flow of Checkout Type decision node
Registration	CreateWishList	[registered] branch of Customer Type decision node, data store + adjacent flow
Registration	RegisterWithTheStore	activity (RegisterWithTheStore)
Registration	ResetPassword	Activity (ResetPassword)
Registration	SelectProductFromWishList	[registered] branch of Customer Type decision node, data store + adjacent flow
Registration	StoreFront	3x branches ([registerWithStore], [updateProfile], and [resetPassword] branch of Select Action node
Registration	UpdatePersonalProfile	activity (UpdatePersonalProfile)
Searching	FindProduct	[search] branch of Product Source node
Searching	SearchProduct	Activity (SearchProduct)
Shipping	CheckoutItems	2x actions
Shipping	CreateQuickCheckoutProfile	action (sequential path)
Shipping	ProcessOrder	[no] branch of Can Order Be Completely Fulfilled decision node, outgoing flow to data store
WishLists	CreateWishList	activity (CreateWishList)
WishLists	FindProduct	[wishlist] branch of Product Source decision node
WishLists	SelectProductFromWishlist	activity (SelectProductFromWishlist)
WishLists	StoreFront	[createWishList] branch of Select Action node
WishListSavedAfterSession	CreateWishList	[guest] branch of Customer Type decision node, data store + adjacent flow
WishListSavedAfterSession	SelectProductFromWishList	[guest] branch of Customer Type decision node, data store + adjacent flow

References

- [Abl06] Shopping Cart Software: eCommerce Solutions & Hosting, www.ablecommerce.com, Last accessed: January 3, 2006.
- [Alu01], D. Alur, J. Crupi, D. Malks, Core J2EE patterns: best practices and design strategies, Sun Microsystems Press, 2003.
- [Ant04] M. Antkiewicz, K. Czarnecki, FeaturePlugin: feature modeling plug-in for Eclipse, In Proc. Eclipse Technology eXchange (ETX) Workshop, OOPSLA'04.
- [Boo91] G. Booch, Object-oriented design with applications, Benjamin/Cummings, 1991.
- [Bru95] G. Bruno, Model-based software engineering, Chapman & Hall, 1995.
- [Cut05] Cutter Consortium, Software project success and failure, <http://www.cutter.com/press/050824.html>, 2005. Last accessed: January 3, 2006.
- [Cza00] K. Czarnecki, U.W. Eisenecker, Generative programming: methods, tools, and applications, Addison-Wesley, 2000.
- [Cza05a] K. Czarnecki, Overview of generative software development, In Proc. Unconventional Programming Paradigms (UPP), LNCS 3566, Springer, 2005, pp. 313–328.
- [Cza05b] K. Czarnecki, M. Antkiewicz, C.H.P. Kim, S. Lau, K. Pietroszek, Model-driven software product lines, In companion proc., OOPSLA'05.
- [Cza05c] K. Czarnecki, S. Helsen, U. Eisenecker, Formalizing cardinality-based feature models and their specialization, In Software Process Improvement and Practice, Vol. 10, No. 1, 2005, pp. 7-29.
- [Cza05d] K. Czarnecki, S. Helsen, U. Eisenecker, Staged configuration through specialization and multi-level configuration of feature models, In Software Process Improvement and Practice, Vol. 10, No. 2, 2005, pp. 143-169.
- [Cza05e] K. Czarnecki, C.H.P. Kim, Cardinality-based feature modeling and constraints: a progress report, In Proc. International Workshop on Software Factories, OOPSLA'05.
- [Cza05f] K. Czarnecki, M. Antkiewicz, Mapping features to models: a template approach based on superimposed variants, In proc. International Conference on Generative Programming and Component Engineering (GPCE'05), LNCS 3676, Springer, pp. 422-437.
- [Cza05g] K. Czarnecki, M. Antkiewicz, C.H.P. Kim, S. Lau, K. Pietroszek, FMP & FMP2RSM: Eclipse plug-ins for modeling features using model templates. In companion proc. OOPSLA'05.
- [Fow03a] M. Fowler, Patterns of enterprise application architecture, Addison-Wesley, 2003.
- [Fow03b] M. Fowler, UML distilled: a brief guide to the standard object modeling language, Addison-Wesley, 2004.

- [Gam95] E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design patterns: elements of reusable object-oriented software, Addison-Wesley, 1995.
- [Gre04] J. Greenfield, K. Short, S. Cook, S. Kent, J. Crupi, Software factories: assembling applications with patterns, models, frameworks, and tools, Wiley, 2004.
- [Hos05] Shopping Cart Solutions, <http://www.hostpronto.com/article/15>, 2005. Last accessed: November 3, 2005
- [Ibm05] IBM, Welcome to the WebSphere Commerce information center, International Business Machines, publib.boulder.ibm.com/infocenter/wchelp/v5r6/index.jsp, 2005. Last accessed: January 3, 2006.
- [Inv03] Inventoryops, Warehouse Management Systems (WMS), INVENTORYOPS.COM, http://www.inventoryops.com/warehouse_management_systems.htm, 2003. Last accessed: January 3, 2006.
- [Kan90] K. Kang, S. Cohen, J. Hess, W. Novak, A. Peterson, Feature-oriented domain analysis (FODA) feasibility study, Technical Report, Carnegie Mellon University, Software Engineering Institute, CMU/SE-90-TR-21, 1990.
- [Kim05] C.H.P. Kim, K. Czarnecki, Synchronizing cardinality-based feature models and their specializations, In Proc. European Conference on Model Driven Architecture, LNCS 3748, Springer, 2005, pp. 331-348.
- [Nat68] P. Naur, B. Randell (eds), Software engineering: Report of a conference sponsored by the NATO Science Committee, Scientific Affairs Division, NATO, 1969.
- [Omg05a] OMG, Meta-Object Facility (MOF), Object Management Group, <http://www.omg.org/mof/>, 2005. Last accessed: January 3, 2006.
- [Omg05b] OMG, Unified Modeling Language: Superstructure version 2.0 formal/05-07-04, Object Management Group, <http://www.omg.org/cgi-bin/apps/doc?formal/05-07-04.pdf>, 2005. Last accessed: January 3, 2006.
- [Ped03] C.C. Peddy, D. Armentrout, Building solutions with Microsoft Commerce Server 2002, Microsoft Press, 2003.
- [Qua03] T. Quatrani, Introduction to Unified Modeling Language, White Paper, IBM, 2003. ftp://ftp.software.ibm.com/software/rational/web/whitepapers/2003/intro_rdn.pdf, Last accessed: January 3, 2006.
- [Raj00] W.E. Rajput, E-commerce systems architectures and applications, Artech House, 2000.
- [Sha96] M. Shaw, D. Garlan, Software architecture: perspectives on an emerging discipline, Prentice-Hall, 1996.
- [Sta94] Standish Group, CHOAS, The Standard Group Report, http://dimoiv.uqac.ca/~sboivin/C2005/8GIF200_Aut05/raw/process/CHAOS_1994.pdf, 2005. Last accessed: January 3, 2006.

[Sun02] SUN, J2EE Patterns Catalog, Sun Microsystems,
<http://java.sun.com/blueprints/corej2eepatterns/Patterns>, 2002. Last accessed: January 3, 2006.

[Szy03] C. Szyperski, Component software: beyond object-oriented programming, Addison-Wesley, 2003

[Vis05] VISA, Card-Not_Present Fraud Detection, Visa,
http://usa.visa.com/business/accepting_visa/ops_risk_management/technical_information.html, 2005. Last accessed: January 3, 2006.

[Wik06a] Wikipedia Entry: Demographics, <http://en.wikipedia.org/wiki/Demographics>. Last accessed: January 3, 2006.

[Wik06b] Wikipedia Entry: Service, <http://en.wikipedia.org/wiki/Service>. Last accessed: January 3, 2006.

[Wik06c] Wikipedia Entry: Service, http://en.wikipedia.org/wiki/Stock_Keeping_Unit. Last accessed: January 17, 2006.

[Wik06d] Wikipedia Entry: Service, <http://en.wikipedia.org/wiki/DIVX>. Last accessed: January 18, 2006.

[Wit96] J. Withey, Investment analysis of s/w assets for PL, Technical Report, Carnegie Mellon University, Software Engineering Institute, CMU/SEI-96-TR-010, Software Eng Inst, 1996.