Vrije Universiteit Brussel

Faculty of Engineering
Department of Electronics and Informatics – ETRO
Master Program of Applied Computer Science

Design Patterns for the Web Semantics Design Method

Thesis submitted in partial fulfilment of the requirements for the degree of Master of Applied Computer Science

Le Van Huyen

Promoter: **Prof. Dr. Olga De Troyer**

2009 – 2010

Acknowledgements

First of all, I would like to thank my promoter Prof. Dr. Olga De Troyer, for her valuable advices, support, patience and encouragement.

I would like to thank Thanh Hoa Province and Hong Duc University, Vietnam for their financial support. I also want to thank Vrije Universiteit Brussel (VUB) for accepting me as a student and VUB's staffs for their help when I arrived at the university.

I would also like to thank my family's members for their love, patient, and encouragement during two years I study far away from them.

Finally, I thank my friends who have helped me a lot when I needed it and always were beside me to encourage me to finish my thesis.

Abstract

Design Patterns in the Web Semantics Design Method

Le Van Huyen

Vrije Universiteit Brussel, 2010

The Web Semantics Design Method (WSDM) is a methodology to develop web application. WSDM uses models throughout its methodology to represent both information requirements and the conceptual structure. Task models are used to model the tasks that users want to perform with the system. When modeling these tasks, designers are often faced with the same type of tasks over and over again. However, each time the designers, normally, have to model these tasks from scratch. In order to foster re-use of tasks in different contexts of use, task patterns have been introduced as abstractions that should be instantiated for a particular context. A process for using, adapting and applying task patterns in the Conceptual Design phases of WSDM has been studied and developed. In particular, it will be demonstrated how patterns can be used for the establishment of tasks in WSDM. In addition, a case study will illustrate the applicability of patterns in WSDM. Furthermore, some patterns have been discovered, formalized and applied.

Table of Contents

Table of Figures

List of Abbreviations

WSDM: Web Semantics Design Method

CTT: Concurrent Task Trees

HCI: Human Computer Interaction

UI: User Interface

GUI: Graphical User Interface

1. Introduction

1.1. WSDM and Design Patterns

The goal of this Master Thesis is investigate the theoretical approach of integrating design patterns into the WSDM process. In WSDM, we could use design patterns to accelerate the design and development process. For instance, in the sub-phase Presentation Design we could easily use some existing user interface patterns (login, paging, breadcrumb patterns, etc.) to present the information to the users. However, before the designer can use these user interface (UI) patterns, at the Conceptual Design level, the designers have to create task models that express the sequence of tasks and the information required within these tasks. This Conceptual Design phase play an important role in WSDM since it does not only define what end-users can do and find on the web site (by means of the task models) but also is the basis for creating the navigational model and the structure of the web site. Also in this phase, patterns could be a useful addition as many websites support similar tasks. Therefore, in this thesis, I will investigate the use of task patterns in the *Conceptual Design* phase. A proposal approach will be given for using task patterns in WSDM.

In the first step of the Conceptual Design phase of WSDM, the designers have to create task models that represent the interaction between users and the system. As, many web applications offer the same kind of functionality (e.g., searching for something, showing details of something, order something, etc.) some of these tasks or their subtasks may occur again and again in different projects. The question here is if we can create a pattern for each of these tasks and save them in a task library for reusing in different projects. Then, whenever a designer faces the same or similar problem, he can just select and pick the most appropriated pattern and can model the task without having to start from scratch.

1.2. Research Objectives

The research in this thesis covers topics that are strongly related to the design of tasks models through the use of tasks patterns. Therefore, the following issues will be examined in detail:

- Defining task patterns

- Proposing a process to integrate design patterns into WSDM
- A first list of task patterns for WSDM
- A case study, which demonstrates the use of design patterns in WSDM

1.3. Thesis Structure

The thesis is structured as follows:

This thesis has eight chapters including this introduction chapter. The following chapters are divided into two parts. In the first part, the background about Design Patterns and WSDM is covered (chapter 2, and 3). The second part (chapter 4, 5, 6, 7 and 8) is research part. These chapters are briefly summarized as follows.

Chapter 2 introduces the concept of patterns, the evolution of patterns and the current state of the art. This chapter briefly describe about how patterns have been developed and used in software engineering as well as in web engineering.

Chapter 3 covers the background information about WSDM. A brief overview of all design phases of WSDM is given.

Chapter 4 is the main chapter of this thesis in which design patterns are integrated into WSDM. The process of defining and integrating task patterns in WSDM will be discussed in detail.

Chapter 5 presents a first list of task patterns. These patterns are used in the case study.

Chapter 6 is about the case study in which the process of integrating design pattern in WSDM will be illustrated in the design of a real web application. An e-commerce website will be created by using the method proposed in chapter 4.

Chapter 7 presents related work. Similar studies and research on using patterns in web engineering are addressed.

Finally, chapter 8 will end this Master Thesis by presenting general conclusions and by giving possible future work.

Part I

Background

2. Design Pattern

2.1. Evolution of Patterns

The notion of patterns was introduced in the field of architecture by Christopher Alexander and his colleagues in "A Pattern Language" (Alexander, Ishikawa et al., 1977) and "The Timeless Way of Building" (Alexander, 1979). They explained the nature of patterns as follows:

"*Each pattern describes a problem that occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.*"

Thus, patterns explicitly focus on a problem within a context of use and guide designers on when, how, and why a solution can be applied. Patterns are practical and describe instances of "good" design while embodying high level principles and strategies.

In (Alexander, Ishikawa et al., 1977) a pattern is defined as follows:

"A design pattern is a three-part rule, which expresses a relation between a certain context, a problem and a solution. The pattern is, in short, at the same time a thing and the rule which tells us how to create that thing, and when we must create it."

It is noteworthy that his definition of pattern focuses not only on a solution for a problem, but also stresses the relevance of the context. A context here means that a pattern should be applicable in different situation, which leads to a generic solution for a certain problem. Furthermore, the definition includes that a pattern also has to offer rules when and how to use it.

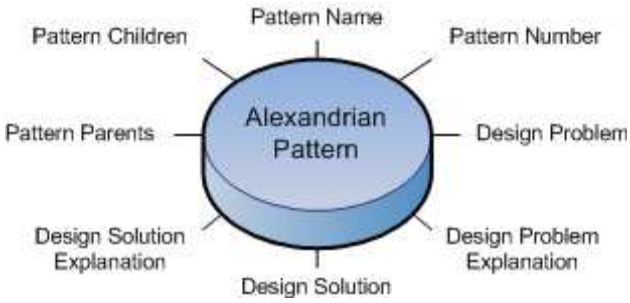


Figure 1. Alexandrian Form of a Pattern (Alexander, Ishikawa et al., 1977)

Another contribution of Alexander was the introduction of a metapattern. He wrote a systematic catalog of patterns, which all were described in the same way. In his schema, a pattern description always includes the points illustrated in the figure 1.

A major point is the relationship between the patterns expressed by means of “Pattern parents” and “Pattern children”. These concepts create links between patterns and thus a pattern language is not only a collection of patterns, but also includes a hierarchical structure. Therefore, the application of a high-level pattern can lead to the application of sub-pattern. This fosters the asset of patterns even more.

The work of Alexander had a great impact on other fields, which resulted in several attempts to translate the idea of formalizing solutions of recurrent problems to other fields.

Gamma et al. (Gamma, Helm et al., 1994) established this concept in the field of computer science. Known as the “Gang of Four” (GoF), the authors introduced software design patterns, which offered reusable solutions for frequent problems in the field of object-oriented software development.

Gamma’s attempt to adopt the idea of Alexander was the first in the field of computer science. In (Gamma, Helm et al., 1994) a pattern is described as follows:

“A design pattern names, abstracts, and identifies the key aspects of a common design structure that make it useful for creating a reusable object-oriented design. … It describes when it applies, whether it can be applied in view of other design constraints, and the consequences and trade-offs of its use.”

Gamma introduced a catalog of 23 patterns, which are categorized in three different classes: Creational Patterns, Structural Patterns and Behavioral Patterns.

The metapattern of Gamma is similar to Alexander’s and contains the following points:

- Pattern Name and Classification
- Intent
- Also Known As
- Motivation
- Applicability
- Structure

- Participants
- Collaborations
- Consequences
- Sample Code
- Know Uses
- Related Patterns

After Gamma's software design patterns, the idea was widely adopted in software development in general. Up to now, there are for instance programming language patterns (Gabriel, 1998), program design patterns (Coplien and Schmidt, 1995), HCI patterns (Ahmed Seffah and Forbrig, 2002; Welie, 2004; Tidwell, 2005; Van Duyne, Landay et al., 2006) .

Most of the developed patterns focus on the internal structure of an application. They are useful for the design of low-level models (e.g., code models), which are mostly used by software developers. However in recent years, patterns have also appeared for "high-level" models, like task models (Sinnig, 2004) and UI models in general (Welie, 2004; Tidwell, 2005; Van Duyne, Landay et al., 2006).

The user interface design community has been a forum for vigorous discussion on pattern language for user interface design and usability engineering. An UI pattern is an effective way to transmit experience about recurrent problems in the HCI domain.

In the HCI field, several groups developed pattern languages. In "The Design of Sites" (Van Duyne, Landay et al., 2006) a set of patterns with respect to all design aspects of the design of web sites are collected. They created patterns for all kinds of web sites, such as e-shops or private web sites. The patterns are hierarchically ordered and divided into twelve groups. The levels of abstraction guide the user through the design of the website by starting with the highest level called "Site Genres", where the user can select his/her genre of the site.

Welie's patterns (Welie, 2004) are quite similar to the idea of Van Duyne, since the patterns are arranged in a hierarchy and references are directed to lower level patterns to support the user for creating interactive applications. Welie also distinguishes patterns between the kinds of application they are made for. There are web design patterns and GUI interface patterns as well, among others. Thus, these patterns offer very concrete recommendations on how to design a web UI.

Tidwell (Tidwell, 2005) collected over 40 patterns, which also describe how to build suitable UIs. She also organizes her patterns into categories but does not introduce a hierarchy like Welie or Van Duyne. The categories are divided by the purpose of the interaction, such as “Showing Complex Data” or “Getting Input From Users”. Hence, the patterns are convenient for all kinds of applications and devices. The author offers screenshot on how to instantiate the pattern on different devices and for different kind of applications.

2.2. Task Patterns

2.2.1. Definition of Task Patterns

In (Sinnig, Forbrig et al., 2003) the authors proposed two different kinds of patterns that are applicable for the user-task model.

Task Patterns describe the activities the user has to perform while pursuing a certain goal. The goal description acts as an unambiguous identification for the pattern. In order to compose the pattern as generic and flexible as possible the goal description should entail at least one variable component. As the variable part of the goal description changes, the content solution part of the pattern will adapt and change accordingly. Task Patterns can be composed out of sub-patterns. These sub-patterns can either be task patterns or feature patterns.

Feature Patterns applied to the user-task model describe the activities the user has to perform using a particular feature of the system. Feature is considered as an aid or a tool the user can use in order to fulfill a task. Examples of these features can be “Keyword Search”, “Login” or “Registration”. Feature patterns are identified by the feature description, which should also contain a variable part, to which the realization if the feature (stated in the pattern) will adapt.

The difference between task and feature patterns is subtle, but noticeable. While task patterns concentrate on a specific goal, the same task can be accomplished in different ways using different feature patterns. That is why feature patterns are important as a classification.

Similarly, the same feature pattern can be used to accomplish different task patterns. Therefore it is safe to say that there is a many-to-many relationship between the two. To summarize, Task patterns are concerned with the user goals (what we need to do), while Feature Patterns are concerned with the system behavior (how we can do it).

2.2.2. Task Modeling

Task models describe the tasks (and sub-tasks) a user has to execute to achieve a certain goal. A task model is a hierarchical structure, which expresses the activities a user has to execute to fulfill this task. A goal is understood as a result a user wants to obtain by executing a set of activities or an attempt to retrieve information from a software system. Most task models support also the concept of objects, which are manipulability by the user to execute a certain task. Objects are also used to represent the state of the system if it is necessary.

The asset of task models is founded in that it is a user-centered approach. Task model designers concentrate on users and capture their activities. Considerations about how a user can reach a goal using a certain software system can foster usability, which is of great interest. Even without using task models for UI generating process they help capturing usability requirements, which are often disregarded.

The most common and well accepted notation for task model is the Concurrent Task Trees (CTT) notation of Paternò (Paternò, Mancini et al., 1997). A CTT model is a set of tasks of different types, and relationships between these tasks. The most important relationship is the composition: tasks are arranged hierarchically. This leads to a directed tree, whose nodes are the tasks. Every task can be decomposed into sub-tasks, except activities, and every task has a super task, except the root task. The graphical visualization as a tree allows a faster interpretation, because it shows the hierarchy of tasks easily.

Paternò distinguishes between four types of task nodes, which are depicted in Figure 2.

Figure 2. Task types in CTT

- **User tasks:** Tasks that are entirely performed by the user, and does not require any interaction with the system. This usually involves tasks where the user will have to make a decision (a choice) before selecting from a number of options. These types of tasks are mainly for clarification, so the reader of the model knows where the user will have to make decisions.

- **Application tasks:** These are tasks entirely performed by the system; they do not require any user interaction. They indicate that the system is doing some

processing work or calculations before the next task can be performed.

- **Interaction tasks:** Tasks that are performed by users interacting with the system. Here, we have both the user and the system involved in the task. A typical example is when a user quits a process on the system: the user presses a button which makes the system ending the process.

- **Abstraction tasks:** these tasks are complex, abstract, and do not fall under any of the categories above. When an abstraction task appears as an elementary task in the tree, it indicates that the task will be decomposed in another task model. It serves as a reference to this task model.

The temporal operators connecting the sub-tasks indicate temporal relationships. CTT distinguishes 8 temporal relationships:

- **Choice** T1 [ ] T2: A choice temporal relationship signals a choice to be made between task T1 and T2. After making the choice, we continue with the next task. One of the tasks will not be performed.

- **Order Independency** T1 |=| T2: Task T1 and T2 can be done in any order and both tasks will have to performed in order to continue.

- **Concurrent** T1 ||| T2: Both tasks can be performed concurrently, at the same time.

- **Concurrent with information exchange** T1 |[]| T2: Both tasks can be performed concurrently, at the same time, but will need to synchronize by exchanging information. This is also called synchronization in earlier versions of CTT.

- **Enabling** T1 >> T2: Before we can do task T2, we will first have to do task T1. T1 enables to do T2.

- **Enabling with information exchange** T1 []>> T2: Same as Enabling but with exchange of information between the tasks.

- **Disabling** T1 [> T2: Also called deactivation. The task T1 can be disabled by performing task T2.

- **Suspend/Resume** T1 |> T2: We can suspend the T1 task and move directly to task T2. While doing T2, we can also go back and resume task T1.

- **Iteration** T*: The task T is iterative; it can be repeated as much as needed.

- **Finite iteration** T(n): The task T is iterative and can be repeated n times.
- **Optional task** [T]: Task T is optional and doesn't need to be performed; it is not mandatory.
- **Recursion** T: The possibility to include in the task specification the task itself. Recursion appears when one of the sub-tasks of a task has the same name as the task itself.

2.2.3. An example of the Use of Patterns for the User Task Model

In this section, we will discuss briefly an example to illustrate how patterns can be applied in task modeling. As an example, we use the Search task pattern (Sinnig, 2004). Suppose the user wants to obtain a subset of data from a system. The type of data is not limited to a special class. The user has to know, which kind of data he/she is interested in. Therefore, the user has to enter several values to formulate his query. After querying, the application displays the subset of data that matches the query.

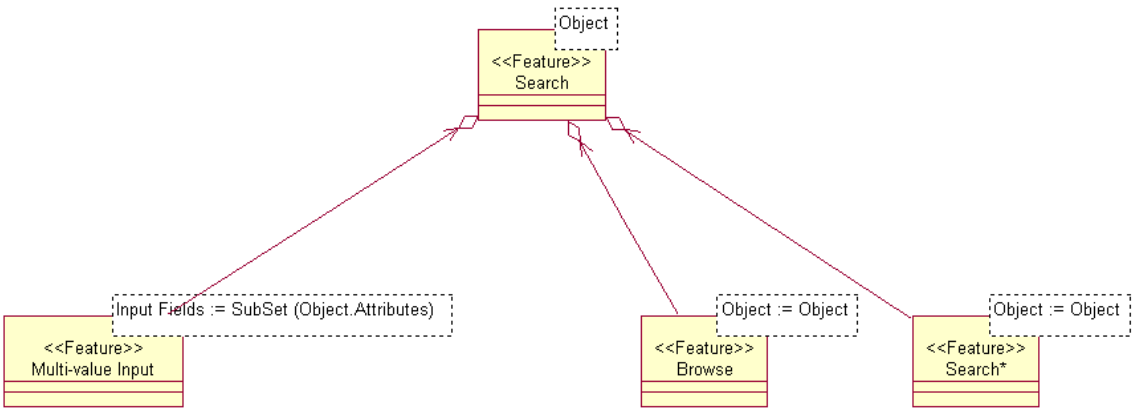


Figure 3. Interface of the Search Pattern (Sinnig, 2004)

As a solution, the pattern suggests to give the user the possibility to enter the search query. Based on this query, a subset of the searchable data is calculated and displayed to the user. The Multi-value Input Pattern (Paternò, 2000; Sinnig, 2004) may be used for the query input. After submission, the results of the search are presented to the user and can then, in turn, either be browsed (Sinnig, 2004) or used as input for refining the search.

In this Search pattern, the variable 'Object' is a placeholder for a particular object of information that the user wants to search for. This pattern is also composed of some other patterns; the Multi-Value Input pattern, Browse pattern, and recursively of itself.

The variable Object in the Search pattern will be assigned to the Object of the Browse and the Search pattern. Moreover, the attributes of the Search object are used to determine the various Input Fields of the Multi-value Input pattern, which is used to conduct the search query.

From this pattern, we can specify the tasks for this pattern as in the CTT given in figure 4. In order to apply and integrate the task structure, the pattern and all its sub patterns must be instantiated and customized to the current context of use.

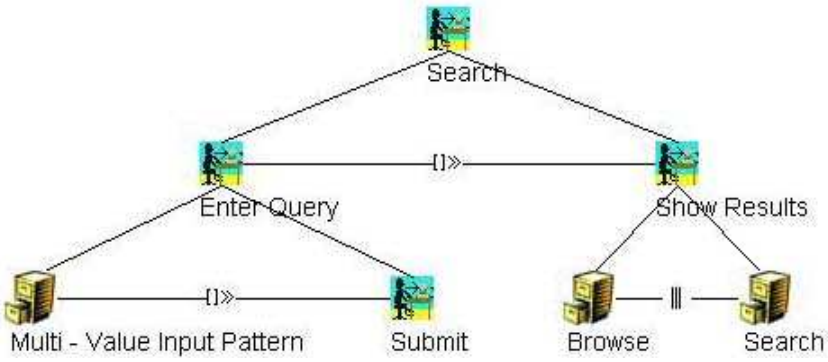


Figure 4. Task Specification of the Search Pattern (Sinnig, 2004)

2.3. Benefit of using patterns

According to Vora (Vora, 2009) in the book "Web Application Design Patterns", the benefits of using pattern can be summarized as follows:

Proven design solutions and guidance for their use: Patterns identify real solutions and are more than abstract principles or guidelines. In addition, by making the context and problem explicit and summarizing the rationale for their effectiveness, patterns explain both how a problem can be solved and why the solution is appropriate for a particular context. However, because it's a generic "core" solution, its use can vary from one implementation to another without making it look "cookie-cutter" or discouraging creativity.

Improved design process: Identifying design patterns and cataloging them can help designers to increase their productivity by reducing time spend "reinventing the wheel." Furthermore, if (user interface) components are built for patterns in the form of a design pattern library, designs can be realized, tested, and iterated rapidly, and can help shorten release cycles.

Reusability and consistent interfaces: Developing a library of reusable user interface components can facilitate development of consistent interfaces both within and across applications. This is particularly useful in large corporations with multiple and distributed design teams, where different solutions may be applied for the same problems by different design groups, leading to inconsistent interfaces among designs produced within the same company. By cataloging and communicating design patterns, teams can increase consistency, predictability, and usability of their designs and it can serve as a corporate memory of design expertise.

A common, shared language: Patterns help supporting and improving communication among team members from diverse disciplines by developing a common language or vocabulary when explaining and discussing the design solutions. This is very important because user interface designers often work in an interdisciplinary team with developers, application domain experts, and users or user representatives, and these groups typically lack a common terminology to exchange design ideas and opinions.

Effective teaching aid and reference tool: Patterns also can be an effective way for experienced designers to offer design guidance to those without a formal background in design. Because of the approach used in documenting patterns, by providing visual and textual description, it is easier for novice interface designers to see examples of their successful usage.

Usable applications: Finally, because patterns are based on a history of successful usage, their use can make the application more usable because interactions recommended by patterns would be familiar to users.

3. Web Semantics Design Method

3.1. Introduction

Web Semantics Design Method (WSDM) is a method for developing website. It was introduced by De Troyer and Leune in 1998 (De Troyer and Leune, 1998). This method is an audience-driven approach, which takes the requirements of the users of the web site as a starting point and uses this as basis for the structuring of data and the web site structure afterwards. WSDM uses models throughout its complete design process to represent both requirements and the conceptual structure. WSDM gives consideration to the fact that web sites usually have different types of visitors that may have different needs and requirements. Fulfilling these requirements leads to higher usability and greater satisfaction.

WSDM method consists of a sequence of phases. For each phase, there is a well-defined method to identify the information and functionality needed for the web system and the structure of the website in an appropriate way. The output of one phase is the input of the following phase.

In this chapter, a brief discussion about the WSDM is provided. The general idea of all phases in WSDM is provided without going into detail. More detail information can be found in (De Troyer, 1998; De Troyer and Leune, 1998; De Troyer, 2001; De Troyer et al., 2005; De Troyer, Casteleyn et al., 2007).

The present status of further research about WSDM can also be consulted on the WSDM web site: http://wsdm.vub.ac.be.

In the chapter 5, the case study presents the full process of WSDM to design a web site.

The overview of the WSDM method is given in the figure 5. It has a sequence of phases:

1. Mission Statement Specification
2. Audience Modeling
3. Conceptual Design
4. Implementation Design
5. Implementation

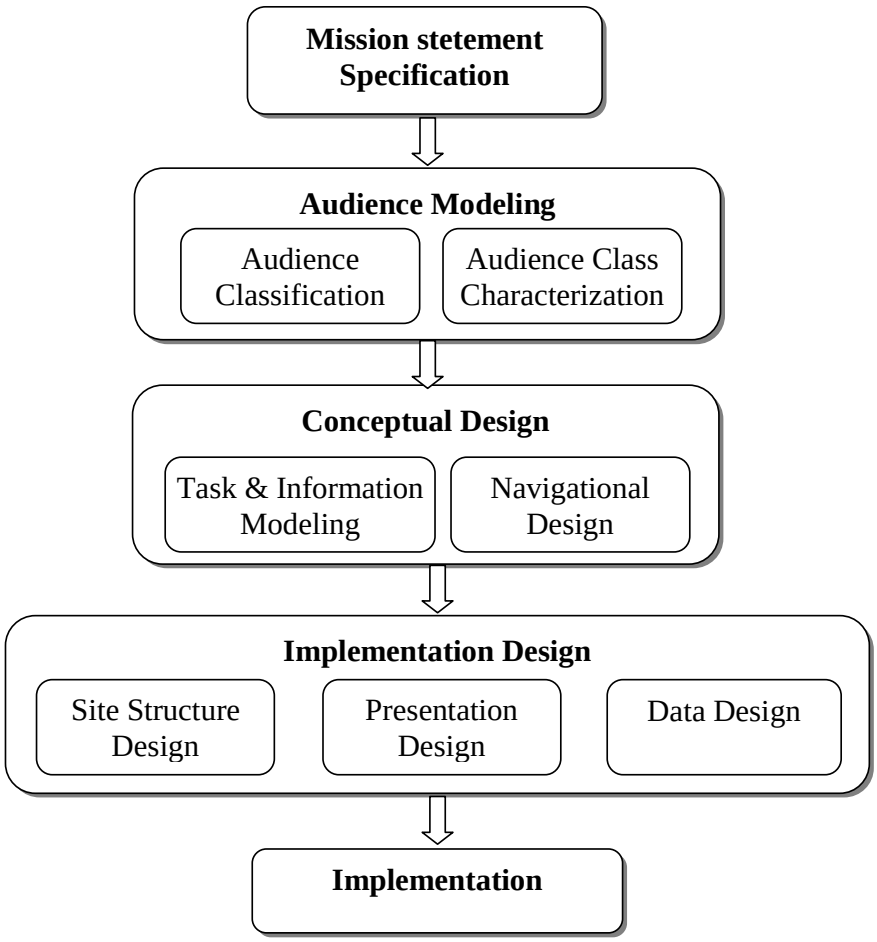


Figure 5. Overview of WSDM method

3.2. Mission Statement Specification

The first phase in WSDM is to define the Mission Statement. The goal of this phase is to identify the purpose of the web system as well as the subject and the target users of the web system. In order to formulate the mission statement of the web system the designers should answer the questions:

- What is the purpose of the web system?
- What are the target users of the web system?
- What are the subjects of the web system?

The purpose of the website should be identified because without a purpose it is impossible to define the functionality and information of the web system; moreover, there will be no proper basis for making design decisions or for evaluating the effectiveness of the web system.

The target users of the web system are the users that we want to address or that will be interested in the web system. Different group of users may have different information requirements that should be reflected in the content of the website. Identifying exactly what users want and what they need could help the designers to create and give the appropriate information for the users. Therefore, we have to identify the target users of the web site.

The subject (topics) of the web system is related to the purpose and the target users of the web system. The subject must allow fulfilling the purpose of the web system, and it must be adapted for the target users.

The output of this phase is the mission statement. It is formulated in natural language and must describe the purpose, subject, and target users of the web systems. In fact, the mission statement establishes the borders of the design process. It allows (in the following phases) deciding which information or functionality to include or exclude, how to structure it, and how to present it.

3.3. Audience Modeling

During Audience Modeling, the target users identified in the mission statement are refined into audience classes. This is done by means of two sub-phases: the audience classification and the audience characterization.

3.3.1. Audience Classification

The target users identified in the mission statement are the input for the audience classification. These target users are refined and classified into audience classes based on differences in their informational and functional requirements. All members of an audience class must have the same set of informational and functional requirements.

In order to identify the audience classes the following method is used:

Step 1: Consider the activities of the organization related to the purpose of web system

Step 2: For each activity

1. Identify people involved
2. Restrict them to the target users
3. Identify their requirement

4. Divide them into audience classes based on different information or functional requirements
5. Decompose the activity if possible and repeat Step 1

By doing this, a hierarchy of audience classes is constructed. At the top of this hierarchy is the class Visitor that represents all target users. The requirements associated with the Visitor class are the requirements that are common to all users in the web system. An audience class is a subclass of another audience class if the set of requirements associated with the subclass contains all the requirements associated with the super class and some extra requirements.

3.3.2. Audience Class Characterization

After identifying all audience classes, the relevant characteristics of each audience class should be specified. Some examples of characteristics are the language, the age, the experience with website in general, the lifestyle… of the members of the audience class. From, Some of these characteristics may be translated into usability requirements while others may be used in the Implementation Phase to design the "look and feel" for each audience class such as choice of colors, fonts, graphics, etc.

3.4. Conceptual Design

At this point in the method, the designer has identified all the requirements and characteristics of the different target users (audience classes). The goal of the Conceptual Design is to turn these informal requirements into high level and formal descriptions from which later on the web system can be generated.

The Conceptual Design covers the conceptual "what and how" of the web system. The conceptual "what" is covered by the Task Modeling sub-phase which models the information and functionality. The conceptual "how" is covered by the Navigational Design sub-phase which models the navigational structure of the Web system.

3.4.1. Task and Information Modeling

The purpose of the Task and Information Modeling is to model in detail the different tasks the members of each audience class need to be able to perform and to describe the data and functionality that is needed for those tasks.

An adapted version of the task modeling technique Concurrent Task Tree (CTT) is used to

model tasks in WSDM. There are some things from the original CTT (has been discussed in the previous chapter) that have been modified and changed in WSDM.

- The first one is the categories of tasks. WSDM does not use all four different kind of task categories as in the original CTT; instead WSDM uses only three categories:
  - Application tasks: Tasks executed by the application
  - Interaction tasks: tasks performed by the users by interaction with the system
  - Abstract tasks: tasks that consist of complex activities

Figure 6. Task types in WSDM

- The operators are used to express the temporal relationship among tasks. In WSDM the using of the operators is not exactly as in the original CTT. There are some differences:
  - The iteration (T*) is changed slightly. In CTT the meaning of this operator is that the action is performed repetitively until it is deactivated by another task. However, in WSDM this operator is used with the meaning that the task can be repeated several times and ends when the one in charge (the user or the web application) decides not to repeat the task.
  - There is new operator in WSDM, the transaction (→ T ←): the task must be executed as a transaction. The whole task (including all its subtasks) is executed completely, or in case there is an error, it should be rollback; no change is applied.
- Furthermore, tasks decomposition is stopped earlier than in the original CTT method. In the original CTT method, the leaves of the tree correspond with elementary user interactions or application functionality. In WSDM, a leaf corresponds with an elementary task, which is a task dealing with one type of information.

By using this adapted CTT version, the task models are created and decomposed until elementary tasks are obtained.

When the task models are completed, for each elementary task, an object chunk is modeled to describe the information and functionality needed to complete this task. WSDM uses Web Ontology Language (OWL) to model the information needs. Since OWL does not have a commonly accepted graphic notation, WSDM uses the Object Relation Model (ORM) graphic notation to present the information modeling in OWL. Because ORM and OWL are very close, the mapping from ORM to OWL is straightforward.

WSDM also extends ORM to provide some extra notations for expressing the functional needs of a task. For the more detail about this can be found in (De Troyer, Plessers et al., 2007).

3.4.2. Navigational Design

The goal of the Navigational Design is to define the conceptual structure of the web system and to model how the members of the different audience classes can navigate through the web system and perform their tasks.

For each audience class a navigational track is created. This can be considered as a sub-site containing all and only the information and functionality needed by the members of the associated audience class.

From all the audience navigational tracks, the navigational model is created to present the structure of the website.

3.5. Implementation Design

The goal of the implementation design is to complement the conceptual design with the necessary details for the implementation. The implementation design consists of three sub phases: the Site Structure Design, the Presentation Design and the Logical Data Design.

3.5.1. Site Structure Design

In this phase, the designers will decide how to present the components from the navigational model in web pages. By default, each component will be placed on a page. However, the designers can group some conceptual navigation components on one page or they can present one component over different pages.

The characteristics of each audience class can be taken into account when deciding

which information should be presented on a page. For example, for old people, the information presented on a page should not be too much. This leads to the idea of splitting a component and present it on several pages.

At this step, several site structures can be created in order to support different devices, context, or platforms. For instance, with the limit of the screen size, we can not present the same amount of information on a mobile device as on a regular computer screen. This requires another site structure to present in a suitable way information on a mobile device.

3.5.2. Presentation Design

The Presentation Design phase defines the look and feel of the web system, as well as the layout of the pages (i.e. positioning of page elements). To enhance a consistent look and feel, templates are used. Therefore, page templates are defined.

In a web system there are many different kinds of pages like a home page and leaf pages. For each of these pages a template is created. These templates are subsequently used in the page design, when for each of the pages defined in the site structure model the layout is defined.

The layout describes how the information and functionality (modeled by means of the object chunks and) assigned to a page (by means of the components) should be laid out on the page. To specify style, WSDM currently relies on Cascading Style Sheets (CSS).

3.5.3. Logical Data Design

In the Task and Information Modeling sub-phase the object chunks are created to present the information that are used in each task as well as in the application. The different object chunks are related by means of a reference ontology that contains the different concepts used in the different object chunks.

While generating the logical data schema, it is important to keep track of the mapping between the reference ontology and the logical data schema, because later on (in the implementation phase) the conceptual queries and updates expressed in the object chunks need to be translated into queries and updates onto the logical database schema. By preference, this process is supported by a CASE-tool, in which case the designer is not burdened with the creation of the logical data schema and the mappings.

In case there is available data from another source, this information can be considered

as external source and it is only needed to define the mapping between the reference ontology and this data store.

3.6. Implementation

The final phase of WSDM, the implementation, takes care of the actual realization of the Web system by using a certain implementation environment (HTML, WML for examples). Based on all the information coming from the models resulting from the previous design phases and based on the available tools, the web system can be generated automatically.

Part II

Research

4. Integrating Design Patterns in WSDM

4.1. Introduction

WSDM is a method to design web system by going through a sequence of phases. In some phases, design patterns can be used to accelerate the design process. In the Implementation Design, web patterns can be used to present information to the users in convenience way. For example, paging pattern (Welie, 2004) presents the results grouped in pages with a fixed number of items and allow the users to move easily from one page of items to another; breadcrumb pattern (Welie, 2004) shows the hierarchical path from the top level to the current page and make each step clickable. These patterns at HCI level could support the users by offering high usability and greater satisfactions. However, before the patterns at the user interface level can be used, at the task level, the designers have to model the tasks that users need to perform and the information associated with these tasks. The presentation level only reflects the task level. The success of a web site is depending, not only on its interface, but also on the quality of the task models.

Moreover, in WSDM, from the Conceptual Design phase the navigation and the structure of the web site are derived. Therefore, the Conceptual Design phase plays and important role in WSDM. Using design pattern in this phase, more specifically, in the sub-phase “Task Modeling” will not only help the designers modeling the tasks more quickly but will also improve the design process in WSDM.

In this chapter, a proposal approach for integrating task patterns in WSDM will be presented.

In order to integrate task patterns in the Conceptual Design phase in WSDM, first task patterns need to exist. Therefore, we will first focus on the definition of a task pattern for its use in WSDM. A task pattern for WSDM consists, like a normal task description, of a task model and its associated object chunks. The task models as well as the object chunks in such a pattern are given in an abstract (i.e. generic) way. When the pattern is used in a concrete context, it may be necessary to adapt and extend the pattern’s object chunks in order to create the concrete object chunks for the current context of use. Moreover, the task model from the task pattern needs to be instantiated to create the concrete task model for the current context of use. From this, the task navigational model is created.

In the following part, we first describe how a pattern can be defined (section 4.2). Next, the process of integrating task patterns in the Conceptual Design phase is described in detail (section 4.3).

4.2. Pattern Definition

Since the design pattern is used in many different fields from architecture to software engineering and HCI, several different formats have been used for describing patterns. The pattern description format used in Sinnig (Sinnig, 2004) is different from other formats since there are some added elements to present the task interface and the task model.

In order to describe task patterns in WSDM, the pattern description format is adapted with some new elements to describe the task models, the object chunks, and the task navigational model. The pattern description format has the following elements:

- **Pattern Name**: The pattern must have a meaningful name. Good pattern names form a vocabulary for discussing conceptual abstractions.
- **Problem**: Statement of the problem that the pattern addressed.
- **Context**: Description the context in which the pattern can be applied. Designers, when considering a new design, use the context description to determine if a particular pattern is appropriate.
- **Solution:** This section describes the solution the pattern offers. Moreover, it shows how the pattern works and how the users interact with the system in order to achieve the certain goal.
- **Rational:** Briefly describes how the use of the pattern improves the task structure and how the goal can be accomplished by using the pattern.
- **Structure:** This part presents the task structure, the object chunks, and the task navigational model of the pattern.
- **Related pattern:** List of related patterns, which might be **predecessor patterns** whose application leads to this pattern; **successor patterns** whose application follows from this pattern; **alternative patterns** that describe a different solution to the same.

In the chapter 5, a list of patterns created in this thesis will be presented using this pattern description format,

4.2.1. Pattern and notation

Let us start with an example. A typical task a user performs in many web applications is selecting something from a list. The task "select item" embodies this basic task and in principle can be used for different kinds of items (for example to select a book, select a category…). Therefore, it would be useful to create a pattern for this task such that it can be reuse in many different applications and for different kinds of items. In order to create such a generalized "Select Item" pattern we must abstract from the particular item we want to select, and replace it with a generic variable.

The following figure gives an example of the task pattern "Select One Item" and the notation used. The left part of the figure shows the icon used for a pattern; the right part of the figure gives the details of the pattern.

This pattern has only one user task: "Select Item". This task is a common and basic task that users usually perform in order to select an item. To make the type of the item to select variable, the variable "Object" is used to indicate the item type (e.g., book, category). Variables are put between square brackets.

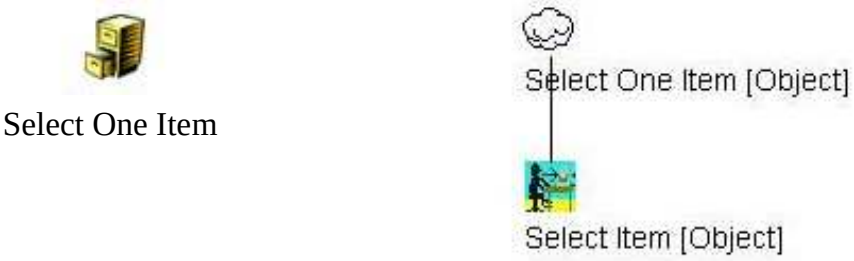


Figure 7. Example task pattern and the notation

4.2.2. The Object Chunks

Similar as for a regular task model in WSDM, an Object Chunk is given for each elementary task in this pattern. For the example given, the object chunk is shown in the following figure.

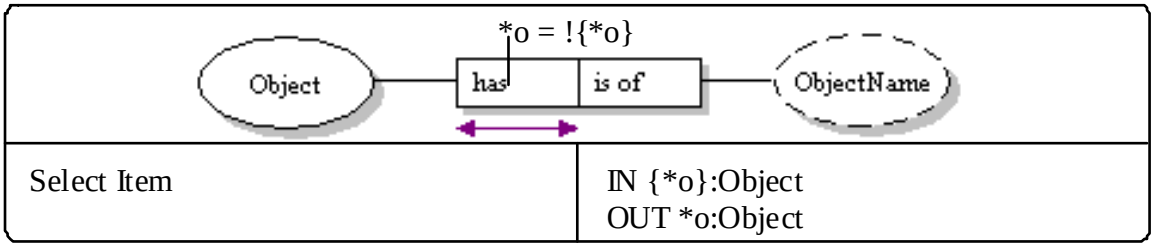


Figure 8. Object Chunk "Select One Item"

The “Object” variable in the pattern presents the abstract object that must be replaced by a concrete object in each context of use. Here, “Object” needs to be replaced by a concrete Object Type (e.g., Book) and ObjectName needs to be replaced by a naming Object Type for the concrete Object Type (e.g., BookTitle).

4.2.3. The Navigational Model

From the task model given for a pattern, the corresponding task navigational model can be created using the same technique as used for a regular task model in WSDM. The task navigational model for the example pattern “Select One Item” is given in the following figure.

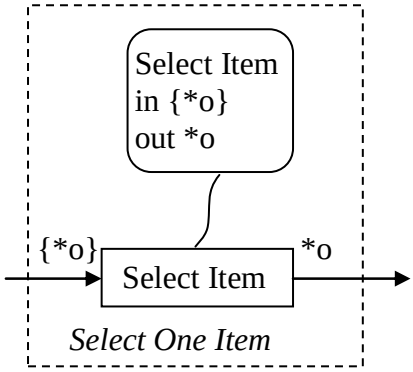


Figure 9. The Navigational Model for the task “Select One Item”

4.3. The Process of Pattern Application

Suppose that we have a library of task patterns and they are ready to be used in the task modeling process. A list of task patterns is given in chapter 5.

In order to create a task model for a particular application, the following process is proposed and it consists of two main steps:

1. Selection: An appropriate pattern is selected to be applied to a task model. By studying the context of use, the designer can decide to select and apply a pattern to express some tasks without the need to design it from the scratch.

2. Adaptation: A pattern is an abstraction that must be instantiated. Therefore, in this step the pattern will be adapted according to the context of use. In a top down process, all variable parts need to be bound to specific values, resulting in a concrete instance of the pattern.

4.4. Example

In order to clarify the previously introduced process, we will now illustrate it with an example. The task “Select and View Book Information” is used for the demonstration.

4.4.1. Selection

In order to view a book’s information, the user should select a book from a list of books. After that, the system will display the detail information of the selected book to the user. For this purpose, the patterns “Select One Item” and “Display Item Information” given as example patterns can be used.

After selecting the patterns “Select One Item” and “Display Item Information” to model the task “Select a Book” and “Display Book Information”, task model of the parent task “Select and View Book information” is given in the following figure.

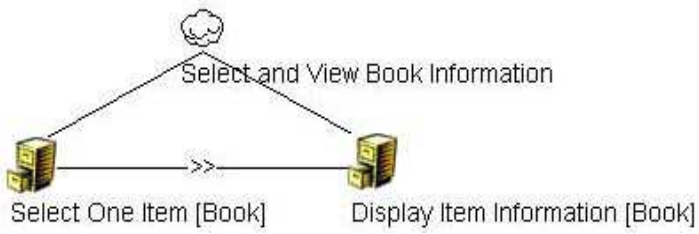


Figure 10. Task Model for the task “Select and View Book information”

In this task model, the Book object type replaces “Object” in the patterns “Select One Item” and “Display Item Information”.

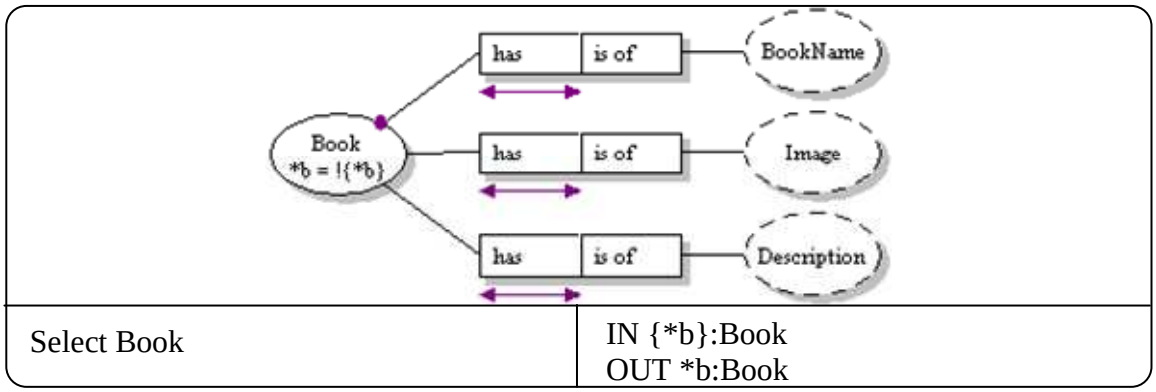
4.4.2. Adaptation

4.4.2.1. Object Chunk Adaptation

If we go back to the section *pattern definition* above, there is an object chunk associated to the task pattern “Select One Item”: object chunks for “Select Item”.

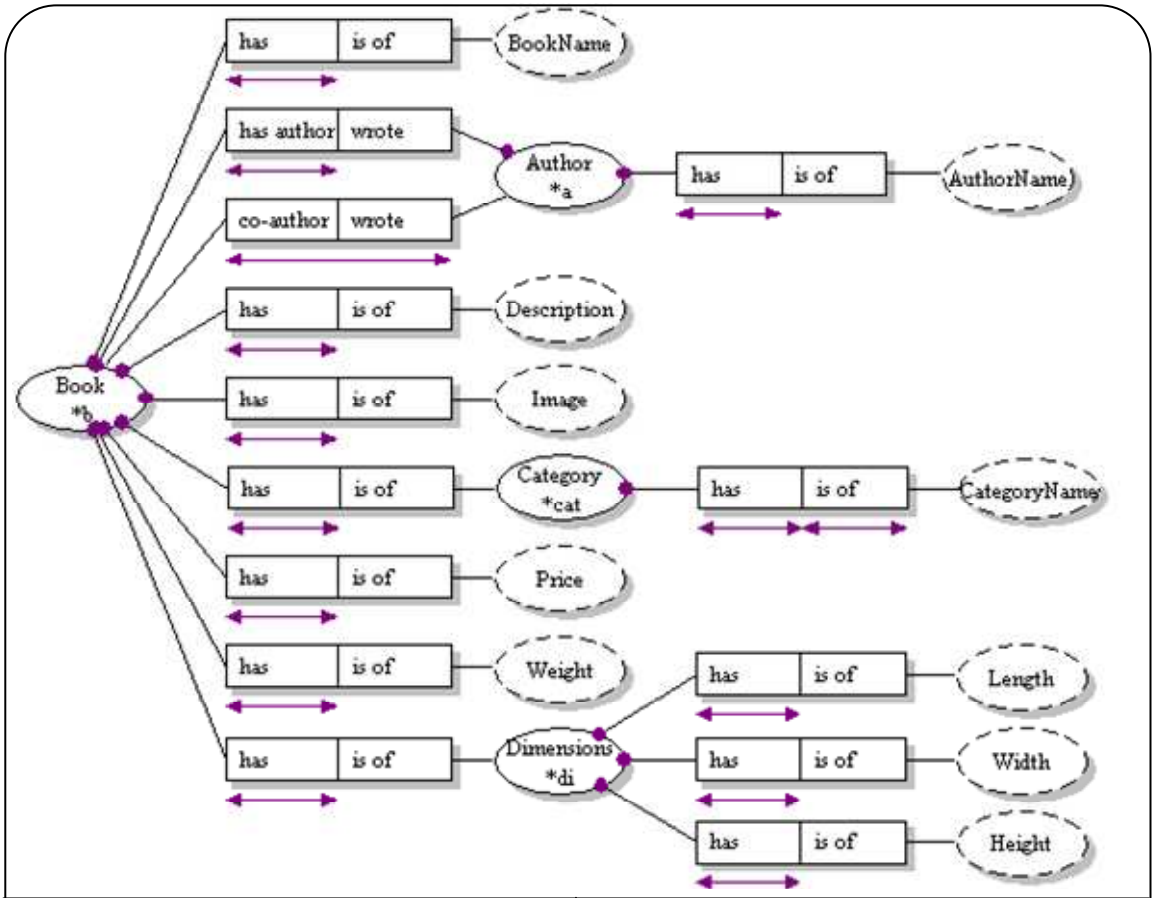
The default object chunk, given in the pattern for “Select Item”, only models the display of the ObjectName for the Object. Now for the current context of use, the task “Select Book” should not only display the book name but also some other properties of the “Book” object. For instance, the designer also wants to display the book’s image and the book’s description. Therefore, at this step and depending on the needs of the application, the designers can decide to add more information and extend the object chunk for this task. This means that he will *adapt* the object chunk in the task

pattern to the current context of use. Furthermore, when the system lists books for the users to select one book and view its information, the designers can decide that the users can click on the book's name or the book's image to select one. This decision leads to the adaptation of the object chunk "Select Book" as follow.



Select Book	IN {*b}:Book OUT *b:Book

Figure 11. Object Chunk "Select Book"



Display Book Info	IN *b:Book

Figure 12. Object Chunk "Display Book Info"

Similarly, for the "Display Item Information" pattern, we have the object chunk "Display Book Info" for presenting to the users the detail information of the selected Book. In the pattern "Display Item Information", the object chunk "Display Item Info" is modeled with some default information like the ObjectName, Description, and CategoryName. Here, we want to present to the users more information about the Book such as the AuthorName, Image, Price, etc. Therefore, the "Display Item Info" is expanded during adaptation. The result is the object chunk "Display Book Info" modeled as in figure 12.

4.4.2.2. Task Instantiation and Expansion

The task model above (figure 10) provides a high level of abstraction of the task "Select and View Book Information" in which we model the tasks "Select a Book" and "Display Book Information" as pattern tasks. These patterns need to be *instantiated* in order to create the full task model.

The following figure presents the full task model for the task "Select and View Book Information" after we instantiated and expanded the patterns "Select One Item" and "Display Item Information". Such a model is called a *fully instantiated* task model.

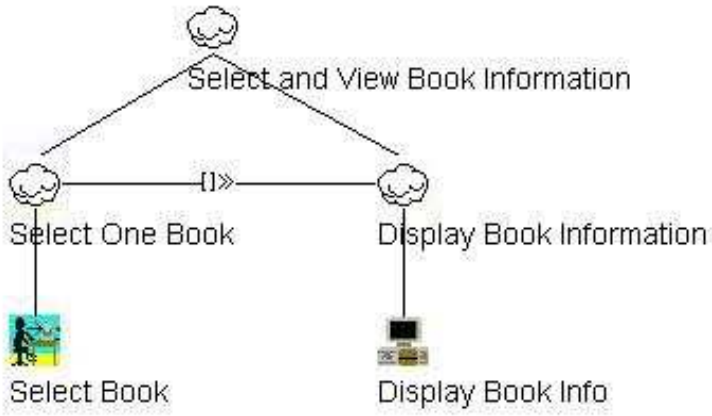


Figure 13. The fully instantiated task model for the task "Select and View Book information"

From this detail task model, the task navigation model can be created. Together with the object chunks that we have created in the previous step, the task navigational model is given in the following figure.

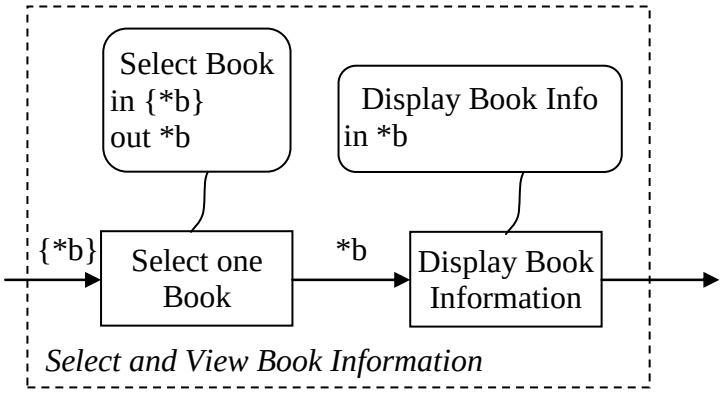


Figure 14. The task navigational model for the task "Select and View Book Information"

From this task navigational model, we realize that the components "Select One Book" and "Display Book Information" are already presented in the task navigational model of the "Select One Item" and "Display Item Information" patterns. In order to make it simpler and easier, a new concept and notation is introduced called the "*pattern component*" – an abstract component that corresponds to the task navigation model fragment of a pattern. A dotted-line rectangle and a character 'P' inside a circle present this notation.

Figure 15. The notation for a "pattern component"

With this new notation, the task navigation model above can be modeled as follow:

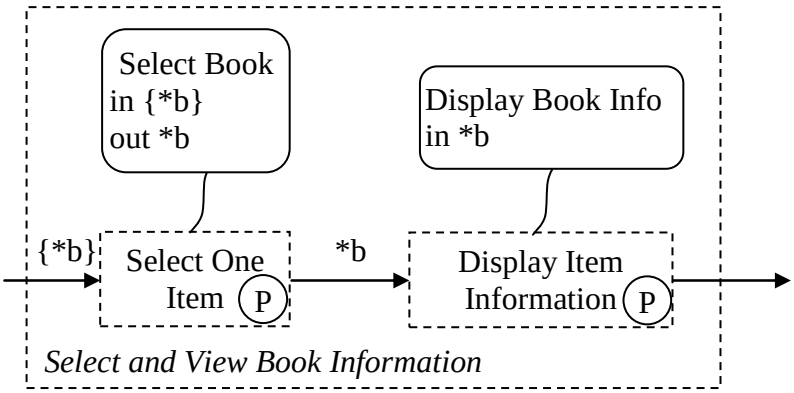


Figure 16. The task navigational model with the "task component"

In a task pattern, one or more object chunks are associated with it. In the Object Chunk Adaptation step, all these object chunks are instantiated in the current context of use. These object chunks, if necessary, can be provided in the navigational model to indicate the component that they are associated with. In the navigational model above, the object chunks “Select Book” and “Display Book Info” are provided to indicate that they are used by the *pattern components* “Select One Item” and “Display Item Information”. Actually, these object chunks can be derived from the tasks “Select One Item [Book]” and “Display Item Information [Book].”

4.5. Summary method

The above process can be summary as follow:

- Step 1: Selecting a pattern to be used in the task modeling
- Step 2: Adaptation
    - Object chunk adaptation: adapt the object chunks from the pattern to the current context of use
    - Task instantiation: Each task pattern will be instantiated resulting in a full task model.

The task navigational model can then be derived from this instantiated task model. However, it is also possible to omit the task expansion and use the navigational model for the pattern as a task pattern component in the task navigational model that is derived from the task model obtained before the expansion.

5. A list of Patterns

In section 4.2 we have defined the pattern description format. In this chapter, a first list of task patterns will be presented. These patterns are derived and defined during the design process of the case study. When elaborating the case study, these patterns were useful and saved me a lot of time. I didn't have to model or create the tasks that appeared more than once; I just had to select the pattern and use it in the case. Of course, after selecting the patterns that I had to adapt it to make it suitable for the context of use.

As only intended as a proof of concept for my study that patterns can be integrated in WSDM, the list of patterns presented is limited and the patterns should be evaluated and improved. However, these patterns are common and abstract enough to be used in different cases. I have applied these patterns in my case study (see chapter 6). In the following will present the first list of patterns in the format has been discussed in chapter 1.

5.1. Register Pattern

Pattern Name: Register

Problem: Web applications often need to uniquely identify users in order to provide personalized content, or opportunities to conduct some tasks (e.g. a purchase), or to prevent unauthorized access to personal and sensitive information (bank account, health record), or to increase convenience such as storing billing and shipping address of users, so that they do not have to retype this information each time they place an order.

Context: This pattern is used when you want to provide the opportunity to end-users to register to the system.

Solution: Registering is usually performed by first entering some personal information that will be validated by the web system and then the system will optionally provide some feedback. After a successful registration, the end-user is known as a new user and will have an account for the system.

In the subtask to enter the users' information, it is very important that some data is mandatory for users to fill in. Often sites ask all sorts of personal details such as username, email address, and password. The required information is not the same in every applications; it is related to the purpose of the site.

After providing the personal information, the users submit this information to register as new user of the website. Depending on the information that the user provided, the system will validate the information and can give feedback information to indicate possible invalid information (for example the username already exists) or some errors that occur during the register processing.

Rational: With the registered account, the web system can provide personalized content to your users. Account registration allows for remembering details about the user; product wish lists, preferences, interests, shipping and billing addresses, and more.

Structure

Task Model

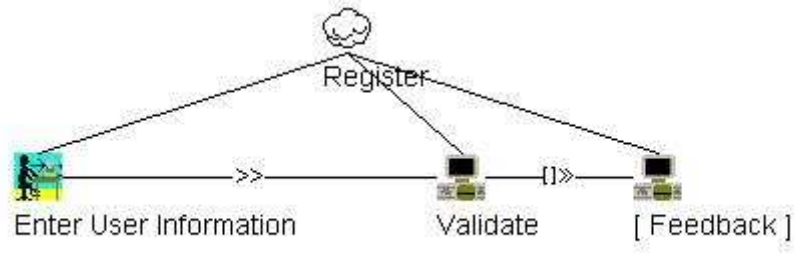


Figure 17. Task Model for the task "Register"

Object Chunks

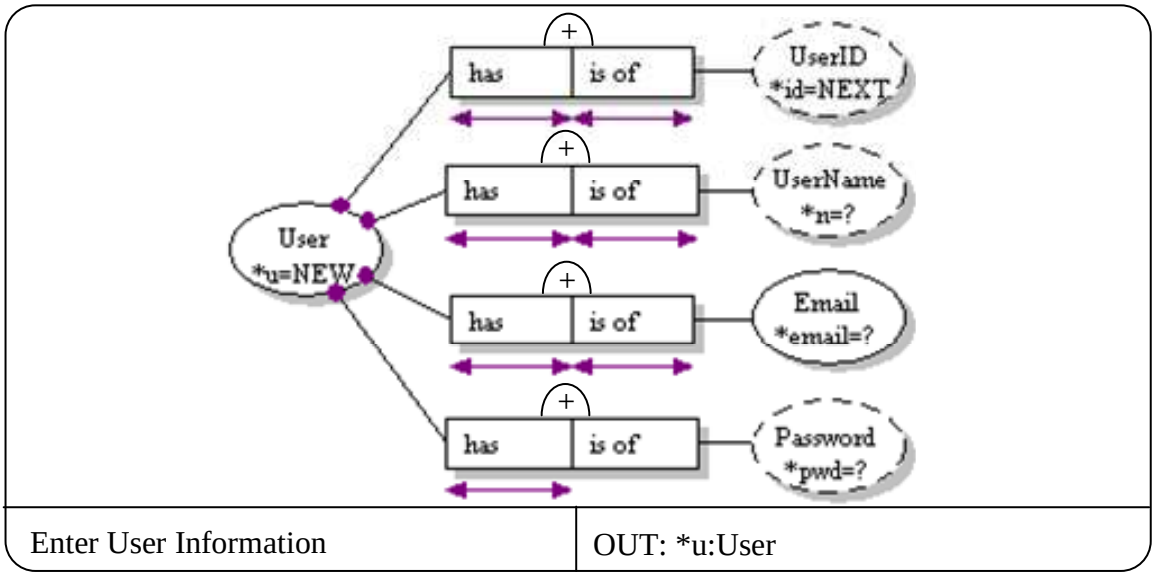


Figure 18. Object Chunk "Enter User Information"

Task Navigational Model

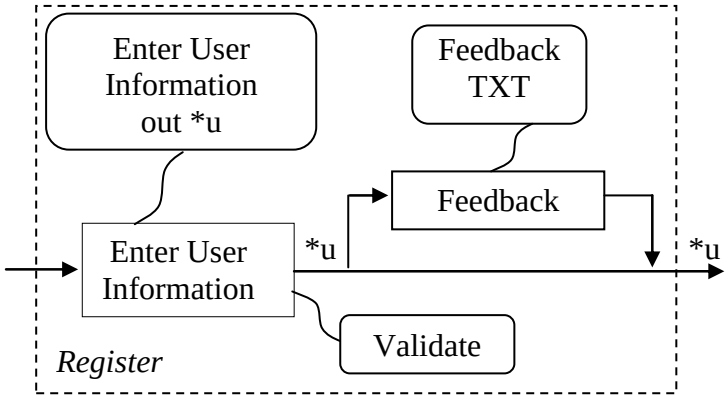


Figure 19. Task navigation for the task "Register"

Related Pattern:

Login: can be used afterwards to allow the end-user to login to the system using the created account

5.2. Login Pattern

Pattern Name: Login

Problem: In some systems, users need to identify themselves in order to access private data or to perform authorized operations.

Context: This pattern can be used to allow end-users that are already known to the system (i.e. have an account) to log into the system using this account.

Solution: In order to login into the system, in general, the user should enter his username and password a (sometimes of the username is an email address). After submission, the application validates the account information and provides feedback whether login was successful or not.

Rational: By using the Login Pattern, the users can uniquely identify themselves. Therefore the application can validate and authorize the user to perform operations according to the user's authorization.

Structure

Task Structure

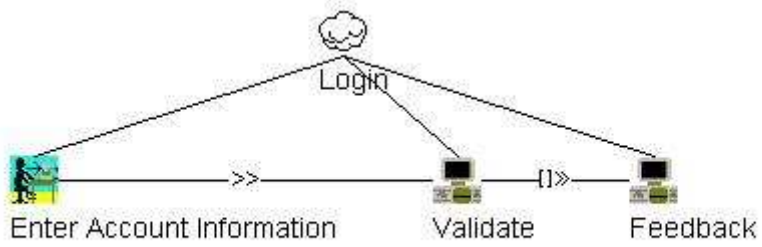


Figure 20. Task Model for the task "Login"

Object Chunks

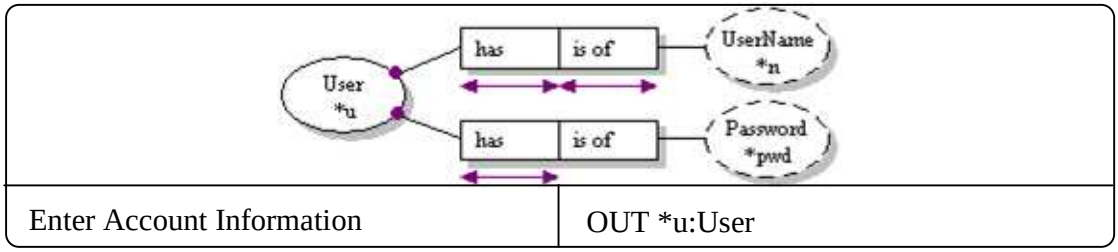


Figure 21. Object Chunk "Enter Account Information"

Navigational Model

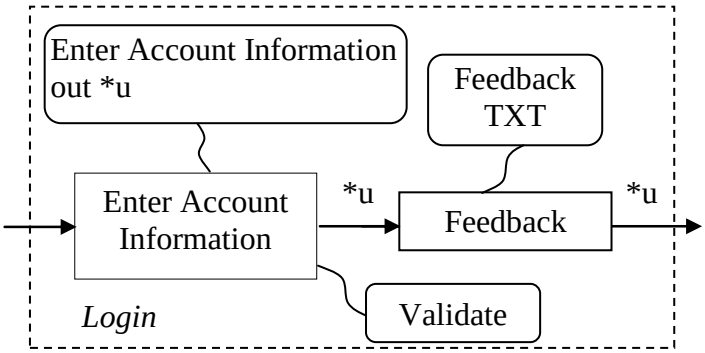


Figure 22. Task navigation for the task "Register"

Related Pattern:

Register: can be used previously to allow an end-user who is not yet known to the system to create an account.

5.3. Select Item

Pattern Name: Select Item

Problem: The users want to select item(s) from a list.

Context: This pattern is used when a user should be able to select an item from a list of items to perform some action on it or when a user should be able to select multiple items from a list of items to perform some action on thoese items (e.g., delete multiple items)

Solution: There are two types of this pattern: Select One Item and Select Multiple Items. They have the same structure; that is one user interaction task "Select Item" to select one item from a list or "Select Multiple Item" to select multiple items. In the task "Select Multiple Item", of course, the user should have the option to select multiple items. This is specified in the object chunk associated with this task.

Rational: With this pattern, an item, respectively a list of items (depending on which type of pattern we use) can be selected by the user and will be returned.

Structure

Task Model

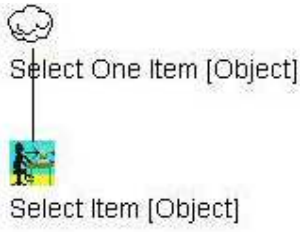


Figure 23. Task Model for the task "Select One Item"

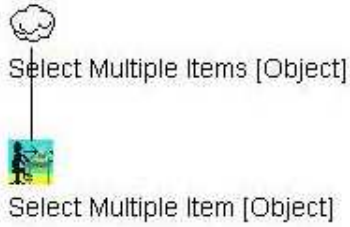


Figure 24. Task Model for the task "Select Multiple Items"

Object Chunks

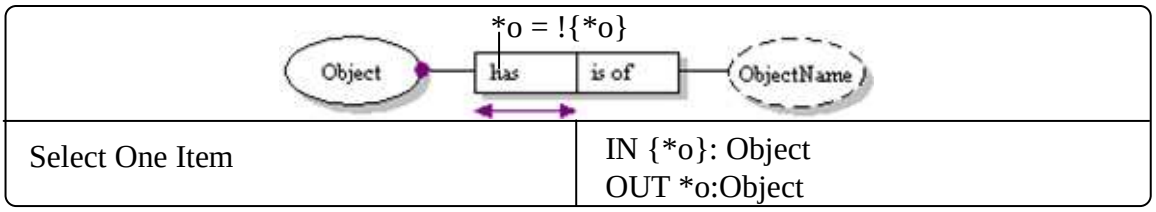


Figure 25. Object Chunk "Select One Item"

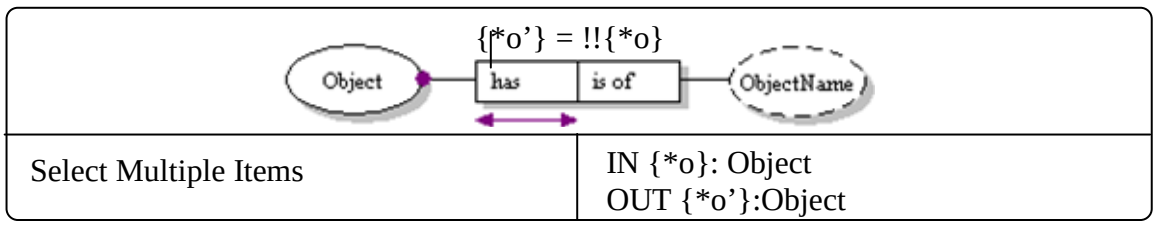


Figure 26. Object Chunk "Select Multiple Items"

Task Navigational Model

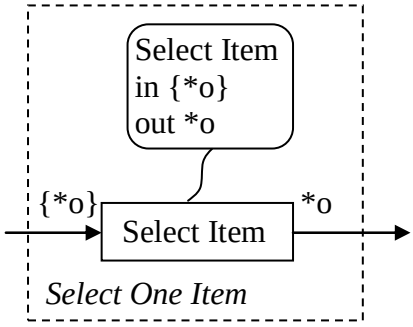


Figure 27. Task navigation for the task "Select One Item"

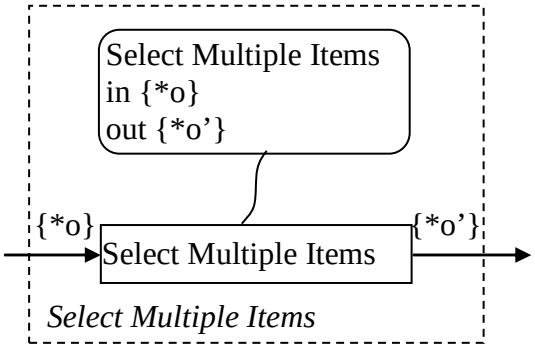


Figure 28. Task navigation for the task "Select Multiple Items"

Related Pattern:

5.4. Update Item Information Pattern

Pattern Name: Update Item Information

Problem: Sometimes, a user should be able to change or update the information available for an item.

Context: This pattern can be used when the item for which the information needs to be updated has already been identified.

Solution: Show the available information of the item to the user and provide the ability to update this information.

Rational: With this pattern, the information of items on the web site can be kept up to date.

Structure

Task Model

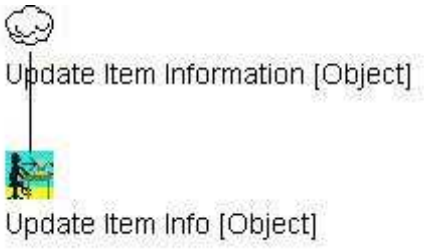


Figure 29. Task Model for the task "Update Item Information"

Object Chunks

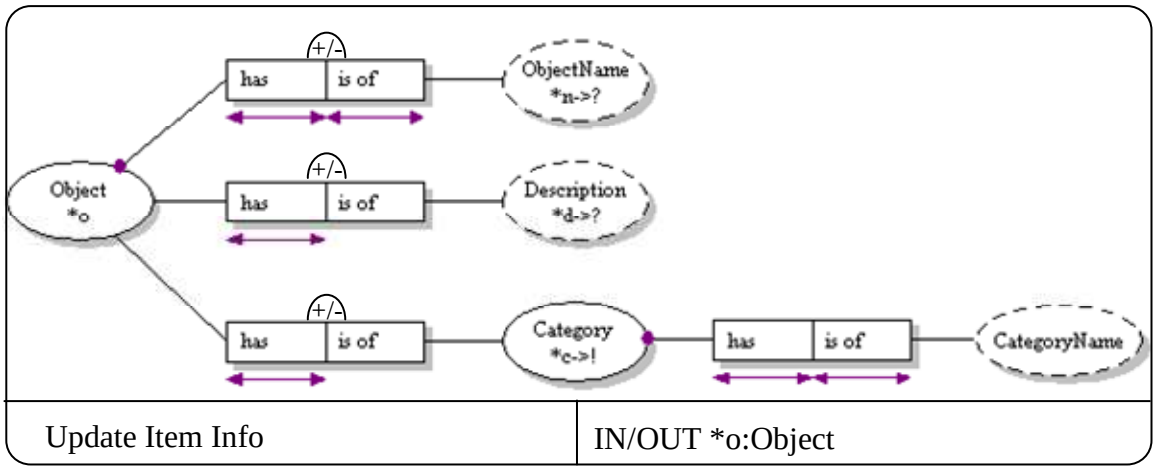


Figure 30. Object Chunk "Update Item Information"

Task Navigational Model

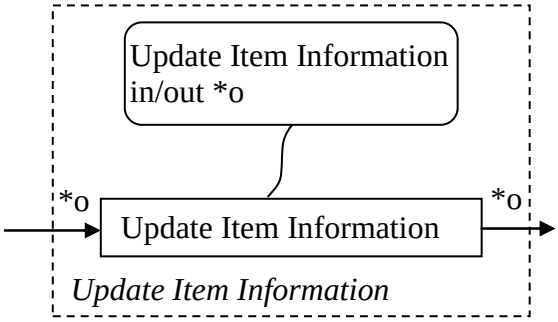


Figure 31. Task navigation for the task "Update Item Information"

Related Pattern:

Select Item: this pattern can be used to select the item for which the information should be updated.

5.5. Display Item Information Pattern

Pattern Name: Display Item Information

Problem: Showing detail information of (a) particular item(s) to the user.

Context: This pattern can be used when the item(s) for which the information needs to be shown has already been identified.

Solution: There are two types of this pattern: Display Item Information and Display Multiple Items Information. This task just simply shows some basic information of an Object(s). In a specific application, depending on the needs, other information can be added when adapting this pattern.

Rational: In order to gain information about the object(s), it must be inspected. An object is externally represented by its attributes and methods. Thus, object information must be derived from its attributes and methods

Structure

Task Model

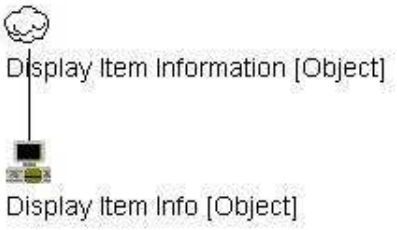


Figure 32. Task Model for the task "Display Item Information"

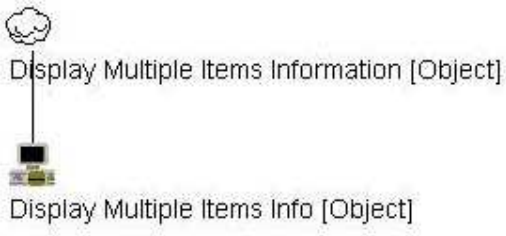


Figure 33. Task Model for the task "Display Multiple Item Information"

Object Chunks

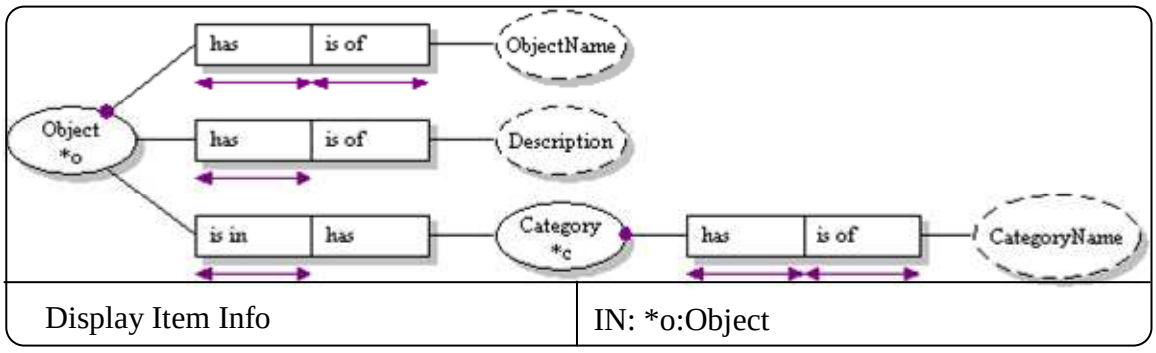


Figure 34. Object chunk "Display Item Information"

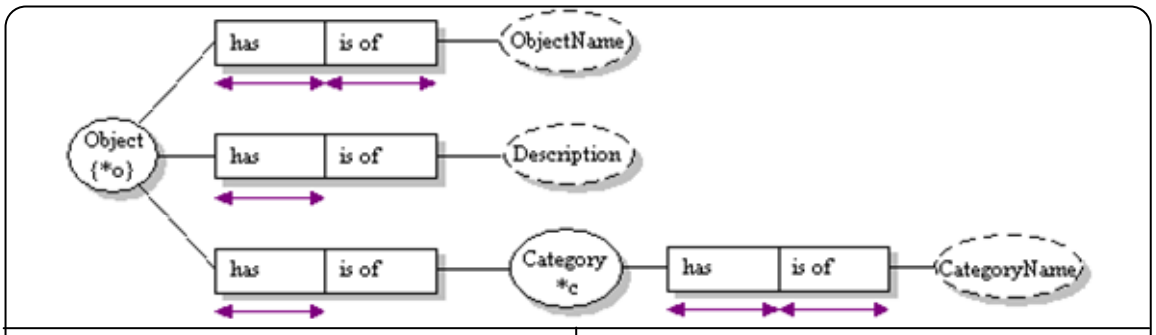


Figure 35. Object chunk "Display Multiple Item Information"

Task Navigational Model

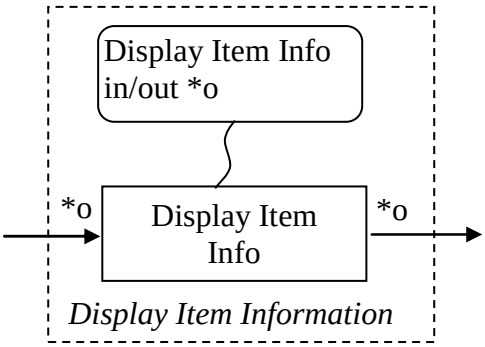


Figure 36. Task navigation for the task "Display Item Information"

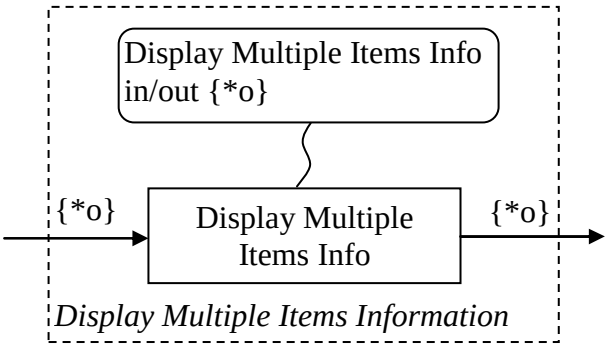


Figure 37. Task navigation for the task "Display Multiple Items Information"

Related Pattern:

Select Item: this pattern can be used to select the item(s) for which the information needs to be shown.

5.6. Confirm or Cancel Pattern

Pattern Name: Confirm or Cancel

Problem: For performing some important operation, you want to ask the user to either cancel or confirm the operation.

Context: This pattern is used to get the confirmation from the users before process an important operation. For example, before processing delete an item, the system should print out a message to notice users and get the decision from the users.

Solution: Provide the task in order to ensure that the users want to continue processing an important task (e.g., placing an order) by providing to the users a way to confirm or cancel the continuation of the process.

Therefore, in this task, the system displays a message to the user to ask him/her if he/she want to continue the process (task). If the user confirms, the system will perform the task, otherwise the system will skip this task.

Rational: By providing this pattern, we can protect the users from performing unwanted actions.

Structure

Task Model

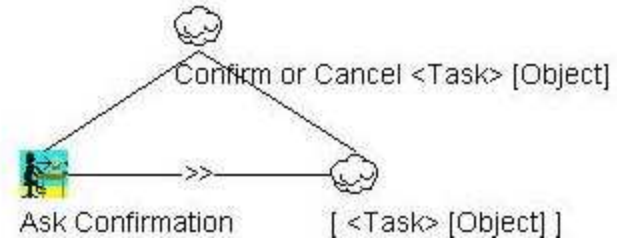


Figure 38. Task Model for the task "Confirm or Cancel"

Object Chunks

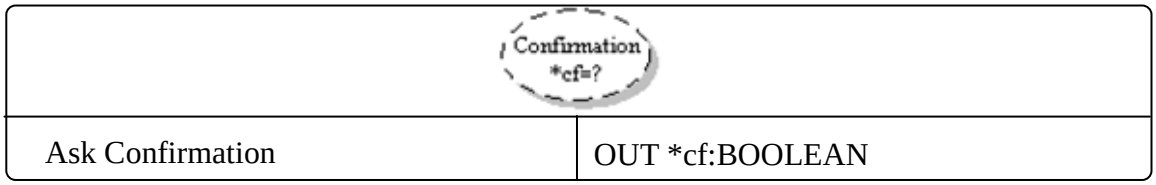


Figure 39. Object Chunk "Ask Confirmation"

Task Navigational Model

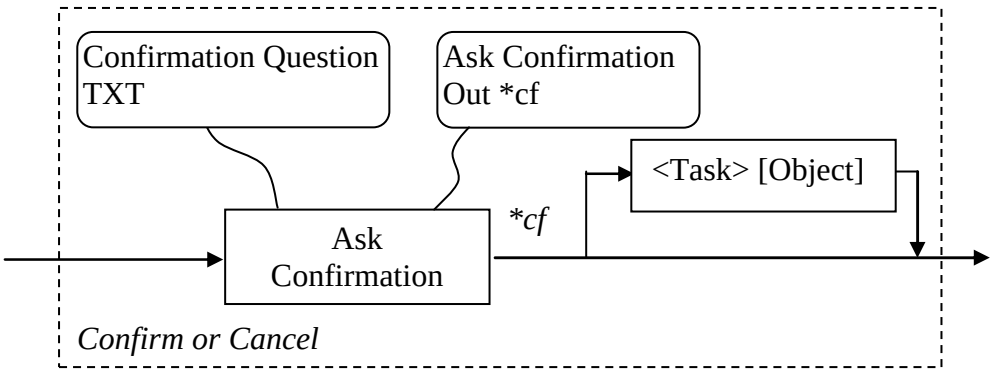


Figure 40. Task navigation for the task "Confirm or Cancel"

In this task navigation model, the task <Task>[Object] is modeled as a task navigation model, which will be derived from the task model for <Task>.

Related Pattern:

Add Item, Delete Item: these patterns can be used in combination with this pattern.

5.7. Add Item Pattern

Pattern Name: Add Item

Problem: Users want to add new content to the web system.

Context: This pattern can be used to save new content in a web system. For example, add new book (in e-commerce website), add new article or news, etc.

Solution: The user has to provide some information for the new object; the user can confirm to save this new object or cancel the operation, in which case no new Object will be created

Rational: Provide the users a task to create new content and also allow them to confirm that they want to add the new item to the system. This pattern is modeled as a transaction task so a new item is added if the users confirm their action, otherwise no item is add in the persistent store.

Structure

Task Model

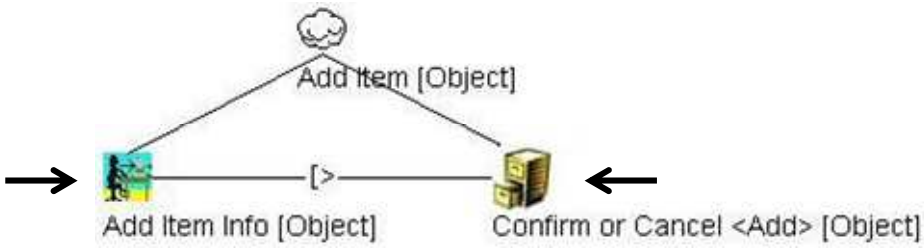


Figure 41. Task Model for the task "Add Item"

Object Chunks

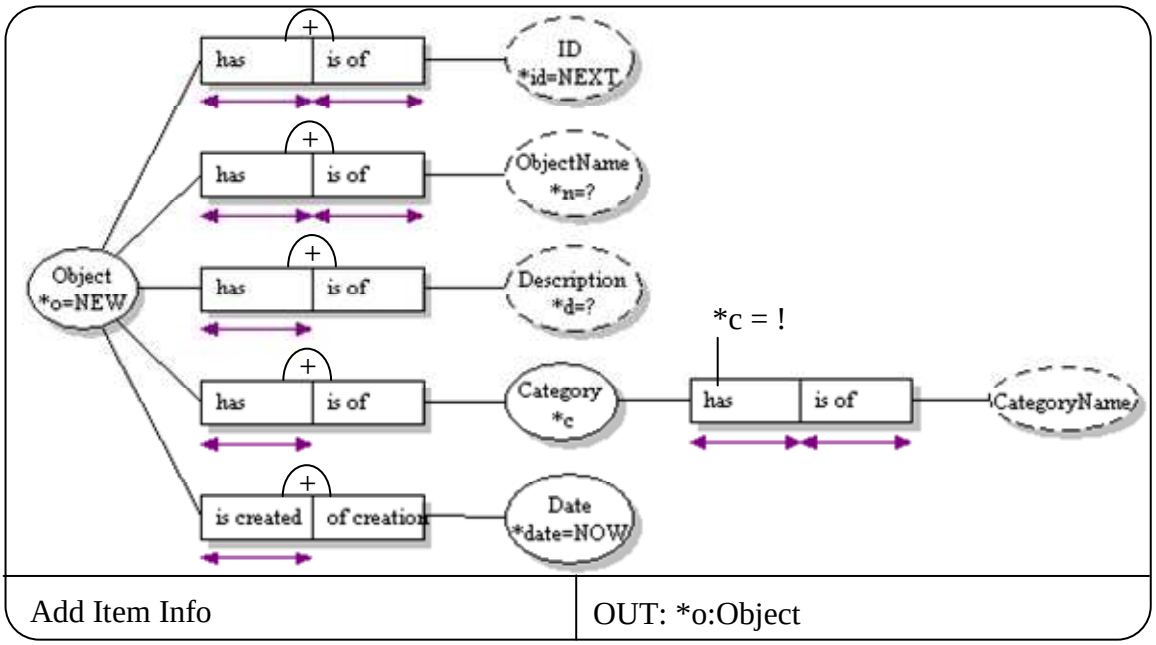


Figure 42. Object chunk "Add Item Info"

Task Navigational Model

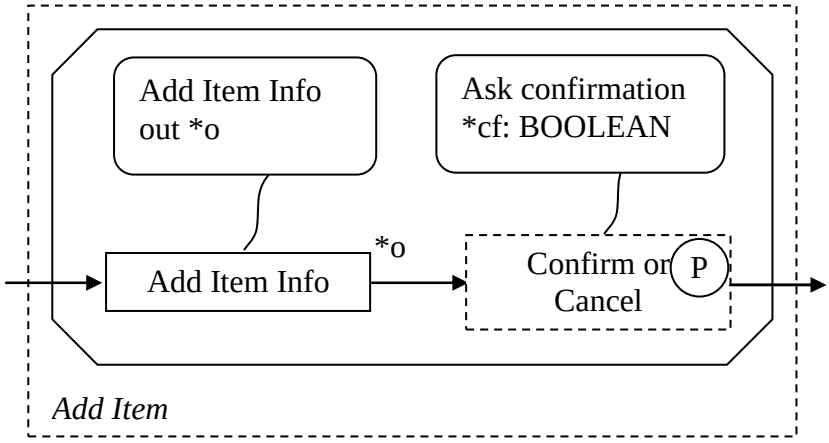


Figure 43. Task navigation for the task "Add Item"

Related Pattern:

Confirm or Cancel: this pattern is used to allow the user to confirm or cancel the add operation

5.8. Delete Items Pattern

Pattern Name: Delete Items

Problem: Users should be able remove items from the system.

Context: This pattern can be used to delete one or multiple items that are stored in the system.

Solution: There are two types of this pattern: Delete One Item and Delete Multiple Items. Provide users a task to remove unwanted items. This task is modeled as follow: the user first selects (the) item(s) that he wants to remove and then the Confirm or Cancel pattern is used to confirm or cancel the delete operation.

Rational: In a web system, there may be a lot of information. In order to keep the system up to date, it must be possible to remove items that are not needed anymore.

Structure

Task Model

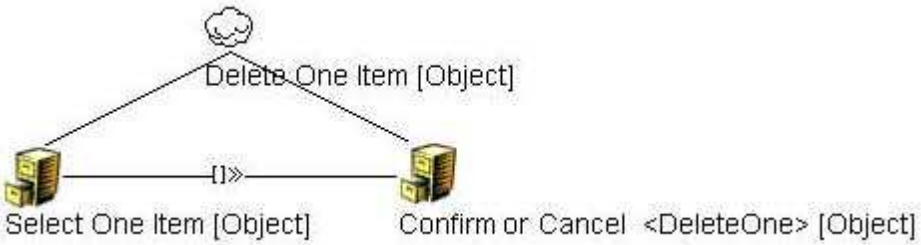
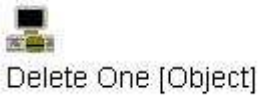


Figure 44. Task Model for the task "Delete One Item"

Where <DeleteOne>[Object] is the following task:



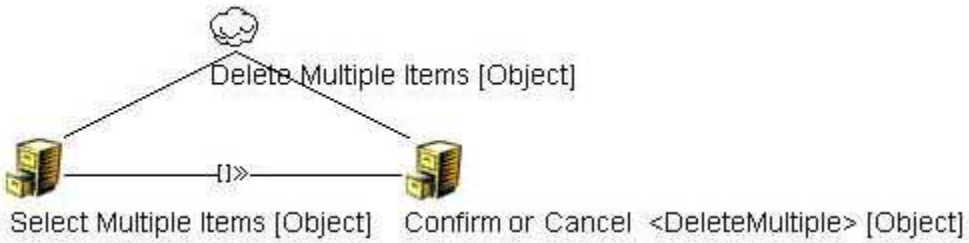


Figure 45. Task Model for the task "Delete Multiple Items"

Where <DeleteMultiple> [Object] is the following task:

Object Chunks

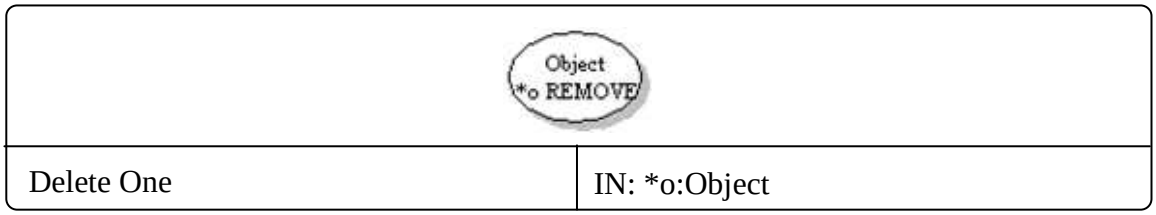


Figure 46. Object chunk "Delete One Item"

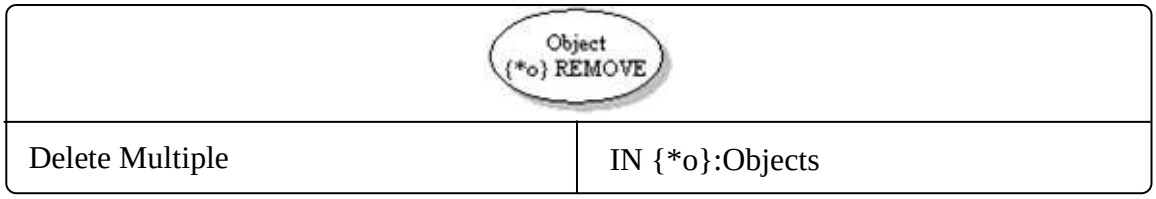


Figure 47. Object Chunk "Delete Multiple Items"

Task Navigational Model

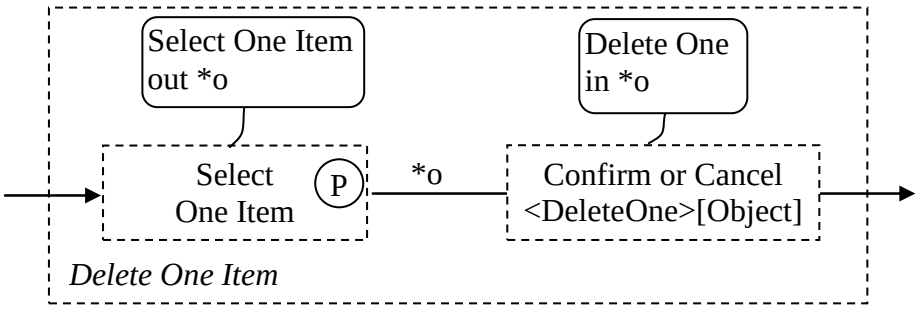


Figure 48. Task navigation for the task "Delete One Item"

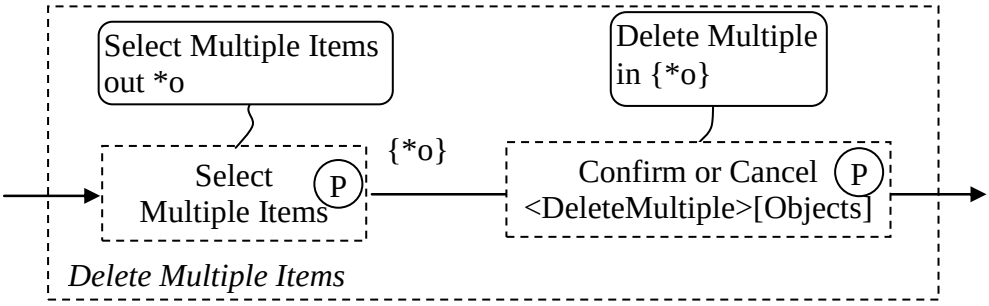


Figure 49. Task navigation for the task "Delete Multiple Items"

Related Pattern:

Confirm or Cancel: this pattern is used to allow the user to confirm or cancel the delete operation.

5.9. Show Results Pattern

Pattern Name: Show Results

Problem: Users need to view the results of some operations like the results of a search or browse.

Context: This pattern can be used to show information about one or more items.

Solution: Some information of the item itself is shown. As default the name and a description is shown, but depending on the concrete context of use, other elements of the item may be included such as a summary, location, category, author etc.

This task simply reuses the pattern "Display Multiple Items Information" to display all items from the result.

Rational: With this pattern the results from other operations can be presented to the user. For example when a user has done some search, e.g. using a Search Box or Advanced Search, they need to be able to inspect the results of this search.

Structure

Task Model

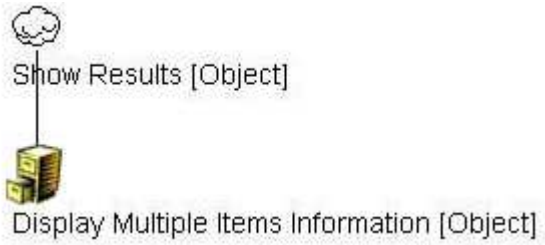


Figure 50. Task Model for the task "Show Results"

Object Chunks

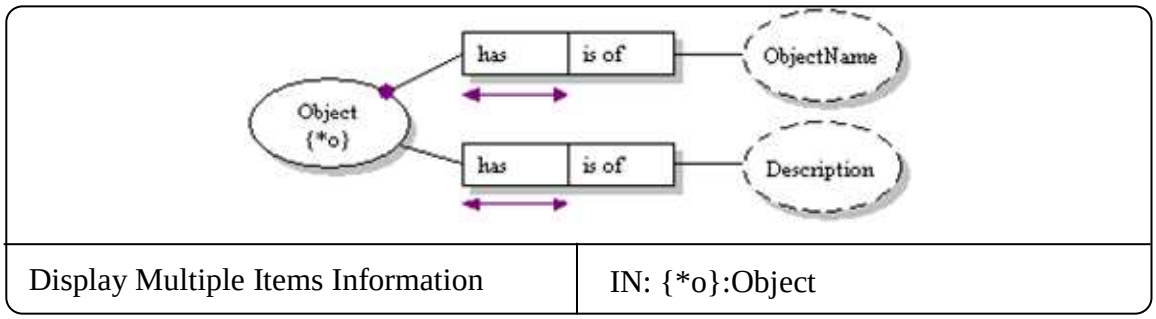


Figure 51. Object chunk "Display Multiple Items Information"

Task Navigational Model

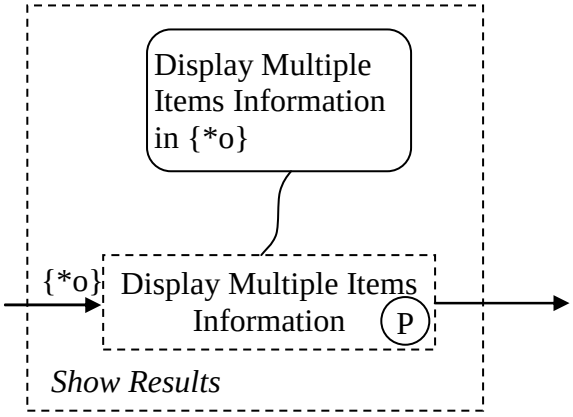


Figure 52. Task navigation for the task "Show Results"

Related Pattern:

Simple Search, Advanced Search, Browse: these patterns can be predecessors of this pattern.

5.10. Browse Pattern

Pattern Name: Browse

Problem: The users need to find an item or specific information.

Context: The pattern is applicable for allowing searching for items that are grouped into a set of categories. By selecting a category the users can view a subset of items that belong to the selected category. In this case, the browse pattern is more suitable than the search pattern.

Solution: List the categories and give the user the possibility to select a category. When a category has been selected, the subset of items that belong to the selected category are shown.. The *Show Results* pattern is used to show information on the item's attributes.

Rational: The search for items is narrowed by first allowing to select a category.

Structure

Task Model

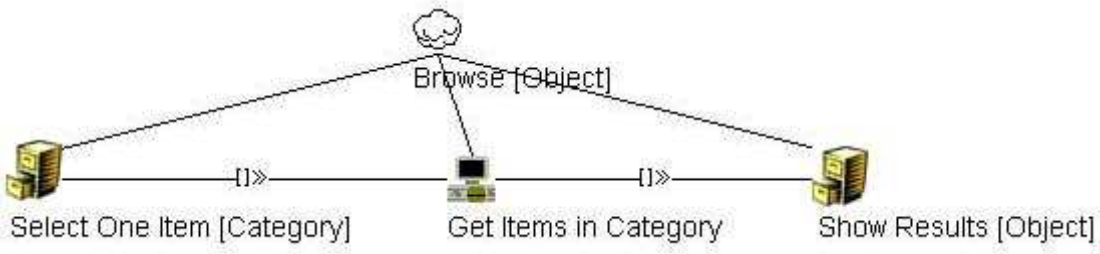


Figure 53. Task Model for the task "Browse"

Object Chunks

See the object chunks in the patterns Select One Item and Show Result

Navigational Model

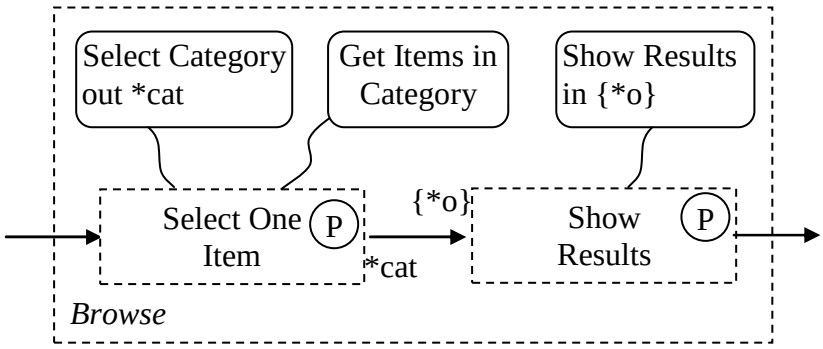


Figure 54. Task navigation for the task "Browse"

Related Pattern:

Simple Search, Advanced Search: these are alternative patterns

5.11. Simple Search Pattern

Pattern Name: Simple Search

Problem: The users need to find an item or specific information.

Context: This pattern can be used if you want to give the possibility to the user to search for information using some keywords.

Solution: From the keywords that the user provides, the search engine will create the query (this is done by using the OR functionality) and execute the query. Next the results from the search engine are presented to the users. The task pattern "Show Results" is used to show the results.

Rational: The desired information could be found quickly and presented to the users.

Structure

Task Model

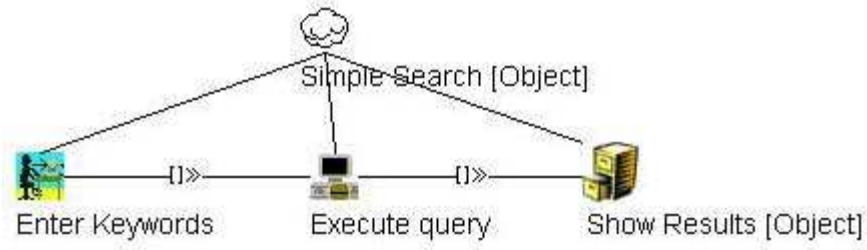


Figure 55. Task Model for the task "Simple Search"

Object Chunks

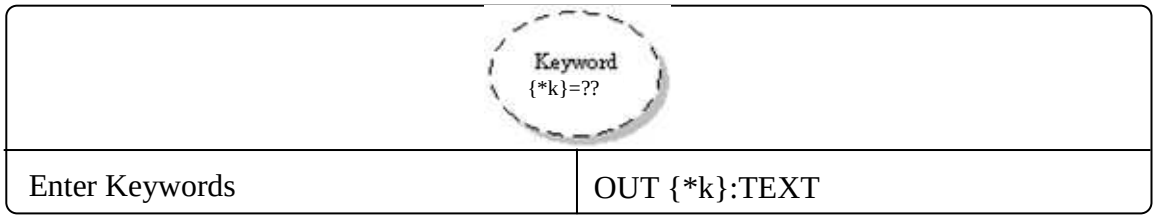


Figure 56. Object Chunk "Enter Keywords"

Task Navigational Model

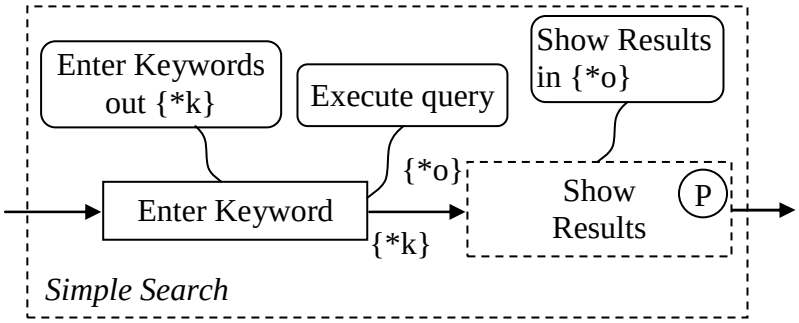


Figure 57. Task navigation for the task "Simple Search"

Related Pattern:

Browse, Advanced Search: these are alternative patterns.

Show Results: this pattern is used to show the results of the search to the user.

5.12. Advanced Search Pattern

Pattern Name: Advanced Search

Problem: The users need to find an item or specific information.

Context: This pattern can be used if you want to give the possibility to the user to search for information using sophisticated search criteria.

Solution: Allow the user to enter complex search criteria. The exact format of the search criteria is dependent on the context of use and should be specified when adapting the pattern to the context of use.

Rational: In some cases, the user is not satisfied with a simple search to find the required information. The advanced search offers a solution to this by adding some more search options. This kind of functionality e.g., controls how the list of search terms is interpreted by the search engine. Typically this includes AND/OR functionality together with exclusion functionality.

However, the desired search options may vary between applications. For example, we can add to the advanced search some options in terms of item types (articles, video, audio...), or in terms of item properties (title, date, location, size, author...). Together with the Search pattern, the Advanced Search can be used in any web site to provide the users a flexible way to find items that they want.

Structure

Task Model

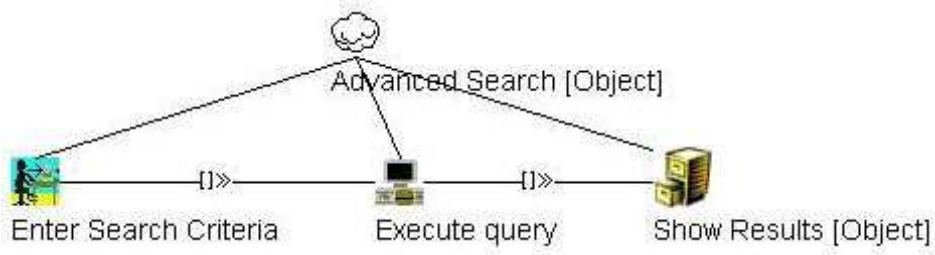


Figure 58. Task Model for the task "Advanced Search"

Object Chunks

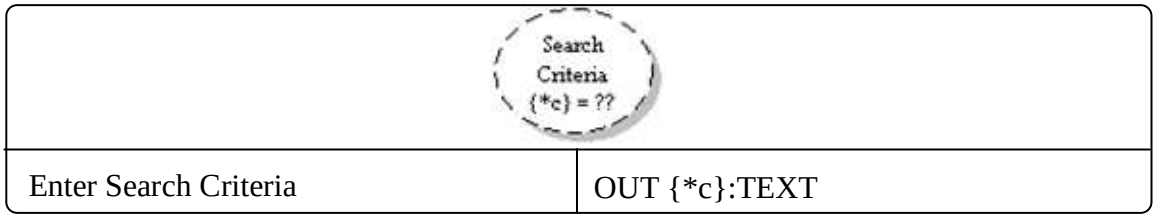


Figure 59. Object Chunk "Enter Search Criteria"

Task Navigational Model

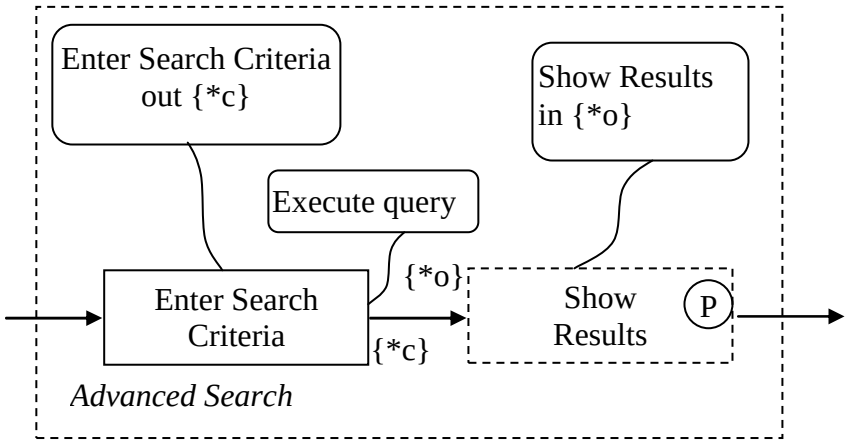


Figure 60. Task navigation for the task "Advanced Search"

Related Pattern:

Browse, Simple Search: these are alternative patterns.

Show Results: this pattern is used to show the results of the search to the user.

6. Case Study

In this chapter, as the proof of concept, a case study is presented. An online Bookshop system is created using WSDM and task patterns. In order to clarify and illustrate the ideas of my approach and its practical relevance I will present step by step how to develop this web system using WSDM.

6.1. Mission Statement

The mission statement of this web application can be formulated as follows: Provide book information online to attract more customers, improve the bookshop management, and get more benefit from online business by providing the ability to users to search, view, select, and buy books online.

6.1.1. Purpose

- To make it easier for users to find, select, and buy books
- Support online buying
- To manage the book shop better

6.1.2. Target Audience

- Customers
- Manager

6.1.3. Subject

- Books

6.2. Audience Modeling

To find the audience classes, we follow the method explained in section 3.3.1.

6.2.1. Activities

With the bookshop online system, the following activities are identified; they are all related:

1. Provide books information Related
2. Provide buying online Related

3. Manage customer profile Related
4. Provide reports (customers, books, sales) Related

6.2.2. People involved

For the activities that are identified above, the people involved in these activities are customers and manager. The following figure depicts the relationship between these groups of people and the activities.

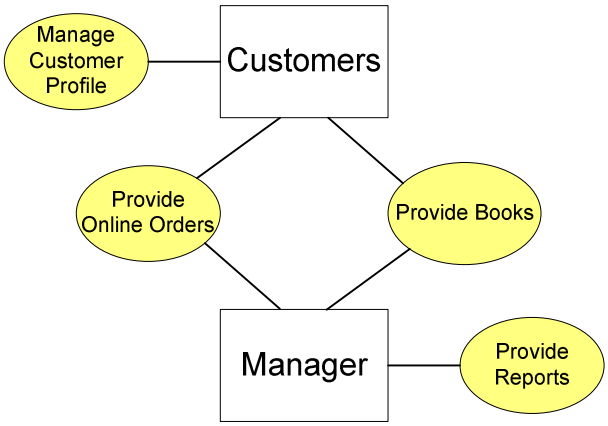


Figure 61. The activities and people involved

The information and functionality requirements for each of these groups are provided as follows.

6.2.2.1. Customers

Information requirements:

- Detailed books' information
- History orders' information
- Customer's profile information

Function requirements:

- Login
- Create customer profile
- Change his/her profile information
- Search book
- Add a book to the shopping cart

- Manage his shopping cart.
- Place order

Navigation requirements:

- Flexible ways to find books

6.2.2.2. Manager

Information requirements:

- Book information
- Book category information
- Information about customers
- Information about books
- Information about sales

Function requirements:

- Login
- Create book category.
- Create new book.
- Manage book categories.
- Manage books.
- Manage orders

6.2.3. Audience classes

From this analysis we can identified the following audience classes:

1. Customer: People who want to get detail information about books before buying.
2. Manager: People who have the responsible to manage books, book categories; manage and get the reports about customer, orders, and sales from the website.

Furthermore, we have the Visitor audience class with the following requirements:

Information requirements:

- Information about the book shop
- Detailed information about books

Function requirements:

- Search book
- Add a book to the shopping cart.
- Manage his shopping cart.
- Register as customer

Navigation requirements:

- Flexible ways to search books

6.2.4. Audience class hierarchy

The audience class hierarchy is shown in Figure 62 with the Visitor class on the top of the hierarchy to presents all the users of the system.

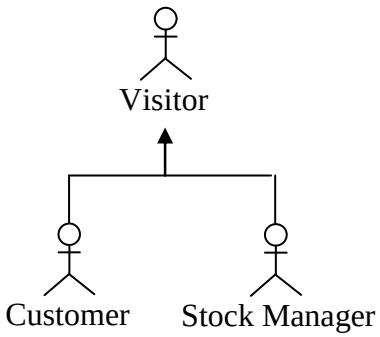


Figure 62. Audience Class Hierarchy for the Bookshop online

6.2.5. Audience Class Characterization

In this second sub-phase of the audience Modeling, the characterization of each audience class is specified as follows.

6.2.5.1. Customer:

- All age
- Have experience with www
- Have knowledge about payment online
- Language: English

6.2.5.2. Manager:

- Age: from 18-60 years old

- Have experience with www
- Have knowledge in business workflow
- Language: English

6.2.5.3. Visitors:

- Age: from 18-60 years old
- Experience with www: variety
- Language: English

6.3. Conceptual Design

With the approach that we have presented in chapter 4, in this phase we will use patterns to model the tasks of the users.

In order to save space for other parts of this thesis, only the task *Create new Book* is presented in detail as an illustrate example for the approach.

6.3.1. Task and Information Modeling

Suppose we have a library of task patterns at our disposal (see chapter 5). In order to save space for other parts in this thesis, two examples of using task patterns in this phase are presented: “Create new Book” and “View history Orders”.

6.3.1.1. Create new Book

When designing the task “Create new Book” the designers decide to use suitable patterns to model this task.

The task “Create new Book” can be modeled by using the pattern “Add Item” in which the variable “Object” will be replaced by the Book object. This task model is presented in the following figure.

Figure 63. Task Model for the task “Create new book”

After selecting the suitable pattern to model our tasks (in this case the "Add Item" pattern), the object chunks and the task navigational model in the selected pattern need to be adapted to the current context of use. The next step will present these adaptations.

In the pattern "Add Item [Object]" there is an object chunk "Add Item Info" associated with its subtask "Add Item Info [Object]".

In our application we will adapt this object chunk for the "Book" object. The Book object not only has some properties like the name, the description… but also has some other properties like the author, the price, the category… of the book. After deciding which information is needed for the Book object the designers will create the object chunk for this task. The object chunk "Add Book Info" which is adapted from the object chunk "Add Item Info" is given in figure 20.

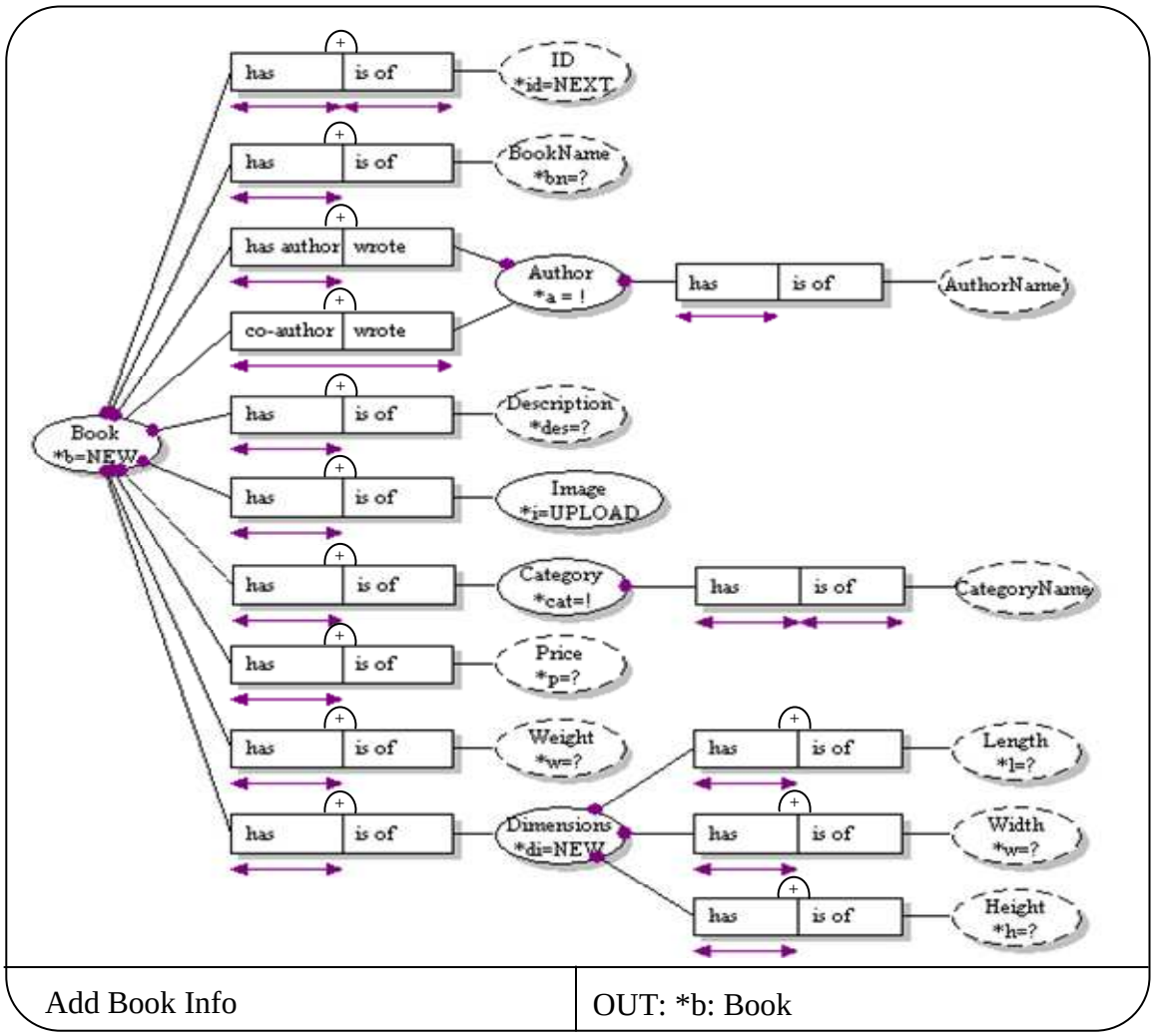


Figure 64. Object Chunk "Add Book Info"

In the structure of the pattern “Add Item”, another pattern is used to model the task “Confirm or Cancel”. The pattern “Confirm or Cancel <Task> [Object]” has one object chunk: “Ask Confirmation” associated with its subtask “Ask Confirmation”. Although this object chunk does not need to be adapted, we do present this object chunk here:

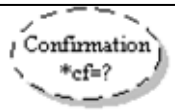
	
Ask Confirmation	OUT *cf:BOOLEAN

Figure 65. Object Chunk “Ask Confirmation”

After adapting the object chunks, the patterns in the task model could be expanded to present the task in full detail. The fully instantiated task model for the task “Create new Book” is given in the following figure.

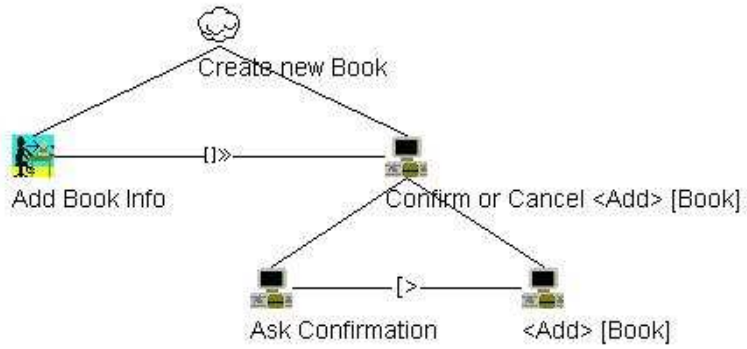


Figure 66. The fully instantiated task model for the task “Create new Book”

After the task modeling step, the task navigation model should be created. For the pattern “Add Item” we have created the task navigational model to present the navigation in this pattern (see figure 23).

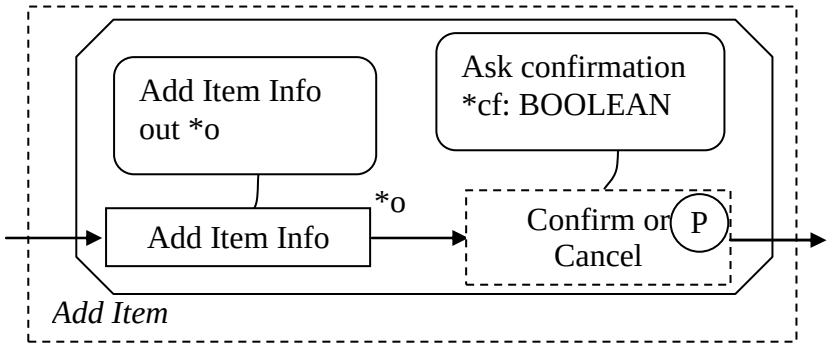


Figure 67. Task navigation for the task “Add Item”

In our application, the task navigational model for the task “Create new Book” will be created from this task navigational model. The concrete object “Book” will replaces the variable “Object” in the pattern, meanwhile the “concrete” object chunks we have created will replace the generic object chunks in the task navigational model.

The result of this step is the “adapted” task navigational model for the task “Add Book” and is presented in the following figure.

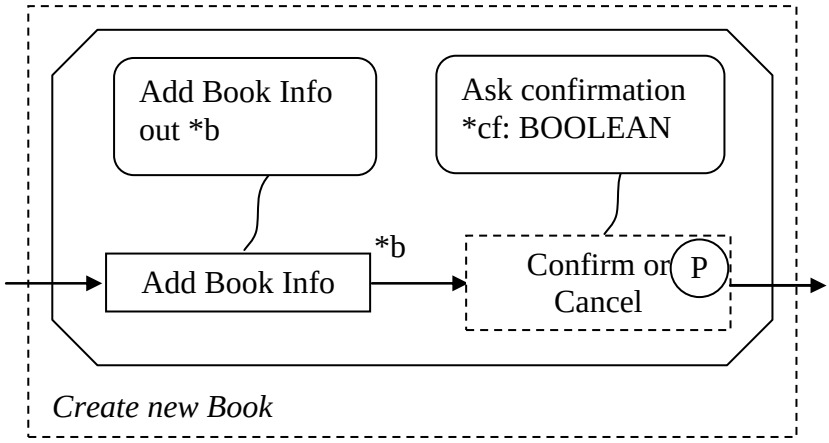


Figure 68. Task navigation for the task “Create new Book”

6.3.1.2. View history orders

In this task, the system lists all history orders to users. After that, optionally, users can select one order to view its content. The “Select One Item [Object]” and “Display Item Information [Order]” patterns are used to present the tasks “Select an Order” and “Display Order Info”. The task is modeled as in the following figure.

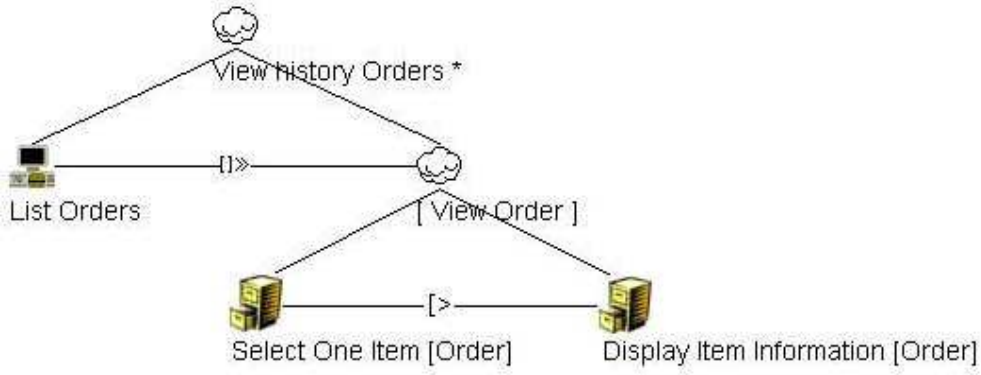
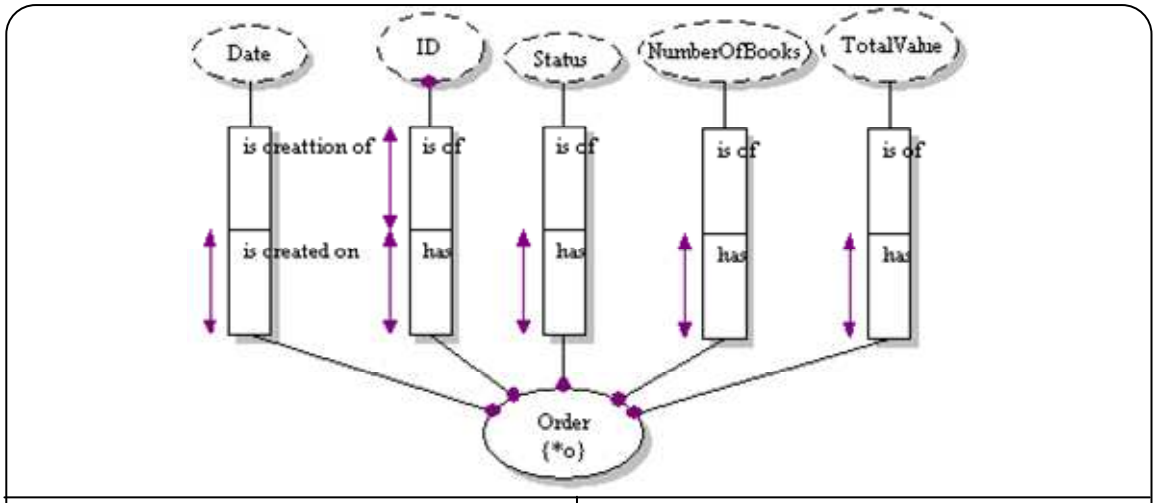


Figure 69. Task Model for the task “View history Orders”

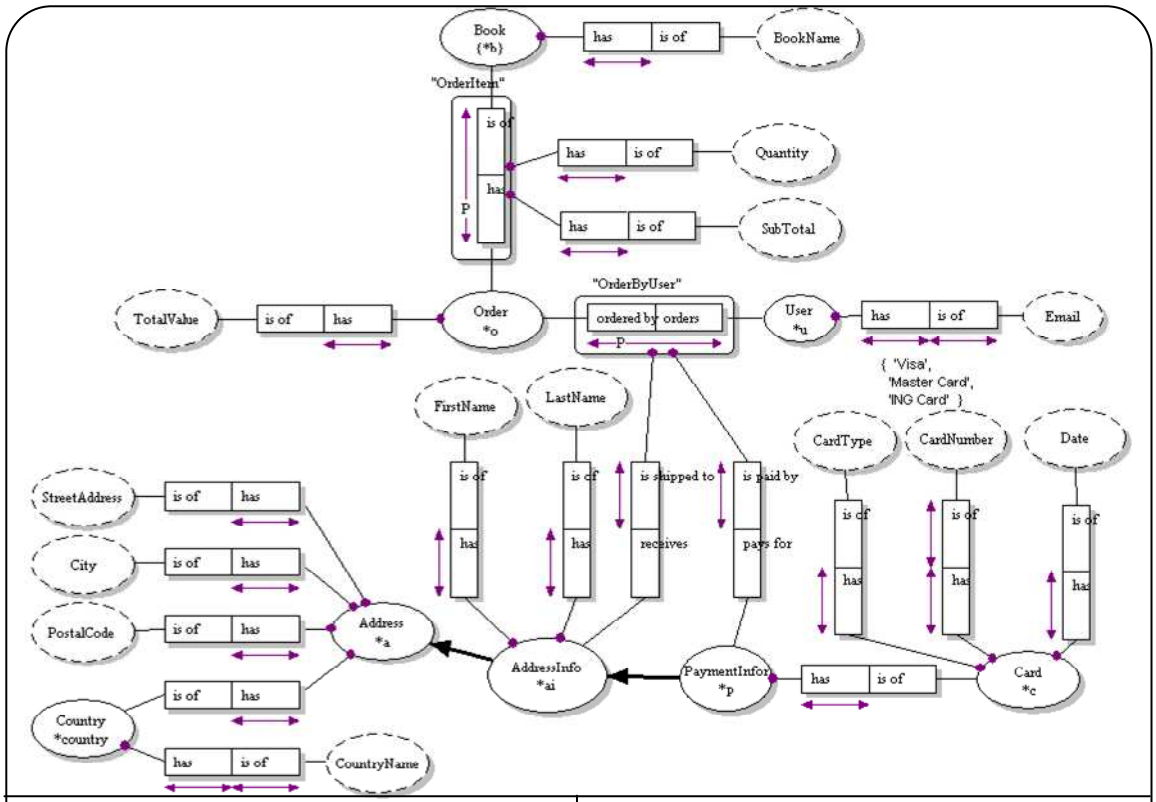
The following part presents all the object chunks associated within this task. Only the object chunks “Select an Order” and “Display Order Info” are adapted respectively

from the object chunks “Select Item” (from pattern “Select One Item”) and “Display Item Info” (from pattern “Display Item Information”).



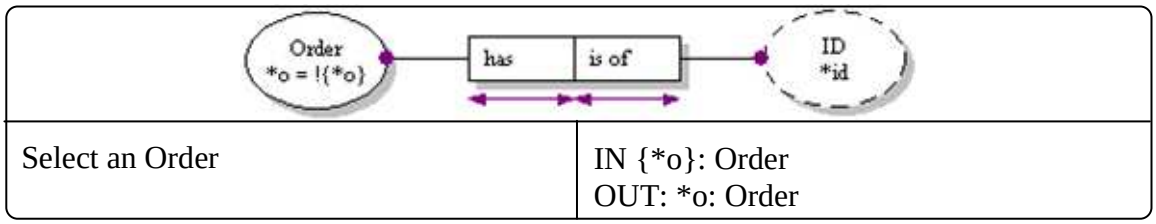
List Orders	IN: {*o}: Order

Figure 70. Object Chunk “List Orders”



Display Order Info	IN: *o: Order

Figure 71. Object Chunk “Display Order Info”



Select an Order	IN {*o}: Order OUT: *o: Order

Figure 72. Object Chunk "Select an Order"

After adapting the object chunks, the patterns in the task model could be expanded to present the task in full detail. The fully instantiated task model for the task "View history Orders" is given in the following figure.

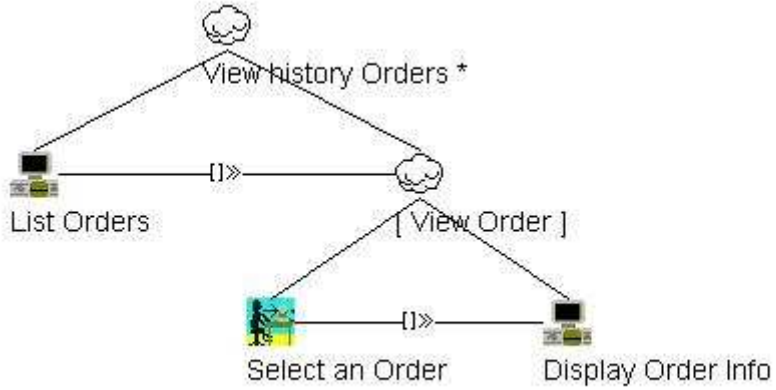


Figure 73. The fully instantiated task model for the task "View history Orders"

Finally, the "adapted" task navigational model for the task "View history Orders" is presented in the following figure.

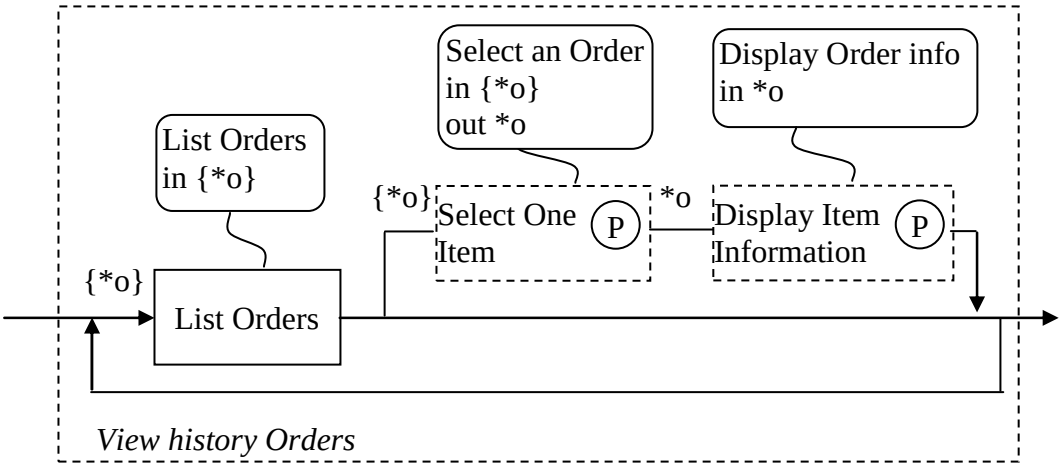


Figure 74. The task navigational model for the task "View history Orders"

6.3.2. Navigational Design

6.3.2.1. Conceptual Structural Model

From the hierarchy of audience classes we derive the conceptual structural model as follow.

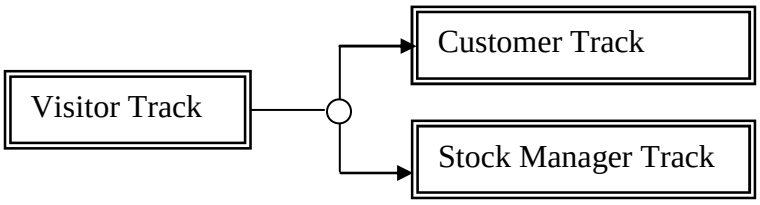


Figure 75. Conceptual Structural Model

6.3.2.2. Audience Navigational Tracks

For each audience class a navigation track is created. From all the *task navigation models* for the audience class the navigation track is composed. The following figure reflects the Visitor's navigational track.

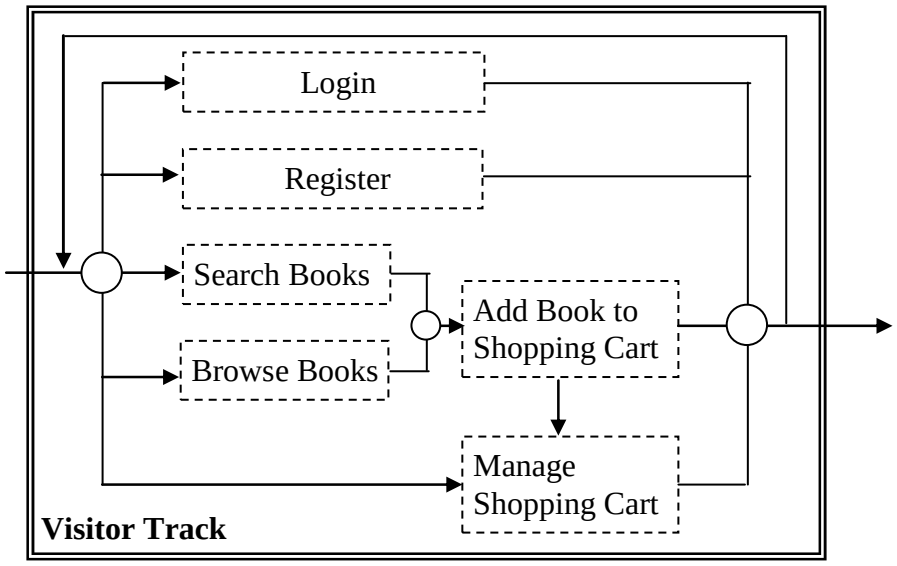


Figure 76. Visitor Navigational Track

The audience navigational track presents the navigation that the members from that class can perform. For example, in the navigational track above, when users finish the task "Add Book to Shopping Cart" they can navigate to the task "Manage Shopping Cart" to edit, remove books from the cart or they can go back to do other tasks.

6.3.2.3. Navigation Structure Design

From all the audience navigation tracks we created in the previous step, the main conceptual navigation structure of the website is combined as the following.

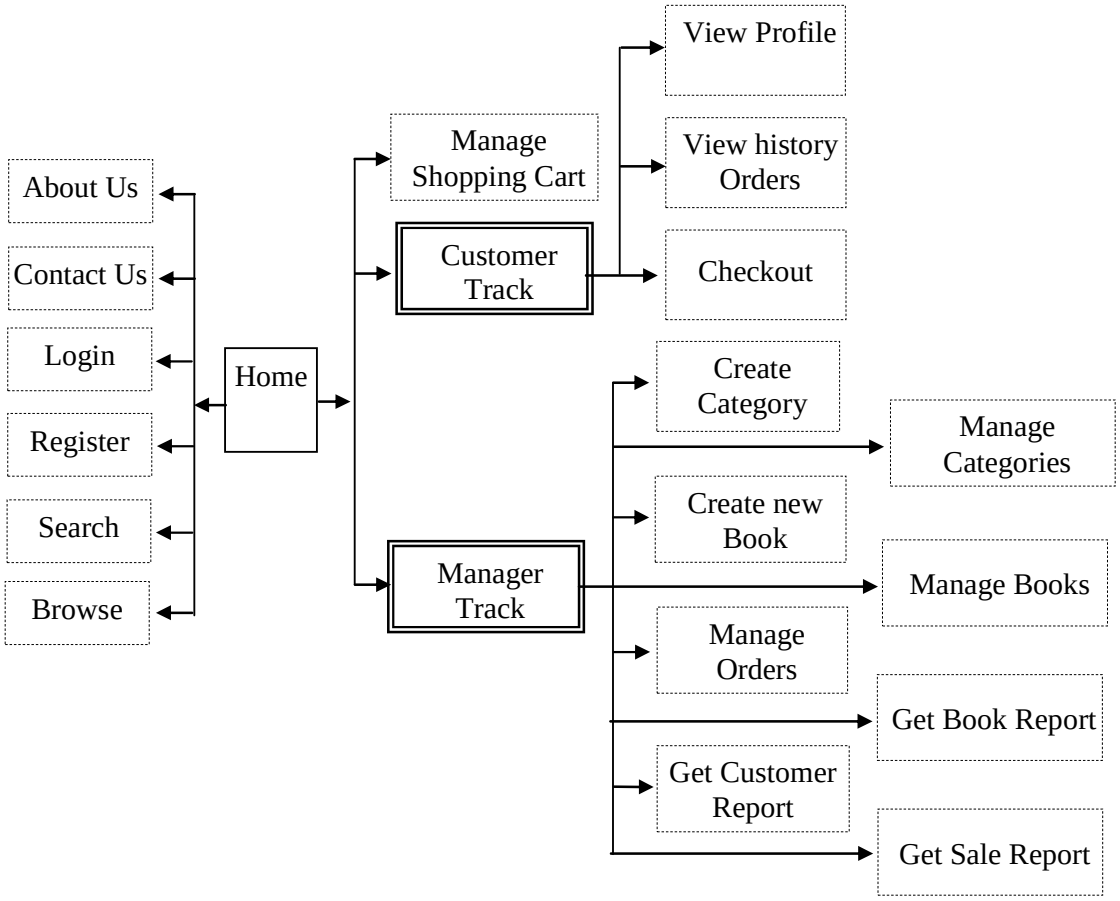


Figure 77. Main Conceptual Navigation Structure

6.4. Implementation Design

6.4.1. Site Structure Design

By default, each component is placed on a page. The site structure of the web application is designed as follows.

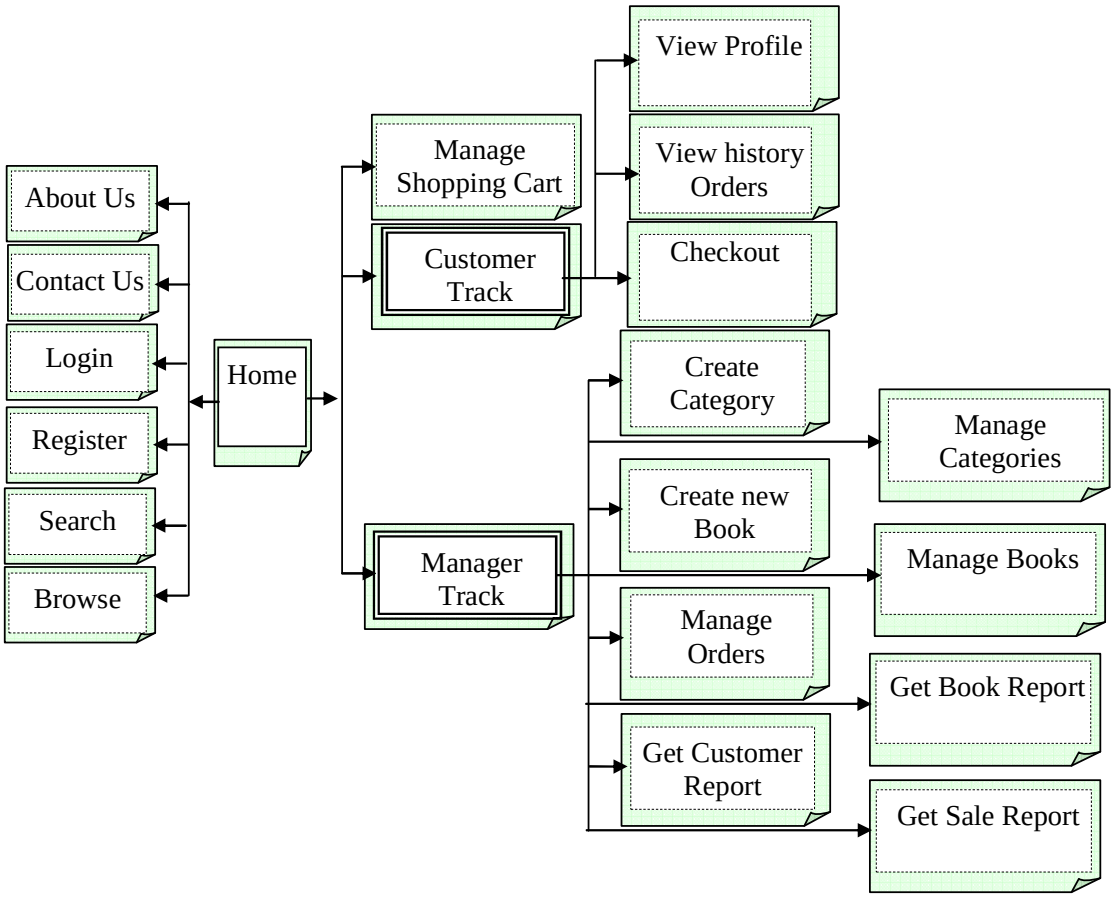


Figure 78. Site Structure Design

6.4.2. Presentation Design

The general template for each webpage is designed with the following elements:

Header

- Logo
- Primary navigation
- Secondary navigation

Sidebar

- Left sidebar
- Right sidebar

Content panel

- Bread crumb trail
- Editable region

For the style, CSS is used. The web elements will have it own color, look and feel… which is consistent through out the website.

The page layout is given in the figure below.

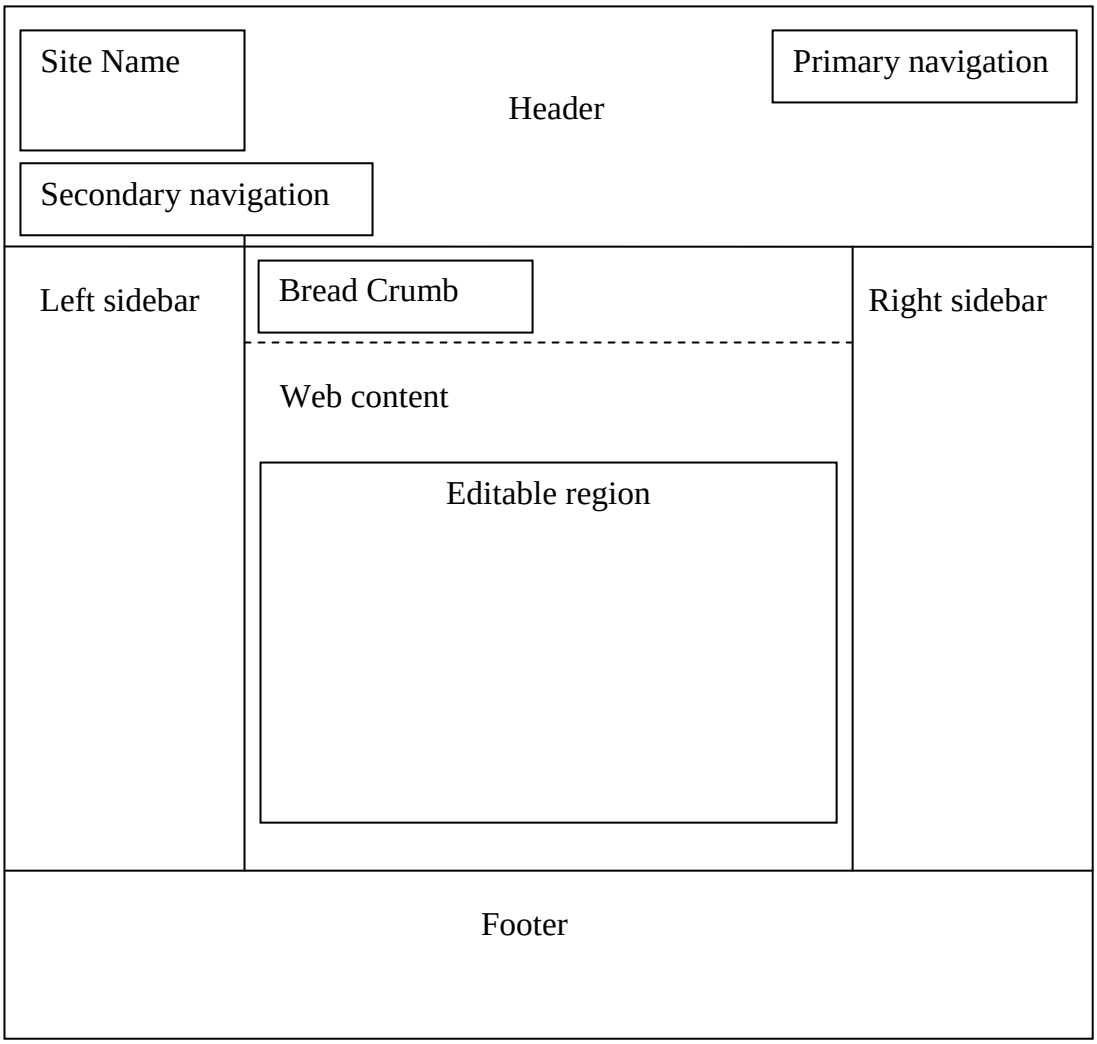


Figure 79. The Presentation Design

6.4.3. Logical Data Design

The object chunks that are created in the Conceptual Design phase are combined to create the business object model. From this business object model, the data schema is generated and then mapped into the database to store the information of the web system

6.5. Implementation

From the different models created during the different design phases, the website can be created.

The following figure shows the homepage of the website.

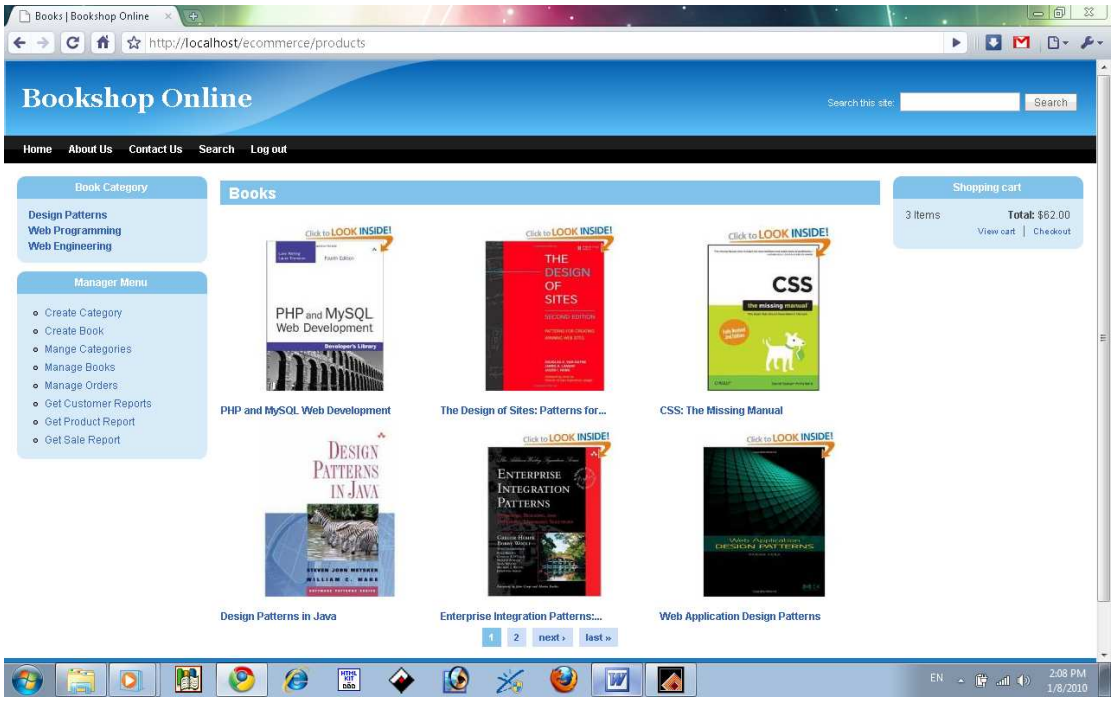


Figure 80. The homepage of the website

7. Related Work

There are many fields in computer science that have adopted the concept of pattern and use them as a solution for re-using. In this part, I will present some of researches and studies most related to task patterns.

In (Breedvelt, Paternò et al., 1997) the idea of using task patterns in order to accelerate the design of task models is addressed. When using the CTT notation to design tasks, the authors realize that some (part of) tasks were recurring in the task structure. CTT is a hierarchy structure so it is easy to separate a part and reuse it. Whenever the designers face with a problem that is similar to one problem that has been found and solved before then they can reuse the solution previously developed. In particular, the task patterns are used as templates for designing the task model of an application. The authors also introduced some of these task templates:

- Multi-Value Input Task
- Search Task
- Evaluation Task
- Recursive Activation Task

In (Paternò, 2000), the term task pattern was coined. Paternò examined the idea of reuse task fragments by using patterns. He introduced a metapattern for task patterns, which is very similar to the metapattern of the GoF. It includes the name, the problem it addresses, the task specification in CTT and the specification of the objects manipulated by the tasks. It is noteworthy that the specification of a task pattern is done in the regular CTT notation and thus has no generic part, which can be adapted to the current context of use. Therefore, the task pattern, as mentioned above, is just a model fragment and a description in which situation it is useful.

The most promising approach of task patterns was done by Sinnig (Sinnig, 2004). He introduced the concept of task pattern as a generic model fragment, which can be customized to the current context of use. Thus, he sees task patterns very similar to the software design patterns of the GoF, but in another domain. He collected a set of patterns, which were partly adopted from the patterns introduced in (Breedvelt, Paternò et al., 1997) and described the patterns in a structured way very similar to Paternò (Paternò, 2001) and Gamma (Gamma, Helm et al., 1995) with the following elements:

- Name
- Problem
- Context
- Solution
- Rational
- Interface
- Related Pattern

My study has some points that are adopted from the above researches but due to the characteristic of WSDM it was necessary to adapt the ideas at several points.

The idea of using the variable “Object” in each pattern as a placeholder for generic object type that can be used within the pattern is adopted from (Sinnig, 2004). This approach is differ when compare to Paternò, since the task pattern is just a fragment of tasks grouped together to achieve some purpose. With the use of a variable as placeholder in patterns the task pattern becomes a generic task and can be used in different contexts of use in which the object will be adapted and instantiated,

Design pattern is used in many fields. Each one considers the context in a different way so there is no common format for task description. In WSDM we need to specify not only the task structure but also the object chunks and the task navigational model. I have added these elements to form the task description format as discussed in chapter 4.

8. Conclusions

In WSDM, task models are used to effectively model the interaction between the users and the application. Designing task models requires a lot of time and experience. This thesis has introduced task patterns as a vehicle supporting a more disciplined and comprehensive form to promote the reuse of tasks in WSDM. I created and used several patterns to represent task models in a generic way as well as to use them in a concrete context of use.

A major contribution of this research is the integration of design patterns into the Conceptual Modeling phase of WSDM. A process for creating and using task patterns has been proposed. The structure of a task pattern in the context of WSDM has been defined and the process of using patterns has been elaborated. I have shown how patterns can be used as building blocks for the creation of tasks models. Furthermore, I have introduced the “pattern component” (a component presents the pattern in the navigational designed), which can be used to further speed up the design by representing the pattern as a high level abstract component in the task navigational model. In that case there is no need to instantiate the patterns in the task models before transforming them into a task navigational model.

Another contribution is a first list a task patterns. During the design process in my case study, I found these patterns occuring in many different situations. It may be needed to improve and extend these patterns later on. However, creating these task patterns is the proof of concept that we can indeed create and use design patterns in WSDM as proposed in the approach I have introduced in the Chapter 4.

My research has answered some questions about the role of patterns in the WSDM development. However, it has led to some other issues that still needs to be further investigated:

Tool support. The current CTT tool does not support the notation for the “task pattern” type (as we know that CTT only supports four different types of tasks: Abstract, Interaction, Application, and User). An extended tool (from CTT or Microsoft Visio, for instance) is needed to support the designers in creating and using task patterns during task modeling.

Platform dependency. WDSM is a systematic methodology to design web system; it does not only support desktop PC platform but also other kinds of devices such as PDA

and mobile phones. In this thesis, the process of integrating design patterns in WSDM is assumed, by default, in PC platform. Therefore, the second issue that needs to be further investigated is designing task patterns that support different devices and platforms.

References

Abi-Aad, R., D. Sinnig, T. Radhakrishnan and A. Seffah (2003). CoU: Context of Use Model for User Interface Design. In Proceedings of HCI International 2003, June 2003, Greece, LEA, pp. 8 - 12.

Ahmed Seffah and P. Forbrig (2002). "Multiple User Interfaces: Towards a Task-Driven and Patterns-Oriented Design Model." Proceedings of the 9th International Workshop on Interactive Systems Design, Specification, and Verification.

Alexander, C. (1979). The Timeless Way of Building, Oxford University Press.

Alexander, C., S. Ishikawa, et al. (1977). A Pattern Language: Towns, Building, Construction. New York, Oxford University Press.

Borchers, J. (2001). A Pattern Approach to Interaction Design. New York, John Wiley & Sons.

Breedvelt, I., F. Paternò and C. Severiins (1997). Reusable Structures in Task Models. In Proceedings Design, Specification, Verification of Interactive Systems '97, June 1997, Granada, Springer, pp. 251-265.

Coplien, J. O. and D. C. Schmidt (1995). Pattern Languages of Program Design, Addison-Wesley Professional.

De Troyer, O. (1998). Designing Well-Structured Web Site: Lessons to be Learned from Database Schema Methodology. In Proceedings of the ER'98 Conference, Lecture Notes in Computer Science: Springer-Verlag.

De Troyer, O. and C. Leune. (1998). WSDM: A User-Centered Design Method for Web Sites. In Computer Networks and ISDN systems, Proceedings of the 7th International World Wide Web Conference: Elsevier: 85-94.

De Troyer, O., Casteleyn, S., Plessers, P. (2005). Using ORM to Model Web Systems, On the Move to Meaningful Internet Systems 2005: OTM 2005 Workshops, International Workshop on Object-Role Modeling (ORM'05), pp. 700-709, Publ. Lecture Notes in Computer Science 3762, Springer

De Troyer, O. et al. (2007). WSDM: Web Semantics Design Method in Web Engineering: Modelling and Implementing Web Applications, Human-Computer

Interaction Series Vol. 12, pp. 303-352, Eds. Gustavo Rossi, Oscar Pastor, Daniel Schwabe, Louis Olsina, Publ. Springer

De Troyer, O. (2001). Audience-driven web design. In Information modelling in the new millennium, IDEA GroupPublishing.

De Troyer, O., P. Plessers, et al. (2007). WSDM: Web Semantics Design Method, Springer London.

Gabriel, R. P. (1998). Patterns of Software: Tales from the Software Community, Oxford University Press, USA.

Gamma, E., R. Helm, et al. (1995). Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley.

Grand, M. (2002). Patterns in Java: A Catalog of Reusable Design Patterns Illustrated with UML, John Wiley & Sons.

Laakso (2010). "User Interface Design Patterns." from http://www.cs.helsinki.fi/u/salaakso/patterns/.

Paternò, F. (2000). Model-Based Design and Evaluation of Interactive Applications. London, Springer.

Paternò, F. (2001). Task Models in Interactive Software Systems. Handbook of Software Engineering & Knowledge Engineering, S. K. Chang, World Scientific Publishing Co.

Paternò, F., C. Mancini, et al. (1997). ConcurTaskTree: a Diagrammatic Notation for Specifying Task Models. IFIP Conference Proceedings, Proceedings of the IFIP TC13 Interantional Conference on Human-Computer Interaction. Syndey, Chapman & Hall, London. Human-Computer Interaction 362–369

Sinnig, D. (2004). The complicity of patterns and model-based UI development. Computer Science. Montreal, Quebec, CONCORDIA. Master.

Sinnig, D., P. Forbrig, et al. (2003). Patterns in Model-Based Development. Software and Usability Cross-Pollination: The Role of Usability Patterns.

Sinnig, D., A. Gaffar, D. Reichart, P. Forbrig and A. Seffah (2004). Patterns in Model Based Engineering. In Proceedings of CADUI 2004, 2004, Funchal, Portugal, p. 197- 210.

Tidwell, J. (2005). Designing Interfaces: Patterns for Effective Interaction Design, O'Reilly Media.

Van Duyne, D. K., J. A. Landay, et al. (2006). The Design of Sites: Patterns for Creating Winning Web Sites, Prentice Hall.

Vora, P. (2009). Web Application Design Patterns, Morgan Kaufmann.

Yahoo. "Yahoo Design Pattern Library." from http://developer.yahoo.com/ypatterns/.

Welie, M. and G. Veer (2003). Pattern Languages in Interaction Design: Structure and Organization. In Proceedings of INTERACT 2003, September 2003, Zuerich, pp. 527-534.

Welie (2004). "Pattern in Iteraction Design." from http://www.welie.com/.