Technical Report TR02 06

Reflective Collaborative Agents for Complex Service Integration

P. Maurine Hatch and Eleni Stroulia

Computing Science Department

University of Alberta

Edmonton, AB, Canada T6G 2E8

{maurine,stroulia}@cs.ualberta.ca

June 2002

Table of Contents

Abstract

With the advent of more and more services available on the Web, a user can have a difficult job of assembling the various pieces of a complex task to arrive at a final solution. Not only would the user need to access each web-based resource through its individual client-side interface, but she would also need to interpret its response to her request, and to manually combine the multiple responses from the different resources to accomplish the complex task.

In this paper, we discuss a multiagent, XML-based framework that supports the development of aggregate applications that rely on semantic-based reflective monitoring and collaboration among several agents to complete the user's task. Our framework makes use of declarative models of the domain information, the task-specific information, and the semantic constraints of this information. Each agent uses these models to interact with the user, to coordinate the information exchange with the various web resources, to monitor and control the execution of the applications, and to act when a failure is detected. When an agent detects a failure, it collaborates with other agents by distributing the tasks to those agents that are capable of completing the task, thus ensuring successful completion of the user's request.

We illustrate our approach and the architecture of the aggregate applications that it produces using a book-buying assistant as an example.

1. Motivation and Backyround

As the World Wide Web continues to grow exponentially, querying it for a particular service can be a frustrating experience for a user. There are many different web-based applications and they usually do not interoperate. Not only does a user with a complex task need to decide in which order to access multiple applications, but she also needs to decide how to combine the information in their responses in order to formulate requests to yet other applications, and how to construct the solution to the task at hand based on all the responses received. The user's task would be much easier if she could simply request the complex service and have the final solution returned to her. This requires the development of an application that combines the functionalities of multiple, independently developed, web-based applications. The application aggregation would need to control the information flow of the overall task, translate the information among the existing applications, and combine the many responses received into a final solution

This interoperation of web-based applications is a challenge because there exists neither an agreed upon representation and semantics for the information required and produced by these applications, nor a uniform access mechanism for their services. Our work towards addressing this challenge has resulted in a multi-agent framework for task-specific aggregation of web-based applications. Each agent within this framework uses an eXtensible Markup Language (XML) [2] -specified domain model for the information representation and semantics, an XML-specified task structure model to represent the workflow and a task agent component that controls and coordinates the process by using these models.

Our framework is based on the TaMeX framework [13]. Similarly to TaMeX, *wrappers* are used to encapsulate the web-based applications that provide the information and services in a given domain. The role of a wrapper is that of an adapter between the application's original user interface and a new interface based on a common XML vocabulary for the domain. The wrappers interact with a *task agent* that acts as an intelligent, task-specific intermediary between the user and all the wrapped applications that are needed to accomplish the user's task. The task agent uses the *task structure model*, which is an explicit representation of a set of steps to accomplish the user's complex tasks, to control the process and information flow. The task agent also uses a *domain model*, which is a specification of the concepts of the subject area of interest, to establish a common language to be used by the user, the wrappers, and the task agent itself.

We have extended TaMeX to include an explicit representation of semantic constraints on the task and domain models of the task agent and the wrappers, and also a model of the distribution of capabilities between several agents. These more complex representations also required that the TaMeX models are specified using XML schema [6,7,8] technology. We have further extended the TaMeX framework by adding reflective monitoring with domain-specific and task-specific semantics. This means that the execution is monitored and the various constraints on the information, both domain-specific and task-specific, are checked. This constraint checking is to ensure that there is valid input to and valid output from each step involved in the successful completion of the user's request. We also converted the framework to a multi-agent framework with the ability for the user-contacted agent to identify and collaborate with other agents. This collaboration is necessary when the reflective monitoring of the user-contacted agent detects a failure condition on one of its subtasks. Other agents are then called into service to ensure that the user's request is satisfactorily deployed.

The rest of this paper is organized as follows. Section 2 discusses the overall architecture of our multi-agent framework for aggregation of web-based applications. Section 3 describes and illustrates the run-time behavior of an aggregate application by using a prototype book-buying assistant as an example. Finally, section 4 summarizes the approach and concludes by identifying the contributions of this work.

2. Overall architecture

The overall architecture of our multi-agent XML-based framework consists of two loosely coupled environments: a design-time environment and a run-time environment. A multi-agent system consists of a group of agents that interact and cooperate to accomplish some set of tasks in a distributed way. Each agent offers a specific service or services to other agents. The solution to a large task is accomplished by combining and coordinating the different services offered by the individual agents. In our case, each agent is responsible for collaborating with the other agent(s) to accomplish the specific tasks of their own users. Once the task is accomplished, each agent is responsible for returning the combined results to their own users.

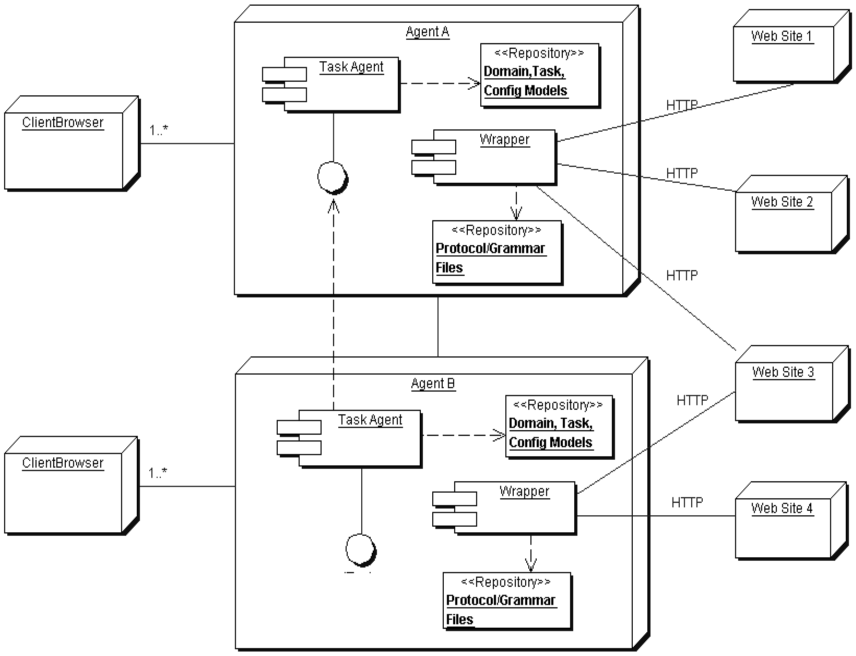


Figure 1: Run-time architecture of our multi-agent XML-based framework.

The run-time architecture is diagrammatically depicted in Figure 1, consisting of two agents. The main components of this architecture are the web-based applications being accessed and the agents that control the processing between the users, these web-based applications, and other agents. Each agent has the same structure and is composed of a task agent component, a wrapper component, and the repositories for the supporting files for these two components. The important supporting files for the task agent component are the domain model, the task structure model, and the constraints specified within these models. These models are computer-usable, declarative representations of the domain and the task structure respectively. The domain is the subject area involved in the specific application being run. The task structure is the specification of the workflow of the overall task of the application. The important supporting files for the wrapper component are the learned request protocol and grammar rules.

The design-time environment supports the development of these models/files needed for the execution of the task agent and the construction of the wrappers for the existing web-based applications for each new user task. These XML models and wrapper-construction files are used as input to the run-time environment. Using these models, each agent within the run-time environment monitors the execution, checks the constraints, and aggregates the outputs of the underlying wrapped web-based applications for its user(s) tasks.

2.1 The task ayent

The role of a task agent is to interact with the user of the multi-agent system, to evaluate the applicable conceptual and functional constraints, and to coordinate the information exchange among the wrappers of the existing underlying applications and other agents needed for accomplishing the user's task.

To accomplish this coordination between the wrappers and other agents, the task agent uses a declarative, task-structure [10,11,12] approach to the representation of its processing mechanism. In this approach, a task is characterized by the type(s) of information it consumes as input and produces as output, and the nature of the transformation it performs between the two. A complex task may be decomposed into a partially ordered set of simpler subtasks. A simple, non-decomposable task, i.e., a leaf task, corresponds to an elementary procedure. The control of processing moves from higher-level complex tasks to their constituent subtasks. At the same time, information flows through the task structure as it is produced and consumed by the tasks. The actual process is non-deterministic because there may be alternative decompositions for a given task, applicable under different conditions, and the subtasks resulting from the decomposition of a complex task are only partially ordered. Complex, high-level tasks get accomplished when all their subtasks are accomplished. The XML-based task structure model used by the task agent is described in section 2.2.

Each task agent implements a set of these XML-specified tasks, with non-deterministic decompositions. The user's interaction with the task agent decides which of the alternative decompositions is employed, and which task is active at each point in time. Also, through some user-interaction leaf tasks, the user specifies the problem and receives the results as they get produced. The wrappers of the individual underlying applications implement elementary leaf tasks using the services of their underlying sources (information-collection tasks). When a particular task is sufficiently decomposed into elementary tasks, the task agent requests the relevant wrappers to accomplish them and to return their results

As well as using a task structure model, the task agent also uses a system configuration model and a domain model for its processing. The system configuration model specifies information needed for execution of the system, such as the name of the server that the task agent is running on, the list of the registered task models, and the list of all the resources that the system knows about. The domain model specifies the entities of the application domain and their constraints. The task agent uses the constraints specified in the models to validate the user's input and the successful execution of the subtasks involved.

2.2 The task structure model

The role of the task structure model is to enable the task agent to:

1. decompose high-level tasks into elementary (leaf) ones
2. present a hierarchical menu of tasks to the user
3. compose individual wrappers' results in a coherent solution to the overall user's task

The task structure is specified using XML. A generic task structure is specified with an XML schema and is part of the design-time environment. See Appendix B for the XML structure diagram and XML schema for the generic task structure used in our framework. An application-specific XML task structure instance is specified, in conformance with the task structure schema, for each high-level complex task to be executed by the user. This task structure instance is used by the task agent to present the task structure to the user as a hierarchical menu where the high-level tasks correspond to the top-level menus and their alternative decompositions are represented in sub-menus. The tasks presented are color-coded so that the user knows which tasks are executable and which have been completed. This representation style gives any specific task agent a simple intuitive interface through which the user can traverse the task structure and invoke the tasks that it can accomplish.

There are five types of tasks in the generic task structure. There are the two user-interaction tasks: the input task that gathers information from the user and the output task that presents results to the user. For each input and output task there is an associated XSL stylesheet [3] for user presentation. There is the wrapper task that is an information-collection task and accesses an external resource. For each wrapper task, a wrapper must be constructed to provide the functionality specified by the task. There is the internal task that performs non-interactive processing. For each internal task, a new component must be implemented to provide the functionality specified by the task. Finally, there is the grouping task that is used to structure the other tasks in sub-groupings to correspond to the intermediate sub-goals of the user's overall task.

2.3 The domain model

The role of the domain model is to establish a common language for specifying the application information that can be understood by the user, the task agent and the wrappers of all the integrated applications. The application information that is specified using this domain model is the information provided by the user, the information produced by the task agent as it elaborates the user's problem specification, and the information provided to and by the wrappers and agents.

The domain model, used by the task agent, specifies:

1. the entities of the application domain
2. the attributes of these entities
3. the composition relationships between these entities
4. the logic constraints of these entities

The domain model is specified using XML documents. An application-specific domain model and some of its constraints are specified with an XML schema. Constraints that cannot be specified in the XML domain schema are specified in an XSLT stylesheet.

2.4 Constraints and reflective monitoring

The role of the constraints is to provide the semantics used by the reflective monitoring process of the task agent. At run time, the task agent checks the constraints to ensure that the information consumed and produced at each stage of the processing is valid. Upon discovery of invalid information, the task agent takes action, such as invoking the services of another agent, to resolve the problem.

There are two main categories of constraints: conceptual constraints, which are domain-specific constraints defining the assumed ontology of the domain, and functional constraints, which are task-specific constraints specifying the nature of the information transformation expected of each task. Table 1 shows a breakdown of the types of constraints within each category, along with their description and where they are specified within our framework.

	Constraint Type	**Description**	**Where Specified**
Conceptual	Value Constraint Single Field	-constraints on the value of a single field of information in the domain model	XML domain schema XSLT stylesheet
	Value Constraint Multiple Fields	-semantics on the relationship between values of multiple fields in the domain model	XSLT stylesheet
Functional	Structural Constraint	-semantics on the relationship between the tasks in the task structure model	XML task structure schema and instance
	Task Parameter	-semantics on the relationship between the values of attributes of the task parameters	Extension to XML domain schema XSLT stylesheet

Table 1: Constraints used by reflective monitoring in our framework

2.5 The Wrappers

The wrapper components in the architecture shown in Figure 1 act as adapters between the web-based resource's original API and a new API based on a common XML vocabulary for the domain. The wrapper receives an information-collection request from the task agent that is the problem specified in terms of this XML domain model. The wrapper responds to this request by returning an answer in the form of an XML document containing one or more instances of some concept (or concepts) in the domain model. This concept is called the target concept and is the entity (or entities) of the XML domain model that answer the problem specification from the task agent.

Our wrapper-construction process is based on the approach developed by Stroulia, Thomson, and Situ [13]. This approach exploits the hierarchical structure of both XML and HTML documents. XML documents are structured hierarchically where each XML element can be composed of simpler XML elements. We rely on this characteristic of XML for creating the semantic map of documents used in wrapping the WWW applications. The XML documents used are those for specifying the domain model, the information-collection request to the wrapper, and the answer returned from the wrapper that contains the instances of the target concept.

Our approach also utilizes the hierarchical structure of HTML. Currently, most web-based applications provide information encoded using HTML. These applications automatically generate HTML documents in response to a particular request type. The visual layout of these generated HTML documents for individual responses is similar, even though the actual content is different. In order to extract an instance of the target concept from one of these HTML responses, the hierarchical structure of this response is traversed to discover and select the needed elements of the target concept. This is done by parsing the HTML document into a tree representation rooted at the < html > tag, so that document subtrees that may contain the information of interest can be efficiently located, using a DOM-like API [1]. The wrapper uses a set of rules, or grammar, to traverse this document tree to locate the instances of the target concept's constituent elements. This extraction grammar is learned at design-time and is formulated using XPATH [5] expressions and is specified in an XML document.

In order to wrap a web-based application, the wrapper needs to know the protocol by which the responses of interest can be requested from the application server, and the grammar for extracting the instances of the target concept from the server's responses. The XML files that contain this information, the learned request protocol file and the grammar rules file, are stored in a repository for use by the wrapper.

The learned request protocol is the interaction protocol between the user's browser and the application. This protocol is specified in XML. Its role is to provide the wrapper component with a means to translate an XML problem specification into an appropriate request to the application server.

The grammar rules are the XPATH expressions for extracting each component of the target concept from the application's response. They are specified in XML. The role of the grammar rules is to provide the wrapper component with a means to extract the instances of the target concept from the application server's responses.

3. Run time Behavior

To illustrate the run-time behavior of the aggregate applications developed with our multi-agent framework, we will use as an example a book-buying assistant prototype. This prototype was built to support explorative comparative book shopping, and more specifically book price checking. Today there are many websites offering books for sale. Knowledgeable consumers with specific constraints and preferences must access several different sites to identify available options. Then they have to compare the results from these different sites prior to making a decision. The aggregation of existing book-selling applications to support tasks such as comparative shopping is a compelling instance of web-based application aggregation. Our book-buying prototype application integrates two different book-selling web-based applications.

As the first step in the development of this application, the entities, their attributes, their compositions, and their constraints needed for the book-buying domain are specified in an XML schema and XSLT stylesheets. This book-buying domain XML schema file and an example conceptual constraint XSLT stylesheet are shown in Appendix A. A pictorial representation of the book-buying domain model is shown in Figure 2.

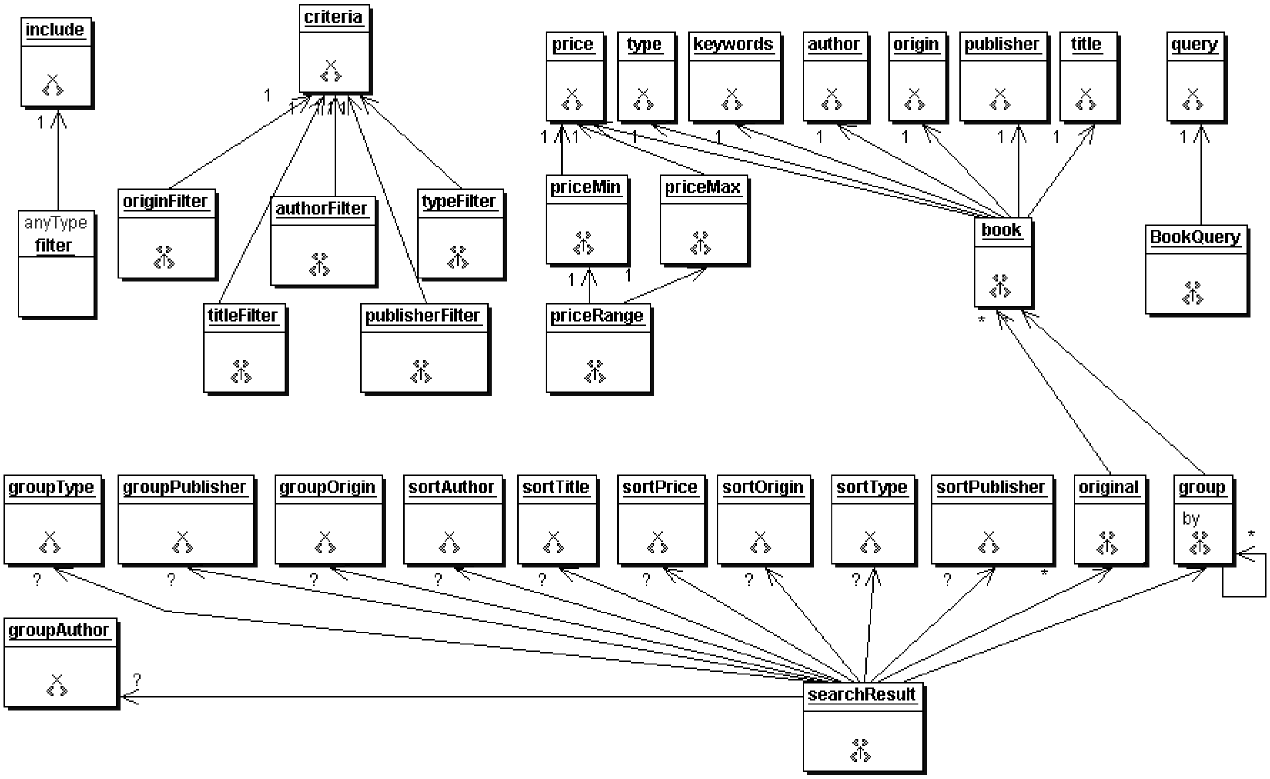


Figure 2: Domain Model XML Structure Diagram for the Book Buying Assistant.

This representation is an XML structure diagram generated by Together 5.5 from the XML schema file of the domain model. The multiplicities are shown on the diagram by '?', '1', or '*', where '?' signifies zero or one instance, '1' signifies exactly one instance, and '*' signifies zero up to an unlimited number of instances. The composition relationships can be seen from this diagram. For example, it can be seen that the searchResult consists of an 'original' entity, a 'group' entity, four grouping indication entities ('groupAuthor', 'groupType', 'groupPublisher', 'groupOrigin'), and six sorting indication entities ('sortAuthor', 'sortTitle', 'sortPrice', 'sortOrigin', 'sortType', 'sortPublisher'). Both the 'original' and 'group' entities can occur zero to many times (shown by '*'). Each grouping indication and sorting indication entity may or may not occur (shown by '?'). It can also be seen that the 'original' and the 'group' entities are both composed of books. Each book is composed of mandatory elements such as price, type, author, etc.

Our example of the book-buying assistant prototype is run with three agents. Each agent contains the same book buying assistant's task structure as diagrammatically depicted in Figure 3. The book buying assistant's XML task structure instance and an example functional constraint XSLT stylesheet are shown in Appendix C.

Though each agent uses the same task structure, each agent has different capabilities within this task structure. Agent A can perform all the tasks except for any of the "Access Book-Selling Resources". Agent B can perform all the tasks. Agent C can perform all the tasks except "Access Chapters" and "Group by Type". In our example, Agent C is the agent that interacts with the user. Agent A and Agent B are the agents that have task capabilities within the book-buying assistant application that Agent C does not have. Agent C will accomplish the user's overall task by traversing the book-buying assistant task structure and performing the tasks within that structure. Agent C will either perform these tasks directly by itself, if capable, or will identify and then collaborate with other agents that are capable of performing these tasks.

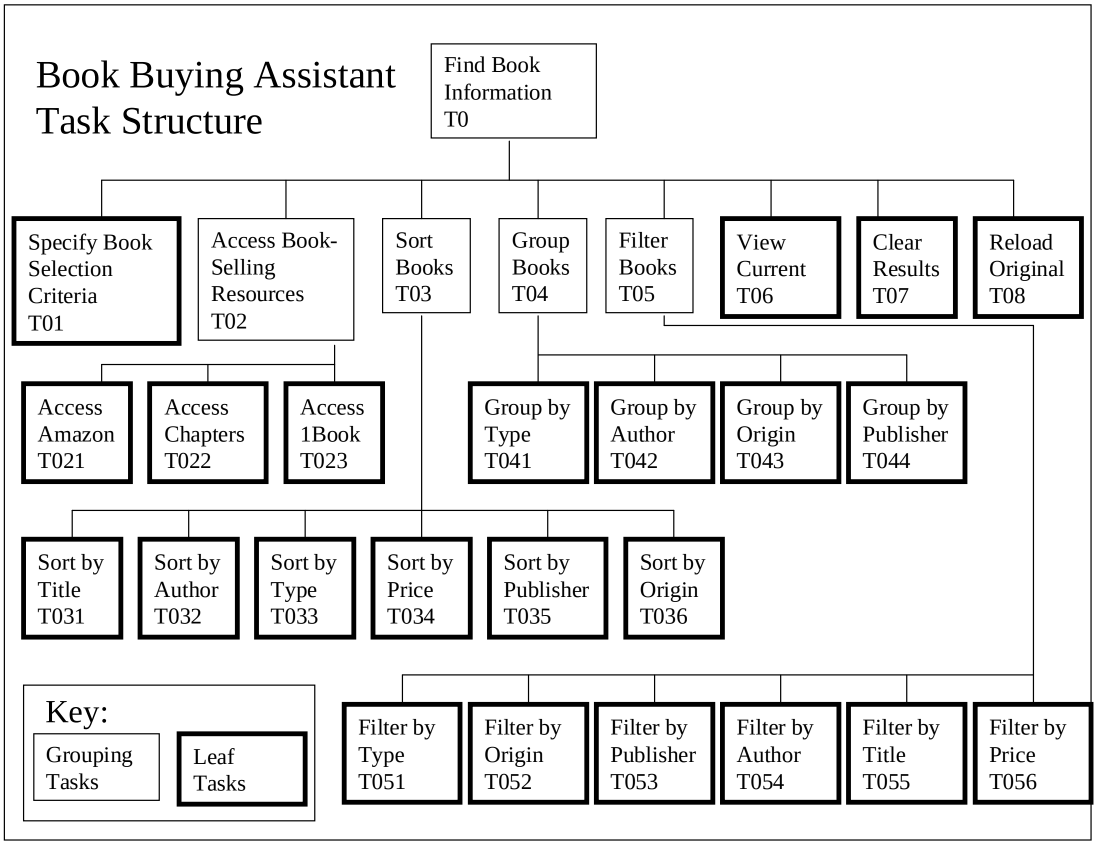
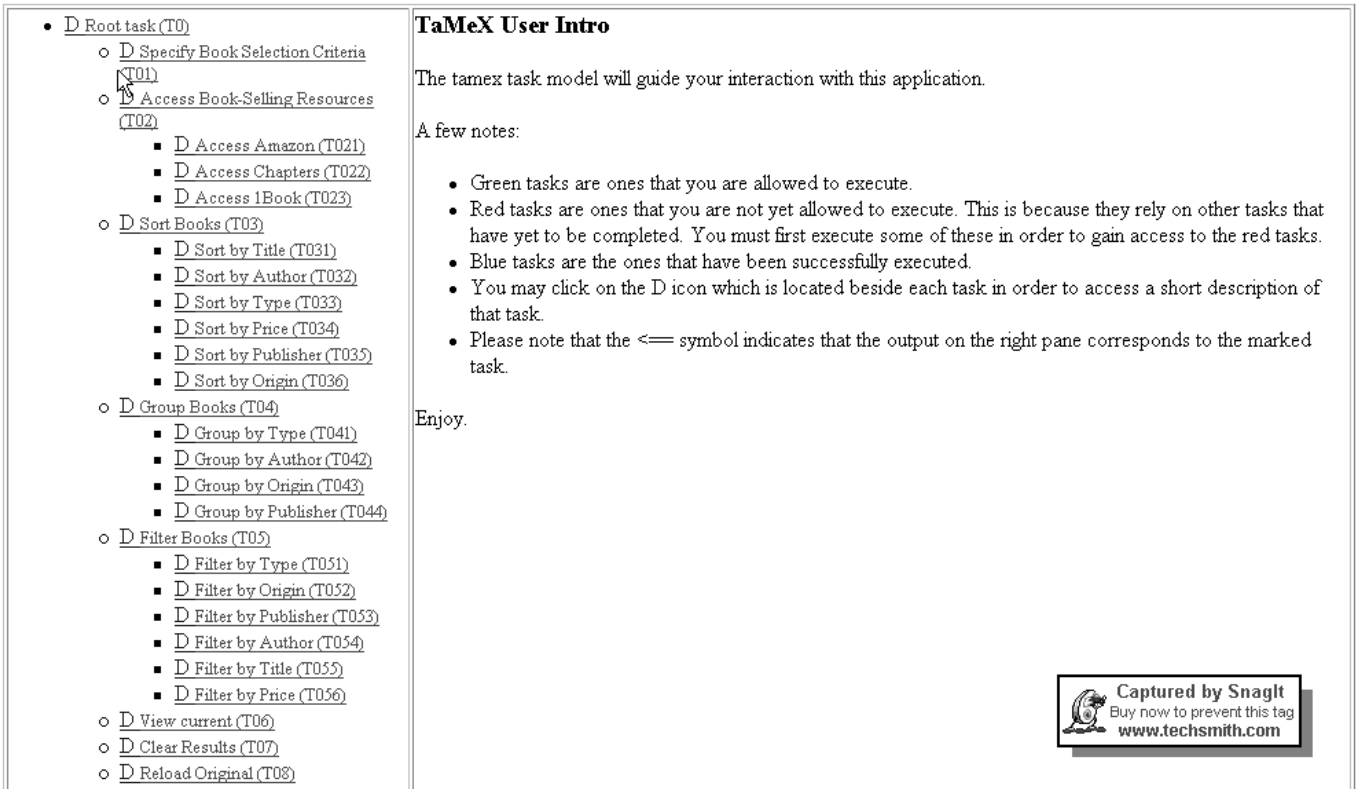


Figure 3: Task Structure for the Book Buying Assistant.

In this task structure, the grouping tasks that are used to arrange the other tasks into sub-groupings are shown as regular solid boxes. The leaf tasks, which do the actual processing, are shown as heavy-lined solid boxes. The overall task of the book buying assistant, i.e., to *find book information*, gets decomposed into the tasks *of specifying book selection criteria*, *accessing book-selling resources* via the wrapped web-based applications, arranging the accessed books by *sorting*, *grouping*, and/or *filtering*, and finally by *viewing the currently* selected and arranged books. There are two tasks to redo the book finding process - one to *clear the results* obtained and start over and the other to *reload the original* accessed results (i.e. before any arrangement of the results). The task agent of an agent capable of executing the book buying assistant uses the XML-specified model of this task structure to produce a task menu on the user's browser interface. This menu, shown in Figure 4, is the user's means of interacting with the task agent. The first step in the process is to specify the problem inputs. This is the user-interaction (input) task of "Specify Book Selection Criteria (T01)". The task agent sends to the browser the *query* entry form, which is generated by the XSL stylesheet corresponding to this input task. Using this form, the user specifies the keyword of the books he wants to find. For the problem at hand, the user specifies the keyword to be "java" by typing "java" in the "keyword" entry field.

After the desired book selection criteria have been specified, the task agent proceeds to accomplish the next task, i.e., to access the book-selling resources (T02) via the wrapped web-based applications. This grouping task is decomposed into a set of information-collection (wrapper) tasks, one for each wrapped web-based application. In this example there are three wrapper tasks, "Access Amazon (T021)", "Access Chapters (T022)", and "Access 1Book (T023)" to access the wrapped applications of the Amazon (www.amazon.com), Chapters-Indigo (www.chapters.ca), and 1BookStreet (www.1bookstreet.com) book selling websites respectively. Each wrapper task is accomplished by invoking a request to the task-specified wrapped web-based application. This process is started when a user selects a wrapper task from the menu. The task agent responds by sending a request to the particular wrapper of the selected application that will produce the desired output given the already-entered set of inputs.



- D Root task (T0)
  - D Specify Book Selection Criteria (T01)
  - D Access Book-Selling Resources (T02)
    - D Access Amazon (T021)
    - D Access Chapters (T022)
    - D Access 1Book (T023)
  - D Sort Books (T03)
    - D Sort by Title (T031)
    - D Sort by Author (T032)
    - D Sort by Type (T033)
    - D Sort by Price (T034)
    - D Sort by Publisher (T035)
    - D Sort by Origin (T036)
  - D Group Books (T04)
    - D Group by Type (T041)
    - D Group by Author (T042)
    - D Group by Origin (T043)
    - D Group by Publisher (T044)
  - D Filter Books (T05)
    - D Filter by Type (T051)
    - D Filter by Origin (T052)
    - D Filter by Publisher (T053)
    - D Filter by Author (T054)
    - D Filter by Title (T055)
    - D Filter by Price (T056)
  - D View current (T06)
  - D Clear Results (T07)
  - D Reload Original (T08)

TaMeX User Intro

The tamex task model will guide your interaction with this application.

A few notes:

- Green tasks are ones that you are allowed to execute.
- Red tasks are ones that you are not yet allowed to execute. This is because they rely on other tasks that have yet to be completed. You must first execute some of these in order to gain access to the red tasks.
- Blue tasks are the ones that have been successfully executed.
- You may click on the D icon which is located beside each task in order to access a short description of that task.
- Please note that the <== symbol indicates that the output on the right pane corresponds to the marked task.

Enjoy.

Figure 4: Task Menu for the Book-Buying Assistant

Since, in this example, the user wants to see all of the books that are available she selects each wrapper task in turn. When the user selects the first wrapper task of "Access Amazon (T021)" Agent C's task agent responds directly (since Agent C is capable of performing this task) by sending a request to the particular wrapper of the Amazon application that produces a list of books given a keyword. The keyword that is used for this access was previously entered by the user in the "Specify Book Selection Criteria (T01)" task and is "java". So, in this case, the T021 task will return a list of all the books with a keyword of "java" that can be found at the Amazon website.

The user then selects the next wrapper task of "Access Chapters (T022)." However, this time Agent C fails at this task because it does not have the capability to access the Chapters-Indigo website. Agent C takes corrective action and identifies that Agent B is capable of accessing the Chapters-Indigo website. After having identified Agent B, Agent C begins its collaboration with this agent by sending Agent B a request to perform this particular task activity along with the variables needed to perform it. Agent B's task agent executes this activity by sending a request to the particular wrapper of the Chapters-Indigo application that produces a list of books given a keyword. After getting the results, Agent B returns this list of books to the originating Agent C. Agent C has now completed the "Access Book-Selling Resources (T02) task.

Agent C then continues traversing the book buying assistant's task structure according to how the user wants to arrange (sort, group, and/or filter) these books. The user performs these arranging tasks ("Sort Books (T03)", "Group Books (T04)", "Filter Books (T05)") iteratively and in any order until she has the selection of books to her liking. She can choose to display the currently selected and arranged books at any time by performing "View Current (T06)". The "Sort Books (T03)" grouping task allows the user to choose one of six choices: *sort by title, sort by author, sort by type, sort by price, sort by publisher*, or *sort by origin*. The "Group Books (T04)" grouping task allows the user to choose one of four choices: *group by type, group by author, group by origin*, or *group by publisher*. The "Filter Books (T05)" grouping task allows the user to choose one of six choices: *filter by type, filter by origin, filter by publisher, filter by author, filter by title*, or *filter by price.*

In our example, the user wants to group the books by whether they are hardcover or paperback so she selects the "Group by Type (T041)" task from her browser's menu. Agent C fails at performing this task directly because it does not have the capability to group books by type. Agent C identifies that Agent A is capable of performing the grouping by type and sends Agent A a request to group the selected books by type. Agent A performs this activity by using the variables passed to it by Agent C as input. Then Agent A returns the result of this task execution, which is a new grouping of books by type, to the originating Agent C. Agent C then indicates to the user that the grouping task has been completed. The user wants to check her grouping so she chooses "View Current (T06)" to display the currently selected and arranged books. Agent C then executes this user-interaction task, the output task of "View Current (T06)" and displays this grouping of books by type to the user. The user is satisfied with the arranged results and so stops the process.

4. Conclusions

In this paper, we discussed an XML-based framework for the aggregation of web-based applications involving agents that collaborate to ensure that the requested service is delivered. The two main contributions of this work are
(a) the semantic-based reflective monitoring of the task-agents' behavior, and
(b) the agents' collaboration and tasks distribution that occurs when an agent fails.
In our framework, agents use declarative models of the application domain, the task structure and the semantic constraints specifying the information in this domain and the nature of the transformations it suffers in the task structure. Each agent uses these models to interact with the user, to coordinate the information exchange among the wrappers of the underlying web applications, to monitor and control the execution of the applications, and to collaborate with the other agents when it cannot alone complete the desired task.

References

1. Document Object Model (DOM) Level 2 Specification, http://www.w3.org/TR/1999/CR-DOM-Level-2-19991210/
2. Extensible Markup Language (XML), http://www.w3.org/XML/
3. Extensible Stylesheet Language (XSL) Version 1.0 W3C Recommendation 15 October 2001, http://www.w3.org/TR/xsl/
4. Muffin, World Wide Web filtering system http://muffin.doit.org/
5. XML Path Language (Xpath) Version 1.0 W3C Recommendation 16 November 1999, http://www.w3.org/TR/xpath/
6. XML Schema Part 0:Primer W3C Recommendation, 2 May 2001, http://www.w3.org/TR/xmlschema-0/
7. XML Schema Part 1:Structures W3C Recommendation, 2 May 2001, http://www.w3.org/TR/xmlschema-1/
8. XML Schema Part 2:Datatypes W3C Recommendation, 2 May 2001, http://www.w3.org/TR/xmlschema-2/
9. XSL Transformations (XSLT) Version 1.0 W3C Recommendation 16 November 1999, http://www.w3.org/TR/xslt/
10. B. Chandrasekaran, "Task Structures, Knowledge Acquisition and Machine Learning", *Machine Learning*, 4:341-347, 1989.
11. E. Stroulia and A.K. Goel, "A Model-Based Approach to Blame Assignment: Revising the Reasoning Steps of Problem Solvers", *Proceedings of the 13th Annual Conference on Artificial Intelligence*, pp. 959-965, AAAI Press, 1996.
12. E. Stroulia and A.K. Goel, "Redesigning a Problem Solver's Operators to Improve Solution Quality. Proceedings of the 15th International Joint Conference on Artificial Intelligence, pp. 562-567, 1997.
13. E. Stroulia, J. Thomson, and Q. Situ, "Constructing XML-speaking wrappers for WEB Applications: Towards an Interoperating WEB", *Proceedings of the 7th Working Conference on Reverse Engineering,* 23-25 November 2000, Brisbane, Queensland, Australia, pp. 59-68, IEEE Computer Society Press.

APPENDICES

Appendix A: Domain Model for the Book Buying Assistant

XML Schema for the Book Buying Domain (BookBuyingDomain.xsd)

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd='http://www.w3.org/2000/10/XMLSchema'>

 <!-- The price of a book -->
 <xsd:element name="price" type="xsd:string"/>

 <!-- the author of a book -->
 <xsd:element name="author" type="xsd:string"/>

 <!-- the query string entered by the user -->
 <xsd:element name="BookQuery">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element name="query" type="xsd:string" minOccurs="1" maxOccurs="1"/>
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>

<!-- the result returned by the server, a list of websites
         with a list of books at each website                 -->
 <xsd:element name="searchResult">
   <xsd:complexType>
     <xsd:sequence>
       <!-- sortXYZ specifies that the set is sorted by XYZ -->
       <xsd:element name="sortType" type="xsd:boolean" minOccurs="0" maxOccurs="1"/>
       <xsd:element name="sortTitle" type="xsd:boolean" minOccurs="0" maxOccurs="1"/>
       <xsd:element name="sortAuthor" type="xsd:boolean" minOccurs="0" maxOccurs="1"/>
       <xsd:element name="sortPrice" type="xsd:boolean" minOccurs="0" maxOccurs="1"/>
       <xsd:element name="sortPublisher" type="xsd:boolean" minOccurs="0"
maxOccurs="1"/>
       <xsd:element name="sortOrigin" type="xsd:boolean" minOccurs="0" maxOccurs="1"/>

       <!-- groupXYZ specifies that the set is grouped by XYZ -->
       <xsd:element name="groupType" type="xsd:boolean" minOccurs="0" maxOccurs="1"/>
       <xsd:element name="groupAuthor" type="xsd:boolean" minOccurs="0"
maxOccurs="1"/>
       <xsd:element name="groupPublisher" type="xsd:boolean" minOccurs="0"
maxOccurs="1"/>
       <xsd:element name="groupOrigin" type="xsd:boolean" minOccurs="0"
maxOccurs="1"/>

       <xsd:element ref="original" minOccurs="0" maxOccurs="unbounded"/>
       <xsd:element ref="group" minOccurs="0" maxOccurs="unbounded"/>
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>

<!-- holds the original result set returned by the resources -->
 <xsd:element name="original">
   <xsd:complexType>
     <xsd:sequence>
```

```
       <xsd:element ref="book" minOccurs="0" maxOccurs="unbounded"/>
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>

<!-- each group can contain either more groups or books. -->
<!-- the by attribute holds a string describing what the grouping is
     by and this value is optional -->
 <xsd:element name="group">
   <xsd:complexType>
     <xsd:choice>
       <xsd:element ref="book" minOccurs="0" maxOccurs="unbounded"/>
       <xsd:element ref="group" minOccurs="0" maxOccurs="unbounded"/>
     </xsd:choice>
     <xsd:attribute name="by" type="xsd:string" use="optional" default=""/>
   </xsd:complexType>
 </xsd:element>

<!-- a single book element in the result returned by the server,
     the filtered result,and the sorted result                      -->
 <xsd:element name="book">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element name="title" type="xsd:string" minOccurs="1" maxOccurs="1"/>
       <xsd:element ref="author" minOccurs="1" maxOccurs="1"/>
       <xsd:element name="publisher" minOccurs="1" maxOccurs="1"/>
       <xsd:element name="type" type="xsd:string" minOccurs="1" maxOccurs="1"/>
       <xsd:element ref="price" minOccurs="1" maxOccurs="1"/>
       <xsd:element name="origin" type="xsd:string" minOccurs="1" maxOccurs="1"/>
       <xsd:element name="keywords" type="xsd:string" minOccurs="1" maxOccurs="1"/>
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>


<!-- the price range of books desired for the filter task -->
 <xsd:element name="priceRange">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element ref="priceMin" minOccurs="1" maxOccurs="1"/>
       <xsd:element ref="priceMax" minOccurs="1" maxOccurs="1"/>
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>

<!-- the minimum price of the price range element -->
 <xsd:element name="priceMin">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element ref="price" minOccurs="1" maxOccurs="1"/>
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>

<!-- the maximum price of the price range element -->
 <xsd:element name="priceMax">
   <xsd:complexType>
```

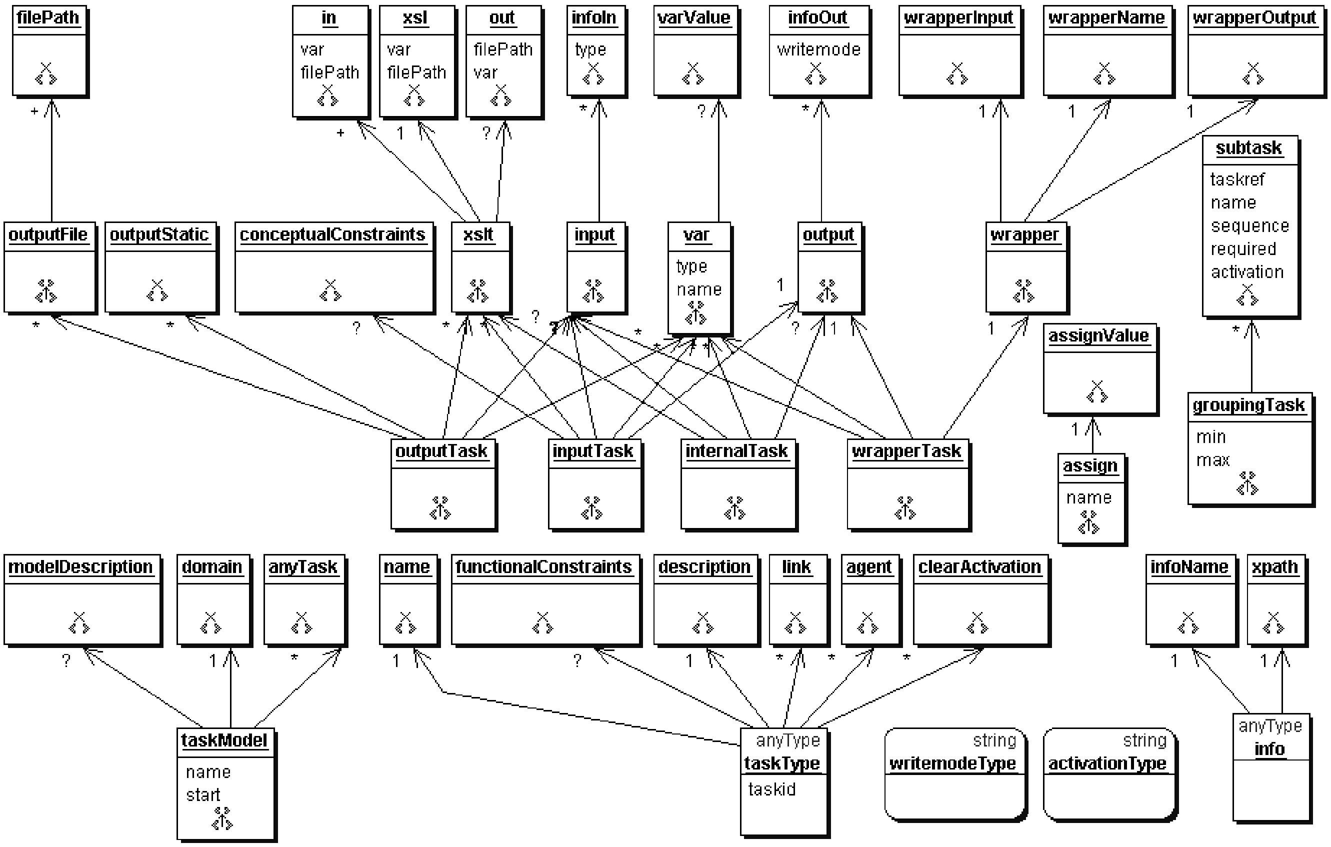
```
     <xsd:sequence>
       <xsd:element ref="price" minOccurs="1" maxOccurs="1"/>
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>

<!-- general filter type includes include element which flags wether
     or not matching elements should be included or excluded, true
     if they are to be included -->
<xsd:complexType name = "filter">
  <xsd:sequence>
    <xsd:element ref = "include" minOccurs = "1" maxOccurs = "1"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:element name = "typeFilter">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="filter">
        <xsd:sequence>
          <xsd:element ref = "criteria" minOccurs = "1" maxOccurs = "1"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<xsd:element name = "originFilter">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="filter">
        <xsd:sequence>
          <xsd:element ref = "criteria" minOccurs = "1" maxOccurs = "1"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<xsd:element name = "publisherFilter">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="filter">
        <xsd:sequence>
          <xsd:element ref = "criteria" minOccurs = "1" maxOccurs = "1"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<xsd:element name = "authorFilter">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="filter">
        <xsd:sequence>
          <xsd:element ref = "criteria" minOccurs = "1" maxOccurs = "1"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<xsd:element name = "titleFilter">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="filter">
        <xsd:sequence>
          <xsd:element ref = "criteria" minOccurs = "1" maxOccurs = "1"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<!-- include filter element -->
<xsd:element name="include" type="xsd:boolean"/>

<!-- the filter criteria for some of the filters -->
<xsd:element name="criteria" type="xsd:string"/>

</xsd:schema>
```

Example Conceptual Constraint XSLT stylesheet

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output method="xml"/>
<xsl:template match="/">

<xsl:if test="(string(number(/Vars/aPriceMin/priceMin))='NaN') or
(string(number(/Vars/aPriceMax/priceMax))='NaN')">
   <xsl:text> The prices must be numeric. </xsl:text>
   <xsl:text> Please enter the prices in the form of nnn.nn  </xsl:text>
</xsl:if>

</xsl:template>
</xsl:stylesheet>
```

Appendix B: Task Structure Model   Generic

XML Structure Diagram for the Generic Task Structure



XML Schema for the Generic Task Structure

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">

<!--*****************************************************************
     High level description of task model
***************************************************************-->
<xsd:element name = "taskModel">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref = "domain" minOccurs = "1" maxOccurs = "1"/>
      <xsd:element ref = "modelDescription" minOccurs = "0" maxOccurs = "1"/>
      <xsd:element ref = "anyTask" minOccurs = "0" maxOccurs = "unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="start" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>

<!--*****************************************************************
     anyTask - an abstract element to be used in a
               substitutionGroup for tasks
***************************************************************-->
<xsd:element name = "anyTask" abstract="true" type="taskType"/>
```

```
<!--****************************************************************
     Individual Task - Grouping Task
        - contains subtask elements and provides structure to
          the task model as viewed by the user
****************************************************************-->
<xsd:element name="groupingTask" substitutionGroup="anyTask">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="taskType">
        <xsd:sequence>
          <xsd:element ref="subtask" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attribute name="min" type="xsd:integer" use="optional" default="-1"/>
        <xsd:attribute name="max" type="xsd:integer" use="optional" default="-1"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<!--****************************************************************
     Individual Task - Input Task
        - gathers interactive input from the user
****************************************************************-->
<xsd:element name="inputTask" substitutionGroup="anyTask">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="taskType">
        <xsd:sequence>
          <xsd:element ref = "var" minOccurs = "0" maxOccurs = "unbounded"/>
          <xsd:element ref = "conceptualConstraints" minOccurs = "0" maxOccurs =
"1"/>
          <xsd:element ref = "input" minOccurs = "0" maxOccurs = "1"/>
          <xsd:element ref = "output" minOccurs = "1" maxOccurs = "1"/>
          <xsd:element ref = "xslt" minOccurs = "1" maxOccurs = "1"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<!--****************************************************************
     Individual Task - Output Task
        - presents results to the user
****************************************************************-->
<xsd:element name = "outputTask" substitutionGroup="anyTask">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="taskType">
        <xsd:sequence>
          <xsd:element ref = "var" minOccurs = "0" maxOccurs = "unbounded"/>
          <xsd:element ref = "input" minOccurs = "0" maxOccurs = "1"/>
          <xsd:element ref = "outputFile" minOccurs = "0" maxOccurs = "unbounded"/>
          <xsd:element ref = "outputStatic" minOccurs = "0" maxOccurs =
"unbounded"/>
          <xsd:element ref = "xslt" minOccurs = "0" maxOccurs = "unbounded"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<!--****************************************************************
     Individual Task - Wrapper Task
        - accesses an external resource
****************************************************************-->
<xsd:element name = "wrapperTask" substitutionGroup="anyTask">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="taskType">
        <xsd:sequence>
          <xsd:element ref = "var" minOccurs = "0" maxOccurs = "unbounded"/>
          <xsd:element ref = "input" minOccurs = "1" maxOccurs = "1"/>
          <xsd:element ref = "output" minOccurs = "1" maxOccurs = "1"/>
          <xsd:element ref = "wrapper" minOccurs = "1" maxOccurs = "1"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<!--****************************************************************
     Individual Task - Internal Task
        - performs non-interactive processing
****************************************************************-->
<xsd:element name = "internalTask" substitutionGroup="anyTask">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="taskType">
        <xsd:sequence>
          <xsd:element ref = "var" minOccurs = "0" maxOccurs = "unbounded"/>
          <xsd:element ref = "input" minOccurs = "0" maxOccurs = "1"/>
          <xsd:element ref = "output" minOccurs = "0" maxOccurs = "1"/>
          <xsd:element ref = "xslt" minOccurs = "0" maxOccurs = "unbounded"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<!--****************************************************************
     Individual Task - Applet Task
        - prepares for and displays an applet.
****************************************************************-->
<xsd:element name = "appletTask" substitutionGroup="anyTask">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="taskType">
        <xsd:sequence>
          <xsd:element ref = "var" minOccurs = "0" maxOccurs = "unbounded"/>
          <xsd:element ref = "input" minOccurs = "0" maxOccurs = "1"/>
          <xsd:element ref = "output" minOccurs = "0" maxOccurs = "1"/>
          <xsd:element ref = "applet" minOccurs = "1" maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<!--****************************************************************
     High level task statement descriptions
        - subtask elements are contained by grouping tasks
****************************************************************-->
<xsd:element name = "subtask">
  <xsd:complexType>
    <xsd:attribute name="taskref" type="xsd:string" use="required"/>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="sequence" type="xsd:integer" use="optional" default="0"/>
    <xsd:attribute name="required" type="xsd:boolean" use="optional"
default="false"/>
    <xsd:attribute name="activation" type="activationType" use="optional"
default="non_auto"/>
  </xsd:complexType>
</xsd:element>

<!--****************************************************************
     Task Statement - Output Task Statement
        - holds info about variables that this task outputs
****************************************************************-->
<xsd:element name = "output">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref = "infoOut" minOccurs = "0" maxOccurs = "unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<!--****************************************************************
     Task Statement - Input Task Statement
       - holds info about variables that are input to this task
****************************************************************-->
<xsd:element name = "input">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref = "infoIn" minOccurs = "0" maxOccurs = "unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<!--****************************************************************
     Task Statement - Xslt Task Statement
        - applies specified xslt(stylesheet) to specified xml
        - contains in - where to get the  input from
                  xsl - which spreadsheet to apply to that input
                  out - where to put the output

Note there can be multiple input sources, each one is converted
to an element, the first one is used as the source document
for the xsl application and all the rest are passed as parameters.

The first one does not recieve the folowing treatement it is just
```

```
used as is like before.

The parameters are as follows the name of the variable will be used
as the name of the parameter if the data is loaded from a variable.
If the data is loaded from a file path contained in a variable
the parameter will also have the same name as the variable.
If the data is loaded from a file path specified as the contents
of the in element then the parameter will have the name param1 for the
first param2 for the second and so on. Numbers will be skipped
if any of the defined variables have the name of the same form,
therefore if a variable called param1 exists then the first
in statement holding an immediate file path will be called param2
not param1.
****************************************************************-->
<xsd:element name = "xslt">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="remoteExecutable">
        <xsd:sequence>
          <xsd:element ref = "in" minOccurs = "1" maxOccurs = "unbounded"/>
          <xsd:element ref = "xsl" minOccurs = "1" maxOccurs = "1"/>
          <xsd:element ref = "out" minOccurs = "0" maxOccurs = "1"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<!--****************************************************************
     Task Statement - OutputFile Task Statement
        - outputs the contents of the specified file
****************************************************************-->
<xsd:element name = "outputFile">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="remoteExecutable">
        <xsd:sequence>
          <xsd:element ref = "filePath" minOccurs = "1" maxOccurs = "unbounded"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<xsd:element name = "outputStatic">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="remoteExecutable">
        <xsd:sequence>
          <xsd:any processContents="skip" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<xsd:element name = "var">
```

```
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="remoteExecutable">
        <xsd:sequence>
          <xsd:element ref = "varValue" minOccurs = "0" maxOccurs = "1"/>
        </xsd:sequence>
        <xsd:attribute name = "name" type="xsd:string" use="required"/>
        <xsd:attribute name = "type" type="xsd:string" use="required"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<xsd:element name = "wrapper">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="remoteExecutable">
        <xsd:sequence>
          <xsd:element ref = "wrapperName" minOccurs = "1" maxOccurs = "1"/>
          <xsd:element ref = "wrapperInput" minOccurs = "1" maxOccurs = "1"/>
          <xsd:element ref = "wrapperOutput" minOccurs = "1" maxOccurs = "1"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<xsd:element name = "applet">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="remoteExecutable">
        <xsd:sequence>
          <xsd:element ref="appletCodeBase" minOccurs="0" maxOccurs="1"/>
          <xsd:element ref="appletCode"     minOccurs="1" maxOccurs="1"/>
          <xsd:element ref="appletAlt"      minOccurs="0" maxOccurs="1"/>
          <xsd:element ref="appletName"     minOccurs="0" maxOccurs="1"/>
          <xsd:element ref="appletWidth"    minOccurs="1" maxOccurs="1"/>
          <xsd:element ref="appletHeight"   minOccurs="1" maxOccurs="1"/>
          <xsd:element ref="appletAlign"    minOccurs="0" maxOccurs="1"/>
          <xsd:element ref="appletHspace"   minOccurs="0" maxOccurs="1"/>
          <xsd:element ref="appletVspace"   minOccurs="0" maxOccurs="1"/>
          <xsd:element ref="appletHeader"   minOccurs="0" maxOccurs="1"/>
          <xsd:element ref="appletFooter"   minOccurs="0" maxOccurs="1"/>
          <xsd:element ref="appletParam"    minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<!--****************************************************************
 Subelements for applet element.
****************************************************************-->
<xsd:element name="appletCodeBase" type="xsd:string"/>
<xsd:element name="appletCode" type="xsd:string"/>
<xsd:element name="appletAlt" type="xsd:string"/>
<xsd:element name="appletName" type="xsd:string"/>
```

```
<xsd:element name="appletWidth" type="xsd:string"/>
<xsd:element name="appletHeight" type="xsd:string"/>
<xsd:element name="appletAlign" type="xsd:string"/>
<xsd:element name="appletHspace" type="xsd:string"/>
<xsd:element name="appletVspace" type="xsd:string"/>

<xsd:element name="appletHeader">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="remoteExecutable">
        <xsd:sequence>
          <xsd:element ref="xslt" minOccurs="1" maxOccurs="1"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<xsd:element name="appletFooter">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="remoteExecutable">
        <xsd:sequence>
          <xsd:element ref="xslt" minOccurs="1" maxOccurs="1"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<xsd:element name="appletParam">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="remoteExecutable">
        <xsd:sequence>
          <!-- the resulting string from this application will be used as the
               value of the parameter. -->
          <xsd:element ref="xslt" minOccurs="1" maxOccurs="1"/>
        </xsd:sequence>
        <!-- name that the parameter will have -->
        <xsd:attribute name="paramName" type="xsd:string" use="required"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<xsd:element name = "domain" type="xsd:string"/>

<!--****************************************************************
 Conceptual Constraints - filepath to a stylesheet with the domain constraints
****************************************************************-->
<xsd:element name = "conceptualConstraints" type="xsd:string"/>

<xsd:element name = "modelDescription" type="xsd:string"/>


<!--****************************************************************
```

```
 Functional Constraints - filepath to a stylesheet with the task constraints
*****************************************************************-->
<xsd:element name = "functionalConstraints" type="xsd:string"/>

<xsd:element name = "name" type="xsd:string"/>

<xsd:element name = "description" type="xsd:string"/>

<xsd:element name = "agent" type="xsd:string"/>

<xsd:element name = "infoName" type="xsd:string"/>

<xsd:element name = "xpath" type="xsd:string"/>

<!--*****************************************************************
     Information elements - store info about which types of
         data are to be stored where
*****************************************************************-->
<xsd:element name = "infoOut">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="info">
        <xsd:attribute name="writemode" type="writemodeType" use="optional"
default="overwrite"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<xsd:element name = "infoIn">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="info">
        <xsd:attribute name="type" type="xsd:string" use="required"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<!--*****************************************************************
     Task Statement Component - in
        - where to get the input from
        - there are 3 options:
        1. if filePath is true - value is a filePath to a file
        2. if var is true - value is the var(variable) from
                            which to load the data
        3. if BOTH filePath and var are true - the value of the
                specified var is taken to be a filePath
                (the variable must be of type filePath)
*****************************************************************-->
<xsd:element name = "in">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base = "xsd:string">
        <xsd:attribute name="filePath" type="xsd:boolean" use="optional"
default="false"/>
```

```
        <xsd:attribute name="var" type="xsd:boolean" use="optional"
default="false"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>

<!--****************************************************************
     Task Statement Component - xsl
        - which spreadsheet to apply to that input
        - there are 3 options:
        1. if filePath is true - value is a filePath to a file
        2. if var is true - value is the var(variable) from
                            which to load the data
        3. if BOTH filePath and var are true - the value of the
                specified var is taken to be a filePath
                (the variable must be of type filePath)
****************************************************************-->
<xsd:element name = "xsl">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base = "xsd:string">
        <xsd:attribute name="filePath" type="xsd:boolean" use="optional"
default="false"/>
        <xsd:attribute name="var" type="xsd:boolean" use="optional"
default="false"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>

<!--****************************************************************
     Task Statement Component - out
        - where to put the output
        - this may be omitted, in which case the application
          output goes to the screen (user display)
        - there are 4 options if not omitted:
        1. if filePath is true - value is a filePath to a file
        2. if var is true - value is the var(variable) to
                            which data is stored
        3. if BOTH filePath and var are true - the value of
                  the specified var is taken to be a filePath
                  (the variable must be of type filePath)
        4. if BOTH filePath and var are false(same as omitted)
            - application output goes to the screen
****************************************************************-->
<xsd:element name = "out">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base = "xsd:string">
        <xsd:attribute name="filePath" type="xsd:boolean" use="optional"
default="false"/>
        <xsd:attribute name="var" type="xsd:boolean" use="optional"
default="false"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
```

```
</xsd:element>

<xsd:element name = "filePath" type="xsd:string"/>

<xsd:element name = "varValue">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:any processContents="skip" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name = "assignValue" type="xsd:string"/>

<xsd:element name = "wrapperName" type="xsd:string"/>

<xsd:element name = "wrapperInput" type="xsd:string"/>

<xsd:element name = "wrapperOutput" type="xsd:string"/>

<xsd:element name = "link" type="xsd:string"/>

<xsd:element name = "clearActivation" type="xsd:string"/>

<!--****************************************************************
     Task Component - taskType complexType
        - used to derive tasks by extension
        - contains the elements common to all tasks
        - tasks currently derived are groupingTask, inputTask
          outputTask, wrapperTask, and internalTask
****************************************************************-->
<xsd:complexType name = "taskType">
  <xsd:sequence>
    <xsd:element ref = "name" minOccurs = "1" maxOccurs = "1"/>
    <xsd:element ref = "description" minOccurs = "1" maxOccurs = "1"/>
    <xsd:element ref = "functionalConstraints" minOccurs = "0" maxOccurs = "1"/>
    <xsd:element ref = "agent" minOccurs = "0" maxOccurs = "unbounded"/>
    <xsd:element ref = "link" minOccurs = "0" maxOccurs = "unbounded"/>
    <xsd:element ref = "clearActivation" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="taskid" type="xsd:string" use="required"/>
</xsd:complexType>

<!--****************************************************************
     Task Statement Component - remoteExecutable complexType
        - used to derive remote executable task statements by extension
        - contains the elements common to all remote executable tasks statements

If the executeRemotely flag is set then the statement is forced to throw
a DefferedExecutionException regardless of the statements abilitity to be
completed locally.

Please note that the external id must be a three part string, containing
task model name|task id|external id of task statement, with the '|' being
the delimiters.
****************************************************************-->
<xsd:complexType name = "remoteExecutable">
```

```
  <xsd:sequence />
  <xsd:attribute name="externalId" type="xsd:string" use="optional" default=""/>
  <xsd:attribute name="executeRemotely" type="xsd:boolean" use="optional"
default="false"/>
</xsd:complexType>

<!--****************************************************************
     Information Component - info complexType
        - used to derive information elements by extension
        - contains the elements common to all info elements
        - info elements currently derived are infoIn and infoOut
****************************************************************-->
<xsd:complexType name = "info">
  <xsd:sequence>
    <xsd:element ref = "infoName" minOccurs = "1" maxOccurs = "1"/>
    <xsd:element ref = "xpath" minOccurs = "1" maxOccurs = "1"/>
  </xsd:sequence>
</xsd:complexType>

<!--****************************************************************
     Task Statement Component - writemodeType
        - this type describes the different writemodes available
****************************************************************-->
<xsd:simpleType name = "writemodeType">
  <xsd:restriction base = "xsd:string">
    <xsd:enumeration value = "append"/>
    <xsd:enumeration value = "overwrite"/>
    <xsd:enumeration value = "mergeRoot"/>
  </xsd:restriction>
</xsd:simpleType>

<!--****************************************************************
     Task Statement Component - activationType
        - describes the types of activation that a task may have
        - automatic or non-automatic
****************************************************************-->
<xsd:simpleType name = "activationType">
  <xsd:restriction base = "xsd:string">
    <xsd:enumeration value = "auto"/>
    <xsd:enumeration value = "non_auto"/>
  </xsd:restriction>
</xsd:simpleType>

</xsd:schema>
```

Appendix C: Task Structure Model for the Book Buying Assistant

XML Instance of the Generic Task Structure for the Book Buying Assistant

```xml
<?xml version="1.0"?>
<taskModel name="myBookBuying" start="T0"
      xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation='../../../config/TaskModel.xsd'>

<domain>../../../domainModels/BookBuying/BookBuyingDomain.xsd</domain>
<modelDescription>This task model allowes you to view and manipulate books from a
variety of resources.(My Own Personal Testing Task Model)</modelDescription>

<groupingTask taskid="T0">
<name>Root task</name>
<description>Select a task to perform</description>
<subtask taskref="T01" name="Specify Book Selection Criteria" sequence="1" />
<subtask taskref="T02" name="Access Book-Selling Resources" sequence="2" />
<subtask taskref="T03" name="Sort Books" sequence="3"/>
<subtask taskref="T04" name="Group Books" sequence="3"/>
<subtask taskref="T05" name="FilterBooks" sequence="3"/>
<subtask taskref="T06" name="View Books" sequence="3" />
<subtask taskref="T07" name="Clear Results" sequence="3"/>
<subtask taskref="T08" name="Reload Originals" sequence="3"/>
<subtask taskref="T09" name="Show applet" sequence="3"/>
</groupingTask>

<inputTask taskid="T01">
  <name>Specify Book Selection Criteria</name>
  <description>Enter your input</description>
  <link>ClearResourceAccess</link>
  <var name="t01query" type="BookQuery"/>
  <output>
    <infoOut writemode="overwrite">
      <infoName>t01query</infoName>
      <xpath>/bookQuery</xpath>
    </infoOut>
  </output>
  <xslt externalId="R01">
    <in filePath="true">../../../taskModels/myBookBuying/t01input.xml</in>
    <xsl filePath="true">../../../taskModels/myBookBuying/t01input.xsl</xsl>
  </xslt>
</inputTask>

<groupingTask taskid="T02" min="1">
<name>Access Book-Selling Resources</name>
<description>Select the resources wanted</description>
<subtask taskref="T021" name="Access Amazon" />
<subtask taskref="T022" name="Access Chapters" />
<subtask taskref="T023" name="Access 1Book" />
</groupingTask>

<wrapperTask taskid="T021">
  <name>Access Amazon</name>
  <description>Access the Amazon website</description>
  <agent>A01</agent>
  <link>FixData</link>
  <link>ClearSorting</link>
  <link>ClearGrouping</link>
  <link>ClearFiltering</link>
  <var name="result" type="searchResult"/>
  <input>
    <infoIn type="BookQuery">
      <infoName>aQuery</infoName>
      <xpath>/bookQuery</xpath>
    </infoIn>
  </input>
  <output>
    <infoOut writemode="mergeRoot">
      <infoName>result</infoName>
      <xpath>/results</xpath>
    </infoOut>
  </output>
  <wrapper>
    <wrapperName>book-amazon</wrapperName>
    <wrapperInput>aQuery</wrapperInput>
    <wrapperOutput>result</wrapperOutput>
  </wrapper>
</wrapperTask>

<wrapperTask taskid="T022">
  <name>Access Chapters</name>
  <description>Access the Chapters website</description>
  <agent>A01</agent>
  <link>FixData</link>
  <link>ClearSorting</link>
  <link>ClearGrouping</link>
  <link>ClearFiltering</link>
  <var name="result" type="searchResult"/>
  <input>
    <infoIn type="BookQuery">
      <infoName>aQuery</infoName>
      <xpath>/bookQuery</xpath>
    </infoIn>
  </input>
  <output>
    <infoOut writemode="mergeRoot">
      <infoName>result</infoName>
      <xpath>/results</xpath>
    </infoOut>
  </output>
  <wrapper>
    <wrapperName>book-chapters</wrapperName>
    <wrapperInput>aQuery</wrapperInput>
    <wrapperOutput>result</wrapperOutput>
  </wrapper>
</wrapperTask>

<wrapperTask taskid="T023">
  <name>Access 1Book</name>
  <description>Access the 1Book website</description>
  <agent>A01</agent>
  <link>FixData</link>
  <link>ClearSorting</link>
  <link>ClearGrouping</link>
  <link>ClearFiltering</link>
  <var name="result" type="searchResult"/>
  <input>
    <infoIn type="BookQuery">
      <infoName>aQuery</infoName>
      <xpath>/bookQuery</xpath>
    </infoIn>
  </input>
  <output>
    <infoOut writemode="mergeRoot">
      <infoName>result</infoName>
      <xpath>/results</xpath>
    </infoOut>
  </output>
  <wrapper>
    <wrapperName>book-1book</wrapperName>
    <wrapperInput>aQuery</wrapperInput>
    <wrapperOutput>result</wrapperOutput>
  </wrapper>
</wrapperTask>

<groupingTask taskid="T03">
<name>Sort Books</name>
<description>Select another task</description>
<subtask taskref="T031" name="Sort by Title" />
<subtask taskref="T032" name="Sort by Author" />
<subtask taskref="T033" name="Sort by Type" />
<subtask taskref="T034" name="Sort by Price" />
<subtask taskref="T035" name="Sort by Publisher" />
<subtask taskref="T036" name="Sort by Origin" />
</groupingTask>

<internalTask taskid="T031">
  <name>Sort by Title</name>
  <description>Sorts book info by title</description>
  <link>T06</link>
  <clearActivation>T032</clearActivation>
  <clearActivation>T033</clearActivation>
  <clearActivation>T034</clearActivation>
  <clearActivation>T035</clearActivation>
  <clearActivation>T036</clearActivation>
  <input>
    <infoIn type="searchResult">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoIn>
  </input>
  <output>
    <infoOut writemode="overwrite">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoOut>
  </output>
  <xslt>
```

```
    <in var="true">aSearchResult</in>
    <xsl filePath="true">../../../taskModels/myBookBuying/sortByTitle.xsl</xsl>
    <out var="true">aSearchResult</out>
  </xslt>
</internalTask>

<internalTask taskid="T032">
  <name>Sort by Author</name>
  <description>Sorts book info by Author</description>
  <link>T06</link>
  <clearActivation>T031</clearActivation>
  <clearActivation>T033</clearActivation>
  <clearActivation>T034</clearActivation>
  <clearActivation>T035</clearActivation>
  <clearActivation>T036</clearActivation>
  <input>
    <infoIn type="searchResult">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoIn>
  </input>
  <output>
    <infoOut writemode="overwrite">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoOut>
  </output>
  <xslt>
    <in var="true">aSearchResult</in>
    <xsl filePath="true">../../../taskModels/myBookBuying/sortByAuthor.xsl</xsl>
    <out var="true">aSearchResult</out>
  </xslt>
</internalTask>

<internalTask taskid="T033">
  <name>Sort by Type</name>
  <description>Sorts book info by Type</description>
  <link>T06</link>
  <clearActivation>T031</clearActivation>
  <clearActivation>T032</clearActivation>
  <clearActivation>T034</clearActivation>
  <clearActivation>T035</clearActivation>
  <clearActivation>T036</clearActivation>
  <input>
    <infoIn type="searchResult">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoIn>
  </input>
  <output>
    <infoOut writemode="overwrite">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoOut>
  </output>
  <xslt>
    <in var="true">aSearchResult</in>
    <xsl filePath="true">../../../taskModels/myBookBuying/sortByType.xsl</xsl>
    <out var="true">aSearchResult</out>
  </xslt>
</internalTask>

<internalTask taskid="T034">
  <name>Sort by Price</name>
  <description>Sorts book info by Price</description>
  <link>T06</link>
  <clearActivation>T031</clearActivation>
  <clearActivation>T032</clearActivation>
  <clearActivation>T033</clearActivation>
  <clearActivation>T035</clearActivation>
  <clearActivation>T036</clearActivation>
  <input>
    <infoIn type="searchResult">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoIn>
  </input>
  <output>
    <infoOut writemode="overwrite">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoOut>
  </output>
  <xslt>
    <in var="true">aSearchResult</in>
    <xsl filePath="true">../../../taskModels/myBookBuying/sortByPrice.xsl</xsl>
    <out var="true">aSearchResult</out>
  </xslt>
</internalTask>

<internalTask taskid="T035">
  <name>Sort by Publisher</name>
  <description>Sorts book info by Publisher</description>
  <link>T06</link>
  <clearActivation>T031</clearActivation>
  <clearActivation>T032</clearActivation>
  <clearActivation>T033</clearActivation>
  <clearActivation>T034</clearActivation>
  <clearActivation>T036</clearActivation>
  <input>
    <infoIn type="searchResult">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoIn>
  </input>
  <output>
    <infoOut writemode="overwrite">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoOut>
  </output>
  <xslt>
    <in var="true">aSearchResult</in>
    <xsl filePath="true">../../../taskModels/myBookBuying/sortByPublisher.xsl</xsl>
    <out var="true">aSearchResult</out>
  </xslt>
</internalTask>

<internalTask taskid="T036">
  <name>Sort by Origin</name>
  <description>Sorts book info by Origin</description>
  <link>T06</link>
  <clearActivation>T031</clearActivation>
  <clearActivation>T032</clearActivation>
  <clearActivation>T033</clearActivation>
  <clearActivation>T034</clearActivation>
  <clearActivation>T035</clearActivation>
  <input>
    <infoIn type="searchResult">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoIn>
  </input>
  <output>
    <infoOut writemode="overwrite">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoOut>
  </output>
  <xslt>
    <in var="true">aSearchResult</in>
    <xsl filePath="true">../../../taskModels/myBookBuying/sortByOrigin.xsl</xsl>
    <out var="true">aSearchResult</out>
  </xslt>
</internalTask>

<groupingTask taskid="T04">
  <name>Group Books</name>
  <description>Routines that group books according to various
categories</description>
  <subtask taskref="T041" name="Group by Type"/>
  <subtask taskref="T042" name="Group by Author"/>
  <subtask taskref="T043" name="Group by Origin"/>
  <subtask taskref="T044" name="Group by Publisher"/>
</groupingTask>

<internalTask taskid="T041">
  <name>Group by Type</name>
  <description>Groups book info by Type</description>
  <link>ClearSorting</link>
  <input>
    <infoIn type="searchResult">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoIn>
  </input>
  <output>
    <infoOut writemode="overwrite">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoOut>
  </output>
  <xslt>
    <in var="true">aSearchResult</in>
    <xsl filePath="true">../../../taskModels/myBookBuying/groupByType.xsl</xsl>
    <out var="true">aSearchResult</out>
  </xslt>
</internalTask>

<internalTask taskid="T042">
  <name>Group by Author</name>
  <description>Groups book info by Author</description>
  <link>ClearSorting</link>
  <input>
    <infoIn type="searchResult">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoIn>
  </input>
  <output>
    <infoOut writemode="overwrite">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoOut>
  </output>
  <xslt>
    <in var="true">aSearchResult</in>
    <xsl filePath="true">../../../taskModels/myBookBuying/groupByAuthor.xsl</xsl>
    <out var="true">aSearchResult</out>
  </xslt>
</internalTask>

<internalTask taskid="T043">
  <name>Group by Origin</name>
  <description>Groups book info by Site of origin.</description>
  <link>ClearSorting</link>
  <input>
    <infoIn type="searchResult">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoIn>
  </input>
  <output>
    <infoOut writemode="overwrite">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoOut>
  </output>
  <xslt>
    <in var="true">aSearchResult</in>
    <xsl filePath="true">../../../taskModels/myBookBuying/groupByOrigin.xsl</xsl>
    <out var="true">aSearchResult</out>
  </xslt>
</internalTask>

<internalTask taskid="T044">
  <name>Group by Publisher</name>
  <description>Groups book info by Publisher</description>
  <link>ClearSorting</link>
  <input>
    <infoIn type="searchResult">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoIn>
  </input>
  <output>
    <infoOut writemode="overwrite">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoOut>
  </output>
  <xslt>
    <in var="true">aSearchResult</in>
    <xsl
filePath="true">../../../taskModels/myBookBuying/groupByPublisher.xsl</xsl>
    <out var="true">aSearchResult</out>
  </xslt>
</internalTask>

<groupingTask taskid="T05">
  <name>Filter Books</name>
  <description>Filters out and in books.</description>
  <subtask taskref="T051" name="Filter books by Type"/>
  <subtask taskref="T052" name="Filter books by Origin"/>
  <subtask taskref="T053" name="Filter books by Publisher"/>
  <subtask taskref="T054" name="Filter books by Author"/>
  <subtask taskref="T055" name="Filter books by Title"/>
  <subtask taskref="T056" name="Filter books by Price"/>
</groupingTask>

<inputTask taskid="T051">
  <name>Filter by Type</name>
  <description>Enter the filter criteria.</description>
  <link>T051b</link>
  <var name="aTypeFilter" type="typeFilter"/>
  <output>
    <infoOut writemode="overwrite">
      <infoName>aTypeFilter</infoName>
      <xpath>/typeFilter</xpath>
    </infoOut>
  </output>
  <xslt>
    <in filePath="true">../../../taskModels/myBookBuying/t01input.xml</in>
    <xsl filePath="true">../../../taskModels/myBookBuying/filterByType.xsl</xsl>
  </xslt>
</inputTask>

<internalTask taskid="T051b">
  <name>Filter by Type</name>
  <description>Filters book info by type.</description>
  <link>filterByTypeMessage</link>
  <input>
    <infoIn type="searchResult">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoIn>
    <infoIn type="typeFilter">
      <infoName>aTypeFilter</infoName>
      <xpath>/typeFilter</xpath>
    </infoIn>
  </input>
  <output>
    <infoOut writemode="overwrite">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoOut>
  </output>
  <xslt>
    <in var="true">aSearchResult</in>
    <in var="true">aTypeFilter</in>
    <xsl
filePath="true">../../../taskModels/myBookBuying/filterByTypeWork.xsl</xsl>
    <out var="true">aSearchResult</out>
  </xslt>
</internalTask>

<outputTask taskid="filterByTypeMessage">
  <name>FilterByTypeMessage</name>
  <description>Displays a short message saying that the search results have been
filtered by type.</description>
  <outputStatic>and the Result set has been filtered by Type.</outputStatic>
</outputTask>


<inputTask taskid="T052">
  <name>Filter by Origin</name>
  <description>Enter the filter criteria.</description>
  <link>T052b</link>
  <var name="anOriginFilter" type="originFilter"/>
  <output>
    <infoOut writemode="overwrite">
      <infoName>anOriginFilter</infoName>
      <xpath>/originFilter</xpath>
    </infoOut>
  </output>
  <xslt>
    <in filePath="true">../../../taskModels/myBookBuying/t01input.xml</in>
    <xsl filePath="true">../../../taskModels/myBookBuying/filterByOrigin.xsl</xsl>
  </xslt>
</inputTask>

<internalTask taskid="T052b">
  <name>Filter by Origin</name>
  <description>Filters book info by origin.</description>
  <link>filterByOriginMessage</link>
  <input>
    <infoIn type="searchResult">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoIn>
    <infoIn type="originFilter">
      <infoName>anOriginFilter</infoName>
      <xpath>/originFilter</xpath>
    </infoIn>
  </input>
  <output>
    <infoOut writemode="overwrite">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoOut>
  </output>
  <xslt>
    <in var="true">aSearchResult</in>
    <in var="true">anOriginFilter</in>
    <xsl
filePath="true">../../../taskModels/myBookBuying/filterByOriginWork.xsl</xsl>
    <out var="true">aSearchResult</out>
  </xslt>
</internalTask>

<outputTask taskid="filterByOriginMessage">
  <name>Filter By Origin Message</name>
  <description>Displays a short message saying that the search results have been
filtered by Origin.</description>
  <outputStatic>and the Result set has been filtered by Origin.</outputStatic>
</outputTask>

<inputTask taskid="T053">
  <name>Filter by Publisher</name>
  <description>Enter the filter criteria.</description>
  <link>T053b</link>
  <var name="aPublisherFilter" type="publisherFilter"/>
  <output>
    <infoOut writemode="overwrite">
      <infoName>aPublisherFilter</infoName>
      <xpath>/publisherFilter</xpath>
    </infoOut>
  </output>
  <xslt>
    <in filePath="true">../../../taskModels/myBookBuying/t01input.xml</in>
    <xsl
filePath="true">../../../taskModels/myBookBuying/filterByPublisher.xsl</xsl>
  </xslt>
</inputTask>

<internalTask taskid="T053b">
  <name>Filter by Publisher</name>
  <description>Filters book info by Publisher.</description>
  <link>filterByPublisherMessage</link>
  <input>
    <infoIn type="searchResult">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoIn>
    <infoIn type="publisherFilter">
      <infoName>aPublisherFilter</infoName>
      <xpath>/publisherFilter</xpath>
    </infoIn>
  </input>
  <output>
    <infoOut writemode="overwrite">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoOut>
  </output>
  <xslt>
    <in var="true">aSearchResult</in>
    <in var="true">aPublisherFilter</in>
    <xsl
filePath="true">../../../taskModels/myBookBuying/filterByPublisherWork.xsl</xsl>
    <out var="true">aSearchResult</out>
  </xslt>
</internalTask>

<outputTask taskid="filterByPublisherMessage">
  <name>FilterByPublisherMessage</name>
  <description>Displays a short message saying that the search results have been
filtered by publisher.</description>
  <outputStatic>and the Result set has been filtered by Publisher.</outputStatic>
</outputTask>

<inputTask taskid="T054">
  <name>Filter by Author</name>
  <description>Enter the filter criteria.</description>
  <link>T054b</link>
  <var name="anAuthorFilter" type="authorFilter"/>
  <output>
    <infoOut writemode="overwrite">
      <infoName>anAuthorFilter</infoName>
      <xpath>/authorFilter</xpath>
    </infoOut>
  </output>
  <xslt>
    <in filePath="true">../../../taskModels/myBookBuying/t01input.xml</in>
    <xsl filePath="true">../../../taskModels/myBookBuying/filterByAuthor.xsl</xsl>
  </xslt>
</inputTask>

<internalTask taskid="T054b">
  <name>Filter by Author</name>
  <description>Filters book info by author.</description>
  <link>filterByAuthorMessage</link>
  <input>
    <infoIn type="searchResult">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoIn>
    <infoIn type="authorFilter">
      <infoName>anAuthorFilter</infoName>
      <xpath>/authorFilter</xpath>
    </infoIn>
  </input>
  <output>
    <infoOut writemode="overwrite">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoOut>
  </output>
  <xslt>
    <in var="true">aSearchResult</in>
    <in var="true">anAuthorFilter</in>
    <xsl
filePath="true">../../../taskModels/myBookBuying/filterByAuthorWork.xsl</xsl>
    <out var="true">aSearchResult</out>
  </xslt>
</internalTask>

<outputTask taskid="filterByAuthorMessage">
  <name>FilterByAuthorMessage</name>
  <description>Displays a short message saying that the search results have been
filtered by author.</description>
  <outputStatic>and the Result set has been filtered by Author.</outputStatic>
</outputTask>

<inputTask taskid="T055">
  <name>Filter by Title</name>
  <description>Enter the filter criteria.</description>
  <link>T055b</link>
  <var name="aTitleFilter" type="titleFilter"/>
  <output>
    <infoOut writemode="overwrite">
      <infoName>aTitleFilter</infoName>
      <xpath>/titleFilter</xpath>
    </infoOut>
  </output>
  <xslt>
    <in filePath="true">../../../taskModels/myBookBuying/t01input.xml</in>
    <xsl filePath="true">../../../taskModels/myBookBuying/filterByTitle.xsl</xsl>
  </xslt>
</inputTask>

<internalTask taskid="T055b">
  <name>Filter by Title</name>
  <description>Filters book info by title.</description>
  <link>filterByTitleMessage</link>
  <input>
    <infoIn type="searchResult">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoIn>
    <infoIn type="titleFilter">
      <infoName>aTitleFilter</infoName>
      <xpath>/titleFilter</xpath>
    </infoIn>
  </input>
  <output>
    <infoOut writemode="overwrite">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoOut>
  </output>
  <xslt>
    <in var="true">aSearchResult</in>
    <in var="true">aTitleFilter</in>
    <xsl
filePath="true">../../../taskModels/myBookBuying/filterByTitleWork.xsl</xsl>
    <out var="true">aSearchResult</out>
  </xslt>
</internalTask>

<outputTask taskid="filterByTitleMessage">
  <name>FilterByTitleMessage</name>
  <description>Displays a short message saying that the search results have been
filtered by title.</description>
  <outputStatic>and the Result set has been filtered by Title.</outputStatic>
</outputTask>

<inputTask taskid="T056">
  <name>Filter by Price</name>
  <description>Enter the min and max of price range</description>

<functionalConstraints>../../../taskModels/myBookBuying/filterByPriceFconstraints.x
sl</functionalConstraints>
  <link>T056b</link>
  <var name="aPriceMin" type="priceMin"/>
  <var name="aPriceMax" type="priceMax"/>

<conceptualConstraints>../../../taskModels/myBookBuying/filterByPriceCconstraints.x
sl</conceptualConstraints>
  <output>
    <infoOut writemode="overwrite">
      <infoName>aPriceMin</infoName>
      <xpath>/priceMin</xpath>
    </infoOut>
    <infoOut writemode="overwrite">
      <infoName>aPriceMax</infoName>
      <xpath>/priceMax</xpath>
    </infoOut>
  </output>
  <xslt>
    <in filePath="true">../../../taskModels/myBookBuying/t01input.xml</in>
    <xsl filePath="true">../../../taskModels/myBookBuying/filterByPrice.xsl</xsl>
  </xslt>
</inputTask>

<internalTask taskid="T056b">
  <name>Filter books by Price</name>
  <description>Filters book info by price.</description>
  <link>filterByPriceMessage</link>
  <input>
    <infoIn type="searchResult">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoIn>
    <infoIn type="priceMin">
      <infoName>aPriceMin</infoName>
      <xpath>/priceMin</xpath>
    </infoIn>
    <infoIn type="priceMax">
      <infoName>aPriceMax</infoName>
      <xpath>/priceMax</xpath>
    </infoIn>
  </input>
  <output>
    <infoOut writemode="overwrite">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoOut>
  </output>
  <xslt>
    <in var="true">aSearchResult</in>
    <in var="true">aPriceMin</in>
    <in var="true">aPriceMax</in>
    <xsl
filePath="true">../../../taskModels/myBookBuying/filterByPriceWork.xsl</xsl>
    <out var="true">aSearchResult</out>
  </xslt>
</internalTask>

<outputTask taskid="filterByPriceMessage">
  <name>FilterByPriceMessage</name>
  <description>Displays a short message saying that the search results have been
filtered by price.</description>
  <outputStatic>and the Result set has been filtered by Price.</outputStatic>
</outputTask>

<outputTask taskid="T06">
  <name>View current</name>
  <description>View current book list.</description>
  <input>
    <infoIn type="searchResult">
      <infoName>result</infoName>
      <xpath>/results</xpath>
    </infoIn>
  </input>
  <xslt>
    <in var="true">result</in>
    <xsl filePath="true">../../../taskModels/myBookBuying/displayBooks.xsl</xsl>
  </xslt>
</outputTask>

<internalTask taskid="T07">
  <name>Clear Results</name>
  <description>Clears all of the current results.</description>
  <link>ClearResultsMessage</link>
  <link>ClearAlmostAllActivation</link>
  <input>
    <infoIn type="searchResult">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoIn>
  </input>
  <output>
    <infoOut writemode="overwrite">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoOut>
  </output>
  <xslt>
    <in var="true">aSearchResult</in>
    <xsl
filePath="true">../../../taskModels/myBookBuying/clearSearchResults.xsl</xsl>
    <out var="true">aSearchResult</out>
  </xslt>
</internalTask>

<internalTask taskid="T08">
  <name>Reload Original</name>
  <description>Reload Original Book List</description>
  <input>
    <infoIn type="searchResult">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoIn>
  </input>
  <output>
    <infoOut writemode="overwrite">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoOut>
  </output>
  <xslt>
    <in var="true">aSearchResult</in>
    <xsl
filePath="true">../../../taskModels/myBookBuying/reloadOriginalSet.xsl</xsl>
    <out var="true">aSearchResult</out>
  </xslt>
</internalTask>

<outputTask taskid="ClearResultsMessage">
  <name>ClearResultsMessage</name>
  <description>Displays a short message saying that the search results have been
cleared.</description>
  <outputStatic>The book set has been cleared, you will need to access the
resources again in order to continue with this interaction.</outputStatic>
</outputTask>

<internalTask taskid="ClearAlmostAllActivation">
  <name>Clear Almost All Activations</name>
  <description>Clears the activation and started flags from all of the tasks except
for the input query one.</description>

  <link>ClearResourceAccess</link>
  <link>ClearSorting</link>
  <link>ClearGrouping</link>
  <link>ClearFiltering</link>

  <clearActivation>T06</clearActivation>
  <clearActivation>T08</clearActivation>
</internalTask>

<internalTask taskid="ClearAllActivation">
  <name>Clear All Activations</name>
  <description>Clears the activation and started flags from all of the
tasks.</description>

  <link>ClearAlmostAllActivation</link>
  <clearActivation>T01</clearActivation>
  <clearActivation>T07</clearActivation>
</internalTask>

<internalTask taskid="ClearResourceAccess">
  <name>Clear Resource Accesses</name>
  <description>Clears the activation of resource access tasks.</description>

  <clearActivation>T02</clearActivation>
  <clearActivation>T021</clearActivation>
  <clearActivation>T022</clearActivation>
  <clearActivation>T023</clearActivation>
</internalTask>

<internalTask taskid="ClearSorting">
  <name>Clear Sorting Tasks</name>
  <description>Clears the activation of sorting tasks.</description>

  <clearActivation>T03</clearActivation>
  <clearActivation>T031</clearActivation>
  <clearActivation>T032</clearActivation>
  <clearActivation>T033</clearActivation>
  <clearActivation>T034</clearActivation>
  <clearActivation>T035</clearActivation>
  <clearActivation>T036</clearActivation>
</internalTask>

<internalTask taskid="ClearGrouping">
  <name>Clear Grouping Tasks</name>
  <description>Clears the activation of grouping tasks.</description>

  <clearActivation>T04</clearActivation>
  <clearActivation>T041</clearActivation>
  <clearActivation>T042</clearActivation>
  <clearActivation>T043</clearActivation>
  <clearActivation>T044</clearActivation>
</internalTask>

<internalTask taskid="ClearFiltering">
  <name>Clear Filtering Tasks</name>
  <description>Clears the activation of filtering tasks.</description>

  <clearActivation>T05</clearActivation>
  <clearActivation>T051</clearActivation>
  <clearActivation>T052</clearActivation>
  <clearActivation>T053</clearActivation>
</internalTask>

<internalTask taskid="FixData">
  <name>Fix data</name>
  <description>Makes sure that wrapper data is of a valid format and
coolated.</description>
  <input>
```

```
    <infoIn type="searchResult">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoIn>
  </input>
  <output>
    <infoOut writemode="overwrite">
      <infoName>aSearchResult</infoName>
      <xpath>/results</xpath>
    </infoOut>
  </output>
  <xslt>
    <in var="true">aSearchResult</in>
    <xsl filePath="true">../../../taskModels/myBookBuying/fixData.xsl</xsl>
    <out var="true">aSearchResult</out>
  </xslt>
</internalTask>

<outputTask taskid="T09">
  <name>Show Explorer Applet</name>
  <description>Shows an applet that allows you to view and categorize the
books.</description>
  <input>
    <infoIn type="searchResult">
      <infoName>result</infoName>
      <xpath>/results</xpath>
    </infoIn>
  </input>
  <xslt>
    <in var="true">result</in>
    <xsl filePath="true">../../../taskModels/myBookBuying/showApplet.xsl</xsl>
  </xslt>
</outputTask>

</taskModel>
```

Example Functional Constraint XSLT Stylesheet

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output method="xml"/>
<xsl:template match="/">

   <xsl:if test="number(/Vars/aPriceMin/priceMin) &gt;
number(/Vars/aPriceMax/priceMax)">
      <xsl:text> The minimum price is greater than the maximum price  </xsl:text>
      <xsl:text> The minimum price is  </xsl:text>
      <xsl:value-of select="/Vars/aPriceMin/priceMin"/>
      <xsl:text> The maximum price is  </xsl:text>
      <xsl:value-of select="/Vars/aPriceMax/priceMax"/>
   </xsl:if>

</xsl:template>
</xsl:stylesheet>
```