Chapter 9
Patterns in Web-Based Information Systems

Abstract Day-to-day experiences suggest that it is not enough to approach Web-application engineering armed with Web guidelines and user manuals on how to use the underlying technologies, such as Java and Simple Object Access Protocol (SOAP). Many of the Web designer and user problems tend to recur in various projects. Web designers often try to reinvent design solutions from scratch. Developers must be able to use proven design solutions emerging from the best design practices to solve common problems. Without this, the designer will not properly apply Web guidelines, or take full benefit of the power of Web technology, resulting in poor performance, poor scalability, and poor usability of the developed applications. In this chapter, we introduce different types of Web design patterns as a vehicle for capturing and disseminating good designs while detailing a motivating example on how Web design patterns can be combined to create a home page. Our investigations are based on several years of Web applications development, ethnographic interviews with Web developers, as well as suggestions from others. Such suggestions include reported best practices for using patterns as a bridge over the gaps between the design practices and software tools. Our experiences also highlighted that in order to render the patterns understandable by novice designers and software engineers who are unfamiliar with Web engineering, patterns should be represented to developers using a flexible structure, to make it easy for the pattern authors, reviewers, and users.

9.1 Introduction

A complex Web site is the place where all the elements of a company, such as graphic identity, customer service, products, services, and structure, come together in one place. Interacting with a Web site conveys much more than the graphic brand identity of the company—it is all about the user experience of moving through the site and interacting with all of its various parts. Web design must incorporate what people do on the site rather than simply how it looks. More consideration toward the user's experience and interactions with the website are necessary, such as how the site is perceived, learned, and mastered. This includes ease-of-use and, most importantly, the needs that the site should fulfill with respect to services and information.

The design of the website should focus on the behavior of users. Consequently, we must analyze and understand users, and provide designs based on user experiences and persona.

For some complex Web sites such as e-commerce, online banking, and educational systems, as well as Web sites for a specific set of users such as scientists, usability is recognized as a key element of the site's success and acceptance by its end users. The bad news is that most of such sites employ horribly misguided methodologies that do not assess real usability. A good methodological framework should address the following concern: More and more, Web sites are designed for an international audience and for universal usability. In this context, a universal design approach should be adopted to accommodate the vast majority of the global population. This entails addressing challenges of technology variety, user diversity, and gaps in user knowledge in ways only beginning to be acknowledged by educational, corporate, and government agencies (Shneiderman 2000).

Web applications are moving away from the paradigm of an online-type brochure with a static presentation to highly interactive web-based software systems. With the advent of Web scripting languages and document object models, the user has been given more sophisticated techniques to interact with server-side services and information. In addition, the convergence of the internet and mobile device technologies has led to the emergence of a new generation of web applications. One of the major characteristics of these new applications is that they allow a user to interact with services using different kinds of computers and devices, with particularities in interaction style. These devices include the traditional office desktop, laptop, palmtop, personal digital assistant (PDA) with and without keyboard, mobile telephone, and interactive television.

In this new technological context, we can distinguish four kinds of interactive Web applications with four different styles of user interfaces:

Traditional Web applications that are based on the Web browser. Most popular Web applications are Web sites, corporate intranet environments, portals, and e-commerce applications;

Embeddable Web services. Examples of basic services can include open uniform resource locator (URL) from FrontPage Explorer, browse through links in a portable document format (PDF) file using Adobe Acrobat and send mails from most Microsoft productive tools;

Web applications that offer an optimized and specialized user interface to a set of Internet features. An example of such applications is e-mail that provides an example of the interplay between specialized applications and toned-down web applications. It is often possible to access your email through a Simple Mail Transfer Protocol (SMTP) client provided as an accessory to the browser (such as Outlook in Microsoft Internet Explorer and Messenger in Netscape Communicator). The same applies to Network News Transfer Protocol (NNTP) clients that enable a user to access Usenet newsgroups on the Internet;

Resource constrained Web applications for small and mobile devices that cannot support the full range of Web application features, because of the lack of screen *space or low bandwidth*. These applications include mobile telephone-embedded Internet applications (read e-mail, browse mobile portal sites, etc.).

This mosaic of applications has led to the emergence of Web engineering as a subdiscipline of software engineering for creating high-quality Web interactive systems. Web engineering is governed by its own set of fundamental principles, even if it borrows many of the software engineering methodologies and theories, and emphasizes similar technical and management activities. There are subtle differences in the way these activities are conducted, but an overriding philosophy dictating a disciplined approach to the development of the system is identical. As an example, we can adapt and refine pattern-assisted engineering to the development of Web applications.

Our objective can be stated as follows: To facilitate the engineering of Web applications while improving the usability and quality of the developed applications; by composing different kinds of patterns such as architectural, navigational, and interaction patterns to create a high level and reusable design with the goal to support the generation of application code. A subobjective of our research is to define a systematic methodology, supported by a Web CASE tool, to glue the patterns together.

9.2 Design Challenges of Web Applications

The following are some of the challenges associated with web design that we are addressing:

First, in an attempt to segment the different aspects of Web application architecture and isolate platform specifics from remaining issues, the vast majority of current industry Web applications have adopted a layered approach. As with other multitiered schemes such as client-server architecture, a common information repository is at the core of the architecture. The repository is accessed strictly through this layer, which in addition to the functions listed, also provides decoupling of the data from the device specific interfaces. In this way, device application developers need to only worry about the standardized middleware interface rather than having to concern themselves with the multitude of application programming interfaces (APIs) put forth by database repository manufacturers. Segmenting the architecture and reducing coupling to stringent specifications allows the designer to quickly understand how changes made to a particular component effects the remaining system. That is because achieving these goals requires a consistent approach to applying both cognitive and social factors to UI design, and that would require independent developers to coordinate their activities. Unfortunately, conspiring at this level may be beyond the abilities of the industry.

Second, web applications are efficient at managing heterogeneous environments. This point is critical, as more and more Web applications will need to interact with very different platforms and devices. This diversity results in computing devices that exhibit drastically different capabilities. For example, PDAs use a pen based input mechanism and have average screen sizes in the range of 3 inches. On the other hand, the typical PC uses a full size keyboard, a mouse, and has an average screen size of 17 inches. Coping with such drastic variations implies much more than mere layout changes. Pen-based input mechanism are slower than traditional keyboards and thus are inappropriate for applications such as word processing that require intensive user input. Similarly the small screens available on many PDAs only provide coarse graphic capabilities and thus would be ill suited for photo editing applications.

Another challenge is that heterogeneity in the computing platform ranging from traditional desktop to mobile phone via PDA is the source of a further complication in Web applications engineering. Certain form factors are better suited to particular contexts. For example, walking down the street, one user may use her mobile telephone's Internet browser to lookup a stock quote. However, it is highly unlikely that this same user reviews the latest changes made to a document using the same device. Rather, it would seem more logical and definitely more practical to use a full size computer for this task. It would therefore seem that the context of use is determined by a combination of internal and external factors. The internal factors primarily relate to the user's attention while performing a task. In some cases, the user may be entirely focused, while at other times greatly distracted by other concurrent tasks. An example of this latter point is when a user—while driving a car—operates a PDA to reference a telephone number. External factors are determined to a large extent by the device's physical characteristics. It is not possible to make use of a traditional PC as one walks down the street—a practice quite common with a mobile telephone. The challenge for a system architect is thus to match the design of a particular device's UI with the set of constraints imposed by the corresponding context of use.

Finally, many system manufacturers and researchers have issued design guidelines to Web-application designers (Lynch and Horton 1999). Recently, Palm Inc. has put forth design guidelines to address the navigation issues, widget selection, and use of specialized input mechanisms such as handwriting recognition. (Macintosh 1992; Microsoft 1995; IBM 2015; Sun Microsystems 2001) have also published their own usability guidelines to assist developers with programming applications targeted at the Pocket PC/Windows CE platform. However, these guidelines are different from one platform or device to another. When designing a multidevice Web application, this can be a source of many inconsistencies. The Java "look-and-feel" developed by Sun is a set of cross-platform guidelines that can fix such problems. However, cross-platform guidelines do not take into account the particularities of a specific device, in particular the platform constraints and capabilities. This can be a source of problems for a user using different kinds of devices to interact with the server side services and information of a Web application. Furthermore, for a novice designer or a software engineer who is not familiar with this mosaic of guidelines, it is hard to remember all design guidelines, let alone using them effectively. It is sometimes difficult to make the trade-offs among these principles when they come into conflict; we often have to figure out the best solution by guessing, or by resorting to other means.

9.3 Web Design Principles

Patterns also are not enough to design usable and useful Web applications. They need to be used in junction with high-level design. For example, the design of the previous home page, we also used the following rules:

- *Organize the page for scanning*. The homepage is organized to be help users scan down the page, trying to find the area that will serve their current goal. Links are the action items on a homepage, and when you start each link with a relevant word, you make it easier for scanning eyes to differentiate it from other links on the page. A common violation of this guideline is to start all links with the company name, which adds little value and impairs users ability to quickly find what they need;
- *Provide clear affordance of links*. Navigation elements, in particular, links must provide clear affordance. Their appearance should help users understand them. The mouse pointer change provided by web browsers, to indicate that the element pointed at a link is not sufficient. The designer can use differences in size to establish a hierarchy between links, but HyperText Markup Language (HTML) text has poor graphic quality and doesn't allow much visual characterization. Differentiation between navigation elements and information is indeed the main affordance problem to be solved;
- *Strive to avoid users making errors***.** We should provide alternative links so that the user can recover from the error quickly and easily, communicate in the user's vocabulary, use Web server, which allows you to customize the error messages. If an error occurs, tell users what the error is, why it occurred, what they can do to fix it.

The same design approach can be applied to the following patterns of pages that we generally use to design a Web site:

1. Central page (one or more)—Central page of your site from which all, other pages can be reached (directly or indirectly). The home page is a specialization of such page. For a large Web site, we can have more than one central page (e.g. university, department, research group, personal web sites).
2. Navigation pages for directing the user to the proper area of your site for the information they are seeking.
3. Content pages provide the information users are seeking when they visit your site. They may also contain navigational links to give users a sense of location within the site and allow them to progress to more information or return to a previous page.
4. Input page (transaction forms, search, feedback) is to collect information from users or establish a dialog with the user.
5. Utilities pages (bookmarks, extra things, help, archive, configuration, etc.).

One of the major problems we found is that mastering and applying type patterns and a large collection of patterns require in-depth knowledge of both the problems and forces at play, and most importantly must ultimately put forth battle-tested solutions. As such, it is inconceivable that pattern hierarchies will evolve strictly from theoretical considerations. Practical research and industry feedback are crucial in determining how successful a pattern-oriented design framework is at solving real-world problems. It is therefore essential to build an "academia–industry bridge" by establishing formal communication channels between industrial specialists in human–computer interaction (HCI) patterns, software design patterns, information architecture patterns as well as software pattern researchers. Such collaboration will lead to a common terminology which is essential for making the large diversity of patterns accessible to common Web designers.

9.4 Case Study: A Detailed Discussion

9.4.1 Overview

This section presents a case study that describes the design of a functional user interface simplified prototype of an "Environmental Management Interactive System" (IFEN), illustrating and clarifying the core ideas of the Pattern-Oriented and Model-Driven Architecture (POMA) approach and its practical relevance.

This environmental management interactive system permits requirement analysis of the environment, its evolution, its economic and social dimensions, and proposes indicators of performance. The main objectives of environmental management are the treatment and distribution of water, improving air quality, monitoring noise, the treatment of waste, the health of fauna and flora, land use, preserving coastal and marine environments, and managing natural and technological risks (IFEN).

A simplified prototype of the "Environmental Management Interactive System" is developed here. The interactive system and corresponding models will not be tailored to different platforms. This prototype illustrates how patterns are used to establish the various models, as well as, the transformation of one model into another while respecting the pattern composition rules described in the Chap. 8 in Sect. 8.5.2, the pattern mapping rules described in Sect. 8.5.3 and the transformation rules described in Sect. 8.6.6.

This case study presents a general overview of the Platform-Independent Model (PIM) and Platform-Specific Model (PSM) models of the "Environmental Management Interactive System" by applying pattern composition steps and mapping rules, as well as transformation rules for the five models. The details of this illustrative case study are presented in this chapter in which the five models representing the same interactive system are illustrated on a laptop platform and on a PDA platform. The five models include the domain model, task model, dialog model, presentation model, and layout model of POMA architecture. Table 9.1 lists the patterns that will be used by the interactive system.

A prototype of a multiplatform interactive system for POMA architecture is implemented. A prototype is implemented in Java language using the Eclipse tool. There is a screenshot of the final layout of the "Environmental Management

Table 9.1 Pattern summary

Pattern name	Model type	Problem
Login	Domain	The user's identity needs to be authenticated in order to be allowed access to protected data and/or to perform authorized operations
Multivalue input form (Seffah and Gaffar 2006)	Domain	The user needs to enter a number of related values. These values can be of different data types, such as "date," "string," or "real"
Submit	Domain	The user needs to submit coordinates to the authentication process to access the system
Feedback	Domain	The user needs help concerning the use of the Login Form
Close	Domain	The need to close the system from the Login form
Find (search, browse, executive summary) (Seffah and Gaffar 2006)	Task	The need to find indicators related to the task concerned, to find environmental patterns related to the indicators, and to find a presentation tool to display the results of the indicators and the environmental patterns
Breadcrumb (Path)	Task	The need to construct and display the path that combines the data source, task, and/or subtask
Index browsing	Task	The need to display all indicators listed as index browsing to navigate and select the desired ones
Adapter	Task	The need to convert the interface of a class into another interface that clients expect; an adapter lets classes work together which could not otherwise be done because of interface incompatibility
Builder	Task	The need to separate the construction of a complex object from its representation, so that the same construction process can create different representations
List	Task	The need to display the information using forms
Table	Task	The need to display the information in tables
Map	Task	The need to display the information in geographic maps
Graph	Task	The need to display the information in graphs
Home page	Task	The need to define the layout of an interactive system home page, which is important because the home page is the interactive system interface with the world and the starting point for most user visits
Wizard (Welie 2004) and (Sinnig 2004)	Dialog	The user wants to achieve a single goal, but several decisions and actions need to be taken consecutively before the goal can be achieved
Recursive activation (Seffah and Gaffar 2006)	Dialog	The user wants to activate and manipulate several instances of a dialog view
Unambiguous format (Seffah and Gaffar 2006)	Presentation	The user needs to enter data, but may be unfamiliar with the structure of the information and/or its syntax
Form (Seffah and Gaffar 2006)	Presentation	The user must provide structural textual information to the system. The data to be provided are logically related
House (Seffah and Gaffar 2006)	Layout	Usually, the system consists of several pages/windows. The user should have the impression that it all "hangs together" and looks like one entity

Fig. 9.1 Graphical representation of the pattern

Interactive System" illustrated in Fig. 9.24. The key features of the current version of this interactive system prototype are the following:

- Support for well-arranged graphical specifications of hierarchy of POMA networks. This is achieved by the notion of a so-called tree explorer, in which the hierarchy of networks can be easily viewed and managed;
- Support for checking the correctness of network dependencies at the syntactic level. The editor contains a list of inputs and an output port for each network in the hierarchy and helps the user bind the right subsystem ports to the higher ports in the network hierarchy;
- Together with architectural compatibility checking, the prototype will allow one to easily define new POMA models by composing and mapping patterns which have already been defined and formalized.

Figure 9.1 shows the graphical representation of the pattern, which is used to exemplify the pattern in this case study.

9.4.2 *Defining the Domain Model*

Acting in the horizontal line of the POMA architecture (Fig. 8.2) in Chap. 8, this model is composed of two types of submodels, [POMA.PIM]-independent domain submodel and [POMA.PSM]-specific domain submodel.

The [POMA.PIM]-independent domain submodel (Fig. 9.2) is obtained by composing patterns and applying the composition rules.

The following example shows the composition of a "**Close**" pattern in eXtensible Markup Language (XML) language:

```
  /* XML
  <!xml version="1.0" >
  <d-class name="Close"
       ….
  Compose-to="xml.Jbutton">
       ….
</d-class>
</xml>
```

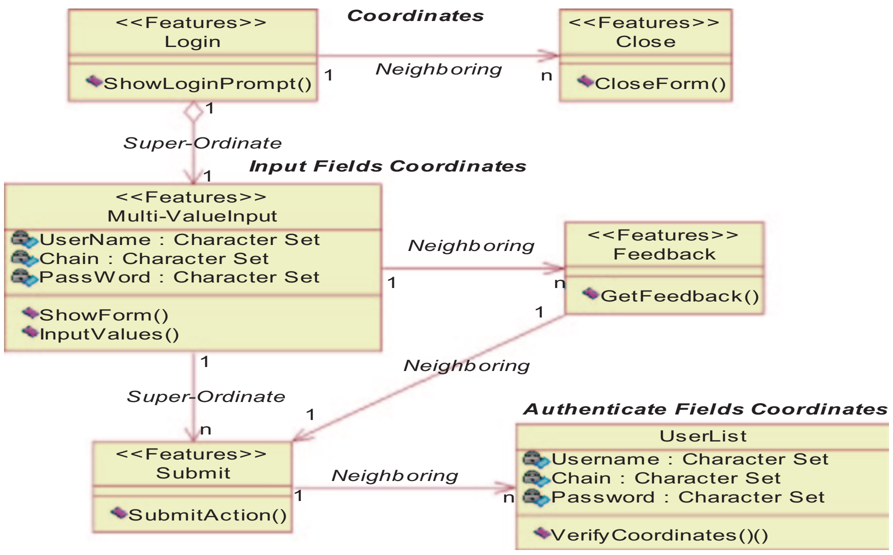


Fig. 9.2 Unified Modeling Language (UML) class diagram of the platform-independent model (PIM) domain model

The [POMA.PSM]-specific domain submodel (Figs. 9.3 and 9.4) is obtained by mapping composed patterns and applying the mapping rules (Table 9.2). This latter model would be used to generate the interactive system's source code by taking into account the generation code rules for a Microsoft platform.

Table 9.2 shows the mapping rules for the domain model patterns for a laptop and PDA platforms.

Therefore, the mapped domain model is obtained. An example of the mapping of a *Close* pattern in Java language follows:

```
    /* Java
    <d-class name="Close"
          ....
    Maps-to="javax.swing.Jbutton">
          ....
</d-class>
```

After the mapping, the PSM domain model is obtained for a laptop platform—Fig. 9.3 and for a PDA platform—Fig. 9.4.

To obtain feedback on the PDA platform, we need to insert *Next* and *Previous* patterns to obtain information in a number of smaller portal displays. The *Next* pattern enables one to access the next feedback information available, and the *Previous* pattern allows a return to the previous feedback information.

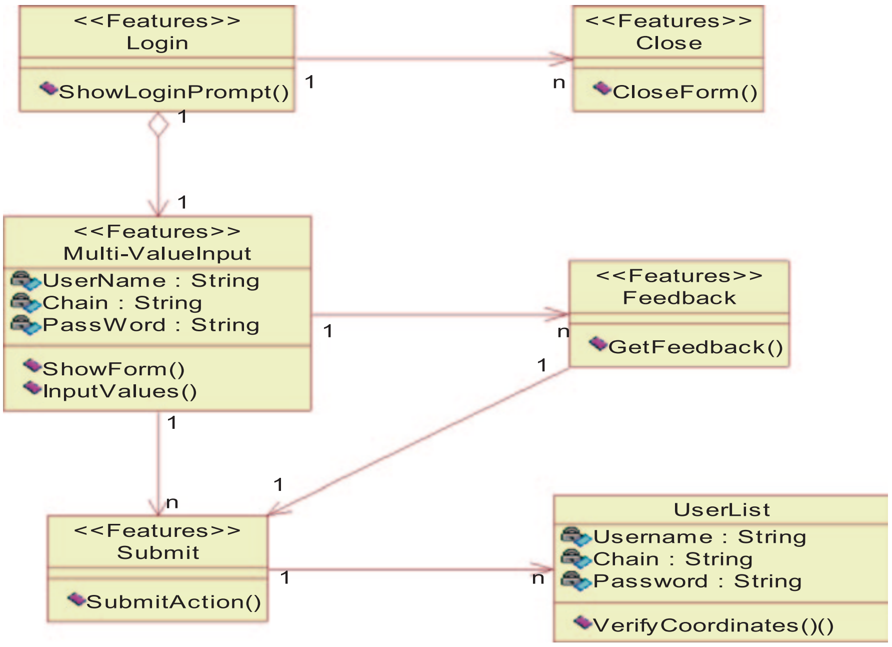


Fig. 9.3 UML class diagram of the platform-specific model (PSM) domain model for a laptop platform

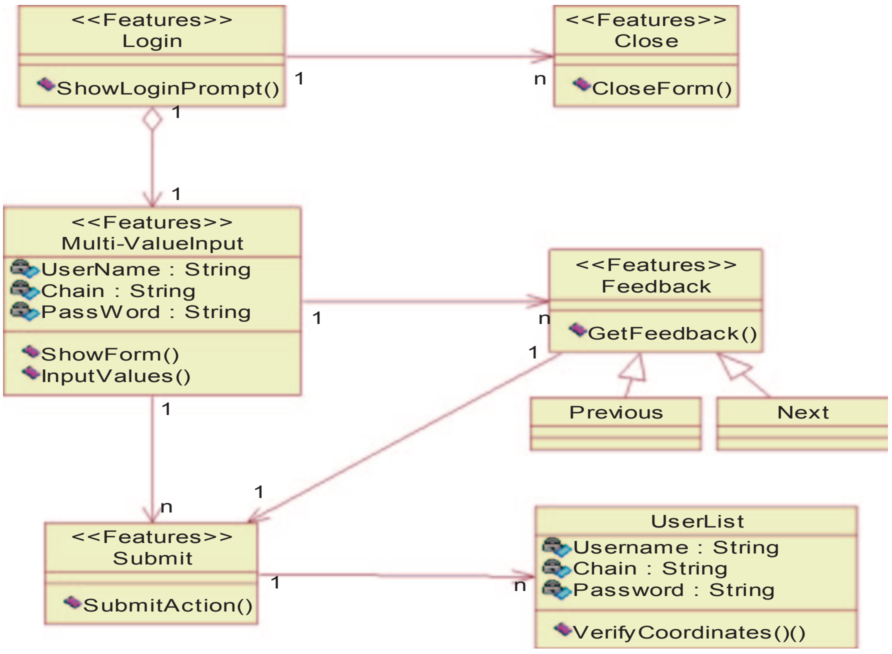


Fig. 9.4 UML class diagram of the PSM domain model for personal digital assistant (PDA) platform

Table 9.2 Example of pattern mapping of the Domain model for laptop and PDA platforms

Patterns of Microsoft platform	Type of mapping	Replacement patterns for laptop platform	Replacement patterns for PDA platform
P1. Login	Identical, scalable, or fundamental	P1. Login	P1.s Login (small interface)
P2. Multivalue input	Identical	P2. Multivalue input	P2. Multivalue input
P3. Submit	Scalable or fundamental	P3. Submit	P3.s Submit (smaller button)
P4. Feedback	Scalable or fundamental	P4. Feedback	P4. Feedback (less items per page) P4.1. Next P4.2. Previous
P5. Close	Identical	P5. Close	P5. Close

PDA personal digital assistant

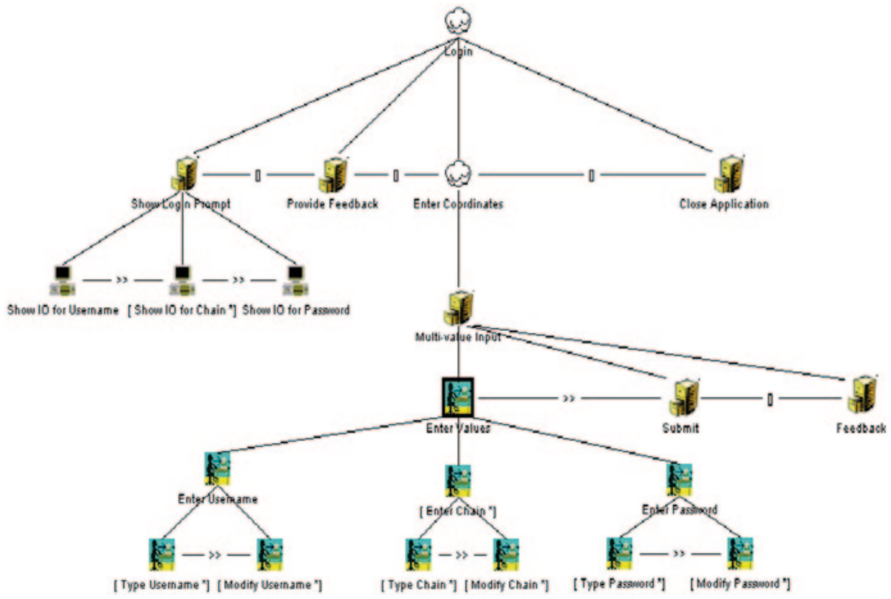


Fig. 9.5 The *Login* pattern on the laptop platform

Figure 9.5 and Fig. 9.6 represent a structure of the *Login* pattern, which enables the user to identify himself or herself in order to access secure or protected data and/or to perform authorized operations.

Figures 9.7 and 9.8 represent an implementation of the *Login* pattern.

Figure 9.7 is an example of implementation of the *Login* Pattern.

Figure 9.8 is an example of implementation of the *Login* pattern.

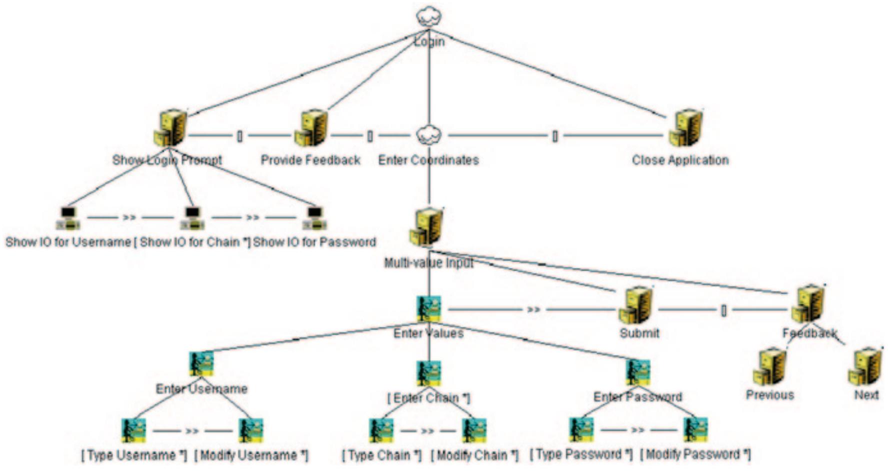

Login
Show Login Prompt
Provide Feedback
Enter Coordinates
Close Application
Show IO for Username [Show IO for Chain *] Show IO for Password
Multi-value Input
Enter Values
Submit
Feedback
Enter Username
[Enter Chain *]
Enter Password
Previous
Next
[Type Username *] [Modify Username *]
[Type Chain *] [Modify Chain *]
[Type Password *] [Modify Password *]

Fig. 9.6 The *Login* pattern on the PDA platform

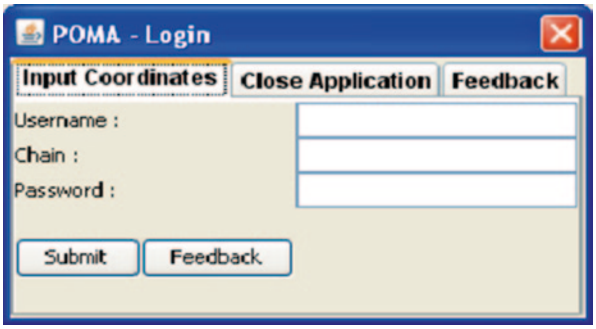


Fig. 9.7 Login view of the interactive system on the laptop platform

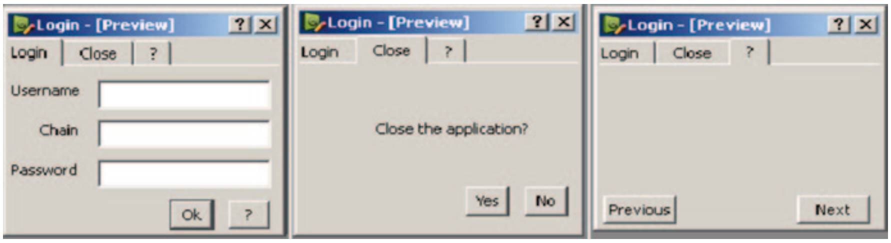


Fig. 9.8 Login view of the interactive system on the PDA platform

The following is an example of the XML source code of the domain model for the laptop platform of an “Environmental Management Interactive System”:

```
<?xml version=“1.0“?>
<xsd:schema xmlns:xsd=“http://www.w3.org/2007/XMLSchema“>
<xsd:group name=”Login”>
      <xsd:sequence>
          <xsd:element name=”ShowLoginPrompt”/>
            <xsd:sequence>
                <xsd:element name=”EnterCoordinates”>
                    <xsd:complexType>
                          <xsd:all>
                                  <xsd:element name=”Multi-ValueInput”>
                                          <xsd:complexType>
                                                  <xsd:all>
                                                          <xsd:element name=”ShowForm”/>
                                                          <xsd:element name=”EnterValues”>
                                                      xsd:complexType>
                                                           <xsd:attribute name=”Username”/>
                                                           <xsd:attribute name=”Password”/>
                                                     </xsd:complexType>
                                  </xsd:element>
                                  <xsd:element name=”Submit”/>
                                  <xsd:element name=”Feedback”/>
                            </xsd:all>
                          </xsd:complexType>
                       </xsd:element>
                    <xsd:element name=”CloseApplication”/>
                 <xsd:element name=”FeedbackForLoginForm”/>
              </xsd:all>
           </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
   </xsd:sequence>
</xsd:group>
```

9.4.3 *Defining the Task Model*

After establishing the domain model for the system in this case study, the task model can be interactively defined. Figure 9.9 depicts the task model structure of the “Environmental Management Interactive System.” Only high-level tasks and their relationships are portrayed. The overall structure and behavior of the interactive system is given. The structure provided is relatively unique for an environmental management interactive system; the concrete “realization” of high-level tasks has been omitted.

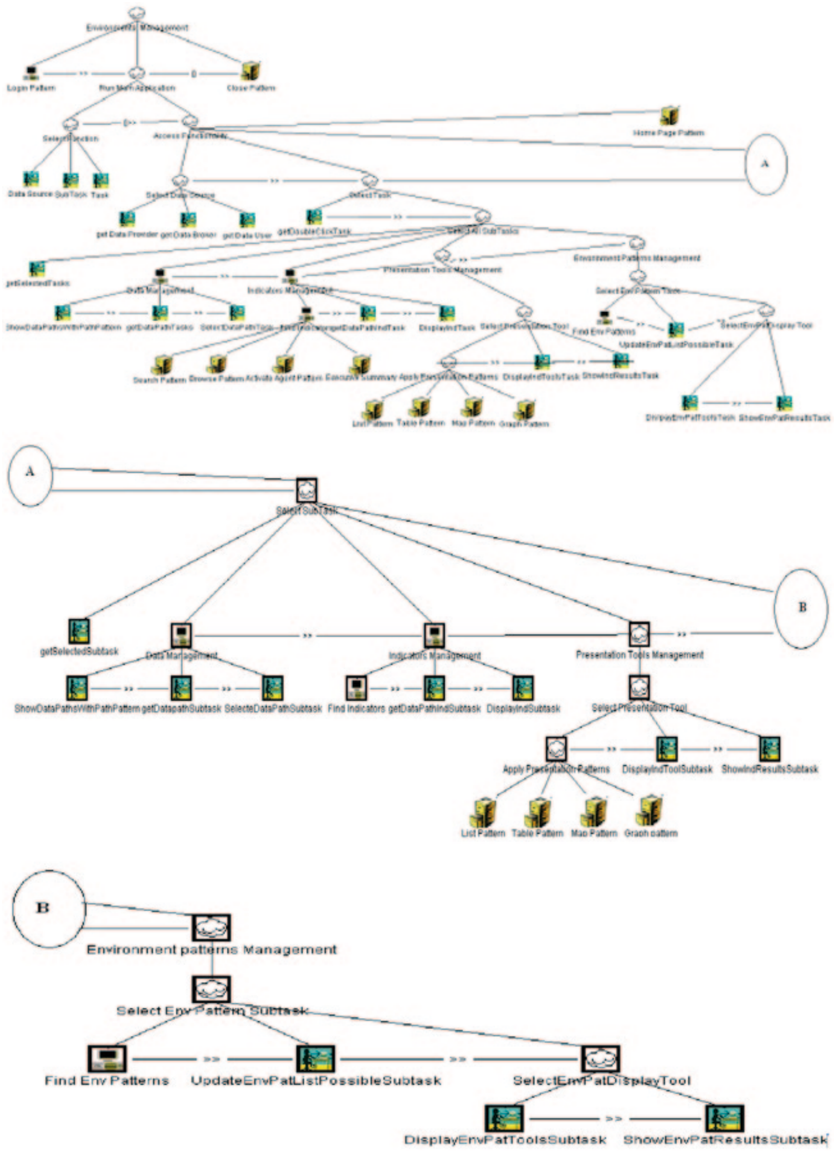


Fig. 9.9 Task model of the environmental management interactive system

A large part of many interactive systems can be developed from a fixed set of reusable components. In the case of the task model, the more those high-level tasks are decomposed, the easier it is to use the reusable task structures that have been gained or captured from other projects or systems. In this case study, these reusable task structures are documented in the form of patterns. This approach ensures an even greater degree of reuse, since each pattern can be adapted to the current use context.

The main characteristics of the environmental management system, modeled by the task structure in Fig. 9.9 can be outlined as follows:

The interactive system's main functionality is accessed by logging into the system (the login task enables the management task). The key features are "adding a guest," which is accomplished by entering the guest's personal information and by "selecting an environment task or subtask" for a specific guest. The two tasks can be performed in any order. The selection process consists of four consecutively performed tasks (related through "enabling with information exchange" operators):

1. Selecting data source to use
2. Selecting task or subtask

   a. Data management
   b. Indicator management
   c. Presentation tool management
   d. Environmental pattern management

Acting in the horizontal direction of the POMA architecture (Fig. 8.2), this model is composed of two types of submodels, which are: [POMA.PIM]-independent task submodels, and [POMA.PSM]-specific Task submodels.

[POMA.PIM]-independent Task submodels (Fig. 9.10) are obtained by composing patterns and applying the composition rules described in the Chap. 8 in Sect. 8.5.2.

[POMA.PSM]-specific task submodels Figs. 9.11 and 9.12 are obtained by mapping composed patterns and applying the mapping rules (Table 9.3). This latter model would be used to generate the system's source code by taking into account the code generation rules.

Figure 9.9 presents a structure of the Task model of the "Environmental Management Interactive System." As shown in Fig. 9.9, the *Login, MultiValue Input Form,* and *Find* patterns can be used in order to complete the task model at lower levels.

Figure 9.10 represents a Unified Modeling Language (UML) class diagram of the platform-independent model (PIM) task model, which is composed of several patterns by applying, manually by the designers, the composition rules described in Sect. 4.1.2. This model underwent mapping by applying the mapping rules (Table 9.3) to obtain another model, which is called a platform-specific model (PSM) task model (Fig. 9.11) for a laptop platform and Fig. 9.12 for a PDA platform.

Table 9.3 shows the mapping rules of the task model patterns for a laptop and PDA platforms.

After the mapping, the PSM Task model is obtained for a laptop platform—Fig. 9.11.

After the mapping, the PSM task model is obtained for a PDA platform—Fig. 9.12.

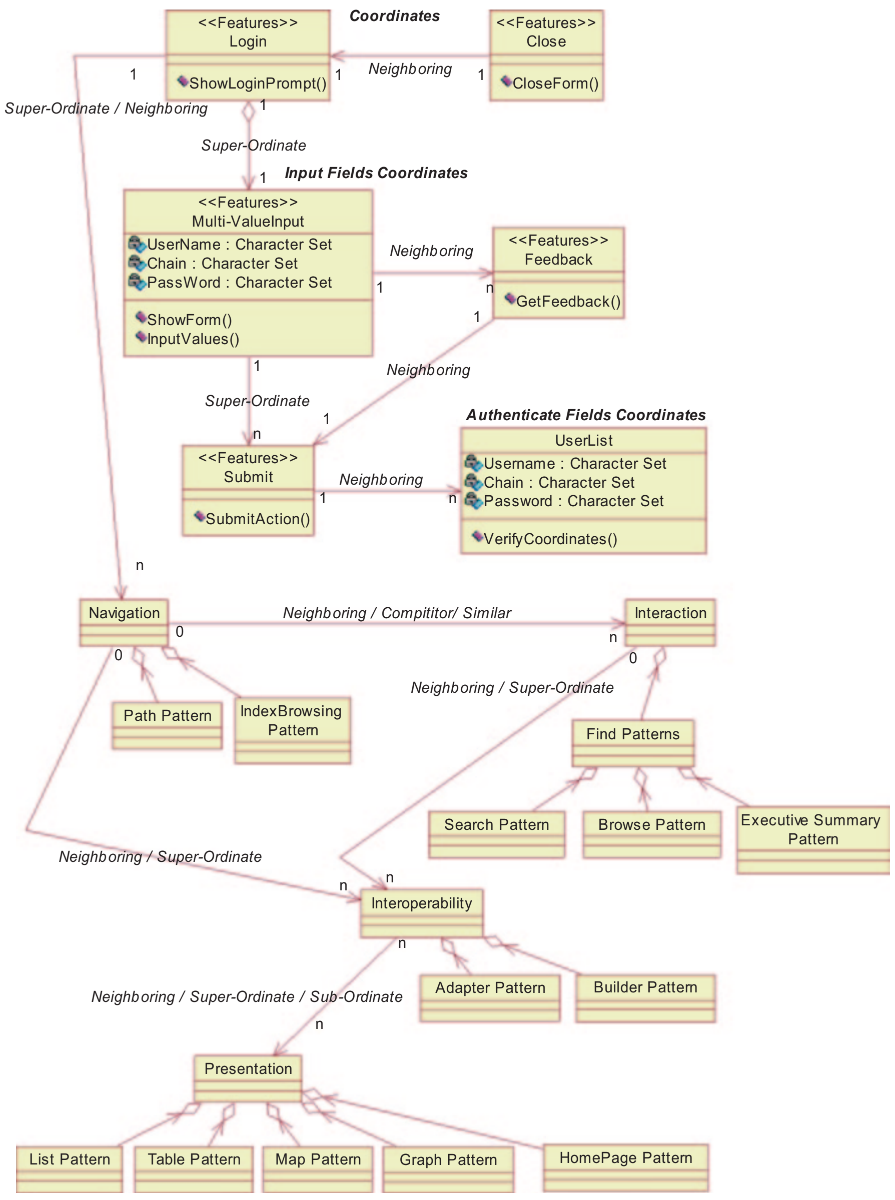


Fig. 9.10 UML class diagram of the PIM Task model

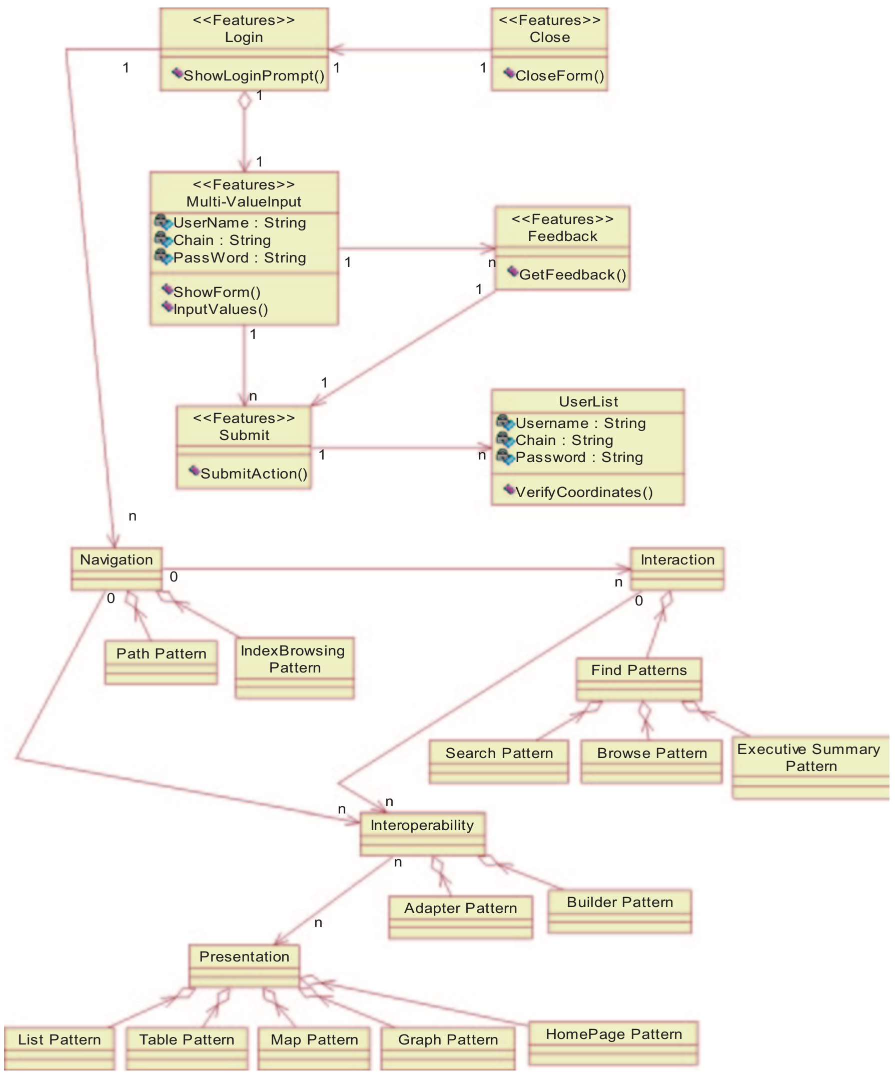


Fig. 9.11 UML class diagram of the PSM Task model mapped for a laptop platform

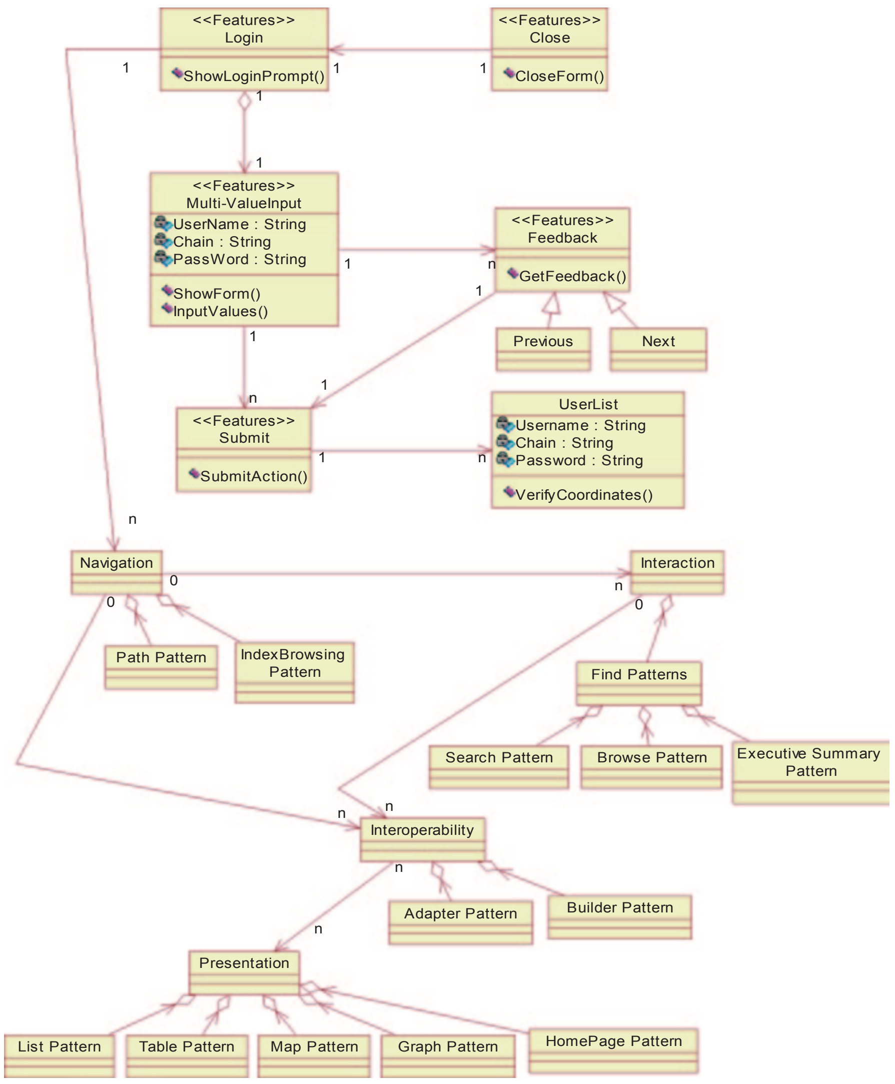


Fig. 9.12 UML class diagram of the PSM Task model mapped for a PDA platform

Table 9.3 Example of pattern mapping of task model for laptop and personal digital assistant (PDA) platforms

Patterns of Microsoft platform	Type of mapping	Replacement patterns for Laptop platform	Replacement patterns for PDA platform
P1. Login	Identical	P1. Login	P1. Login
P2. Multivalue input	Identical, scalable, fundamental	P2. Multivalue input	P2. Multivalue input
P3. Submit	Scalable or fundamental	P3. Submit	P3.s Submit (smaller button)
P4. Feedback	Identical, fundamental	P4. Feedback	P4. Feedback P4.1. Previous P4.2. Next
P5. Close	Identical	P5. Close	P5. Close
P6. Find (search, browse, executive summary)	Identical, scalable	P6. Find (search, browse, executive summary)	P6. Find (search, browse, executive summary)
P7. Path (Breadcrumb)	Identical, scalable (Laptop) Scalable or funda-mental (PDA)	P7. Path (Breadcrumb)	P7.1s Shorter bread-crumb trial P7.2 Drop-down “His-tory” menu
P8. Index browsing	Identical	P8. Index browsing	P8. Drop-down menu
P9. Adapter	Identical	P9. Adapter	P9. Adapter
P10. Builder	Identical	P10. Builder	P10. Builder
P11. List	Identical	P11. List	P11. List
P12. Table	Identical	P12. Table	P12. Table
P13. Map	Identical	P13. Map	P13. Map
P14. Graph	Identical	P14. Graph	P14. Graph
P15. Home Page	Identical	P15. Home page	P15. Home page

PDA personal digital assistant

The following is an example of the XML source code portion of the task model for a laptop platform of the “Environmental Management Interactive System”:

```
<?xml version= ‘1.0’?>
<!DOCTYPE TaskModel PUBLIC “http://giove.cnuce.cnr.it/CTTDTD.dtd”
“..\..\..\..\Teresa\CTT\CTTDTD.dtd”>
<TaskModel
NameTaskModelID=”C:\Momo\PhD\These\Prototype\Environmental_Management_CTT.xm
l”>
<Task Identifier=”Environmental Management” Category=”abstraction” Iterative=”false”
Optional=”false” PartOfCooperation=”false” Frequency=”null”>
 <Name> null </Name>
 <Type> null </Type>
 <Description> null </Description>
 <Precondition> null </Precondition>
 <TimePerformance>
 <Max> null </Max>
 <Min> null </Min>
 <Average> null </Average>
 </TimePerformance>
 <Object name=”null” class=”null” type=”null” access_mode=”null” cardinality=”null”>
 <Platform> null </Platform>
 <InputAction Description=”null” From=”null”/>
 <OutputAction Description=”null” To=”null”/>
 </Object>
 <SubTask>
<Task Identifier=”Login Pattern” Category=”application” Iterative=”false” Optional=”false”
PartOfCooperation=”false” Frequency=”null”>
 <Name> null </Name>
 <Type> null </Type>
 <Description> null </Description>
 <Precondition> null </Precondition>
 <TemporalOperator name=”SequentialEnabling”/>
 <TimePerformance>
 <Max> null </Max>
 <Min> null </Min>
 <Average> null </Average>
 </TimePerformance>
 <Parent name=”Environmental Management”/>
 <SiblingRight name=”Run Main Application”/>
 <Object name=”null” class=”null” type=”null” access_mode=”null” cardinality=”null”>
 <Platform> null </Platform>
 <InputAction Description=”null” From=”null”/>
 <OutputAction Description=”null” To=”null”/>
 </Object>
 </Task>
  …
 <SubTask>
<Task Identifier=”ShowDataPathsWithPathPattern” Category=”interaction” Iterative=”false”
Optional=”false” PartOfCooperation=”false” Frequency=”null”>
<Name> null </Name>
<Type> null </Type>
<Description> null </Description>
<Precondition> null </Precondition>
<TemporalOperator name=”SequentialEnabling”/>
<TimePerformance>
<Max> null </Max>
<Min> null </Min>
<Average> null </Average>
</TimePerformance>
<Parent name=”Data Management”/>
<SiblingRight name=”getDataPathTasks”/>
<Object name=”null” class=”null” type=”null” access_mode=”null” cardinality=”null”>
<Platform> null </Platform>
<InputAction Description=”null” From=”null”/>
<OutputAction Description=”null” To=”null”/>
</Object>
</Task>

  …
</SubTask>
</Task>
</TaskModel>
```

9.4.4 Defining the Dialog Model

Acting in the horizontal line of the POMA architecture (Fig. 8.2), the dialog model is composed of two types of submodels, [POMA.PIM]-independent dialog submodel, and [POMA.PSM]-specific dialog submodel.

[POMA.PIM]-independent dialog submodel (Fig. 9.13) is obtained by composing patterns and applying, manually by the designers, the composition rules described in Sect. 8.5.2.

The Wizard dialog pattern emerges as the best choice for implementation. It suggests a dialog structure where a set of dialog views is arranged sequentially and the "last" task of each dialog view initiates the transition to the subsequent dialog view. Figure 9.14 depicts the Wizard dialog pattern's suggested graph structure.

[POMA.PSM]-specific dialog submodel (Fig. 9.15) is obtained by mapping composed patterns and applying the mapping rules (Table 9.4). This [POMA.PSM]

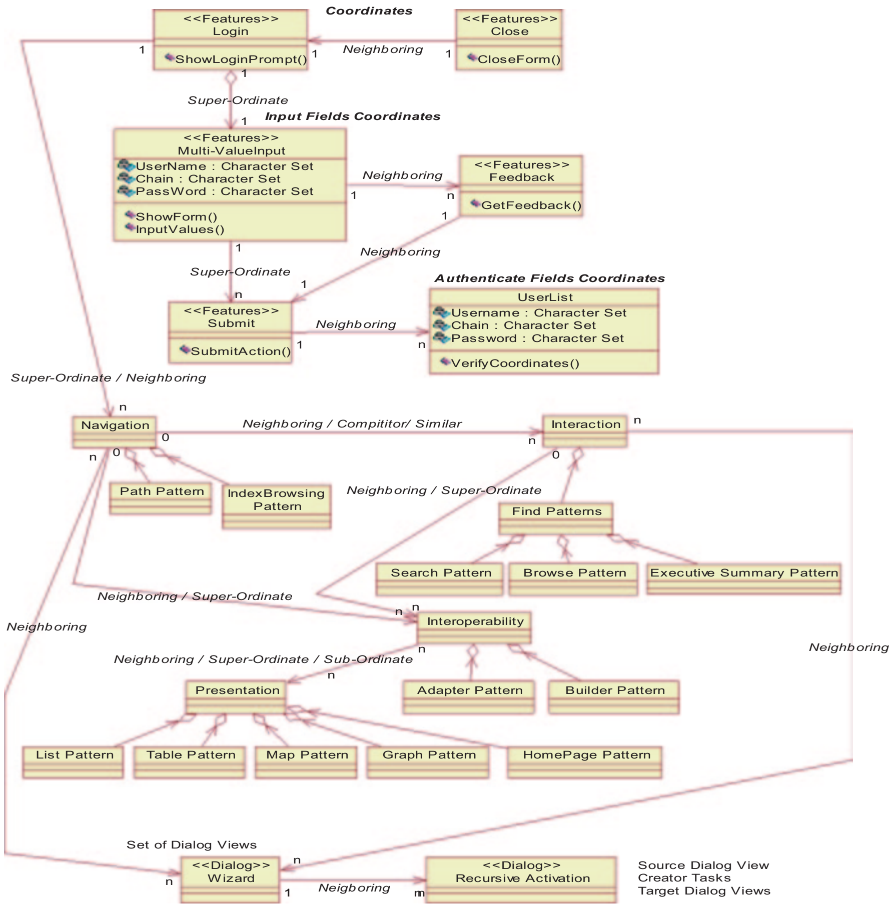


Fig. 9.13 UML class diagram of a PIM dialog model

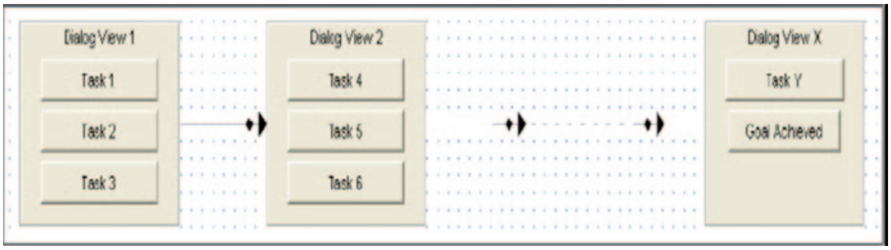


Fig. 9.14 Graph structure suggested by the Wizard pattern

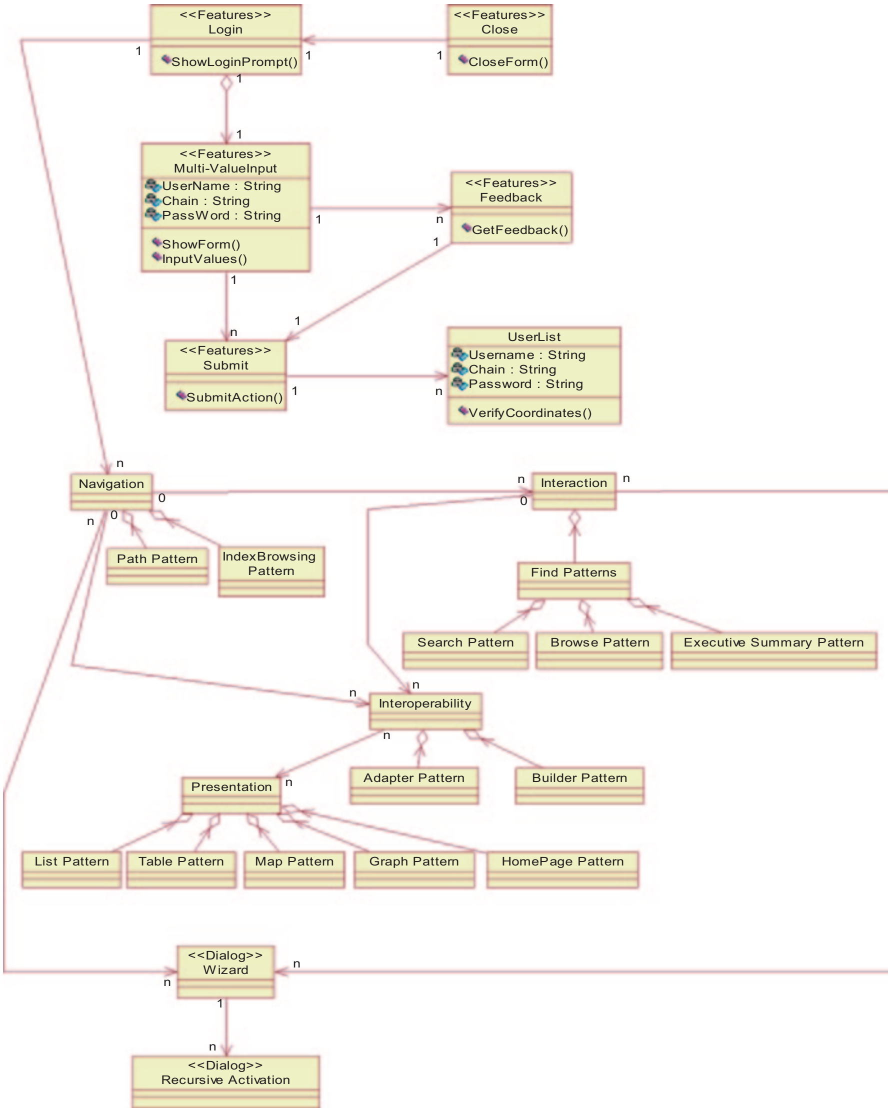


Fig. 9.15 UML class diagram of the PSM dialog model for a laptop platform

Table 9.4 Example of pattern mapping of dialog model for laptop and PDA platforms

Patterns of Microsoft platform	Type of mapping	Replacement patterns for laptop platform	Replacement patterns for PDA platform
P1. Login	Identical	P1. Login	P1. Login
P2. Multivalue input	Identical, scalable, fundamental	P2. Multivalue input	P2. Multivalue input
P3. Submit	Scalable or fundamental	P3. Submit	P3.s Submit (smaller button)
P4. Feedback	Identical, fundamental	P4. Feedback	P4. Feedback P4.1. Previous P4.2. Next
P5. Close	Identical	P5. Close	P5. Close
P6. Find (search, browse, executive summary)	Identical, scalable	P6. Find (search, browse, executive summary)	P6. Find (search, browse, executive summary)
P7. Path (Breadcrumb)	Identical, scalable (Laptop) Scalable or fundamental (PDA)	P7. Path (Breadcrumb)	P7.1s Shorter bread crumb trial P7.2 Drop-down "History" menu
P8. Index browsing	Identical	P8. Index browsing	P8. Drop-down menu
P9. Adapter	Identical	P9. Adapter	P9. Adapter
P10. Builder	Identical	P10. Builder	P10. Builder
P11. List	Identical	P11. List	P11. List
P12. Table	Identical	P12. Table	P12. Table
P13. Map	Identical	P13. Map	P13. Map
P14. Graph	Identical	P14. Graph	P14. Graph
P15. Home page	Identical	P15. Home page	P15. Home page
P16. Wizard	Identical	Wizard	P16. Wizard
P17. Recursive activation	Identical	Recursive activation	P17. Recursive activation

model is used to generate the interactive system's source code by taking into account the code generation rules.

Figure 9.13 represents a UML class diagram of the PIM dialog model, which is composed of several patterns. This model underwent mapping by applying the mapping rules (Table 9.4) to obtain another model, which is called PSM dialog model (Fig. 9.15 for a laptop platform and Fig. 9.16 for a PDA platform).

However, the sequential structure of the subtask process must be slightly modified in order to enable the user to view the details of multiple subtasks at the same time. Specifically, this behavior should be modeled using the recursive activation dialog pattern. This pattern is used when the user wishes to activate and manipulate several instances of a dialog view.

Table 9.4 shows the mapping rules of the dialog model patterns for laptop and PDA platforms.

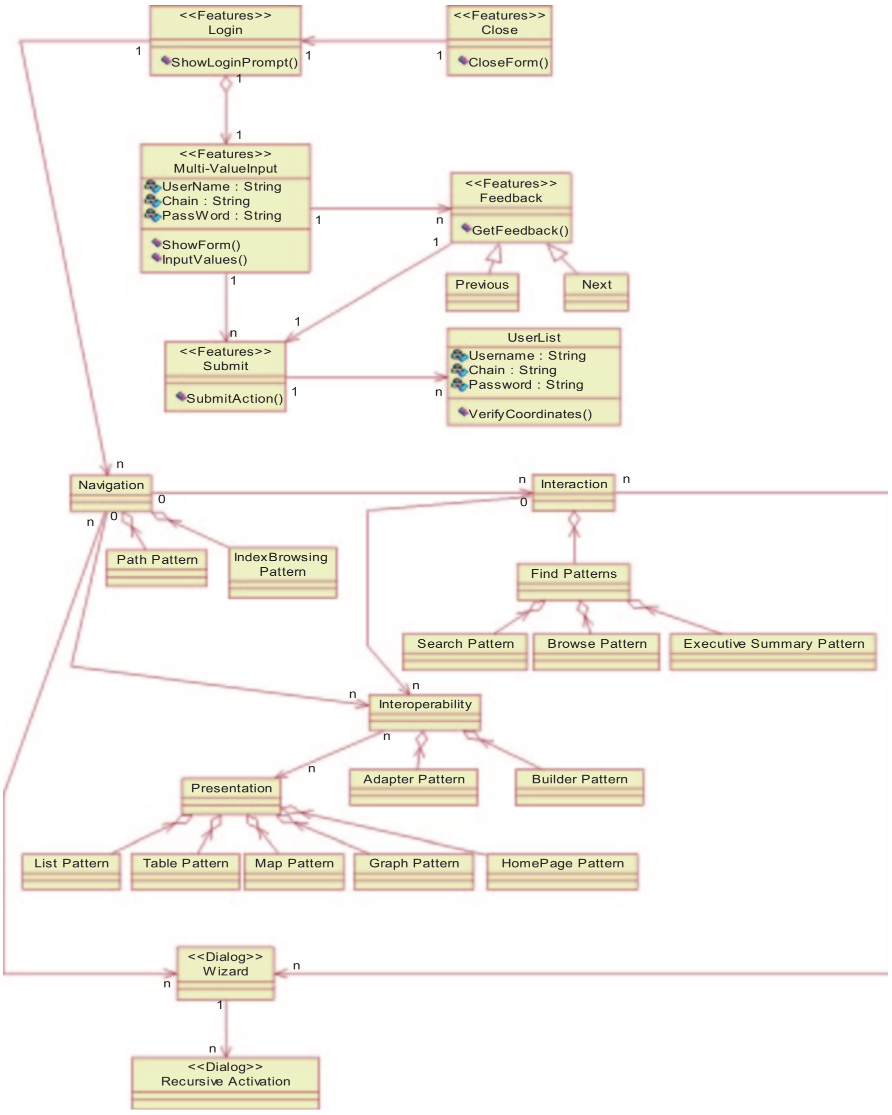


Fig. 9.16 UML class diagram of the PSM dialog model for a PDA platform

After the mapping, the PSM dialog model is obtained for a laptop platform—Fig. 9.15.

After the mapping, the PSM dialog model is obtained for a PDA platform—Fig. 9.16.

Figure 9.17 depicts the various dialog view interactions of the “Environmental Management Interactive System’s” suggested dialog graph structure for laptop and PDA platforms.

9.4.5 Defining the Presentation and Layout Models

In order to define the presentation model for this case study, the grouped tasks of each dialog view are associated with a set of interaction elements including forms, buttons, and lists. Style attributes, such as size, font, and color, remain unset and will be defined by the layout model.

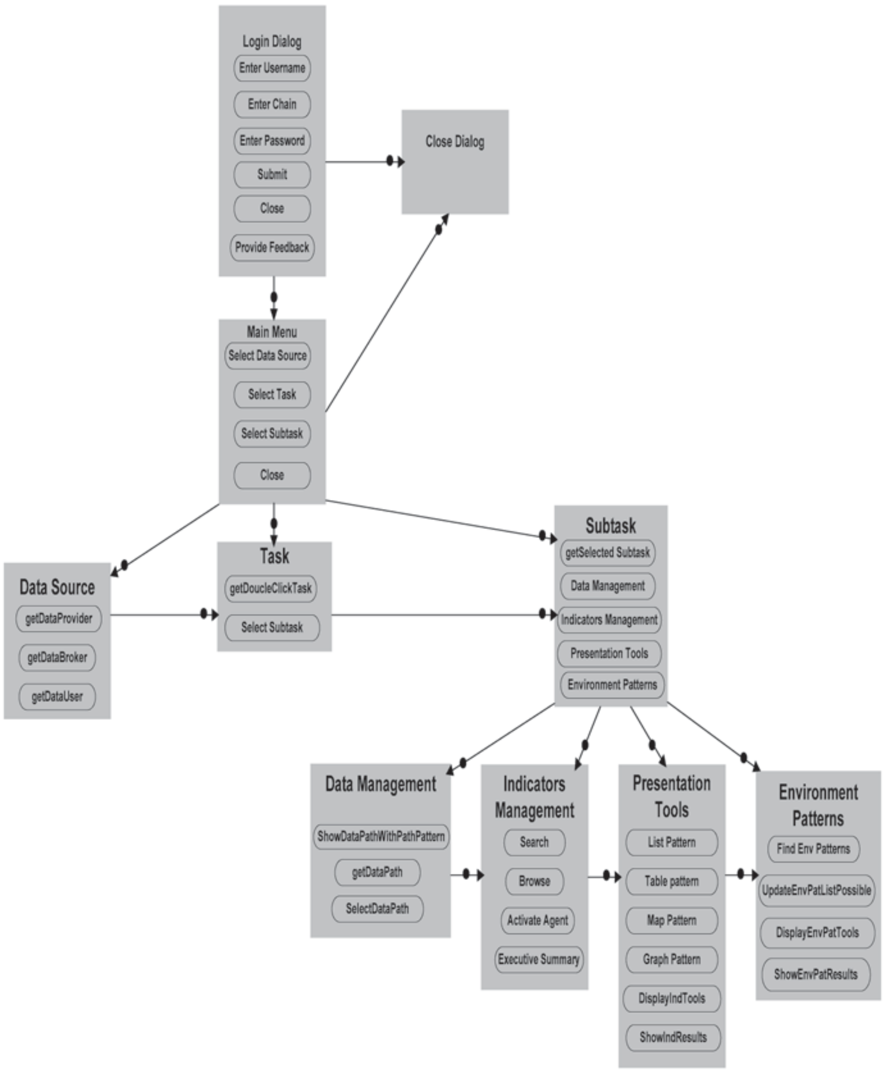


Fig. 9.17 Dialog graph of the environmental management interactive system for laptop and PDA platforms

A significant part of the user tasks of the system revolves around providing structured textual information. This information can usually be split into logically related data chunks.

At this point, the form presentation pattern, which handles this precise issue, can be applied using a form for each related data chunk, populated with the elements needed to enter the data. Moreover, the pattern refers to the unambiguous format pattern which can be employed. The purpose of this pattern is to prevent the user from entering syntactically incorrect data, and is achieved in the following way: Depending on the domain of the object to be entered, the instance of the pattern provides the most suitable input interaction elements by drawing on information from the business object model.

Acting in the horizontal line of the POMA architecture (Fig. 8.2) in Chap. 8, the model is composed of two types of submodel, [POMA.PIM]-independent presentation submodel, and [POMA.PSM]-specific presentation submodel.

IM]-independent presentation submodel, and [POMA.PSM]-specific presentation submodel.

The [POMA.PIM]-independent presentation submodel (Fig. 9.18) is obtained by composing patterns and applying the composition rules.

The [POMA.PSM]-specific presentation submodel (Figs. 9.19 and 9.20) is obtained by mapping composed patterns and applying the mapping rules (Table 9.). This model is used to generate the system's source code by taking into account the code generation rules.

Figure 9.18 represents a UML diagram of the PIM presentation model, which is composed of several patterns. This model underwent mapping by applying the mapping rules (Table 9.5) to obtain another model, which is called the PSM presentation model (Fig. 9.19 for a laptop platform and Fig. 9.20 for a PDA platform).

Table 9.5 shows the mapping rules of the patterns of the presentation model for laptop and PDA platforms.

After the mapping, the PSM presentation model is obtained for a laptop platform—Fig. 9.19.

After the mapping, the PSM presentation model is obtained for a PDA platform—Fig. 9.20.

In the layout model, the style attributes, that have not yet been defined, are set in keeping with the standards set for the "Environmental Management Interactive System." According to the house style pattern (which is applicable here), colors, fonts, and layouts should be chosen to give the user the impression that all the system windows share a consistent presentation and appear to belong together. Cascading style sheets have been used to control the visual appearance of the interface. In addition, to assist the user when working with the system, meaningful labels have been provided. The labeling layout pattern suggests the adding of labels for each interaction element. Using the grid format, the labels are aligned to the left of the interaction element.

The Layout model determines how the loosely connected XUL (Appendix III) fragments are aggregated according to an overall floor plan. In this case study, the task is fairly straightforward since the UI is not nested and consists of a single

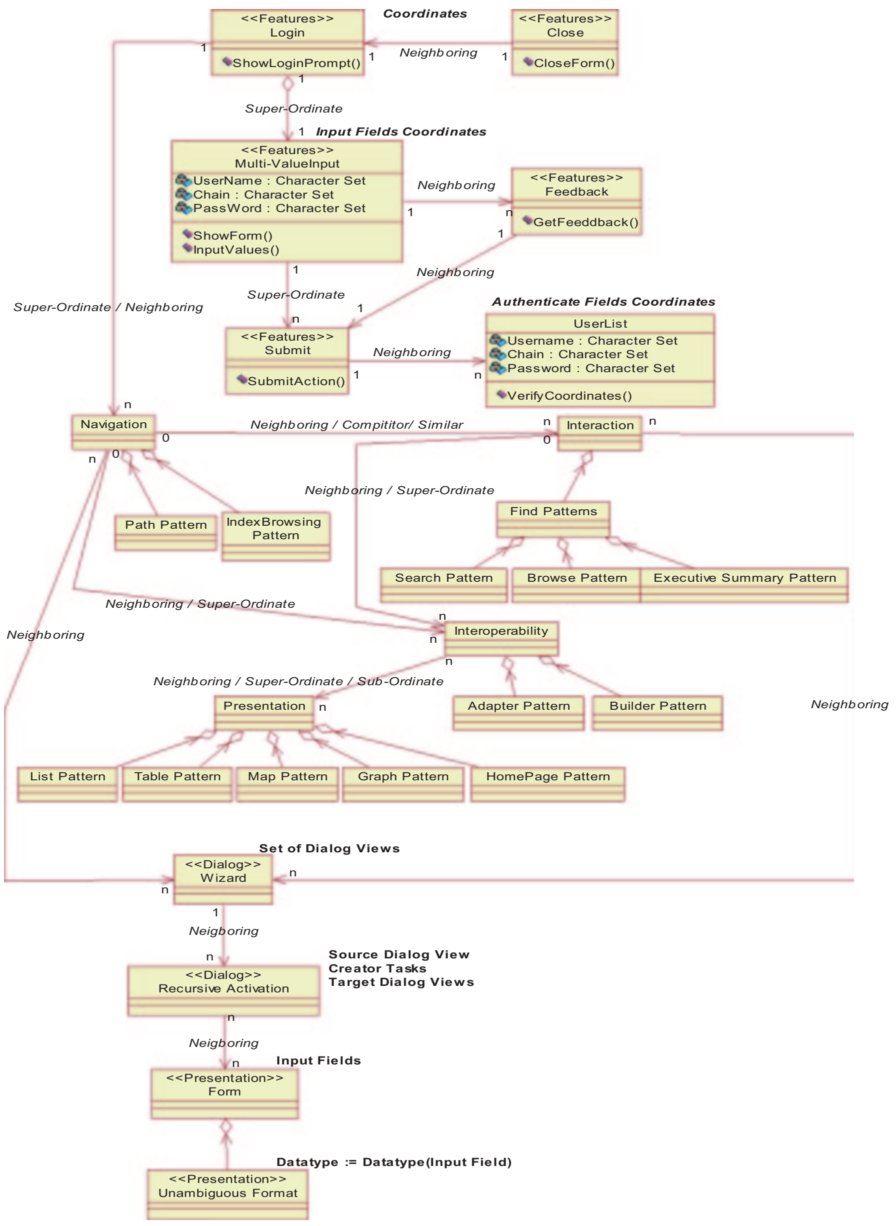


Fig. 9.18 UML class diagram of a PIM presentation model

container. After establishing the layout model, the aggregated XUL code can be rendered, along with the corresponding XUL skins, as the final UI. All interfaces are shown in the final UI rendered on the Windows XP platform.

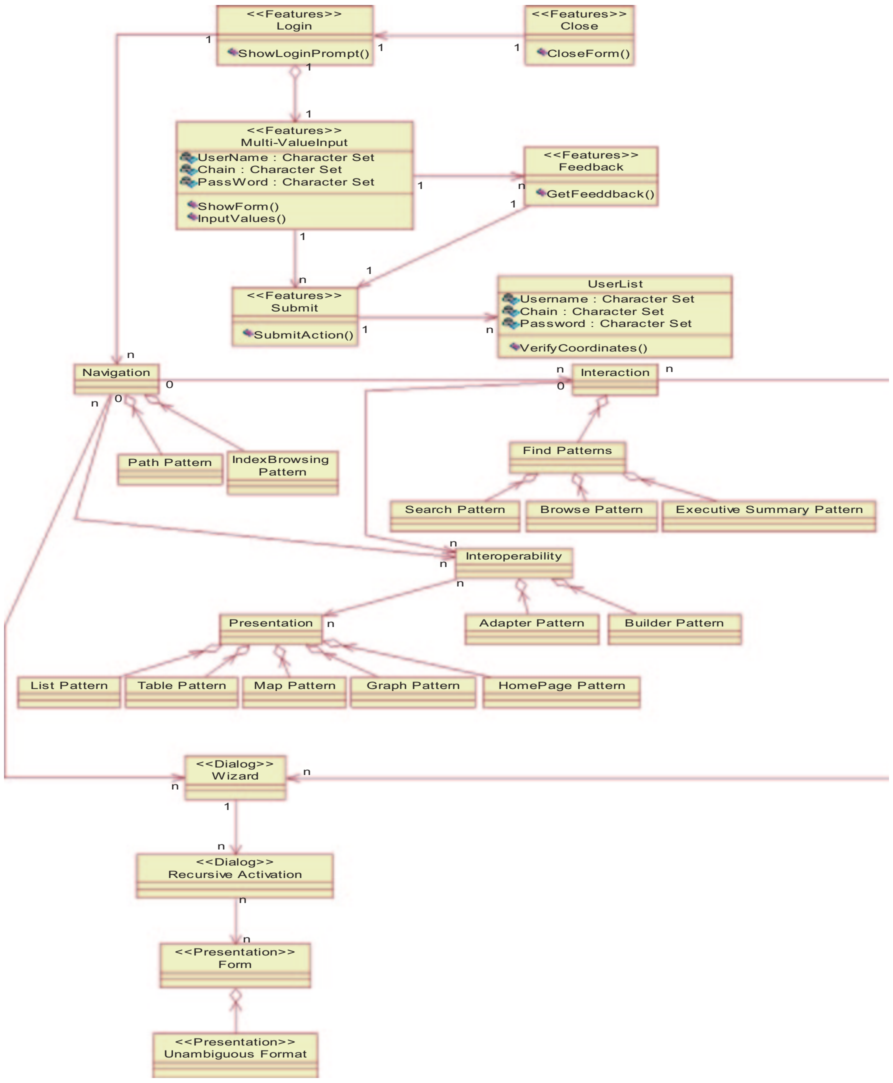


Fig. 9.19 UML class diagram of the PSM presentation model for a laptop platform

Acting in the horizontal direction of the POMA architecture (Fig. 8.2), this model is composed of two types of submodels, [POMA.PIM]-independent layout submodel, and [POMA.PSM]-specific layout submodel.

[POMA.PIM]-independent layout submodel (Fig. 9.21) is obtained by composing patterns and applying the composition rules.

[POMA.PSM]-specific layout submodel (Figs. 9.22 and 9.23) is obtained by mapping composed patterns and applying the mapping rules (Table 9.6). This

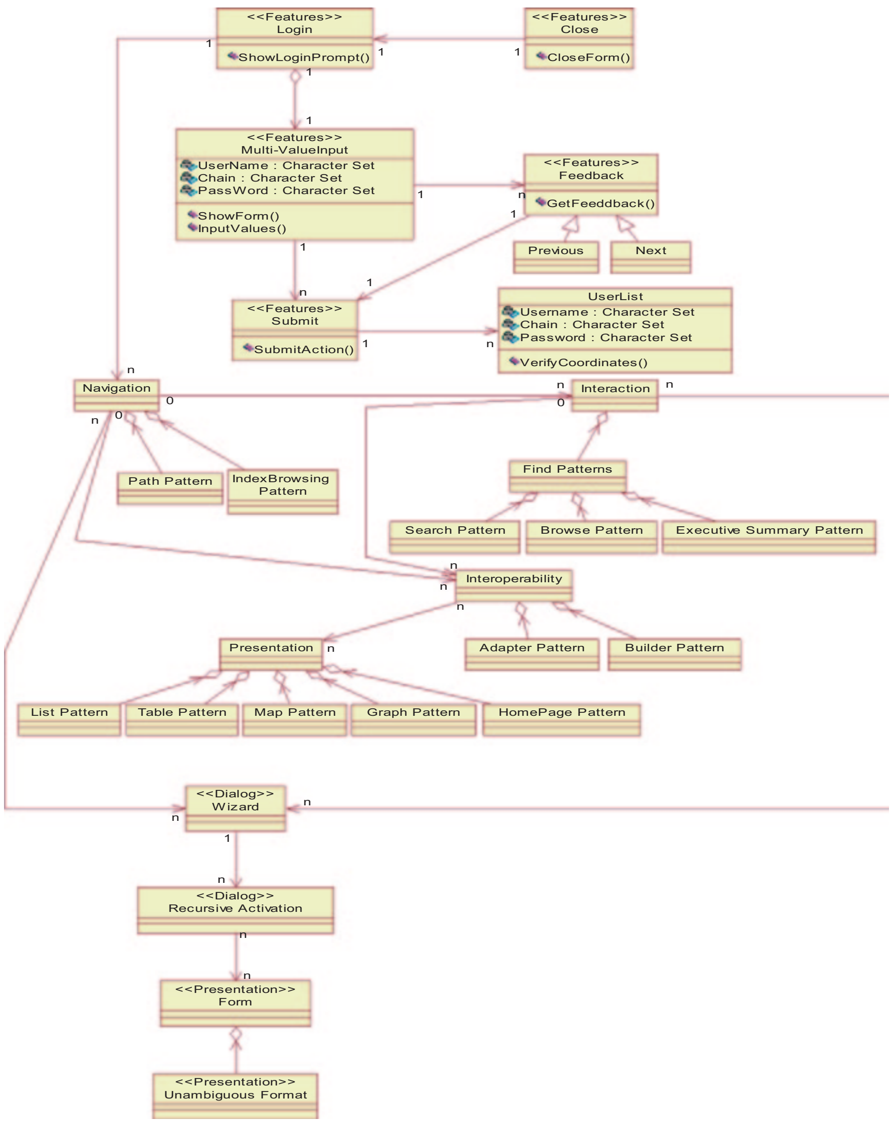


Fig. 9.20 UML class diagram of the PSM presentation model for a PDA platform

[POMA.PSM] model is used to generate the system's source code by taking into account the code generation rules (not included in this research).

Figure 9.21 represents a UML class diagram of the PIM layout model which is composed of several patterns. This model underwent mapping by applying the mapping rules (Table 9.6) to obtain another model, which is called the PSM layout model (Fig. 9.22 for a laptop platform and Fig. 9.23 for a PDA platform).

Table 9.5 Example of pattern mapping of the presentation model for laptop and PDA platforms

Patterns of Microsoft platform	Type of mapping	Replacement patterns for laptop platform	Replacement patterns for PDA platform
P1. Login	Identical	P1. Login	P1. Login
P2. Multivalue input	Identical, scalable, fundamental	P2. Multivalue input	P2. Multivalue input
P3. Submit	Scalable or fundamental	P3. Submit	P3.s. Submit (smaller button)
P4. Feedback	Identical, fundamental	P4. Feedback	P4. Feedback P4.1. Previous P4.2. Next
P5. Close	Identical	P5. Close	P5. Close
P6. Find (search, browse, executive summary)	Identical, scalable	P6. Find (search, browse, executive summary)	P6. Find (search, browse, executive summary)
P7. Path (Breadcrumb)	-Identical, scalable (Laptop) -Scalable or fundamental (PDA)	P7. Path (Breadcrumb)	- P7.1s. Shorter bread crumb trial - P7.2. Drop-down "History" menu
P8. Index browsing	Identical	P8. Index browsing	P8. Drop-down menu
P9. Adapter	Identical	P9. Adapter	P9. Adapter
P10. Builder	Identical	P10. Builder	P10. Builder
P11. List	Identical	P11. List	P11. List
P12. Table	Identical	P12. Table	P12. Table
P13. Map	Identical	P13. Map	P13. Map
P14. Graph	Identical	P14. Graph	P14. Graph
P15. Home page	Identical	P15. Home page	P15. Home page
P16. Wizard	Identical	P16. Wizard	P16. Wizard
P17. Recursive activation	Identical	P17. Recursive activation	P17. Recursive activation
P18. Unambiguous format	Identical	P18. Unambiguous format	P18. Unambiguous format
Form	Identical	Form	Form

Table 9.6 shows the mapping rules for the patterns of the layout model for laptop and PDA platforms.

After the mapping, the PSM layout model is obtained for a laptop platform—Fig. 9.22.

After the mapping, the PSM Layout model is obtained for a PDA platform—Fig. 9.23.

Figure 9.24 is the final layout of the "Environmental Management Interactive System."

The results of this experimentation of POMA architecture are as follows:

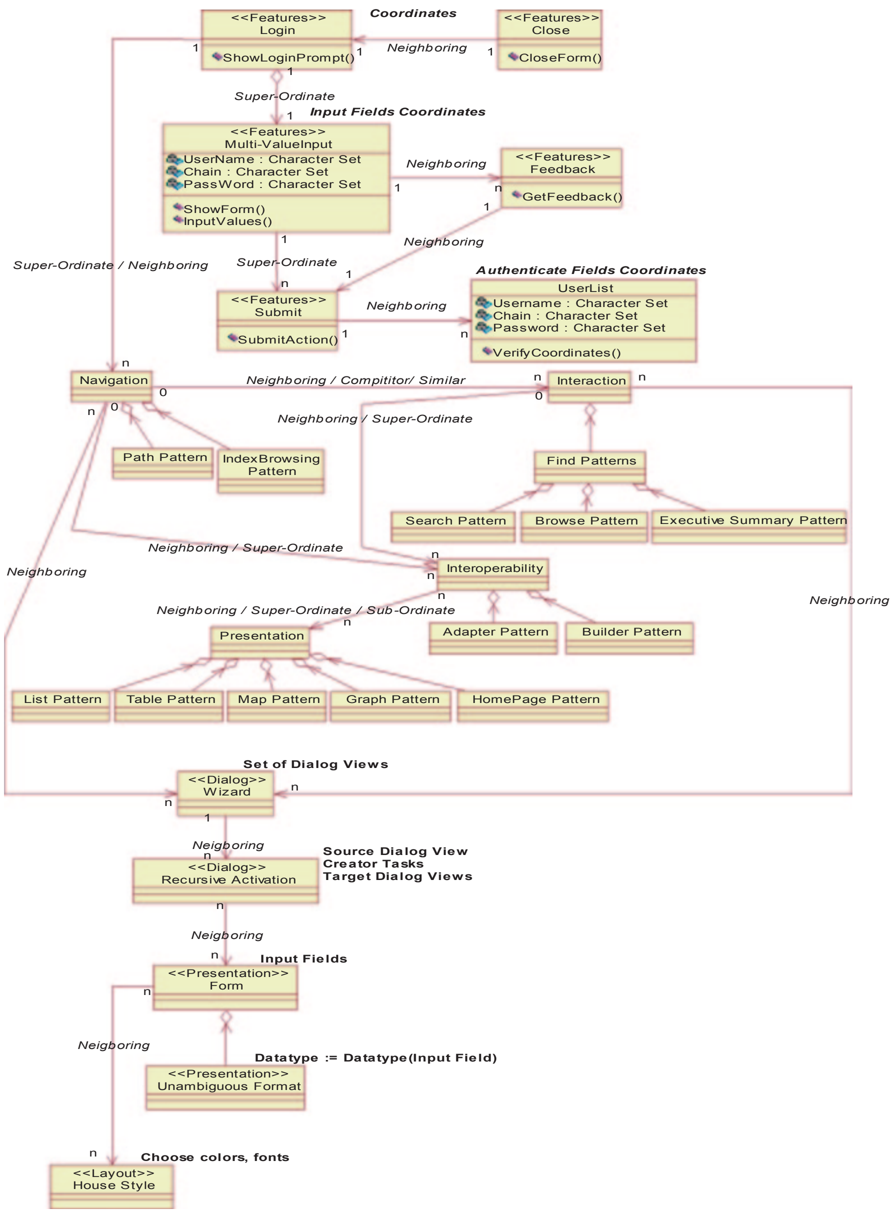


Fig. 9.21 UML class diagram of a PIM layout model

- POMA integrates easily patterns and models together to design interactive systems for different platforms;
- POMA uses easily the pattern composition, pattern mapping and model transformation rules to implement interactive systems for different platforms.

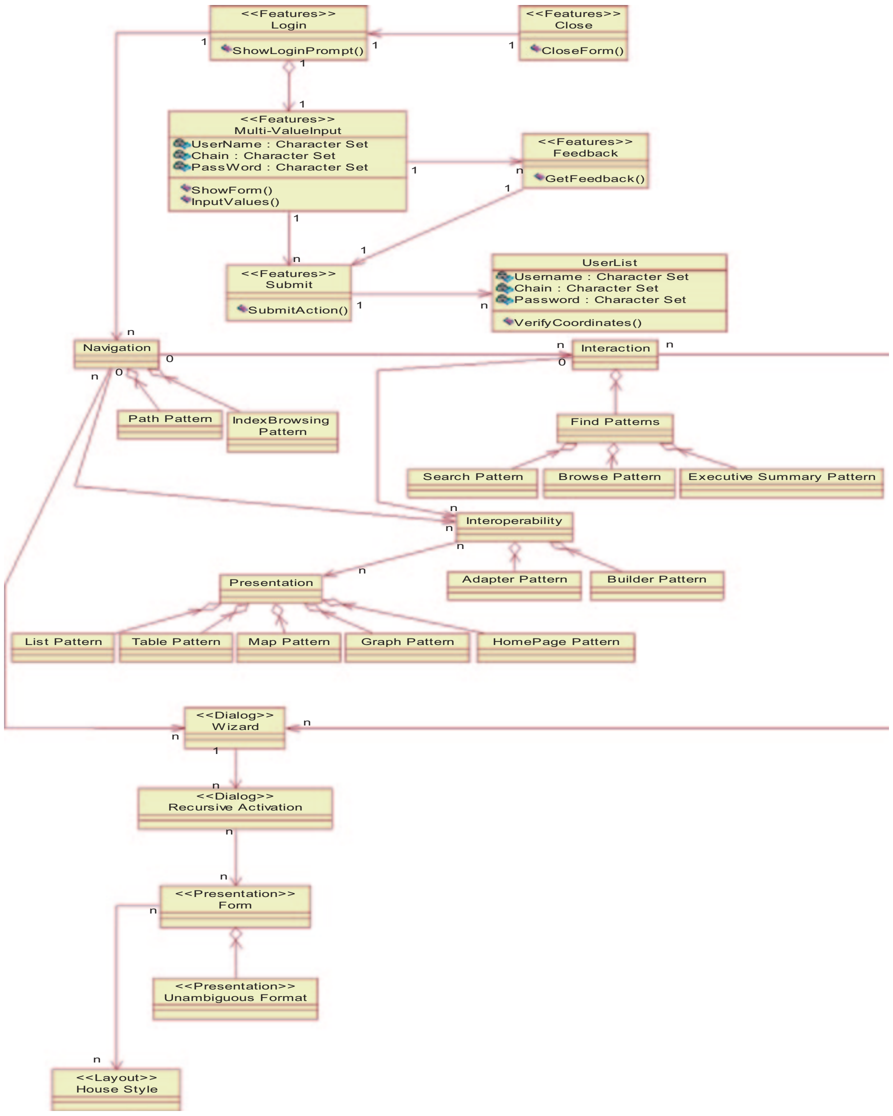


Fig. 9.22 UML diagram of PSM Layout Model for a laptop platform

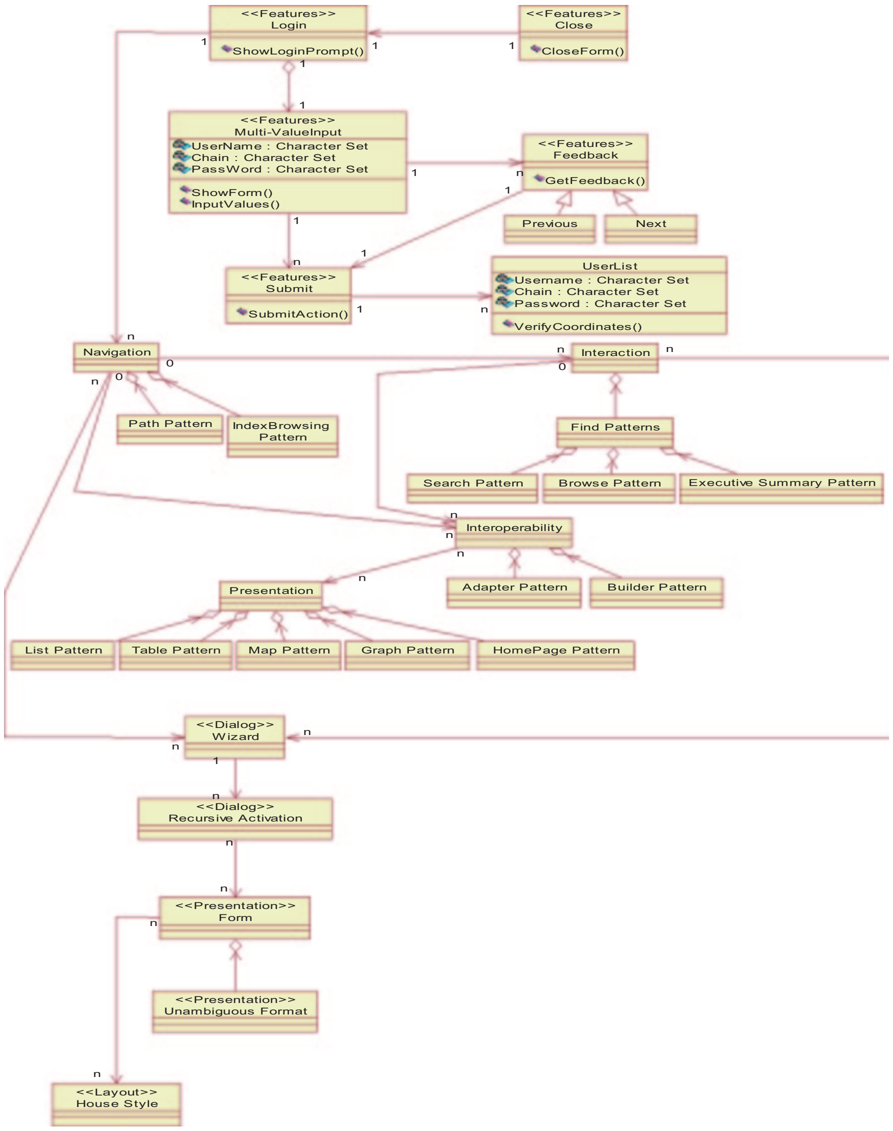
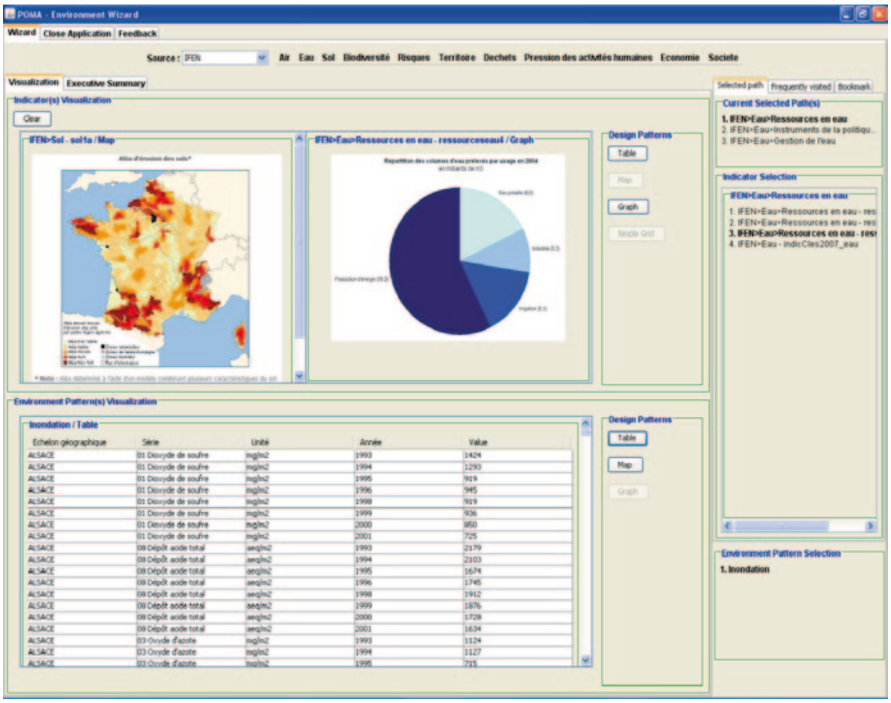


Fig. 9.23 UML diagram of PSM layout model for a PDA platform

Table 9.6 Example of pattern mapping of the layout model for laptop and PDA platforms

Patterns of Microsoft platform	Type of mapping	Replacement patterns for Laptop platform	Replacement patterns for PDA platform
P1. Login	Identical	P1. Login	P1. Login
P2. Multivalue input	Identical, scalable, fundamental	P2. Multivalue input	P2. Multivalue input
P3. Submit	Scalable or fundamental	P3. Submit	P3.s. Submit (smaller button)
P4. Feedback	Identical, fundamental	P4. Feedback	P4. Feedback P4.1. Previous P4.2. Next
P5. Close	Identical	P5. Close	P5. Close
P6. Find (search, browse, executive summary)	Identical, scalable	P6. Find (Search, Browse, Executive Summary)	P6. Find (Search, Browse, Executive Summary)
P7. Path (Breadcrumb)	-Identical, scalable (Laptop) -Scalable or fundamental (PDA)	P7. Path (Breadcrumb)	- P7.1s. Shorter bread crumb trial - P7.2. Drop-down "History" menu
P8. Index browsing	Identical	P8. Index browsing	P8. Drop-down menu
P9. Adapter	Identical	P9. Adapter	P9. Adapter
P10. Builder	Identical	P10. Builder	P10. Builder
P11. List	Identical	P11. List	P11. List
P12. Table	Identical	P12. Table	P12. Table
P13. Map	Identical	P13. Map	P13. Map
P14. Graph	Identical	P14. Graph	P14. Graph
P15. Home page	Identical	P15. Home page	P15. Home page
P16. Wizard	Identical	P16. Wizard	P16. Wizard
P17. Recursive activation	Identical	P17. Recursive activation	P17. Recursive activation
P18. Unambiguous format	Identical	P18. Unambiguous format	P18. Unambiguous format
P19. Form	Identical	P19. Form	P19. Form
P20. House style	Identical	P20. House style	P20. House style



1. IFEN>Eau>Ressources en eau
2. IFEN>Eau>Instruments de la politiqu...
3. IFEN>Eau>Gestion de l'eau

1. IFEN>Eau>Ressources en eau - res
2. IFEN>Eau>Ressources en eau - res
3. IFEN>Eau>Ressources en eau - res
4. IFEN>Eau - indicCles2007_eau

1. Inondation

Echelon géographique	Série	Unité	Année	Value
ALSACE	01 Dioxyde de soufre	mg/m2	1993	1424
ALSACE	01 Dioxyde de soufre	mg/m2	1994	1293
ALSACE	01 Dioxyde de soufre	mg/m2	1995	919
ALSACE	01 Dioxyde de soufre	mg/m2	1996	945
ALSACE	01 Dioxyde de soufre	mg/m2	1998	919
ALSACE	01 Dioxyde de soufre	mg/m2	1999	936
ALSACE	01 Dioxyde de soufre	mg/m2	2000	850
ALSACE	01 Dioxyde de soufre	mg/m2	2001	725
ALSACE	08 Dépôt acide total	aeq/m2	1993	2179
ALSACE	08 Dépôt acide total	aeq/m2	1994	2103
ALSACE	08 Dépôt acide total	aeq/m2	1995	1674
ALSACE	08 Dépôt acide total	aeq/m2	1996	1745
ALSACE	08 Dépôt acide total	aeq/m2	1998	1912
ALSACE	08 Dépôt acide total	aeq/m2	1999	1876
ALSACE	08 Dépôt acide total	aeq/m2	2000	1728
ALSACE	08 Dépôt acide total	aeq/m2	2001	1634
ALSACE	03 Oxyde d'azote	mg/m2	1993	1124
ALSACE	03 Oxyde d'azote	mg/m2	1994	1127
ALSACE	03 Oxyde d'azote	mg/m2	1995	715

Fig. 9.24 Screenshot of the Environmental Management Interactive System for a Laptop platform

9.5 Key Issues and Contributions

A prototype of this case study was developed for an environmental management interactive system. This prototype was developed in Java Eclipse tool. In this case study, patterns were identified and applied for each of the models that were used during development.

The main purpose of the prototype of this case study is to show that model-driven architecture development consists of model transformation and that mapping rules from the abstract to the concrete models are specified and—more importantly—automatically supported by tools.

In the case study, UML notation was used to design the five models (domain, task, dialog, presentation, and layout). XML notation was also used to describe the five models and the different types of patterns proposed by the POMA architecture. UML and XML allow one to communicate the modeling semantics between the different models, helping tailor the application and corresponding models to different platform and user roles.

This chapter created a practical multiplatform architecture for interactive systems engineering. The main contributions are:

1. The creation of six architectural levels and categories of patterns (navigation patterns, interaction patterns, visualization patterns, presentation patterns, interoperability patterns, and information patterns) (Taleb et al. 2006), (Taleb et al. 2007a) and (Taleb et al. 2007c);
2. The creation of different relationships between patterns which are used to create a pattern–oriented design using composition rules and mapping rules and to generate specific implementations suitable for different platforms from the same pattern-oriented design (Taleb et al. 2006) and (Taleb et al. 2007c);
3. The use of five categories of models: domain model, task model, dialog model, presentation model, and layout model (Taleb et al. 2007b) and (Taleb et al. 2010a);
4. The creation of different model transformation rules to transform only the PIM and PSM models between them such as: PIM to PIM, PIM to PSM, and PSM to PSM (Taleb et al. 2010b);
5. Development of the "Environmental Management Interactive System" case study. The case study illustrates and clarifies the core ideas of this approach and its applicability and relevance to multiplatform development.

References

IBM (2015) IBM design language: find your voice. http://www.ibm.com/design. Accessed 15 April 2015

Lynch PJ, Horton S (1999) Web style guide: basic design principles for creating web sites. Yale University Press, New Haven

Macintosh (1992) Human interface guidelines. Apple computer company. Addison Wesley, Cupertino. http://interface.free.fr/Archives/Apple_HIGuidelines.pdf†

Microsoft (1995) The windows interface guidelines for software design. Microsoft Press, Redmond. http://www.ics.uci.edu/~kobsa/courses/ICS104/course-notes/Microsoft_Windows-Guidelines.pdf†

Seffah A, Gaffar A (2006) Model-based user interface engineering with design patterns. J Syst Soft 80(8):1408–1422. doi:10.1016/j.jss.2006.10.037. (15 pages)

Shneiderman B (2000) Universal usability. Commun ACM 43(5):84–91

Sinnig D (2004) The complicity of patterns and model-based UI development. Master of computer Science. Concordia University Press, Montreal, p 148

Sun Microsystems (2001) Java look and feel design guidelines. Addison Wesley Professional http://java.sun.com/products/jlf/ed2/book/†

Taleb M, Javahery H, Seffah A (2006) Pattern-oriented design composition and mapping for cross-platform web applications. The XIII International Workshop. DSVIS 2006. July 26–28 2006. Trinity College Dublin Ireland. doi:10.1007/978-3-540-69554-7. ISBN 978-3-540-69553-0. Vol. 4323/2007. Publisher Springer-Verlag Berlin Heidelberg. Germany

Taleb M, Seffah A, Abran A (2007a) Pattern-oriented architecture for web applications. 3rd International Conference on Web Information Systems and Technologies (WEBIST). March 3–6, 2007. ISBN 978-972-8865-78-8. pp 117–121. Barcelona. Spain

Taleb M, Seffah A, Abran A (2007b) Model-driven design architecture for web applications. The 12th International Conference on Human Centered Interaction International (FIC-HCII). July 22–27, 2007. Beijing International Convention Center. Beijing. P. R. China. Vol 4550/2007, pp 1198–1205. Publisher Springer-Verlag Berlin Heidelberg. Germany

Taleb M, Seffah A, Abran A (2007c) Patterns-oriented design for cross-platform web-based information systems. The 2007 IEEE International Conference on Information Reuse and Integration (IEEE IRI-07). August 13–15, 2007. Pages 122–127. Las Vegas. USA

Taleb M, Seffah A, Abran A (2010a) 'Investigating model-driven architecture for web-based interactive systems'. eMinds: Int J Hum Comput Interact II(6):1697–9613

Taleb M., Seffah A, Abran A (2010b) 'Transformation rules in POMA architecture'. The 2010 International Conference on Software Engineering Research and Practice (SERP'10), July 12–15, 2010, CSREA Press, pp. 161–166. ISBN: 1–60132–160–0, Las Vegas, Nevada, USA

Welie VM (2004) Patterns in interaction design. http://www.welie.com/