A UsiXML Proposal for a Pattern-Oriented and Model-Driven Architecture for Interactive Systems

Mohamed Taleb
Department of Software Engineering & Information Technology
Software Engineering Research Laboratory
École de technologie supérieure, University of Quebec, Montreal, Quebec, Canada
mohamed.taleb.1@ens.etsmtl.ca

Ahmed Seffah
Depatment of Computer Science and Software Engineering
Human-Centered Software Engineering Group
Concordia University, Montreal, Quebec, Canada
seffah.ahmed@yahoo.fr

Alain Abran
Department of Software Engineering & Information Technology
Software Engineering Research Laboratory
École de technologie supérieure, University of Quebec, Montreal, Quebec, Canada
alain.abran@etsmtl.ca

***Abstract*—Despite its obvious and well-publicized potential to support the model-driven engineering of user interfaces, the (re)use of the rich variety of Human-Computer Interaction (HCI) design patterns, we have today has not achieved the acceptance and widespread applicability of HCI design patterns within the existing model-driven engineering framework. This paper proposes a specification and a User Interface eXtensible Markup Language (UsiXML)-based formalization of a unifying Pattern-Oriented and Model-driven Architecture (POMA). We have already introduced a set of extensions, called the POMA Markup Language (POMAML), designed to facilitate the specification of all the intrinsic components of the POMA architecture, including its patterns and models, and the relationships between these two artifacts within the model.**

Keywords-Pattern-Oriented; Model-Driven Architecture; UsiXML; User Interface; POMA.

I. INTRODUCTION

Day-to-day experience suggests that it is not enough to approach a complex design with a set of models and model-driven engineering languages and tools. The developers must also be able to use (reuse) proven solutions emerging from the best model-driven practices for building models and their transformations, as well as for generating code for diverse platforms.

Without these solutions, developers are unable to properly define valid models, and so cannot take full advantage of the power of the model orientation, resulting in poor performance. Invalid models will lead to poor scalability and usability. Furthermore, the designer might find himself "reinventing the wheel" when attempting to develop an application.

We propose to enhance, extend, or rethink the activities and artifacts of the model-driven engineering frameworks using patterns for model construction, transformation, and mapping. We proposed POMA (Pattern-Oriented and Model-driven Architecture) [1] as a unifying architecture to bridge the gap between patterns and models, as well as between the model-driven engineering and pattern-oriented design frameworks.

Specifically, we consider the possible extensions to the User Interface eXtensible Markup Language (UsiXML) collection of models [2] and the four basic levels of model abstraction defined in the Cameleon Reference Framework [2]. UsiXML defines, validates, and standardizes an open User Interface Description Language, while increasing the productivity and reusability of multi-platform and multi-context interactive applications. It also improves the usability and accessibility of these applications.

In our ongoing research, we are aiming at specifying and representing the components of the POMA architecture. We suggest extensions to the concepts of UsiXML to formalize a language called POMAML (Pattern-Oriented and Model-driven Architecture Markup Language). In other words, POMA is a unifying architecture to bridge the gap between patterns and models using POMAML.

This paper is organized as follows. Section 2 introduces related work on POMA fundamentals, the basic concepts of the POMA architecture, and the basic structural notation of UsiXML. Section 3 primarily describes the application of UsiXML in the POMA architecture. Section 4 presents an illustrative case study. Section 5 presents a summary and directions for future work.

II. RELATED WORK

Over the past two decades, research on interactive system and User Interface (UI) engineering has resulted in several architectural models, which constitute a major contribution not only to facilitate the development and maintenance of interactive systems, but also to promote the standardization, portability, and ergonomic usability (ease of use) of the interactive systems developed. Such architectures provide a clear separation of concerns [3]. In particular, they decouple the UI from the system semantics, and define the reusable and the standardized UI components.

A number of UI languages and notations have been suggested to specify architecture and model user interfaces for different platforms and at different levels of abstraction. For example, User Interface Markup Language (UIML) [4] is a meta-language that allows the developer to describe the UI in generic terms and to use style descriptions to map the UI to various target platforms. UIML was developed to address the need for a uniform UI description language for building multi-platform systems. eXtensible User-interface Language (XUL) [5; 6] is an official Mozilla initiative,

 ISBN: 978-1-61208-290-5

which provides an XML-based language for describing window layout. The goal of XUL is to build cross platform systems that are easily portable to all the operating systems on which Mozilla runs. XUL provides a clear separation between the UI definition (the various widgets that make up the UI) and its visual appearance (the layout and the "look and feel").

EXtensible Interface Markup Language (XIML) followed a declarative interface modeling language called MIMIC [7], and provides a way to describe the UI without worrying about its implementation. The aim of XIML is to describe the various abstract aspects (domain, task, and user) and concrete aspects (presentation and dialog) of the UI throughout the development life cycle. In addition, XIML supports the definition of mapping from abstract elements to concrete elements [8].

TERESA [9] provides tools to allow developers to interactively define mappings between the various models. Web Services eXtensible Markup Language (WSXML) [10] integrates Web services and XML into the Service Oriented Architecture. Web services constitute a technological approach that is well suited to bridge information systems, and can enable this integration, even when systems are implemented on disparate platforms or through differing technologies. XML is useful for a variety of data exchange applications, and is a foundation technology for such enterprise strategies as Web services, and likely has a future in enterprises.

GrafiXML, developed by Limbourg et al. [2], is an original UI builder, in that it enables designers and developers to design several UIs simultaneously for multiple contexts of use, i.e., for many users, platforms, and environments. GrafiXML is an intelligent UI builder, in that it maintains model consistency between these representations through a set of mappings based on the UI ontology.

Following the lead of the object-oriented software design community, HCI practitioners investigated design patterns as one possible way to capture and use the best design practices. An HCI design pattern is defined as a named, reusable solution to a recurring user problem in different contexts of use, including the various computing platforms (Web, Graphical User Interface (GUI), mobile applications, etc.). Relationships between patterns have been explored to combine related patterns into pattern languages, resulting in a lingua franca for design [12].

Hundreds of HCI design patterns are freely available on the Web. However, providing a list of patterns and their loosely defined relationships, as is done for most HCI pattern languages, is insufficient for effectively driving design solutions. Understanding when a pattern is applicable during the design process and how it can be used, as well as how and why it can or cannot be combined with other related patterns, are key notions in the application of patterns.

Javahery and Seffah [3] proposed a design approach, called Pattern-Oriented Design (POD), which provides a framework for guiding designers through stepwise design suggestions. At each predefined design step, designers are given a set of applicable patterns. This process is in stark contrast to the current use of pattern languages, where there is no defined link to any sort of systematic method. Pattern relationships are explicitly described, which allows designers to compose patterns based on an understanding of these relationships. In POD, patterns are building blocks at different levels of abstraction, which makes them extremely useful for designers when driving the UI design based on user experiences [14; 15; 3].

The proposed Pattern-Oriented and Model-driven Architecture (POMA) (Figure 1) [1] identifies an extensive list of pattern categories and types of models aimed at providing a pool of proven solutions to these problems. The models of patterns span several levels of abstraction, such as domain, task, dialog, presentation, and layout. The proposed POMA architecture illustrates how several individual models can be combined at different levels of abstraction into heterogeneous structures, which can then be used as building blocks in the development of interactive systems.

The various components of the POMA architecture are detailed in [1], and include:

- The architectural levels and various categories of patterns [16], [17], and [19];
- The Platform Independent Model (PIM) and Platform Specific Model (PSM) [18];
- The pattern composition rules for selecting and composing patterns corresponding to each type of PIM model [16] and [18];
- The rules for mapping patterns and PIM models to produce PSM models for multiple platforms [16] and [18];
- The rules for transforming PIM to PIM models and PSM to PSM models [20];
- The rules for source code generation;
- The generation of the whole of application.

The rationale and strengths of the POMA architecture are as follows:

- POMA facilitates the use of patterns by beginners as well as experts;
- POMA supports the automation of both the pattern-driven and model-driven approaches to design;
- POMA supports the communication and reuse of individual expertise regarding good design practices;
- POMA can integrate all the new technologies, including traditional office desktops, laptops, Palmtops, PDAs (with or without keyboards), mobile telephones, and interactive televisions, among others.

 ISBN: 978-1-61208-290-5

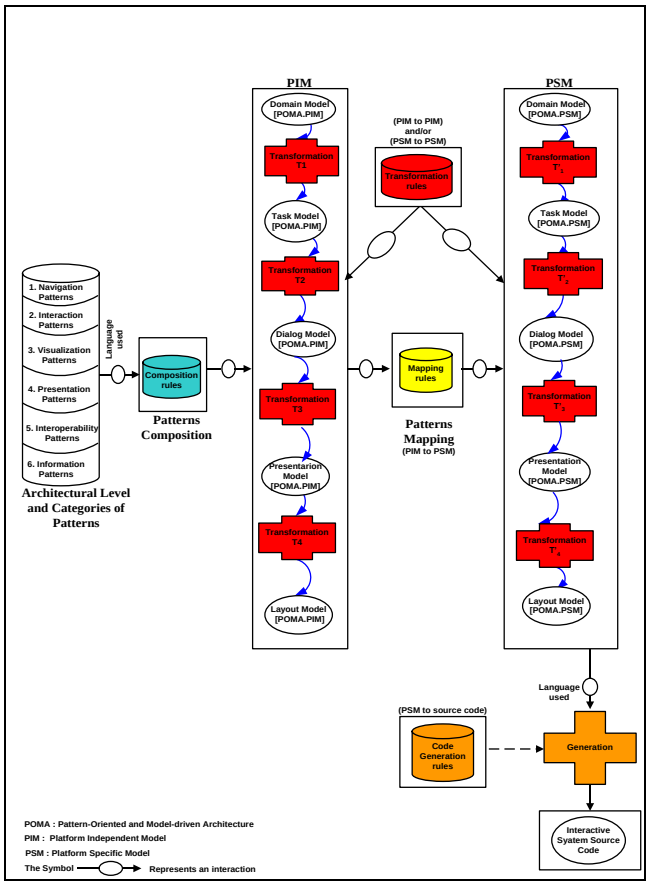


Figure 1. POMA architecture for interactive system development [1].

The User Interface eXtensible Markup Language (UsiXML) [21; 24] is an XML-compliant markup language. It is an approach for describing the structure and presentation aspects of the UI to describe dialog modeling [22]. Limbourg et al. [2] describe the structured UsiXML, based on the following four basic levels of abstraction defined in the Cameleon reference framework. This framework is intended to represent the UI development life cycle for context-sensitive interactive applications. In other words, the framework defines UI development steps for multi-context interactive applications. It structures development processes for two contexts of use into four development steps (each development step being able to manipulate any specific artefact of interest as a model or a UI representation):

1. Task and Concepts (the highest level), where the user task is defined based on his viewpoint, along with the various objects that are manipulated by it.
2. Abstract User Interface (AUI): abstracts the Concrete User Interface (CUI) into a UI definition that is interaction modality independent (e.g., graphical/vocal interaction);
3. Concrete User Interface (CUI): abstracts the Final User Interface (FUI) into a UI definition that is independent of any computing platform;
4. Final User Interface (FUI): a UI running on a particular platform, either by interpretation or by execution.

UsiXML is defined as a set of XML schemas, each corresponding to one of the models within the scope of the language. It consists of a User Interface Description Language (UIDL), which is a declarative language capturing the essence of what a UI is, or should be, independently of physical characteristics. It describes the constituent elements of the UI of an application at a high level of abstraction: widgets, controls, containers, modalities, interaction techniques, etc. Despite that, UsiXML does not require the use of any particular development process, which means that designers are free to choose the most appropriate abstraction level at which to begin their projects [23].

III. POMA COMBINED WITH THE USIXML APPROACH

To tackle some of the weaknesses identified in related work, a set of UsiXML concepts proposes to specify and formalize the POMA architecture within the UsiXML perspective (Figure 3) and its language, which is called the POMA Markup Language (POMAML) and is described in section 4 (Pattern-Oriented Modeling Architecture Markup Language). The formalization is achieved in visual, structural, and formal notations using XML for modeling the patterns and models of the POMA components described in section II in order to generate the specifications for various types of UI engineered for interactive systems. Our aim is to persevere with this objective, and continue to design and reuse POMA architecture specifications that span different levels of abstraction, such as the domain, task, dialog, presentation, and layout models, until the final layout of the various UIs has been generated.

Because of the number of concepts it embodies, UsiXML is used to illustrate the POMA architecture (Figure 3). On the left is a series of development steps that comply with the Cameleon reference framework [22], and on the right are the concepts supported by UsiXML, and the transformations and mappings applied to it. POMA architecture based on UsiXML classifies UIs for supporting a target platform and a context of use, and enables to structure the development life cycle into five levels of abstraction and patterns categories as follows (Figure 2):

1. Categories patterns library. These patterns of different categories are defined and formalized in XML language;
2. Five categories of models in PIM and PSM (Task, Domain, Dialog, Presentation, Layout) used in POMA architecture, providing examples, for a model-driven architecture for interactive systems to resolve many recurring design problems, examples of which include: (1) decoupling the various aspects of Web applications such business logic, the user interface, navigation and information architecture; (2)

 ISBN: 978-1-61208-290-5

isolating platform-specific problems from the concerns common to all interactive systems.

3. Abstract User Interfaces (AUI) of PIM. This abstraction level defines a generic user interface description of PIM models completely independent of the considered UI toolkit and multi-platforms.
4. Concrete User Interface (CUI) Platform Independent Model (PSM) for different platforms (Laptop, PDA, Cellular, Palmtop, interactive television, iPhone, etc.). This level defines the graphical concrete user interfaces, including the concrete interaction objects (CIO), for each specific platform.
5. Final User Interface (FUI). This level is to generate the source code of the entire application for a specific platform.

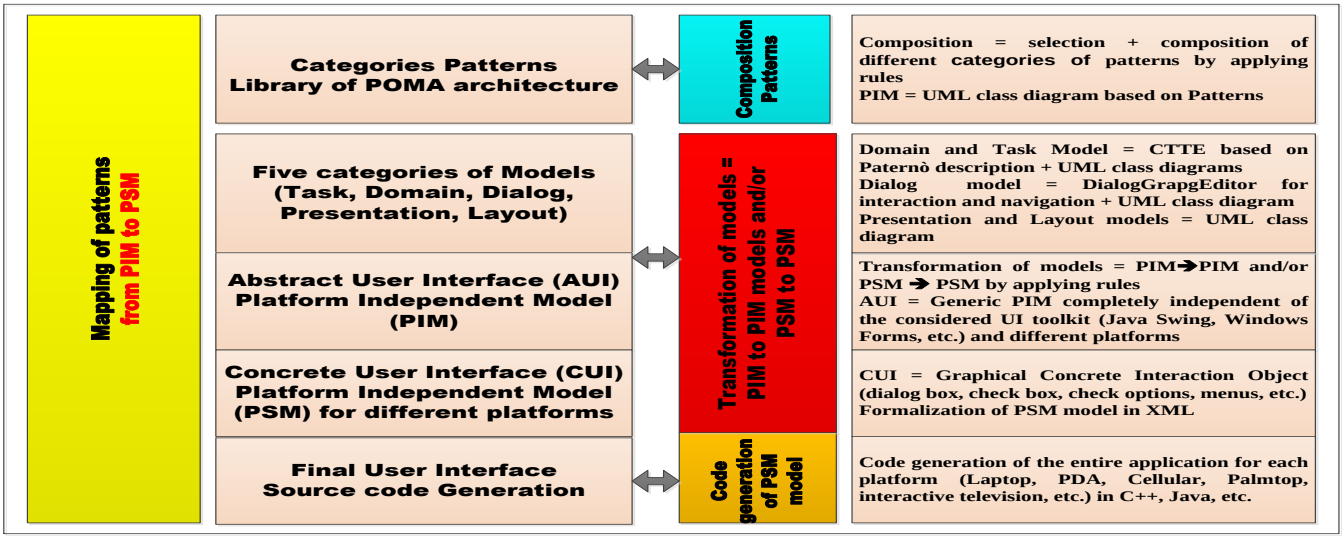


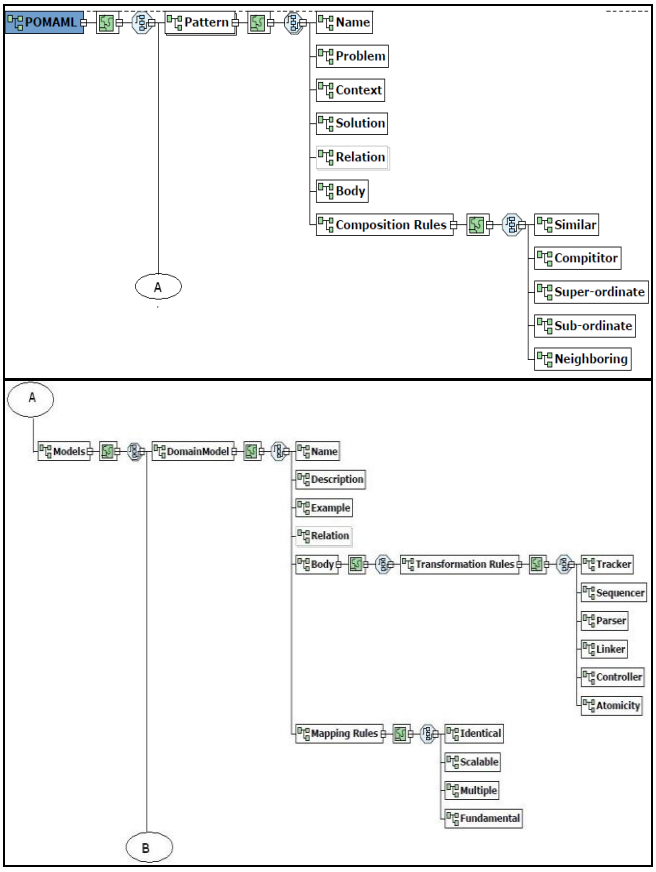
Figure 2. POMA architecture in UsiXML perspective.

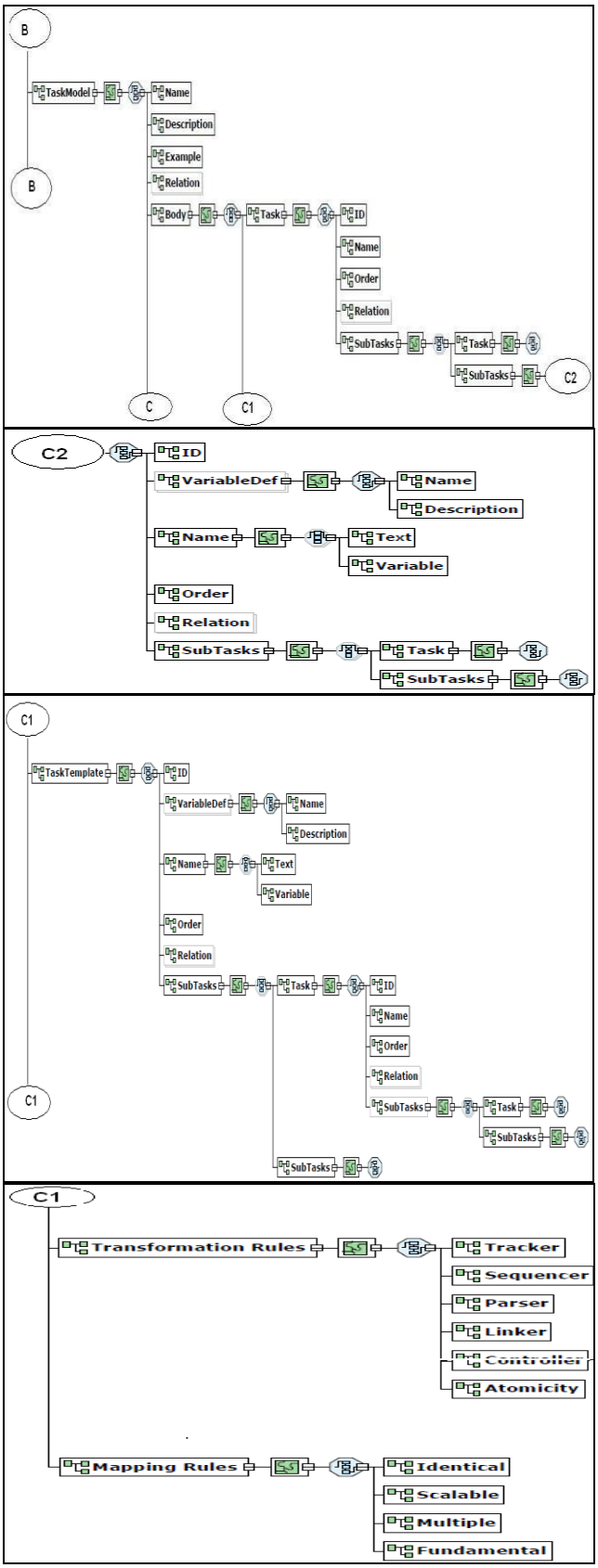
In this section, we describe a design that illustrates and clarifies the core ideas underlying the approach combining the POMA architecture with the UsiXML, and explain its practical relevance. The proposed POMA architecture combined with UsiXML (Figure 2) shows how UsiXML concepts are used to represent the components of the POMA architecture to generate the source code of the various concrete UIs of the application.

With the POMA architecture, it is possible to design a formalism to describe a software architecture based on the composition of several patterns to generate different types of applications. This formalism can take three forms:

- Structural, using the XML formalization language called POMAML;
- Formal, using mathematical methods and concepts;
- Visual, using UML specifications such as sequence diagrams and class diagrams.

Here, we focus essentially on the use of the structural notation to describe the entire POMAML language (Figure 3) of the POMA architecture components, such as patterns, composition rules, levels of PIM and PSM models, transformation rules, mapping rules, and generation rules based on the XML notation.





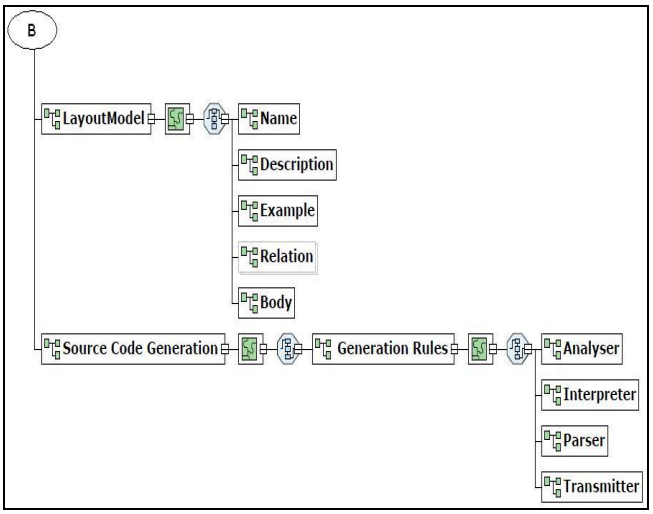


Figure 3. POMAML language

IV. AN ILLUSTRATIVE CASE STUDY

This following example presents the domain model (Figure 4) of the POMA architecture for a laptop platform using UsiXML concepts. In this case, the more those high-level tasks are decomposed, the easier it is to use the reusable task structures that have been obtained or captured from other projects or systems. Here, these reusable task structures are documented in the form of patterns. This approach ensures an even greater degree of reuse.

```
<xs:element name="DomainModel">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Name" type="xs:string"/>
      <xs:element name="Description" type="xs:string"/>
      <xs:element name="Example" type="xs:string"/>
      <xs:element name="Relation" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="Body">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Transformation Rules">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="Tracker"/>
                  <xs:element name="Sequencer"/>
                  <xs:element name="Parser"/>
                  <xs:element name="Linker"/>
                  <xs:element name="Controller"/>
                  <xs:element name="Atomicity"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="Mapping Rules">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Identical"/>
            <xs:element name="Scalable"/>
            <xs:element name="Multiple"/>
            <xs:element name="Fundamental"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Figure 4. Domain Model in POMAML language

V. SUMMARY AND FUTURE WORK

In this paper, we have discussed a perspective from which the POMA architecture is specified and represented using the UsiXML approach. Previously, we had provided a set of extensions, called POMAML, which makes it possible to generate source code in different programming languages for each platform of an interactive system.

Our research has resulted in the integration and formalization of UsiXML for the POMA architecture. It has also led to avenues for further research, such as:

- Description of a process for the generation of source code from POMA's five PSM models;
- Development of a tool that automates the POMA architecture-based engineering process;
- Standardization of the POMA architecture to all types of systems, and not only to multi-platform interactive systems;
- Quality assurance of the applications produced, since a pattern-oriented architecture will also have to provide for the encapsulation of quality attributes and facilitate prediction;
- Validation of the migration, usability, and overall quality of the POMA architecture for interactive systems using existing methods;
- Evaluation of the effectiveness and learning time of the POMA architecture for both novices and expert users.

REFERENCES

[1] M. Taleb, A. Seffah, and A. Abran, "Interactive Systems Engineering: A Pattern-Oriented and Model-Driven Architecture", The 2009 International Conference on Software Engineering Research and Practice (SERP'09), July 2009, pp. 636-642, Las Vegas, USA.

[2] Q. Limbourg, J. Vanderdonckt1, B. Michotte1, L. Bouillon1, and V. López-Jaquero, "USIXML: A Language Supporting Multi-path Development of User Interfaces", vol. 3425/2005, *Engineering Human Computer Interaction and Interactive Systems*, July 2005, pp. 200-220, DOI 10.1007/b136790, ISBN 978-3-540-26097-4, Springer Berlin/Heidelberg Publisher.

[3] H. Javahery and A. Seffah, "A Model for Usability Pattern-Oriented Design", in Proceedings of TAMODIA2002, July 2002, pp. 104-110, Bucharest, Romania.

[4] M. Abrams, C. Phanouriou, A. L. Batongbacal, S. M. Williams, and J. E. Shuster, "UIML: An Appliance-Independent XML User Interface Language", Proceedings of the 8th International WWW Conference, May 1999, pp. 1695–1708, Elsevier Science Publishers, Toronto, Canada.

[5] XUL, The XML User Interface Language, 2004, at: http://www.xulplanet.com, [retrieved: April, 2013].

[6] XUL, XUL Tutorial, 2004, at: http://www.xulplanet.com/tutorials/xultu, [retrieved: April, 2013].

[7] A. Puerta and D. Maulsby, "Management of Interface Design Knowledge with MODI-D", in Proceedings of IUI'97, January 1997, pp. 249-252, Orlando, FL, USA.

[8] A. Puerta and J. Eisenstein, "Towards a General Computational Framework for Model-Based Interface Development Systems", in Proceedings of IUI'99, January 1999, pp. 171-178, Los Angeles, CA, ACM Press, New York, USA.

[9] TERESA, Transformation Environment for Interactive Systems Representation, 2004, at: http://giove.cnuce.cnr.it/teresa.html, [retrieved: April, 2013].

[10] P. Classon and J. Prem, "SOA: Integrating XML and Web Services," LiquidHub Inc., 2004, at: http://www.liquidhub.com/docs/Horizons_WSXML_primer_v3.pdf, [retrieved: April, 2013].

[11] B. Michotte and J. Vanderdonckt, "GrafiXML, A Multi-Target User Interface Builder based on UsiXML", Fourth International Conference on Autonomic and Autonomous Systems, IEEE Computer Society, March 2008, pp. 15-22, DOI 10.1109/ICAS.2008.29, ISBN 0-7695-3093-1/08.

[12] T. Erickson, "Lingua Franca for Design: Sacred Places and Pattern Language", in Proceedings of Designing Interactive Systems, August 2000, pp. 357-368, ACM Press, New York (NY), USA.

[13] J. O. Borchers, "Pattern Approach to Interaction Design", Proceedings of the DIS 2000 International Conference on Designing Interactive Systems, August 2000, pp. 369-378, ACM Press, New York, USA.

[14] A. Granlund and D. Lafrenière, "A Pattern-Supported Approach to the User Interface Design Process", Workshop Report, UPA'99 Usability Professionals' Association Conference, June-July 1999a, Scottsdale, AZ.

[15] M. Taleb, H. Javahery, and A. Seffah, "Pattern-Oriented Design Composition and Mapping for Cross-Platform Web Applications, the 13th International Workshop, DSV-IS 2006, vol. 4323/2007, Springer-Verlag, July 2006, Trinity College, Dublin Ireland, DOI 10.1007/978-3-540-69554-7, ISBN 978-3-540-69553-0, Berlin Heidelberg, Germany.

[16] M. Taleb, A. Seffah, and A. Abran, "Pattern-Oriented Architecture for Web Applications", 3rd International Conference on Web Information Systems and Technologies (WEBIST 2007), March 2007, pp. 117-121, ISBN 978-972-8865-78-8, Barcelona, Spain.

[17] M. Taleb, A. Seffah, and A. Abran, "Model-Driven Design Architecture for Web Applications", The 12th International Conference on Human Centered Interaction International (FIC-HCII 2007), Beijing International Convention Center, Beijing, P. R. China, vol. 4550/2007, July 2007, pp. 1198-1205, Springer-Verlag, Berlin Heidelberg, Germany.

[18] M. Taleb, A. Seffah, and A. Abran, "Pattern-Oriented Design for Cross-Platform Web-based Information Systems", The 2007 IEEE International Conference on Information Reuse and Integration (IEEE IRI-07), August 2007, pp. 122-127, Las Vegas, USA.

[19] M. Taleb, A. Seffah, and A. Abran, "Transformation Rules in POMA architecture", The 2010 International Conference on Software Engineering Research and Practice (SERP'10), July 2010, pp. 636-642, Las Vegas, USA.

[20] UsiXML, What is UsiXML?, Université catholique de Louvain, Belgium, 2007, at: http://www.usixml.org/index.php?mod=pages&id=2, [retrieved: April, 2013].

[21] Q. Limbourg, J. Vanderdonckt, B. Michotte, L. Bouillon, M. Florins, and D. Trevisan: "UsiXML: A User Interface Description Language for Context-Sensitive User Interfaces", in Proceedings of the AVI'2004 Workshop on Developing User Interfaces with XML: Advances on User Interface Description Languages, UIXML'04, Gallipoli, Italy, EDM-Luc, May 2004, pp. 55–62.

[22] M. Winckler, F. M. Trindade, A. Stanciulescu, and J. Vanderdonckt, "Cascading Dialog Modeling with UsiXML," Proceedings of the 15th Int. Workshop on Design, Specification, and Verification of Interactive Systems DSV-IS'2008, Kingston, Canada, Lecture Notes in Computer Sciences, vol. 5136, Springer, Berlin, July 2008, pp. 121-135.

[23] Cover Pages (website hosted by OASIS): online resource for markup language technologies, "User Interface eXtensible Markup Language (UsiXML), 2005, at: http://xml.coverpages.org/userInterfaceXML.html#usixm, [retrieved: April, 2013].

 ISBN: 978-1-61208-290-5