The XIS Approach and Principles

Alberto Rodrigues da Silva
Instituto Superior Técnico / INESC-ID
alberto.silva@acm.org

Abstract

XIS is a R&D project which mission is to analyze, develop and evaluate mechanisms and tools to produce information systems from a more efficient and productive way than it is done currently. XIS project is influenced by MDA reference model, and is mainly based on three principles: it is based on high-level models specification; it is based on generative programming techniques; and it is component-based architecture-centric. XIS is not a conceptual research plan, it is a working on project with concrete results and produced systems. In this paper we overview the XIS project by introducing its main elements, such as the XIS approach; XIS platform; XIS/UML profile; and XIS/XML language. Finally, we present the main conclusions and the work that will be handled in the near future.

1 Introduction

Traditionally (and unfortunately) software industry puts a considerable emphasis and investment at the production level activities (such as programming, testing and integration of software components) in opposition to activities more related to the project and design (such as business and requirements engineering, analysis and design). According the XIS approach the emphasis should be put in the project and design activities and, consequently, the effort in production activities should be minimized and performed as automatically as possible. Of course, the "how to do" issues are also relevant, but are mainly handled by software architects that should provide elegant, flexible and reusable architectures.

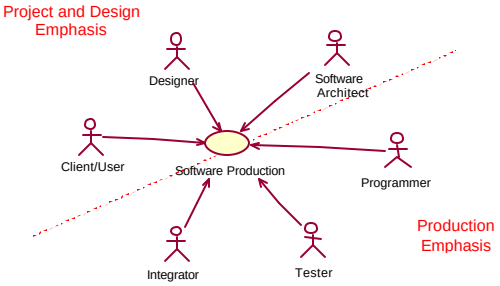

Figure 1. Key participants in the software industry.

The idea subjacent of the XIS project is not new by itself. At least since two decades ago researchers and engineers have been discussing this problem and have been arguing that software production should be more efficient, predictable and manageable. Despite this idea is not new, there was not such a success in the past: several projects based on that idea, particularly influenced by CASE tools technology, were too expensive and abandoned due to several reasons [12,13].

Nevertheless, we believe that nowadays the situation has changed positively. At least three technical issues are available today (that were not existed in the past) that contribute decisively to support this vision. First, there is a visual modeling language, the UML (*Unified Modeling Language*) [1,3,4,w3] that is a *de facto* standard, recognized and used by most of the industry. Second, there is a standard language to represent data and metadata, such as XML (*Extensible Markup Language*) [8,w2,w4] and in particular XMI (*XML Metadata Interchange*) [5]. Third, there is the recognition and the knowledge of high-quality design patterns and software architectures as the consequence of years of experience, reflection and reengineering of best practices in numerous projects and situations [14,15,16,17]. Hence, we believe there are currently technical conditions that allow the success of XIS' ideas.

The approach and tools presented in this paper is mainly conducted by our own experience of mentoring, designing and developing Information System projects. Among other best practices and patterns well known in the software engineering, we argue that the concurrent conjugation of the three main principles of XIS would short the time to develop and consequently the time to market new products; and would improve the overall quality and robustness of these products. One the other hand, this approach obliges a learning investment from designers, programmers and testers, and mainly requires a superior involvement from software architects.

This paper is structured around six sections. Section 1 describes the context and the motivation of the XIS project. Section 2 overviews the XIS project and its main elements, namely the XIS approach, the XIS platform, the XIS/UML profile and the XIS/XML specification language. Sections 3,4 and 5 discuss with some detail the most relevant principles adopted by the XIS approach: based on models specification (Section 3), based on generative programming techniques (Section 4), and component-based architecture-centric (Section 5). Finally, Section 6 concludes the paper, compares this work against the MDA (*Model Driven Architecture*) approach [6], and overviews the work to be developed in the near future.

2 The XIS "Big Picture"

The XIS project intends to apply different techniques and mechanisms in order to accelerate and to improve the software engineering activities. To reach such goals the project defines the following elements:

- *XIS approach* is a software development approach inspired by a set of best practices or principles, namely, it is based on high-level models or specifications; it is component-based architecture-centric; and it is based on generative programming techniques. Hence, XIS follows in essence the MDA philosophy with some specific particularities that are discussed in the end of this paper.
- *XIS platform* is a CASE tool to support the technical actors of the software development process according the XIS approach. The XIS platform is sustained by a repository that keeps related information, such as models, applications, software architectures, generated artifacts and even information concerning the software process itself (e.g., generation steps, tests and integration milestones). The XIS platform is a web-based, multi-user, multi-application and multi-architecture tool.
- *XIS/UML profile* is a set of coherent UML extensions that allows a high-level, visual modeling way to design information systems.
- *XIS/XML language* is a XML language that provides a textual, structured, understandable and compact way to specify information systems. While XIS/UML profile allows the definition of information systems with visual models, XIS/XML is its textual counterpart, that allows the definition of information systems with XML-based, textual specifications.

2.1 XIS Approach's Main Tasks

Figure 2 gives the big picture of the XIS approach's main tasks and actors. Broadly, XIS approach receives system requirements (e.g., functional, non-functional and development requirements) as its main *input*, and produces a set of artifacts (e.g., source code, configuration scripts or data scripts) as its main *output*.

We start by identify the tasks performed by the **software architect**, which are critical to the correct operation of the XIS approach. Specifically, the architect should be responsible by the following tasks: (1) define a suitable and easy-to-use XIS/UML profile; (2) provide the T2 transformation support, particularly define templates that would support the transformation from XMI to the XIS/XML format (see Section 3.4 and 4); and (3) define software architecture templates, in order to support T3 transformation, i.e., the transformation from XIS/XML specification to the final software artifacts (see Section 4).

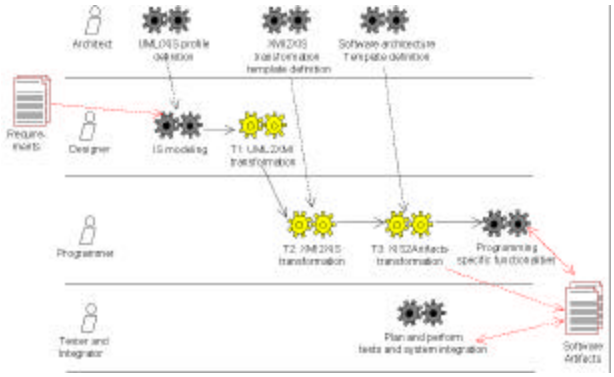

Figure 2. Key elements of the XIS approach.

Starting from system requirements (created for instance from meetings among designers, clients and end-users) it is **designer**'s responsibility to produce a suitable and corresponding information system model (the "IS modeling" task). The correctness and quality of those models is essential to obtain good results in the subsequent tasks. T1, T2 and T3 correspond to the successive application of three transformations, which transform high-level specifications into a full-size set of software artifacts (see Section 4 for more details).

Because it is not possible to model all the system requirements (for instance, business rules or non-functional requirements) at the XIS/UML level, it is required the programmer intervention. Consequently, **programmers** are required to produce specific components, typically helper source code, such as facades, adapters, controllers and business logic [14,16]. These activities are suggested in the Figure 2 by means of the "Programming Specific Functionalities" task.

Finally, it is necessary the intervention of **testers and integrators** to prepare and to perform different tests in order to guarantee the system quality. These activities are suggested in the Figure 2 by the "Plan and Perform Tests and System Integration" task.

2.2 XIS Approach's Main Principles

It is evident from Figure 2 the main principles associated with the XIS approach. First, the fact that XIS **is based on models specification** evidences the importance of the XIS/UML profile and XIS/XML language definition and the quality of the produced information systems models. Second, the fact XIS **is component-based architecture-centric** requires that architects define software architectures (that involves producing different artifacts, such as the "architecture template" that is used by the T3 transformation), as well as should manage them in the XIS platform. On the other hand, it is programmers' responsibility to choose the convenient architecture that should be used by T3 transformation. Third, the fact that

XIS ***is based on generative programming techniques*** is evidenced by T1, T2 and T3 transformations.

These principles are discussed and analyzed in the next sections of this paper.

3 XIS is based on Models Specification

3.1 Extension Mechanisms and UML Profiles

UML (*Unified Modeling Language*) is a well-known OMG standard used widely by the software industry. UML is a modeling language to visually represent different perspectives of information systems eventually represented at different abstraction levels [1,3,4].

In order to satisfy current but also future needs for software modeling, the UML proponents defined a somehow innovative and polemic concept, which allows basically extending UML by third parties without special requirements (and without have to change its own metamodel) [3,4,7]. To do that, UML provides three mechanisms, called "extensions mechanisms": constraints, tag-values (i.e., metadata) and stereotypes (i.e., metatypes). These mechanisms allow: (1) to introduce new modeling elements in order to increase the expressiveness and understandability of final models; (2) to define standard items that were not considered relevant enough to be part of the UML metamodel; (3) to define extensions better aligned with specific programming languages or with specific development processes; and (4) to associate arbitrary semantic information to some elements.

By their own nature and characteristics, extensions are not usually known, understandable, supported and accepted by the majority of UML users. In order to have a better way to deal with extensions, that are steadily proposed and promoted by different parties, OMG proposed (in the UML version 1.3) the concept of profile. So, "UML profile" is basically a name given to a predefined and coherent group of stereotypes, tags and constraints, which provides a specialization and configuration of UML for a specific application domain or a specific development process [7].

3.2 XIS/UML Profile

The XIS UML profile (or "XIS/UML profile") is a coherent group of UML extensions that allows to model information systems according the XIS approach. This profile is in fact inspired by a set of well-known best practices and features such as [9,16,17]: (1) software architectures, and in particular the MVC (*model-view-control*) architecture; (2) business entities; (3) predefined user interfaces workflows; (4) generic user interfaces workflows, for instance following high-level languages such as UIML [10,w5]; and (5) business workflows, for instance following high-level languages such as BPEL4WS [20].

The first three referred features have been considered in the current effort of the XIS design while the others should be considered in the near future. (In this paper we present the XIS/UML profile accordingly its current state.)

The XIS/UML profile advocates modeling an information system following the MVC architecture, based on four complementary models, precisely: the domain model (*Model*) and the business entities model (*BusinessEntity*); the workflows model (*Controller*); and the views model (*View*). These four models should be organized in different UML packages with corresponding dependencies, as illustrated in the Figure 3.

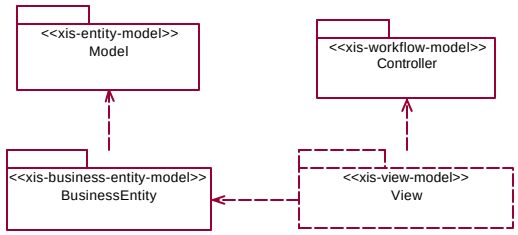

Figure 3. Typical organization of IS modeling following the XIS/UML profile.

Entity Model (Model)

In the domain model you put the classes and relationships corresponding to the entities identified normally during the inspection of the problem domain. These classes are identified by the «xis-entity» stereotype.

The MyContacts Case Study

MyContacts is a PIM (personalized information manager) advanced system (such as Microsoft's Outlook) that provides the following features: contacts management, reporting, labels printing, XML data import/export, and data synchronization among different versions of the MyContacts products family.

MyContacts products family is developed in Java according the following conditions:

- MyContacts-Mobile, for mobile devices, on top of the J2ME (Java 2 Micro Edition);
- MyContacts-Standard, for desktop PC, on top the J2SE (Java 2 Standard Edition);
- MyContacts-Enterprise, for enterprise environment, on top of the J2EE (Java 2 Enterprise Edition).

MyContacts is the first information system developed according the XIS approach. It is basically a contact management system having, as the main novelty, the support of a integrated set of complementary applications running on different platforms and software architectures, to be precise J2SE, J2EE and J2ME.

This case study will be used along this section to support the explanation.

Figure 4 depicts a simplified representation of an entity model regarding the MyContacts case study.

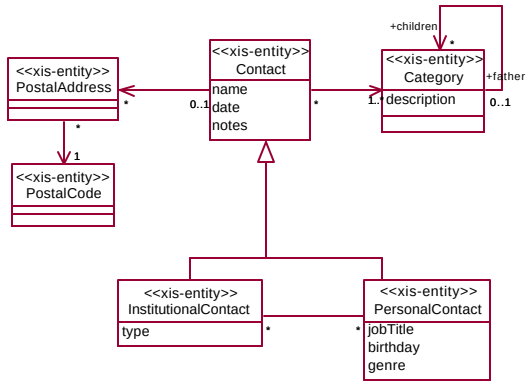

Figure 4. Outline of MyContacts' domain model.

Business Entities Model (BusinessEntity)

The emphasis of the business entities model is the definition of business entities, i.e., classes with «xis-business-entity» stereotypes, which are in fact semantic aggregation of several entities defined in the domain model. However, these new "business-entities" provide a superior level in terms of granularity, business, or end-user interaction comparing with the others.

Some business entities are mapped one-to-one regarding a domain model entity (e.g., "CategoryBE" and "PostalAddressBE" from Figure 5) while others (e.g., "PersonalContactBE" from Figure 5) aggregate several ones. In this situation it is convenient to add semantic to the involved relationships in order to clarify the role or the order of the entity in relation to the business entity. We support that semantic through two tag-values, namely: `role` (supporting the "master", "detail" or "lookup" values); and `index` (use to specify the relative order). Additionally, the designer can enhance the model legibility giving expressive names to these relationships (such as we do in Figure 5).

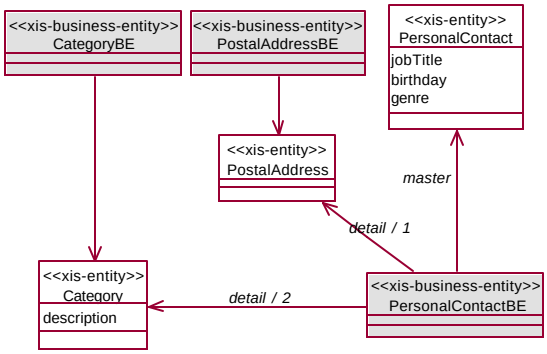

Figure 5. Outline of MyContacts' business entities model.

For complex business entities, the `Role` tag-value allows to specify master-detail or lookup relationships, which are very appropriate for code generation techniques, particularly for SQL scripts (in order to guarantee referential integrity among different tables) or user interfaces generation.

Worflows Model (Controller)

The workflows model is useful to define the dasses corresponding to predefined UI workflows. These classes are identified by the «xis-ui-workflow» stereotype and are detailed through equivalent state diagrams. As illustrated in Figure 6, the user interface workflow "SimpleEditor" is detailed through a specific state diagram.

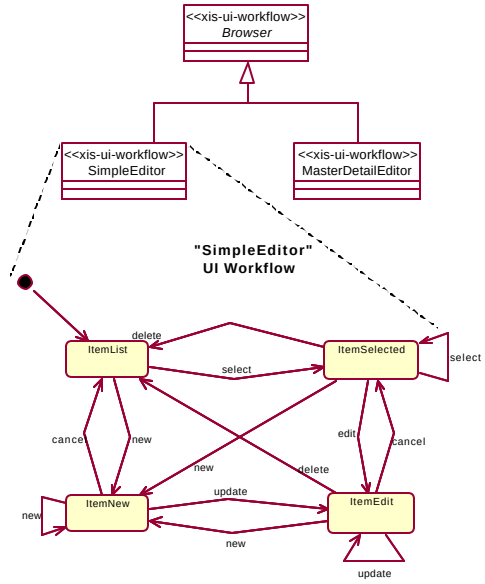

Figure 6. Outline of MyContacts' workflows model. Details of the "SimpleEditor" controller.

Views Model (View)

Finally, as suggested in the Figure 7, in the views model the designer define classes of «xis-ui-view» stereotypes, which is essentially the way to establish associations between business entities and predefined UI workflow.

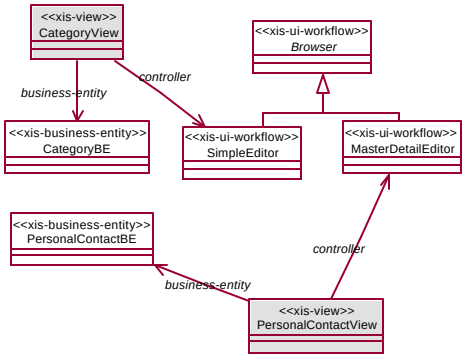

Figure 7. Outline of MyContacts' views model.

In that way, and according the MVC architecture, a business entity (M) can be viewed and managed in different views (V) through different redefined user interface workflows (C).

3.3 XMI Language

XMI (*XML Metadata Interchange*) is the OMG standard for metadata representation [5,w3]. XMI specifies a schema to represent metadata according the UML metamodel. XMI main goal is to provide interoperation of metadata (and in particular of UML models) independently of platforms, languages, repositories and CASE tools.

3.4 XIS/XML Language

XMI has the merit to allow specifying UML models in a XML format, independently of proprietary tools and repositories.

However, XMI has some drawbacks that bring us to design a new language, the XIS/XML language. In particular, it presents the following weaknesses: the size, the fragmentation and the complexity to represent the involved information. (As an example and curiosity, the MyContacts XIS/UML model is specified along 13000 text lines in XMI, while only about 500 lines in XIS/XML language.) Of course, these issues are not big problems by themselves because XMI code is to be handled by computational tools. However, with this additional layer – the XIS/XML, which provides a more structured, understandable and compact format to describe the involved information – we provide at least the following properties: human beings can define or debug easily the information involved on, directly in text/XML format; there is a real independence of the XIS project regarding the CASE tool and or the visual notation used on; and finally, the XMI code is not yet generated commonly and in the same way among the most relevant CASE tools.

4 XIS is based on Generative Programming Techniques

Another principle adopted by the XIS project involves the use of generative programming techniques [11,w6]. This best practice is naturally interconnected with the two others (i.e., based on models specification and component-based architecture-centric), functioning as ”glue” between them. Generative programming techniques transform system specifications into a set of final software artifacts taking into account a given architecture.

An interesting point of the XIS approach is the theoretical possibility to produce code (and other corresponding artifacts) in different programming languages according software architectures, as it is suggested by the MDA philosophy. That issue would be researched and evaluated in future work.

As illustrated previously in Figure 2, the XIS automatic generative process consists in the consecutive application of three transformations, which can be formalized by the next expression:

```
IS = T3(T2(T1(XIS/UML-Model, XIS/UML-Profile), XIS/XSL),
SoftArch)
```

Or separately by:

`XIS/UML-Model` (defined through a CASE-Tool and based on the XIS/UML profile)

```
XMI-Model= T1(XIS/UML-Model, XIS/UML-Profile)

XIS/XML-Model= T2(XMI-Model, XIS/XSL)

IS (FinalSystem)= T3(XIS/XML-Model, SoftArch)
```

Where:

- **`XIS/UML-Model`** corresponds to the original model specified according the XIS/UML profile.
- **`T1`** is the first transformation. T1 transforms the `XIS/UML-Model` into the equivalent model in XMI format. XMI parsers, such as XMIToolkit [w12], available nowadays by the majority of modern CASE tools, support naturally T1 transformation.
- **`T2`** is the second transformation. T2 transforms the model specified in XMI format into an equivalent model in the XIS/XML format. This second transformation is convenient due to the reasons discussed in the Section 3.4. T2 is performed by a XSLT parser and based on a developed XIS/XSL script (in fact a set of XSL files).
- **`T3`** is the third transformation. T3 produces automatically multiple artifacts from the models specified in XIS/XML and taking into account a given software architecture (**`SoftArch`**). A generic code generator, called XISGenerator (which provides some interesting properties such as generality, modularity and flexibility in order to support different software architectures), is the key element of this T3 transformation.
- **`IS`** is the final system and consequently it consists of a suite of multiple and integrated artifacts such as source code, configuration, SQL scripts, HTML, image and documentation files.

It is obvious that there are several important issues concerning the refereed transformations, in particular concerning the powerful and complex T3 transformation, that require a deeper explanation, and should be explained in future papers. Nevertheless, we have to discuss some key issues concerning this T3 transformation, in particular the following issues: iterative and incremental generative

approach; shared repository support; and integration of generated with non-generated code.

Iterative and incremental approach

First of all, a generative approach should be iterative and incremental. A well-known problem of classic generative approaches is the difficulty to delimit their action because they usually adopt "blind" or "brut-force" strategies: if there is a new requirement or feature, "all" the involved artifacts are regenerated without any criteria. For relatively small systems the "brut-force" approach can be feasible, however, an increasing number of systems tend to be large and complex with a disparate number of shared components. Consequently, in these systems, classic generative approaches become promptly impractical. In order to cope with these issues a clever approach is recommended based on the iterative and incremental model. In simple and practical terms, the programmers or architects define their own specific generative processes, which are a set of specific generative steps. Thus, for each generative step, the architects specify the artifacts that would be generated.

Shared Repository Support

On the contrary as it is suggested in Figure 2 (due to simplicity reasons), T3 transformation does not execute directly from the XIS/XML specification. In fact, T3 process is based on information kept in the XIS repository. Consequently, T3 should be better viewed as a pair of two sub-processes. Firstly, a reasonable simple process (T3.1) responsible by populating the XIS repository with the information defined according the XIS/XML specification. Secondly, a complex process (T3.2) that is responsible by the automatic generation of a multitude of software artifacts (such as Java code, JSP, HTML, ant-build, XML and SQL script files) related together to a specific application. This process is of course based on the XIS repository's information and on the previously selected architecture.

The XIS repository is supported by a relational DBMS (actually the MS-SQLServer) with a complex data model in order to manage all the relevant elements, specifically: (1) the XIS metamodel's elements (i.e., the elements defined at the XIS/UML profile or the XIS/XML levels) such as entities, attributes, enumerations, relationships, workflows, transitions, events, or views; (2) elements supporting the generative process itself (i.e., T3.2), such as architectures, contexts, applications, generative steps, generative processes, generative options, or software artifacts.

Integration of Generated with Non-Generated Code

A problem that may arise is the integration of generated with non-generated code. It could not be easy to understand the generated code. Not as much because of code layout, format and style, since any generator can write well-formatted code, but with the dealing of the low-level details of the generated code. However, in most cases, it is possible to use it like a black box framework, which is only used and not read by the programmer.

A few approaches exist in OO scenarios to successfully integrate generated and non-generated code. Generated code can simply call non-generated code contained in libraries, which is an important best practice, since it leads to generating few code as possible and rely on pre-implemented components that are called from the generated code. In another approach, a non-generated framework can call generated parts. To make this approach more practicable, such framework could be implemented against abstract classes or interfaces which the generated code implements.

Another possible approach is to subclass non-generated classes in generated code. These base classes can contain useful generic methods that contribute to compact and restrict the generated code. The base classes can also contain abstract methods that will be implemented by the generated subclasses.

5 XIS is Component-Based Architecture-Centric

The software development process supported by architectures is a well-known best practice, in particular for complex and large systems development [14,15,16,17].

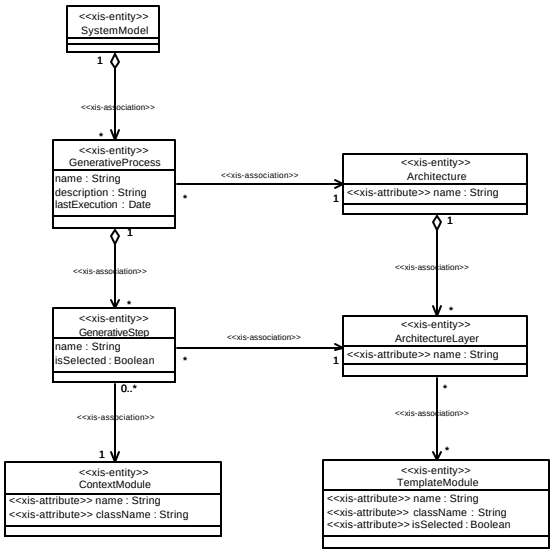

Figure. 8. XIS metamodel regarding architectures and generative processes.

As suggested initially in the Figure 2, it is the architect responsibility to define software architectures (that involves producing different artifacts, such as the

"architecture template" that is used during the T3 transformation), as well as to put and manage them in the XIS platform. On the other hand, it is the programmer responsibility to choose the convenient architecture that should be used by the T3 transformation.

Figure 8 shows how architectures are arranged in the XIS platform: software architecture can be structured around several layers (e.g., data model, business, user interface layers), and each layer can be defined through a set of templates. Templates are software component that generate software artifacts (usually software code) according a specific set of rules and objectives. Hence, templates generate final artifacts according an imperative approach (currently through the Java programming language) and of course using the information of the system designed according the XIS metamodel.

Also from Figure 8 it is evident that distinct *generative processes* can generate a specific *system* iteratively, each one referring distinct *software architecture*.

The XIS platform supports the management of multiple architectures in order to provide productive, flexible and powerful development capabilities. In the current state of the project we have been defining and using three main Java-based architectures, precisely based on J2SE, J2EE e J2ME. (The details of this feature would be further explained in a future paper.)

6 Conclusions and Future Work

The vision subjacent of the XIS project is not new by itself. It is effectively a revisitation of existing expectations that in the past did not have so much success: *the ideia that the building of information systems would be performed almost automatically starting from high-level and platform independent specifications.* Meaning in the end, that classic tasks like programming should be performed almost automatically or at least not so manually intensive as it is done nowadays and in general, with all the well-known costs and problems.

The proposed approach starts to be known increasingly by the expression "software development from models and component-based architectures" and we believe *this time* would be possible to succed in that vision atending the maturity of software enginneering and its respectives technologies. (For instance, according our preliminary analysis, the XIS approach allows the development of information systems with 60 to 80% of generated code compared to the total size of the final systems.) This general approach is being progressively discussed and promoted in different context such as the OMG and in particular its MDA (*Model Driven Architecture*) workgroups [6].

MDA is also increasingly analysed and promoted by software companies. However, there are some subtle differences between the MDA and the XIS approach that deserve to be discussed accordingly. First, MDA is just a high-level reference model or philosophy while XIS is a concrete project with a proposal of an approach, tools and specific languages (i.e., XIS/UML and XIS/XML). Second, MDA recommends the definition of two kinds of models – platform independent models (PIMs) and platform specific models (PSMs) –, as well as respective PIM2PIM, PIM2PSM and PSM2PSM mappings. On the other hand, in the XIS approach there are just PIM models, which are high-level and platform independent specifications of information systems following the MVC architecture and represented through the XIS/UML profile or directly through the XIS/XML language. The mapping from these XIS models to a specific platform and programming language is provided directly by architects when they define software architecture templates (as illustrated in Figure 2), and finally by programmers when they select the models and the architecture templates need to start T3 transformations.

Figure 9 summarizes the XIS approach, being evident its principles and their relationships in the MDA terminology. For instance, the PIM2PSM mappings are handled by the generative techniques, and involves high-level models (PIMs) as well as component-based software architectures (PSMs).

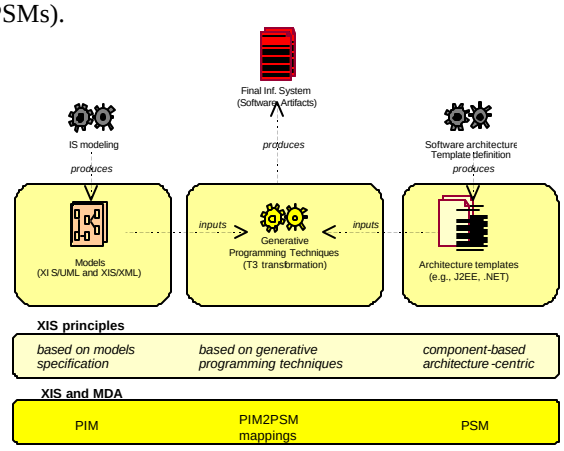

Figure 9. XIS and MDA relationships.

It should be stressed that **XIS is not a conceptual research plan, it is a working on project with concrete results and produced systems** (The interested reader is invited to contact the author or visit the project web site [w11] for more information). The XIS project started on September of 2001 with the goal to serve gradually as a strategical context, or "umbrella", to host several research projects that will be carried out during the next period of 2-4 years. Actually, in this context, two projects (a MSc thesis and a final year project) finished recently, and other projects (PhD, MSc and final year projects) are starting with the mission to deep and improve the XIS project's goals introduced in this paper. On the other hand, we plan

to use the XIS project results to help the development of concrete systems in order to get positive feedback and enhance it following an iterative and incremental approach.

There is a set of open issues that would be handled and researched in a near/medium term, specially: variation

- At present, the project have been focused in the development of Web-based distributed and large-scale information systems on the top of Java frameworks such as J2SE, J2EE e J2ME. Obviously a relevant work would be the support of other architectures, such as Java variations (e.g., based on Struts [w9] or OpenSymphony [w10] MVC's frameworks) or on Microsoft's .NET platforms.
- To propose understandable and efficient mechanisms to specify UI views, meaning human-machine interactions specified visually, abstractly and independently of specific platforms or architectures. The analysis of UIML language [10,w5] would be a reasonable starting point to achieve that work.
- To propose understandable and efficient mechanisms to specify applicational views, meaning views and workflows to support comunication among a local (i.e., intra-organization) or large-scale (i.e., inter-organizations) set of applications. The analysis of web services and workflow technologies [19,20,w7, w8] would the starting point to carry out that research.
- To enhance the features provided by the XIS platform, allowing eventually its integratation with other ones related to requirement management, project management, and other common tasks such as iterative generation, test and integration of software components.

Acknowledgments

My thanks to the students Gonçalo Lemos, Tiago Matias and Miguel Luz for their interest, relevant contributions and the development of the first XIS prototypes.

References

[1] G. Booch, J. Rumbaugh, I. Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, 1999.

[2] P. Kruchten. *The Rational Unified Process* , 2nd edition. Addison Wesley, 2000.

[3] OMG. *UML Resource Page.* http://www.omg.org/uml/

[4] OMG. *Meta-Object Facility (MOF™) Version1.4,* OMG UML working Group, http://www.omg.org/cgi-bin/doc?formal/2002-04-03

[5] OMG. *XML Metadata Interchange (XMI) Version1.2,* http://www.omg.org/cgi-bin/doc?formal/2002-01-01

[6] OMG. *Model Driven Architecture.* http://www.omg.org/mda/

[7] OMG. *"White Paper on the Profile mechanism", Version 1.0, OMG Document ad/99-04-07.* OMG UML Working Group.

[8] W3C. *Extensible Markup Language (XML) 1.0*. W3C XML Working Group. http://www.w3.org/TR/REC-xml

[9] G. Grasner, S. Pope. "A cookbook for using the model-view-controller user interface paradigm in Smalltalk-80". *Journal of Object-Oriented Programming*, 1 (3), 1988.

[10] UIML.org. *UIML v2.0 Draft Specification* , 2000.

[11] J. Craig Cleaveland. *Program Generators with XML and JAVA.* Prentice-Hall, 2001.

[12] W. J. Orlikowski. *CASE tools as Organizational Change: Investigating incremental and radical changes in systems development* . MIS Quarterly, vol. 17, no. 3, September 1993.

[13] I. Vessey, S. L. Jarvenpaa, N. Tractinsky. *Evaluation of Vendor Products: CASE Tools as Methodology Companions*. Communications of the ACM, vol. 38, no. 1, January 1995.

[14] E. Gamma, R. Helm, R. Johnson, J. Vlissides. *Design Patterns – Elements of Reusable Object-Oriented Software.* Addison Wesley, 1994.

[15] C. Hofmeister, R. Nord, D. Soni. *Applied Software Architecture.* Addison Wesley, 1999.

[16] *The Software Patterns Series*. Addison Wesley, 1996-2002.

[17] M. Juric, et al. *J2EE Design Patterns Applied*. Wrox Press, 2002.

[18] C. McClure. "The CASE Experience", *Byte*, April 1989.

[19] E. Christensen et al., W3C, *Web Services Description Language (WSDL)*, 2001, http://www.w3.org/TR/wsdl

[20] F. Curbera et al. *Business Process Execution Language for Web Services*, Version 1.0, July 2002. http://www-106.ibm.com/developerworks/ webservices/library/ws-bpel/

Electronic-Web References

[w1] Software Engineering Institute, http://www.sei.cmu.edu

[w2] World Wide Web Consortium (W3C), http://www.w3c.org

[w3] Object Management Group (OMG), http://www.omg.org

[w4] XML.com, http://www.xml.com

[w5] UIML.org, http://www.uiml.org/index.php

[w6] Generative Programming Group, http://www.generative-programming.org

[w7] W3C, *Web Services Activity*, 2002, http://www.w3.org/2002/ws/

[w8] WebServices.Org, *The Web Services Community Portal*, 2002, http://www.webservices.org

[w9] Structs, Apache Jakarta Project. http://jakarta.apache.org/struts/index.html

[w10] WebWorks, OpenSymphony Group. http://www.opensymphony.com/

[w11] INESC-ID's Information System Group. *The XIS Project*. http://berlin.inesc-id.pt/projects/xis/

[w12] IBM AlphaWorks, *XMIToolkit*. http://www.alphaworks.ibm.com/tech/xmitoolkit/