A Methodology to Design Information Retrieval Systems (MDIRS)

João Ferreira,
ISEL, Lisbon, Portugal
jferreira@deetc.isel.ipl.pt,

Alberto Silva
Inesc-ID, IST, Lisbon, Portugal
alberto.silva@acm.org

José Delgado
IST, Lisbon, Portugal
Jose.Delgado@tagus.ist.utl.pt

Abstract

MDIRS is methodology to define the actors and the steps to build efficiently information retrieval (IR) System. MDRIS main mission is to analyze, develop and evaluate mechanisms and tools to produce IR systems from a more abstract, high level, efficient and productive way than it is done currently. MDIRS project is influenced by MDA reference model, and is mainly based on three principles: namely, high-level models specification; generative programming techniques; and it is component-based architecture-centric. In this paper we detail the methodology generative programming techniques used to produce IR Systems 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 MDIRS 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. The approach and tools presented in this paper is mainly conducted by our own experience of mentoring, designing and developing IR 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 MDIRS 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. The MDIRS intends to apply different techniques and mechanisms in order to accelerate and to improve the IR systems development. To reach such goals the methodology the following elements: (1) *MDIRS platform* is a CASE tool to support the technical actors of the software development process according the MDIR approach. The MDIRS 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 MDIRS platform is a web based, multi-user, multi-application and multi architecture tool; (2) *IRML/UML profile* is a set of coherent UML extensions that allows a high-level, visual modeling way to design information systems. This models acts, as standards libraries; (3) *MDIRS/XML language* is a XML language that provides a textual, structured, understandable and compact way o specify information systems. While MDIRS/UML profile allows the definition of information systems with visual models, XIS/XML [1] is its textual counterpart that allows the definition of information systems with XML-based, textual specifications.

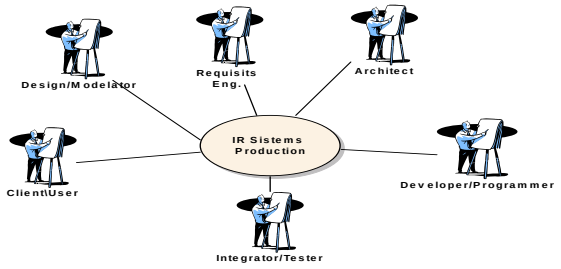

Fig. 1. Main actors on the production of IR systems

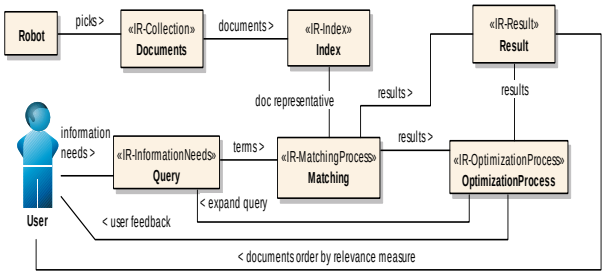

Fig. 2. MainIR system modules.

2. Information Retrieval Systems

IR systems have the mission to recover user's information when expressed in appropriate queries or profiles. The system returns a list of documents order by subjects' relevance according to the user information needs. Google is the most successful IR system. Apart of the high complexity involved, all IR systems share the main principals and conceptual modules, like (Figure 2): (1) a robot to pick up information; (2) an index process to reduce the size of documents' collections; (3) a query interface module that allows the users to input their information needs and prepare queries for the matching process; (4) a matching process that compares document representatives with the representatives of users' information needs; (5) an optimization process to improve results and (6) a presentation module. In spite of being the same in conceptual development these modules are always constructed in a different way and is difficult to use common code. Due to this fact, there is a lack of personalized IR systems and a lack of scientific work on this subject. IR research community is strongly involved on finding new methods for matching, optimizing and modeling users' behavior but no relevant work related with modeling IR systems. There is a similar approach to the website construction [2,3,4,5], but that can not be considered IR systems modeling. Starting from this point makes sense to develop IR systems through a methodology that allows the re-use of software modules and increases the modularity facilitated through automatic code generation.

3. MDIRS Approach's Main Tasks

Figure 3 gives the big picture of the methodology proposed based on tasks and actors. Broadly, MDIRS 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 started by identifying the tasks performed by the **Architect**, which are critical to the correct operation for the MDIRS approach. Specifically, the architect should be responsible by the following tasks: (1) define a specific language for IR area, IRML [6]; (2) define IR abstracts models; (3) define and select the IR Infrastructure (4) define software architecture templates, in order to support the creation of generation templates. **Requisites Engineer** is responsible to identify the requirements, identified for instance from meetings among clients and end-users and produces the requisites specification that will guide the process.

Designer's responsibility to produce a conceptual IR system based on three views: (1) the use case view; (2) the information view; (3) the process view. All these views are based on the language defined based on UML. **Developer (Programmer)** produces the IR system (almost finished) via MDA approach [7]. Suitable and corresponding information system model (the "IR 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 nonfunctional 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 [8,9]. These activities are suggested in the Figure 3 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 3 by the "Prepares, Perform Tests and integrate the IR system" task.

4. MDIRS based on Generative Programming Techniques

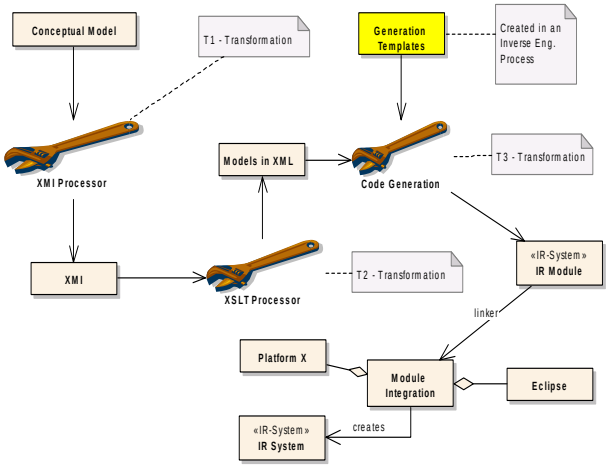

Fig. 3. Code generation from models with appropriate templates.

Another principle adopted involves the programming techniques [10,11]. 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. The generative programming techniques transform system specifications into a set of final software artifacts taking into account a given architecture. The generation's process is described on Figure 4 and has three main transformations: (1) T1 transforms conceptual model into XMI format. Several tools perform this transformation; (2) T2 transforms XMI into XML format based on scripts developed; (3) T3 produces automatically several artifacts from model specified in XML and with appropriate templates created in an inverse engineering process, we generate the modules of the IR system, which are linked the Eclipse Platform [12], a well-known and stable platform with widespread support in the Java community. Although commonly known to software developer community as an open-source IDE (Integrated Development Environment) for Java software development, Eclipse should be best described as "an open universal framework for building developer tools" [12]. Consequently, **programmers** are involved to produce specific components, typically helper source code, such as facades, adapters, controllers and business logic.

5. CASE STUDY, MYTV

We have been developing several examples to show the advantages to develop IR systems by using the proposed methodology [13]: (1) MyEnterpriseNews, a system to find relevant news about a subject; (2) MynewsPaper, a personalized news paper; (3) MyDocument, a system that organizes documents in a department or company using a local hierarchical knowledge space. In all these systems have common modules and we, use the same user interfaces modules. In this paper we concentrate on a filtering system of TV programs (**MyTV**) applied to two worlds, Text and Image: (1) **MyTV programming guide,** is a personalized notification system about TV programs (in this case, broadcasted by the Portuguese cable operator TvCabo <www.tvcabo.pt>) to registered users. The User profile is based on a list of programs; (2) **MyTV Personalized Television**, is a system to help users to minimize the zapping problem high number of channels available, or reading the programming guides in advance, but this imposes effort on the user. To avoid this, we propose a system that is able to send and alert (based on a user profile) on a small window regarding interesting programs broadcasted during the period that the television is switched on. These systems have the challenge of identifying video programs automatically and to match them against the user profile. This is a new system that we will create by using parts of an existing system, demonstrating the potential of this new approach.
Main differences of both systems are on information representatives. In one system MyTy programming guide we have text information related to Tv programs and on the other MyTV personalized television we work with low-level properties of video identified by the algorithms: (1) shot boundary detection, (2) GofGopColor, descriptor; (3) EdgeHistogram, descriptor; (4) MotionActivity, descriptor. From these algorithms, results follow low level characteristics using MPEG-7 in a video segment: (1) Color, GofGopColor, calculates color histogram in a space HSV; (2) EdgeHistogram calculates edges for 16 zones; (3) MotionActivity, from 1 to 5 measures of movement intensity; (4) density of cuts. To convert this high level information space (text) to how level (image descriptors) we introduce converters from high to low level (and vice versa) are run from a test collection of programs by an estimation process. Tuning these processes is a complex task. For example, a football game has green as predominant color, global movements with direction change, movement in low speed (repetitions).

6. CONCLUSIONS AND FUTURE WORK

We propose a new methodology to develop IR systems using a novel model-based approach as well as generative programming techniques. The vision underlying the methodology is not new by itself. It is effectively a revisitation of existing expectations that in the past did not have so much success: *the idea that the building of information systems would be performed almost automatically starting from high-level and platform independent specifications*. Meaning that, in the end, classic tasks like programming would be performed almost automatically. Applying the proposed methodology to the development of IR systems bring us the following benefits: (1) increase the widespread of personalized IR systems, because it would be easier to develop these kind of systems if there are available a certain number of models and involved templates; (2) re-use of models and templates at high-level in order to realize this potential; (3) promote the collaboration among different research groups; and (4) better productivity, which means less time and low cost spent in development.
An interesting point of this approach is the theoretical possibility to produce code in different programming languages, according a set of software architectures and frameworks (in this context, according different IR platforms), as it is suggested by the MDA philosophy

[14]. That issue will be evaluated and researched in future work.

Through the proposed approach, re-use of IR programming modules will be much easier, and personalization of an IR system will be faster because with this framework developers need to change or to create specific modules that will work on top of other common IR modules. Automatic code generation allows the creation of a novel generation of IR CASE tools for the construction of specific (personalized) IR applications, supporting advanced features such as connection devices, multimedia and geographic location. Obviously, these are very ambitious ideas, that can be detailed and focused in a medium turn research agenda and they are the application of XIS project [1] to IR.

References

1. http://berlin.inesc-id.pt/projects/xis/
2. F. Garzotto, P. Paolini, D. Schwabe HDM - A Model-Based Approach to Hypertext Application Design ACM Transactions on Information Systems, Vol 11, No 1, pp. 1-26, 1993
3. D. Schwabe, G. Rossi. The Object-Oriented Hypermedia Design Model. Communications of the ACM, Vol 38, No 8, 1995.
4. K. Takahashi, E. Liang, Analysis and Design of Web-based Information Systems. Sixth WWW Conference, 1997. http://www6.nttlabs.com/papers/PAPER245/Paper245.html
5. M. Bichler, S. Nusser. W3DT - The Structural Way of Developing WWW-sites, Proceedings of ECIS'96, 1996
6. J.Ferreira, A. Silva J. Delgado. IRML – Information Retrieval Modeling Language. Published at 6th IASTED International in Modeling, Simulation and Optimization. Garbone, 11-13 September 2006.
7. MDA Guide V1.0.1. Available at http://www.omg.org/cgi-bin/doc?omg/03-06-01.
8. E. Gamma, R. Helm, R. Johnson, J. Vlissides. DesignPatterns – Elements of Reusable Object-Oriented Software. Addison Wesley, 1994.
9. The Software Patterns Series. Addison Wesley, 1996-2002.
10. J. C. Cleaveland. Program Generators with XML and Java. Prentice-Hall, 2001.
11. Generative Programming Group. http://www.generative-programming.org.
12. J. S. Saraiva, Alberto Rodrigues da Silva, Eclipse.NET: An Integration Platform for ProjectIT-Studio, in Proceedings of the 1st International Conference of Innovative Views of .NET Technologies (IVNET05), (Portugal, Porto, June 2005) , ISEP and Microsoft .
13. Shah R. working the Eclipse Platform, http://www-106.ibm.com/developerworks/opensource/library/os-eclipse, March 2003.

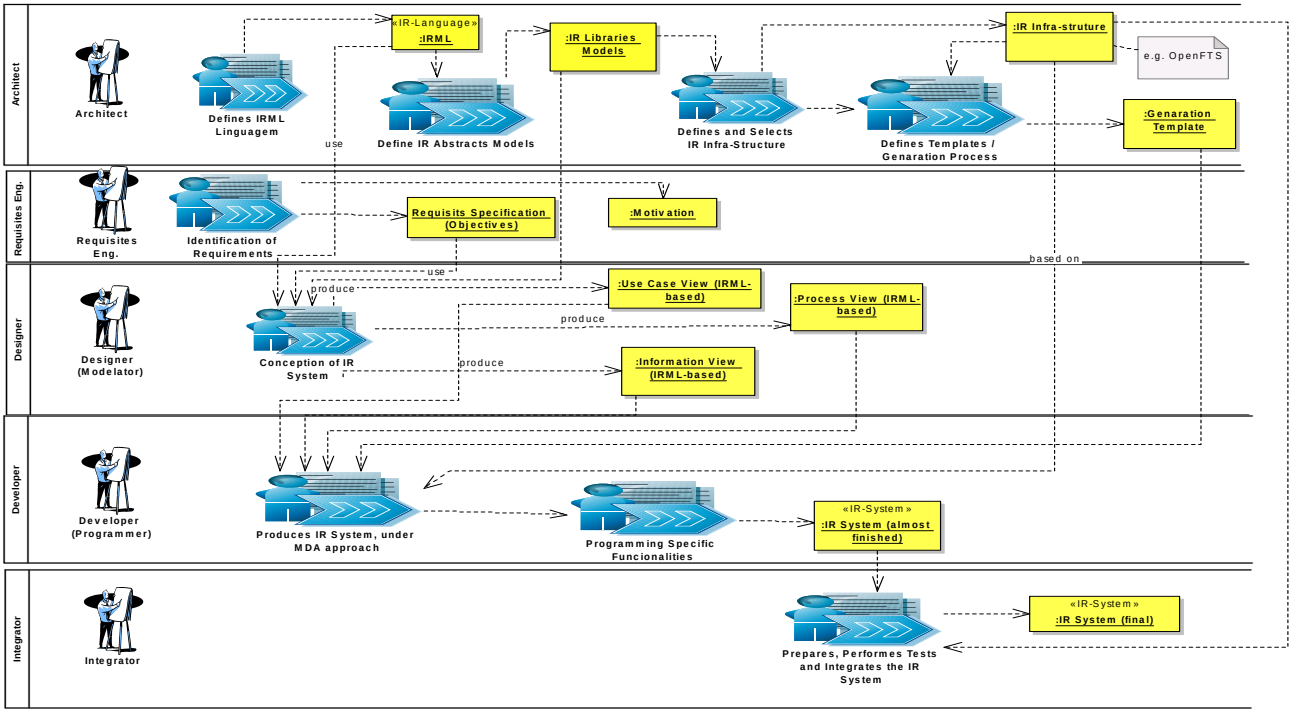

Fig. 4. Key elements of the MDIRS approach