IR-Case Tool

João Ferreira[1], Alberto Silva[2], and José Delgado[3]

[1] *ISEL*, [2] *INESC-ID*, [2,3]*IST*,
[1]jferreira@deetc.isel.ipl.pt
[2]alberto.silva@acm.org
[3]Jose.Delgado@tagus.ist.utl.pt

ABSTRACT

We propose a new approach based on a methodology assisted by a IR-Case tool for the creation of IR (Information Retrieval) systems inspired on a set of best practices or principles: it is based on high-level models or specifications; it is component-based architecture centric; it is based on generative programming techniques. This approach follows in essence the MDA (Model Driven Architecture) philosophy with some specific characteristics. We propose 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). Generically, this methodology 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, that will be linked in the IR-Case tool proposed, generating the IR-System. These aspects are implemented in a tool (IR-Case tool), providing a roadmap where designers can follow as well as model-to-model transformation templates in order to accelerate their system development tasks. This step facilitates the construction and consequently will contribute for the personalized IR-Systems and also a test platform for IR-Algorithms and IR-Process.

Keywords

UML, MDA, Retrieval Information, Case Tool.

1. INTRODUCTION

Information Retrieval (IR) has been developed during the last four decades based on algorithms and methods. The development of IR systems is a complex process, usually performed by groups of IR researchers or commercial companies. Usually the creation of IR systems is not a collaborative effort among groups in spite of the existence of commons modules among the IR systems. These efforts highlight a lack of availability of specific (Personalized) IR systems, because it is a complex task. At the beginning of this decade, some modular IR platforms have been done [1,2,3] but these modules are still too large to allow flexibility. On the other hand, if we have smaller modules, the new appearing problem would be how to assemble them together, like a LEGO construction without instructions. To avoid these problems, some IR systems [1,2] give some flexibility through the availability of several options in a predefined API. This scenario leads us to a new approach regarding IR systems' construction based on models, that integrates the best practices and fundaments around the Model Driven Architecture (MDA) paradigm and specification of requirements, such as modularization, separation of concerns, reutilization, use-case driven, model-to-model and model-to-code transformations [4,5,6]. This approach has been applied in software engineering areas [7,8,9,10], and the most related works were on personalization of websites using modeling methods [11,12,13,14], due to the diversity of personalization policies over the development cycle of websites. We intend to go further steps ahead integrating on this process the creation of IR systems. This approach makes sense due to the use of common modules parts in different IR systems and also due the diversity of retrieval approaches. There is a big number of different IR systems [15,16,17,18] always constructed from zero. In spite all the diversity of systems, they relay on the same principals, use statistic proprieties of documents, creation of documents' representatives with smaller dimensions, interfaces that users allow to perform queries, expand, feedback, matching and optimization methods. On the other hand, from the MDA/MDE perspective it is important to stress the following references: (1) MDA [19] is the OMG's framework for software development life cycle, driven by the activity of modeling [20], which makes models first-class entities; (2) MOF [21] is the foundation of OMG's approach, by supporting model exchange and transformations which can be applied to any modeling language, as long as it is MOF-based; (3) UML [22] is a general-purpose modeling language, originally designed to specify, visualize, construct and document information systems (nevertheless, UML is not restricted to modeling software and is often used for other purposes like business process modeling); (4) the XMI format [23] is a standard commonly used to exchange UML models between tools, although it can also be used with other MOF-based metamodels; (5) the MOF QVT [24] is another standard, still under finalization, for defining query, view and transformation operations on MOF-based models, effectively allowing the transformation of source models into target models, which makes QVT a critical component of MDA. Today there is a couple of commercial tools such as Telelogic's DOORS and TAU G2, and IBM's RequisitePro, Software Modeler, and Software Architect already present a mature and deep integration, enabling engineers to enforce traceability between requirements, models, and source code, through a common environment, but none of these tools are prepared to constructed IR-Systems or to taken in account specifies of IR-systems.

2. Methodology

2.1 Main Actors

We propose a new methodology for the creation of IR systems inspired on a set of best practices or principles: it is based on high-level models or specifications; it is component-based architecture centric; it is based on generative programming techniques. This approach follows in essence the MDA philosophy with some specific characteristics. We propose 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). Figure 1 overviews the methodology, in particular the main actors and corresponding tasks. Generically, this methodology 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. The tasks performed by the software architect are critical to the process of creation of an IR system.

The architect is responsible by the following tasks: (1) to define a suitable and easy-to-use **UML profile (IRML)**; (2) based on IRML, to define **abstract models**; (3) to define and to select an IR infra-structure to support the IR system; (4) to **develop templates**: (i) to model transformation features, such as “Model2Model Transformation Templates”, (ii) to produce new models, (iii) to develop model-to-code template features, such as “Model2Code Transformation Templates”, and (iv) to produce software and documentation artifacts from models, using generative programming techniques. Starting from system requirements (created for instance from meetings, interviews, JAD sessions among designers, clients, end-users and other stakeholders) [25]. The Requisites Engineer collects objectives and identifies the motivation to create the system. The Designer is responsible for the design of the IR system through available models, producing an integrated set of models (the “Design System” task). Still, the designers can apply model transformations automatically according the “Model2Model Transformation Templates” developed previously by architects. This task can be useful in certain situations in order to simplify or to accelerate the design task. The correctness and quality of the models produced are essential to obtain good results in the subsequent tasks. After the design intervention, programmers apply model-to-code transformations, which means to apply generative code techniques to models, based on templates provided by the architects. Because it is not possible to capture and to design all the system requirements, programmer intervention is still required The Developer creates systems via the MDA approach proposed (see Figure 2). Conceptual models are transformed into XMI (T1) and simplified in XML (T2). From these XML models, and with appropriate templates created in an inverse engineering process, we generate the modules of the IR system, which are linked the Eclipse Platform, 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” [26]. Consequently, programmers are involved to produce specific components, typically helper source code, such as facades, adapters, controllers and business logic.

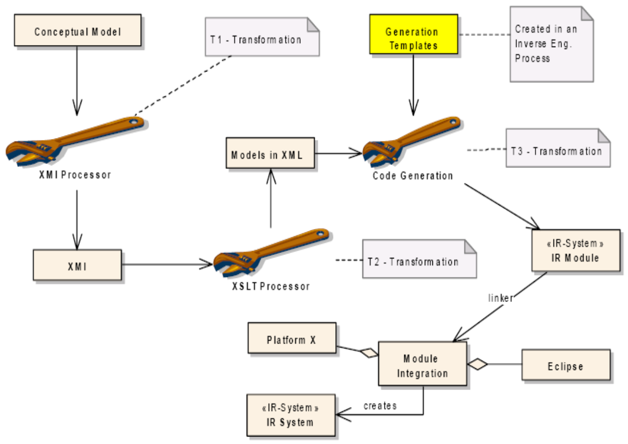

Figure 2: Code generation from models with appropriate templates.

2.2 IRML

Based on UML we propose a meta-model for IR (IRML), based on new a stereotype specific for IR [27]. **IRML** is a set of coherent UML extensions (Figure 4, is a proposal) that allow a high-level, visual modeling way to design interactive systems. The main idea is to have a standard language UML which could act as: (1) standardization and systematization of IR concepts and notation; (2) establishment of rules and guides for code generation through appropriate templates, which transform XML system models in written code expressed in a predefined chosen language. This templates are constructed based on the knowledge of final code on an inverse engineering process; (3) development of IR-systems (construction) based on models; (4) central database could store models and templates; (5) common interface between modules. To simplify the design of IR-Systems we propose three views. These views act like the different views in an architectural project of a house and simplify the process by dividing the problem in smaller pieces. The views are chosen between a compromise of simplicity (more views) and complexity (less views). We observed that these three views simplify the process of construction of IR-System and at the same time are enough to describe the problem:
(1) The **IR-UseCaseView** defines IR-Actors and their actions on the system. In this view the external relations of the system and the main objectives are defined: (2) **IR-InformationView** defines the system data input and output. In this view, the data flow is shown using class diagrams; (3) **IR-ProcessView**, in which attributes and sequence of actions to transform input into output are defined according to the proposed objectives.

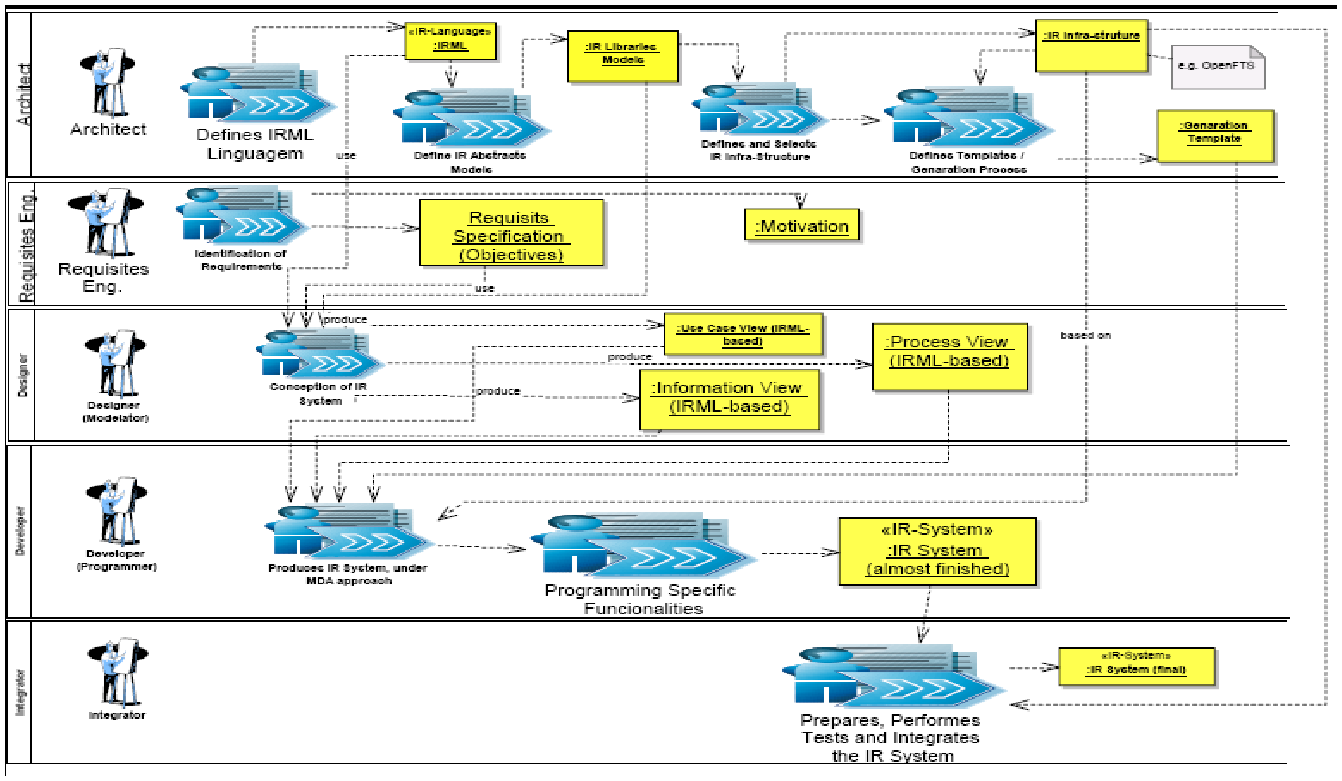
Architect
Architect
Defines IRML Linguagem
«IR-Language» :IRML
use
Define IR Abstracts Models
:IR Libraries Models
Defines and Selects IR Infra-Structure
:IR Infra-struture
e.g. OpenFTS
Defines Templates / Genaration Process
:Genaration Template
Requisites Eng.
Requisites Eng.
Identification of Requirements
Requisits Specification (Objectives)
:Motivation
use
Designer
Designer (Modelator)
produce
Conception of IR System
:Use Case View (IRML-based)
produce
:Process View (IRML-based)
produce
:Information View (IRML-based)
based on
Developer
Developer (Programmer)
Produces IR System, under MDA approach
Programming Specific Funcionalities
«IR-System» :IR System (almost finished)
Integrator
Integrator
Prepares, Performes Tests and Integrates the IR System
«IR-System» :IR System (final)

Figure 1: Methodology for the creation of IR systems.

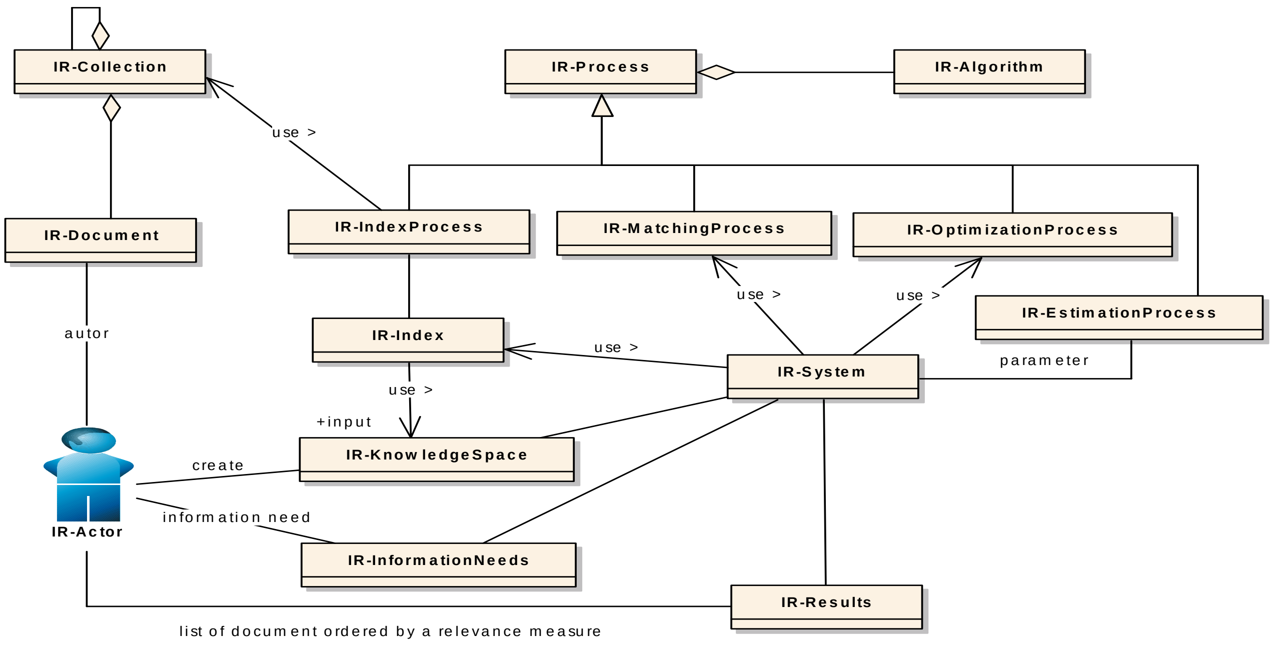

Figure 3: Metamodel for IRML.

2.3 Models and Templates creation

Basic and specific IR models were created through a modeling tool for UML (e.g. Enterprise Architecture) and converted automatically into XML format. Models are created in high level follow IRML language in different levels of complexity storage in a central place for future use. This action is performed on a UML tool (Enterprise Architecture 5.0) which had capacity of produce the XMI (XML Metadata Interchange) code from the model identified. The XMI is a standard of OMG for data exchange between platforms. Also the UML and the XMI are the base for MDA (Model Driven Architecture) which goal (Figure 2) is to define a set of rules and techniques used to modify one model in order to get a different new model. Mappings are used to transform PIM (Platform Independent Model) into PSM (Platform Specific Model) and the opposite. The UML profiles play an important role in MDA, since MDA leverage usage between models. There are several techniques to perform this transformation and generate automatic code. Templates are a pre-build form, having as input an simplified XML of each model, and the output is the source code of that model. To each model, it is necessary a template for a specific platform. This template is being developed into one of the three platforms of IR (OpenFts, Terrier and Lemur). These templates will be created using an engineer inverse process performed from the final code that we want to reach. Models and templates were stored in a central database for future use. This is a first step and on a second phase, we intend to create templates independent of any platform and provide a variety of different granular modules that allow the users to easily build an IR system. Models and templates available are stored in a central database for future reuse. Developers can contribute with templates, models, and users with available models and templates to create appropriate IR Systems. The proposed approach allows reusing knowledge captured in previous projects and IR-System constructed by using "applicational templates". This allows accelerating the process by employing the concept of develop-by-reuse [28] and minimizing error introduction through a constant quality improvement process of the captured expertise. This same approach, but according to another perspective, can be applied having as its main goal the creation of a templates repository, thus supporting the concept of develop-for-reuse [28], whose main objective is to define a requirements documents templates catalog/library. These later templates, "architectural templates", are based on re-factored requirements, hence improving their range of possible application, assuring simultaneously that they incorporate the best practices of requirements documents and use cases writing patterns [29].

2.4 IR-Case Tool

IR-Case tool is an integrated environment that supports the central tasks of the IR software development, mainly: requirements specification, architecture definition and system design. The IR-Case tool is built on top of the Eclipse.NET platform [30], using knowledge of ProjectIT performed at INESC-ID's Information System Group [31, 32], which provides an extensible plugin-based architecture framework for developing other tools. Figure 4 presents the high-level IR-Case tool components architecture and Figure 2 shows the steps to transform models in modules (small pieces of programming code) and link then to build a IR-System.

Main tasks to be performed are: (1) **Components (IR-Module), c**onceptual models are transformed into XMI (T1) and simplified in XML (T2). From these XML models, and with appropriate templates created in an inverse engineering process, we generate the modules of the IR system, which are linked the Eclipse Platform, a well-known and stable platform with widespread support in the Java community; (2) **Integration**, the components generated from models through templates and the platform components used should be integrated into a program (IR-System). For this process we will use the Eclipse Platform that is a well-known and stable platform, commonly known as an open-source IDE (Integrated Development Environment). This integration platform provides the means that allow these components to communicate efficiently and release the IR-system. Code duplication bug will have less probability of appearance; (3) **Platform**, this approach allows the creation of IR systems by the integration of different modules on a specific program language (C#, Java, etc) or in different IR-platforms (e.g. Lemur, Terrier, Okapi, Smart, etc).

2.5 Modularity

This methodology and IR-Case tool make sense if we make available small pieces, which allow the diversity construction of IR-System like a LEGO construction through the availability of small modules and templates. We start first templates to big modules (e.g main modules of Lemur and Terrier IR systems). Also in future we intend: (1) increasing models and templates available at central system database; (2) creation of templates for different programming language (e.g. C#, Java, …); (3) improve existing templates and models.

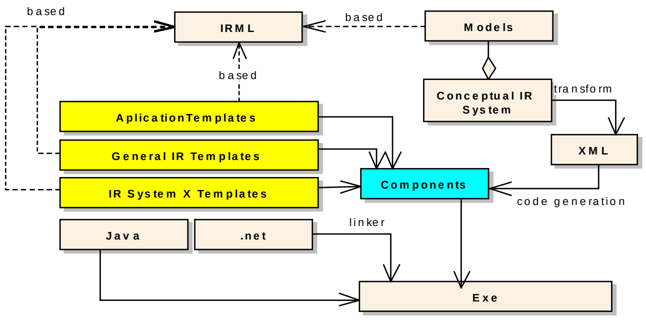

Figure 4: Top-vision of the model-based approach for IR systems development.

3. WebSearchTester: Test Platform for IR-Systems

To show the advantages the construction of IR systems by using the proposed IR-Case tool we start building a test IR system called WebSearchTester, (for details see [33]), to test advantage of IR-Case tool on the construction of different IR systems using common modules. A framework is a program skeleton that defines the basic concepts of an application domain usually developed to support a whole set of related applications in contrast to developing individual applications from scratch. In the IR domain, a useful IR framework should provide the option of applying different models for indexing and retrieving information (e.g. the probabilistic or the vector space model). The main purposes of the WebSearchTester framework are: (1) create a common platform that support the following IR models: Boolean, Vectorial, Document generation (classical probabilistic, implemented Okapi measure), Query generation (Languages model), Logistic Regression (LR), Inference network, Concept space model, Probabilistic distribution, Link analyses, KL divergence, Markov chain, Fusion of results from different models and also Classification. Details of this implementation can be found at [34]; (2) easy way to test and create IR algorithms and process, collaborative environment adequate to the exchange of ideas; (3) make available better and comparable evaluation of different models based on a common infrastructure; (4) promote research in a collaborative environment; (5) propose a common framework for IR, filtering and classification applications; (6) support personalization analysis: can be used to rank IR results in terms of users' preferences.

4. Conclusions

We proposed a new way of construction IR systems based on models and automatic code generation. The potential behind this approach is enormous, providing tools for collaboration increase between groups and construct IR system that can easily be changed. Models and templates could be stored and organized in a central database to be used in a distributed and universal way. This tool could also be used to build a central and common test system where IR investigators could test their ideas and new IR algorithms without big efforts and in a standard way.

5. REFERENCES

[1] Lemur <www.lemurproject.org/>

[2] Terrier <ir.dcs.gla.ac.uk/terrier/>

[3] Okapi < http://www.soi.city.ac.uk/~andym/OKAPI-PACK/>

[4] OMG\ MDA. <http://www.omg.org/mda/>

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

[6] Kotonya, G., Sommerville, I., Requirements Engineering Processes and Techniques, NY. Jonh Wiley & Sons, 1998.

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

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

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

[10] M. Juric, et al. J2EE Design Patterns Applied. Wrox. Press.

[11] Koch, N., Kraus, A. and Hennicker, R. (2001). The Authoring Process of the UML-based Web.Engineering Approach. In Proceedings of the 1st International Workshop on Web-Oriented Software Technology.

[12] De Troyer, O. and Leune, C (1998). WSDM: A User-Centered Design Method for Web Sites. In Computer Networks and ISDN systems Volume 30, Proceedings of the 7th International WWW Conference, pages 85-94, Elsevier.

[13] Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S. and Matera, M.(2002). Designing Data-Intensive Web Applications. Morgan Kaufmann Publishers Inc..

[14] Frasincar, F., Houben, G.-J. and Vdovjak R. (2002). Specification framework for engineering adaptive web applications. 11th WWW Conf., Web Engineering Track.

[15] <http://bit.csc.lsu.edu/~kraft/retrieval.html>

[16] <http://www.dcs.gla.ac.uk/idom/ir_resources/ir_sys/>

[17] <http://www2.sims.berkeley.edu/resources/collab/>

[18] <http://www.glue.umd.edu/~dlrg/filter/software.html>

[19] Object Management Group, (2003) "MDA Guide v 1.0.1".

[20] Kleppe, A., Warmer, J., Bast, W., (2003) "MDA Explained: Architecture: Practice and Promise", Addison-Wesley, 2003.

[21] Object Management Group, (2006) "Meta Object Facility (MOF) Core Specification", January 2006.

[22] Object Management Group, "Unified Modeling Language: Superstructure–Specification Version2.0", August 2005.

[23] Object Management Group, (2005) "MOF 2.0/XMI Mapping Specification, v2.1", September 2005. http://www.omg.org/cgi-bin/apps/doc?formal/05-09-01.pdf

[24] Object Management Group, (2005) "MOF QVT Final Adopted Specification", November 2005. http://www.omg.org/cgi-bin/apps/doc?ptc/05-11-01.pdf

[25] Kotonya, G., Sommerville, I., Requirements Engineering Processes and Techniques, NY. Jonh Wiley & Sons, 1998.

[26] Shah R. working the Eclipse Platform, http://www-106.ibm.com/developerworks/opensource/library/os-eclipse.

[27] Ferreira, João; Silva, Alberto; Delgado, José (2006). IRML - Information Retrieval Modeling Language. Proceedings of MSO 2006, 6th IASTED, September, Gaborone, Botswana.

[28] Cybulski, J., "Application of Software Reuse Methods to Requirements Elicitation from Informal Requirements Texts", PhD Thesis, La Trobe University, Australia, March 2001.

[29] Examples of requirements document's templates: http://www2.ics.hawaii.edu/~johnson/413/lectures/5.2.html, http://www.devdaily.com/java/misc/ReEnableExternalUser/node1.shtml>

[30] http://www.sourceforge.net/projects/eclipsedotnet

[31] Saraiva, J., Silva, A. , (2005) "Eclipse.NET: An Integration Platform for ProjectIT-Studio", Proceedings of the 1st Int. Conf. on Innovative Views on .NET Technology, (Porto, Portugal, June2005), ISEP and Microsoft.

[32] Saraiva, J., (2005) "Desenvolvimento Automático de Sistemas – Relatório Final de Trabalho Final de Curso".

[33] <http://www.deetc.isel.ipl.pt/matematica/jf/>

[34] Ferreira, João; Silva, Alberto; Delgado, José (2005). A modular platform applicable to all statistical retrieval models, Proceedings of the ITA05, September, in Wrexham, Wales.