# The ProjectIT-RSL Language Overview

Carlos Videira [1], João Leonardo Carmo[2], Alberto Rodrigues da Silva[3]

[1] INESC-ID and Universidade Autónoma de Lisboa,
Rua de Santa Marta, nº 56, 1169-023 Lisboa, Portugal
cvideira@acm.org
[2] INESC-ID and Instituto Superior Técnico,
Rua Alves Redol, nº 9 –1000-029 Lisboa, Portugal
joao_leonardo@netcabo.pt
[3] INESC-ID and Instituto Superior Técnico,
Rua Alves Redol, nº 9 –1000-029 Lisboa, Portugal
alberto.silva@acm.org

**Abstract.** Requirements engineering is widely considered to be an essential activity for the successful development of information systems. This paper briefly presents a new initiative called "ProjectIT-Requirements" and describes the results achieved in the definition of a requirements specification language, called "ProjectIT-RSL", and the implementation of a prototype using VisualStudio.NET. This is the first step of a process that will enable the automatic generation of UML models and programming code, based on the MDD approach.

## 1 Introduction

In the Information Systems Group of INESC-ID [http://gsi.inesc-id.pt/] we have been developing efforts to increase the productivity of the software development process. Recently we started a research program, called "ProjectIT", whose goal is to develop a complete software development workbench with support for requirements engineering, analysis and design, generative programming techniques, and project management activities [5]. Our experience in the area of requirements engineering led us to conclude that to achieve success, requirements must somehow be formalized, and although previous initiatives were an important contribution [3], for a number of different reasons they have not been widely adopted. On the other hand, most of the existing tools are above all requirements management tools that do not automate important tasks and reduce the work involved in the overall process. For example, they do not promote the reuse of requirements, neither do they allow a tight integration between requirements and other software process artefacts (such as models and code).

Within this context we started the ProjectIT-Requirements project [6], which combines the benefits of formalizing the requirements specification with the need to use a simple notation understandable by non-technical stakeholders. Our hypothesis is that we should adopt a "controlled natural language", a subset of natural language with specific rules for requirements specification, with a limited vocabulary and a simplified grammar.

## 2 The ProjectIT-RSL Language

After defining the overall ProjectIT-Requirements architecture [6], we developed an initial prototype of the main deliverables of the project, upon which all the others depend: the **Requirements Language**, defined by a metamodel (a brief overview is shown in Figure 1), and by a grammar for defining the rules to map these concepts into sentences; the **Requirements Editor**, which acts like a traditional editor for introducing controlled requirements text; and the **Requirements Compiler** and **Intellisense** features, responsible for checking the requirements definitions, introduced in the editor, against the syntactic and semantic rules defined by language.

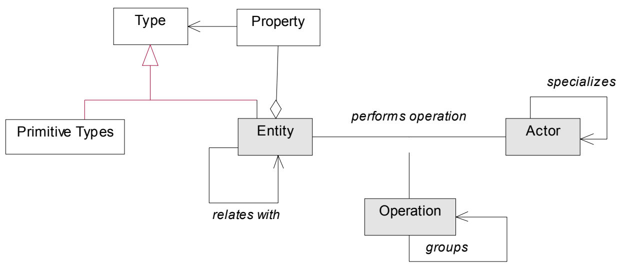


**Figure 1** – ProjectIT-RSL metamodel

Our initial proposal for the ProjectIT-RSL language is based upon the analysis of how requirements are most often described: normal language sentences where *actors* carry out *actions* which imply the access to one or more *entities*.

- **Actors** are defined as active resources (e.g., an external system or an end-user) that perform operations involving typically one or more entities.
- **Entities** are the static resources that are affected by the operations (e.g., a document or the data about a client or an invoice stored in a database). Entities have **Properties** that represent and describe their state.
- **Operations** are described by their respective workflows, which consist of a sequence of simpler Operations that affect Entities. This recursive definition will end in atomic and primitive Operations (e.g., create, update or delete operations) provided by default by our framework.

## 3 Prototype Development

In order to evaluate the preliminary version of the ProjectIT-RSL in a high-productive environment we decided to take the benefits of the features provided by Visual Studio .NET and .NET Framework [http://msdn.microsoft.com/] and so we decided to build a prototype in this environment. The choice is quite obvious, since this development environment provides some of the features we consider important such as intellisense and syntax validation when writing code. Microsoft also provides the Visual Studio Industry Partner Program [http://msdn.microsoft.com/vstudio/extend/], abbreviated VSIP, which are a set of COM APIs that enable the integration of new features in the Visual Studio.NET development environment, such as the possibility of adding new languages, or creating new types of projects.

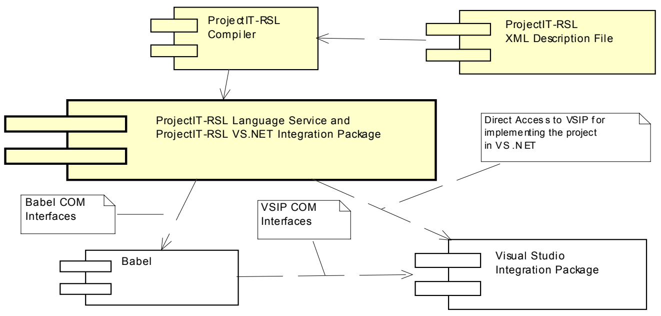
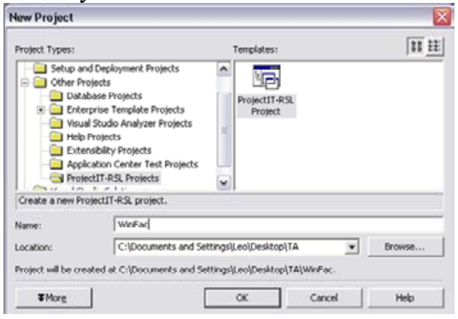

ProjectIT-RSL Compiler
ProjectIT-RSL XML Description File
ProjectIT-RSL Language Service and ProjectIT-RSL VS.NET Integration Package
Direct Access to VSIP for implementing the project in VS.NET
Babel COM Interfaces
VSIP COM Interfaces
Babel
Visual Studio Integration Package

**Figure 2** – ProjectIT-Requirements tools architecture

These VSIP APIs are very powerful, but difficult to understand and very low-level from the developer's point of view (they are written in C++ and the code is difficult to understand). Hence, we decided to use the Babel library that comes together with VSIP, but provides a higher abstraction layer above it. For example, the intellisense feature is handled directly by Babel and can be accessed from it. We also decided to use implementations of Lex and Yacc (Flex [http://www.gnu.org/software/flex/] and Bison [http://www.gnu.org/software/bison/bison.html]) to implement the Language Checker in the Editor, and also for the compiler of the ProjectIT-RSL, based upon syntax and semantic rules well known from programming languages. Figure 2 shows how the different components of the architecture integrate to provide the global functionality.

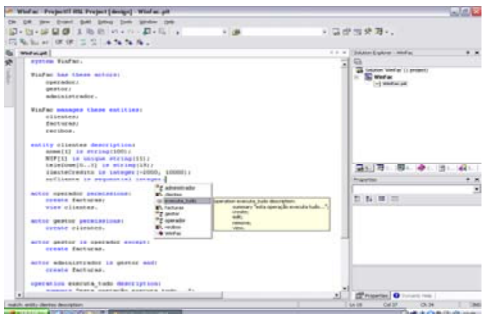


Create a new ProjectIT-RSL project.

Name: WinFac

Location: C:\Documents and Settings\Leo\Desktop\TA

Project will be created at C:\Documents and Settings\Leo\Desktop\TA\WinFac.


WinFac - ProjectIT-RSL Project [design] - WinFac.pit
Solution Explorer - WinFac
Solution 'WinFac' (1 project)
WinFac
WinFac.pit
Properties
administrador
clientes
executa_tudo
facturas
gestor
operador
recibos
WinFac

```
system WinFac.

WinFac has these actors:
    operador;
    gestor;
    administrador.

WinFac manages these entities:
    clientes;
    facturas;
    recibos.

entity clientes description:
    nome[1] is string(100);
    NIF[1] is unique string(15);
    telefone[0..*] is string(15);
    limiteCredito is integer(-2000, 10000);
    nrCliente is sequencial integer;

actor operador permissions:
    create facturas;
    view clientes.

actor gestor permissions:
    create clientes.

actor gestor is operador except:
    create facturas.

actor administrador is gestor and:
    create facturas.

operation executa_tudo description:
```

**Figure 3** – Creating a new ProjectIT-RSL project and using the ProjectIT-RSL editor

We have already implemented a complete prototype integrated in Visual Studio.NET, which includes: (1) the possibility of creating new projects (as shown in Figure 3); (2) the ProjectIT–RSL editor (also in Figure 3), which supports full syntax highlighting, the auto-complete feature and on-the-fly syntactic language verification, suggesting the available correction choices; and (3) the ProjectIT-RSL compiler.

Besides detecting additional errors in the requirements written, the compiler produces a XML file currently conformant with the XIS/UML profile [4], which will be the input to ProjectIT-MDD, the component of ProjectIT that is intended to specify, simulate and develop information systems according with the MDD approach. In this respect, our work has some similarities with the projects described in [1] and [2], although we will follow a less formal approach [6].

## 4 Conclusions and Future Work

The results achieved with this prototype and with the current ProjectIT-RSL metamodel confirm that our proposal is a suitable and effective approach to requirements specification. Using a controlled natural language to help requirements description will ease the involvement of the non-technical stakeholders in the requirements specification and management process.

The future work, besides improving ProjectIT-RSL (for example, adding support for the specification of more complex and versatile operations and workflows), includes the integration of these features in a non-proprietary development environment (the ProjectIT-Studio tool [4]). Simultaneously, we will research for more advanced features, such as requirements reuse based on requirements architectures.

Although the ProjectIT-RSL is an initiative in the area of requirements specification, the evolution of the ProjectIT-RSL language will integrate and extend the current XIS/UML profile [4], resulting in the ProjectIT/UML profile, a common metamodel for all ProjectIT sub-systems [5]. This profile will provide mechanisms to support the automatic generation of UML models of the system to be developed, and when processed by ProjectIT-MDD tools, will enable the generation of design and code artefacts. This will be a seamless integration, supported by the capability of ProjectIT-MDD tools to read the XML generated by ProjectIT-Requirements.

## References

1. Bryant, B., et al. (2003) From Natural Language Requirements to Executable Models of Software Components, *Proc. of Monterey Workshop on Software Engineering for Embedded Systems: From Requirements to Implementation*, pp. 51-58, Chicago, Illinois
2. Lamsweerde, A. (2003) Goal Oriented Requirements Engineering: From System Objectives to UML Models to Precise Software Specifications, *Proc. of the 25th International Conference on Software Engineeering*, IEEE Computer Society.
3. Lamsweerde, A. (2000) Formal Specification: a Roadmap, Proceedings of the conference on The future of Software engineering, pp 147-159, Limerick, Ireland.

4. Silva, A. R., Lemos, G., Matias, T., Costa, M. (2003) The XIS Generative Programming Techniques, Proc.of the 27th COMPSAC Conference, IEEE Computer Society.
5. Silva, A. R. (2004) O Programa de Investigação "ProjectIT", *White Paper, Relatório INESC-ID*, Grupo de Sistemas de Informação.
6. Videira, C., Silva, A. R. (2004) ProjectIT-Requirements, a Formal and User-oriented Approach to Requirements Specification, *Proc. of the JIISIC 2004 Conference*.