UNIVERSIDADE TÉCNICA DE LISBOA

INSTITUTO SUPERIOR TÉCNICO

LICENCIATURA EM ENGENHARIA INFORMÁTICA E DE COMPUTADORES

Relatório de Trabalho Final de Curso

ProjectPRO

PLATAFORMA DE GESTÃO DE REQUISITOS

http://berlin.inesc.pt/projects/ProjectPro

Julho de 2003

Aluno

Vítor Manuel Cesário Moreira – Nº 46995

Professor Orientador

Professor Alberto Manuel Rodrigues da Silva

Resumo

No âmbito deste Trabalho Final de Curso foi desenvolvido o ProjectPRO, uma plataforma de gestão de requisitos que suporta uma metodologia proposta também no âmbito deste trabalho. Esta ferramenta pretende facilitar a comunicação entre clientes, analistas e equipa de desenvolvimento, promovendo a gestão e reutilização de requisitos.

A metodologia proposta sugere entre vários aspecto, a utilização de um glossário de termos nas descrições textuais dos requisitos, uniformizando assim os conceitos envolvidos num sistema. A forma de especificação de requisitos possibilitará a utilização do repositório de requisitos ao longo de todo o processo de desenvolvimento de software.

Pretende-se assim que o repositório da ferramenta contenha informação que possa ser utilizada desde a fase de concepção até à fase de manutenção, passando pela própria fase de desenvolvimento. Tarefas como o planeamento, análise, desenho, desenvolvimento, teste gestão de alterações e controlo de qualidade poderão assim ter como ponto de partida a informação associada aos requisitos. Utilizam-se deste modo os requisitos e a informação a estes associada para tarefas de gestão de projecto.

Palavras-chave:

Projecto, *release*, subsistema, requisito, caso de uso, caso de negócio, validação de requisitos, qualidade de requisitos, teste de requisitos, entidades informacionais, agente de interface, equipa de desenvolvimento, processo de desenvolvimento de software.

Índice

Capítulo 1

1 Introdução

Apresenta-se neste capítulo a visão geral do trabalho desenvolvido no âmbito deste Trabalho Final de Curso (TFC). É apresentado o enquadramento do trabalho, os objectivos, a calendarização de actividades, a organização do relatório e as notações adoptadas ao longo do mesmo.

1.1 Introdução

Este TFC foi realizado individualmente, tendo em conta realização do último ano do curso em regime integrado com o mestrado. O tema "Plataforma de Gestão de Requisitos" foi proposto ao orientador, tendo sido aceite. O trabalho começou com o estudo e análise dos mecanismos, metodologias e ferramentas existentes nesta área. De seguida foi proposta uma metodologia de definição e especificação de requisitos, assim como uma ferramenta de suporte a esta considerando novas formas de especificação e gestão de requisitos. Depois de analisada esta proposta, bastante bem aceite pelo orientador, passou-se à delineação de uma ferramenta genérica de suporte à metodologia proposta. Finalmente foi iniciado o desenvolvimento de uma ferramenta de suporte tendo em conta aspectos da ferramenta genérica delineada. Esta ferramenta, designada por "ProjectPRO", permite estruturar os requisitos e utilizar a informação a eles associada para tarefas de gestão de projecto, a um nível profissional e no âmbito de projectos de grande dimensão, daí o nome atribuído.

1.2 Enquadramento

A ideia base deste trabalho surgiu na sequência de uma análise empírica ao processo de desenvolvimento de software e de mecanismos como a passagem de modelos de classes para modelos de dados e destes para a geração de código, por exemplo. Pretendeu-se assim analisar a possibilidade de obter uma estruturação que permita desde o início do processo de desenvolvimento a uniformização e reutilização de informação, como seja a definição das entidades informacionais a partir dos requisitos dos sistemas a desenvolver. Uma parte do trabalho foi integrada com o projecto da cadeira de Introdução aos Agentes Autónomos (IAA), tendo consistido no desenvolvimento de agentes de interface para o ProjectPRO, como será posteriormente detalhado.

1.3 Objectivos

Este trabalho tem como principais objectivos: (1) a delineação de uma metodologia de definição e estruturação de requisitos; (2) a delineação de uma ferramenta genérica de suporte à metodologia proposta, e (3) o desenvolvimento de uma ferramenta de suporte.

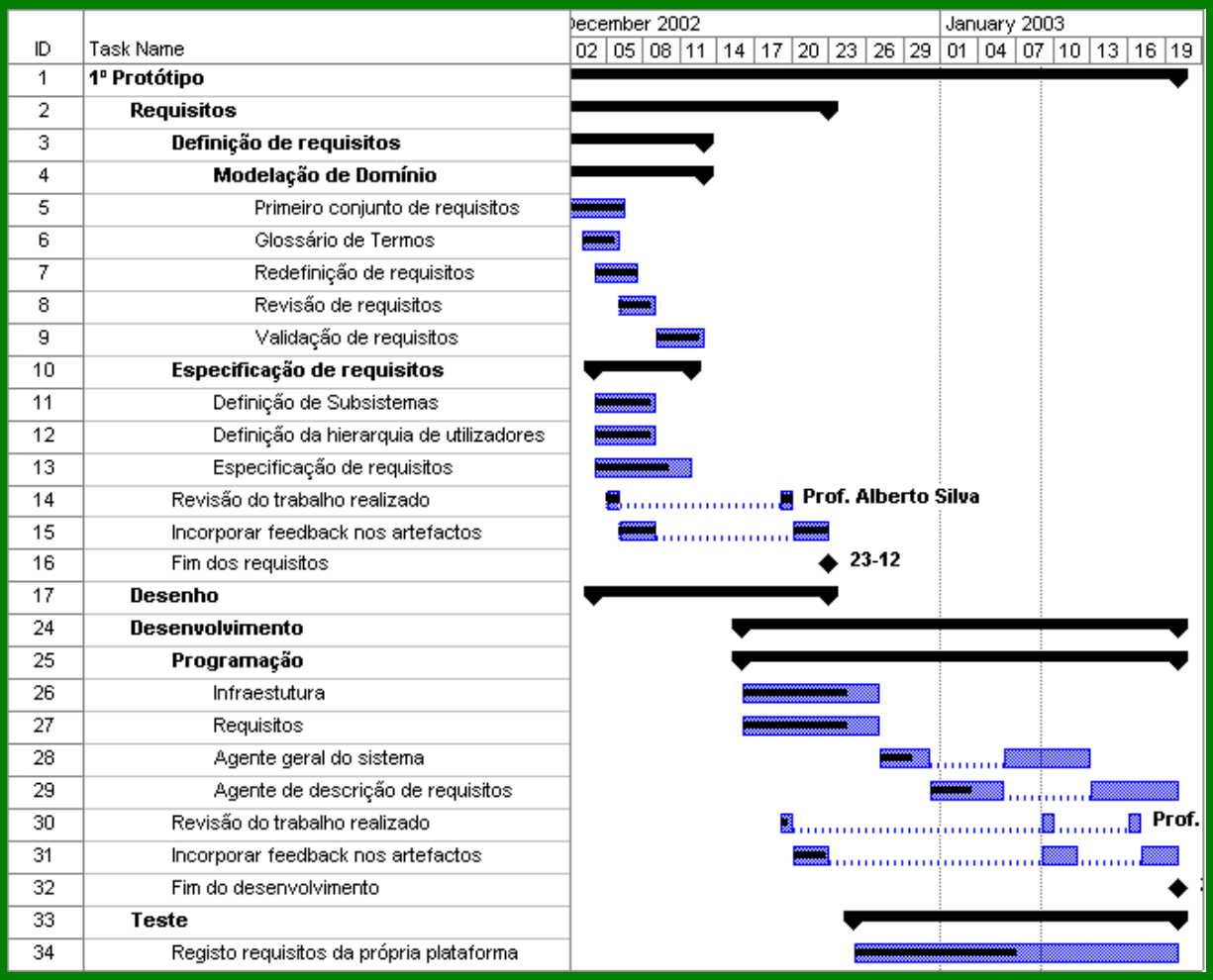
A plataforma de gestão de requisito pretende facilitar a comunicação entre cliente, analistas e equipa de desenvolvimento, promovendo a gestão e reutilização de requisitos, assim como suportar algumas tarefas de gestão de projecto. O desafio inicial foi avaliar até que ponto um repositório de requisitos poderia suportar todo o processo de desenvolvimento de software.

1.4 Actividades desenvolvidas

A primeira fase do TFC foi o levantamento do estado da arte na área da Engenharia de Requisitos. Com base nisso foi delineada a metodologia de definição, especificação e gestão de requisitos assim como uma ferramenta genérica para dar suporte à metodologia. Estas actividades decorreram essencialmente nos meses de Outubro e Novembro de 2002 e deu origem ao artefacto [Moreira2002]. Este artefacto encontra-se disponível no *Web Site* do projecto em http://berlin.inesc.pt/projects/ProjectPro/ e nele encontra-se: (1) a análise de algumas técnicas, metodologias e ferramentas utilizadas na área dos requisitos; (2) a delineação da metodologia de definição e especificação de requisitos; e (3) a delineação da uma ferramenta genérica de suporte à metodologia.

Depois de analisado o artefacto [Moreira2002], foi abordada a questão de desenvolver uma ferramenta de raiz ou trabalhar sobre uma ferramenta já existente. Optou-se por desenvolver uma aplicação de raiz de modo a experimentar na prática todas as tarefas envolvidas no surgimento de um produto, desde a sua concepção até à implementação. Por outro lado, com esta decisão, conseguiu-se uma autonomia em relação a produtos e metodologias existentes, dando uma maior liberdade nas escolhas efectuadas, discutidas sempre com orientador.

No início de Dezembro de 2002 foram especificados os requisitos para o primeiro protótipo da aplicação. A definição e a especificação de requisitos seguiram a metodologia proposta, com se pode constatar no planeamento de actividades do primeiro protótipo apresentado na Figura 1.1.



ID	Task Name
1	**1º Protótipo**
2	**Requisitos**
3	**Definição de requisitos**
4	**Modelação de Domínio**
5	Primeiro conjunto de requisitos
6	Glossário de Termos
7	Redefinição de requisitos
8	Revisão de requisitos
9	Validação de requisitos
10	**Especificação de requisitos**
11	Definição de Subsistemas
12	Definição da hierarquia de utilizadores
13	Especificação de requisitos
14	Revisão do trabalho realizado
15	Incorporar feedback nos artefactos
16	Fim dos requisitos
17	**Desenho**
24	**Desenvolvimento**
25	**Programação**
26	Infraestutura
27	Requisitos
28	Agente geral do sistema
29	Agente de descrição de requisitos
30	Revisão do trabalho realizado
31	Incorporar feedback nos artefactos
32	Fim do desenvolvimento
33	**Teste**
34	Registo requisitos da própria plataforma

Figura 1.1 – Planeamento de actividades para o primeiro protótipo

Durante o mês de Dezembro de 2002 foi desenvolvido o primeiro protótipo do ProjectPRO. Devido ao trabalho realizado no âmbito da cadeira de IAA, o foco até ao final de Janeiro de 2002 foi no desenvolvimento dos agentes de interface já referidos.

No mês de Fevereiro, concluídos os agentes, foi elaborado artigo "Agentes de Interface no ProjectPRO – Plataforma de Gestão de Requisitos" (Apêndice E), o qual foi submetido à Workshop de Agentes (WIAA2003), na qual foi apresentado o primeiro protótipo do ProjectPRO com os agentes de interface. Estas actividades decorreram no âmbito da cadeira de IAA.

O trabalho desenvolvido até meados de Julho de 2003 foi essencialmente a implementação de funcionalidades que concretizam os aspectos principais da ferramenta genérica apresentada em [Moreira2002], o que constituiu o segundo protótipo do ProjectPRO. Na Figura 1.2 é apresentado o planeamento de actividades efectuadas no âmbito do segundo protótipo.

Figura 1.2 – Planeamento de actividades para o segundo protótipo

1.5 Organização do Relatório

O relatório encontra-se organizada em seis capítulos e cinco apêndices conforme se resume de seguida.

No Capítulo 2 – "Estado da Arte", é feito um breve levantamento do estado da arte na área da Engenharia de Requisitos, abordando os processos de definição, especificação e gestão de requisitos.

No Capítulo 3 – "Metodologia Proposta", é apresentada a metodologia de definição e especificação de requisitos proposta, a qual facilita todo o processo de recolha e gestão dos mesmos.

No Capítulo 4 – "Ferramenta de suporte proposta", apresenta-se a delineação uma ferramenta genérica de gestão de requisitos.

No Capítulo 5 – "Ferramenta desenvolvida – ProjectPRO", apresenta-se a ferramenta desenvolvida para suportar a metodologia proposta. São abordadas as metodologias e técnicas de desenvolvimento e a arquitectura de software, assim como alguns mecanismos importantes e funcionalidades desenvolvidas.

No Capítulo 6 – "Conclusão", é apresentada uma análise global do trabalho realizado. São resumidos os principais contributos deste trabalho e são identificadas as questões em aberto. Finalmente é apresentada uma visão integrada com a Tese de Mestrado e alguns comentários finais.

No Apêndice A – "Modelo de dados do ProjectPRO", encontram-se as entidades do modelo de dados da ferramenta desenvolvida.

No Apêndice B – "Estrutura da solução utilizado no desenvolvimento do ProjectPRO"

No Apêndice C – "Relatórios produzidos pelo ProjectPRO", encontram-se dois relatórios produzidos pela ferramenta desenvolvida.

No Apêndice D – "Manual de Utilização do ProjectPRO", encontra-se o Manual de Utilização da ferramenta desenvolvida.

No Apêndice E – "Artigo Agentes de Interface no ProjectPRO – Plataforma de Gestão de Requisitos ", encontra-se o artigo elaborado para a *workshop* de agentes WIAA2003.

1.6 Notações Adoptadas

Ao longo do relatório são adoptadas genericamente as seguintes regras de notação textual:

- Nomes e expressões em inglês são escritos em *itálico.* As excepções são expressões vulgarmente adoptadas para o Português (e.g., software, bit), expressões intensamente usadas ao longo do texto (e.g., Internet, Web), ou nomes de produtos e tecnologias de origem anglo-saxónica (e.g. Microsoft SQL Server, JavaScript).
- Frases e expressões que se pretendam destacar são escritas com ênfase (a "**negrito**").
- Nomes de classes são apresentados numa fonte de tamanho fixo (i.e., Courier).

Relativamente à representação de diagramas será utilizada, sempre que for adequado, a linguagem UML (Unified Modeling Language).

Capítulo 2

2 Estado da Arte

Neste capítulo é feito um breve levantamento do estado da arte na área da Engenharia de Requisitos. São abordados os processos de definição, especificação e gestão de requisitos, assim como algumas metodologias, técnicas e ferramentas existentes. Este assunto é apresentado em [Moreira2002].

2.1 Introdução

Durante o **processo de desenvolvimento de software** (Figura 2.1), são criados, alterados e usados diversos tipos de informação que são chamados **artefactos**. Dos primeiros artefactos a surgir salientam-se os **requisitos**. A tarefa de definir o que um **cliente** pretende de um sistema de informação, de agora em diante designado apenas por **sistema**, não é trivial. É necessário dispor de um mecanismo que permita capturar as necessidades dos clientes (requisitos do sistema) de modo a poderem ser comunicados a todos os intervenientes no **projecto**. É importante que os requisitos sejam armazenados de modo a permitir acções básicas como a sua pesquisa ou revisão. A descrição dos requisitos deve ser compreendida por clientes, utilizadores e equipa de desenvolvimento. Cada requisito deve ter um estado e deve ser possível associá-lo a outros artefactos do processo de desenvolvimento (e.g. casos de uso) que estejam relacionados com este. Se os requisitos tiverem uma representação adequada será possível ter processos fáceis e eficientes de validação, revisão e verificação da satisfação dos mesmos durante o desenvolvimento.

Figura 2.1 - Principais fases do Processo de Desenvolvimento de Software

A concepção de um software é feita a nível do **universo do discurso** (*UoD*) e do **domínio do problema**. As principais tarefas desta fase são o **planeamento** e a **análise**. No final desta fase, deve-se saber responder às seguintes questões: (1) o que fará o sistema para cada um dos seus utilizadores? (modelo de casos de uso simplificado); (2) que tipo de arquitectura terá o sistema? (esboço com os principais subsistemas); e (3) qual o planeamento e custo do desenvolvimento do sistema? (identificar riscos, prioridades e estimar custos).

2.2 Problemática dos requisitos

Como já foi referido, o levantamento de requisitos de um sistema não é trivial. Este facto deve-se por exemplo à grande dimensão de alguns sistemas e à mudança dos requisitos. Os principais problemas dos requisitos são [Oberg2001]:

- Nem sempre são óbvios e existem muitas fontes.
- Nem sempre são fáceis de exprimir.
- Existem muitos tipos de requisitos com diferentes níveis de detalhe.
- O número de requisitos pode-se tornar a gestão bastante difícil.
- Relacionam-se entre si e com outras entidades do processo de desenvolvimento de software.
- Têm diferentes propriedades, a nível de prioridade, benefício e tangibilidade.
- Há vários responsáveis, pelo que necessitam de uma gestão multi-funcional.
- Por vezes têm de ser alterados.
- Podem ser *time-sensitive*.

O *Gráfico 2-1* resume os resultados de um estudo de 1996 acerca dos problemas relacionados com requisitos [Oberg2001]. Neste estudo participaram gestores de desenvolvimento e pessoal de controlo de qualidade de software. O gráfico mostra a percentagem de participantes que passaram por cada um dos problemas mencionados.

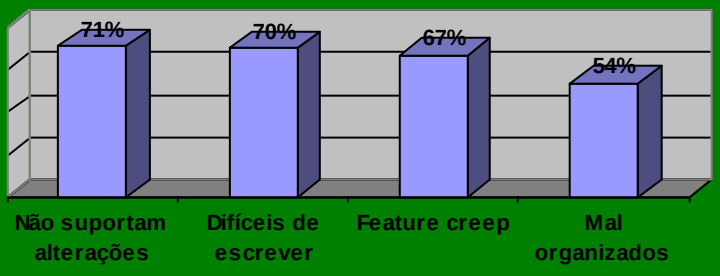


Gráfico 2-1 – Problemas mais comuns dos requisitos

2.3 Tipos de requisitos

Os requisitos são geralmente classificados de acordo com a sua especificidade. Os tipos de requisitos mais gerais e vastamente utilizados são [Jacobson99]:

- **Requisitos funcionais**: especificam uma acção que o sistema deve ser capaz de fazer, sem quaisquer restrições físicas, tais como comportamentos de entrada e saída do sistema, lógica de decisão ou algoritmos.
- **Requisitos não-funcionais**: especificam propriedades do sistema, tais como restrições de implementação, desempenho, segurança, manutenção ou extensão e que especificam restrições físicas aos requisitos funcionais.
- **Requisitos de desenvolvimento**: descrevem restrições ao processo de desenvolvimento do sistema e não são perceptíveis pelos utilizadores, como linguagem de programação ou metodologias.

2.4 Artefactos relacionados com requisitos

Tipicamente são produzidos, em linguagem natural, dois tipos de documentos de requisitos [Rosenberg99]:

- **Definição de requisitos:** descreve os requisitos de modo a serem entendidos pelos clientes e utilizadores. Define um entendimento entre o cliente e a equipa de desenvolvimento sobre o que o sistema deve fazer. É normalmente escrito pelos clientes e analistas de requisitos.
- **Especificação de requisitos:** reescreve o documento de definição de requisitos em termos técnicos, mais apropriado à equipa de desenvolvimento e às actividades de desenho. É normalmente escrito pelos analistas de requisitos.

Existe um conjunto de artefactos relacionados com requisitos que facilitam diversas tarefas, nomeadamente, a planificação de fases posteriores do processo de desenvolvimento de software (e.g. a fase de elaboração), tais como: (1) lista de funcionalidades; (2) primeira versão do modelo de negócio e/ou de domínio que descreve o contexto do sistema; (3) primeiro conjunto de modelos de casos de uso e de requisitos suplementares; (4) alguns protótipos de interfaces de utilizador; (5) lista inicial de riscos e prioridades de requisitos e casos de uso; (6) casos de teste; (7) relatórios de erros; e (8) pedidos de alterações.

2.5 Qualidade de requisitos

É importante detectar erros no princípio do processo de desenvolvimento de software, pois deste modo, o custo da sua correcção será menor, evitando que o erro seja propagado para o domínio da solução.

A escrita dos requisitos deve seguir um conjunto de regras de modo a permitir a sua verificação objectiva, tais como [Rito2001]: (1) escrever uma quantidade para cada advérbio e adjectivo de modo a que o significado dos qualificadores seja claro e não ambíguo; (2) substituir pronomes por nomes de entidades; e (3) assegurar que cada substantivo é definido exactamente uma única vez nos documentos de requisitos.

Existem diversos aspectos de qualidade que poderão ser considerados, de acordo com o âmbito de cada sistema, tais como [Anneliese90]: (1) clareza; (2) compreensão; (3) consistência; e (4) custo. Tipicamente, a equipa de controlo de qualidade (*quality assurance team*) revê cada secção ou parágrafo do documento de requisitos e classifica-os em cada um dos aspectos referidos. Os especificadores (*specification writers*) e os utilizadores também devem avaliar o documento, pois este deve ser compreendido por todos os intervenientes.

2.6 Técnicas e Metodologias

De um modo geral, no processo de recolha de requisitos são realizadas as seguintes actividades, dando cada uma origem a diferentes artefactos [Jacobson99]:

1. Listar requisitos candidatos.

2. Compreender o contexto do sistema:

3. **Modelação de negócio**: descrever processos de negócio que serão suportados.

4. **Modelação de domínio**: descrever conceitos importantes do domínio e interligá-los.

5. Recolher requisitos funcionais.

6. Recolher requisitos não-funcionais.

7. Recolher requisitos de desenvolvimento.

2.6.1 O Rational Unified Process

O Rational Unified Process (RUP) é um *framework* de metodologias da Rational. O RUP defende a modelação visual UML, o controle de qualidade permanente, é conduzido por casos de uso, centrado numa arquitectura, e é um processo iterativo e incremental [Silva2001]. *O workflow* da análise de requisitos no RUP é constituído pelas seguintes tarefas [Oberg2001]: (1) definir o modelo de negócio; (2) definir o modelo de domínio;

(3) especificar requisitos suplementares; (4) fazer uma lista de *features*; (5) identificar actores e casos de uso; e (6) especificar cenários, estados e definir as suas prioridades.

2.6.2 Os casos de uso e de negócio

O conceito de **caso de uso** (*use-case)*, introduzido no início dos anos 1990 por Jacobson, é vastamente utilizado para capturar requisitos [Jacobson99]. Nesta abordagem, os requisitos do cliente são capturados em diagramas designados por casos de uso. Num caso de uso, o importante é definir o que o sistema deve fazer na óptica dos utilizadores, não sendo de todo importante como é que o sistema será internamente.

Alistair Cockburn propôs uma *template* de casos de uso [Cockburn98] de modo a uniformizar a descrição dos mesmos (Tabela 2-1).

USE CASE #	< the name is the goal as a short active verb phrase>	
Goal in Context	<a longer statement of the goal in context if needed>	
Scope & Level	<what system is being considered black box under design> <one of : Summary, Primary Task, Sub function>	
Preconditions	<what we expect is already the state of the world>	
Success End Condition	<the state of the world upon successful completion>	
Failed End Condition	<the state of the world if goal abandoned>	
Primary, Secondary Actors	<a role name or description for the primary actor>. <other systems relied upon to accomplish use case>	
Trigger	<the action upon the system that starts the use case>	
DESCRIPTION	Step	Action
	1	<put here the steps of the scenario from trigger to goal delivery, and any cleanup after>
	2	<...>
EXTENSIONS	Step	Branching Action
	1a	<condition causing branching> : <action or name of sub use case>
SUB-VARIATIONS		Branching Action
	1	<list of variation s>
RELATED INFORMATION	<Use case name>	
Priority:	<how critical to your system / organization>	
Performance	<the amount of time this use case should take>	
Frequency	<how often it is expected to happen>	
Channels to actors	<e.g. interactive, static files, database, timeouts>	
OPEN ISSUES	<list of issues awaiting decision affecting this use case >	
Due Date	<date or release needed>	
...any other management information...	<...as needed>	
Super ordinates	<optional, name of use case(s) that includes this one>	
Subordinates	<optional, depending on tools, links to sub use cases>	

Tabela 2-1 Template de casos de uso proposta por Alistair Cockburn

O modelo de casos de uso representa uma visão do sistema do ponto de vista de quem modela os requisitos funcionais [Rosenberg99]. Os casos de uso devem ser utilizados para decompor o domínio do problema e delimitar o sistema, definindo o que cada **actor** deve fazer e as diferentes sequências de acções, designadas por **cenários** (principais ou alternativos). Podem ser usados essencialmente para especificar requisitos funcionais.

Existem ainda os **casos de negócio** que representam um sistema (aqui um negócio) na perspectiva da utilização e que reflectem como é que este dá valor aos seus utilizadores (aqui clientes e sócios).

2.6.3 A metodologia ICONIX

A metodologia ICONIX, proposta por Doug Rosenberg, define-se como um processo de desenvolvimento de software simples e prático [Rosenberg99]. As principais características são: (1) é conduzido por casos de uso; (2) é iterativo e incremental; (3) usa o UML como linguagem de modelação; (4) apresenta um alto grau de rastreabilidade; e (5) considera quatro fases de desenvolvimento (análise de requisitos, análise e desenho preliminar; desenho e implementação). Nesta metodologia, um caso de uso descreve uma unidade de comportamento, enquanto que um requisito descreve uma regra que rege esse comportamento [Silva2001].

2.6.4 A Matriz Volere

A Matriz Volere é uma *template* da Atlantis System Guild que procura estruturar a utilização de linguagem natural na definição de requisitos, procurando estendê-la e não substitui-la. Na definição de requisitos são considerados os seguintes tipos de requisitos [Robertson2001]: (1) requisitos funcionais (e.g. lógica de decisão e algoritmos); (2) requisitos não-funcionais (e.g. desempenho, segurança, cultural, legal e político); (3) restrições de projecto (*Project Constraints*), tais como interoperabilidade, custo ou prazo; (4) controladores de projecto (*Project Drivers*), tais como objectivos do projecto; e (5) propriedades de projecto (*Project Issues)*, tais como condições em que o projecto será desenvolvido.

2.7 Ferramentas de suporte à Gestão de Requisitos

Existem algumas ferramentas que dão suporte à gestão de requisitos. Foram analisadas as seguintes ferramentas: (1) Rational Requisite PRO, a ferramenta de gestão de requisitos da Rational, usada para documentação e gestão ao longo do processo de desenvolvimento de software; (2) Reconcile, da Compuware Software, que é um sistema de gestão de requisitos que permite criar requisitos, alterá-los e produzir relatórios de requisitos; e (3) Analyst PRO, da GodaSoftware, que suporta actividades do ciclo de desenvolvimento de software como a especificação de requisitos, a análise de requisitos, a implementação de sistemas, testes de software baseados em requisitos, manutenção e melhoramentos. Na Tabela 2-2 encontra-se uma tabela comparando estas ferramentas em diversos aspectos.

Funcionalidades	Rational Requisite Pro	Reconcile	Analyst Pro
Requisitos funcionais e não funcionais	Permite. Os tipos de requisitos são parameterizáveis	Permite	Não identificado
Estruturação dos passos de casos de uso	Não permite	Não permite	Não identificado
Definição de actores de caos de uso	Não permite	Não permite	Não identificado
Criação de inclusão/extensão de casos de uso	Não permite	Não permite	Não identificado
Hierarquia de requisitos	Permite	Permite	Não identificado
Hierarquia de casos de uso	Permite	Permite	Não identificado
Histórico de alterações	Permite	Permite	Permite
Atribuição de requisitos a elementos da equipa	Permite mas não é parameterizável	Permite mas não é parameterizável	Permite
Capacidade de *reporting*	Relatórios personalizados de requisitos	Diversos relatórios de requisitos	Relatórios personalizados de requisitos
Processo de validação de requisitos	Só integrado com outra aplicação	Permite e pode ser integrado com outra aplicação	Permite
Relações de rastreabilidade	Permite	Permite	Não identificado
Colaboração	Permite apenas grupo de discussão	Não identificado	Permite
Integração com outras aplicações	Permite	Permite	Não identificado
Relação com fase de testes	Só integrado com outra aplicação	Permite e pode ser integrado com outra aplicação	Permite
Definição de glossário de termos	Permite	Não permite	Não identificado
Utilização de glossário de termos na definição de requisitos	Não permite	Não permite	Não identificado
Reutilização de requisitos	Permite importar requisitos	Não identificado	Permite importar e requisitos
Geração de manual de utilizador	Integrado com outra aplicação	Não permite	Não identificado
Acesso via Web	Permite, excepto para gestão de projectos e utilizadores	Permite apenas consultar documentação	Não permite mas existe versão em rede

Tabela 2-2 – Comparação de algumas ferramentas de gestão de requisitos

Capítulo 3

3 Metodologia proposta

Neste capítulo é apresentada a metodologia de definição e especificação de requisitos proposta, a qual procura facilitar todo o processo de recolha e gestão dos mesmos. Esta metodologia encontra-se delineada em [Moreira2002].

3.1 Introdução

Nesta metodologia são considerados aspectos das diferentes metodologias e ferramentas de gestão de requisitos analisadas anteriormente. A base da metodologia está na estruturação dos requisitos, a primeira informação sobre o sistema a desenvolver. Isto permite, entre outras coisas, evitar ambiguidades que são bastante frequentes quando se utiliza exclusivamente linguagem natural. De seguida são enumerados os principais objectivos a considerar:

- Facilitar a comunicação entre clientes, analistas e equipa de desenvolvimento.
- Promover a reutilização e gestão de requisitos.
- Controlar a evolução durante o processo de desenvolvimento de software.
- Suportar a revisão e validação conjunta entre clientes e analistas.
- Definir métricas como prioridade, satisfação, fiabilidade e evolução.
- Estabelecer uma ponte entre o espaço do problema e o espaço da solução.
- Disponibilizar um repositório que suporte diferentes fases do processo de desenvolvimento de software.

Pretende-se deste modo que o repositório de requisitos constitua o centro de todo o processo de desenvolvimento de software, comos e encontra representado na Figura 3.1.

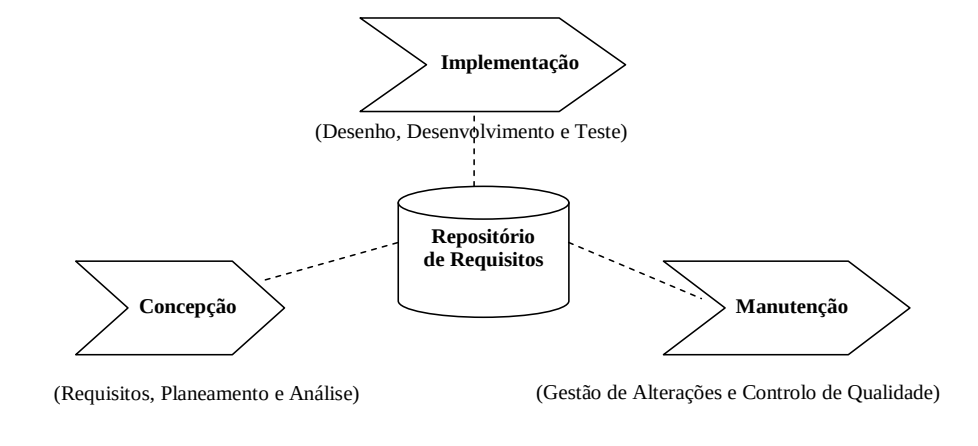


Figura 3.1 – A centralização do repositório de requisitos no processo de desenvolvimento de software

Das tarefas essenciais para atingir os objectivos apresentados destacam-se: (1) definir um vocabulário de projecto através da definição de um glossário de termos; (2) construir uma visão do sistema que descreva o problema a ser resolvido e as principais *features*; e (3) criar relatórios de estado e progresso. Para além destas tarefas essenciais existem outras que devem também ser levadas em conta, por exemplo: (1) determinar os tipos de requisitos a serem usados; (2) definir atributos a serem definidos para cada um dos requisitos; (3) escolher a forma na qual vão ser descritos os requisitos; (4) definir as relações de rastreabilidade necessárias; e (5) estabelecer um procedimento de proposta, revisão e alteração de requisitos.

Cada requisito deve ter um estado e deve ser possível associá-lo a outros artefactos do processo de desenvolvimento (e.g. casos de uso) que estejam relacionados com este. Pretende-se que os requisitos tenham uma representação adequada que possibilite ter processos fáceis e eficientes de validação, revisão e verificação da satisfação dos mesmos durante o desenvolvimento.

3.2 Processo de definição de requisitos

Na definição de requisitos deve-se descrever os requisitos de modo a serem entendidos pelos clientes e utilizadores, definindo um entendimento entre estes e a equipa de desenvolvimento sobre o que o sistema deverá ser. A base deste processo deve passar pela caracterização do universo do discurso (*UoD*), definindo os termos que deverão ser utilizados na definição de requisitos de modo a permitir uma maior organização, reutilização e uniformização de toda a informação. Para suportar a compreensão do contexto do sistema devem-se seguir duas aproximações:

- **Modelação do negócio**, na qual se descrevem os processos de negócio que serão suportados pelo sistema de modo a poder compreendê-los e eventualmente melhorá-los.
- **Modelação do domínio**, onde se descrevem os conceitos importantes como objectos de domínio assim como as suas interligações.

3.2.1 Modelação de negócio

A este nível deve-se descrever os **processos de negócio** que serão suportados pelo sistema de modo a poder compreendê-los e eventualmente melhorá-los. O importante é definir as entidades, as operações que estas devem disponibilizar e quem as executa. A modelação de processos de negócio poderá ser feita segundo duas vertentes: (1) situação actual (*as-is*); e (2) situação futura (*to-be*). A modelação de negócio pode não ser necessária para sistemas muito específicos ou técnicos como um *pacemaker* ou um controlador de vídeo. Sempre que fizer sentido, deve-se seguir a seguinte sequência actividades na modelação de negócio (Figura 3.2):

1. Definir as **entidades de negócio**, como uma factura, encomenda ou proposta.

2. Definir os **actores de negócio** (quem se relaciona com as entidades de negócio) que devem ser usados como ponto de partida para derivar um primeiro conjunto de actores e casos de uso do sistema a desenvolver.

3. Definir os principais **casos de negócio** (*business use case*)

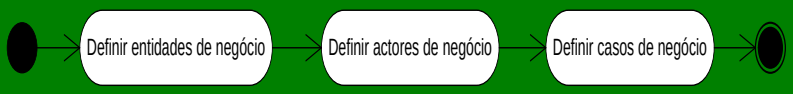


Figura 3.2 – Actividades da Modelação do Negócio

3.2.2 Modelação de domínio

A este nível deve-se descrever os conceitos importantes como **entidades informacionais** assim como as suas interligações. A identificação destas entidades (objectos de domínio) deve ser o ponto de partida para criação de um glossário de termos que: (1) permite definir os nomes pelos quais devem ser tratados os objectos do domínio e que devem ser utilizados na linguagem entre os intervenientes da equipa de projecto; (2) reduz ambiguidades na linguagem: quanto menos sinónimos existirem nos requisitos menos complexa será a sua análise; e (3) facilita a identificação de classes na fase de análise.

A modelação de domínio deve seguir uma abordagem iterativa e incremental, sugerindo-se a seguinte sequência de actividades (Figura 3.3):

1. Registar um primeiro conjunto de requisitos básicos, nomeadamente: **requisitos funcionais** (e.g. lógica de decisão e algoritmos); **requisitos não-funcionais** (e.g. desempenho, segurança, cultural, legal ou político); e **requisitos de desenvolvimento** (e.g. linguagem de programação ou metodologias).

2. Definir um **glossário de termos** a partir das entidades dos processos de negócio e requisitos registados. Sempre que se justifique devem ser associados sinónimos a cada um dos termos para facilitar a detecção de incoerências nas descrições textuais por uma ferramenta de suporte.

3. **Reescrever** os requisitos registados procurando uniformizar-se as descrições textuais de acordo com as entidades definidas no glossário de termos.

4. **Registar** novos requisitos recorrendo sempre aos termos definidos no glossário de termos.

5. **Actualizar** o glossário de termos com novos termos ou sinónimos.

6. **Rever** os novos requisitos registados.

7. **Repetir** o processo desde o ponto 4 até todos os requisitos se encontrarem definidos.

8. **Validar** a qualidade dos requisitos e, eventualmente, revê-los. A validação deve ser realizada de acordo com parâmetros como sejam a clareza, compreensão ou consistência. Na validação devem participar quer elementos da equipa de desenvolvimento, quer utilizadores pois os requisitos devem ser compreendidos por todos os que se relacionam directa ou indirectamente com estes.

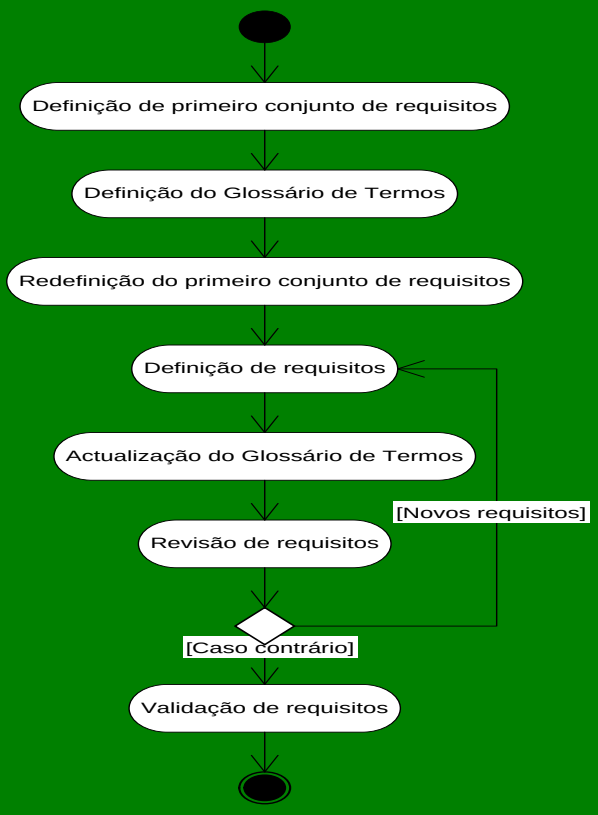


Figura 3.3 – Actividades da Modelação do Domínio

3.3 Processo de especificação de requisitos

Na especificação de requisitos deve-se procura-se detalhar a definição de requisitos em termos técnicos, mais apropriado à equipa de desenvolvimento e às actividades de concepção. Neste processo serão considerados os seguintes aspectos: subsistemas, hierarquia de actores, estrutura de requisitos e casos de uso.

Normalmente um sistema poder ser organizado em diversos **subsistemas**. É importante especificar os subsistemas que constituem o sistema assim como as interfaces de comunicação entre eles. Após a definição dos subsistemas, deve-se rever os requisitos de modo a relacioná-los com os subsistemas definidos.

Com base numa primeira análise dos requisitos é provável que se identifiquem desde logo diferentes actores do sistema. Se for o caso, deve-se especificar a **hierarquia de actores** com algumas credenciais associadas. Após a definição desta hierarquia deve-se proceder à revisão dos requisitos já registados para uniformizar as descrições textuais com os actores definidos.

Na especificação de requisitos deve-se então procurar detalhar da forma mais uniforme possível cada um dos requisitos. De seguida é enumerada com detalhe a informação que deve ser associada a cada requisito, sempre que faça sentido:

- Identificador que permite referenciar o requisito (e.g. RF1, RND2 ou RD3).
- Designação que reflicta a intenção do requisito.
- Tipo que permita agrupar requisitos em categorias (e.g. funcional, não funcional ou de desenvolvimento).
- Subsistemas aos quais o requisito diz respeito.
- Entidades às quais o requisito diz respeito, o que permite agrupar requisitos por entidade.
- Descrição textual utilizando as entidades e termos definidos no glossário de termos assim como os actores definidos na respectiva hierarquia.
- Release a que o requisito diz respeito, o que permite ter grupos de requisitos para cada uma das releases do sistema.
- Prioridade, de forma a suportar o planeamento de actividades de desenvolvimento (e.g. crítico ou importante).
- Custo, de forma a suportar a análise de custo (e.g. homem/hora).
- Benefício de forma a dar apoio à decisão (e.g. aumenta os potenciais clientes).
- Estado de progresso, para analisar o estado do sistema e a prever o final de actividades (e.g. proposto ou concluído).
- Risco, de forma a suportar a análise de risco (e.g. significante ou elevado).

- Grau de necessidade que reflicta a necessidade do cliente ter o requisito implementado (e.g. facultativo ou obrigatório).
- Versão para suportar a revisão e alteração de requisitos.
- Nível de detalhe, pois quando existem muitos tipos de requisitos com diferentes níveis de detalhe é útil distinguir o nível de detalhe (e.g. superficial ou detalhado).
- Nível de tangibilidade, de forma a suportar o planeamento (e.g. facilmente tangível ou dificilmente tangível).
- Origem, que pode ser quem propôs ou o motivo na origem do requisito.
- Nível de progresso em relação ao desenvolvimento do requisito. Pode ser discreto (e.g. nulo, avançado ou concluído) ou contínuo (e.g. percentagem).
- Responsáveis pelo requisito, para no caso de dúvida saber-se quem contactar.
- Atribuições a indivíduos da equipa de projecto destacados para trabalharem em algo com base neste requisito, o que facilita o planeamento de actividades.
- Intervenientes (indivíduos da equipa de projecto) que trabalharam em algo com base neste requisito, o que permite no caso de problemas saber quem esteve envolvido.
- Requisitos relacionados com este, o que permite saber as dependências e analisar o impacto de alterações.
- Relações de rastreabilidade com entidades de fases posteriores do processo de desenvolvimento de software que se basearam neste requisito (e.g. diagramas de actividades ou de sequência e casos de teste), o que facilita a gestão de alterações.
- Documentação suporte utilizada na definição do requisito (e.g. actas, protótipos de interfaces de utilizador, rascunhos, relatórios ou outros tipos de documentos). Será útil quer para os analistas que saberão tudo o que se relaciona com o requisito quer para os programadores que terão aí uma fonte de algoritmos e outra informação que não se encontra tão detalhadamente nos requisitos.
- Informação de histórico e auditoria (e.g. data de criação e alteração e responsáveis).
- Informação de validação (e.g. clareza, compreensão ou consistência).
- Informação relacionada com testes de software (e.g. Grau de correcção e/ou de completude).
- Observações diversas relacionadas com o requisito e que devam de estar junto deste, como pontos em aberto.

Os **casos de uso** devem ser utilizados para decompor o domínio do problema e delimitar o sistema. Os casos de uso devem ser vistos como uma extensão de um requisito, devendo por isso conter todos os elementos referidos na estrutura dos requisitos.

De seguida são apresentados outros elementos que se devem associar a um caso de uso:

- **Descrição**, que deve ser uma frase na voz activa, com um verbo no infinitivo.
- **Nível de abstracção**, poderá ser abstracto ou concreto.
- **Actores**, obrigatório e tem de ser um dos actores definidos.
- **Cenários** e respectivas **sequências de acções**, constituídas por um número de ordem e uma descrição da acção. Deve-se especificar: (1) pré e pós condições (e.g. como e quando o caso de uso começa e termina); (2) entidades envolvidas; (3) cenário principal; (4) cenários alternativos (e.g. situações de excepção); e (5) inter-relações de generalização, inclusão ou extensão**.**
- **Casos de uso relacionados** com o caso de uso (dependências).
- **Requisitos satisfeitos** por este caso de uso.
- **Métricas,** como uma performance (e.g. o tempo que este caso de uso deve demorar) ou uma frequência (e.g. número de vezes que é efectuado).

3.4 Gestão de requisitos

A estruturação apresentada facilita a gestão integrada de requisitos. A nível da gestão são importantes as seguintes actividades que devem, sempre que possível, ser suportadas por processos computacionais efectivos e eficientes: (1) comparar os objectivos do sistema com os requisitos para confirmar se são todos cobertos e se são todos necessários; (2) verificar a correcção e completude dos requisitos; (3) verificar a satisfação de requisitos durante o desenvolvimento; (4) definir as regras de negócio de forma a poderem ser usadas e verificadas; (5) criar relatórios de estado e de progresso; e (6) verificar as relações de rastreabilidade dos requisitos da definição à implementação para analisar, por exemplo, o impacto de alterações.

De modo a apoiar a gestão de requisitos assim como a planificação das fases seguintes do processo de desenvolvimento de software, existe um conjunto de artefactos que devem ser produzidos. Dos artefactos que deverão ser produzidos destacam-se: (1) glossário de termos do sistema; (2) listagem de funcionalidades agrupadas por subsistemas e/ou versões e respectivas regras e negócio; (3) plano de atribuições (listagem de funcionalidades atribuídas aos diferentes elementos da equipa de projecto); (4) listagem dos casos de uso; (5) pedidos de alterações; (6) propostas; (7) lista de riscos; (8) lista de prioridades; (9) relatório de progresso; e (10) plano de testes. Deste modo, a informação recolhida sobre os requisitos serve de base a actividades de **gestão de projecto**.

Capítulo 4

4 Ferramenta de suporte proposta

É fundamental que o processo de gestão de requisitos seja suportado por ferramentas CASE. Para dar suporte à metodologia proposta, foi delineada uma ferramenta genérica de gestão de requisitos. A proposta integral encontra-se em [Moreira2002]. Neste capítulo apresentam-se os aspectos essenciais da ferramenta delineada. A maior parte destes aspectos pode ser vista como requisitos da plataforma.

4.1 Aspectos gerais

Existem alguns aspectos que devem ser contemplados por uma ferramenta de suporte à metodologia proposta, como por exemplo: (1) capacidade de decomposição, estruturação e reutilização de requisitos e casos de uso; (2) possibilidade de tipificação de requisitos; (3) existência de glossário de termos com entidades parametrizáveis e reutilizáveis; (4) processo de validação; (5) relações de rastreabilidade; (6) suporte à derivação de casos de testes; (7) produção automática de manuais de utilizador; (8) colaboração; e (9) suporte à gestão de projecto (e.g. gestão de riscos, prazos ou custos).

4.2 Projectos e equipa de projecto

Na fase de concepção faz sentido falar em **projecto** e **equipa de projecto**. Os requisitos devem ser criados no âmbito de um projecto. Por conseguinte, a ferramenta deve ser multi-projecto, sendo que para cada projecto deve ser possível definir propriedades como (Figura 4.1): (1) a sua designação; (2) os objectivos alvo (deverão ser satisfeitos por todos os requisitos, i.e. não deve existir nenhum requisito que vá contra um objectivo alvo); e (3) data de início e conclusão (uma primeira aproximação). Deve ser possível definir os subsistemas, *releases* e equipa de projecto.

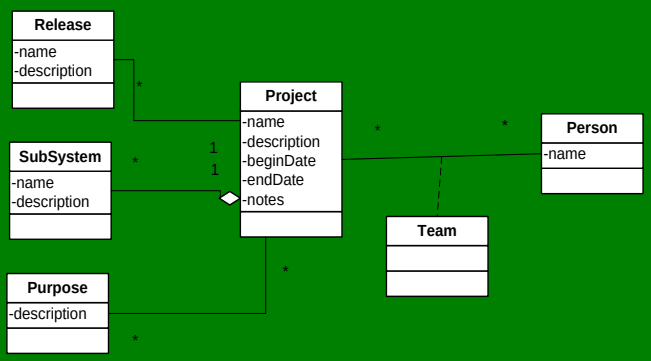


Figura 4.1 – Estrutura de projectos e respectiva equipa de projecto

4.3 Glossário de termos

Deve existir um glossário de termos que permita a definição de entidades informacionais (termos com propriedades). O nome destas entidades deve poder ser utilizado nas descrições textuais dos requisitos. A cada termo devem ser associados termos relacionados (sinónimos) que permitam a detecção de inconformidades nas descrições textuais dos requisitos. Por exemplo, num Sistema Gestão Comercial, poderíamos considerar como entidades informacionais: factura, cliente, vendedor ou armazém. Na Figura 4.2 encontra-se a estrutura do glossário de termos.

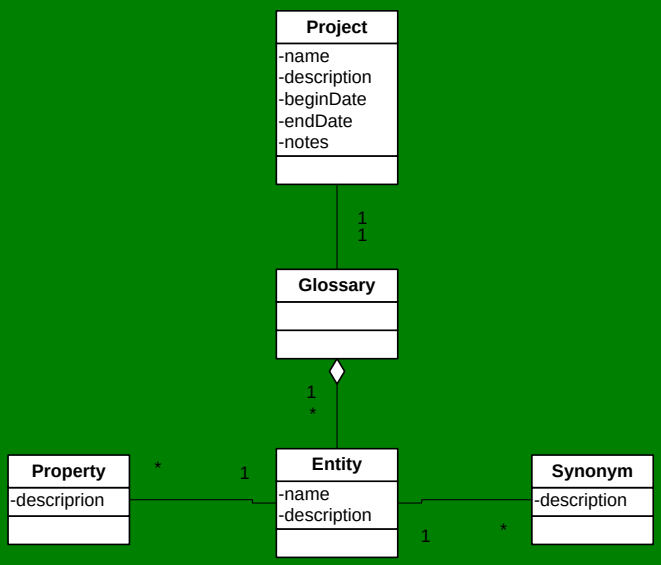


Figura 4.2 – Estrutura do Glossário de Termos

4.4 Requisitos e regras de negócio

Os atributos dos requisitos devem ser os referidos na metodologia proposta. A estrutura básica de um requisito encontra-se representada na Figura 4.3. Deve ser possível utilizar explicitamente termos do glossário de termos para a designação e descrição textual dos requisitos. Para além disso deve existir um mecanismo de detecção de inconsistências entre as palavras escritas e os termos do glossário de termos, recorrendo para isso aos sinónimos associados. De acordo com a metodologia descrita, cada requisito deve estar associado obrigatoriamente a um subsistema e, opcionalmente a uma entidade do glossário de termos. Enquanto não forem definidos os subsistemas, deve ser assumido um subsistema por omissão correspondente à designação do projecto. A partir do momento em que se definem os subsistemas, os requisitos serão assinalados com uma marca de aviso para alertar o utilizador que este requisito ainda não foi "atribuído" a um subsistema específico. Deve existir um processo expedito de atribuição de subsistemas a requisitos e de associação de requisitos a subsistemas.

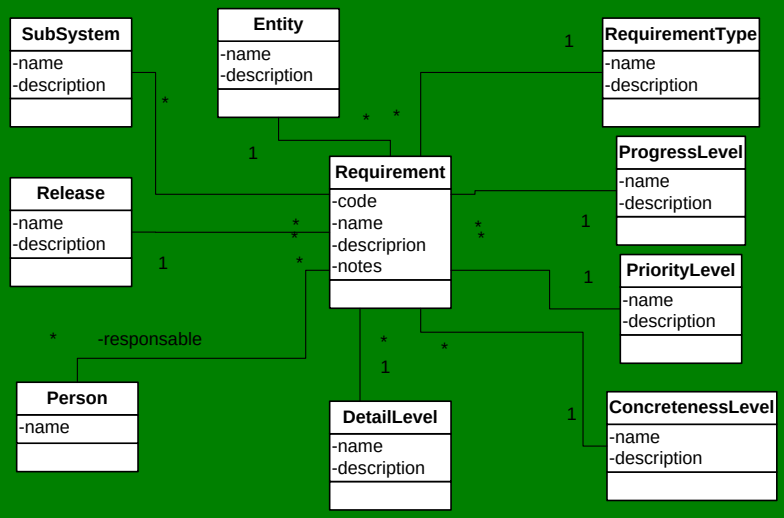


Figura 4.3 – Estrutura básica de um requisito

De seguida é apresentado um protótipo que sugere a organização dos atributos dos requisitos (Figura 4.4). Deve existir uma **hierarquia de requisitos** que possibilite ter **sub-requisitos** de requisitos e definir as relações entre requisitos. As **regras de negócio** poderão ser definidas como sub requisitos de um requisito. Deve ainda existir uma categorização de regras de negócio de modo a serem reutilizadas.

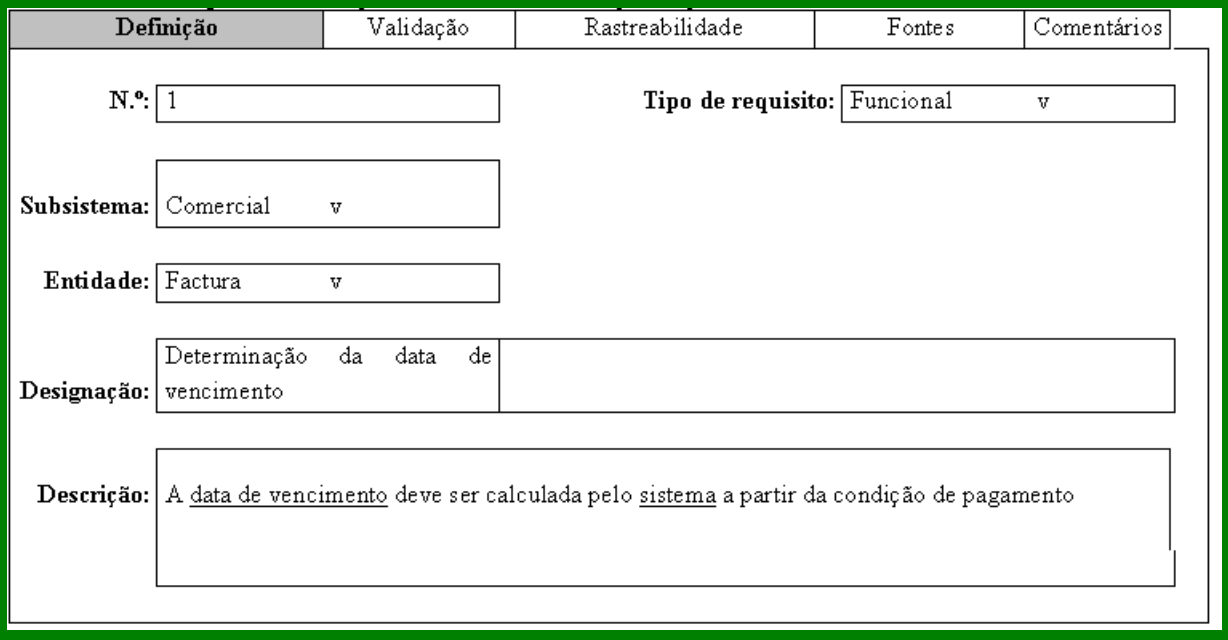


Figura 4.4 – Protótipo da organização dos atributos dos requisitos

Por exemplo, num Sistema de Gestão Comercial, poderíamos ter a seguinte parameterização de regras de negócio e hierarquia de requisitos para uma factura:

<RegNeg1>:

Designação: Intervalo de desconto

Descrição: Desconto ente 0 e 100%

Req1: Criar factura com data e desconto

Req1.1: Data anterior ou igual à actual

Req1.2: <**RegNeg1**> **–** O que acontece é que a descrição da regra de negócio parameterizada é reutilizada na descrição textual do sub-requisito podendo eventualmente ser complementada, nomeadamente com regras de excepção.

4.5 Casos de uso e de negócio

Tal como foi apresentado na metodologia proposta, os casos de uso devem ser vistos como uma extensão de um requisito, devendo por isso conter todos os elementos referidos na estrutura dos requisitos. Na Figura 4.5 encontra-se a representada a hierarquia de requisitos e a estrutura dos casos de uso. Na Figura 4.6 encontra-se um protótipo que sugere a organização dos atributos de um caso de uso.

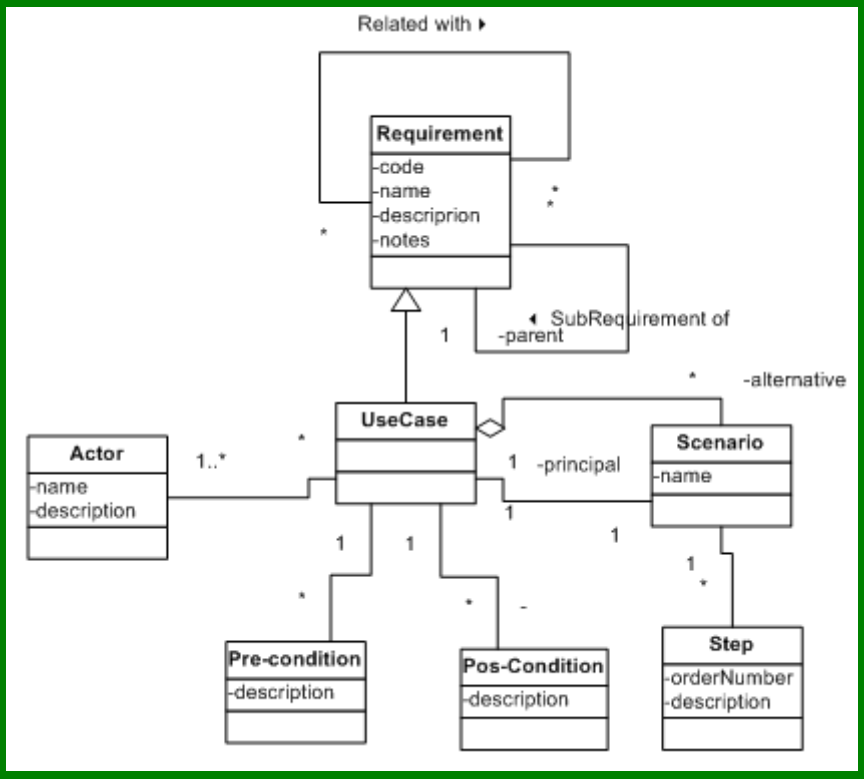


Figura 4.5 – Estrutura de um Casos de Uso

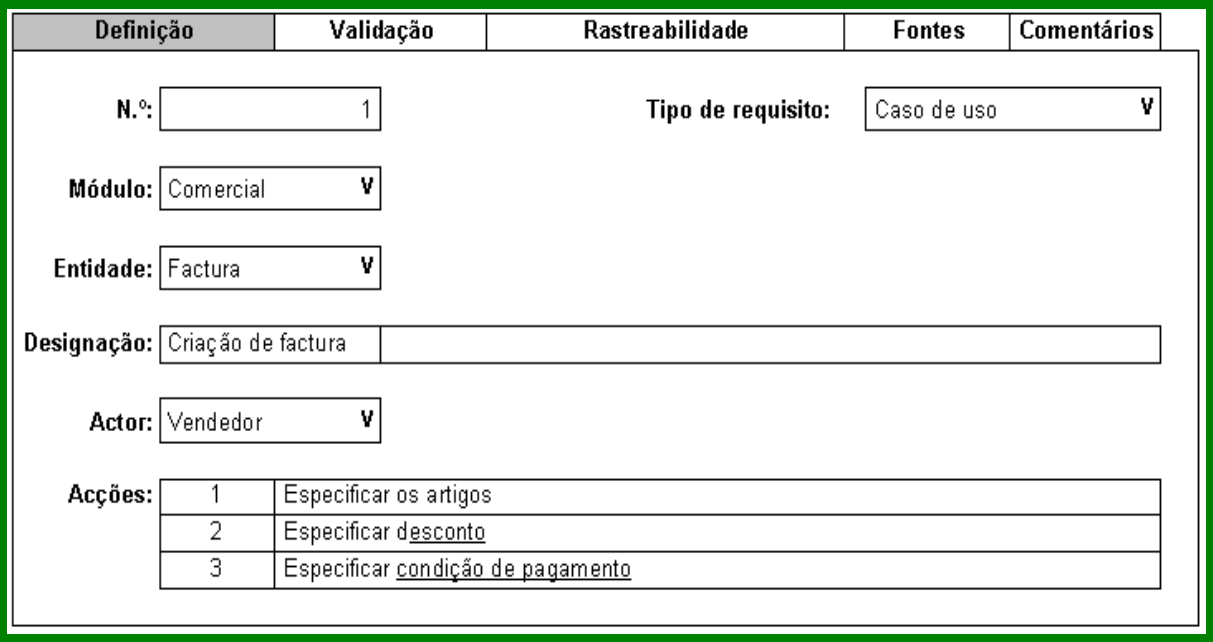


Figura 4.6 - Protótipo da organização dos atributos de um caso de uso

4.6 Requisitos diversos

Validação e teste de requisitos

Para suportar a validação e teste de requisitos, deve existir um mecanismo genérico de definição de critérios de validação e teste e respectivos valores possíveis (Figura 4.7), devendo ser depois possível classificar os requisitos em cada um destes critérios, de acordo com os valores definidos.

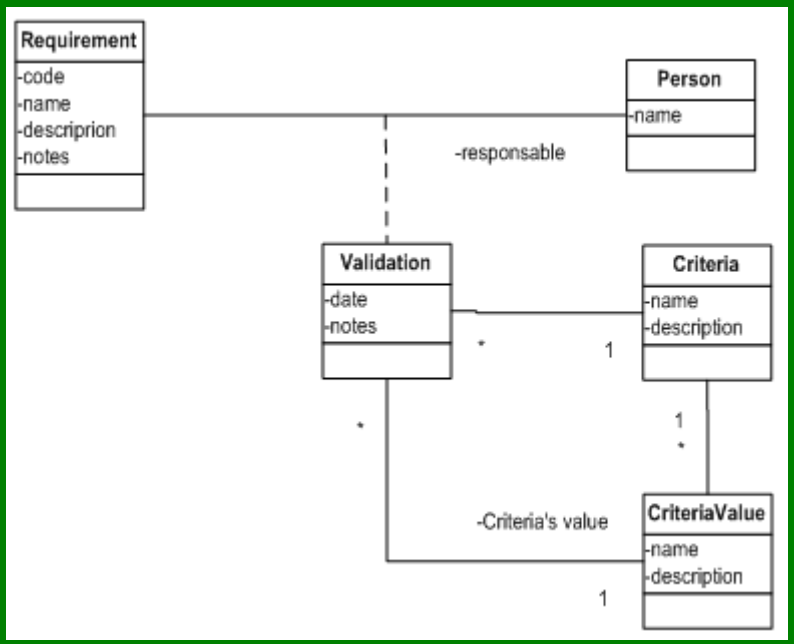


Figura 4.7 – Validação e teste de requisitos

Artefactos

A capacidade de *reporting* é um aspecto importante numa ferramenta de gestão de requisitos. Deve ser possível obter os artefactos da fase de concepção referidos na metodologia proposta, os quais devem ser disponibilizados aos diversos intervenientes (Figura 4.8).

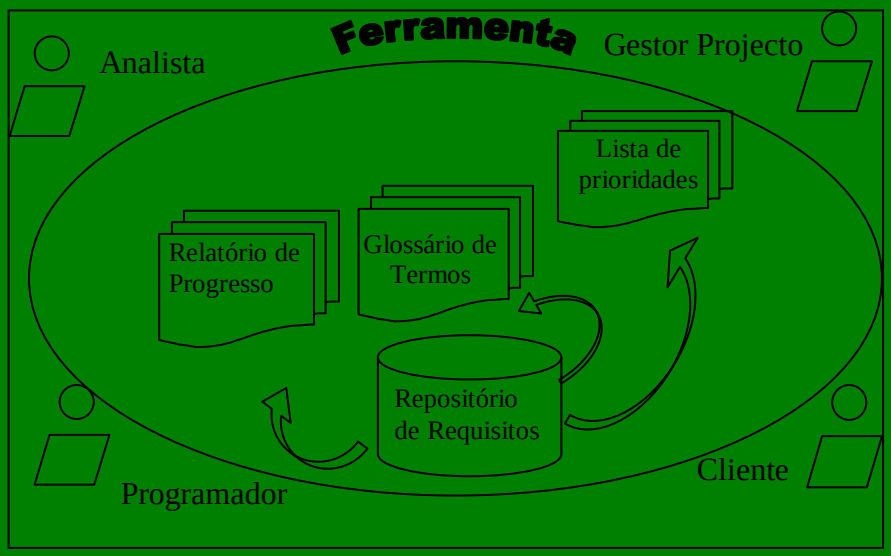


Figura 4.8 – Alguns artefactos que devem ser disponibilizados aos diversos intervenientes e que devem ser obtidos a partir do repositório de requisitos

Alteração de requisitos

A alteração de requisitos deve desencadear, em função do seu estado, um mecanismo de aviso aos utilizadores relacionados com esse requisito. O aviso poderá ser via e-mail e/ou por alerta quando o utilizador entra no sistema.

Fase de testes de software

Os requisitos e casos de uso deverão providenciar a informação necessária à definição dos casos de teste da fase de teste de software. Deve, por conseguinte, existir um mecanismo de geração de casos de teste. Os casos de teste obtidos deverão ser referenciados no conjunto de entidades de fases posteriores do processo de desenvolvimento de software que foi referido na estrutura dos requisitos.

Geração do manual de utilizador

A partir do repositório construído deve ser possível gerar o esqueleto do manual de utilizador Os subsistemas deverão ser a base da estrutura do manual: cada subsistema deve corresponder tipicamente a um capítulo. Os requisitos, casos de uso e o glossário de termos (particularmente as entidades informacionais) devem constituir o ponto de partida para a criação, respectivamente, da lista de funcionalidades, de *walkthroughs* e do glossário.

Forward & Reverse Engineering

Deve ser existir um processo de ***forward engineering*** que permita a identificação de candidatos a classes da fase de análise do desenvolvimento, a confirmação das classes efectivas e a exportação da definição das classes para um formato universal como o XML. A existência deste formato facilitará a interoperabilidade com ferramentas de modelação como o Rational Rose, o Microsoft Visio ou o ERWin. Seguem-se as principais heurísticas a serem consideradas neste processo com vista à obtenção de um esqueleto do glossário de classes:

- Nomes são candidatos a **entidades**.
- Propriedades de nomes são candidatos a **atributos** das entidades correspondentes.
- Verbos são candidatos a **operações**.
- Verbos nas frases são candidatos a **associações**.
- Entidades são candidatos a **classes**.
- Entidades com atributos comuns são candidatas a serem **classes herdadas**.

Poderá ser interessante analisar a possibilidade de suporte de ***reverse engineering*****,** passando de classes para entidades informacionais e de diagramas de sequência e actividades para casos de uso.

Interoperabilidade

A ferramenta deve possuir idealmente uma estrutura suficientemente genérica para permitir:

- Interoperabilidade com ferramentas de especificação de casos de uso, como o Rational Requisite PRO.
- Interoperabilidade com ferramentas de modelação como o Rational Rose, o Microsoft Visio ou o ERWin.
- Interoperabilidade com ferramentas de gestão de actividades como o Microsoft Project que permita criar tarefas a partir dos requisitos e dos recursos alocados.
- Interoperabilidade com ferramentas de gestão de alterações (Rational ClearQuest*)*, gestão de informação de testes de software (Rational TestManager) e automatização de relatórios e documentação (Rational SODA*)*.

Outros requisitos

De seguida são apresentadas outros requisitos da ferramenta:

- Os utilizadores devem trabalhar no contexto de um projecto. Os administradores devem poder trabalhar num âmbito geral tendo acesso a todos os projectos, o que permite uma análise multi-projecto. Por exemplo, para adicionar trabalhar sobre um atributo de alguns projectos, deve ser possível fazê-lo sem ter de mudar para o contexto de cada um dos projectos.
- A hierarquia dos projectos por *releases* e subsistemas deve ser visível e navegável.
- Uma larga gama de informação deverá ser parametrizável, designadamente: tipos de requisitos (e.g. funcional, não-funcional, de desenvolvimento), níveis de concretização, detalhe e prioridade de requisitos.
- Deve ser possível obter uma matriz de rastreabilidade onde se identifiquem as relações entre requisitos.
- O repositório da ferramenta deve poder ser armazenado idealmente em diferentes suportes (e.g. Microsoft Access, Oracle ou Microsoft SQL Server), devendo a ferramenta, por conseguinte, usar interfaces abertas (e.g. ODBC) para acesso ao repositório.
- Deve ser possível trocar comentários ou questões entre os vários membros da equipa de projecto definida. Os comentários e questões devem ser associados a um requisito específico, a um subsistema ou a um projecto em geral.
- Deve existir um mecanismo de notificações (e.g. avisos de alterações ou pedidos de validação). As notificações devem ser enviadas por e-mail e/ou apresentadas imediatamente quando o utilizador entra no sistema.
- Os requisitos devem poder ser comunicados entre os elementos da equipa de projecto para entre outras coisas serem ser validados.
- O repositório central de informação (onde se encontram requisitos, prioridades e prazos, por exemplo) deve ser acessível via Web para qualquer tipo de tarefa (e.g. criação, consulta, alteração e eliminação).

Capítulo 5

5 Ferramenta desenvolvida – ProjectPRO

Para concretizar alguns aspectos da ferramenta genericamente delineada no Capítulo 4 foi desenvolvido "ProjectPRO". Este nome deriva da componente de gestão de projectos que a ferramenta procura disponibilizar, baseado na ideia de ter o repositório de requisitos como o centro do processo de desenvolvimento de um projecto. Neste capítulo são abordadas as metodologias e técnicas de desenvolvimento e a arquitectura de software, assim como alguns mecanismos importantes e funcionalidades desenvolvidas.

5.1 Metodologias e tecnologias de desenvolvimento

O desenvolvimento do ProjectPRO seguiu uma abordagem iterativa e incremental. Foram realizadas duas iterações. O primeiro protótipo incluiu as funcionalidades básicas, como a definição das entidades base (projectos, *releases*, subsistemas, tipos de requisitos, e afins), a definição de requisitos e os agentes, não existindo ainda o portal com gestão de utilizadores. O segundo protótipo incluiu as restantes funcionalidades e foram estendidas as funcionalidades do primeiro protótipo. Foi então desenvolvido o portal, numa primeira fase sem a gestão de utilizadores, o mecanismo de perfis e permissões, tendo sido implementado na última fase a gestão de utilizadores e preferências, assim como os relatórios da aplicação.

No desenvolvimento do ProjectPRO, as principais ferramentas utilizadas foram: o Microsoft Visual Studio .NET para a programação, o Microsoft Visio para a modelação da aplicação e da base de dados, o Microsoft SQL Server onde são armazenados os dados da aplicação e o Crystal Reports, para a elaboração dos relatórios produzidos pela aplicação. Foi ainda utilizado Microsoft Agents (MSAgent) para a integração dos agentes de interface que auxiliam a utilização da aplicação.

Os modelos de classes e dados foram modelados recorrendo ao Microsoft Visio. A passagem do modelo de dados para o Microsoft SQL Server foi feita através do mecanismo *Update Database Wizard* do Microsoft Visio. São utilizadas duas bases de dados: a base de dados do portal e a base de dados aplicacional. Na base de dados do portal encontra-se as entidades relacionadas com a gestão de utilizadores, preferências e de conteúdos apresentados a cada utilizador. Na base de dados aplicacional encontram-se as entidades relacionadas com a gestão de projectos, requisitos, recursos humanos, perfis e permissões dos intervenientes, de acordo com as diferentes *releases* e subsistemas dos projectos. O modelo de dados completo é constituído por 47 entidades e encontra-se no Apêndice A.

No Apêndice B encontra-se a estrutura da *solution* do Microsoft Visual Studio .NET utilizada durante o desenvolvimento, onde se pode ter uma ideia da organização dos *packages* desenvolvidos (dos quais `Framework, Project e Requirements` são os principais, Figura 5.1).

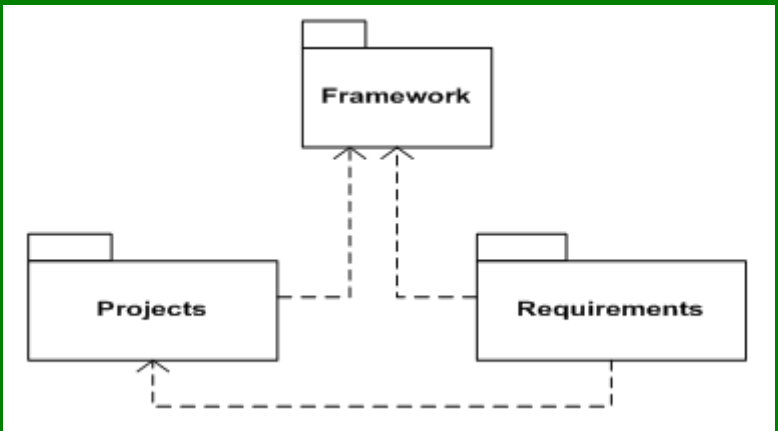


Figura 5.1 – Diagrama de packages do ProjectPRO

5.2 Arquitectura de software

O ProjectPRO é uma aplicação Web desenvolvida sobre a .NET Framework, tendo sido as principais tecnologias e linguagens utilizadas: SQL Server, C#, VBScript, JavaScript, HTML e ASP.NET, que disponibiliza tecnologia recente no que respeita a páginas dinâmicas e *web form's*.

Toda a administração e utilização são realizadas através de um *web browser* de uma forma simples e rápida, dispensando qualquer tipo de instalação de componentes de software nas máquinas dos utilizadores. O *web browser* pode ser de diferentes fabricantes porque a .NET Framework efectua a tradução automática para *browser's legacy*. É necessário então que o servidor ProjectPRO providencie acesso permanente à Internet com IP fixo para que seja acedido fora da sua intranet. Deste modo, os utilizadores podem aceder ao sistema a partir de qualquer cliente Web e com um único ponto de acesso, ou seja com uma única instância do ProjectPRO. As principais vantagens deste modelo residem no facto de os utilizadores nunca terem de se preocupar com os aspectos tecnológico, tais como instalação, licenciamento, configuração, cópias de segurança, ou actualização de novas versões. O modelo exige que a arquitectura computacional de suporte apresente níveis de desempenho e escalabilidade de acordo com número de potenciais utilizadores.

5.2.1 Camadas Aplicacionais

O ProjectPRO tem uma arquitectura de quatro camadas: camada de acesso a dados (*Data Access)*, camada de lógica de negócio (*Business Rules*), camada de fachada (*Business Façade)* e camada de apresentação *(Web User Interface*). Estas camadas permitem ter desempenho e evolução superiores. De acordo com o sugerido pela Microsoft, existe um nível de *System Framework*, onde se encontram os objectos partilhados pelas diferentes camadas (Figura 5.2).

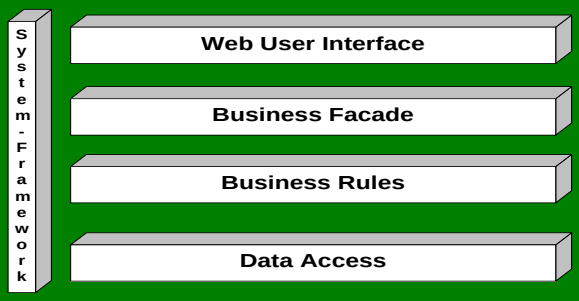


Figura 5.2 – Camadas aplicacionais do ProjectPRO

Ao nível do *System Framework* existem três classes principais para cada uma das entidades do modelo de dados:

- `<Nome-da-entidade>DataSet`: *Strong Typed Dataset* com o mapeamento das relações "pai-filho" da entidade.
- `<Nome-da-entidade>Table`: *Strong Typed Data Table* para a entidade.
- `<Nome-da-entidade>Row`: *Strong Typed Data Row* para a entidade.

Camada de Apresentação (*Web User Interface*)

Esta camada consiste num conjunto de interfaces que são responsáveis pela apresentação dos dados ao utilizador. A interface com o utilizador trata principalmente dos aspectos de visualização de informação. Todos os pedidos de informação são efectuados à camada de fachada. Com este nível de abstracção torna-se muito mais simples realizar a manutenção da interface com o utilizador.

São utilizados os *web forms* do ASP.NET com *code-behind*. Isto significa que o código das páginas ASP.NET não é colocado directamente na página, mas sim numa classe que é utilizada por cada uma das páginas. De notar que este modo de programação em ASP.NET facilita imenso a manutenção futura de código. Ao invés de se trabalhar num único ficheiro que contém *tags* HTML juntamente com código pode-se separar nitidamente os dois. O servidor utilizado é o Internet Information System (IIS).

Nesta camada foram bastante utilizados os *web controls*, que permitem ter uma interface integrada e bastante reutilizável. Existem três tipos de *web controls* associados às entidades principais: um para pesquisa sobre as entidades, um para a sua edição e outro para a edição das entidades descendentes. Existem ainda outros controlos mais genéricos como *Lookup* e o *Toolbar*. O *Lookup* permite a selecção de entidades sob a forma de lista de valores e que permite a criação e edição da entidade seleccionada na lista de valores, abrindo uma nova página no portal. O *Toolbar* disponibiliza botões para criação, edição, eliminação e pesquisa. Estes controlos são reutilizados em diferentes *web pages*. Deste modo a gestão de alterações a nível de interface é elementar, devido à reutilização destes controlos.

Camada de Fachada (*Business Facade*)

Esta camada disponibiliza uma interface consistente aos objectos do negócio subjacentes, procurando isolar o cliente das mudanças à lógica subjacente do negócio. Recebe o input da camada de apresentação (*WebUI*), usa a camada do acesso a dados para acessos de leitura e passa os pedidos à camada de lógica de negócio para acessos de escrita. É aqui que se encontram os métodos mais elaborados.

Nesta camada, existe também uma classe (`Nome-da-EntridadeBusinessFacade`) para cada entidade do modelo de dados com:

- Métodos de Suporte Transaccional, tais como *Begin Transaction*, *Commit Transaction* e *Rollback Transaction*
- Métodos frequentemente utilizados, por exemplo, pela Camada de Apresentação: tais como métodos que medeiam com os métodos de *Load*, *Update*, *Delete* e validação da camada de lógica de negócio.

Camada de Lógica de Negócio (*Business Rules*)

Esta camada contém a lógica da aplicação, não sendo necessário qualquer conhecimento acerca da base de dados, nem da estrutura da informação nem da forma como as consultas são realizadas. Como a camada com que interage é a de acesso a dados, tudo o que tem que conhecer é quais os métodos por ela disponibilizados e como lhes aceder. Para além disso, a interacção com a base de dados para uma camada distinta permite à lógica de negócio abstrair-se de onde realmente vem a informação, podendo até os parâmetros dos objectos que lhe são retornados ser resultado de bases de dados distintas.

Note-se que para conseguir total interoperabilidade entre estas duas camadas todos os métodos da camada de acesso a dados devem ser disponibilizados como *web services* (mecanismo que não foi implementado), para que possam ser acedidos com facilidade de qualquer outra máquina e permita que a plataforma para desenvolvimento da camada de lógica de negócio possa ser distinta da de serviços de dados.

Nesta camada, existe uma classe (`Nome-da-EntridadeBusinessRules`) por cada uma das entidades do modelo de dados que contém essencialmente:

- Métodos de *Load*, *Update* e *Delete*
- Métodos de validação de regras de negócio: expressões regulares, campos únicos e a dimensão de campos, por exemplo.

Camada de Acesso a Dados (*Data Access*)

Contém os dados que são necessários à aplicação e permite operações CRUD (*Creation*, *Retrieval*, *Update* e *Delete*) sobre os dados das bases de dados. Ao exterior são oferecidas capacidades de execução de consultas e actualizações sobre os dados. O Sistema de Gestão de Base de Dados (SGBD) usado foi o Microsoft SQL Server, tirando partido das potencialidades dos *stored procedures*. Foram definidos *stored procedures* de *Insert*,

Load, *Update* e *Delete* para cada uma das entidades do modelo de dados. São vastamente utilizados os *Strongly Typed Data Sets* disponibilizados pelo ADO.NET para manipulação de dados em memória desligados da base de dados. A definição das *connection strings* para as bases de dados encontram-se parameterizadas no ficheiro *web.config*, localizado na raiz da directoria virtual do ProjectPRO.

Nesta camada existe uma classe para cada uma das entidades do modelo de dados:

- *Nome-da-Entidade*`DataAccess`: Encapsula o acesso à base de dados relacional recorrendo a *Data Adapters* (para Microsoft SQL Server) com suporte transaccional. Contém métodos de *Load*, *Update* e *Delete*, recorrendo aos *stored procedures* já referidos.

5.3 Alguns mecanismos importantes

Autenticação

Em ASP.NET existem vários modos de o fazer o processo de determinação da identidade do utilizador. O processo utilizado foi o de validação do *username* e *password* (inseridos num formulário do portal) através da informação da base de dados e recorrendo também ao mecanismo *Forms Authentication* do ASP.NET, que proporciona maior segurança e fiabilidade. De referir que a *password* dos utilizadores encontra-se codificada na base de dados, utilizando o algoritmo de *hashing MD5*.

Tratamento de Erros

O tratamento de erros a nível dos dados inseridos nos formulários não estarem de acordo com o esperado, é feito através de dois mecanismos distintos. O primeiro é por recurso ao *JavaScript*, implementado na camada de apresentação, de forma a testar a validade dos dados do lado do cliente sem que tenha de existir qualquer interacção com as camadas subjacentes. Outro mecanismo de validação é a nível da camada de lógica de negócio, onde o objectivo é repetir a validação realizada ao nível do cliente. Esta duplicação tem como origem a necessidade de precaver a possibilidade de o cliente usar um browser sem suporte para *JavaScript*, ou por outro lado por razões de segurança dado que a validação por *JavaScript* não é difícil de quebrar. Para além disso também trata dos erros relacionados com inconsistências nos dados aquando da sua inserção na base de dados. Como resultado será reenviado o formulário ao utilizador, mantendo todos os valores onde não foram encontrados problemas intactos e acrescentando referências acerca dos campos que deram origem ao erro para que o utilizador os possa alterar.

Style sheets e imagens

As imagens e estilos utilizados na camada de apresentação podem ser facilmente alterados. Todas as imagens encontram-se na directoria *imagens*, pelo que basta substituir as mesmas por novas imagens para estas serem actualizadas no portal. A maior parte das imagens tem um nome intuitivo, pelo que é fácil identificá-las. Por exemplo, existem ficheiros de imagem distintos para a criação e edição de cada uma das entidades principais, que podem ser actualizados com imagens mais adequadas a cada uma das entidades. A maior parte dos estilos (cores de fundo e texto das páginas, de caixas de texto, tipos e tamanho de letra ou linhas de tabelas, por exemplo) estão definidos num ficheiro de *style sheet* (*ProjectPRO.css* localizado na directoria *Stylesheets*) que permite definir estilos em cascata. A estrutura deste tipo de ficheiros permite facilmente alterar os estilos definidos.

Deployment

O *deployment* para uma máquina final é extremamente simples: basta copiar os ficheiros *.aspx*, *web.config*, *.rpt* e as *DLLs* para a directoria virtual do IIS, assim como as directorias *images*, *stylesheets*, *SystemAgent* e *RequirementsAgent*. As DLLs para serem executadas precisam de estar colocadas num subdirectoria *\bin* de acordo com a estrutura de *assemblies* da .NET Framework. Para além disso o servidor IIS precisa de ter permissões de execução para essas directorias.

5.4 Requisitos e funcionalidades desenvolvidas

Na fase inicial foram definidos, de forma simples, alguns requisitos básicos a serem contemplados no primeiro protótipo. Estes requisitos foram registados no ProjectPRO, o que permitiu avaliar as funcionalidades. No Apêndice C encontram-se alguns relatórios produzidos pela ferramenta, nomeadamente o "Relatório de requisitos" e o "Relatório de validações" de um projecto de demonstração.

O trabalho desenvolvido no âmbito do projecto de desenvolvimento da cadeira de Introdução aos Agente Autónomos (IAA) foi a introdução de dois agentes de interface que dão suporte aos processos de alertas e sugestões diversas ao utilizador (Assistente Geral do Sistema) e de uniformização da descrição textual de requisitos de acordo com a informação do glossário de termos (Assistente de Definição de Requisitos).

No Apêndice D encontra-se o "Manual de Utilização do ProjectPRO", onde se encontra uma descrição completa de todas funcionalidades, do ponto de vista do utilizador.

5.5 Assistente Geral do Sistema

Este assistente dá suporte ao processo de alertas e sugestões, acompanhando toda a utilização do sistema. Quando o utilizador entra no portal do ProjectPRO, o Assistente Geral do Sistema (AsGS) aparece no canto superior esquerdo do portal e começa por cumprimentar o utilizador e dar-lhe as boas vindas. De seguida desloca-se para o canto inferior esquerdo do portal onde ficará alojado e apresenta uma dica sobre a utilização do sistema. A frase da mensagem de cumprimento inicial não é sempre a mesma. Esta faz

referência à parte do dia em que ocorre a entrada no sistema (dia, tarde ou noite) e é feito um breve comentário (por exemplo se for noite o AsGS faz uma exclamação sobre o facto de se estar a trabalhar nessa altura). A mensagem de boas vindas também varia. Estas mensagens estão definidas no ficheiro *SystemAgent.ascx* e podem ser alteradas. Da próxima vez que ocorrer um pedido ao servidor, passarão a ser utilizadas as mensagens alteradas. Este facto é importante pois não é necessário compilar qualquer código, o que aumenta a facilidade de manutenção e expansão do agente.

A zona do portal onde o agente fica alojado no início (canto inferior esquerdo) foi seleccionada estrategicamente, de modo a não distrair demasiado o utilizador e não ficar sobre o conteúdo das páginas. Contudo o agente pode ser movido sem qualquer problema para qualquer parte do ecrã.

Este agente disponibiliza cinco comandos: *Ocultar*, *Dicas* e *Sobre*, *Configurar* e *Desactivar*. Quando o utilizador proceder ao pedido de eliminação de informação e o assistente se encontrar activo, este pergunta se pretende mesmo apagar um requisito, uma vez que a informação será perdida.

O AsGS exprime emoções diferentes dependendo da personagem seleccionada para este agente. Por exemplo, no comando *Dicas*, a personagem *Merlin* surge com uma lâmpada iluminada, dando uma ideia de sugestão. Já no comando *Sobre* apresenta-se com as duas mãos juntas e baixando um pouco a cabeça, expressando a sua disponibilidade e respeito para com o utilizador.

Para entreter os utilizadores, o agente faz animações de vez em quando, dizendo frases em relação ao que está a fazer ou fazendo simplesmente perguntas retóricas ao utilizador. Estas animações dependem da personalidade de cada personagem. Por exemplo a personagem pode ler, escrever, ou simplesmente falar (entre outras animações dependentes do próprio personagem MSAgent Por exemplo a personagem *VGirl* tem um personalidade mais discreta, limitando-se a falar. As frases de entretenimento são também configuráveis no ficheiro *SystemAgent.ascx*.

A estrutura deste agente encontra-se ilustrada na Figura 5.3.

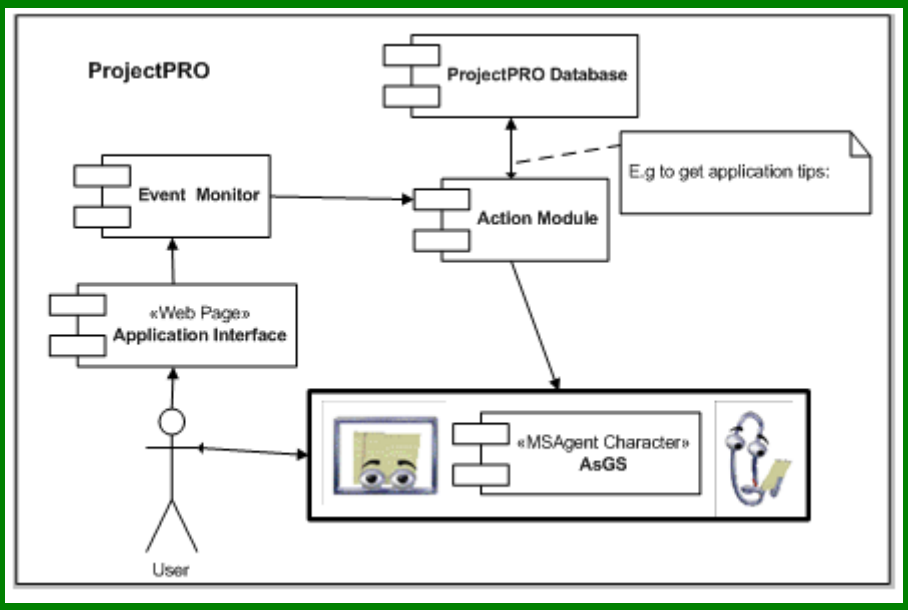


Figura 5.3 – Estrutura do Assistente Geral do Sistema (AsGS).

5.6 Assistente de Definição de Requisitos

O Assistente de Definição de Requisitos (AsDR) acompanha o processo de uniformização da descrição textual de requisitos de acordo com a informação do glossário de termos. O AsDR pode ser activado ou desactivado pelo utilizador e possibilita ainda: (1) a consulta e utilização dos termos do glossário de termos na descrição textual dos requisitos; (2) a consulta e utilização das propriedades das entidades definidas na descrição textual dos requisitos; e (3) um mecanismo básico de substituição de texto.

Este agente apresenta-se ao utilizador quando este entra na página de definição de requisitos. A primeira vez que o utilizador entra na página, o agente aparece num sítio estratégico, de modo a não tapar o conteúdo da página. Contudo, o agente pode ser movido para qualquer parte da página, sem qualquer problema. Para evitar que o agente apareça cada vez que a página é carregada, distraindo (e até mesmo irritando) o utilizador, o agente é carregado mas não é mostrado sempre. Deste modo o utilizador pode fazê-lo aparecer quando necessitar. Para aumentar a credibilidade do agente, a posição onde este estava é guardada entre pedidos da página ao servidor. Caso contrário este apareceria sempre na posição por omissão, que não era espectável, uma vez que se este desaparece numa determinada posição deve aparecer na mesma.

Tal como foi referido no AsGS, as mensagens do AsDR são configuráveis e estão definidas no ficheiro *RequirementsAgent.ascx*, podendo ser alteradas as frases correspondentes, advindo daí as vantagens já referidas.

A estrutura deste agente encontra-se ilustrada na Figura 5.4.

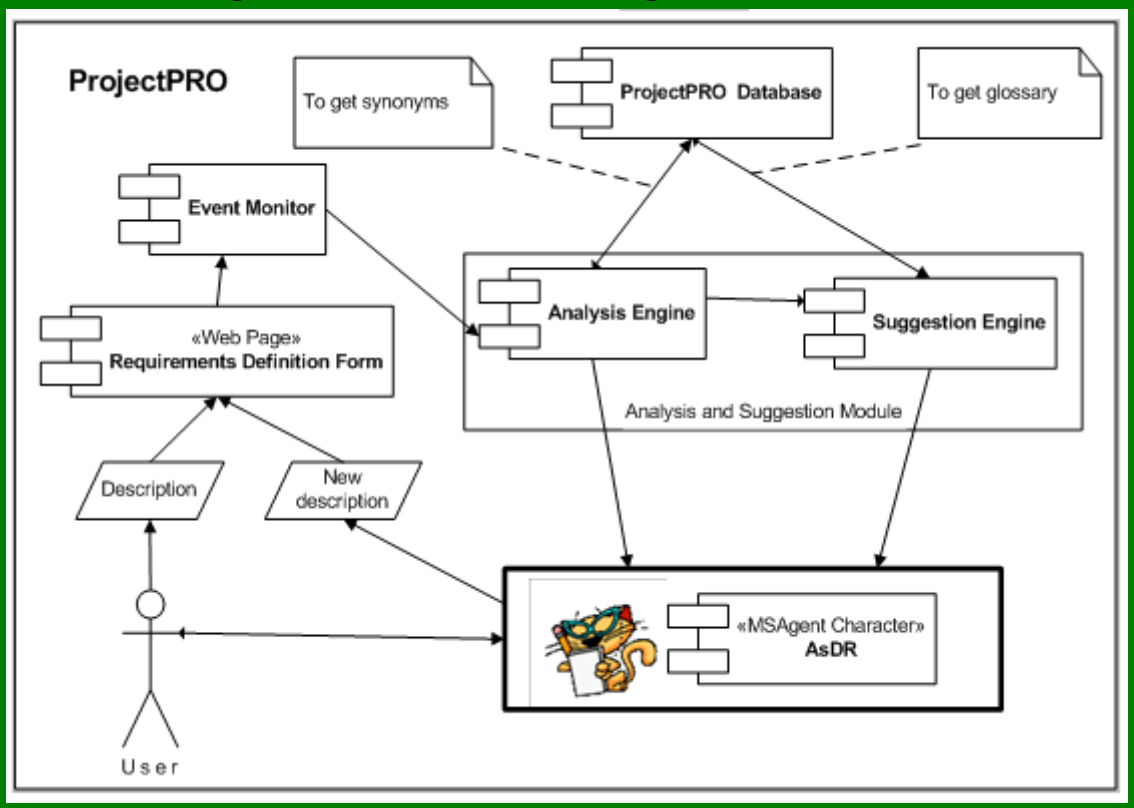


Figura 5.4 – Estrutura do Assistente de Definição de Requisitos (AsDR).

Capítulo 6

6 Conclusão

Para finalizar é apresentado neste último capítulo uma análise global do trabalho realizado, validando os objectivos iniciais. São referidos os principais contributos deste trabalho e são identificadas as questões em aberto. Finalmente é apresentada uma visão integrada com a Tese de Mestrado e alguns comentários finais.

6.1 Validação dos objectivos iniciais

Os objectivos iniciais foram concretizados de acordo com o pretendido inicialmente, tendo sido superadas as expectativas iniciais. Do trabalho realizado conclui-se de facto que um repositório de requisitos poder suportar todo o processo de desenvolvimento de software.

A metodologia proposta foi bastante bem delineada, assim como a ferramenta proposta. A utilização do glossário de termos nas descrições textuais dos requisitos permite uniformizar os conceitos envolvidos num projecto, aumentando assim a qualidade da informação. A forma de especificação de requisitos possibilita a utilização do repositório de requisitos ao longo de todo o processo de desenvolvimento de software, o que é excelente do ponto de vista de reutilização e centralização de informação.

A ferramenta desenvolvida foi desenvolvida com bastante sucesso em duas iterações. Como seria de esperar, esta a ferramenta suporta a metodologia proposta. A versão actual do ProjectPRO disponibiliza bastantes funcionalidades que não estavam previstas inicialmente. Refira-se, por exemplo, o desenvolvimento dos agentes de interface no âmbito da cadeira de IAA, em particular o Assistente de Definição de Requisitos, que apresenta um mecanismo inovador ao nível da gestão de requisitos. Os assistentes desenvolvidos mostram-se bastante úteis e importantes no contexto de uma aplicação Web e onde se trabalham com descrições textuais. A gestão de equipas de projectos e o controle de acessos a nível de *releases* e de subsistemas permite a gestão de projectos com intervenientes espacialmente distantes, uma vez que o ProjectPRO pode ser acedido a partir de um simples *web browser*.

A utilização do ProjectPRO para fazer o registo de alguns dos próprios requisitos permitiu realizar testes e experimentar a usabilidade da ferramenta. O portal ProjectPRO constitui também um elemento bastante importante, com uma gestão de utilizadores e de conteúdos totalmente personalizados. O facto de cada interveniente poder ter várias contas de utilizador permite ter configurações diferentes para cada uma das contas, mostrando-se útil quando um indivíduo está relacionado com vários projectos.

6.2 Principais contributos

A elaboração deste trabalho permitiu experimentar diversos aspectos do processo de desenvolvimento de uma aplicação, desde a sua análise ao desenvolvimento.

A análise realizada inicialmente e a delineação da metodologia proposta constituíram um marco importante, nunca antes experimentado, contribuindo, por exemplo, com conhecimento sobre a elaboração de documentos técnico-científicos.

Delinear a ferramenta de suporte à metodologia proposta foi também uma fase gratificante do trabalho, pois permitiu experimentar o processo de levantamento de requisitos e a concepção da ferramenta. Esta fase contribuiu ainda para a consolidação de ideias no que respeita ao trabalho que poderá ser realizado no âmbito do mestrado.

O desenho e desenvolvimento da ferramenta permitiram aumentar ainda mais a experiência já adquirida em dois anos de desenvolvimento de um projecto a nível profissional, também ele desenvolvido com as tecnologias utilizadas neste TFC.

O facto de ter desenvolvido o trabalho individualmente permitiu uma maior autonomia e responsabilização.

6.3 Trabalho futuro

A versão actual do ProjectPRO constitui uma base bastante desenvolvida de uma ferramenta com potencial na área da gestão de projectos, em geral, e dos requisitos, em particular. Como tal existem diversas funcionalidades que poderão ser desenvolvidas. Algumas destas funcionalidades encontram-se apresentadas na ferramenta proposta. Refira-se, por exemplo, a geração de casos de testes a partir dos requisitos ou a integração com ferramentas de modelação do domínio, no que respeita à geração de classes a partir das entidades definidas no glossário (através da extensão da estrutura destas entidades).

Os aspectos relacionados com a Internacionalização e Localização são importantes numa ferramenta com o propósito do ProjectPRO, constituindo um ponto em aberto.

Ao nível da interacção dos agentes de interface, foram detectados alguns problemas quando se encontram dois ou mais *browsers* abertos com os agentes activos. Isto deve-se essencialmente ao facto de o componente MSAgent não suportar eficazmente várias instâncias de uma personagem no mesmo ambiente de trabalho, podendo acontecer em determinadas situações a troca de diálogos ou do foco das personagens nos diferentes *browsers*. Este aspecto não é muito crítico se tivermos em conta a normal utilização do portal por parte de um utilizador.

6.4 Integração com a Tese de Mestrado

A Tese de Mestrado consistirá na continuação do trabalho realizado no âmbito do TFC. O trabalho a realizar será assim feito sobre o ProjectPRO. Poderá ser feito um estudo mais aprofundado sobre a gestão da qualidade, gestão de riscos, estimação e previsão na área da Engenharia de Requisitos e de Gestão de Projectos. Complementarmente poderão ser realizadas outras actividades, tais como: (1) fazer um caso de estudo com o ProjectPRO numa empresa de desenvolvimento de software; (2) integração com uma ferramenta no que respeita à geração de classes; ou (3) desenvolvimento de um módulo mais avançado de gestão de projectos e mecanismo de geração de artefactos.

6.5 Comentário finais

Foi fabuloso e enriquecedor ter realizado este trabalho ao longo do último ano da licenciatura. A realização deste TFC proporcionou a oportunidade de experimentar diferente aspectos abordados ao longo do curso, como metodologias, técnicas, ou até mesmo as boas práticas de programação.

Finalmente, uma nota de agradecimento ao Professor Alberto Silva por ter aceitado orientar este TFC e pela orientação do trabalho, em particular na análise e revisão dos artefactos elaborados no âmbito do trabalho.

Referências

[Anneliese90] Anneliese von Mayrhauser, **Software Engineering – Methods and Management**, Academic Press Inc., 1990.

[Cockburn98] **Basic Use Case Template – version 2**, Alistair Cockburn, 1998.

[Jacobson99] I. Jacobson, G. Booch e J. Rumbaugh, **The Unified Software Development Process**, Addison Wesley, 1999.

[Moreira2002] Vitor Moreira, **ProjectPRO – Proposta Inicial**, Instituto Superior Técnico, Lisboa, Novembro 2002. http://berlin.inesc.pt/projects/ProjectPro/ProjectPro-doc-inicial.pdf.

[Oberg2001] Roger Oberg et al, **Applying Requirements Management with Use Cases – Version 1.4**, Rational Software Corporation, 2001.

[Rito2001] **Engenharia de Requisitos**, António Rito da Silva, Acetatos da cadeira de Engenharia da Programação do Instituto Superior Técnico, 2001.

[Robertson2001] James & Suzzane Robertson, **Volere – Requirements Specification Template – Version 8**, Atlantis Systems Guild, 2001.

[Rosenberg99] Doug Rosenberg with Kendall Scott, **Use Case Driven Object Modeling with UML – A Practical Approach** , Addison Wesley, 1999.

[Silva2001] Alberto Silva, Carlos Videira, **UML - Metodologias e Ferramentas Case**, Centro Atlântico, 2001.