PROJECTO FINAL DE CURSO
LICENCIATURA EM INFORMÁTICA

Sistema de gestão de requisitos
REM

- RELATÓRIO -

Orientador:
Engº António Frazão

Alunos:
Paulo Tomé – 2100859
Ricardo Magalhães – 2100984

Ano lectivo 2006/2007

Resumo

Os requisitos de um sistema, especialmente naqueles de grande dimensão, estão em constante evolução. Isto sucede porque o problema abordado não ficou bem definido antes da especificação dos requisitos ou mesmo antes de o sistema ser implementado. Por outro lado, existe também a possibilidade de os próprios requisitos sofrerem alterações no decorrer do desenvolvimento do projecto, reflectindo evoluções tecnológicas ou mudanças na caracterização do problema.

Este Projecto Final de Curso insere-se no âmbito da engenharia de software, mais específicamente no processo de gestão de requisitos. O objectivo é desenvolver uma ferramenta de gestão de requisitos que auxilie o gestor de projecto no processo de desenvolvimento de software.

Abstract

The requirements of a system are in constant evolution, especially in large systems. This occurs because the addressed problem was not properly defined before the software requirement specification stage or even before the system implementation stage. On the other hand it is also possible for the requirements to be changed during the project development, either because of technological evolutions or changes in the features of the problem.

The current project is included in the software engineering theme, more specifically in the requirement management process theme. The goal is to develop a software tool for requirements management in order to assist the project manager in the software development process.

Agradecimentos

Queremos agradecer em primeiro lugar ao nosso professor e orientador Engº António Frazão pelo apoio, orientação e preocupação demonstradas, não só ao longo do desenvolvimento deste projecto, como desde a nossa entrada na SISCOG.

Queremos também agradecer ao Engº João Paulo Varandas pela disponibilidade e orientação prestadas durante nos últimos meses, especialmente pela sua paciência e pelas suas ideias.

Ao Engº Pedro Matos agradecemos toda a ajuda prestada, assim como o entusiasmo demonstrado nas várias fases deste projecto.

Ao Engº Rodrigo Belo agradecemos a ideia que, surgindo de uma conversa informal, serviu de inspiração ao tema deste projecto.

Queremos ainda agradecer à Engª Paula Rodrigues pelos esclarecimentos prestados em torno do tema dos testes de software.

Como não poderia deixar de ser, queríamos agradecer à SISCOG, nomeadamente à sua adminstração, cujo patrocínio permitiu a realização deste trabalho.

Por fim gostaríamos de agradecer à Dra. Patrícia Câmara Pestana pela colaboração nos testes de usabilidade do sistema, nomeadamente pela disponibilidade, paciência e sugestões.

Índice

1. Introdução

1.1. Processo de desenvolvimento de software

No longínquo ano de 1970, um engenheiro informático americano de nome Winston Royce publica um trabalho no qual descreve um modelo linear de desenvolvimento de software que mais tarde viria a ser conhecido como o modelo “em cascata”. Neste modelo, o autor identifica e isola as seguintes etapas no desenvolvimento de software: especificação de requisitos, concepção, implementação, verificação e manutenção. Desde então, foram idealizados inúmeros modelos de desenvolvimento de software, fossem variações de modelos anteriormente definidos ou abordagens mais diferenciadas, como os modelos iterativos. O que importa salientar é que foi o ínicio de um debate que prossegue até aos dias de hoje àcerca do melhor método a utilizar para desenvolver software com qualidade, de uma forma rápida e com o menor custo possível. Analisando as diversas abordagens, antigas ou modernas, verificamos que o elo comum são os seus componentes ou fases, que se mantiveram praticamente inalteradas: a *especificação de requisitos*, a *concepção*, a *implementação* e os *testes* (ver Figura 1). De facto, as fases do desenvolvimento de software, salvo pequenas diferenças de interpretação ou de denominação, são as mesmas e continuam a fazer sentido hoje em dia. Uma vez que este projecto versa sobre requisitos, e estes são um elemento comum a todas as metodologias de desenvolvimento de software, podemos portanto afirmar que o assunto em questão é independente do metodologia de desenvolvimento de software utilizada e aplicável a todas. Uma vez que o requisito é o elemento central deste trabalho, vamos também demonstrar que a sua “vida” não se resume à etapa de recolha de conhecimento e elaboração da especificação.

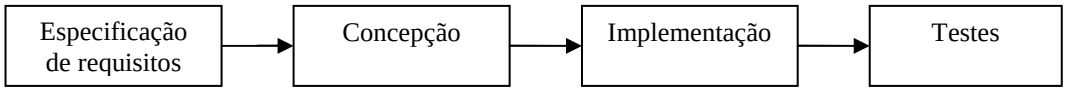

Figura 1: Fases do processo de desenvolvimento de software

1.2. O papel dos requisitos

Cada fase do desenvolvimento de um projecto pode ser perspectivada ao requisito, como iremos analisar a seguir.

Especificação de requisitos

Esta fase pode ser dividida em pelo menos duas sub-fases distintas: A recolha de informação e a especificação dos requisitos. Ao resultado da recolha de informação junto do cliente vamos denominar “requisitos de cliente” e ao resultado da especificação de requisitos vamos denominar “requisitos de software”. Daqui em diante utilizaremos estas denominações de forma a distinguir os dois tipos de requisitos.

Concepção

A concepção visa definir uma arquitectura que suporte a especificação aprovada na fase anterior. É comum que durante a fase de concepção surjam dúvidas ou problemas de concepção que requeiram uma revisão da especificação.

Implementação

A implementação representa a materialização da concepção. É a fase de construção da solução para os requisitos de software definidos. A especificação de requisitos pode, mesmo nesta fase, ser alvo de pequenos ajustes por causa de detalhes de implementação.

Testes

Nesta fase verifica-se se o produto está de acordo com o pretendido pelo cliente, que mais uma vez está expresso nos requisitos de software, e em última análise nos requisitos de cliente.

Nesta perspectiva, o requisito pode tornar-se particularmente importante no processo de desenvolvimento de software, podendo mesmo chegar-se a caracterizar o estado de um projecto pelo estado dos seus requisitos. Esta problemática que envolve a gestão de projecto e dos seus requisitos ao longo do processo de desenvolvimento de software constitui o tema do presente trabalho e é detalhada nos próximo capítulos.

1.3. Objectivos

1.3.1. Objectivo principal

O presente trabalho tem como objectivo principal o desenvolvimento de uma ferramenta de gestão de requisitos de software. De forma mais genérica, pretende-se que a ferramenta a desenvolver torne mais eficiente o processo de desenvolvimento de software ao concentrar num só sistema a informação referente aos requisitos de um

projecto e a informação referente a outro conjunto de entidades que contribuem para o seu controlo. A ferramenta destina-se a ser utilizada pelos elementos de uma equipa de desenvolvimento de software, podendo a sua utilização estar centralizada num gestor de requisitos (tipicamente o gestor do projecto) ou distribuída por diversos elementos da equipa.

1.3.2. Objectivos adicionais

Base de dados de objectos persistentes

Durante a fase de concepção foi determinado um objectivo tecnológico adicional que consistiu na utilização de uma base de dados de objectos persistentes em substituição da arquitectura tradicional de base de dados, como forma de aprofundar conhecimento e apostar numa tecnologia em expansão. Mais pormenores àcerca desta experiência podem ser consultados na secção Tecnologia utilizada.

Usabilidade

Durante a fase de implementação foi determinado outro objectivo, que consistiu em obter um elevado nível de usabilidade da ferramenta. Esta questão surgiu durante o desenvolvimento da interface com o utilizador, quando se tornou evidente que era absolutamente necessária uma cuidadosa organização do crescente número de elementos gráficos, assim como do processamento das operações a elas associadas. Mais pormenores relativos a esta questão podem ser consultados na secção Usabilidade.

2. Descrição do problema

Durante um projecto de desenvolvimento de software, é desejável que o gestor possa aferir a qualquer momento o estado do seu projecto. Os requisitos podem ser um bom indicador uma vez que são transversais a todas as etapas do desenvolvimento. Esta transversalidade traduz-se especificamente em ligações directas à concepção, ao código e aos testes. Quando orientada ao requisito, a avaliação do estado de um projecto é mais objectiva, pois pode ser mais facilmente qualificada e quantificada. A informação do estado dos requisitos pode ser muito útil para, por exemplo, detectar falhas e atrasos, para prever problemas ou mesmo para planear trabalho.

No entanto, apesar da sua importância, o controlo de requisitos é frequentemente feito de forma informal ou manual. Os processos mais comuns assentam num conjunto de documentos estáticos que necessitam de actualizações frequentes, como sejam os produzidos por folhas de cálculo ou processadores de texto. Estes documentos por sua vez também necessitam de ser controlados, nomeadamente, quanto a versões e respectivo estado, obrigando à existência de um arquivo documental hierárquico cuja complexidade pode tornar-se considerável. Como consequência, a consulta destes documentos pode ser demorada e a actualização dos conteúdos ser não só demorada como complicada e repetitiva. Por outro lado, a dispersão da informação torna os procedimentos de controlo menos eficientes e a análise do estado de um projecto excessivamente complexa e trabalhosa, por exemplo, quando para se obter determinada informação é necessário cruzar dados de diversas fontes. Basta que a actualização da informação nas diversas fontes não seja sincronizada para que a consistência dos resultados fique compometida. Esta situação torna-se mais complicada quando a informação está distribuída por diferentes equipas/departamentos da organização, cujos trabalhos não decorrem necessariamente em paralelo entre si. Esta problemática constitui o tema deste trabalho, assim como o desenvolvimento duma ferramenta que torne mais eficiente o controlo de requisitos e como consequência a gestão de um projecto.

3. Análise de requisitos

O processo de controlo de requisitos de software, enquadrado na gestão de um projecto, não é estanque à etapa da especificação de requisitos. Este processo não começa quando um requisito de software é inicialmente escrito, e não termina quando o seu texto é aceite tanto pelo cliente como pelo fornecedor. Nas secções seguintes vamos caracterizar o problema quanto às entidades envolvidas e às suas relações.

3.1. Caracterização das entidades

Vamos de seguida identificar e caracterizar as entidades consideradas relevantes para o processo de controlo de requisitos.

3.1.1. Entidades do domínio do problema

As entidades do domínio do problema foram identificadas como sendo aquelas estão directamente relacionadas com o processo de desenvolvimento de software, e estão ilustradas na Figura 2.

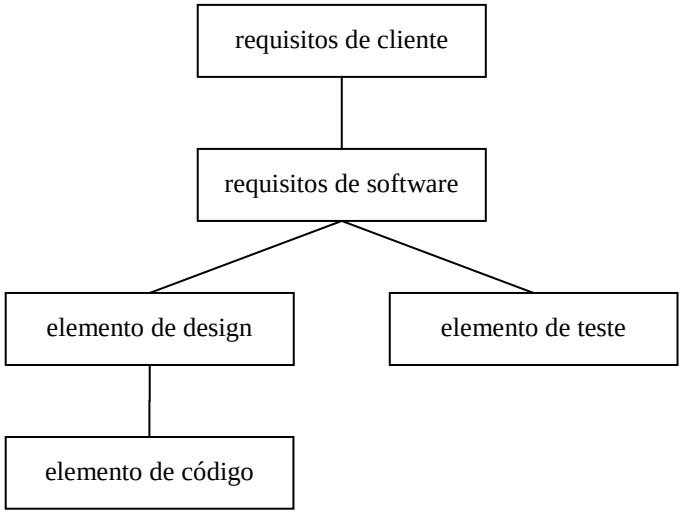

Figura 2: Modelo preliminar

Requisitos de cliente

Antes de existirem requisitos de software, existem requisitos de cliente que são a expressão das necessidades ou desejos do cliente. Estes requisitos são por natureza tendencialmente genéricos e relativamente informais na forma de escrita. Esta informação é geralmente discutida entre cliente e fornecedor, passando por diversas fases até se atingir um acordo e definir uma versão final. A versão final deste

documento encontra-se normalmente integrada num contrato celebrado entre cliente e fornecedor para desenvolvimento do software. Em alguma situações o gestor do projecto é parte interveniente na discussão dos requisitos de cliente, outras vezes recebe apenas a sua versão final após a celebração do contrato. De qualquer forma, o objectivo do projecto é dar resposta a todos os requisitos de cliente definidos no contracto celebrado entre as partes. A informação histórica de cada requisito é considerada importante e deve ser preservada para memória futura, nomeadamente as alterações que sofreu ao longo do tempo, quem as realizou e porquê. Em princípio, o projecto termina quando o cliente der como satisfeitos todos os requisitos de cliente contratados.

Requisitos de software

Os requisitos de software são criados com base nos requisitos de cliente, transpondo o que o cliente quer para o que o sistema deve fazer. Este tipo de requisitos é específico, detalhado e sistemático, escrito duma forma bastante cuidada. Como os requisitos de cliente, os requisitos de software são normalmente discutidos entre cliente e fornecedor até se atingir um acordo e definir uma versão final. A versão final deste documento constitui a especificação, que deve no mínimo cobrir os requisitos de cliente definidos na fase anterior. Apesar da aprovação formal da especificação no início do desenvolvimento, esta está sujeita a alterações durante as etapas seguintes, à medida que forem surgindo dúvidas ou problemas. A informação histórica dos requisitos de software é também importante e deve ser preservada para memória futura, nomeadamente as alterações que sofreu ao longo do tempo, quem as realizou e porquê. Considera-se que um requisito está terminado no âmbito de um projecto quando se encontra totalmente desenhado, implementado e testado.

Elementos de design

Os elementos de design correspondem à concepção da solução para um determinado problema. Neste âmbito, cada requisito de software representa uma peça de um puzzle que precisa de ser montado. Cada peça é encarada como um pequeno problema para o qual é necessário conceber uma solução. Os elementos de design servem portanto para orientar de forma técnica a implementação de requisitos de software. Um elemento de design corresponde usualmente a um documento ou a uma secção de um documento que representa uma funcionalidade, sendo que uma funcionalidade corresponde normalmente a um conjunto de requisitos. A informação se um requisito está ou não concebido é considerada importante para o gestor de projecto.

Elementos de código

Os elementos de código identificam partes do código criadas para dar resposta à concepção dos requisitos de software. Pode corresponder a um valor, uma função, um ficheiro, ou outro qualquer elemento da implementação. A informação se um requisito está ou não implementado é considerada importante para o gestor de projecto.

Elementos de teste

Os elementos de teste descrevem casos de teste para a aplicação. Quando um projecto é orientado ao requisito, os testes geralmente também o são. A vida de um teste tem habitualmente duas fases, o desenho e a execução. Da execução resultam naturalmente dois estados possiveis, o aprovado ou o reprovado. Esta informação é importante para o gestor de projecto.

3.1.2. Entidades auxiliares

Foram identificadas outras duas entidades que, embora não façam parte do domínio do problema, são importantes para a estruturação da informação dos requisitos e para o seu controlo.

Definições

Uma definição é uma entidade acessória aos requisitos. É comum que documentos de especificação contenham um capítulo ou um anexo com definições. Uma definição não é mais do que a abstração de um conceito que usado repetidamente por diversos requisitos. Através do uso de definições pode-se evitar repetições desnecessárias no texto, facilitando a leitura e a interpretação do mesmo assim como futuras alterações. Mais do que isso, existe frequentemente um ligação directa entre uma definição e um elemento de código, ficando por isso a especificação mais perto implementação. Este facto não é de particular importância para o gestor de projecto, mas é imprescindível para a correcta interpretação dos requisitos.

Grupos de requisitos

Os requisitos, de acordo com a definições formais, devem ser entidades autónomas. No entanto é aceitável que possam ter caracteristicas comuns, mais ou menos formais. Por formais entende-se o facto de serem aplicáveis a uma particular aplicação ou serem relativos a determinada funcionalidade. Por informais entende-se um contexto como seja o facto de estarem nas mesma condições de desenvolvimento ou no contexto de

uma alteração,específica. Um grupo de requisitos serve portanto para agrupar requisitos com características comuns, e é considerado como um conceito importante para o gestor de projecto.

3.2. Caracterização das funcionalidades

Foi definido um conjunto mínimo de funcionalidades que o sistema de gestão de requisitos deveria apresentar. A ideia base era eliminar o clássico arquivo de documentação relacionado com especificações, e simultaneamente integrar num só sistema um conjunto adicional de informações relacionadas com requisitos que permitisse ao gestor, duma maneira mais eficiente, avaliar o estado do seu projecto em desenvolvimento. Segue-se uma lista com as operações principais inicialmente idealizadas.

Introdução de dados

O sistema deveria permitir a introdução de novos requisitos, assim como de outras entidades independentemente da fase de desenvolvimento do projecto. Isto permitiria que a utilização do sistema acompanhasse um projecto desde o início, ou viesse a substituir outro sistema já em utilização.

Rastreabilidade de requisitos

Seria possível relacionar um requisito de software com os requisitos de cliente que lhe deram origem, com a concepção, com o código relacionado e com os testes efectuados, permitindo ao gestor de projecto ter uma visão mais clara e abrangente destas relações. Estas ligações poupariam inúmeros documentos de controlo e muito trabalho. A centralização da rastreabilidade dos requisitos tornaria mais fácil e rápida a compreensão e manutenção desta informação, por exemplo para análise de impacto de alterações.

Alteração de requisitos

O controlo da mudança de um requisito não deveria ser demasiado rígido, para que cada gestor de projecto pudesse determinar as suas regras. O ideal seria que ao gestor fosse permitido definir as suas próprias regras e determinar as permissões a atribuir aos restantes utilizadores do sistema. Isto facilitaria a integração da ferramenta nos métodos de trabalho de cada equipa de desenvolvimento, tanto a nível de fluxos como a nível de segurança.

Controlo do estado de requisitos

Seria possível controlar o estado de um requisito, saber se este estava revisto, aprovado, implementado ou testado, por exemplo. Os estados de um requisito definiriam as fases da vida do requisito. O controlo dinâmico destes estados evitaria a manutenção de vários documentos com informação cruzada, como folhas de cálculo, através da centralização da informação num só sistema. Nesta altura os estados possíveis de um requisito ainda estão por determinar.

Histórico de requisitos

Um requisito poderia ser alterado pelo utilizador, mas o sistema deveria manter um registo das alterações efectuadas. A acompanhar cada alteração de requisito deveria estar uma data, o nome do utilizador que a efectuou e a razão da alteração. Para outras entidades este registo de alterações não era necessário. Esta característica permitiria abandonar a utilização dos arquivos com inúmeros ficheiros de texto. Para fins de análise, a evolução dos requisitos poderia servir de indicador para, por exemplo, encontrar os assuntos especificados sujeitos a maior número de alterações, qualidade de escrita dos requisitos, complexidade de desenvolvimento, eficiência de equipas, entre outros. Assim sendo, o sistema deveria permitir ao utilizador consultar o histórico de um requisito.

Remoção de dados

O utilizador poderia remover requisitos do sistema, mas o seu histórico deveria ser preservado. Outras entidades poderiam ser completamente apagadas. O histórico de informação de testes, ou de concepção, não é crítico como é o de requisitos. Manter num sistema requisitos que foram removidos pode revelar-se útil para compreender a evolução dum projecto, tanto em desenvolvimento como em manutenções futuras.

Consulta de dados no passado

O utilizador poderia ver o estado do projecto numa data passada. Este tipo de informação é quase impossível de determinar num sistema convencional de ficheiros de arquivo com documentos de controlo sumarizados. Por exemplo, esta informação poderia servir para analisar, à posteriori, problemas que tenham afectado o desenvolvimento e formular estratégias de melhoria.

Agrupamento de requisitos

O utilizador poderia agrupar requisitos como quisesse, e efectuar operações sobre esse grupo. Uma vez que cada requisito seria encarado como uma entidade independente, tornava-se necessário definir âmbitos formais, como secções da especificação ou funcionalidades para os agrupar. Mas também seria útil ao gestor de projecto poder definir outros âmbitos como, por exemplo, os requisitos sob a responsabilidade de alguém, os requisitos afectados por algum problema, etc. A definição de grupos deveria ser uma operação fácil e rápida de executar e desfazer.

Exportação

O sistema, a pedido do utilizador, produziria um documento de formato a definir com informação de grupos, requisitos e definições, numa determinada data. Esta funcionalidade facilitaria a impressão de documentos como as especificações, parciais ou totais.

4. Concepção da solução

4.1. Descrição do modelo de objectos

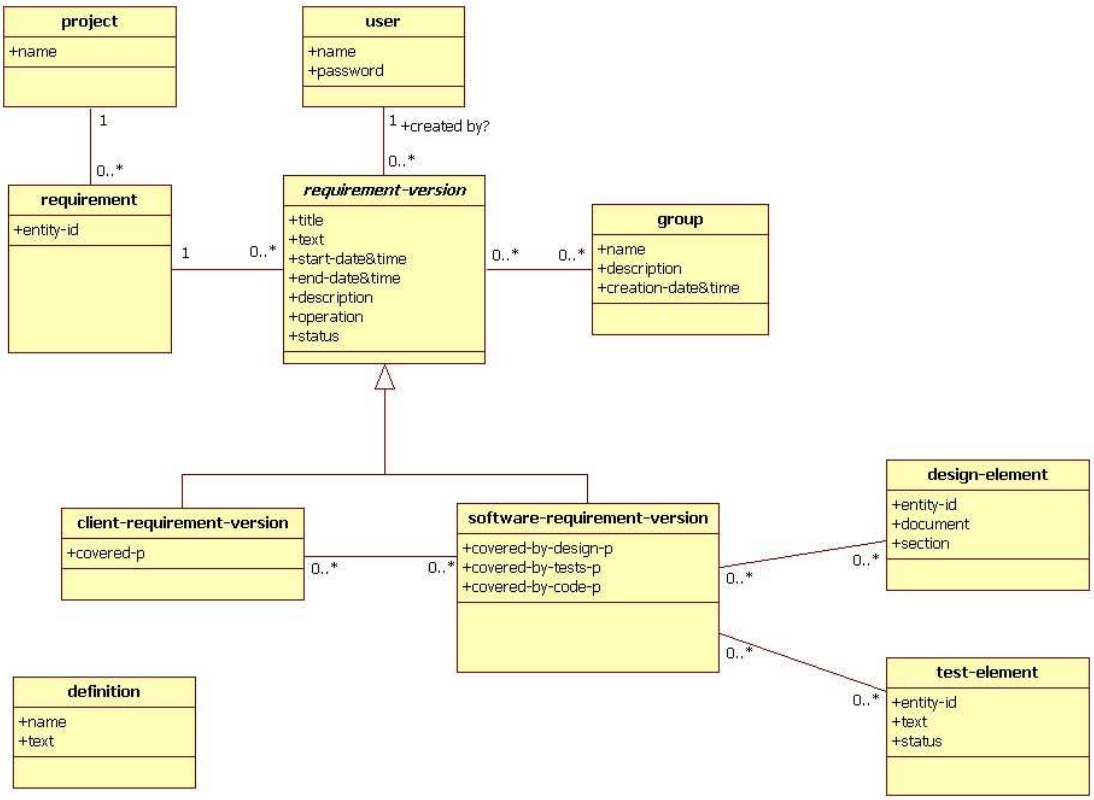

Figura 3: Modelo de objectos do Gestor de Requisitos

Utilizador

A classe *user* representa o utilizador do Gestor de Requisitos. Cada utilizador é identificado univocamente por um nome e caracterizado pelo atributo *password* em que está armazenada a palavra passe do respectivo utilizador.

Projecto

A classe *project* representa o projecto no âmbito do qual vão ser criadas e geridas as entidades que o constituem, tais como os requisitos de cliente e de software, os elementos de teste e de design as definições/conceitos e os grupos. É caracterizado por ter um atributo *name* que corresponde ao nome do projecto e que o identifica univocamente.

Requisito

Um requisito é representada pela classe *requirement* e consiste numa entidade que agrega o conjunto de todas as suas versões. Possui um atributo *entity-id* que corresponde ao identificador do requisito e está sempre associado a um projecto.

Versão de requisito

Uma versão de requisito é representada pela classe *requirement-version* e generaliza os conceitos de versão de requisito de cliente e versão de requisito de software (descritos abaixo). É caracterizado por ter os atributos *start-date&time* e *end-date&time* que, em conjunto, constituem o intervalo temporal de uma versão de requisito.

As versões pertencentes a um mesmo requisito têm intervalos temporais contíguos, isto é, todas as versões de um requisito, com a excepção da primeira versão e da versão mais recente, possuem uma versão de requisito que a antecede temporalmente e uma versão de requisito que a precede. Uma versão de requisito pode ter o atributo *end-date&time* nulo sendo neste caso considerada a versão mais recente do requisito ao qual pertence. Um requisito pode ter no máximo uma versão mais recente. Um requisito que não possua uma versão mais recente representa um requisito que foi removido.

Uma versão de requisito possui os atributos *title* e *text* que indicam respectivamente o título e o texto do requisito ao qual a versão pertence e que estão em vigor durante o intervalo temporal para o qual a versão de requisito é válida.

Cada versão de requisito está associada a um utilizador no sentido em que cada nova versão de requisito originada a partir de uma operação efectuada sobre um requisito terá sempre informação sobre o utilizador que originou essa alteração.

Uma versão de requisito é caracterizada por possuir os atributos *description*, onde é guardado um texto com um comentário escrito pelo utilizador durante a operação que originou a criação da versão do requisito e o atributo *operation*, que representa o tipo de operação que deu origem à criação da nova versão do requisito de software. Cada versão de requisito possui ainda o atributo *status* que representa uma fase no ciclo de vida do requisito a que pertence.

A classe *requirement-version* está especializada nas classes *client requirement version* e *software requirement version*.

Elemento de teste

Um elemento de teste representa um caso de teste desenhado para testar um ou vários requisitos de software. É representado pela classe *test-element* e é identificado

univocamente pelo atributo *entity-id.* Possui ainda o atributo *status* que representa uma fase no ciclo de vida do elemento de teste.

Elemento de design

Um elemento de design representa um apontador para o documento e a respectiva secção onde se encontra descrito um elemento de design. É representado pela classe *design-element* e é identificado univocamento pelo atributo *entity-id*.

Grupo

Um grupo consiste numa entidade que permite agrupar requisitos de cliente e requisitos de software num único conjunto. É representado pela classe *group* e identificado univocamente pelo atributo *name*.

Definição

Uma definição consiste numa entidade cujo texto contido no atributo *text* permite auxiliar na compreensão do significado de um conceito ou definição. É representada pela classe *definition* e é identificada univocamente pelo atributo *name*.

4.2. Princípios operacionais

Dinamismo das versões de requisito

Considera-se um requisito como a entidade abstracta que resulta do conjunto de todas as suas versões. A criação de novas versões dos requisitos resulta de alterações aos seus dados. O facto de ter sido modelada a classe *requirement-version* está relacionado com a pretensão de guardar o histórico das alterações efectuadas sobre os requisitos ao longo do tempo. Assim, a versão mais recente de um requisito é a versão que contêm a informação relativa a um requisito à data actual; todas as outras versões do mesmo requisito são consideradas versões históricas e contêm informações sobre o requisito em intervalos temporais no passado.

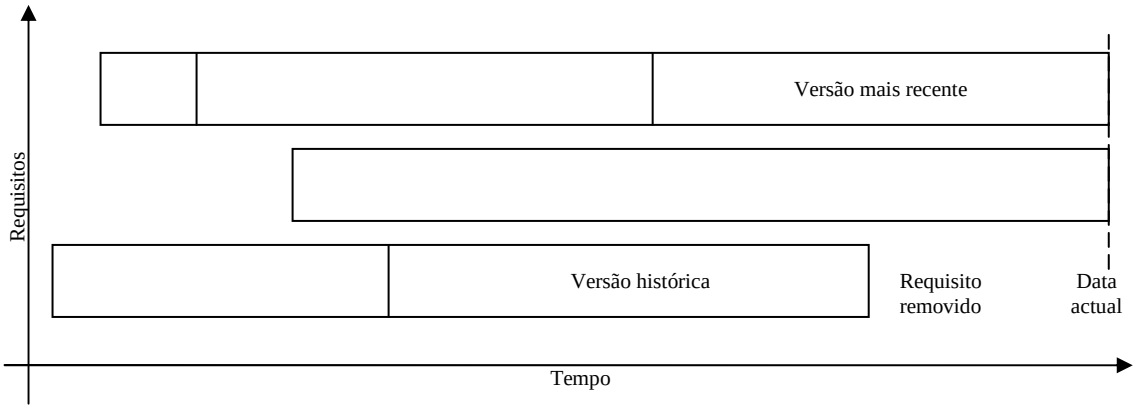

Figura 4: Versões de requisitos ao longo do tempo

Tipos de alterações a requisitos

Classificam-se as alterações feitas aos dados contidos nos requisitos em ‘alterações relevantes’ ou ‘não relevantes’. Uma ‘alteração relevante’ corresponde a uma alteração feita a um requisito que justifica a criação de uma nova entrada no histórico do requisito. Uma ‘alteração não relevante’ não dá origem à criação de uma nova entrada no histórico do requisito. Assim, as alterações relevantes criam sempre novas versões de requisitos, enquanto que as alterações não relevantes originam a alteração da versão mais recente do requisito. Como consequência existe uma correspondência entre o número de entradas no histórico de um requisito e o número de versões de um requisito.
Inicialmente consideramos que a mudança do texto ou do estado de um requisito daria origem a uma alteração relevante e que a mudança do título do requisito, do relacionamento com outras entidades ou da sua cobertura originaria uma alteração não relevante. Com a introdução da ‘operação de congelamento’ e da noção de ‘responsabilidade da alteração de um requisito por parte de um utilizador’ tivemos que alterar esta classificação. Assim, foram adicionadas à regra anterior dois casos particulares: quando um requisito se encontra no estado “congelado”, qualquer alteração ao requisito é considerada uma alteração relevante; quando o utilizador que faz uma alteração a um requisito é diferente do utilizador responsável pela última alteração realizada sobre esse requisito a alteração é também considerada relevante.

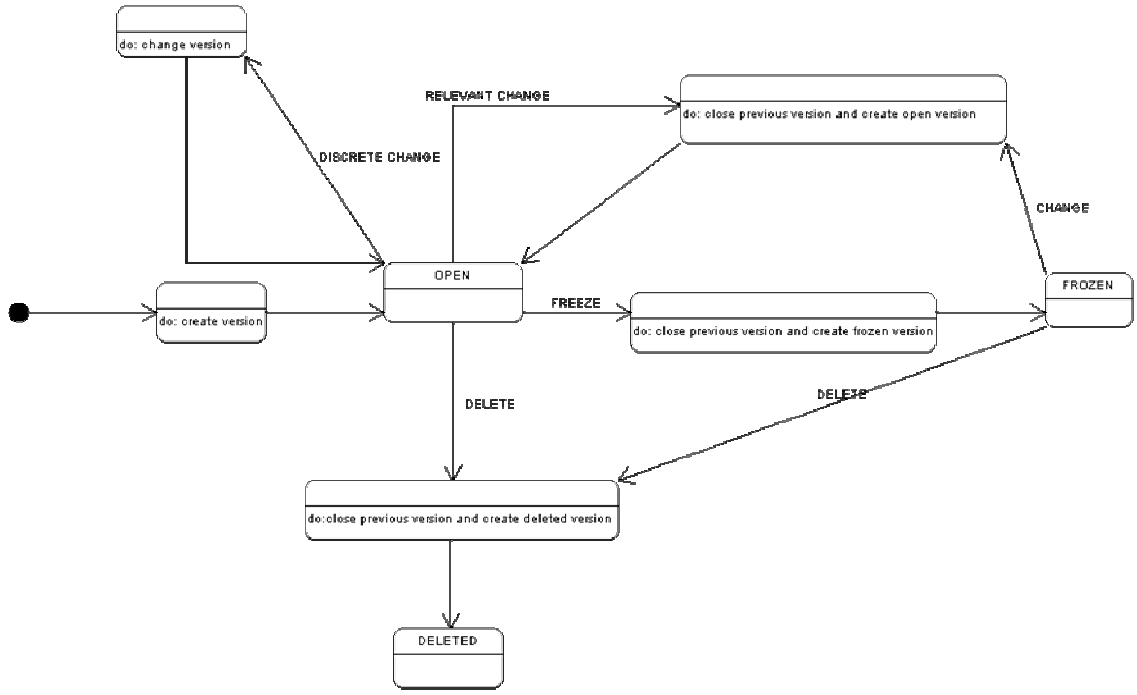

Figura 5: Diagrama de estados de um requisito

Posteriormente criamos um terceiro tipo de alteração à informação contida nos requisitos: a ‘alteração de texto’. Este tipo de alteração é exclusiva dos requisitos de software. Uma ‘alteração de texto’ é também uma ‘alteração relevante’ uma vez que resulta da alteração do texto de um requisito e também origina a criação de uma nova versão do requisito. Mas, para além disso, origina a ‘propagação da alteração do texto de um requisito de software no estado dos elementos de teste’, isto é, quando se altera o texto de um requisito de software ao qual estão associados elementos de teste cujo estado seja “Approved” ou “Failed” estes são alterados, mediante autorização do utilizador, para o estado de “Pending”.

Modo ‘versão mais recente’ versus modo ‘histórico’

No modo ‘versão mais recente’ podem-se fazer todo o tipo de operações definidas sobre os requisitos: criação, edição, remoção, congelamento e consulta. No modo ‘histórico’ apenas se podem fazer operações de consulta sobre requisitos.

4.2.1. Decisões de concepção

Interface

De início foi tomada a decisão de a aplicação ser suportada por apenas uma base de dados onde iriam ser guardados os objectos relativos a cada um dos projectos existentes mais os objectos transversais a todos os projectos (utilizadores e projectos). Mas devido

à tecnologia de base de dados de objectos persistentes utilizada ter como limitação o não permitir o carregamento de dados por critério, tal teria como consequência que ao abrir a ligação à base de dados todos os objectos fossem carregados para memória, independentemente do projecto a que pertencessem. Não sendo lógico estar a ocupar memória com objectos pertencentes a projectos não seleccionados pelo utilizador e de modo a evitar ter de fazer filtragens aos dados por projecto sempre que fosse necessário fazer uma listagem de objectos de um determinado tipo para excluir os dos outros projectos, optou-se pela criação de uma base de dados por projecto mais uma de controlo que contêm a informação relativa aos utilizadores e projectos.
Uma vez que apenas pode existir uma ligação a uma base de dados num determinado momento, a base de dados de controlo é carregada assim que a aplicação é inicializada. Após o utilizador ter feito "login" e selecionado um projecto para trabalhar, a base de dados de controlo é fechada e é carregada a respectiva base de dados do projecto.
Só é possível mudar de utilizador após fechar o projecto carregado (salvando as alterações realizadas ou não).

Porque não foi modelado o conceito de elemento de código?

Embora na classe *software-requirement-version* exista o atributo *covered-by-tests-p* e a associação à classe *test-element*, o atributo *covered-by-design-p* e a associação à classe *design-element*, existe também o atributo *covered-by-code-p* mas foi decidido não modelarmos o conceito de elemento de código. Isto porque foi considerado que não existe uma relação directa entre requisitos de software e elementos de código mas sim entre elementos de design e elementos de código. Esta é portanto uma relação indirecta, logo não modelável na abordagem por nós tomada.

Geração automática de identificadores

Para facilitar as operações de criação de novos requisitos de cliente, requisitos de software, elementos de teste e elementos de design foi decidido criar um mecanismo de geração automática de identificadores. Assim, partindo do pressuposto que os identificadores são constituídos por dois campos: um campo prefixo regular, composto por caracteres alfanuméricos e um campo sufixo exclusivamente composto por algarismos, este mecanismo gera automaticamente um conjunto de identificadores com igual número de propostas quantas o número de diferentes prefixos existem para a entidade nas quais o campo prefixo é mantido e o campo sufixo é constituido pelo sucessor do valor máximo encontrado.

4.3. Tecnologia utilizada

Como plataforma base de desenvolvimento foram usadas ferramentas e tecnologias desenvolvidas pela "Franz Inc.", nomeadamente o ambiente de desenvolvimento "Allegro CL 8.1 Free Express Edition" para Windows.

Linguagem

O sistema de informação "Requirements Manager" foi concebido usando a linguagem de programação Common Lisp Object System (CLOS). A escolha desta linguagem de programação deveu-se aos seguintes factores: 1) ser uma linguagem que suportava todos os requisitos impostos pela arquitectura por nós definida; 2) possuirmos experiência no uso desta linguagem no desenvolvimento de sistemas de informação; 3) estarmos integrados numa organização onde o CLOS é a principal linguagem de programação no desenvolvimento de software e onde existe um conjunto alargado de profissionais que poderia colaborar em futuros desenvolvimentos da ferramenta.

Interface gráfica

Para o desenvolvimento da interface gráfica foi usado o "Allegro Common Graphics" que é um componente do IDE do Allegro CL baseado numa biblioteca de funções gráficas para Windows.

Base de dados

Para implementar a persistência da informação foi decidido usar a base de dados relacional de objectos persistentes "AllegroCache 2.1.8". Esta ferramenta permite extender as capacidades da linguagem de programação orientada a objectos CLOS com as capacidades de uma base de dados relacional. Para o conseguir utiliza o conceito de persistência de objectos. Considera-se um objecto como persistente quando este existe para além do final da sessão onde foi criado. Esta tecnologia permite que as tradicionais operações de carregamento e armazenamento de informação numa base de dados sejam transparentes para o programador.

Esta solução foi adoptada por ser gratuita e de fácil integração no Allegro CL. Além disso, e apesar da sua juventude, esta tecnologia apresenta um conjunto de características interessantes tais como o bom desempenho com grandes volumes de dados (Milhões de objectos e Terabytes de dados), e gestão transparente da concorrência quando em modo servidor/cliente. Isto garante a escalabilidade dos projectos baseados nesta arquitectura.

O estudo e a utilização prática desta tecnologia revelou, no entanto, alguns aspectos que ainda podem ser melhorados. Por exemplo, as operações de procura apresentam ainda algumas limitações, e as operações de carregamento e armazenamento têm de ser totais, ou seja, não podem ser realizadas por critérios. Apesar destas restrições, consideramos acertada a escolha desta solução que trouxe uma nova perspectiva à concepção da solução e influenciou o modelo de classes definido.

5. Implementação

5.1. Camadas de software

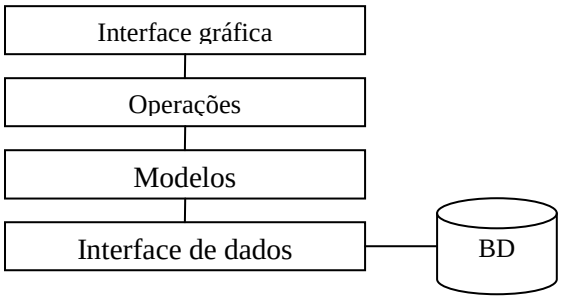

Figura 6: Arquitectura interna

Interface de dados

A camada de mais baixo nível é a camada de *interface de dados*. Esta é responsável por todos os acessos à informação guardada nas base de dados, que poderão ser uma simples leitura, uma inserção ou actualização da informação, e pela gestão das ligações à base de dados de controlo e bases de dados associadas a cada um dos projectos.

Modelos

Acima desta encontra-se a camada de *modelos* onde foram implementadas as classes definidas no modelo de classes. Esta encontra-se intimamente relacionada com a camada de interface devido à utilização de objectos persistentes.

Operações

Entre as camada dos *modelos* e a camada da *interface gráfica* encontra-se a camada de *operações*. Esta é responsável pelas operações (criação, alteração, remoção e congelamento) executadas sobre as diversas entidades existentes em cada projecto. Durante a implementação da aplicação foi estabelecido um protocolo ao nível desta camada que permitiu integrar a comunicação entre as camadas dos *modelos* e da *interface gráfica*.

Interface gráfica

A camada da *interface gráfica*, a de mais alto nível, é responsável pela interacção do sistema com o utilizador da aplicação.

5.2. Interface homem-máquina

O sistema REM foi concebido tendo em conta o perfil do utilizador a que se dirige. O utilizador tipo dum sistema com esta características tem conhecimentos de informática e pertence a uma equipa de desenvolvimento ou manutenção de software. A manipulação do sistema pode estar centralizada no gestor do projecto, ou ser partilhada com outros elementos da equipa, por exemplo developers e/ou testers. Apesar destes pressupostos, a simplicidade da interacção homem-máquina foi um objectivo a que nos proposemos atingir, sendo alvo de diversos debates e revisões ao longo do desenvolvimento do sistema. Em termos genéricos, a interface baseia-se numa janela principal que apresenta a estrutura ilustrada na Figura 7:

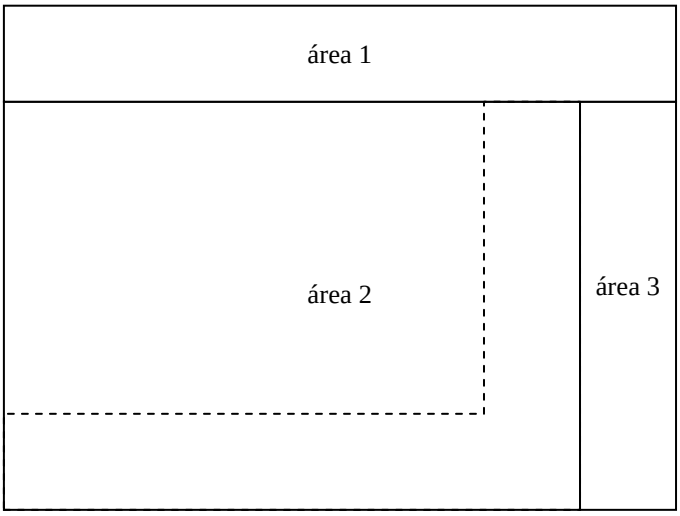

Figura 7: Estrutura esquemática da janela principal

A área 1 é destinada à identificação, selecção e filtragem dos dados contidos nas restantes áreas da janela.

A área 2 é a área dos dados. Nesta área é possível seleccionar e visualizar listagens resumidas das diversas entidades intervenientes no sistema. Dentro desta área, existe uma outra área de tamanho e formato variável (assinalada a tracejado) que permite a visualização rápida de pormenores relativos à instância seleccionada da listagem.

Por último, a área 3 é onde se encontram botões com as operações disponíveis a realizar no momento, de acordo com as condições e entidades seleccionadas.

Sobre a janela principal, e dependendo da operação chamada pelo utilizador, podem surgir outras janelas mais pequenas e simples, como por exemplo, para introdução/edição de dados.

5.2.1. Usabilidade

Foram realizados esforços para que a utilização do sistema fosse simples, prática, intuitiva e agradável. Este esforços revelam-se a vários níveis, exemplificados a seguir:

- Cada elemento gráfico da interface foi posicionado de forma a reflectir uma ordem de visualização de acordo com a cultura ocidental, da esquerda para a direita e de cima para baixo. Nesta trajectória, primeiro surge o enquadramento dos dados, depois os dados em si e por fim as operações disponíveis de realizar.
- Noutro exemplo, a sequência de carregamento da aplicação solicita ao utilizador a introdução de dados de diversos âmbitos, sem que este necessite de navegar em opções de menu para o fazer, diminuindo o tempo de carregamento da aplicação e a quantidade de operações que o utilizador tem de realizar para o concretizar.
- Noutra vertente, a disponibilidade dos botões e opções de menu varia conforme as condições do momento, para que o utilizador possa facilmente distinguir as operações que pode executar das que não pode executar. Isto permite evitar a constante produção de mensagens de erro no écran, que se torna irritante para o utilizador, não facilitando a aprendizagem do funcionamento da aplicação.
- A maioria dos botões apresenta títulos de compreensão fácil, e nos poucos que apresentam imagens foram adicionadas tooltips com descrições mais claras.
- A utilização de pequenos icons nas listas de entidades permite distinguir ou agrupar elementos de acordo com determinadas características de uma forma mais natural.
- As áreas de visualização rápida permitem navegar nas listagens e ir analisando as características mais importantes dos elementos seleccionados sem a necessidade de abrir uma outra janela.
- Dependendo do contexto, a abertura de novas janelas é feita em regime de exclusividade ou em paralelo. No regime de exclusividade apenas a janela aberta sobre a principal pode ser manipulada, contendo esta, botões de navegação para se visualizar o elemento anterior ou posterior da listagem; em modo paralelo, podem ser abertas diversas janelas em simultâneo para permitir a compararação de múltiplos elementos.

- A introdução em série de elementos do mesmo tipo foi também alvo de atenção, com a introdução de uma opção que permite ao utilizador definir a acção a tomar após a inserção de um novo elemento, como por exemplo sair ou introduzir outro.
- Outra funcionalidade interessante é a adopção de menus de right-click com as mesmas operações disponíveis nos botões e outras adicionais. Estes menus permitem a execução de certas operações sobre um elemento quando este se encontra fora do seu contexto, ou seja, quando este surge listado em janelas pertencentes a outros elementos.
- Outro exemplo é a geração automática de identificadores, como sugestão ao utilizador no processo de inserção de novas entidades. Esta funcionalidade não só garante a unicidade do novo identificador, como as sugestões apresentadas preservam a lógica das anteriormente definidas.
- Consideramos que as situações anteriormente descritas melhoram a usabilidade da ferramenta e resultam numa experiência de utilização agradável e natural.

6. Testes

Foram efectuados testes unitários ao longo do desenvolvimento da ferramenta. Cada camada de software foi testada independentemente antes da integração com outras camadas. Nas fases de integração a interacção entre cada duas camadas foi testada isoladamente de forma a detectar o maior número possível de erros. Após a integração da interface gráfica, foram efectuados testes de funcionalidade e de usabilidade. Estes testes contaram com a colaboração de colegas de profissão e de outras pessoas sem formação informática. Ao longo deste processo diversos erros foram encontrados, assim como problemas de usabilidade. As situações encontradas foram sendo discutidas com os envolvidos e com os orientadores do projecto. As correcções e alterações efectuadas foram principalmente ao nível da implementação, embora alguma situações tenham obrigado à revisão da concepção. A elaboração do manual do utilizador foi igualmente um bom teste final de funcionalidade e de usabilidade, revelando pequenos bugs e dando origem as pequenas melhorias.

7. Conclusão

O objectivo deste projecto foi desenvolver uma ferramenta de gestão de requisitos que seja utilizável pelo gestor de projecto ao longo de todo o processo de desenvolvimento de software. Consideramos ter atingido os objectivos a que nos proposemos, respeitando todos os requisitos, quer de cliente, quer de software, definidos durante a fase de análise de requisitos. Apesar destes resultados, consideramos que antes de uma eventual entrada em produção deste sistema seria importante melhorar a interface de dados com outros sistemas de forma a facilitar a transição entre métodos de trabalho.

7.1. Futuros desenvolvimentos

Desde o início deste trabalho que foram identificadas inúmeras funcionalidades que fariam sentido incluir numa ferramenta de controlo de requisitos e que seriam extremamente úteis ao gestor de projecto. Não foi no entanto possível realizar todas as funcionalidades devido às limitações inerentes a um trabalho deste tipo. Assim sendo, ficou decidido incluir nesta primeira versão as funcionalidades essenciais à avaliação das potencialidades da ferramenta, apresentando já um conjunto interessante de características mas, realista de forma a garantir o cumprimento dos prazos do projecto. Segue-se uma lista de ideias que se traduzem em possíveis alterações a introduzir em futuras versões da ferramenta:

- Modo cliente/servidor (vários utilizadores simultâneos no mesmo projecto)
- Gestão de projectos (criar, alterar, remover, copiar, backups, etc)
- Gestão de utilizadores (adicionar, remover, níveis de permissões, etc)
- Melhorar propagação de alterações
- Definição de estados das entidades e respectivas regras de transição
- Critérios de filtragem e procura especializados por entidade
- Aassociação de definições ao texto de requisitos
- Help online
- Log de operações realizadas numa sessão
- Adicionar operação “undo” selectiva
- Introdução de barras de progresso para operações mais demoradas
- Exportação de requisitos para XML
- Importação de requisitos
- Produção de relatórios pré-definidos (sumários, estatísticas, etc)