INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Processo de Análise de Negócio

BRUNO MIGUEL FERNANDES SILVA
Julho de 2016

POLITÉCNICO
DO PORTO

isep Instituto Superior de Engenharia do Porto

Instituto Superior de Engenharia do Porto ⊙ Rua Dr. António Bernardino de Almeida, 431 ⊙ 4249-015 Porto
Tel. +351 228 340 500 ⊙ Fax +351 228 321 159 ⊙ mail@isep.ipp.pt ⊙ www.isep.ipp.pt

INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

isep

Business Analysis Process

BRUNO MIGUEL FERNANDES SILVA
Julho de 2016

INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

MESTRADO EM ENGENHARIA INFORMÁTICA

isep




Business Analysis Process

BRUNO MIGUEL FERNANDES SILVA
Julho de 2016

POLITÉCNICO
DO PORTO

Business Analysis Process

2015 / 2016

Bruno da Silva

1100901@isep.ipp.pt

Mestrado em
Engenharia Informática

Ramo de Sistemas de Informação e Conhecimento

Julho de 2016

Orientador ISEP: **Paulo Jorge Oliveira**

À minha família.

Agradecimentos

Quero agradecer a todos os professores que se cruzaram comigo no ISEP, com destaque para as professoras Fátima Rodrigues, Isabel Praça e Goreti Marreiros pelo apoio ao longo do Mestrado, à professora Susana Nicola pelo aconselhamento na análise de valor e em especial ao meu orientador Paulo Jorge Oliveira, pelo apoio, acompanhamento e aconselhamento durante o mestrado e na realização da dissertação.

Em último – mas não menos importante – lugar, quero agradecer à minha família e aos meus amigos, em especial à minha esposa, Tânia Azinheiro, pelo apoio e pela paciência, pois o tempo que lhes pude dedicar nestes últimos seis anos foi muito reduzido devido à faculdade, trabalho, estudo e projetos que estive envolvido durante os mesmos. Queria também dedicar ao futuro elemento da família, Salvador.

Sumário Executivo

A análise de negócio e processo de requisitos pertencem à fase de engenharia de *software* na realização de um *software*, onde são coletadas as informações necessárias junto dos *stakeholders* e geridas, para que se desenvolva exatamente o que está alinhado com a estratégia da empresa e/ou do interesse dos clientes.

Este trabalho tem como objetivo colmatar uma falha no processo de desenvolvimento de *software* existente na grande maioria das empresas, a gestão de requisitos e a análise de negócio.

O objetivo deste trabalho é evidenciar o processo de análise e gestão de requisitos, o processo, as boas práticas e técnicas de levantamento de negócio. São também identificados os maiores riscos, ferramentas que apoiam a especificação e gestão de requisitos. Desenvolve-se, também, um modelo para gerir os requisitos, a sua priorização, o seu risco, cobertura de teste necessária e avalia-se a implementação. Isto permite um aumento de performance e diminuição de risco desta fase do projeto de desenvolvimento de *software.*

Palavras-chave (Tema): **Análise de Negócio, Gestão de Requisitos, Gestão de Risco.**

Palavras-chave (Tecnologias): **Engenharia de Software, Segurança, Requisitos e Qualidade.**

Índice

Índice de Figuras

Índice de Tabelas

Índice de Gráficos

Notação e Glossário

Stakeholder	Parte interessada num projeto. Pode ser o cliente para o qual se está a realizar um projeto, acionistas, etc.
Sponsor	É o elemento chave de um projeto. Providencia os recursos financeiros e ajuda na gestão de projetos em assuntos como fundos, clarificação de âmbito (*scope*), monitorização de progresso e influência sobres outros intervenientes de forma a beneficiar o projeto. Sem ele um projeto não iniciaria.
Output	Um output é o resultado de um trabalho descrito. Um output deve ser minimamente completo para permitir trabalho sucessivo.
Quality Assurance	Departamento ou função responsável por garantir que o *software* tem o comportamento esperado.
Testers	Papel que define a pessoa responsável por testar a aplicação.
Mock-ups	Desenho de ecrã de forma a ser possível mostrar muito rapidamente uma ideia para se validar. Muitas vezes também chamado de *wireframe*.
Storyboard	Conjunto de *Mock-ups* relacionados e agregados.
Analista de Negócio	Também conhecido como *business analyst*. É a pessoa responsável por fazer a análise do negócio.
Backlog	Lista de desenvolvimentos, técnicos e funcionais, necessários para determinada aplicação.
IDE	IDE é a abreviatura de *Integrated Development Environment* que significa ambiente de desenvolvimento integrado, como o *Eclipse, Netbeans* ou *Visual Studio*.
Release	Versão de *software* lançada para o mercado.

1 Introdução

Neste capítulo é apresentado e enquadrado o projeto juntamente com o planeamento, as tecnologias utilizadas e a organização de todo o relatório.

1.1 Enquadramento

"47% of unsuccessful projects fail to meet goals due to poor requirements management"

Pulse of the Profession®, PMI, Agosto 2014

Como referido pelo PMI (*Project Management Institute)*, 47% dos projetos falham os objetivos devido à ineficiente análise e gestão de requisitos. A falha numa eficiente análise e gestão de requisitos causa a implementação de funcionalidades incorretas, desproporcionadas, irrealistas e testadas de forma errônea.

A análise de requisitos deve ser focada no negócio e não na implementação e deve ser realizada de forma a garantir a maior aproximação possível ao pretendido pelos *stakeholders* (clientes ou pessoas interessadas). É uma fase pertencente à área de engenharia de *software* à qual ainda se dá pouca relevância. No entanto, está a ganhar destaque, uma vez que é de extrema importância que seja bem realizada, já que é nesta fase que se realiza o levantamento das necessidades da aplicação, para depois ser modelada, desenvolvida e testada.

Por ser exatamente a primeira fase do desenvolvimento de *software*, qualquer falha aqui faz com que o modelo e a arquitetura sejam mal executados, a implementação seja diferente da pretendida e os testes não sejam focados no mais importante para os *stakeholders*.

A sua desconsideração geralmente traduz-se num resultado diferente do pretendido, sendo necessários custos extraordinários para poder adaptar a solução ao desejado (muitas vezes através de soluções provisórias e alternativas às boas práticas de desenvolvimento de software, devido a questões de tempo e da arquitetura mal estruturada).

Estudos demonstram que a fraca gestão de requisitos faz desperdiçar 5,1% (em empresas com elevado aproveitamento) a 10% (empresas de aproveitamento baixo) dos recursos financeiros, o que se traduz num desperdício de 5,1 a 10 milhões para cada 100 milhões gastos (Project Management Institute, 2014).

Por outro lado, a relação da gestão de requisitos com os testes é muito importante uma vez que é inviável e financeiramente impossível testar tudo numa aplicação e, assim, é possível identificar os cenários de teste mais importantes e as funcionalidades mais relevantes, para que estas possuam uma cobertura de teste mais elevada.

Ao longo do tempo e da evolução de um *software* o número de requisitos aumenta, tornando a sua gestão demasiado complexa para os procedimentos habituais, pelo que se pode assumir que existem então dois grandes pontos na gestão de requisitos: a gestão de um requisito durante o seu desenvolvimento e a gestão dos requisitos como um todo.

1.2 O Problema

Identificar o problema entre uma fraca gestão de requisitos, falha no objetivo do projeto e no desperdício de recursos é um passo importante para resolver o problema. Por estas razões o PMI elaborou um estudo para verificar a extensão do problema.

Neste estudo apenas metade das empresas envolvidas afirma ter os recursos adequados para efetuar uma boa gestão de requisitos. Aliás, o estudo demonstra até que existe um reconhecimento limitado daquilo que são as competências necessárias para efetuar uma boa gestão de requisitos. Apenas 24% das organizações consegue identificar e tenta desenvolver as competências necessárias, sendo que 34% das organizações admite que faz pouco ou até nada no que toca à identificação e desenvolvimento de competências para uma eficaz gestão de requisitos.

No que toca aos processos e boas práticas da gestão de requisitos, menos de metade das organizações diz que os faz convenientemente (Gráfico 1).

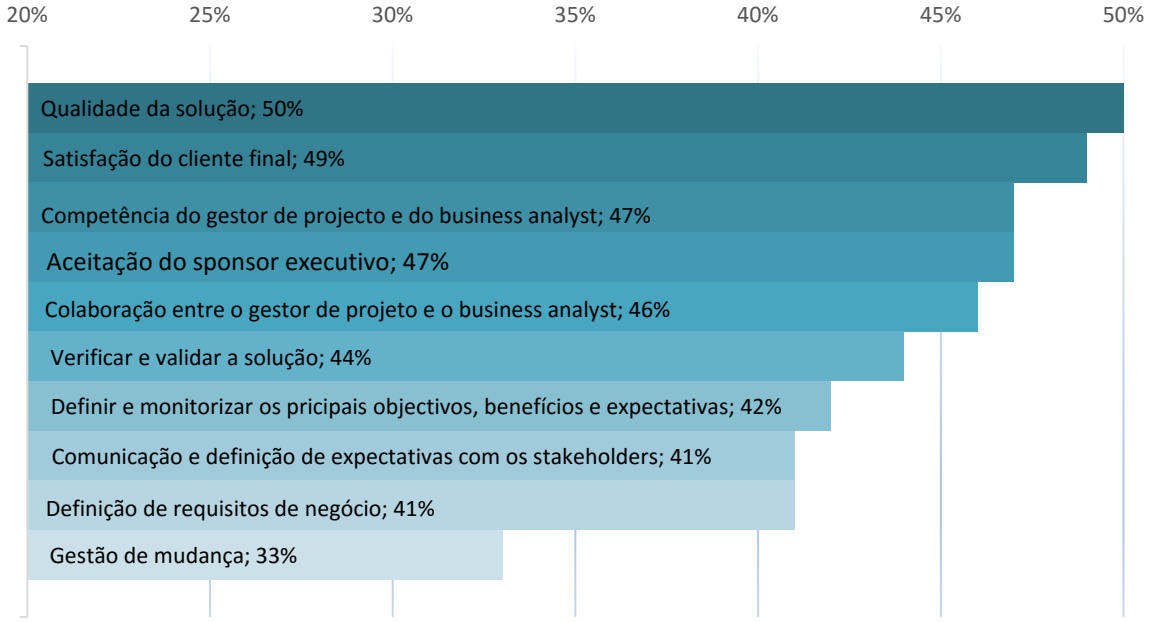


Gráfico 1 – Processos e Boas Práticas da Gestão de Requisitos Bem Realizados (Project Management Institute, 2014).

Estes dados revelam uma falta de capacidade no desenvolvimento de processos, boas-práticas e competências na gestão de requisitos. Para que qualquer uma destas deficiências seja abordada corretamente, as empresas devem avaliar a importância da gestão de requisitos e ligá-la ao sucesso dos projetos. No entanto, a maioria das organizações como um todo – quer a gestão de topo como os *stakeholders* – não dão o valor devido à gestão de requisitos como uma competência crítica para os projetos e iniciativas estratégicas (*Gráfico 2*).

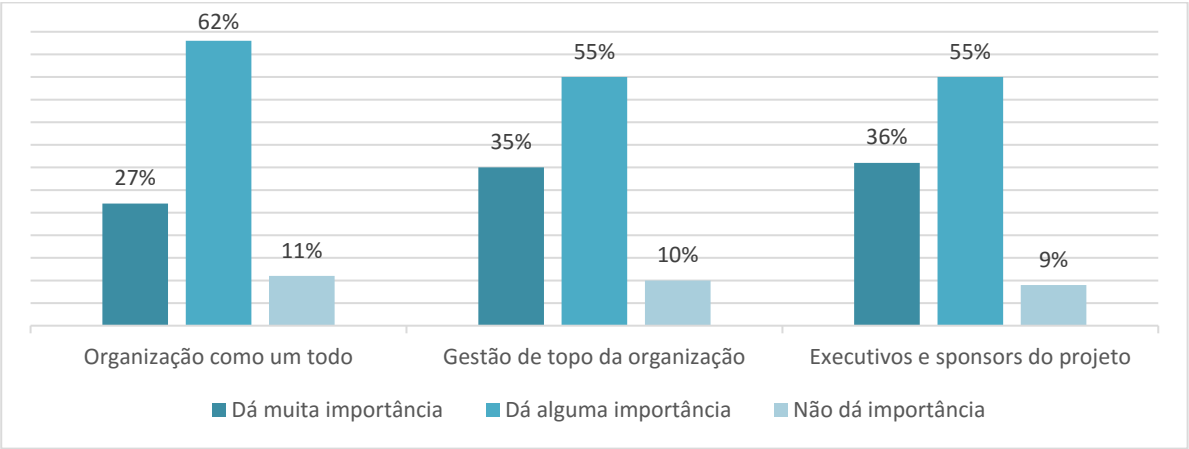


Gráfico 2 – Valor dado pelas organizações como uma competência crítica para os projetos e iniciativas estratégicas *(Project Management Institute, 2014)*.

Coletivamente, os recursos inadequados, o desenvolvimento limitado de competências, os processos e práticas informais e a falta de apoio nos postos mais elevados das organizações, evidenciam a fraca maturidade das empresas para a gestão de requisitos. Apenas uma em cada 5 empresas revela maturidade[1] na gestão de requisitos.

A grande maioria das empresas (87%) reconhece em alguma altura que são necessárias melhorias (35% revela que são necessárias muitas alterações e 52% algumas melhorias). Este estudo revela também que 58% das empresas encontra-se a definir melhorias nos seus processos e boas-práticas e que 53% está a rever os seus processos. Adicionalmente, 48% das empresas está a atribuir competências às pessoas da organização (Gráfico 3).

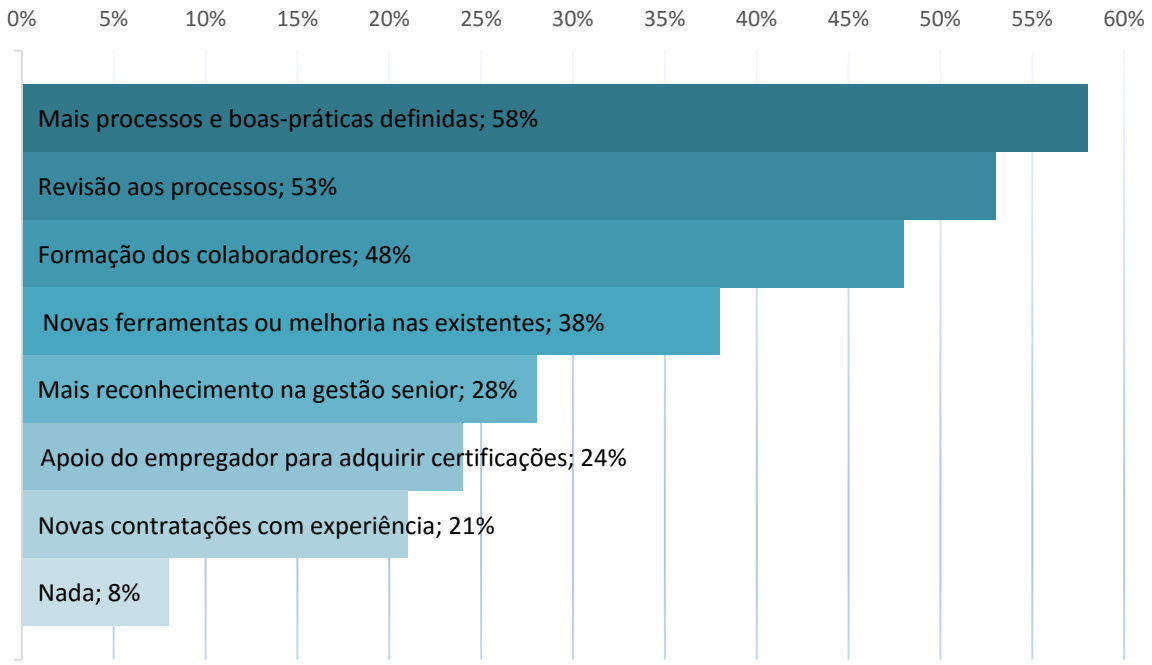


Gráfico 3 – Foco das organizações para melhorar a performance da gestão de requisitos (Project Management Institute, 2014).

Apesar de uma grande maioria das organizações se aperceber do problema de competências para a gestão de requisitos no sucesso dos projetos da organização, este estudo evidencia uma lacuna entre esse entendimento e a sua prática. 71% das organizações referencia que a comunicação aos *stakeholders* é extremamente ou muito importante, no entanto, apenas 59% das empresas diz que o faz sempre ou regularmente. Lacunas similares podem ser encontradas em todo o processo de requisitos, como se pode observar no Gráfico 4 acerca das atividades consideradas críticas pelas empresas e da sua realização por parte das empresas. Devido ao reconhecimento da natureza destas competências, as organizações devem focar-se em realizá-las nos seus projetos (Project Management Institute, 2014).

[1] Maturidade é relacionada com os elevados níveis de capacidade e eficiência que uma empresa demonstra às suas pessoas, processos e ferramentas quando realiza gestão de requisitos num projeto. É conseguida através da contínua monitorização, identificação de áreas de melhoria, quer no processo, quer nas competências e nas atividades baseadas nos requisitos. É assegurada pelo reconhecimento organizacional e das lideranças dos projetos (Bieg, 2015).

Este estudo revela também que não interessa quem faz a gestão de requisitos (se o gestor do projeto ou o analista de negócio), mas sim como esta gestão é feita. É evidente que as empresas com alto nível de performance[2] reconhecem a criticidade da realização da gestão de requisitos, ao contrário das outras[3] (Project Management Institute, 2014).

Gráfico 4 – Atividades críticas e a frequência com que é realizada (Project Management Institute, 2014).

[2] Empresas de baixa performance são as empresas que atingem 60% ou menos dos projetos dentro do tempo estipulado, do orçamento e dos objetivos.

[3] Empresas de alta performance são empresas que atingem 80% ou mais dos projetos dentro do tempo estipulado, do orçamento e dos objetivos.

Ter os recursos necessários (pessoas, tempo, etc.) e reconhecer e desenvolver as aptidões necessárias para uma boa gestão de requisitos leva a um projeto com melhor performance, como é possível verificar no Gráfico 5 e Gráfico 6.

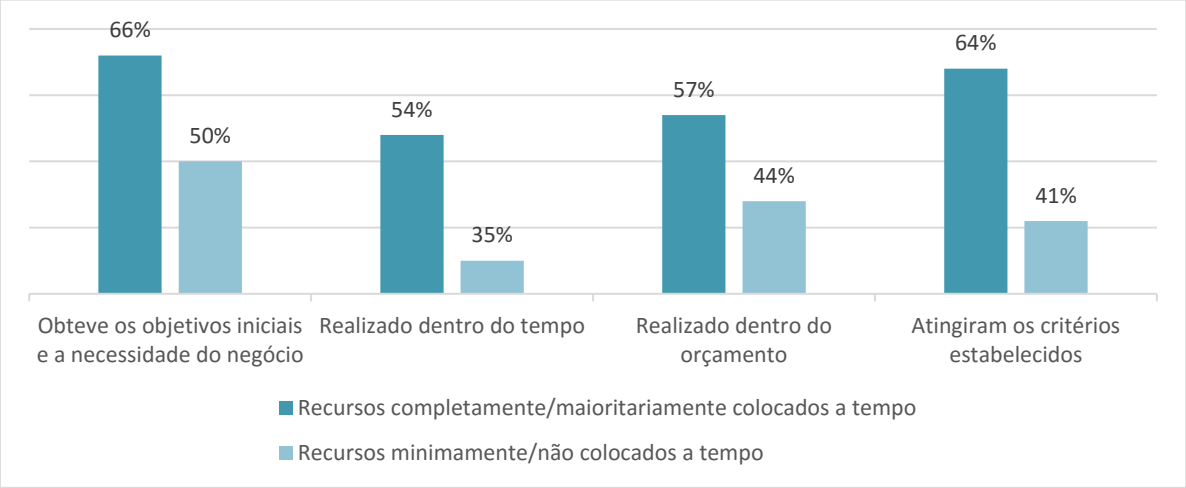


Gráfico 5 – Diferença entre colocar os recursos (tempo, pessoas, etc.) a tempo no projeto e não o fazer (Project Management Institute, 2014).

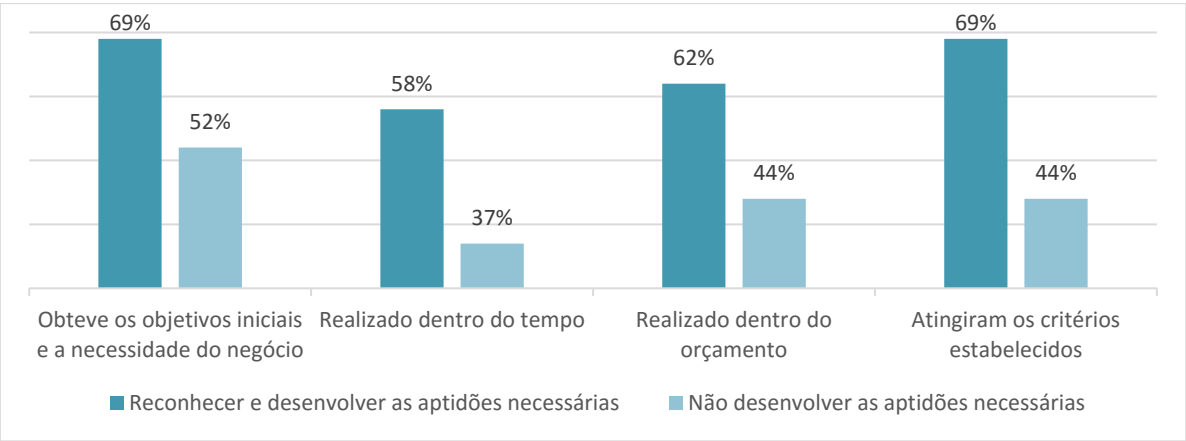


Gráfico 6 – Diferença entre reconhecer e desenvolver as aptidões necessárias aos colaboradores para a gestão de requisitos e não o fazer (Project Management Institute, 2014).

Em termos de processo as organizações com processos de gestão de requisitos e boas-práticas conseguem resultado melhores, particularmente na validação e verificação dos requisitos. Um processo formal resulta melhor uma vez que assegura validação de objetivos implicando uma melhor performance no projeto (Gráfico 7).

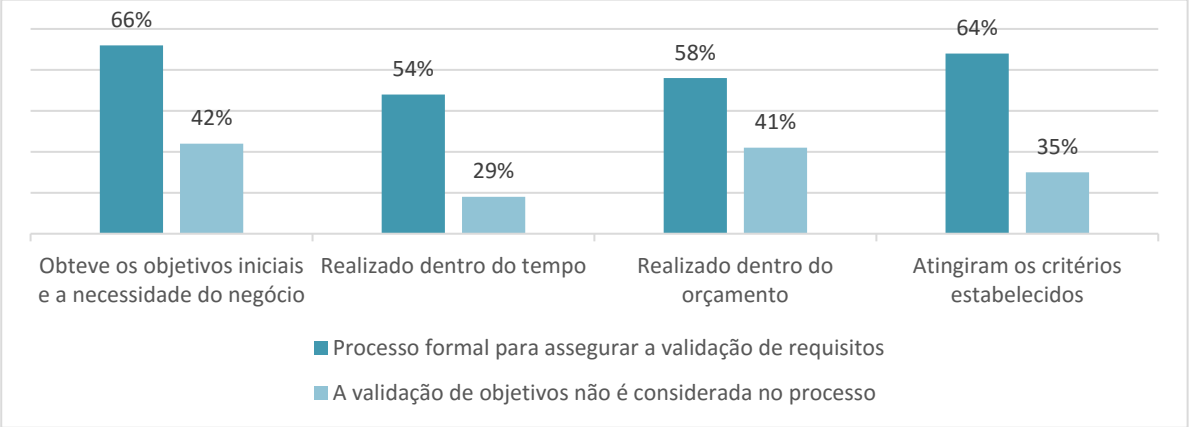


Gráfico 7 – Processo formal de validação de requisitos contra a desconsideração da validação no processo (Project Management Institute, 2014).

Organizações de alta-performance quando comparadas com as de baixa performance nos 10 princípios base das práticas de gestão de requisitos são claramente superiores nos processos de gestão de requisitos (Gráfico 8).

Gráfico 8 – Práticas de gestão de requisitos em organizações de elevada e baixa performance (Project Management Institute, 2014).

No que toca a fatores culturais numa organização, o estudo demonstra que estes são fatores críticos, sendo necessário que a gestão de requisitos seja reconhecida, quer pela gestão de topo e executiva,

quer pelos *sponsors* dos projetos e o resto dos colaboradores, para obter resultados significativamente melhores.

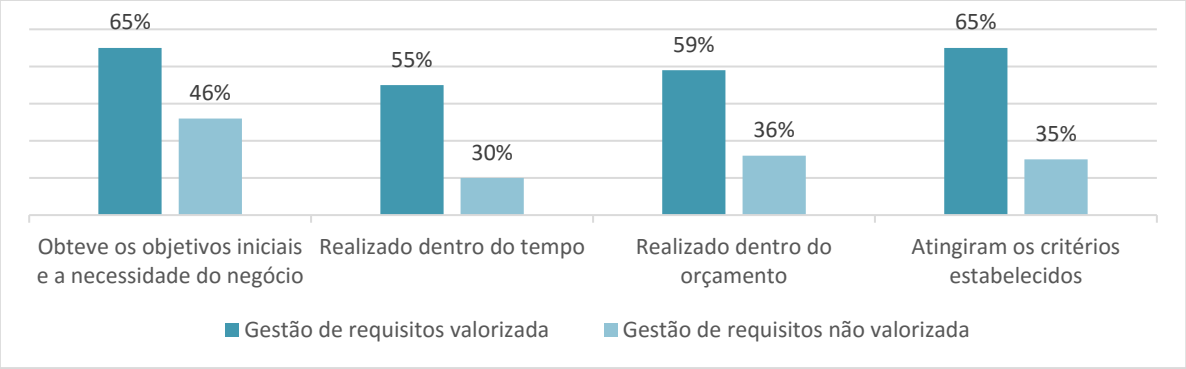


Gráfico 9 – Performance quando a gestão de requisitos é valorizada e desvalorizada pela empresa (Project Management Institute, 2014).

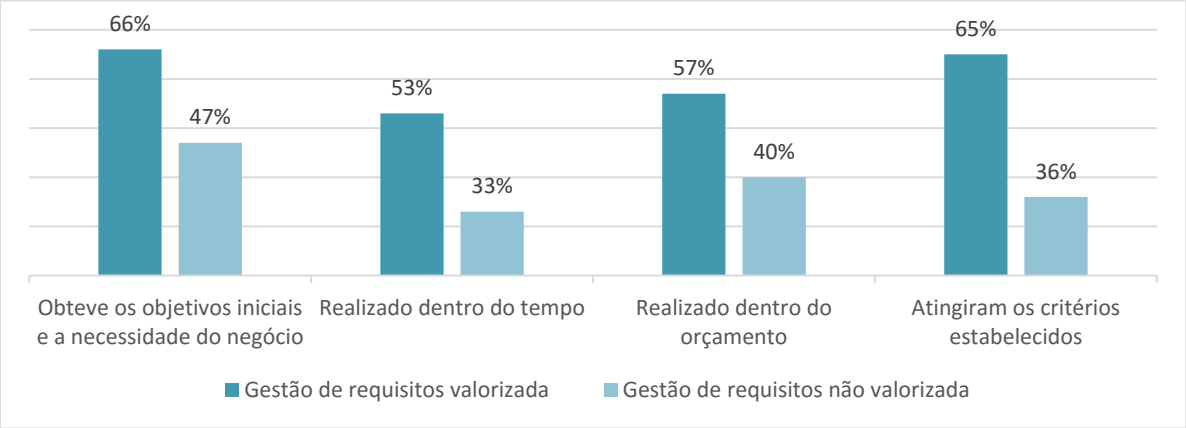


Gráfico 10 – Performance quando a gestão de requisitos é valorizada e desvalorizada pela gestão de topo (Project Management Institute, 2014).

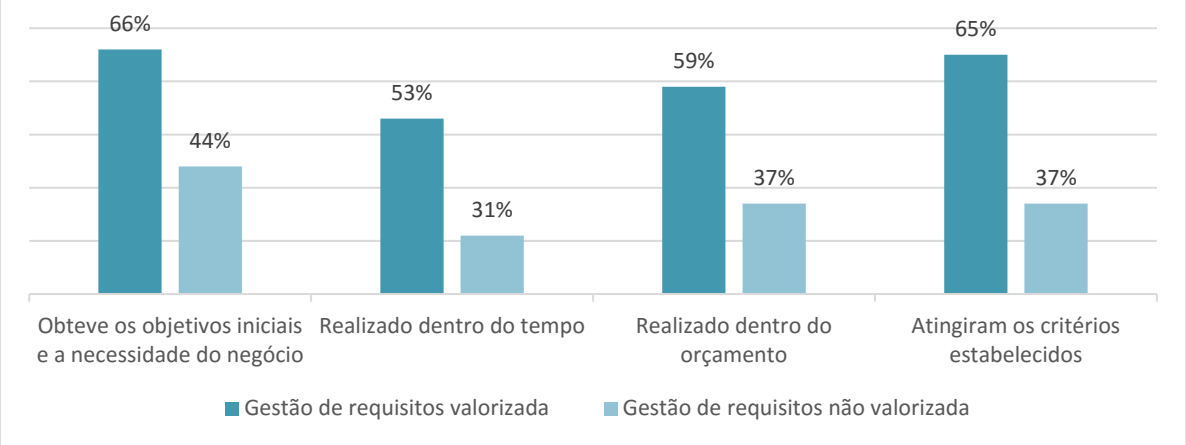


Gráfico 11 – Performance quando a gestão de requisitos é valorizada e desvalorizada pelos sponsors executivos (Project Management Institute, 2014).

Quando comparadas as empresas de alta performance com as de baixa performance no que toca à gestão de requisitos, não existe surpresa na diferença entre estas uma vez que a gestão de requisitos é claramente mais valorizada nas empresas de alta performance, como podemos verificar no Gráfico 12, sendo no entanto ainda abaixo do desejado.

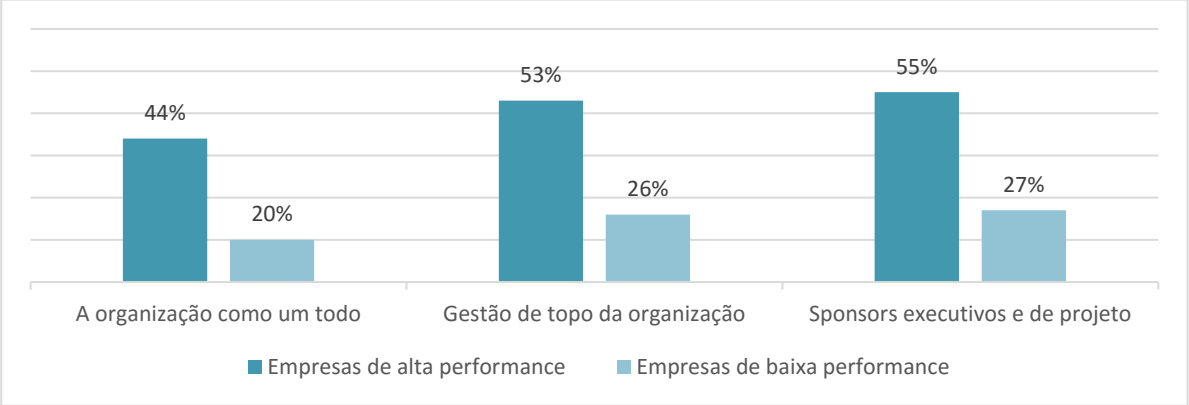


Gráfico 12 – Valorização da gestão de requisitos nas empresas (Project Management Institute, 2014).

A alocação dos recursos a tempo para realizar tarefas de gestão de requisitos, o reconhecimento do desenvolvimento das competências das pessoas, a estandardização, formalização e maturidade dos processos de gestão de requisitos, a aplicação constante das boas práticas de requisitos, a garantia de que a empresa, como um todo, valoriza esta gestão como uma competência crítica para os projetos, sendo colocado o compromisso da gestão de topo por trás da gestão de requisitos, conseguiram mitigar o impacto negativo que a fraca gestão de requisitos tem nos projetos assim como o desperdício financeiro que origina (Project Management Institute, 2014).

Para além do problema da gestão de requisitos, existem outros problemas no processo de análise de requisitos, tal como priorizar os requisitos a desenvolver, qual a cobertura de testes e o que testar. Atualmente estas tarefas são deixadas ao critério do bom senso do analista de negócio, existindo apenas um conjunto de boas práticas para ajudar na tomada de decisão.

1.3 Análise de Valor

Esta dissertação reflete um processo de análise e gestão de requisitos, pretendendo dar resposta de forma genérica à temática e ao controlo de implementação, de forma a possibilitar a redução de custos do desenvolvimento de *software*. O processo está direcionado a gestores de requisitos e analistas de negócio ou empresas que pretendam aumentar a sua eficiência nesta área. Existem muitos artigos e livros por vezes com ideias concorrentes que referenciam as boas práticas e alguns processos, no entanto, não existe nenhuma conceptualização do processo ou forma de interligar todas as fases de desenvolvimento, nem sequer um modelo físico. Este modelo e processo incorpora priorização de requisitos através da negociação, PBS (*product breakdown structure)*, questionário de perguntas frequentes, gestão de seguranças e validação da implementação.

Uma proposta de valor bem definida permite identificar claramente e inequivocamente aquilo que vamos oferecer ao cliente e deve, em termos ideais, ser associado a um benefício. No caso do processo apresentado, a proposta de valor é criar maior qualidade e eficiência no desenvolvimento de *software,* uma vez que permite poupar tempo e recursos assim como avaliar a solução final.

O modelo corretamente usado pelas pessoas com as competências adequadas irá permitir uma redução de custos e maior assertividade na implementação.

Este modelo e processo poderia adotar muitas formas, sendo, no entanto, necessário respeitar dependências entre os vários pontos assim como alguns pontos-chave do processo. No entanto, e conforme o tamanho de uma empresa ou os papéis que a pessoa assume numa empresa, poderá fazer sentido achatar o modelo. Por exemplo, se o gestor de requisitos assumir também o papel de

stakeholder, este não terá qualquer interesse em realizar a negociação dos requisitos a priorizar uma vez que ele é o único decisor.

O modelo de *canvas* (Figura 1) não se aplica na íntegra nesta dissertação, uma vez que se trata de um modelo e processo, no entanto, podem-se definir alguns pontos:

<table>
<tr>
<td rowspan="2">Parceiros-chave

Open Web Application Security Project (OWASP)

Business Analysis Body of Knowledge (BABOK) e as diferentes aplicações de análise de negócio.</td>
<td>Atividades-chave

Criação de um novo processo para engenharia de requisitos, verificação da sua implementação através do mesmo, aumentando a qualidade final de software e reduzindo erros, eliminando riscos e baixando o custo final de software.</td>
<td rowspan="2">Proposta de Valor

Quantificar a criação de valor através do resultado em aplicações mais seguras, maior controlo na qualidade, melhorias na gestão e menos tempo pelos vários intervenientes.

Apoio ao desenvolvimento de software de forma mais efetiva e com mais qualidade</td>
<td>Relação com o Cliente

Não existe qualquer relação com cliente uma vez que se trata de um modelo/processo que pode ser implementado pelas empresas de tecnologia de informação apenas com identificação do autor de origem.</td>
<td rowspan="2">Segmentação de Cliente

Empresas de tecnologias de informação que pretendem melhorar os seus processos.</td>
</tr>
<tr>
<td>Recursos-chave

Analistas de negócio, gestores de requisitos e todas as entidades intervenientes no processo.

Capital Intelectual.</td>
<td>Canais de Distribuição

Qualquer um, internet, boca-a-boca, etc. Qualquer canal que permita difundir o modelo.</td>
</tr>
<tr>
<td colspan="3">Custo de Estrutura

Não existe. Uma vez terminado é um documento a ser comunicado.</td>
<td colspan="2">Fontes de Receita

Não existe, uma vez que a informação é gratuita e pública.</td>
</tr>
</table>

Figura 1 – Análise canvas.

Os parceiros para a realização deste modelo e processo são a *Open Web Application Security Project (OWASP), Business Analysis Body of Knowledge (BABOK)* e as diferentes aplicações de análise de negócio.

A atividade chave deste modelo/processo é a criação de um novo processo para engenharia de requisitos, verificação da sua implementação através do mesmo, aumentando a qualidade final de *software* e reduzindo erros, eliminando riscos e baixando o custo final.

Os recursos-chave para a implementação deste modelo são os analistas de negócio, gestores de requisitos e todas as entidades intervenientes no processo.

Os canais de distribuição deste modelo podem ser qualquer um, internet, boca-a-boa, etc. Qualquer canal que permita difundir o modelo. A estrutura de custos e fluxos de receita não existem, uma vez que se trata de informação transmitida gratuitamente.

Neste processo não existe qualquer relação com o cliente, uma vez que se trata de um modelo/processo que pode ser implementado pelas empresas de tecnologia de informação, sem

qualquer relação com o autor. O segmento de clientes são as empresas de tecnologias de informação que pretendem melhorar os seus processos.

Após a implementação deste modelo no processo de desenvolvimento de *software* é possível quantificar a criação de valor através do resultado em aplicações mais seguras, maior controlo na qualidade, melhorias na gestão e menos tempo pelos vários intervenientes. Poderemos verificar isso de forma automatizada através de algumas tecnologias que serão apresentadas neste documento, uma vez que este processo possui uma fase de avaliação.

De forma mais objetiva podemos usar o modelo AHP (*Analytic hierarchy process*) através da comparação do uso ou não uso deste processo/modelo de forma. É possível comparar os resultados de erros de segurança no código, a medição de satisfação dos *stakeholders* no final do desenvolvimento, a aceitação dos desenvolvimentos a realizar pelos *stakeholders* e a clareza da informação transmitida. Estes pontos são utilizados para se avaliar a solução.

Como a maioria das empresas não possui gestão de requisitos por não dar a devida relevância, isto faz com que a informação se perca rapidamente ou se desenvolva código de forma pouco assertiva, o que irá sempre traduzir-se em prejuízos significativos.

1.4 Objetivos

O grande objetivo deste trabalho é melhorar a qualidade de desenvolvimento de *software* e reduzir o custo da sua produção e manutenção. Através de um novo processo de requisitos, processo este que balanceia a engenharia de requisitos e a análise de negócio, incluindo a gestão e especificação de requisitos, apresentação de formas de documentação, ferramentas, gestão de prioridades e garantias de qualidade na entrega. Pretende-se dotar as empresas e profissionais com conhecimento da área de uma nova visão sobre a mesma.

De forma a definir bem qual o propósito da solução pretendida, serão respondidas algumas perguntas da definição do problema, inspirado e adaptado de "*The Thinker's Guide to Engineering Reasoning*", (Foundation for Critical Thinking, 2007), lecionado no módulo de Análise de problemas, pesquisa e escrita técnico-científica do Mestrado em Engenharia Informática.

Qual o propósito deste processo?

O processo a definir pretende dar resposta de forma genérica à gestão de requisitos e controlo de implementação de forma a possibilitar a redução de custos do desenvolvimento de *software*. Pretende-se também apresentar algumas ferramentas que agilizem o processo ou ajudem a melhor definir requisitos.

Assim, o objetivo é desenhar um processo e conceptualizar um modelo de forma de interligar todas as fases da análise de negócio e gestão de requisitos ao desenvolvimento, criando um modelo físico para aplicar no desenvolvimento de *software*.

Quem é o destinatário?

O modelo e processo apresentado têm como destinatário todos os gestores de requisitos, analistas de negócio e empresas com interesse em aumentar a sua eficiência.

O modelo irá satisfazer as exigências dos clientes?

Corretamente usado pelas pessoas com as competências adequadas irá permitir uma redução de custos e maior assertividade na implementação.

Pode o modelo realizado ser adaptado?

Este processo sugere algumas ferramentas que podem, ou não, ser utilizadas mediante a eventual existência de outras ferramentas na empresa para o mesmo fim. No entanto, o modelo apresentado poderá ser alterado mediante a dimensão da empresa e o foco do projeto, devendo o processo ser respeitado assim como os pontos-chave do modelo.

O quanto é importante o *time-to-market*?

É relevante o *time-to-market* deste processo uma vez que algumas empresas já começam a implementar *software* segundo o seu entendimento de análise de requisitos. Quanto mais cedo este processo for apresentado, mais atenção terá pelas pessoas e pelas empresas que pretendem investir nesta área.

Um ponto de vista é tipicamente presumido pelas pessoas pelo que é necessário avaliar outros pontos de vista. Que outros pontos de vista merecem consideração?

Esta proposta tenta tirar as mais-valias dos vários processos de engenharia, concretizando-as num processo e modelo prático, pelo que é assente em diversos pontos de vista de diversos autores.

Que condições de ambiente ou operacionais são assumidos?

É assumido que para a utilização deste modelo a empresa deve ser de média ou grande dimensão e que existe alguém responsável pela sua manutenção.

Que pressupostos foram assumidos sobre a disponibilidade de informação?

Existem já diversos artigos e publicações acerca da área, no entanto, não são consensuais e é uma área em evolução, como aconteceu com os testes numa fase anterior.

Qual é a fonte de informação?

A fonte de informação usada neste processo e modelo é baseado em diversas *frameworks* como o BABOK e OWASP, livros e artigos sobre requisitos, processo de engenharia de requisitos, gestão de requisitos, gestão de risco e negociação. Estas fontes podem ser consultadas no Questionário.

Que informação nos falta?

Existe pouca literatura sobre a implementação e execução de uma visão macro do processo de desenvolvimento, à qual o processo e modelo a desenhar pretende responder.

Como podemos obtê-lo? Que provas de conceito devem ser realizadas?

Deve ser realizada uma prova de conceito com a execução do modelo num caso concreto.

Já consideramos todas as fontes relevantes?

As fontes consideradas são as fontes com maior peso na matéria e no assunto.

Existe um caminho para a futura evolução? (design e atualização)

A adaptabilidade a diferentes tipos de empresa do modelo é algo que irá necessitar de atualização.

Há um final de vida a considerar?

Tudo evolui, pelo que é necessário existir uma evolução, mas não existe um final de vida para o modelo.

Quais são as implicações mais importantes da falha do modelo?

As implicações mais importantes na falha do modelo traduzem-se na não redução da despesa, mas na manutenção da mesma.

Quais as caraterísticas insensíveis à mudança?

As caraterísticas insensíveis à mudança neste modelo são as necessidades existentes das diferentes fases.

Que benefícios podem existir em derivações do modelo?

A derivação do modelo potencia a sua adaptabilidade à realidade de mais empresas. Por exemplo para empresas mais pequenas, com diferentes posições das pessoas na empresa, divisão de tarefas, utilização de outras ferramentas de gestão, etc.

1.5 Abordagem Metodológica

A abordagem preferencial para a resolução deste problema passa pelo estudo da área de análise de negócio e gestão de requisitos, técnicas de levantamento de negócio e priorização, análise dos maiores riscos na implementação (segurança) e ferramentas para a especificação.

Existe um conjunto de livros de referência já identificados e referidos nesta dissertação que falam das boas práticas, algumas ferramentas possíveis de utilizar que surgiram muito recentemente, *frameworks* de segurança, entre outros. Após este estudo a informação é sintetizada e desenvolvido um processo e modelo que dá resposta aos diferentes pontos, identificando as diferentes fases necessárias para a gestão, os pontos necessários e os resultados para cada uma das fases.

1.6 Planeamento

Este trabalho foi planeado e realizado em três fases principais:

- Investigação – leitura de livros e artigos sobre a área, investigação de ferramentas e experimentação;
- Desenvolvimento – realização do modelo e processo;
- Avaliação – avaliação da solução proposta.

Imediatamente a seguir ao planeamento iniciou-se a **investigação**, que durante a sua evolução conduziu muitas vezes a pesquisa mais específica, à aprendizagem de novos conceitos e a aprendizagem de conhecimentos mais generalistas de gestão de projeto. Foram lidos diversos livros da área e artigos relacionados, estando estes referenciados na secção Referências Bibliográficas.

Relativamente ao **desenvolvimento** do projeto, o modelo/processo encontra-se desenhado e explicado quer pelas diferentes fases, diferentes resultados tangíveis necessários e a forma de gerir os requisitos. Este desenvolvimento foi amadurecendo desde o primeiro esboço através de testes da sua implementação e frequente análise dos resultados obtidos, até chegar ao modelo apresentado. Durante o desenvolvimento foram identificados novos pontos de investigação, que foram atendido ou não conforme o âmbito da dissertação.

No que diz respeito à **avaliação,** o documento descreve a forma como a nova metodologia e processo foram avaliados, os resultados obtidos das avaliações, sendo estes normalizados e processados de forma a verificar a diferença antes e depois do processo, assim como o potencial para melhoria de processo e qualidade em empresas de *software*.

1.7 Contributos deste Trabalho

Este trabalho contribui para:

- Dotar qualquer empresa de conhecimento da área de análise de negócio e gestão de requisitos;
- Fornecer um modelo e processo de análise de negócio e gestão de requisitos, que possibilita à empresa:
    - Minimizar os tempos de desenvolvimento;
    - Atingir assertividade no que é pedido.

São também mostradas algumas ferramentas da área e boas práticas para a documentação eficiente de requisitos, como a resposta a questionários para avaliar se o requisito está corretamente realizado.

São identificados pontos de extensão ou a possibilidade de adequar o processo a qualquer empresa, uma vez que o modelo poderá ser desmembrado para obter mais ou menos passos, assim como adequação a outras ferramentas. O que é importante são os *outputs* gerados, sendo as ferramentas apresentadas uma forma de auxílio.

1.8 Organização da Dissertação

Esta dissertação foi desenvolvida de forma iterativa durante o decorrer do projeto, sendo a maioria da mesma composta pela pesquisa desenvolvida, por ser a maior componente do projeto.

Na Introdução é realizada uma breve descrição do projeto, a razão pelo qual é importante, os objetivos e abordagem tomada e o seu planeamento e contributos para a comunidade.

No capítulo Estado da Arte encontra-se a explicação da análise de negócio, as formas de engenharia de requisitos e de obtenção dos mesmos. A tipificação dos requisitos em diversos blocos, as áreas de conhecimento necessárias para a gestão de requisitos, ferramentas possíveis de ser utilizadas para uma maior clareza na sua elaboração.

Como este trabalho trata também da qualidade do código e da avaliação de segurança do mesmo, será abordada a segurança e a possibilidade de avaliação qualitativa do código.

No capítulo de Design da Solução será apresentado o modelo e processo desenvolvido, explicando ao pormenor os diferentes pontos e interligação com ferramentas.

Na avaliação da solução encontra-se identificada a forma de avaliar o presente processo e modelo. Os resultados obtidos são analisados de forma a verificar se o modelo realizado produz realmente benefícios.

No capítulo das Conclusões é exposto um pequeno resumo dos objetivos atingidos, limitações, trabalho futuro e apreciação final do trabalho.

2 Estado da Arte

Neste capítulo serão apresentados os tópicos da análise e especificação de requisitos, métodos usados e processos existentes. Para maior detalhe acerca dos mesmos poderá ser consultada a bibliografia referenciada nesta dissertação.

2.1 Definições e Conceitos da Análise de Negócio

Para percebermos este tema é importante falar de algumas definições de termos utilizados.

Na literatura existem muitas definições sobre o que é a análise de negócio, uma boa definição encontrada:

"Análise de negócio é o conjunto de tarefas e técnicas utilizadas para funcionar como elo de ligação entre os *stakeholders*, de forma a compreender a estrutura, políticas e operações de uma organização recomendando soluções que permitem à organização atingir os seus objetivos

Define a capacidade que uma organização necessita para fornecer produtos e serviços para os *stakeholders* externos. Inclui a definição de metas organizacionais, a sua conexão com objetivos específicos, determinação do plano para atingir as metas e objetivos e definição de como as entidades externas interagem" (International Institute of Business Analysis, 2009).

Já para a função de Analista de Negócio, pode considerar-se a seguinte definição:

"Analistas de negócio devem analisar e sintetizar a informação fornecida pelos diferentes *stakeholders* (clientes, *staff*, executivos, etc.). É responsável por extrair as suas necessidades e não os seus desejos sendo um facilitador de comunicação entre diferentes pontos da organização tomando um papel fulcral no alinhamento das necessidades de negócio com o desenvolvimento informático, servindo de «tradutor» entre estes dois grupos" (International Institute of Business Analysis, 2009).

O analista de negócio pode ter diferentes cargos numa organização, pode ser um *requirement engineer*, um *business architect,* um *product owner*, etc. Como esta atividade é realizada por pessoas com diferentes competências, é difícil existir um processo uniformizado para a especificação de requisitos para os vários papéis, uma vez que cada um faz da maneira que lhe aprece mais indicada.

Para percebermos melhor as definições mais comuns do significado de requisito[4]:

- Condição ou capacidade necessária de um *stakeholder* para resolver um problema ou atingir um objetivo.
- Condição ou capacidade que necessita de ser cumprida ou pertencer a uma solução ou componente de uma solução de forma a satisfazer um acordo, *standard*, especificação ou outros documentos impostos formalmente.

Existem competências que uma pessoa necessita para desenvolver corretamente uma análise de negócio tais como (International Institute of Business Analysis, 2009):

- **Pensamento analítico e solução de problemas** que suporta a identificação de problemas de negócio, proposta de soluções para a sua resolução e perceção das necessidades dos *stakeholders*.

[4] Baseado no IEEE 610.12-1990: IEEE Standard Glossary of Software Engineering Terminology.

- **Caraterísticas Comportamentais** que suportam o desenvolvimento das relações com os *stakeholders* incluindo qualidades como a ética, confiança e organização pessoal.
- **Conhecimento de Negócio** que suporta a compreensão do ambiente onde a análise é efetuada, assim como o conhecimento geral das soluções existentes.
- **Competências de Comunicação** que suportam a especificação e comunicação dentro dos *stakeholders*. São necessários para ouvir e perceber como o negócio é compreendido, a mensagem e o formato da mesma.
- **Competências de Interação** suportam o trabalho com um elevado número de *stakeholders* envolvendo as habilitações de trabalhar como parte de uma equipa e ajudar a equipa a tomar decisões.
- **Aplicações de Software** são utilizadas para facilitar o desenvolvimento colaborativo, registando e distribuindo os requisitos para os *stakeholders*. As forças e fraquezas das várias ferramentas devem ser conhecidas.

2.2 Engenharia de Requisitos

Uma boa definição de requisitos:

"Requisitos envolvem as necessidades e desejos do cliente: Quais as capacidades do sistema? Um requisito é o que deve acontecer, não como deve acontecer. Desta forma, os requisitos apresentam o âmbito da conceção e da implementação em vista" (Cannegieter & Arensen, 2008).

A distinção entre tipos de requisitos:

"Um requisito funcional é a descrição de um comportamento que um sistema irá apresentar sobre condições específicas. Um requisito não funcional é a descrição de uma propriedade ou característica que um sistema deve apresentar ou um constrangimento que deve ser respeitado" (Wiegers & Beatty, 2013).

A engenharia de requisitos é, então, "a descoberta, documentação e gestão dos requisitos" (Wiegers & Beatty, 2013). O "*Standish Group*", uma organização de consultoria de investigação focada na gestão de projeto, tem reivindicado que uma clara declaração de requisitos é o terceiro fator que mais contribui para o sucesso de um projeto IT (Johnson, 2008).

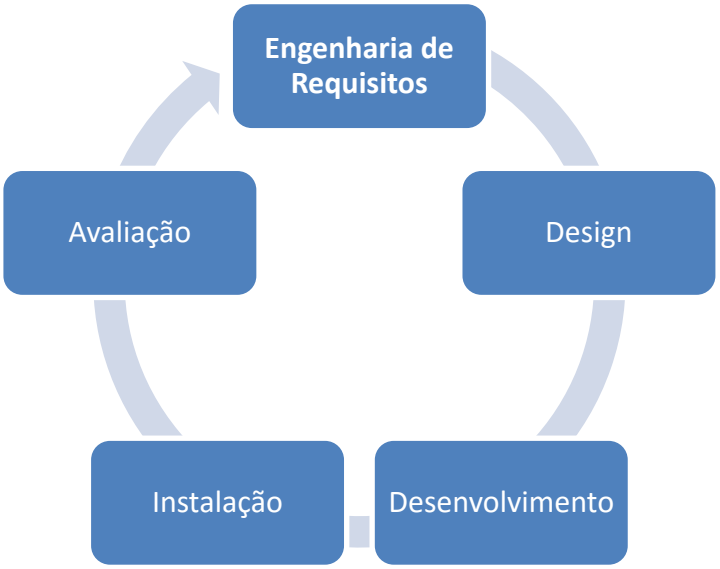


Gráfico 13 – Ciclo de vida do desenvolvimento de software.

A engenharia de requisitos segundo o ciclo de vida do software (Langer, 2008) apresentado mostra uma estrutura comum para o desenvolvimento iterativo de *software*. Esta representação varia conforme o autor, mas a engenharia de requisitos teve sempre um papel promissor no processo (Nuseibeh & Easterbrook, 2000) (Nurmuliani, et al., 2004). Os requisitos identificados no processo de engenharia de requisitos alimentam o processo de *design* onde são desenhadas as soluções para as satisfazer. Após completar o ciclo, a avaliação alimenta diretamente a engenharia de requisitos sobre a forma de lacunas, erros, alterações, entre outros.

Dentro da Engenharia de requisitos temos o uso de várias áreas de conhecimento (Langer, 2008) (Nurmuliani, et al., 2004), como:

- Identificação de Requisitos;
- Análise de Requisitos;
- Especificação de Requisitos;
- Priorização de Requisitos;
- Verificação de Requisitos após o desenvolvimento;
- Gestão de requisitos para controlar os pontos anteriores.

Outras fontes podem incluir outros pontos, como validação, análise de impacto, negociação, análise custo/benefício, *quality assurance*, gestão de alteração, entre outros.

2.2.1 Processos de Engenharia de Requisitos

O processo de engenharia de requisitos *Method Engineering* foi definido como a disciplina de desenhar, contruir e adaptar métodos, técnicas e ferramentas para o desenvolvimento de sistemas de informação (Brinkkemper, 1996). Neste trabalho foi também identificado que nem todos os projetos são iguais, pelo que em alguns casos seria pertinente criar variâncias ao que *Brinkkemper* denominou de *Situational Method Engineering*. Esta notação foi suportada por *Rolland & Prakash* (Rolland & Prakash, 1996) e mais tarde tornou-se ela própria um campo de investigação.

O processo *Assembly Based Method Engineering* foi proposto como o processo anterior em tempo real (Ralyté, et al., 2003). Este método foi posto em prática e foram propostos novos passos para o processo (Van de Weerd, et al., 2006):

- Analisar situações específicas da implementação e identificar necessidades;
- Selecionar métodos candidatos que atendem a um ou mais necessidades identificadas;
- Analisar os métodos e registar blocos (fragmentos) de desenvolvimento;
- Selecionar ferramentas e juntar blocos úteis em novos métodos de forma a obter casos.

O *Method Association Approach*, que é uma adaptação do anterior (Luinenburg, et al., 2008) e revisto pela Universidade de Sorbonne e a Universidade de Utrecht (Deneckere, et al., 2015), adicionou dois novos passos, dividindo o último passo em “agrupar e associar funcionalidades com os métodos candidatos” e “seleção do design e assemblagem das funcionalidades” e “validação do método de design”.

O processo direcionado a um produto tem também as suas influências na engenharia de requisitos, uma vez que em vez de reunir todas as necessidades de um cliente para cada projeto, deve também cobrir requisitos existentes em novos clientes. Os requisitos podem então ser identificados a partir do mercado, incluindo tendências, normas, boas-práticas e legislação do mercado. Com esta abordagem, a engenharia de requisitos torna-se mais do que orientada aos clientes, sendo orientada para o

mercado global. Este processo é conhecido como MDRE (*Market Driven Requirement Engineering*) (Regnell & Brinkkemper, 2005).

O processo MDRE, embora similar ao processo de Engenharia de Requisitos, pode ser diferente em vários aspetos. O foco na identificação, por exemplo, muda um ou alguns clientes para um mercado inteiro. Como determinado em estudos de caso acerca do MDRE: "[para software de produto] *o processo de engenharia de requisitos deveria ser capaz de identificar requisitos baseados nas necessidades do utilizador final e selecionar um conjunto de requisitos que resultam no software de produto que pode competir no mercado"* (Regnell, et al., 1998).

Outro exemplo de uma variação no processo MDRE é o aumento da dependência na priorização e da análise. Com tantas exigências é necessário filtrar na priorização e análise. *Carlshamre* referiu: "No desenvolvimento de *software* orientado ao Mercado, o planeamento da *release* é uma das tarefas mais críticas. Selecionar um conjunto de requisitos para realização em determinada *release* é tão complexo como importante para o sucesso de um *software* de produto" (Carlshamre, 2002).

2.3 Áreas de Conhecimento da Gestão de Requisitos

A Gestão de requisitos contempla as tarefas de estabelecer uma base de requisitos e manter a sua rastreabilidade, controle de mudanças e gestão de configuração (Project Management Institute, 2016).

As áreas de conhecimento permitem compreender e delinear as tarefas que são necessárias realizar. Estas tarefas podem ser realizadas de forma sucessiva, interativa ou em simultâneo. As áreas de conhecimento não tentam representar fases de um projeto, no entanto, poderá ser útil organizá-las desta forma:

Planeamento e monitorização da análise de negócio é a área de conhecimento que cobre como o analista de negócio define as áreas de atividade necessárias para determinar o esforço necessário a uma determinada tarefa. Esta tarefa deve incluir a identificação dos *stakeholders*, a seleção das técnicas de análise de negócio, o processo usado para gerir os requisitos e como monitorizar o progresso do trabalho (International Institute of Business Analysis, 2009). Em conjunto com o desenvolvimento de um plano de gestão de projeto, trata-se de uma parte crítica das atividades globais de planeamento. O planeamento das atividades de requisitos assegura a melhor aproximação aos requisitos (Project Management Institute, 2016).

Identificação descreve como a análise de negócio procede com os *stakeholders* de forma a identificar e perceber as suas necessidades e preocupações ou um problema no negócio atual, percebendo o ambiente onde eles trabalham. Este processo visa perceber as necessidades que para eles são explícitas e não transmitidas, tipicamente realizado antes do projeto. No entanto, durante o curso de um projeto, é normal existir mudança, normalmente devido a fatores externos, altura em que devem ser revistas para garantir que as decisões feitas anteriormente permanecem válidas (International Institute of Business Analysis, 2009) (Project Management Institute, 2016).

Gestão de Requisitos e Comunicação descreve como são geridos os conflitos, problemas e alterações de forma a assegurar que os *stakeholders* e a equipa de projeto continuam de acordo no âmbito da solução, como os requisitos são transmitidos e como o conhecimento obtido é mantido para uso futuro (International Institute of Business Analysis, 2009).

Análise descreve como os analistas de negócio identificam uma necessidade, refinam e clarificam a sua definição para que possa ser desenvolvida. Esta área descreve a definição do problema, o *business*

case, estudo de fiabilidade e definição do âmbito da solução (International Institute of Business Analysis, 2009).

Priorização define como as necessidades de negócio devem ser priorizadas e progressivamente elaboradas. Envolve análise de necessidades para definir soluções, identificar melhorias, verificar e validar os resultados dos requisitos (International Institute of Business Analysis, 2009).

Especificação é a realização do *output* usado para a implementação do requisito para que a equipa de desenvolvimento implemente uma solução que responda às necessidades dos *stakeholders* e do *sponsor*.

Avaliação da solução descreve como os analistas de negócio avaliam as propostas de soluções, de forma a determinar a melhor solução às necessidades, identificar lacunas e deficiências e determinar soluções alternativas ou mudanças necessárias para a solução. Descreve também como as soluções implementadas poderão ser avaliadas para verificar se os objetivos originais foram cumpridos.

Fecho do projeto ou da fase assegura que o resultado é transitado de um estado de desenvolvimento para um estado de manutenção. A avaliação das atividades da solução é realizada à medida do necessário de forma a assegurar que a solução continua a responder às necessidades do negócio e continua a entregar o valor esperado (Project Management Institute, 2016).

Monitorização e controlo dos requisitos de forma a garantir que o âmbito é continuamente gerido através de alterações ao projeto e que as alterações aos requisitos seja apenas colocada quando aprovadas (Project Management Institute, 2016).

Cada área de conhecimento deverá ter uma tarefa associada que resultam num *output* que trás valor à organização sendo esse resultado algo visível, específico, útil e mensurável. Uma tarefa considera-se completa quando uma pessoa ou grupo pode utilizar o *output* produzido.

Antes de se começar por identificar os requisitos, o projeto deve já estar aprovado e os *stakeholders* definidos.

2.3.1 Divisão dos Diferentes *Stakeholders*

Um *stakeholder* é uma pessoa interessada em determinado requisito. Por isso, antes de se começar a identificar os requisitos é preciso identificar quais os papéis dos diferentes *stakeholders* e qual a responsabilidade que eles possuem num projeto. Uma boa prática é utilizar uma matriz RACI (Project Management Institute, 2015):

- R – Responsável – Pessoa interessada e que necessita do requisito;
- A – *Accountable* – Pessoa(s) que aprovam a realização do requisito, incluindo o caso de negócio quando garantido. Pessoa responsável sobre determinado requisito;
- C – Consultar – Pessoa ou grupo a ser consultado para obter informações para perceber o problema atual ou oportunidade;
- I – Informar – Pessoa ou grupo que será informado sobre os resultados da necessidade.

Deve definir-se os diferentes *stakeholders* necessários e respetivas seguintes ações, de forma a consultar as pessoas conforme a ação necessária (Tabela 1). Esta matriz é útil para o analista de negócio e gestor de projeto que tem interesse na identificação dos *stakeholders*.

Tabela 1 – Exemplo de uma matriz RACI para um projeto.

	Sponsor	Gestor de Produto	Analista de Negócio	Equipa de Desenvolvimento	Gestor de Projeto
Identificar problema ou necessidade	A	C	R	C	
Analisar o estado atual da funcionalidade	A	I	R	C	
Recomendar uma ação	I	A	R	C	C
Prepara o caso de negócio	I	A	R	C	I

2.4 Problemas Comuns na Obtenção de Requisitos

Um grande problema na obtenção de requisitos são as lacunas não mencionadas. Conhecer estas lacunas o mais cedo possível permite planear e desenvolver com mais qualidade, evitando a necessidade de implementar alterações em funcionalidades incorretamente implementadas. Falhar estas lacunas ou ignorá-las podem trazer efeitos devastadores para um projeto. Investigar estas lacunas apenas após o processo de engenharia de requisitos estar concluído pode exceder o custo em 10 a 200 vezes (Boehm, 1991):

Existem alguns problemas comuns na obtenção dos requisitos:

- Conhecimento Tácito (“Esqueci-me de mencionar que o sistema aceita pedidos automaticamente. Estou tão habituado que esqueci-me de o referir…”);
- Falsos Pressupostos (“O sistema não gera relatórios automaticamente?”);
- Conhecimento pelo qual ninguém se responsabiliza (“Ninguém sabe porque o sistema funciona assim. A pessoa que tomou essas decisões já cá não trabalha mas deveria ter documentado…”).

O problema de encontrar requisitos escondidos por detrás do conhecimento tácito do cliente não é novo, pelo que *Nuseibeh* e *Easterbrook* propuseram que os analistas aprendessem da psicologia cognitiva, porque: “Por exemplo, os especialistas do domínio do problema, muitas vezes têm grandes quantidades de conhecimento tácito que não é passível de introspeção, pelo que as suas respostas a perguntas realizadas por analistas podem não coincidir com o seu comportamento” (Nuseibeh & EasterBrook, 2000). *Ratchev* reconheceu o mesmo problema: “A conversão desse conhecimento em informações explícitas depende muito da utilização do conhecimento tácito, ou seja, o conhecimento pessoal que está implícito e não pode ser facilmente comunicado. A principal desvantagem deste tipo de conhecimento é que ele evita a discussão crítica” (Ratchev, et al., 2003).

Estes problemas podem ser evitados através de diferentes fontes, como documentação do sistema atual, dos processos ou um conjunto de regras de negócios, outros *stakeholders* e através de outros meios como sessões de trabalho, entrevistas, questionários ou sessões demonstrativas.

2.5 Tipificação de Requisitos

O termo requisito é um termo que gera muita discussão, uma vez que as caraterísticas necessárias para algo ser considerado um requisito possui fronteiras muito frágeis. Uma boa forma de gerir

requisitos começa pela sua catalogação, existindo assim (International Institute of Business Analysis, 2009):

- **Requisitos de negócio** – objetivos, necessidades e metas da organização, as razões para o início de um projeto e respetivas métricas para o seu sucesso;
- **Requisitos do *stakeholder*** – Necessidades de um *stakeholder,* onde se descreve como se irá interagir com a solução. Servem de ponte entre os requisitos de negócio e os requisitos da solução;
- **Requisitos da solução** – Caraterísticas da solução de forma a atingir os objetivos de negócio e normalmente partidos em subcategorias, particularmente quando descrevem uma solução de *software*:
    - Funcionais – Descrevem o comportamento da solução. Descrevem a forma de realizar determinado comportamento ou operação;
    - Não funcionais – Capturam as condições que não se relacionam diretamente com o comportamento ou a funcionalidade, mas descrevem condições ou qualidades que o ambiente ou a solução necessitam. Podem ser considerados requisitos de performance, capacidade, segurança, disponibilidade, apresentação da interface de utilizador e arquitetura da solução.
- **Requisitos de Transição** – Descrevem capacidades que uma solução necessita para facilitar a transição do estado atual para o futuro que serão descartados após esta transição. Ao contrário dos outros requisitos, estes são temporários.

Os requisitos da solução podem ainda ser catalogados (dentro dos funcionais e não funcionais) de outra forma relacionada com as dimensões. Sobre uma necessidade existem 7 dimensões principais possíveis de extrair dando estas, origem a diversos requisitos:

- Funcionais:
    - Utilizador;
    - Ação;
    - Dados;
    - Regras de Negócio;
- Não Funcionais:
    - Interface;
    - Ambiente;
    - Qualidade (por exemplo performance);

Esta tipificação define o RBS (Requirement Breakdown Structure), um exemplo é apresentado na Figura 2.

Esta divisão é uma divisão possível, sendo por vezes necessário simplificar conforme a dimensão e tipo de projeto. Por exemplo, em algo completamente novo, normalmente não existem requisitos de transição, assim como em alguns projetos os requisitos funcionais podem ser apenas de utilizador ou de sistema.

A utilização de *user stories* nos requisitos funcionais com apoio de outras formas de documentação podem minimizar a sua tipificação a requisitos funcionais, uma vez que é possível extrair as diferentes dimensões a partir dos mesmos.

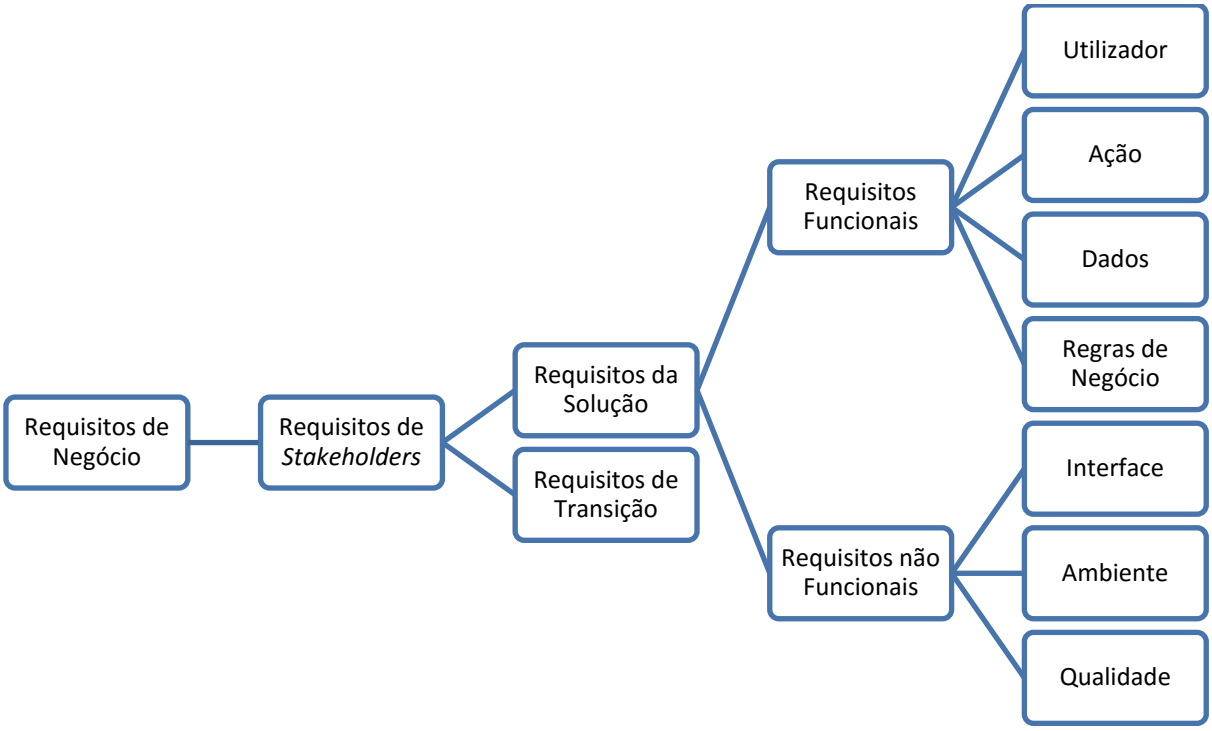


Figura 2 – As 7 dimensões do produto de um requisito (Gottesdiener, 2015).

2.6 Processos de Obtenção de Requisitos

Os processos de obtenção de requisitos são normalmente composto por quatro fases principais (www.phoenix.tc-ieee.org, 2016):

- Identificação de requisitos (identificar *stakeholders*, objetivos, necessidades, expetativas e limites do sistema);
- Análise de Requisitos (Classificação, modelação conceptual, design de arquitetura e alocação de requisitos e negociação de requisitos);
- Especificação de requisitos;
- Avaliação de requisitos (revisão, protótipos, validação do modelo e testes de aceitação);

2.7 Ferramentas de Requisitos

No mercado começam agora a surgir ferramentas de requisitos, embora a grande maioria tem como base a especificação em documentos (*Word*, *Excel*, etc.) onde a aplicação trata de gerir e organizar os mesmos através da sua tipificação e incluindo algumas estatísticas úteis para a gestão. O *Tracecloud* (tracecloud, 2015) ou *SpiraTest* (Inflectra Corporation, 2015) são ferramentas deste tipo, sendo esta última uma ferramenta para gestão de testes adaptada para requisitos.

Outras ferramentas autointitulam-se como ferramentas de gestão de requisitos, mas na realidade são ferramentas de gestão de projeto que se podem adaptar à realidade da gestão dos requisitos, como por exemplo o *Jira* (Atlassian Pty Ltd, 2015), *Requirement One* – também com possibilidade de especificação em documentos (RequirementONE Inc., 2015), *Producteev* (Jive Software, 2015), etc.

Existem outras ferramentas que muitas vezes se identificam como ferramentas de requisitos, no entanto, estão limitadas à realização de diagramas, o que é algo necessário, mas não definidor da especificação de requisitos. Temos como exemplo o *Gliffy* (Gliffy, Inc., 2015), *Yworks* (yWorks GmbH, 2015)e o *Draw.io* (Draw.io, 2015).

Existem também as que fazem alguns diagramas e possuem uma parte de gestão documental, como o *Visure Requirements* (Visure Solutions, S.L., 2016).

Não existe nenhuma solução completa, no entanto, a gestão de requisitos pode ser realizada por qualquer ferramenta de gestão de projetos, desde que tenha suporte para *backlog* no caso de utilização de metodologias ágeis.

Por outro lado, ferramentas completas de especificação de requisitos são mais difíceis de encontrar, mas estão finalmente a surgir soluções. Algumas destacam-se por serem mais completas, como o *Modelio Requirements Analyst* (Modelio, 2015), o *Yakindu Requirements* (Itemis AG, 2015)*, o Visual Paradigm* (Visual Paradigm, 2016) e o *Blueprint Requirements Center* (Blueprint Software Systems Inc., 2016), sendo estas duas últimas soluções muito recentes. Estas ferramentas poderão ser completadas ou já incluir outras ferramentas necessárias, como ferramentas de prototipagem e de gestão de tarefas.

Uma caraterística muito importante para conciliar a gestão com a especificação de requisitos é a comunicação entre ferramentas.

2.7.1 Pontos-chave na Avaliação de Ferramentas de Requisitos

Os pontos-chave para a avaliação de uma ferramenta de requisitos centram-se nas seguintes questões:

- Especificação para aprovação dos *stakeholders*;
- Especificação para entendimento da equipa de desenvolvimento;
- Redução de ambiguidades na comunicação entre as pessoas envolvidas no projeto;
- Comunicação entre sistemas;
- Velocidade de realização dos modelos e de manutenção;
- Controlo de versões;
- Exportação para manuais;
- Análise de impacto e dependências entre requisitos;
- Gestão de requisitos;
- Integração com gestão de projetos para atribuição de tarefas;
- Licenciamento.

A especificação para aprovação dos *stakeholders* e para entendimento da equipa de desenvolvimento traduz-se na realização de diversos diagramas, *mock-ups*, textos e imagens que ajudam à perceção do que vai ser realizado.

Para a redução de ambiguidades na comunicação entre as pessoas envolvidas no projeto, existe o conceito de dicionário que clarifica o significado de determinada palavra (que muitas vezes é facilmente percetível para o negócio, mas não para as pessoas). Existem aqui ferramentas que permitem a relação entre termos.

Para a comunicação entre sistemas, existe uma norma de comunicação de requisitos entre máquinas que permite fazer com que ele possa ser portável entre máquinas. Este formato é o *reqif* (*Requirements Interchange Format*) que é uma estrutura em XML que, associada aos seus dados modelo (*metadata*), permite a comunicação de forma estandardizada. O suporte a este formato torna-se muito importante na escolha de uma ferramenta, uma vez que possibilita a sua integração com outras.

Outra qualidade imprescindível de uma ferramenta de requisitos é a sua rapidez, isto é, a velocidade que demora a especificar e a manter os requisitos existentes, relações e diagramas. Estas alterações tornam também o controlo de versões imprescindível para a ferramenta, especialmente se pudermos atribuir versões ao requisito.

Outra característica a ter em conta é a possibilidade de extração de documentação sob a forma de manuais (de forma automatizada) para ser possível enviar para as pessoas intervenientes do projeto, as análises de impacto que ajudam na análise do gestor de requisitos a prever impactos ou alterações a funcionalidades existentes.

A possibilidade de ter gestão de requisitos na ferramenta, a integração com outras ferramentas como gestão e projeto, modelação e mesmo programação pode constituir uma solução integrada, permitindo tudo disponibilizado numa só ferramenta.

Outro ponto importante é o licenciamento da ferramenta, uma vez que o custo é sempre algo que pesa numa empresa e que depende sempre da dimensão da mesma.

2.7.2 Modelio Requirements

A ferramenta Modelio Requirements (Modelio Requirements, 2015) foi demonstrada num *webinar* e facultada para teste e foram identificados os seguintes pontos-chave:

- Ligações tipificadas entre requisitos de forma a permitir extrair dependências entre elas (derivação, parte de…, refinamento, etc.);
- Ligações entre os diferentes diagramas de modelação e requisitos de forma a obter ligação com as diferentes fases de modelação (rastreabilidade);
- Dicionário de termos e conceitos de negócio para reduzir ambiguidade de termos utilizados entre os diferentes intervenientes e associação dos termos através de regras e tipificações como: antónimo, contexto, sinónimo, relacionado, etc. Visualização das relações possíveis através de diagrama ou tabela, sendo a tabela passível de ser exportada para ficheiro Excel;
- Controlo de versões dos requisitos e organização de requisitos em pastas, sendo possível organizar por área e/ou tipo (no entanto de forma manual);
- Possibilidade de organizar os requisitos hierarquicamente;
- Geração automática de diagramas de dependência entre casos de uso e entre artefactos (possibilidade de ver o que é usado e o que usa determinado caso de uso num só diagrama);
- Possibilidade de anexar documentos ou imagens como por exemplo um *mock-up* ou documentação originada pelos *stakeholders* (*pdf, word*, etc.);
- Customização dos campos usados para definir os requisitos;
- Integração com ferramentas de documentação baseados na *web* (Por exemplo o *Confluence* (Atlassian Pty Ltd, 2015));
- Possibilidade de criar modelos para os manuais (*templates* de documentação) a exportar onde se indica a informação a introduzir e o tipo (*web, Word, pdf*, etc.);
- Exportação para manuais de utilizador (*pdf, Excel* e *Word*) e utilização dos mesmos bidireccionalmente (ao alterar no manual, altera automaticamente no *modelio*);
- Matriz de dependência entre requisitos;
- Geração automática de análise de impacto sobre alterações;
- A parte de modelação de modelo de domínio é gratuita e usa formatos estandardizados como *UML, BPMN, SysML, TOGAF* e *XMI*;

- Usa formatos estandardizados de requisitos como *CMMI, TOGAF, SysML, DoDAF, MODAF, NAF, Agile* com *product backlog;*
- Integrado nas diferentes fases de desenvolvimento (requisitos, design, análise, gestão, desenvolvimento, etc.);

Mais detalhes e imagens sobre a ferramenta podem ser consultados no anexo 1 – Modelio Requirements.

2.7.3 Yakindu Requirements

Yakindu (Itemis, 2015) é uma ferramenta mais orientada para a especificação de requisitos e não para modelação como a *Modelio Requirements* (Modelio Requirements, 2015), sendo os seus pontos fortes:

- Simples de usar;
- Distribuída como *plugin* do eclipse (Eclipse, 2015), ferramenta muito usada para programação ou em IDE próprio;
- Possibilidade de criar vários projetos, disponibilizando uma vista agregadora para os requisitos e especificação de requisitos, sendo estes também separados;
- Possível relacionar requisitos e hierarquizar os mesmos;
- Sincronização automática com ferramentas de gestão de projeto através do plugin *Mylin* (MyLin, 2015), criando automaticamente numa metodologia de gestão de projetos *agile*, *storys* (desenvolvimento curtos) e *epics* (desenvolvimentos longos)*;*
- Tipificação de requisitos sendo possível criar novos tipos por projeto;
- Definição das caraterísticas do requisito como o estado ou a data da última alteração;
- Utilização da norma *reqif* para comunicação entre sistemas;
- A edição dos diagramas é feita através de texto, o que se torna extremamente produtivo realizando também as ligações para outros diagramas, ecrãs e requisitos;
- Permite controlo de versões através de um repositório *git* (Software Freedom Conservancy, 2016) (apenas criam versões do texto realizado para fazer os diagramas);
- Gera automaticamente os diagramas de *package*, de casos de uso, de interação entre casos de uso e de fluxo do caso de uso através do texto escrito;
- Existem também outros diagramas complementares, como o diagrama de entidades, de ciclo de vida e de interação de ecrã onde é possível adicionar *mock-ups*;
- Permite também realizar alguma documentação através de uma linguagem própria;
- Geração de relatórios e documentação em formato *pdf* e *html* e de estatísticas de esforço necessário para desenvolver a funcionalidade especificada.

Mais detalhes e imagens sobre a ferramenta podem ser consultados no anexo 2 – Yakindu Requirements.

2.7.4 Visual Paradigm

O *Uexceler* do *Visual Paradigm* (Visual Paradigm, 2016) é uma ferramenta muito recente e possui uma aproximação ao processo de gestão de requisitos. Encontra-se já bastante completo tentando responder a diversas necessidades com as seguintes funcionalidades:

- Usa a *user story* com a premissa "***As a*** *<user role>* ***I want to*** *<action/behaviour>* ***so that*** *<benefit>*" para identificação dos requisitos;

- Gera automaticamente o diagrama de casos de uso com base na premissa e permite ainda adicionar novas ações para o ator em questão. Caso se modifique o gerado é atualizada a premissa;
- Permite definir os requisitos, preenchendo alguma informação de forma automatizada com a possibilidade de priorizar, tipificar, definir o autor, os pressupostos, as pré-condições e pós-condições (estas também podem ser uma ligação direta com um caso de uso ou um diagrama), a complexidade e o seu estado em desenvolvimento;
- Conforme as relações existentes entre artefactos cria automaticamente as pré e pós-condições;
- Gera automaticamente o diagrama de fluxo permitindo alterar o mesmo, ligar os diagramas existentes ou adicionar novos diagramas como diagramas de estado, casos de uso, atividade, sequência;
- Permite realizar análise de impacto tendo em conta o que existe, permitindo visualizar o que uma alteração pode afetar;
- É possível adicionar referências e artefactos usados durante o levantamento, como *websites*, diagramas, imagens, documentos, etc.;
- Permite descrever o plano de testes e indicar as configurações necessárias para os realizar;
- Permite criar histórias de interação utilizador-sistema para os diferentes requisitos. Possibilitando também a sua organização tendo filtros de pesquisa, assim como o estado no desenvolvimento;
- Permite realizar *mock-ups,* possíveis de relacionar com os requisitos;
- Permite realizar *storyboards*, reutilizando os diversos requisitos;
- Possui um dicionário de termos para ajudar a reduzir ambiguidade entre intervenientes;
- Permite realizar modelação UML (Diagramas ER, classes, sequência, etc.) e *reverse-engineering*[5] de código diretamente na ferramenta;
- É possível visualizar diagramas por tipo e exportar os diagramas e informações pretendidos, quer para ambiente *web,* quer para *pdf*.

Mais detalhes e imagens sobre a ferramenta podem ser consultados no anexo 3 – Visual Paradigm.

2.7.5 Blueprint Requirements Center

Blueprint (Blueprint Software Systems Inc., 2016) é uma solução baseada na *cloud* que permite realizar os requisitos de forma colaborativa com suporte para o ciclo de vida dos requisitos. Esta ferramenta possui como pontos de destaque:

- Passível de se trabalhar em qualquer local através de uma ligação à *internet* (*cloud*):
- Gestão de todos os documentos para a especificação de requisitos (fontes de informação, diagramas, ecrãs, *mock-ups*, casos de uso e documentos gerados);
- Possui um Dicionário onde é possível definir um termo e sua definição, no entanto, não é possível criar relações entre termos e conceitos;
- Para cada requisito é possível catalogar o mesmo pelo seu tipo, definir a *release, stakeholder*, prioridade, risco, iteração de desenvolvimento, esforço necessário, estado entre outros;
- Permite criar diversos diagramas como casos de uso, diagrama de entidades, diagramas de fluxo e de estado;

[5] Extrair informação de *design* tendo por base o código-fonte.

- Permite realizar análises de impacto automatizadas e visualizar no imediato os diagramas e documentos associados sem navegar para outro local;
- Permite criar o diagrama e a *user story* com facilidade de forma a explicar a interação com o utilizador, contruindo o diagrama/texto de forma automatizada;
- Permite criar *mock-ups* possíveis de se relacionar com os requisitos;
- Possui uma área para gestão de capturas de ecrã;
- Permite realizar *storyboards* com fluxos de comportamento, podendo incluir *mock-ups* e capturas de ecrã;
- Possui um sistema de aprovação de requisitos, o que permite gerir facilmente o que está pronto a implementar;
- Possui diferentes visões sobre os requisitos, de forma hierárquica, individual, por tipo, etc.;
- Permite visualizar de forma estatística o progresso do projeto em termos de requisitos;
- Permite especificar os testes com a premissa **Dado** <pré-condição> **quando** <condição> **então** <resultado>;
- Possui uma ferramenta de comunicação interna onde os intervenientes podem deixar os comentários referente a cada requisito;
- Como se trata de uma ferramenta na *internet* é muito simples partilhar, permitindo também exportar para os formatos de ficheiro *pdf*, *word* e *excel*;
- Permite bloquear a edição do requisito onde só poderão ser realizados comentários sobre o mesmo;
- É possível "seguir" um requisito, isto é, sempre que é comentado é enviado um *e-mail*;
- Possui uma área de gestão de requisitos onde tem os seus detalhes, tarefas associadas, dependências, defeitos, discussões sobre o mesmo, revisões, gráficos, casos de teste, execução dos testes e alterações realizadas. Permite a reutilização de artefactos ao longo dos requisitos;
- Possui controlo de versões e comparador entre elas.

Esta solução embora, bastante completa, não possui integração com ferramentas de projeto o que torna muito difícil existir a ligação da especificação com o desenvolvimento. O seu preço é também bastante elevado, sendo apenas possível saber o valor exato através de negociação, no entanto, é superior a 1000 euros para uma empresa pequena com um conjunto limitado de utilizadores.

Mais informação e imagens podem ser encontradas em 4 – Blueprint Requirements Center.

2.7.6 Avaliação das Ferramentas de Requisitos

De forma a avaliar as diferentes aplicações foi realizado um quadro comparativo com os requisitos mais importantes:

Tabela 2 – Tabela comparativa de software de requisitos.

	Modelio	Yakindu	Visual Paradigm	BluePrint
Suporte *reqif* *(formato para intercomunicação entre aplicações)*	Não	Sim	Não	Apenas realiza a importação para a aplicação
Foco da aplicação	Modelação com suporte e integração de requisitos	Especificação de requisitos	Ferramenta de Gestão e especificação de Requisitos	Ferramenta de Gestão e especificação de Requisitos

	Modelio	Yakindu	Visual Paradigm	BluePrint
Velocidade de especificação	Médio	Alta	Médio	Aplicação lenta para a especificação
Tempo de Manutenção	Médio	Baixo	Médio	Baixo
Integração com ferramentas de desenvolvimento e UML	Ferramenta Própria integrada apenas com ferramentas da empresa (Modelio)	Através de qualquer plugin do Eclipse para UML e uso do Eclipse.	Ferramenta própria de modelação, sem integração.	Possui o que é necessário para a especificação de requisitos e tem integração parcial com testes e desenvolvimento.
Versionamento	Manual (voltar a um momento preciso não é possível)	Integrado com git (não cria versões dos diagramas)	Apenas suportado no *vpository* (proprietário) – licença extra	Sim
Análises de impacto	Sim	Não	Sim	Sim
Estimativas de desenvolvimento	Não	Sim	Não	Não, mas é possível definir
Dependências e relações entre requisitos	Sim	Sim	Sim	Sim
Dicionário	Sim	Não	Sim	Sim, mas sem relações entre conceitos
Exportação para manuais	Sim	Sim	Sim	Sim
Mock-ups	Não	Possível de integrar no eclipse (ex: wireframe sketchup)	Sim	Sim
Integração com ferramentas de gestão de projeto	Não	Possível de integrar no eclipse (ex: *mylin*) apenas para realização da tarefa	Não	Apenas através de ferramenta própria
Outras funcionalidades pertinentes		Possibilidade de integrar a geração de testes automáticos através de por exemplo o *SWTBot*		Gera testes automáticos; Sistema de aprovação. Discussão.
Licença[6]	**Licença Anual:** 800€ por computador	**Licença anual:** 1 utilizador – 79€ Versões empresariais sobre consulta	**Licença Vitalícia:** 1 utilizador – 709€ Multiutilizador – 886$ Online – $80000 **Avença mensal** 1 utilizador – 31$/Mês Integrado com o *Visual Paradigm*	**Avença anual** Preço negociado. Para uma empresa pequena superior a 1000€

Legenda:

Solução ineficiente Solução parcial Boa Solução

[6] Licenças do Visual Paradigm (Visual Paradigm, 2016) são valores aproximados uma vez que se tratam da conversão Dólar para Euro realizada em Fevereiro de 2016

Como é possível avaliar, nenhuma das ferramentas é perfeita, tendo cada uma as suas forças e fraquezas. No entanto, a *Blueprints* (Blueprint Software Systems Inc., 2016) é realmente a ferramenta mais completa das comparadas, embora o valor para a sua aquisição seja muito elevado.

Por outro lado, o *Yakindu* (Itemis AG, 2015) é uma opção com um investimento muito inferior e pode ser integrado num ambiente de desenvolvimento através do eclipse, o que abre a possibilidade de integrar com inúmeras ferramentas como modelação, programação e mesmo gestão de projeto (por exemplo o *Jira* suporta integração no eclipse).

A aplicação da *Visual Paradigm* (Visual Paradigm, 2016) responde a uma parte substancial da gestão e especificação de requisitos, sendo uma solução de valor mais baixo que a *Blueprints* (Blueprint Software Systems Inc., 2016) e mais elevado que o *Yakindu* (Itemis AG, 2015).No entanto, não possui qualquer interligação para uma ferramenta de gestão de projeto de forma a atribuir tarefas a uma equipa. O controlo de versões é possível por ferramenta própria.

Para a realização de *mock-ups*, apenas o *Visual Paradigm* (Visual Paradigm, 2016) e o *Blueprints* (Blueprint Software Systems Inc., 2016) possuem soluções integradas, sendo na próxima secção comparadas as soluções neste tópico.

2.8 Mock-Ups

A questão da prototipagem é algo muito importante na especificação de requisitos, uma vez que através de uma imagem se consegue demonstrar aos diferentes *stakeholders,* de uma forma muito rápida, já que uma imagem é mais clara e intuitiva, minimizando erros oriundos das diferentes interpretações do que está escrito.

Através da sua utilização mitiga-se o risco de não ir de encontro ao imaginado pelos *stakeholders*. Por outro lado, permite definir *a priori* agregações de conceitos, terminologia incorreta, fluxo de operacionalidade da funcionalidade, entre outros aspetos pertinentes. Foram estudadas diversas ferramentas, sendo as ferramentas com mais relevância o *Balsamiq* (Balsamiq Studios, LLC, 2015), o *WireFrame Sketcher* (WireframeSketcher, 2015) e a integradas nas ferramentas *Visual Paradigm* (Visual Paradigm, 2016) e *Blueprint* (Blueprint Software Systems Inc., 2016), uma vez que o objetivo é de abranger todo o processo de requisitos.

2.8.1 Balsamiq

O *Balsamiq* (Balsamiq Studios, LLC, 2015) é uma ferramenta para a realização de *mock-ups* com centenas de componentes e bibliotecas gratuitas. Permite fazer bibliotecas próprias para utilização, possibilita a realização de *mock-ups* de forma muito rápida e possui também diferentes bases de contexto como *smartphone*, aplicação de *internet*, *tablet*, aplicação de computador, etc. Permite ainda reaproveitar blocos já realizados para usar como novos componentes e configurar os componentes base existentes, permitindo também a possibilidade de agregar vários *mock-ups* num projeto de forma a ter um agregador.

Mais detalhes e imagens sobre a ferramenta podem ser consultados nos anexo 1 – Balsamiq.

2.8.2 WireFrame Sketcher

O *Wireframe Sketcher* (WireframeSketcher, 2015) é uma ferramenta para a realização de *mock-ups* integrada com o eclipse. Ao criar um novo projeto pode-se logo escolher o tipo de projeto (Android, iOS, Web, etc.). Possui imensos componentes para o desenvolvimento do *mock-up* e permite

configurar algumas propriedades por componente e gerir a ordem a ser apresentada – *outline*, possui também uma visão agregadora de diversos *mock-ups* de forma a criar o *storyboard* e é possível ter vários *storyboards* por projeto usando os mesmos *mock-ups* o que permite o seu reaproveitamento.

Mais detalhes e imagens sobre a ferramenta podem ser consultados no anexo 2 – WireFrame Sketcher.

2.8.3 Visual Paradigm

O *Visual Paradigm* (Visual Paradigm, 2016) possui uma ferramenta integrada que possui o conceito de *storyboard* que permite criar vários *mock-ups* para esse *storyboard*. No final da criação de vários permite uma apresentação sequencial do *storyboard*. É também possível utilizar o mesmo *mock-up* para diferentes *storyboards*.

Permite realizar *mock-ups* para diferentes contextos (*web, tablet, smartphone*, aplicação de computador, etc.), possuindo componentes específicos para cada um dos mesmos.

Mais detalhes e imagens sobre a ferramenta podem ser consultados no anexo 3 – Visual Paradigm.

2.8.4 Blueprint Requirements Center

O Blueprint (Blueprint Software Systems Inc., 2016) possui uma ferramenta integrada que possui o conceito *Storyboard*, consiste no fluxo entre *mock-ups*. Permite também adicionar capturas de ecrã entre os mesmos e utilizar o mesmo *mock-up* ou captura de ecrã em diferentes fluxos, para demonstrar ao *stakeholder* o comportamento. Possui um conjunto limitado de componentes para realizar os *mock-ups*. Mais informação pode ser encontrada em B.4 – Blueprint Requirements Center.

2.8.5 Avaliação das Ferramentas de Mock-up

Tabela 3 – Tabela comparativa de software de mock-ups.

	Balsamiq	WireFrame Sketcher	Visual Paradigm	Blueprint
Instalação	Aplicação Própria	Integrado com Eclipse ou aplicação própria	Integrada na aplicação de requisitos do *Visual Paradigm*	Aplicação Web
Controlo de versões	Não	Integrado com *Git* no Eclipse (sem pré-visualização do histórico)	Não	Limitado
Componentes	Existem imensas bibliotecas passíveis de ser instaladas e é possível criar bibliotecas próprias	Muitos componentes, mas não é fácil criar componentes próprios	Existem poucos componentes	Existem poucos componentes
Velocidade	Rápido	Médio	Médio	Médio
Reutilização de *mock-ups* em vários *storyboards*	Não	Sim	Sim	Sim
Outros	Plugin para *Jira*, *Confluence* e *Google-drive*			Gestão de capturas de ecrã

	Balsamiq	WireFrame Sketcher	Visual Paradigm	Blueprint
Licença[7]	**Licença Vitalícia:** 1 utilizador – 79€ 10 utilizadores – 709€ 100 utilizadores – 3944€ **Online (mês):** 3 projetos – 11€ 100 projetos – 177€	**Licença Vitalícia:** 1 utilizador – 88€ 10 utilizadores –789€ 100 utilizadores –7888€	**Licença Vitalícia:** 1 utilizador – 709€ Multiutilizador – 886€ Online – 70901€ **Avença mensal** 1 utilizador – 32€/Mês Integrado com o *Visual Paradigm*	**Licença:** Preço negociado. Para uma empresa pequena superior a 1000€

Legenda:

Solução ineficiente | Solução parcial | Boa Solução

No caso da escolha da ferramenta de requisitos da *Visual Paradigm* (Visual Paradigm, 2016), a escolha torna-se clara na ferramenta para *mock-ups*. *Balsamiq* (Balsamiq Studios, LLC, 2015) é sem dúvida uma das mais poderosas ferramentas do mercado, uma vez que é possível reutilizar componentes. Estes componentes podem ser criados no próprio *Balsamiq* (Balsamiq Studios, LLC, 2015), assim como existe uma panóplia de modelos gratuitos. Entre *storyboards* não é possível reaproveitar *mock-ups*. O *Wireframe Sketcher* (WireframeSketcher, 2015) pode ser integrado em diversos ambientes como, por exemplo, o eclipse. Não possui tantos componentes como o *Balsamiq* (Balsamiq Studios, LLC, 2015) nem a sua customização, no entanto, não deixa de ser uma excelente opção.

2.9 Conclusão na Escolha de uma Ferramenta de Requisitos

No caso do *Blueprints* (Blueprint Software Systems Inc., 2016), tem-se uma solução muito completa para todo o ciclo de vida de requisitos e processo, desde a identificação até à validação e teste, no entanto, é a ferramenta mais dispendiosa.

No caso do *Visual Paradigm* (Visual Paradigm, 2016), existe uma falta de integração com ferramentas de gestão de projeto e a sua gestão de requisitos fica um pouco atrás das restantes soluções, pelo que sempre dependerá de uma ferramenta de gestão de projeto.

O *Modelio* (Modelio, 2015) tem também um défice no que diz respeito à integração com uma ferramenta de gestão de projeto, sendo, no entanto, muito relevante na ligação com a modelação UML, uma vez que nessa área é muito completa.

Podemos verificar que o *Yakindu* (Itemis, 2015) não é uma ferramenta completa, no entanto, como se trata de um *plugin* do eclipse, a solução passa pela sua utilização em conjunto com o *Wireframe Sketcher* (WireframeSketcher, 2015) e uma ferramenta de gestão de projeto como o *Jira* (Atlassian Pty Ltd, 2015). Estas ferramentas aqui identificadas podem ser integradas no eclipse. Esta poderá ser a solução menos dispendiosa.

O *Balsamiq* (Balsamiq Studios, LLC, 2015) é uma excelente ferramenta para a realização de *mock-ups*, no entanto, a sua utilização está sempre isolada das restantes ferramentas de gestão de requisitos.

[7] Valores aproximados uma vez que se tratam da conversão Dólar para Euro realizada em Fevereiro de 2016

2.10 Segurança e Avaliação da Qualidade de Código

A questão da segurança em *software* é algo de extrema relevância, uma vez que pode trazer custos incalculáveis a uma empresa, sobretudo na exposição do nome da empresa. A injeção de código ou o acesso a dados sensíveis são alguns dos cenários críticos. Foi estudada uma *framework* sobre a área para identificar os problemas e formas de os resolver, o *OWASP* (Open Web Application Security Project, 2015), que é utilizada sobretudo para o mundo Web e uma vez que cada vez mais se trabalha em aplicações web (e as regras identificadas também podem ser extrapoladas para outros tipos de aplicação).

2.10.1 OWASP

A *OWASP* (Open Web Application Security Project, 2015) é uma organização não lucrativa focada em melhorar a segurança do *software*.

2.10.1.1 Os 10 Ataques mais Comuns

Os erros de segurança mais comuns nas aplicações, respetiva análise de impacto, exemplificação do problema, referências e soluções possíveis são disponibilizados na página de *internet* da OWASP (Open Web Application Security Project, 2015). Os 10 ataques mais comuns são (Ionescu, 2015):

- **Injeção de Código** – Injeções de código SQL, OS, LDAP que ocorrem quando dados não fidedignos são enviados a um interpretador como parte de um comando ou instrução. O ataque pode despoletar a execução de comandos não desejados ou aceder a dados sem autorização;
- **Autenticação violada e gestão de sessão** – As funcionalidades de autenticação e gestão de sessão são muitas vezes implementadas incorretamente, permitindo ao atacante comprometer chaves, palavras-chave, *tokens* de sessão ou tirar proveito de erros de implementação para assumir a identidade de outro utilizador;
- **Scripts Cross-site (XSS)** – Erros XSS ocorrem quando a aplicação usa dados não fidedignos e envia-os para um navegador de *internet* sem validação ou "*escaping*" (se não for *escaped* pode ser interpretado como código e executado). Estes ataques permitem executar *scripts* no navegador de *internet* da vítima, permitindo roubar sessões de utilizador, deformar *websites* ou redirecionar o utilizador para sites maliciosos;
- **Referência direta de um objeto insegura** – Uma referência direta de um objeto ocorre quando uma referência interna de desenvolvimento, tais como um ficheiro, diretório ou chave da base dados é exposta. Sem um controlo de acesso para verificar, os atacantes podem manipular a referência para aceder a dados não autorizados;
- **Más configurações de segurança** – Boa segurança requer existir uma configuração segura bem definida e aplicada nas *frameworks* utilizadas, servidor aplicacional, servidor *web*, servidor de base de dados e plataformas. Estas definições devem ser definidas, implementadas e mantidas, sendo os valores por defeito normalmente inseguros;
- **Exposições de dados sensíveis** – Muitas aplicações não protegem os dados sensíveis, como os números de cartões de crédito, credenciais de autenticação, etc. Os atacantes podem roubar ou modificar os dados para realizar fraude, roubo de identidade ou outros crimes. Dados sensíveis merecem proteção extra como encriptação;
- **Faltas de controlo de acesso por funcionalidade** – A maioria das aplicações verificam o nível de acesso antes de disponibilizar a funcionalidade, no entanto, as aplicações necessitam de comunicar com o servidor sempre que verificam as permissões. Se os pedidos não forem

verificados, o atacante pode forjar pedidos de forma a aceder a funcionalidades sem permissão;

- **Falsificação de pedido de *Cross-site (CSRF)*** – Um ataque de CSRF força o explorador de *internet* de uma vítima com a autenticação estabelecida e envia um pedido *http*, incluindo as *cookies*[8] da vítima e outros dados de autenticação. Isto permite ao atacante forçar o explorador de *internet* da vítima a gerar pedidos que a aplicação considera legítimos;
- **Utilização de componentes com vulnerabilidades conhecidas** – Componentes como bibliotecas, *frameworks* e outros módulos de *software* quase sempre correm com todos os privilégios. Se um componente vulnerável é fácil de tirar partido, pode facilitar uma grande perda de dados ou obter acesso total ao servidor. Esta vulnerabilidade permite causar quase todos os riscos imagináveis;
- **Redireccionamentos e Reencaminhamentos não validados** – Aplicações web frequentemente redirecionam ou reencaminham utilizadores para outras páginas de *internet,* usando dados não fidedignos para determinar a página destino. Sem a validação apropriada, o atacante pode redirecionar vítimas para ataque de *phishing*[9] ou reencaminhar para *websites* de *malware*[10].

2.10.1.2 Avaliação de Risco de uma Funcionalidade

De acordo com os riscos identificados pela organização OWASP (Open Web Application Security Project, 2015), desenvolveu também uma metodologia para catalogação do risco, dividida em 6 passos:

- Identificação do risco;
- Fatores para estimar a probabilidade;
- Fatores para estimar o impacto;
- Determinar a severidade do risco;
- Decidir o que corrigir;
- Personalizar o modelo de catalogação do risco.

O primeiro passo trata a **identificação do risco**. É necessário recolher informação acerca da ameaça existente, o ataque a ser usado, a vulnerabilidade envolvida e o impacto de um acesso inadvertido. Podem, existir vários grupos de atacantes e diferentes impactos possíveis. De forma geral é sempre melhor usar o pior cenário possível uma vez que vai gerar o maior número de risco.

Existem dois conjuntos de fatores principais, o agente de ameaça envolvido e os fatores de vulnerabilidade. Estes decompõem-se em vários níveis para estimar a probabilidade com um peso de 1 a 9 tais como (Open Web Application Security Project, 2016):

- Agente de ameaça envolvido:
    - Nível de técnica que ele possui:
        - Competências de penetração na segurança (9);
        - Competências de rede e programação (6);
        - Utilizador avançado de computador (5);

[8] Pequeno conjunto de dados enviados por um *site* e armazenado no computador do utilizador.

[9] Ataques para obter informações e dados pessoais importantes através de mensagens falsas. Criminosos conseguem obter nomes de utilizadores e respetivas senhas, cartões de crédito ou contas bancárias.

[10] Malware é um *software* malicioso.

        - Algumas habilidades técnicas (3);
        - Sem competências técnicas (1).
    - Motivo que possui para encontrar e tirar partido:
        - Alto (9);
        - Médio (4);
        - Baixo (1).
    - Oportunidade e recursos necessários:
        - Sem acesso ou sem necessidade de recursos (9);
        - Algum acesso ou recursos necessários (7);
        - Acesso especial ou recursos necessários (4);
        - Acesso total, ou recursos necessários muito dispendiosos (0).
    - Tamanho e caraterísticas do grupo:
        - Programadores ou administradores de sistemas (2);
        - Utilizadores da intranet (4);
        - Parceiros (5);
        - Utilizadores autenticados (6);
        - Utilizadores anónimos (9).
- Fatores de vulnerabilidade:
    - Facilidade de ser encontrada a vulnerabilidade:
        - Através de ferramentas automáticas (9);
        - Fácil (7);
        - Difícil (3);
        - Praticamente impossível (1).
    - Facilidade de tirar partido da vulnerabilidade:
        - Através de ferramentas automáticas (9);
        - Fácil (7);
        - Difícil (3);
        - Teoricamente (1);
    - Conhecimento da vulnerabilidade:
        - Conhecimento público (9);
        - Óbvio (7);
        - Escondido (3);
        - Desconhecido (1).
    - Deteção de intrusos:
        - Não registado (9);
        - Registado sem revisões dos registos (8);
        - Registado e revisto (3);
        - Deteção ativa pela aplicação (1).

Os fatores para determinar um impacto subdividem-se em impactos técnicos e de negócio:

- Fatores Técnicos (dados utilizados e funcionalidades disponibilizadas):
    - Perca de confidencialidade:
        - Obtenção de todos os dados (9);
        - Obtenção de dados críticos (7);
        - Obtenção de poucos dados críticos (6);

        - Obtenção de dados não sensíveis (6);
        - Obtenção de dados mínimos não sensíveis (2).
    - Perca de integridade:
        - Corrupção de todos os dados (9);
        - Dados corrompidos extensivamente (7);
        - Alguns dados corrompidos (5);
        - Poucos dados corrompidos (3);
        - Dados corrompidos insignificantes (1).
    - Perca de disponibilidade:
        - Todos os serviços perdidos (9);
        - Serviços primários interrompidos (7);
        - Serviços secundários interrompidos (5);
        - Poucos serviços primários interrompidos (5);
        - Poucos serviços secundários interrompidos (1).
    - Encontrar o autor:
        - Completamente anónimo (9);
        - Possível de encontrar (7);
        - Facilmente encontrado (1).
- Fatores de Negócio:
    - Impacto financeiro:
        - Bancarrota (9);
        - Efeito significativo no lucro anual (7);
        - Efeito menor no lucro anual (3);
        - Menor que o custo de corrigir a vulnerabilidade (1).
    - Impacto na reputação:
        - Por em causa a marca (9);
        - Perda de confiança (5);
        - Perda das contas principais (4);
        - Impacto mínimo (1).
    - Não conformidade:
        - Alta violação dos perfis (7);
        - Clara violação (5);
        - Violação menor (2).
    - Violação da privacidade:
        - Milhões de pessoas (9);
        - Milhares de pessoas (7);
        - Centenas de pessoas (5);
        - Um indivíduo (3).

Usando os fatores anteriores é possível calcular a gravidade do risco através de uma matriz entre os pesos atribuídos ao risco de impacto e a probabilidade onde, entre 0 e 3 é baixo, entre 3 e 6 é médio e entre 6 e 9 é alto (Tabela 4).

Após os riscos estarem classificados, é identificada uma lista de prioridades das necessidades de resolução, sendo regra geral, ordenados por severidade. Existem erros que não valem a pena corrigir devido ao esforço necessário.

Tabela 4 – Tabela para cálculo da gravidade do risco.

Impacto Alto	Médio	Alto	Crítico
Impacto Médio	Baixo	Médio	Alto
Impacto Baixo	Nota	Baixo	Médio
	Probabilidade Baixa	Probabilidade Média	Probabilidade Alta

Este processo de avaliação é normalmente usado para avaliar as funcionalidades existentes ou no final do seu desenvolvimento. Esta dissertação irá propor uma outra forma de abordar a temática, colocando esta análise antes do desenvolvimento fazendo com que este seja realizado respeitando as condições estipuladas

2.10.1.3 Open SAMM

Esta organização desenvolveu também uma *framework* destinada a ajudar as empresas a formular e implementar estratégias de segurança de *software* dependendo dos riscos. Esta *framework Open SAMM – Software Assurance Maturity Model* (OpenSAMM Project, 2009), ajuda a:

- Avaliar o *software* que uma organização possui;
- Apoiar na construção de um programa para assegurar o desenvolvimento de um *software* seguro;
- Demonstrar melhorias concretas no processo de segurança de *software*;
- Definir e medir atividades de segurança dentro de uma organização.

Esta *framework* define uma série de atividades a serem realizadas durante o desenvolvimento de *software* (Figura 3).

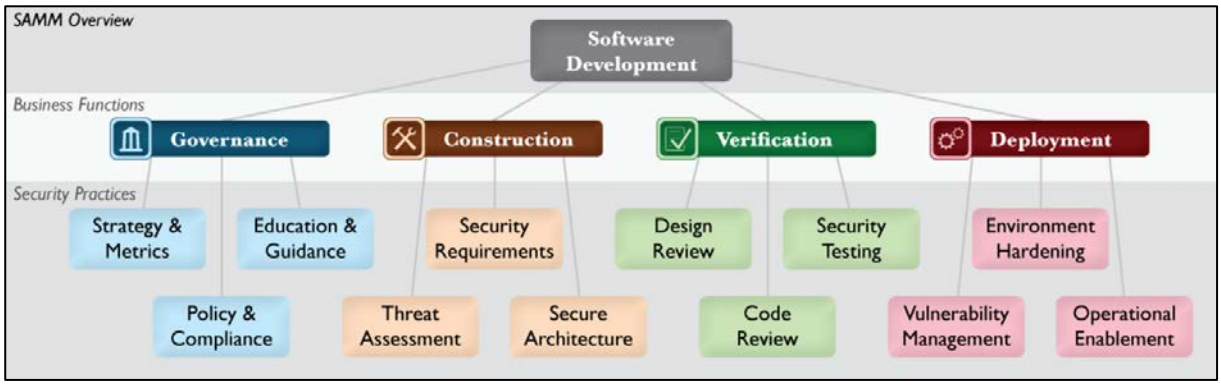


Figura 3 – Framework Open SAMM (OpenSAMM Project, 2009).

Na área da gestão existem os seguintes pontos a considerar:

- Estratégia e métricas – Estabelecer um plano de entregas estratégico e unificado para a segurança dentro da organização, definir a tolerância ao risco, medir o valor dos dados e ativos de *software* e alinhar as despesas de *software* com indicadores de negócio relevantes;
- Políticas e Conformidade – Identificar o que motiva a organização, estabelecer o nível base de segurança e conformidade compreendendo os riscos por projeto, medir os projetos comparando com políticas de organização e *standards*;

- Educar e guiar os recursos da empresa – Disponibilizar acesso aos recursos de segurança e implementação às pessoas constituintes do projeto, educar as pessoas no ciclo de vida do *software* com orientação ao papel na empresa e disponibilização de formação.

Na área de implementação:

- Avaliar a ameaça – Identificar e perceber as ameaças de alto-nível para a organização e para os projetos;
- Requisitos de segurança – Considerar a segurança de forma explícita durante o processo de requisitos e aumentar a granularidade dos requisitos de segurança, derivados da lógica de negócio juntamente com os riscos conhecidos;
- Arquitetura Segura – Introduzir apoio de especialistas no processo de desenho, direcionar o desenho para estruturas e serviços conhecidos e seguros e controlar o desenho de forma formal, validando a utilização de componentes seguros.

Na área da verificação:

- Revisão do desenho – Suportar revisões do desenho de *software* recorrentes para mitigar riscos conhecidos, disponibilizar tarefas de revisão usando as boas práticas de segurança e desenvolver diagramas para recursos sensíveis;
- Revisão do código – Descobrir vulnerabilidades ao nível do código, utilizar ferramentas de análise de código e integrar a prática durante o desenvolvimento;
- Testes de segurança – Estabelecer um processo para elaborar testes de segurança básicos na implementação e requisitos de *software*, testar durante o desenvolvimento de forma mais completa e eficiente através de testes de automação, usar ferramentas específicas para teste antes da implementação.

Na área de implantação:

- Gestão de vulnerabilidades – Perceber o plano para responder a relatório de vulnerabilidade e incidentes, elaborar pedidos para melhorar a consistência e comunicação e melhorar a análise e a recolha de dados para obter informação para usar no planeamento;
- Melhorar o ambiente – Perceber a base operacional do ambiente para aplicações e componentes de *software*, melhorar a confiança nas operações da aplicação no ambiente e validar a saúde da aplicação e estado do ambiente operacional segundo as melhores práticas;
- Capacitação operacional – promover comunicação entre o desenvolvimento e utilizadores para dados críticos, melhorar as expectativas, melhorar operações de segurança através de procedimentos detalhados, comunicar e validar os artefactos realizados.

Esta *framework* explica para cada um dos passos, o que realizar, quais os resultados esperados, quais as métricas de sucesso, a formação necessária por pessoa e respetivos custo. Estes pontos são muito importantes para a gestão de projeto e para decisões de topo.

A avaliação da ameaça e os requisitos de segurança devem começar a ser tratadas na parte do processo correspondente à identificação de requisitos. A arquitetura de segurança deve ser englobada na parte de modelação.

A parte de verificação deve já ocorrer dentro do processo de desenvolvimento, devendo ser realizada da forma mais automática possível através de:

- Avaliação de código através de ferramentas;

- Realização de testes unitários;
- Realização de testes automáticos funcionais:
    - Incluir o fluxo normal da aplicação;
    - Incluir fluxos de erro e validação;
    - Incluir utilização incorreta da aplicação.
- Realização de testes de segurança.

Na parte de implantação, obtém-se o novo ciclo de desenvolvimento, a identificação de novos requisitos ou melhorias ao sistema para entrar em novo desenvolvimento.

2.10.2 Avaliação da Qualidade do Código

Para a avaliação da qualidade do código, é necessário obter métricas de qualidade, identificar problemas de segurança, realizar a revisão do código de forma muito eficiente e utilizar ferramentas que possibilitam esta melhoria. Existem diversas ferramentas, tais como *Codacy* (Codacy, 2016), *SQuORE* (SQUORING Technologies, 2016), *Semmle* (Semmle Ltd, 2016) e *Sonarqube* (Sonarsource SA, 2016).

Com a utilização destas ferramentas é possível obter métricas sobre a qualidade do código produzido e validar de forma automática, bem como identificar os possíveis erros de segurança. Esta validação pode ser de extrema importância ou não, dependendo do tipo de dados que são necessários tratar. Por exemplo, no caso de dados sensíveis deve ser assegurado que o código não possua erros de segurança, pois existe um risco elevado. Por outro lado, existem zonas em que a segurança da informação não tem qualquer importância, dado a informação ser pública.

Devem ser construídas as regras de avaliação do código de acordo com a análise do risco, garantindo assim a qualidade da aplicação.

Como existe um esforço grande para que o código esteja totalmente correto de acordo com as regras identificadas, e porque muitas vezes não há tempo de o fazer, é apresentada uma forma de identificar e quantificar o que é mais importante, de forma a calcular o valor mínimo para aceitação da implementação do requisito.

Uma das ferramentas mais utilizadas é o *Sonarqube*, demonstrada no anexo C.1 – SonarQube.

3 Design da Solução

Os métodos estudados da engenharia de requisitos identificam as várias fases da engenharia de requisitos, mas não a forma como esta poderá ser conseguida em termos de conceptualização processual com as diferentes áreas da análise de negócio. Por outro lado, a definição da análise de negócio estudada identifica os grandes blocos e refere a ordem cronológica em que estas ocorrem, mas carecem da junção com a engenharia de requisitos, assim como não coloca num modelo conceptual com a divisão e definição clara dos diferentes entregáveis necessários para que exista rastreabilidade de requisitos. Existe assim um défice na integração destas duas grandes áreas e na sua rastreabilidade, impedindo assim uma eficaz gestão de requisitos. Foi então proposto um fluxo que visa colmatar estas lacunas que pode ser verificado na Figura 4. Este fluxo deverá ser seguido pela pessoa que realiza a especificação de requisitos, sendo apenas alguns dos tópicos possíveis de rastrear (aqueles que produzem entregáveis). Os passos não rastreáveis dão origem a um documento final na gestão de requisitos, no entanto, estes dependem da interação com entidades externas.

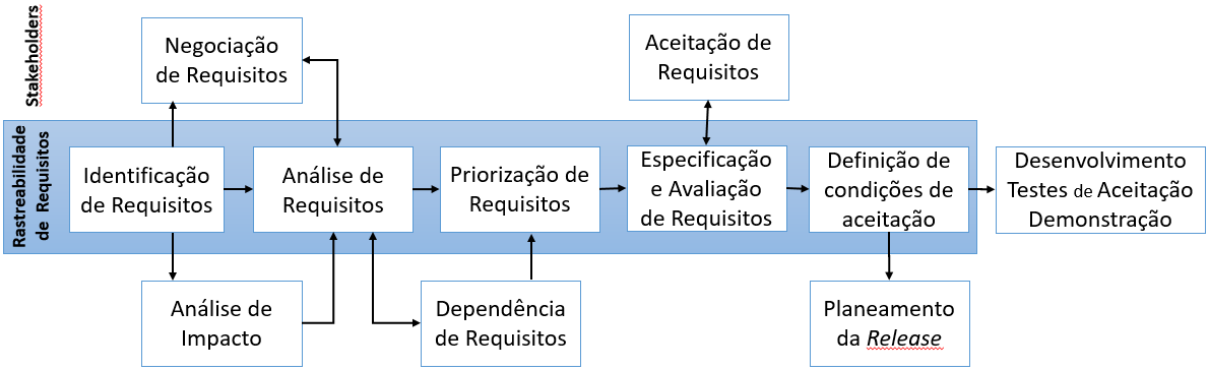


Figura 4 – Fluxo de gestão de requisitos.

3.1 Gestão de Requisitos

A gestão de requisitos deve gerir todas as fases do processo de análise de negócio e de requisitos.

Existem algumas práticas-chave para uma eficiente gestão de requisitos, devendo ser criado um lote de requisitos revistos, aprovados e aceites pelos *stakeholders* para uma *release,* para de seguida:

- Gerir as diferentes versões dos documentos de requisitos;
- Adotar e seguir à risca um processo de gestão de mudança;
- Realizar sempre análises de impacto sobre os requisitos;
- Armazenar os atributos dos requisitos;
- Controlar o estado e fase de cada requisito;
- Seguir o requisito no desenho, desenvolvimento e testes;
- Usar uma ferramenta ou modelo para gestão de requisitos;

Deve ser assegurado que todos os documentos produzidos estão disponíveis para todos, atualizados e a sua edição restrita a utilizadores autorizados. Versões não controladas causam problemas como a necessidade de refazer trabalho, qualidade degradada e falta de segurança nos compromissos das entregas por parte dos responsáveis de desenvolvimento (Wiegers, 2015).

A gestão de alteração de requisitos deve, idealmente, ser proposta, revista e aprovada e incluir sempre uma análise de impacto e suportada por uma ferramenta, salvaguardando que uma ferramenta não é um processo, e que o processo deverá ser sempre respeitado (Wiegers, 2015).

Uma alteração de requisitos pode resultar de uma renegociação de requisitos dependentes com os *stakeholders* dessas dependências.

3.2 Identificação de Requisitos

A identificação de requisitos é o processo de aprendizagem, extração e descoberta das necessidades dos *stakeholders* (Hickey & Davis, 2004), assim como a extração de conhecimento de diversos documentos (Finkelstein & Potts, 1985) ou outras fontes. Incluem práticas como *workshops* e interação com os *stakeholders*. Os requisitos identificados a partir de *stakeholders* ou clientes normalmente resultam apenas numa fração dos requisitos a desenvolver, outros são transmitidos através de sessões de transferência de conhecimento, casos de uso ou pedidos de desenvolvimento. É também possível extrair requisitos através da análise do sistema atual, de documentos da área, demonstrações das funcionalidades desenvolvidas, questionários, técnicas de obtenção iterativa e interação direta com o sistema (utilização do sistema).

A premissa base no levantamento de requisitos começa com a sua identificação. Uma estrutura globalmente aceite é a utilização de *user stories*:

As a** <user role> **I want to** <action/behaviour> **so that** <benefit>**.

Em português e exemplificando a identificação de requisitos:

Como um **comprador** eu quero **comprar** um **novo livro**
pagando com **cartão de crédito** de forma a poder desfrutar da leitura do livro.

Identificamos aqui o **utilizador** (comprador), a **ação** (comprar), os **dados** (novo livro e cartão de crédito).

Extraindo as regras de negócio: A data de expiração do cartão de crédito é em 10/10/2020.

Extraindo as interfaces: Sistema de interface com o sistema de inventário do produto.

Extraindo o ambiente: Comprar online.

Extraindo atributos de qualidade (performance neste caso): Em menos de dois minutos.

Estes requisitos também podem ser derivados de transições entre estados, por exemplo, no exemplo anterior, o estado do livro após a encomenda pode criar vários requisitos como:

Como um **comprador** eu quero **ver o estado da minha encomenda** de forma a poder saber o estado em que ela se encontra.

Esta premissa não é tão facilmente aplicada a requisitos não funcionais, pelo que nesses casos normalmente não é utilizada ou usada com o preenchimento do benefício opcional.

3.3 Negociação de Requisitos

Uma **negociação de requisitos** é realizada sempre que são identificados conflitos entre *stakeholders,* sendo da responsabilidade do gestor de requisitos estabelecer pontes e utilizar técnicas para apoiar na negociação entre eles (Boehm & Egyed, 1998). Existem quatro pontos a proceder quando acontece um conflito (Paul Grünbacher, 2005):

- Identificação dos conflitos – Expor e descrever os conflitos existentes entre *stakeholders*;
- Análise dos conflitos – Verificar a forma e como os conflitos ocorrem;

- Resolução de conflitos – Identificar a forma de resolver os conflitos e negociar com os diferentes *stakeholders* uma forma de compatibilização;
- Documentação da resolução de conflitos – Deve existir uma documentação sobre os conflitos encontrados que deverá incluir os passos anteriores e a aceitação de todos os *stakeholders*.

Em cada iteração de desenvolvimento e na entrada de novos requisitos a prioridade deve ser reavaliada.

3.4 Análise de Impacto

Quando um produto é vendido, novos requisitos podem levar a soluções com impacto no produto atual. Adicionar uma funcionalidade provavelmente irá ter impacto em outras funcionalidades (van Gurp, et al., 2001). Os impactos podem ser de um conjunto de pequenas configurações, até um efeito em cascata (Fayad, 2002) que poderá causar alterações significativas a todo o produto.

A **análise de impacto** deve (Wiegers, 2015):

- Identificar impactos com requisitos implementados;
- Identificar desenho da aplicação afetada, código e testes;
- Impacto na interface de utilizador, base de dados, relatórios, etc.;
- Identificar outros sistemas, bibliotecas, ou *hardware* afetado;
- Verificar se a alteração irá ter penalizações nos recursos do computador para desenvolvimento, testes ou ambientes de produção;
- Verificar se é tecnicamente possível;
- Verificar se irá afetar a qualidade ou performance da aplicação;
- Verificar se viola regras de negócio;

Alguns destes impactos poderão ser realizados por diferentes pessoas, normalmente detentoras do conhecimento de determinada área, como áreas técnicas, de negócio ou infraestrutura.

3.5 Dependência de Requisitos

A interdependência entre requisitos consiste numa dependência entre um ou mais requisitos, podendo tomar diversas formas (Carlshamre, et al., 2001):

- AND – um grupo de requisitos que necessita de ser satisfeito completamente;
- REQUIRES – um requisito que é necessário para ter outro;
- TEMPORAL – um requisito é necessário antes de outro;
- CVALUE – um requisito perturba o valor de outro;
- ICOST – um requisito aumenta o custo de outro requisito ou o projeto em global;
- OR – pelo menos um dos requisitos do grupo tem de ser satisfeito

Com a dependência de requisitos é possível perceber as dependências existentes, permitindo rapidamente identificar a ordem de desenvolvimento.

3.6 Análise de Requisitos

Análise de requisitos é o processo de investigar o domínio do problema e requisitos de forma a desenvolver um melhor entendimento dos objetivos, necessidades e espectativas dos *stakeholders* (Aurum & Wohlin, 2005).

Nesta fase são definidos os objetivos, a estrutura de negócio – normalmente através de um PBS (*product breakdown structure)* ou RBS (*requirement breakdown structure)* – e os requisitos são verificados e detalhados.

Caso na análise de impacto se encontre uma solução completa ou parcial para os requisitos pedidos, esta é apresentada aos *stakeholders* para validar se satisfazem o propósito pretendido e negociado qual o défice necessário implementar. Caso a análise retorne condicionantes tecnológicas, limitações ou outros constrangimentos, são analisadas alternativas para serem apresentadas aos *stakeholders* e negociadas com os mesmos.

3.6.1 Processo Ágil da Análise de Requisitos

Um processo ágil de requisitos foca-se num ciclo entre a descoberta e a entrega. Na descoberta é necessário identificar as necessidades, analisar as mesmas, aprender e investigar sobre o assunto e planear para depois ser realizado e entregue. Durante o desenvolvimento é necessário acompanhar e no final avaliar para extrair mais necessidades dos *stakeholders* (Figura 5) (Gottesdiener, 2015).

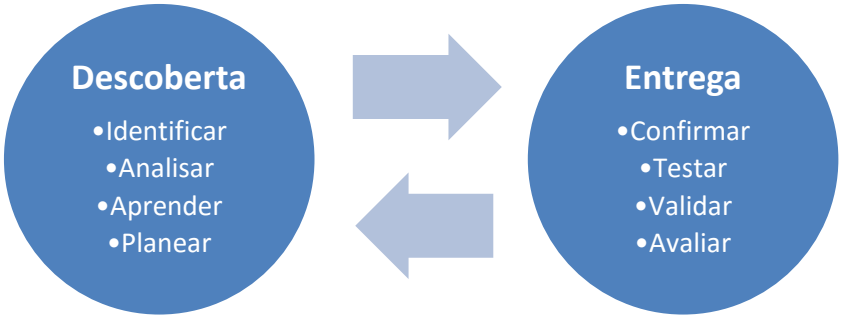


Figura 5 – Processo ágil da especificação de requisitos.

Desta forma, temos entregas sucessivas, satisfazendo as necessidades mais importantes dos *stakeholders* e entregando valor constante e iterativo aos seus desejos.

Algo muito importante em desenvolvimentos ágeis é a alimentação do *backlog*, algo que normalmente na metodologia SCRUM é da responsabilidade do *Product Owner*. O *backlog* deve ser alimentado por problemas e objetivos de negócio (Beatty, 2015).

3.7 Priorização dos Requisitos

Após a organização e análise dos requisitos a desenvolver é necessário priorizar junto dos *stakeholders* quais os prioritários para se poder realizar a especificação. É necessário negociar quais os mais importantes, quais podem abdicar e quais são cruciais para desenvolvimento.

Em equipas ágeis o *backlog* é o principal *input* de uma equipa de desenvolvimento, sendo a lista constantemente priorizada de acordo com a estratégia da empresa e as necessidades do *stakeholder*. Fora de equipas ágeis deve ser mantido de igual forma uma lista de requisitos à entrada.

Existem muitas práticas e técnicas com vários graus de formalização, alguns exemplos práticos são a escala MoSCoW (Clegg & Barker, 1994), análise de custo-benefício (Karlsson & Ryan, 1997) e o teste dos 100 dólares (Berander & Andrews, 2005).

3.8 Especificação de Requisitos

A especificação de requisitos é o processo formal de documentação dos requisitos (Vessey & Conger, 1994), onde é definido como estes vão funcionar, como são apresentados e como serão realizados. É necessário realizar elementos para apoio na compreensão da necessidade, suporte ao desenvolvimento e redução da ambiguidade para os diferentes intervenientes. Algumas das formas possíveis são:

- Indicação das pré-condições pós-condições e pressupostos;
- Diagramas e esquemas para aquisição de conhecimento (normalmente estes são apenas realizados até obtermos a perceção do que é necessário, não sendo usado para outros intervenientes estando sempre incompletos);
- Diagramas de fluxo;
- Diagramas de interação utilizador-máquina;
- Captura de *user stories* (captura de quem, o quê e porquê de um requisito);
- *Mock-ups* de ecrãs e de agregação de conceitos de negócio;
- Documentos de análise das funcionalidades existentes;
- Identificação de cenários não contemplados (GAP's) na implementação existente;
- Alterações necessárias às funcionalidades existentes.

Alguns destes diagramas são usados para negociação com os *stakeholders* – diagramas e esquemas para aquisição de conhecimento, diagramas de trabalho e fluxo, *mock-ups*, diagramas de interação utilizador máquina, demonstração de funcionalidades similares existentes, etc.

Outros são para suporte ao desenvolvimento – diagramas de fluxo, interação utilizador-máquina, *mock-ups*, identificação de GAP's na implementação, alterações necessárias às funcionalidades existentes, etc.

É sempre necessário avaliar cada ponto, com a pergunta "E se?", de forma a identificar caminhos alternativos.

3.8.1 Avaliação de Requisitos

A avaliação de requisitos consiste num conjunto de práticas para testar a qualidade das especificações (Paetsch, et al., 2003), sendo avaliados se estão completos, consistentes, descritivos do pretendido, são mensuráveis, rastreáveis e possuem as pré e pós condições. No final da implementação são usados para validar que foi desenvolvido conforme o pretendido. Alguns dos elementos criados na especificação de requisitos podem ser usados nesta fase, como os *mock-ups*, diagramas de fluxos, cenários de teste, assim como outras formas de validar a implementação da especificação. Os diagramas de fluxo são usados para validar todos os fluxos possíveis, no entanto, por vezes apenas é possível validar parte deles. Neste caso deve ser definido quais os cenários críticos a serem testados.

3.8.2 Aceitação de Requisitos

Os requisitos deverão ser apresentados aos *stakeholders* de forma a serem validados e aceites. Caso não o sejam, devem ser alterados até que tal aconteça.

3.9 Definição de Condições de Aceitação

Nesta fase define-se as condições de aceitação para serem utilizados na fase de avaliação. Pode ser usado um formato semelhante da premissa de identificação de requisitos para ser garantida pela qualidade (Beatty, 2015).

Dado <pré-condição> **quando** <condição> **então** <resultado>.

E seguindo o exemplo dado, pode-se definir como critério de aceitação,

Dado o pagamento por cartão de crédito **quando** o pagamento é recusado **então** é apresentada uma mensagem de erro ao utilizador.

Nesta fase devem também ser colocadas as condições de aceitação de performance, qualidade, cobertura de testes unitária mínima, etc.

3.10 Planeamento da Release

O planeamento da r*elease* é um grupo de práticas específicas para *software* de produto. Após a priorização dos requisitos, estes são planeados para o desenvolvimento do produto (Günther & Saliu, 2005), onde os responsáveis pelas equipas de desenvolvimento devem assumir um compromisso formal relativo a entregas periódicas de desenvolvimentos, juntamente com as estimativas necessárias de pessoas necessárias e orçamento para o cumprir. É sempre necessário manter atualizado o tempo estimado necessário para cada um dos requisitos, incluindo a criação de desenho, desenvolvimento e testes para criação ou modificação, resolução de erros e documentação (Wiegers, 2015).

Quer os requisitos sejam parte da próxima ou de uma *release* futura, depende dos fatores usados para priorização, como o valor da solução para os utilizadores e o custo de desenvolvimento. O processo de requisitos permite à cabeça estimar os custos de desenvolvimento, de forma a possibilitar prever o que poderá ficar dentro de determinada *release* ou passar para uma futura.

3.11 Desenvolvimento

Esta é a fase onde a arquitetura é desenhada e o código é escrito de forma a implementar o que se encontra especificado nos requisitos.

3.12 Testes de Aceitação

A implementação realizada sobre os requisitos deverá ser de acordo com a especificação original. Esta pode ser considerada a fase de testes de aceitação (Marciniak & Shumskas, 1994). Este ponto não pertence ao processo de requisitos, mas sim ao processo de desenvolvimento de *software*, onde as pessoas responsáveis pela qualidade de *software* irão desenhar os cenários de teste e validar a implementação, de acordo com os requisitos, de forma a recusarem ou aceitarem o desenvolvimento.

3.13 Demonstração

De forma a concluir o processo de desenvolvimento de *software,* deverá ser realizada uma demonstração para obter a validação do gestor de requisitos que a levará para aprovação junto aos *stakeholders*.

3.14 Rastreabilidade de Requisitos

A rastreabilidade é um conjunto de atividades que apoiam o acompanhamento do requisito ao longo do seu ciclo de vida, quer dentro do projeto, quer entre projetos (Gotel & Finkelstein, 1994).

Isto inclui acompanhamento em todas as fases do processo já identificadas no fluxo da Figura 4. A partir deste fluxo e após identificação dos pontos que possuem entregáveis é necessário estruturar a informação para servir todas as fases do processo. A proposta do processo e modelo passa por dividir

alguns dos pontos em fases mais atómicas e dar resposta aos pontos da Figura 6 a azul-escuro, através de entregáveis distintos:

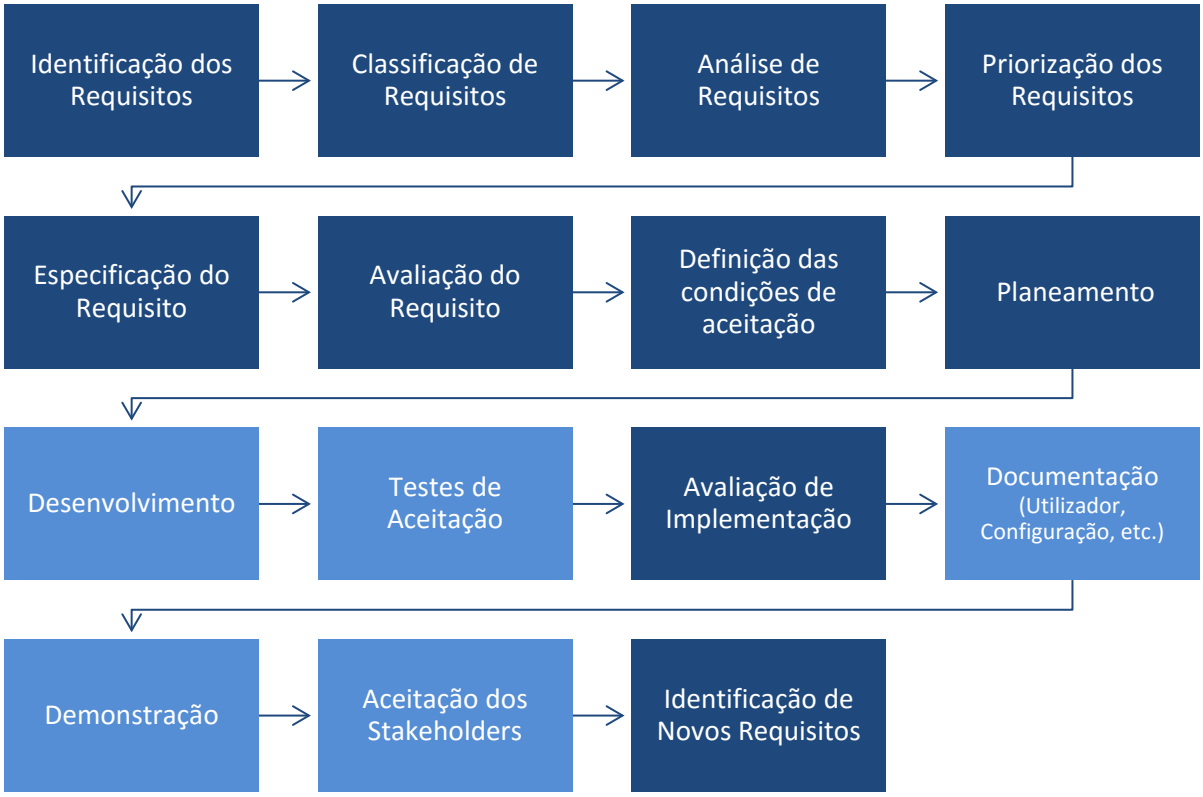


Figura 6 – Fases necessárias para um processo de requisitos.

3.15 Conceptualização do modelo

Depois da definição do modelo procedeu-se à sua materialização. Esta conceptualização está desenvolvida em Excel[11] podendo ser migrada para qualquer outra ferramenta. Possui algumas configurações para partes que são variáveis de empresa para empresa ou de projeto para projeto, tais como a definição do RBS usado para tipificação dos requisitos, a data e a versão da *release e os estados possíveis do progresso (*Figura 7*)*.

Requirement Breakdown Structure (RBS)		Release	Date	estado
Business	Negócio	v0.1	Q1'16	fechado
Administration	Administração	v0.2	Q2'16	em progresso
User	Utilizador	v0.3	Q3'16	reaberto
Action	Ação	v1.0	Q4'16	por fazer
Data	Dados	v1.1	Q1'17	
Business Rules	Regras de Negócio	v1.2	Q2'17	
Interface	Interface	v1.3	Q3'17	
Environment	Ambiente	v1.4	Q4'17	
Quality	Qualidade	v2.0	Q1'18	
Transition	Transição	v2.1	Q2'18	

Figura 7 – Configuração do RBS, datas de release e estados do requisito.

[11] O modelo produzido pode ser descarregado para utilização a partir do seguinte endereço: https://dl.dropboxusercontent.com/u/16647308/tese%20mestrado/RequirementModel.xlsx

Na **identificação dos requisitos** é realizada uma lista de todos requisitos pretendidos pelos *stakeholders*, juntamente com todos os requisitos identificados como défices ou falhas no negócio ou sistema (por exemplo *performance*). Aqui também são identificados potenciais conflitos entre os *stakeholders*, para a sua negociação e consenso.

Na conceptualização (Figura 8) usa-se a folha "gestão" colocando na coluna "nome" algo que identifique inequivocamente o requisito e na coluna "pedido por" a pessoa que o solicitou. Em "*link*" deverão ser colocados ligações para todos os documentos referidos pela pessoa que o pediu. Na coluna "conflito" deverão ser colocados outros requisitos que entrem em conflito. Estes conflitos terão também de ser resolvidos nesta fase para que a análise incida apenas naquilo que é acordado entre as partes que pediram os requisitos que se encontram em conflito. Este ponto deve ser iterativo até existir um lote que cumpra os objetivos estabelecidos.

Requisitos			Rela
Nome	Pedido por	link para documentação (Análise de GAP, maquetes, casos de uso, etc.)	Conflito
Nome do Requisito 1	bruno	www.links.pt	
Nome do Requisito 2	paulo	c:\ISEP\DEI	
Nome do Requisito 3	miguel		ISEP.DEI.4
Nome do Requisito 4	jorge		

Figura 8 – Identificação de requisitos.

Na **classificação de requisitos** é definido o RBS (*Requirement Breakdown Structure*) para o projeto e os requisitos serão tipificados segundo o mesmo (Figura 9).

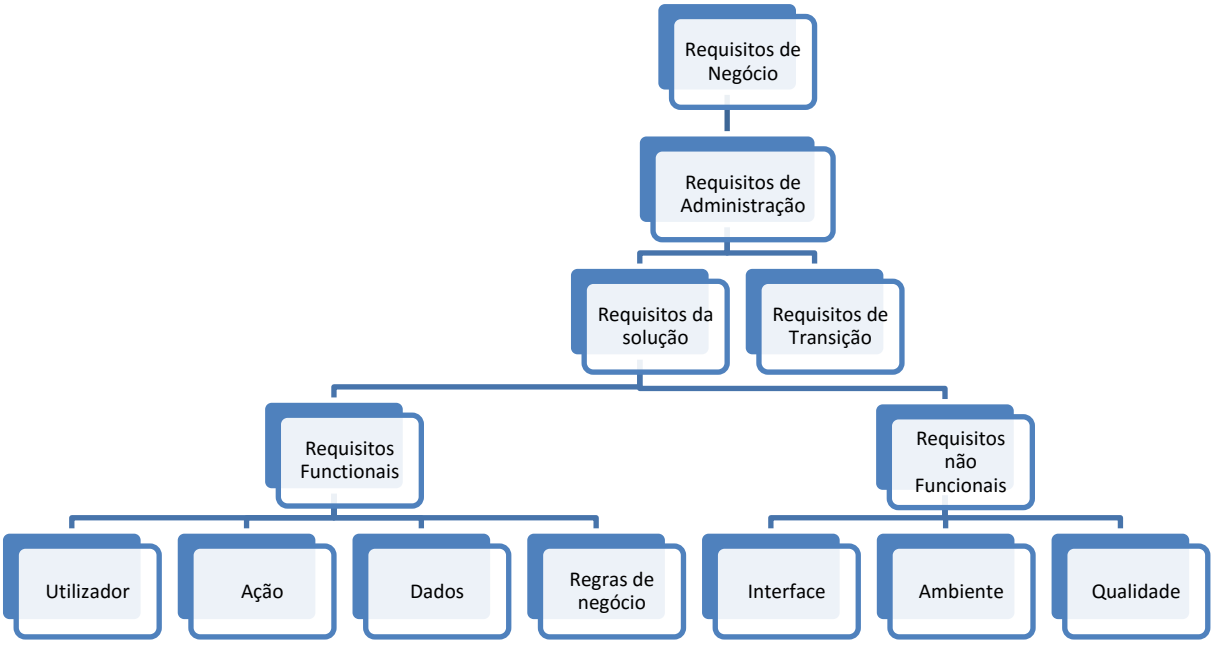


Figura 9 – Exemplo de um RBS (Requirement Breakdown Structure) usado na conceptualização.

Na folha "Gestão" é selecionado na coluna "Tipo" (Figura 10) qual o tipo de requisitos de acordo com o RBS.

Nome	Tipo (RBS)
Nome do Requisito 1	Negócio
Nome do Requisito 2	Administração
Nome do Requisito 3	Utilizador
Nome do Requisito 4	Ação
Nome do Requisito 5	Dados
Nome do Requisito 6	Regras de Negócio
Nome do Requisito 7	Interface
Nome do Requisito 8	Ambiente
Nome do Requisito 9	Qualidade
Nome do Requisito 10	Transição

Figura 10 – Atribuição de tipos aos requisitos segundo o RBS definido.

Na **análise de requisitos** é definido o PBS (*Product Breakdown Structure*) que trata a divisão do projeto por áreas de negócio ou módulos e mapeados os requisitos para cada área. São também indicadas as dependências entre requisitos e identificadas soluções que a aplicação já possua, que possam satisfazer o requisito e qual o défice que é preciso contemplar (Figura 11).

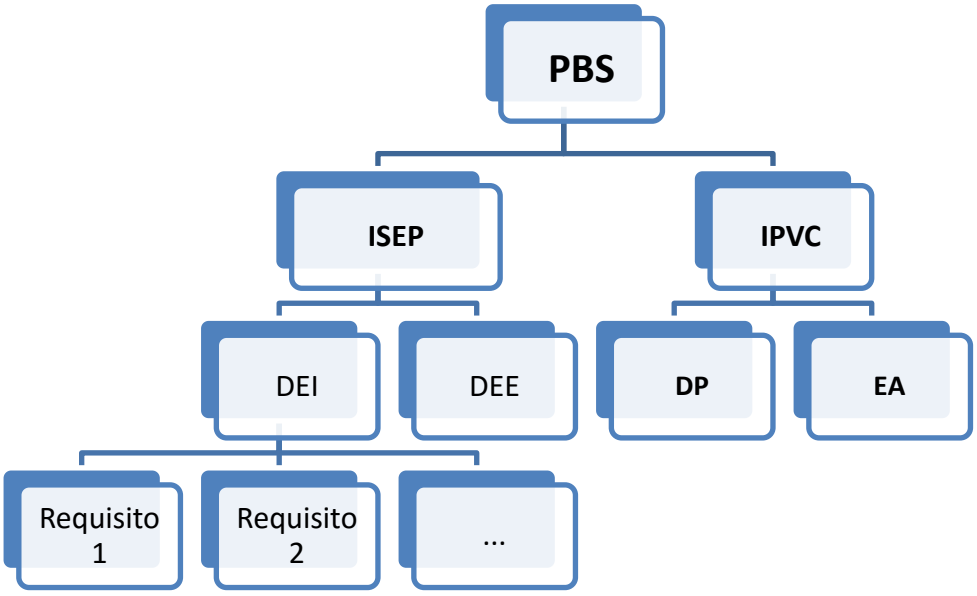


Figura 11 – Exemplo de um PBS (Product Breakdown Structure).

A divisão pelo PBS é atribuída a cada um dos requisitos, que é também usada para calcular um identificador único que servirá de referência, sendo necessário preencher as colunas necessárias para ser possível representar a PBS. Na conceptualização está dividido em "Módulo", "Área" e "Número de Sequência" que geram automaticamente o "ID". São também adicionadas as dependências entre requisitos na coluna "dependência", tipo de dependência na coluna "tipo de dependência" (Figura 12) e documentos de análise efetuados como funcionalidades existentes que podem resolver parcial ou totalmente o solicitado, ou impactos à aplicação existente. Estes documentos deverão ser adicionados na coluna "links" (Figura 8).

Requisitos					Relações	
Módulo	Área	Número de Sequência	ID	Nome	Tipo de dependência	Dependência
ISEP	DEI	1	ISEP.DEI.1	Nome do Requisito 1		
ISEP	DEI	2	ISEP.DEI.2	Nome do Requisito 2		
ISEP	DEI	3	ISEP.DEI.3	Nome do Requisito 3		
ISEP	DEI	3.1	ISEP.DEI.3.1	Nome do Requisito 4	icost	ISEP.DEI.3.3.2
ISEP	DEI	3.2	ISEP.DEI.3.2	Nome do Requisito 5	requires	ISEP.DEI.3.3.2
ISEP	DEI	3.3	ISEP.DEI.3.3	Nome do Requisito 6		

Figura 12 – Atribuição do PBS ao requisito e dependências entre eles.

Na fase de **priorização de requisitos** é avaliada a visibilidade de cada um para cada *stakeholder,* de forma a identificar o seu papel no requisito (responsável por ele, afetado, necessário consultar ou apenas informar) de forma a serem comunicados de forma apropriada, em caso de alteração. Deve também ser indicada qual a prioridade que ele concede ao requisito.

Na concetualização para cada *stakeholder* define-se a visibilidade (RACI) e a prioridade que este concede ao mesmo, segundo uma escala (MoSCoW). Automaticamente e tendo em conta todos os requisitos existentes é calculada a prioridade com que este deve ser desenvolvido (Figura 13).

Stakeholder Management		Priorização						
		Stakeholder A		Stakeholder B				
ID	Nome	Visibilidade	Prioridade	Visibilidade	Prioridade	Prioridade Máx.	Índice Importância	RANKING
ISEP.DEI.1	Nome do Requisito 1	R	M	A	S	4	8	2
ISEP.DEI.2	Nome do Requisito 2	N/A	N/A	N/A	N/A	4	4	4
ISEP.DEI.3	Nome do Requisito 3	N/A	N/A	N/A	N/A	3	5	3
ISEP.DEI.3.1	Nome do Requisito 4	N/A	N/A	N/A	N/A	4	10	1

Figura 13 – Priorização de requisitos e definição da visibilidade do requisito aos stakeholders.

A **especificação de requisitos** dá origem a um conjunto de documentos que serão utilizados pelas equipas para o desenvolvimento, teste, validação com os *stakeholders*, etc. Os documentos gerados dependem do tipo de implementação necessária, podendo ser:

- Análise de negócio;
- Desenvolvimento de *mock-up*;
- Diagramas de fluxo;
- Análise do sistema atual;
- Descrição de cenários de teste;
- Documentos de configuração necessários;
- Etc.

O caminho para os documentos gerados deverá ser colocado na página de "gestão", coluna "*links*" (Figura 8).

As *user stories* devem também ser definidas nesta fase para que o requisito fique claro. Deverão ser preenchidos na folha "*User Story*" (Figura 14).

user story	Condição de aceitação
Como <papel/ator> **eu quero** <ação/comportamento> **de forma a** <benefício>.	**Dado** <pré-condição> **quando** <condição> **então** <resultado>.
Como <papel/ator> **eu quero** <ação/comportamento> **de forma a** <benefício>.	**Dado** <pré-condição> **quando** <condição> **então** <resultado>.
Como <papel/ator> **eu quero** <ação/comportamento> **de forma a** <benefício>.	**Dado** <pré-condição> **quando** <condição> **então** <resultado>.

Figura 14 – User stories e condições de aceitação.

No final da especificação, esta deve sempre ser avaliada de forma a verificar se a especificação:

- Está completa;
- Está consistente;
- É a descrição e não a solução;
- É mensurável;
- É possível rastrear;
- As pré-condições e pós-condições existem.

Para obter esta medição deverá ser respondido ao questionário sobre o requisito na folha "avaliação" (Figura 15).

É também neste momento que o resultado da especificação de requisitos deve ser avaliado, de forma iterativa, com os *stakeholders* até serem aprovados para passarem à fase seguinte. A matriz de visibilidade da folha "Prioridade" deve ser tida em conta para a validação do requisito e para informar todos os intervenientes da forma apropriada.

ID	Nome do Requisito	Completa	Consistente	Descrição (não Solução)	Mensurável	Rastreável	Existem pré e pós condições	% Qualidade
ISEP.DEI.1	Nome do Requisito 1	V	V	V	V	V	-	83%
ISEP.DEI.2	Nome do Requisito 2	V	V	V	V	-	-	67%
ISEP.DEI.3	Nome do Requisito 3	V	V	V	-	-	-	50%
ISEP.DEI.3.1	Nome do Requisito 4	V	V	-	-	-	-	33%
ISEP.DEI.3.2	Nome do Requisito 5	V	-	-	-	-	-	17%

Figura 15 – Avaliação da especificação do requisito.

A **definição das condições de aceitação** é uma fase onde será avaliado o risco da funcionalidade, para que o desenvolvimento a tenha em conta na implementação e são definidas as condições de aceitação para serem testadas pela equipa de qualidade.

As condições de aceitação devem também ser definidas nesta fase para que o requisito fique claro. Deverão ser preenchidos na folha "*User Story*" (Figura 14).

O caminho para os documentos do plano de testes e especificação de testes deverá ser colocado na página de "gestão", coluna "*links*" (Figura 8).

O risco da funcionalidade será calculado respondendo a um questionário segundo OWASP na folha "Cobertura de Testes" (Figura 16). O resultado deste questionário irá calcular a cobertura de testes unitários mínimos para aceitação do requisito. Existem outros tipos de testes que devem ser realizados como, por exemplo, testes funcionais automáticos e manuais.

Identificação		Risco			Testes
ID	Nome	Agente de ameaça envolvido & Factores de Vulnerabilidade	Fatores para determinar um impacto - Fatores Técnicos & Negócio	Risco Total	Percentagem de testes unitários
ISEP.DEI.1	Nome do Requisito 1	RISCO MÉDIO	RISCO ALTO	RISCO ALTO	63,19%
ISEP.DEI.2	Nome do Requisito 2	RISCO BAIXO	RISCO ALTO	RISCO MÉDIO	52,08%
ISEP.DEI.3	Nome do Requisito 3	RISCO MÉDIO	RISCO ALTO	RISCO ALTO	57,64%
ISEP.DEI.3.1	Nome do Requisito 4	RISCO MÉDIO	RISCO ALTO	RISCO ALTO	63,19%
ISEP.DEI.3.2	Nome do Requisito 5	RISCO MÉDIO	RISCO ALTO	RISCO ALTO	63,19%
ISEP.DEI.3.3	Nome do Requisito 6	RISCO MÉDIO	RISCO ALTO	RISCO ALTO	63,19%
ISEP.DEI.3.3.1	Nome do Requisito 7	RISCO MÉDIO	RISCO BAIXO	RISCO BAIXO	44,44%
ISEP.DEI.3.3.2	Nome do Requisito 8	RISCO MÉDIO	RISCO ALTO	RISCO ALTO	68,75%
ISEP.DEI.3.3.3	Nome do Requisito 9	RISCO ALTO	RISCO ALTO	RISCO CRÍTICO	98,61%

Figura 16 – Cálculo de risco de um requisito e cobertura de testes unitários mínima aceitável.

Após a definição das condições de aceitação é realizado o **planeamento** para ser usado no planeamento da *release,* onde são apresentados os requisitos às equipas de forma a avaliar a sua complexidade (em equipas ágeis normalmente através de *story points,* em equipas não ágeis uma escala de complexidade), tempo necessário e qual a *release* em que está planeada a sua entrega. Esta complexidade normalmente é medida através das medidas dificuldade, desconhecimento e esforço necessário.

Esta informação é preenchida na folha "Planeamento", onde é indicada a equipa que se compromete com o planeamento. É também indicada a complexidade de implementação, o tempo para a sua execução, este último usado para verificar o que pode ficar dentro de cada *release* (Figura 17).

Release						
ID	Nome	Equipa	Story Points	Tempo (hora)	Tempo dia	Release
ISEP.DEI.1	Nome do Requisito 1	produto	50	3	0,43	Q1'16
ISEP.DEI.2	Nome do Requisito 2	produto	40	4	0,57	Q2'16
ISEP.DEI.3	Nome do Requisito 3	produto	20	2	0,29	Q3'16
ISEP.DEI.3.1	Nome do Requisito 4	produto	10	3	0,43	Q4'16

Figura 17 – Planeamento dos requisitos.

Ao longo destas etapas, a folha de gestão vai sendo automaticamente preenchida com informação de gestão para que o gestor de requisitos possua uma visão centralizada (Figura 18) e possa tomar ações com antecedência. Por exemplo, quando é detetado que determinado requisito produz dependências para outros, mesmo que este seja menos prioritário deverá ser realizado antes, por uma questão de dependências com outros desenvolvimentos.

Relações		Gestão					
Dependência	Dependentes?	Release	Qualidade do Requisito	Testes Unitários	Prioridade	Estado	Entrega
		Q1'16	83%	63,19%	2	reaberto	
		Q2'16	67%	52,08%	4	em progresso	
		Q3'16	50%	57,64%	3	fechado	01/01/16
ISEP.DEI.3.3.2		Q4'16	33%	63,19%	1	fechado	20/02/16
ISEP.DEI.3.3.2		0	17%	63,19%	-	por fazer	
		0	100%	63,19%	-	por fazer	
		0	83%	44,44%	-	por fazer	
	Y	0	67%	68,75%	-	fechado	01/01/16

Figura 18 – Visão de gestão de requisitos preenchida de forma automática.

Após o **desenvolvimento** e o **teste** é necessário realizar a **avaliação de desenvolvimento**. Esta fase é a fase de verificação onde é confirmado se as regras e os documentos de especificação são respeitados para poderem ser entregues ao *stakeholder*. Se não estiver de acordo, volta para a implementação. Caso esteja tudo conforme foi pedido, é registado na coluna de entrega a data de aceitação (Figura 18) e realizada toda a **documentação** funcional e de configuração, utilizando os recursos tradicionais como Microsoft Word ou baseado em tecnologias *web* como o Confluence (Atlassian Pty Ltd, 2015).

Na **demonstração** é apresentada a funcionalidade aos *stakeholders* para aprovação e identificados possíveis desvios para correção. Em processos ágeis, aqui são identificados **novos requisitos**.

Ao longo de todo o processo deve existir um identificador do estado do requisito, de forma a poder acompanhar o seu desenvolvimento (Figura 18).

4 Avaliação da Solução

A melhor forma de avaliar esta solução é através de métricas definidas em conjunto com um inquérito, realizado às pessoas que definem requisitos, *business analysts*, pessoas que os irão consumir (arquitetos, equipa de desenvolvimento e *testers*) e através dos resultados obtidos do desenvolvimento com a apresentação aos *stakeholders*.

4.1 Métricas de Avaliação

As métricas definidas para avaliação são:

- O tempo consumido para a finalização do requisito;
- Documentação existente;
- Qualidade do código;

É possível utilizar o método AHP (*Analytic hierarchy process*) para se avaliar a solução. O grande objetivo deste trabalho é melhorar a qualidade de desenvolvimento de *software* e reduzir tempos, pelo que os critérios definidos serão os resultados de tempo consumido, a qualidade do código obtido, a satisfação da funcionalidade pelos *stakeholders* e as respostas às diferentes questões realizadas no questionário.

As respostas obtidas serão quantificadas e a sua média calculada, para depois juntamente com os pesos calculados, poderem ser comparadas e avaliadas (antes e depois da implementação do processo).

De seguida, de forma a calcular os pesos, foram definidos os níveis de importância de comparações binárias (Saaty, 1991), onde 1 significa igual importância, 3 fraca importância, 5 forte importância, 7 muito forte importância e 9 importância absoluta. Assim sendo, tem-se na Tabela 5:

Tabela 5 – Matriz para cálculo das importâncias para avaliação da solução.

	Tempo consumido	Qualidade do código	Satisfação dos *stakeholders*	Qualidade das análises de negócio	Documentação existente	Análise de segurança	Gestão de requisitos
Tempo consumido	1	1	1	3	5	7	5
Qualidade do código	1	1	1	3	5	7	5
Satisfação dos *stakeholders*	1	1	1	3	5	7	5
Qualidade das análises de negócio e requisitos	1/3	1/3	1/3	1	3	5	3
Documentação existente	1/5	1/5	1/5	1/3	1	3	1
Análise de segurança	1/7	1/7	1/7	1/5	1/3	1	1
Gestão de requisitos	1/5	1/5	1/5	1/3	1	1	1

Normalizando a soma das colunas para 1:

Tabela 6 – Matriz normalizada com a soma das colunas a 1.

Tempo consumido	8/31	8/31	8/31	8/29	15/61	7/31	5/21
Qualidade do código	8/31	8/31	8/31	8/29	15/61	7/31	5/21
Satisfação dos *stakeholders*	8/31	8/31	8/31	8/29	15/61	7/31	5/21
Qualidade das análises de negócio e requisitos	8/93	8/93	8/93	8/87	9/61	5/31	1/7
Documentação existente	5/97	5/97	5/97	3/98	3/61	3/31	1/21
Processo de priorização	1/27	1/27	1/27	1/54	1/61	1/31	1/21
Gestão de requisitos	5/97	5/97	5/97	3/98	3/61	1/31	1/21

E calculando a média das linhas para se obter o peso do critério:

Tabela 7 – Pesos dos critérios.

Tempo consumido	179/712	=	0,251
Qualidade do código	179/712	=	0,251
Satisfação dos *stakeholders*	179/712	=	0,251
Qualidade das análises de negócio e dos requisitos	108/943	=	0,115
Documentação existente	15/277	=	0,054
Processo de priorização	13/404	=	0,032
Gestão de requisitos	43/957	=	0,045

Após estes passos, são comparados os cenários da utilização do modelo e processo proposto com um modelo e processo já existente na empresa (modelos usados para determinado documento ou mesmo no caso de nada ser usado). Para verificar a consistência dos julgamentos realizados nesta avaliação, quando comparados com enormes quantidades de julgamentos aleatórios, é calculado o rácio de consistência:

Primeiro é calculado o λmax:

Tabela 8 – Cálculo do λmax.

	Tempo consumido	Qualidade do código	Satisfação *stakeholders*	Qualidade das análises	Documentação existente	Análise de segurança	Gestão de requisitos	Matriz de Pesos	Valores de Somatório multiplicado pelo peso do vetor prioridade	Coluna anterior dividida pelo vetor prioridade
Tempo consumido	1	1	1	3	5	7	5	179/712	1,8184638	7,2332191
Qualidade do código	1	1	1	3	5	7	5	179/712	1,8184638	7,2332191
Satisfação dos *stakeholders*	1	1	1	3	5	7	5	179/712	1,8184638	7,2332191
Qualidade das análises	1/3	1/3	1/3	1	3	5	3	108/943	0,8240748	7,1953939
Documentação existente	1/5	1/5	1/5	1/3	1	3	1	15/277	0,3846371	7,1029651

	Tempo consumido	Qualidade do código	Satisfação *stakeholders*	Qualidade das análises	Documentação existente	Análise de segurança	Gestão de requisitos	Matriz de Pesos	Valores de Somatório multiplicado pelo peso do vetor prioridade	Coluna anterior dividida pelo vetor prioridade
Análise de segurança	1/7	1/7	1/7	1/5	1/3	1	1	13/404	0,2258112	7,0175173
Gestão de requisitos	1/5	1/5	1/5	1/3	1	1	1	43/957	0,3202806	7,1281054
λmax (média)										7,163377017

Para através do mesmo calcular o índice de consistência:

CI = (λmax – n) / (n-1)

CI = (7,163377017 – 7) / (7-1) = 0,027

Da tabela de *Saaty* obtém-se o índice de consistência aleatório:

RI(7) = 1,32

Para finalmente calcular o rácio de consistência

CR = CI/RI = 0,027 / 1,32 = 0,02

Pode-se verificar que o rácio de consistência é de 0,02 que é um valor inferior a 0,1. Uma inconsistência inferior a 10% implica que o ajustamento é reduzido, em comparação com os valores reais dos *eigenvector*, pelo que é um teste consistente.

Na secção seguinte aborda-se a forma como foram obtidos os valores para a realização do cálculo.

4.2 Forma de Obter Dados para Avaliação

4.2.1 Questionário

O questionário é já realizado de forma comparativa entre os dois modelos, sendo possíveis quatro respostas, “melhorou muito” que equivale a um ponto, “melhorou” que equivale a meio ponto, “é igual” que equivale a zero pontos e “piorou” que equivale a 1 ponto negativo. Sobre estes valores é realizada uma média pelo número de respostas para quantificação dos resultados, onde um valor negativo significa que piorou e um valor positivo que melhorou.

As perguntas são diferentes conforme o papel da pessoa, tendo como respostas possíveis “melhorou muito”, “melhorou”, “é igual” ou “piorou”, sendo cada uma das perguntas catalogada num ponto de avaliação:

- Qualidade das análises de negócio e dos requisitos;
- Processo de priorização;
- Gestão de requisitos;
- Satisfação da solução perante *stakeholders*.

Ao **gestor de requisitos** foram realizadas as seguintes perguntas, onde as três primeiras encaixam no processo de priorização e as quatro últimas na gestão de requisitos:

1 – A gestão de conflitos (evidenciação e negociação com os *stakeholders*) deste novo processo;

2 – A forma como os requisitos são priorizados;

3 – A identificação de dependências entre requisitos;

4 – A rastreabilidade dos requisitos;

5 – A avaliação da qualidade dos requisitos – retrospetiva do trabalho;

6 – A gestão dos vários intervenientes no processo;

Ao **analista** foram realizadas as seguintes perguntas que encaixam na avaliação da qualidade da análise:

7 – O detalhe fornecido nos requisitos;

8 – O acesso à informação sobre determinado requisito;

9 – A identificação de dependências que condicionam a solução;

Para o **gestor de *releases*** foram feitas questões da avaliação de gestão de requisitos:

10 – A facilidade em estimar o âmbito na entrega da *release*;

11 – A facilidade em realizar o planeamento para a *release*;

12 – O compromisso e garantia de entregas na *release*;

Para os ***stakeholders*** foram realizadas perguntas sobre a sua satisfação relativa ao processo, sendo apenas a primeira pergunta acerca processo de priorização. A última questão já tem em conta o número de recusas antes e depois do processo que foi analisado pelos *stakeholders*:

13 – A perceção de dependências entre requisitos e entre *stakeholders*;

14 – A clareza da informação sobre o que realmente vai ser desenvolvido;

15 – A perceção e aceitação dos compromissos perante todos os intervenientes;

16 – A apresentação e antecipação do resultado final (pré-visualização do que será realizado);

17 – O resultado obtido na entrega quanto às expetativas na entrega;

18 – A envolvência dos *stakeholders* no processo e na decisão do desenvolvimento;

19 – A aceitação após implementação;

A equipa de **desenvolvimento** recebeu perguntas sobre a qualidade da análise:

20 – A clareza de informação de requisitos (o que é necessário fazer, como e porquê);

21 – A independência perante outros intervenientes no processo dependentes da comunicação (foi necessário realizar menos esclarecimentos para desenvolvimento);

22 – A perceção da dependência entre requisitos para perceber a ordem necessária para os desenvolvimentos;

23 – A facilidade em que são realizadas as estimativas de execução quer a nível de complexidade quer a nível de tempo;

24 – A clareza na cobertura de testes unitários necessários;

Na equipa de **testes,** e à semelhança da equipa de desenvolvimento, as perguntas foram sobre a qualidade da análise:

25 – A informação de como realizar os cenários de teste;

26 – A independência perante outros intervenientes no processo dependentes da comunicação (foi necessário realizar menos esclarecimentos para desenvolvimento);

27 – A identificação das condições de aceitação necessárias para teste;

4.2.1 Qualidade do Código

A análise de segurança é realizada no desenho do requisito, onde é realizada o cálculo da cobertura de testes automáticos a obter que depois é verificada com a cobertura de testes unitários existente. Neste ponto fez-se uma avaliação de retrospetiva antes da existência deste processo, onde se calculou a quantidade de testes unitários necessária e a existente, assim como se executou o SonarQube (Sonarsource SA, 2016) de forma a verificar os riscos e qualidade do código.

A qualidade do código foi certificada através da ferramenta sonar, onde foi possível verificar a cobertura de testes unitários sobre o código.

Outro elemento que pesou nesta avaliação foi o número de *bugs* (erros) abertos na ferramenta de gestão de projetos utilizada.

4.2.2 Documentação Existente

A documentação existente no final de uma funcionalidade exemplo foi comparada. Existem diversos tipos de documentação, tendo sido avaliadas as seguintes documentações:

- Documentação de requisitos;
- Documentação de testes;
- Documentação de arquitetura;
- Documentação técnica;
- Documentação de configuração;
- Documentação funcional;
- Ligação entre os diversos documentos.

4.2.3 Tempos de Desenvolvimento

O tempo consumido para o desenvolvimento foi medido através de uma ferramenta de gestão de projetos, onde foram medidos os tempos necessários para a realização da funcionalidade. Foram também calculados os tempos de levantamento de requisitos, testes e correção de erros.

4.3 Resultados Obtidos

A metodologia e processo foram testados em duas empresas, uma de pequena dimensão e outra de dimensão média.

4.3.1 Questionário

O resultado do questionário pode ser visualizado na Tabela 9 e Tabela 10 onde é apresentado o número da pergunta (Questionário), a que avaliação corresponde, a quem foi direcionada a pergunta, o número de pessoas que respondeu à pergunta, o número de seleção das respostas e o resultado final.

Tabela 9 – Resultados do questionário da empresa de pequena dimensão.

Pergunta	Avaliação	Público-Alvo	Número respostas	Melhorou muito	Melhorou	É igual	Piorou	Resultado (0 a 1)
1	Processo de Priorização	Gestor de Requisitos	2	2				1
2	Processo de Priorização	Gestor de Requisitos	2	2				1
3	Processo de Priorização	Gestor de Requisitos	2	1	1			0,875
4	Gestão de Requisitos	Gestor de Requisitos	2	1	1			0,875
5	Gestão de Requisitos	Gestor de Requisitos	2	2				1
6	Gestão de Requisitos	Gestor de Requisitos	2	1	1			0,875
7	Qualidade Análise	Analista	4	4				1
8	Qualidade Análise	Analista	4	1	3			0,8125
9	Qualidade Análise	Analista	4		3	1		0,6875
10	Gestão de Requisitos	Gestor da *Release*	1	1				1
11	Gestão de Requisitos	Gestor da *Release*	1	1				1
12	Gestão de Requisitos	Gestor da *Release*	1	1				1
13	Processo de Priorização	*Stakeholders*	4	4				1
14	Satisfação dos *stakeholders*	*Stakeholders*	4	3	1			0,9375
15	Satisfação dos *stakeholders*	*Stakeholders*	4	4				1
16	Satisfação dos *stakeholders*	*Stakeholders*	4	3	1			0,9375
17	Satisfação dos *stakeholders*	*Stakeholders*	4	4				1
18	Satisfação dos *stakeholders*	*Stakeholders*	4	4				1
19	Satisfação dos *stakeholders*	*Stakeholders*	4	4				1
20	Qualidade Análise	Programador	12	4	8			0,8333
21	Qualidade Análise	Programador	12	2	10			0,7917
22	Qualidade Análise	Programador	12	5	7			0,8542
23	Qualidade Análise	Programador	12	4	8			0,8333
24	Qualidade Análise	Programador	12	12				1
25	Qualidade Análise	*Tester*	3	2	1			0,9167
26	Qualidade Análise	*Tester*	3	2	1			0,9167
27	Qualidade Análise	*Tester*	3	3				1

Tabela 10 – Resultados do questionário da empresa de média dimensão.

Pergunta	Avaliação	Público-Alvo	Número respostas	Melhorou muito	Melhorou	É igual	Piorou	Resultado (0 a 1)
1	Processo de Priorização	Gestor de Requisitos	3	2	1			0,9167
2	Processo de Priorização	Gestor de Requisitos	3	3				1
3	Processo de Priorização	Gestor de Requisitos	3	2	1			0,9167
4	Gestão de Requisitos	Gestor de Requisitos	3	2	1			0,9167
5	Gestão de Requisitos	Gestor de Requisitos	3	1	2			0,8333
6	Gestão de Requisitos	Gestor de Requisitos	3	1	1	1		0,75
7	Qualidade Análise	Analista	7	7				1
8	Qualidade Análise	Analista	7	3	4			0,8571
9	Qualidade Análise	Analista	7		4	3		0,6428
10	Gestão de Requisitos	Gestor da *Release*	1	1				1
11	Gestão de Requisitos	Gestor da *Release*	1	1				1
12	Gestão de Requisitos	Gestor da *Release*	1	1				1
13	Processo de Priorização	*Stakeholders*	6	3	1	2		0,7917
14	Satisfação dos *stakeholders*	*Stakeholders*	6	5	1			0,9583
15	Satisfação dos *stakeholders*	*Stakeholders*	6	5	1			0,9583
16	Satisfação dos *stakeholders*	*Stakeholders*	6	4	2			0,9167
17	Satisfação dos *stakeholders*	*Stakeholders*	6	5	1			0,9583
18	Satisfação dos *stakeholders*	*Stakeholders*	6	4	1	1		0,875
19	Satisfação dos *stakeholders*	*Stakeholders*	6	6				1
20	Qualidade Análise	Programador	17	13	4			0,9412
21	Qualidade Análise	Programador	17	5	12			0,8235
22	Qualidade Análise	Programador	17	5	7	5		0,75
23	Qualidade Análise	Programador	17	13	4			0,9412
24	Qualidade Análise	Programador	17	15	2			0,9706
25	Qualidade Análise	*Tester*	12	8	4			0,9167
26	Qualidade Análise	*Tester*	12	9	3			0,9375
27	Qualidade Análise	*Tester*	12	10	2			0,9583

Na Tabela 11 apresenta-se o resultado do questionário por área de avaliação. Os valores encontram-se entre 0 e 1, onde um valor abaixo de 0,5 significa que piorou e acima que melhorou.

Tabela 11 – Melhoria do processo com base no questionário.

Grupo de Avaliação	Empresa A antes do processo	Empresa A depois do processo	Empresa B antes do processo	Empresa B depois do processo
Gestão de Requisitos	0,5	0,958333	0,5	0,916667
Processo de Priorização	0,5	0,96875	0,5	0,90625
Qualidade Análise	0,5	0,876894	0,5	0,885361
Satisfação *stakeholders*	0,5	0,979167	0,5	0,944444

4.3.2 Qualidade do Código

Quanto à qualidade do Código a empresa A tinha colocado como regra uma cobertura de testes unitários na ordem dos 50%, enquanto a empresa B de 80%. Na Tabela 12 é possível visualizar a percentagem de testes necessária para cada desenvolvimento baseado no risco de determinado desenvolvimento calculado através do modelo. Nas colunas referentes à cobertura de testes instituída na empresa é possível verificar que existem várias que não atingiam um mínimo de aceitação (a vermelho), assim como o gasto em desenvolvimento de testes unitários em funcionalidades com menos importância.

Tabela 12 – Percentagem de testes unitários.

Requisito	Empresa A	Empresa B	Percentagem de testes necessária
Desenvolvimento A	50%	80%	63,19%
Desenvolvimento B	50%	80%	52,08%
Desenvolvimento C	50%	80%	57,64%
Desenvolvimento D	50%	80%	63,19%
Desenvolvimento E	50%	80%	44,44%
Desenvolvimento F	50%	80%	13,89%
Desenvolvimento G	50%	80%	44,44%
Desenvolvimento H	50%	80%	68,75%
Desenvolvimento I	50%	80%	98,61%
Desenvolvimento J	50%	80%	63,19%
Média	**50%**	**80%**	**56,95%**

Legenda: A vermelho os desenvolvimentos que não possuem cobertura de testes suficiente.

Com esta medição verifica-se que se poderá gastar menos tempo, em testes unitários menos prioritários e assegurar aqueles críticos que têm uma maior cobertura, permitindo assim poupar recursos e obter maior qualidade como se pode verificar na última linha.

Fazendo o cálculo dos objetivos atingidos pelas empresas, no que respeita à cobertura de testes unitários, é possível verificar que nenhuma das empresas possui a cobertura de testes adequada ao requisito (Tabela 13).

Tabela 13 – Percentagem de testes unitários adequados aos requisitos de segurança.

Empresa A	Empresa B	Respeitando a percentagem calculada
30%	90%	100%

O SonarQube (Sonarsource SA, 2016) pode também ser utilizado para ver a cobertura total de testes unitários sobre o código da aplicação (Figura 19), no entanto, não é orientado aos fatores críticos mas permite verificar se uma alteração ao código tem implicações nos testes existentes.

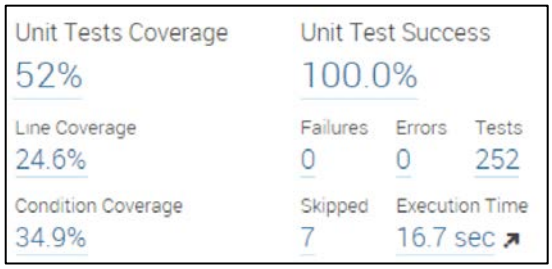


Figura 19 – Cobertura de testes unitários obtido pelo SonarQube de forma.

Quanto à qualidade do código, onde as métricas devem ser instituídas pela empresa, verificou-se através do SonarQube (Sonarsource SA, 2016) que estas não seguem as políticas, uma vez que foram detetados erros e código duplicado (Figura 20).

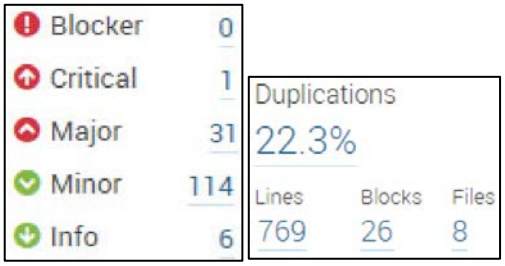


Figura 20 – Resultados encontrados na empresa B através da utilização do SonarQube.

Outro ponto pertinente na avaliação da qualidade, é o número de erros que foram encontrados na fase de testes, sendo que foi usada uma ferramenta de gestão de projeto para os monitorizar. A Tabela 14 representa a medição de erros antes, e depois, da implementação do processo e pode-se concluir que este número baixou.

Tabela 14 – Erros encontrados pela equipa de testes ou stakeholders após desenvolvimento.

	Empresa A antes	Empresa A depois	Empresa B antes	Empresa B depois
Quantidade de erros encontrados	30	5	20	4

Na Tabela 15 pode-se visualizar a quantidade de erros encontrados no desenvolvimento para as empresas A e B, que não seriam visíveis se não fosse utilizado o SonarQube (SonarQube, 2015).

Tabela 15 – Erros encontrados no código através do SonarQube.

Erros	Empresa A	Empresa B	Respeitando a política da empresa
Bloqueantes	0	0	0
Críticos	2	1	0
Altos	42	31	5
Menores	175	114	N/A
Informativos	23	6	N/A

A redução no tempo necessário para testes unitários é refletida na avaliação dos tempos de desenvolvimento em conjunto com os testes. No entanto, encontrar estes erros no final consome

tempo de outros intervenientes ao longo do processo e não apenas da equipa de desenvolvimento, pelo que devem ser imediatamente mitigados no desenvolvimento, o que irá aumentar o seu tempo.

4.3.3 Documentação Existente

A documentação existente foi avaliada tendo em conta antes do processo e depois do processo. Com a existência da documentação de requisitos, toda a outra documentação fica facilmente completa, uma vez que existe um ponto de partida e um entendimento comum. Os valores presentes na Tabela 16 foram calculados tendo por base uma ferramenta de gestão de projeto que monitorizou o tempo usado na documentação e a altura em que foram realizadas. Com o uso do modelo pode-se verificar que a documentação fica praticamente pronta à primeira, sendo necessárias alterações pontuais.

Tabela 16 – Eficiência da realização da documentação.

	Empresa A						Empresa B					
	Antes			Depois			Antes			Depois		
Documentação	Tempo (h)	Tempo alteração	Eficiência	Tempo (h)	Tempo alteração	Eficiência	Tempo (h)	Tempo alteração	Eficiência	Tempo (h)	Tempo alteração	Eficiência
Requisitos	5	0	100,00%	13	0	100,00%	4	0	100,00%	12	0	100,00%
Testes	3	4	42,86%	5	0	100,00%	4	2	66,67%	6	0	100,00%
Arquitetura	6	0	100,00%	5	0	100,00%	6	0	100,00%	5	0	100,00%
Técnica	3	0	100,00%	2	0	100,00%	3	0	100,00%	2	0	100,00%
Configuração	3	2	60,00%	3,8	0,2	95,00%	3	2	60,00%	3,8	0,2	95,00%
Funcional	4	3	57,14%	5,5	0,5	91,67%	4	3	57,14%	5,5	0,5	91,67%
Ligações entre documentação	0	1	0,00%	1	0	100,00%	0	0	0,00%	1	0	100,00%
Eficiência Total			**63,27%**			**98,10%**			**66,67%**			**98,10%**

4.3.4 Tempos de Desenvolvimento

De forma a avaliar os tempos de desenvolvimento, utilizou-se de igual forma uma ferramenta de gestão de projeto para monitorizar os tempos onde os resultados encontram-se na Tabela 17. Aqui é possível verificar que não houve qualquer ganho na empresa B, em termos de tempo necessário para implementação.

Tabela 17 – Tempo consumido antes e após implementação do processo.

	Empresa A			Empresa B		
Tempo Consumido	Antes (h)	Depois (h)	Diferença (h)	Antes (h)	Depois (h)	Diferença (h)
Análise de requisitos	5	13	-8	4	12	-8
Desenvolvimento	35	41	-6	30	36	-6
Correção de erros	14	5	9	10	4	6
Testes	8	3	5	6	2	4
Documentação	3,4	3,2	0,2	3,1	2,9	0,2
Total	**65,4**	**65,2**	**0,2**	**53,1**	**56,9**	**-3,8**
Percentagem			0,3%			-7,16%

De forma a atribuir-se uma normalização para ser utilizado no método AHP, considerou-se que o melhor resultado inicial acima identificado equivale a 1 e o pior a 0,5 dando origem aos resultados na Tabela 18, através da aplicação da proporcionalidade inversa.

Tabela 18 – Normalização dos dados a utilizar no AHP.

	Empresa A		Empresa B	
Tempo Consumido	**Antes (h)**	**Depois (h)**	**Antes (h)**	**Depois (h)**
Total	**65,4**	**65,2**	**53,1**	**56,9**
Valor a usar no método AHP	0,5	0,50813	1	0,8455285

4.3.5 Resultados Finais

Os resultados finais da avaliação das duas empresas podem ser consultados na Tabela 19, onde a qualidade do código tem apenas em conta os testes unitários, uma vez que os restantes são relacionados com uma ferramenta de validação de código e não propriamente com o modelo e processo apresentado.

Tabela 19 – Resultados finais da implementação do modelo e processo.

	AHP	Empresa A		Empresa B	
Grupo de Avaliação	Peso calculado	Antes	Depois	Antes	Depois
Gestão de Requisitos	0,045	0,5	0,958333	0,5	0,916667
Processo de Priorização	0,032	0,5	0,96875	0,5	0,90625
Qualidade Análise	0,115	0,5	0,876894	0,5	0,885361
Satisfação dos *Stakeholders*	0,251	0,5	0,979167	0,5	0,944444
Documentação Existente	0,054	0,6327	0,981	0,6667	0,981
Qualidade do Código	0,251	0,30	1	0,90	1
Tempo Consumido	0,251	0,5	0,50813	1	0,845529
Total		**0,4564658**	**0,852253342**	**0, 7344018**	**0, 925323753**

É possível verificar que a aplicação deste modelo e processo trazem claros benefícios às empresas de desenvolvimento de *software*. Embora o tempo necessário para o desenvolvimento seja mais elevado com este processo, os ganhos a nível de qualidade, documentação e aceitação final são substanciais.

5 Conclusões

5.1 Objetivos Realizados

O processo proposto tem por base a área de análise de negócio e engenharia de requisitos, produzindo um modelo que aborda estas duas grandes áreas e estabelece uma nova forma de abordar ambas, de forma unificada. O processo está direcionado a gestores de requisitos e analistas de negócio ou empresas que pretendam aumentar a sua eficiência no desenvolvimento de *software*.

Todo o ciclo de vida de requisitos está contemplado no processo e modelo, desde a identificação dos requisitos, a sua negociação com os diferentes *stakeholders* e análise de requisitos que poderá dar origem a novos processos de negociação, pressupondo uma análise de impacto do sistema atual. A análise de requisitos permite também evidenciar as dependências entre requisitos, de forma a se poder priorizar estes, eliminando os riscos entre o desenvolvimento das várias partes. A partir desta priorização, os requisitos são especificados e avaliados para serem aprovados junto dos *stakeholders*. Após a sua aprovação são definidas as condições de aceitação dos mesmos e avaliado o planeamento da *release* ou o tempo esperado para terminar o desenvolvimento. No final do processo é avaliado se o desenvolvimento será aceite. Todo este processo permite uma eficaz gestão de requisitos e gestão de mudança.

O modelo inclui diversos pontos-chave para a eficiente gestão e especificação de requisitos, tais como a definição do PBS (*Product Breakdown Structure*), a gestão dos diferentes requisitos com dados como a prioridade, o tempo esperado de entrega, o estado dos requisitos, a data de conclusão, a sua qualidade e a cobertura de testes unitários. Esta prioridade é negociada com os vários *stakeholders*. Cada requisito possui uma *user story* com a indicação do que este deverá ser (à exceção dos requisitos não funcionais), condições de aceitação e o seu tipo. O modelo calcula também a cobertura de testes unitários necessários para obter uma confiança mínima no requisito e centraliza as ligações para a localização de informação relacionada.

Este modelo foi materializado numa forma de folha de cálculo, no entanto, poderia facilmente ser transposto para qualquer outra ferramenta, de acordo com as ferramentas utilizadas pela empresa como, por exemplo, o Jira (Atlassian Pty Ltd, 2015) e Confluence (Atlassian Pty Ltd, 2015).

O uso de um correto e completo processo de requisitos com a utilização de *mock-ups* por ecrã, permitindo ao programador ver qual o resultado final esperado; a indicação das pré-condições do desenvolvimento, que permitem saber exatamente o que é necessário para o desenvolvimento; as pós-condições do desenvolvimento, que permitem saber o final esperado; os pressupostos do desenvolvimento, que indicam que são assumidas algumas condições pelas quais não é necessário qualquer desenvolvimento; o diagrama de fluxo e interação utilizador-máquina com o comportamento da aplicação, explicando passo-a-passo o que o requisito/funcionalidade deve realizar; a captura de *user stories* com a perceção da necessidade e do benefício a obter (podem existir várias para o mesmo diagrama de fluxo ou *mock-up*); os documentos de análise, que permitem ver qual o ponto em que é necessário realizar a alteração ou nova implementação conforme seja um *GAP* de implementação existente ou nova; todos estes elementos permitem aumentar consideravelmente a interpretação do requisito e a qualidade de implementação, bem como reduzir a ambiguidade e a incerteza.

A utilização destes elementos permite também uma comunicação transversal de alguns dos documentos entre todos os intervenientes do processo, desde analistas, programadores, arquitetos de *software*, *stakeholders*, comerciais, gestores, etc.

Como foi confirmado pela avaliação do processo e do modelo, quando usados pelas pessoas com conhecimentos de gestão de requisitos e análise de negócio, foi possível melhorar a qualidade do *software*, contribuindo para a redução de ambiguidade entre os diferentes intervenientes no processo de desenvolvimento. Permitiu, também, definir os passos do processo em pequenos blocos que possuem sempre uma entrega (resultado tangível), assim como definir as fronteiras entre os mesmos. Garantiu-se que, no final do processo, existia documentação atualizada e coerente, ao contrário do que se verificava nos processos anteriores.

Por outro lado, a qualidade do código produzido aumentou, uma vez que permite de forma acertada realizar os testes ao que realmente importa, aumentando a cobertura de testes a pontos críticos da aplicação (identificado com maior cobertura de testes) e diminuindo aos pontos menos relevantes (identificado com menor cobertura de testes). Nas empresas onde foi avaliado o processo desenhado, tinham uma estratégia de cobertura de testes fixa para qualquer implementação, independentemente do risco. Verificou-se também que o número de erros encontrados após o desenvolvimento diminuiu, o que fez com que existissem menos correções, traduzindo-se num menor tempo de desenvolvimento e teste.

5.2 Limitações e Trabalho Futuro

Este modelo trata os processos de análise de negócio e engenharia de requisitos. A conceptualização precisa de refinamento na matriz de dependências. Este modelo encontra-se num nível mais conceptual, tentando abstrair de qualquer tecnologia, pelo que foi usado o *Excel* por ser uma ferramenta bastante difundida entre analistas. Uma grande melhoria seria a sua implementação numa ferramenta própria, integrada com ferramentas de gestão de projeto, documentação e avaliação de código, usando as mais-valias de cada uma.

Este trabalho podia também ser especializado às diferentes metodologias, como por exemplo SCRUM ou *Waterfall*. No entanto, para esta dissertação o pretendido era ser genérico e independente de qualquer metodologia, fazendo apenas referência em alguns pontos para simplificar a perceção.

Como continuidade deste trabalho deveriam ser abordadas as técnicas de negociação de requisitos, possíveis modelos de documentos para as diferentes fases, de forma a orientar a informação importante a ser depositada nos mesmos ou a utilização e demonstração de ferramentas possíveis para documentação. Todos estes novos passos deverão sempre ter em conta a rastreabilidade dos mesmos, para rapidamente serem localizados.

A extensibilidade deste modelo para outros modelos e áreas de negócio que não apenas o desenvolvimento de *software,* poderia dar uma nova visibilidade à forma como se veem as tarefas rotineiras no dia-à-dia das diferentes áreas. Com este processo vê-se aplicabilidade a áreas como a gestão industrial, construção civil, marketing, etc. A sua abstração a estes novos modelos e áreas poderiam trazer um resultado muito interessante, mudando o paradigma de como as pessoas realizam os seus processos.

A avaliação e as condições de aceitação do processo abordam os testes unitários, no entanto, sendo estes internos e controlados como se de uma caixa negra se tratasse, não é possível a sua verificação

na integração. Para colmatar esta lacuna seria necessário realizar outros tipos de testes, tais como testes funcionais automáticos ou manuais. Para atingir este ponto, seria também necessário incluir no processo a especificação de testes, no entanto, estes tocam mais na área de qualidade do que na análise de negócio ou na especificação de requisitos. Esta dissertação apenas refere as condições de aceitação do desenvolvimento, condições que devem estar ligadas a um processo de qualidade e indicar quais os cenários de teste a realizar sobre as mesmas, o que se podia traduzir num incremento de qualidade, dependendo do processo de qualidade da empresa. Outros elementos, como os diagramas de fluxo, poderiam ser utilizados para mais facilmente desenhar os cenários de teste.

À medida que esta dissertação foi realizada, muitos pontos de extensibilidade foram surgindo, pelo que esta pode ser usada como uma base para muitas melhorias, para muitos pontos diferentes de investigação e encadeamento com as mais diversas áreas. Espera-se que a mesma possa ser tomada como fonte de inspiração para outros trabalhos, uma vez que a forma de abordar a área de engenharia de requisitos e análise de negócio, nesta dissertação, poderá levar a novos pensamentos e suscitar novas questões a estudar.

A.Ferramentas de Especificação de Requisitos

1. Modelio Requirements

Após a avaliação da ferramenta Modelio Requirements (Modelio Requirements, 2015) segundo as boas práticas de requisitos, e a demonstração do seu potencial através de um *webinar* sobre a ferramenta, foram identificados os seguintes pontos fortes:

- Ligações tipificadas entre requisitos de forma a permitir extrair dependências entre elas (Derivação, parte de, refinamento, etc.). É possível visualizar na Figura 21 um diagrama exemplo e na Figura 22 a matriz de dependência entre requisitos, gerada pela aplicação sendo possível alterar dos dois lados;

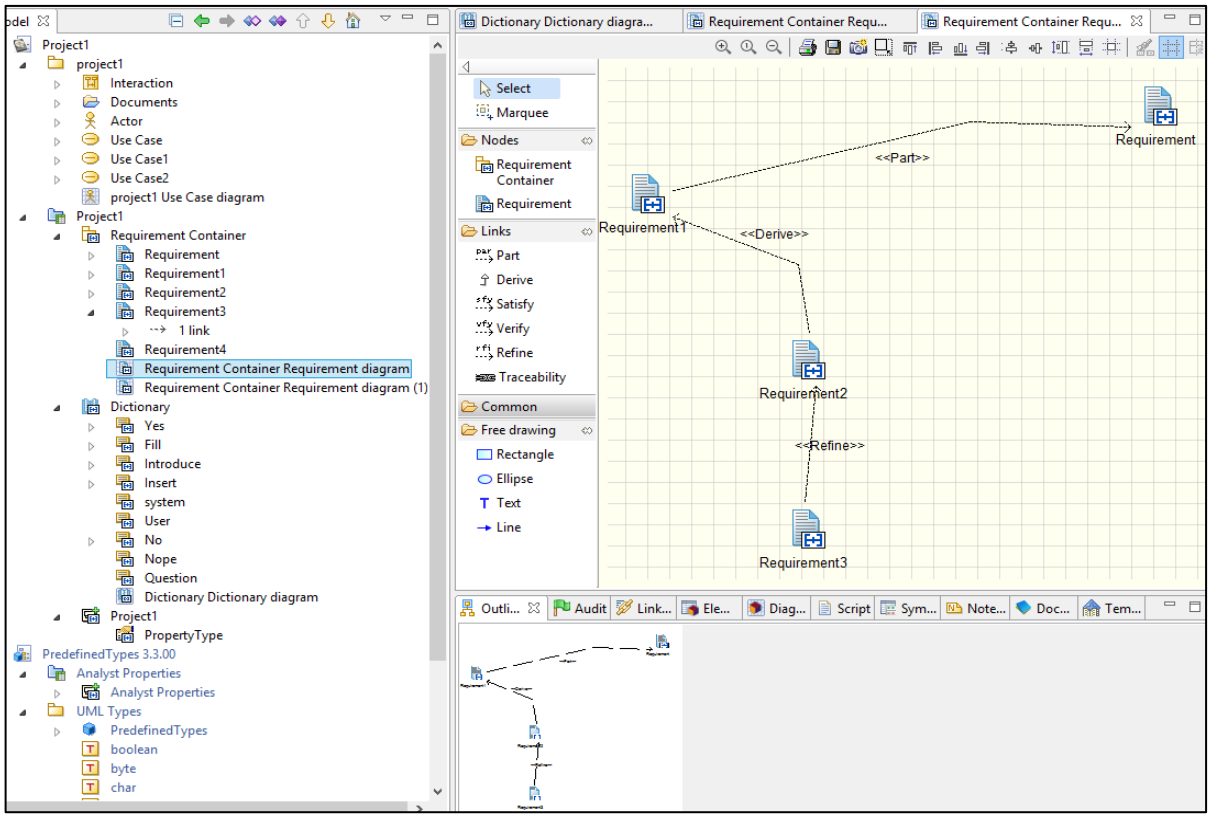


Figura 21 – Ligação tipificada entre requisitos no Modelio.

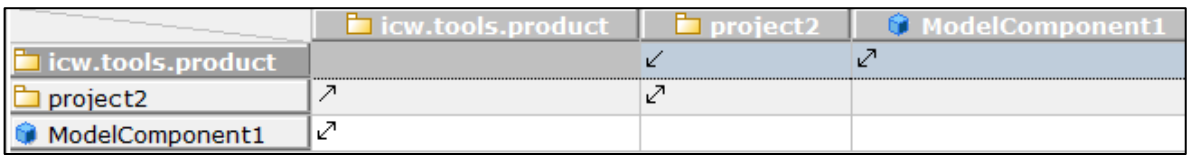


Figura 22 – Matriz de dependência entre requisitos no Modelio.

- Ligações entre os diferentes diagramas de modelação e requisitos de forma a obter ligação com as diferentes fases de modelação (rastreabilidade), demonstrado na Figura 23;

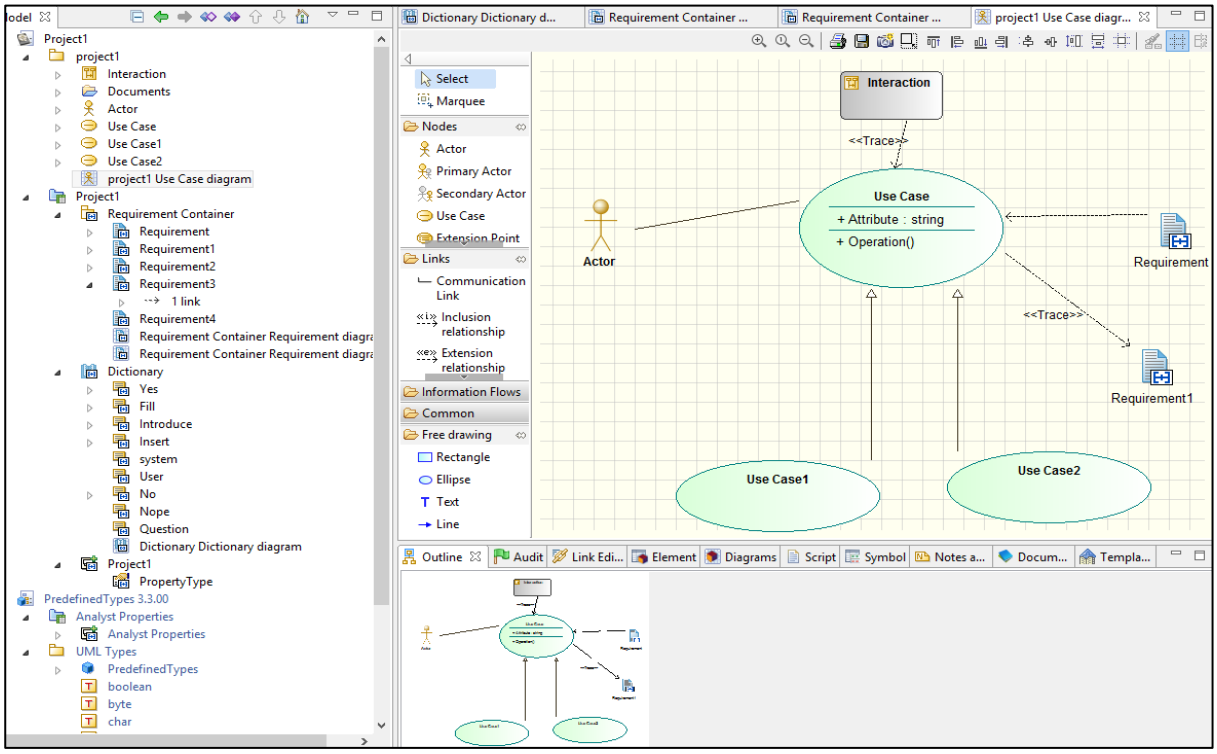


Figura 23 – Ligação entre requisitos, casos de uso e outros diagramas UML no Modelio.

- Dicionário de termos e conceitos de negócio para reduzir a ambiguidade de termos utilizados entre os diferentes intervenientes e associação dos termos através de regras e tipificações como: antónimo, contexto, sinónimo, relacionado, etc. Possibilidade de ver os termos em forma de diagrama (Figura 24) ou diretamente numa tabela exportável para *Excel* (Figura 25);

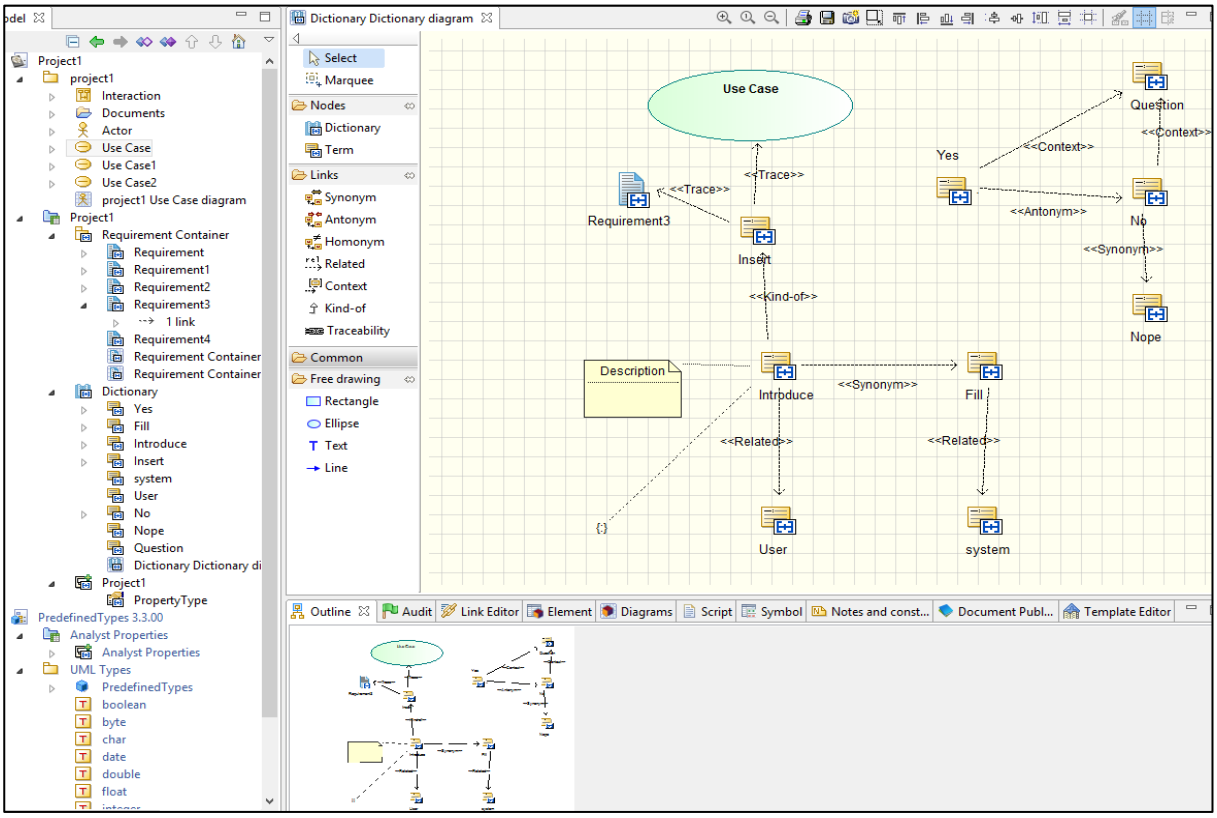


Figura 24 – Dicionário de termos e relação com outros termos no Modelio.

	Id	Definition	Identifier	Documentation	Source	Reference	Acceptance
0	Fill			<Click to create>			Local
1	Introduce			<Click to create>			Local
2	Insert			<Click to create>			Local
3	system			<Click to create>			Local
4	User			<Click to create>			Local
5	No			<Click to create>			Local
6	Nope			<Click to create>			Local
7	Question			<Click to create>			Local
8	Yes	Business person that makes		<Click to create>			Local

Figura 25 – Tabela de termos no Modelio.

- Versionamento dos requisitos e organização de requisitos em pastas, sendo possível organizar por área e/ou tipo (Figura 26):

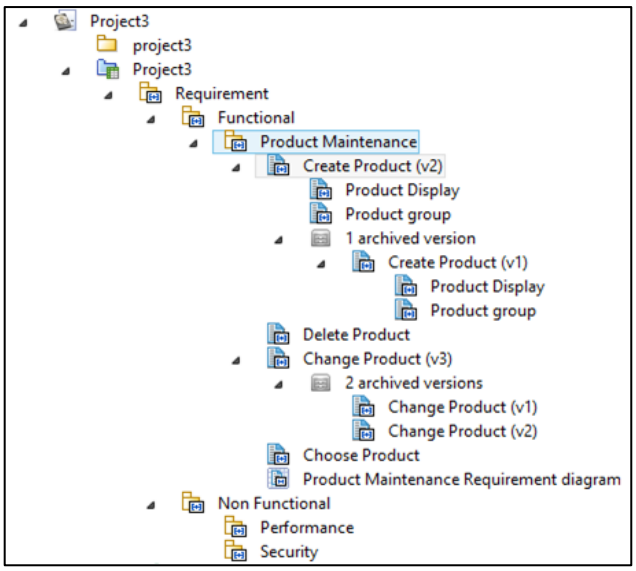


Figura 26 – Organização dos Requisitos em grupos e versionamento no Modelio.

- Possibilidade de organizar os requisitos hierarquicamente (Figura 27):

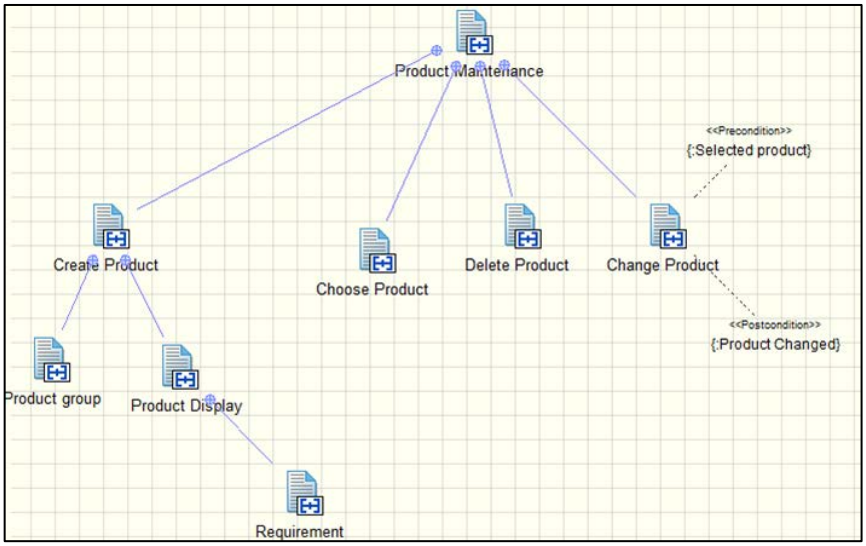


Figura 27 – Hierarquia de requisitos no Modelio.

- Geração automática de diagramas de dependência entre casos de uso e artefactos (possibilidade de ver o que é usado e o que o usa num só diagrama) – Figura 28.

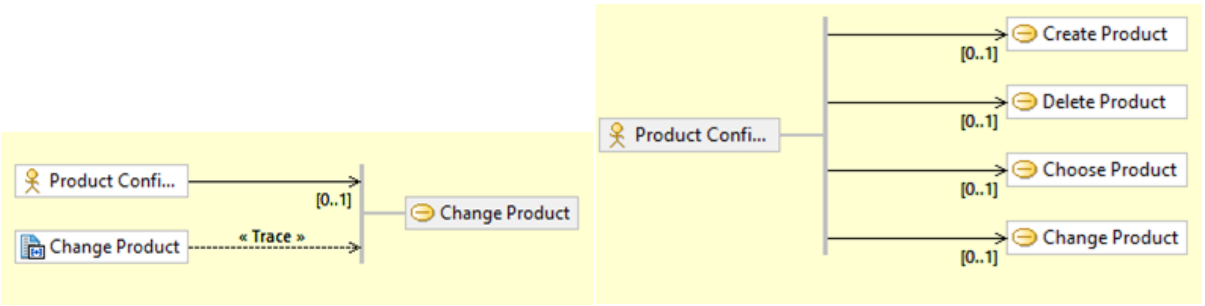


Figura 28 – Análise de impacto, utilização, dependências entre artefactos no Modelio.

- Possibilidade de anexar documentos ou imagens, como por exemplo um *mock-up* ou documentação dos *stakeholders* (*pdf, word*, etc.);
- Customização dos campos usados nos requisitos;
- Integração com ferramentas de documentação baseados na *web* (Por exemplo o *Confluence* (Atlassian Pty Ltd, 2015));
- Possibilidade de criar *templates* para os manuais (documentação) a exportar, onde se indica a informação a introduzir e o tipo (*web, word, pdf*, etc.);
- Exportação para manuais de utilizador (*pdf, excel* e *word*) e utilização dos mesmos bidireccionalmente (ao alterar no manual, altera automaticamente no *modelio*);
- Matriz de dependência entre requisitos;
- Geração automática de análise de impacto sobre alterações;
- A parte de modelação de modelo de domínio é gratuita e usa formatos estandardizados como *UML, BPMN, SysML, TOGAF* e *XMI*;
- Usa formatos estandardizados de requisitos como *CMMI, TOGAF, SysML, DoDAF, MODAF, NAF, Agile* com *product backlog;*
- Permissões para as pessoas das diferentes fases de desenvolvimento (requisitos, design, análise, gestão, desenvolvimento, etc.).

2. Yakindu Requirements

Yakindu Requirements (Itemis, 2015) é uma ferramenta mais vocacionada para a especificação de requisitos e não para modelação como a *Modelio Requirements* (Modelio Requirements, 2015), no entanto, possui diversos pontos fortes:

- Muito simples de usar sendo um *plugin* do eclipse (Eclipse, 2015), ferramenta muito usada para programação;
- Possui uma divisão por projeto separando os requisitos da especificação (Figura 29);

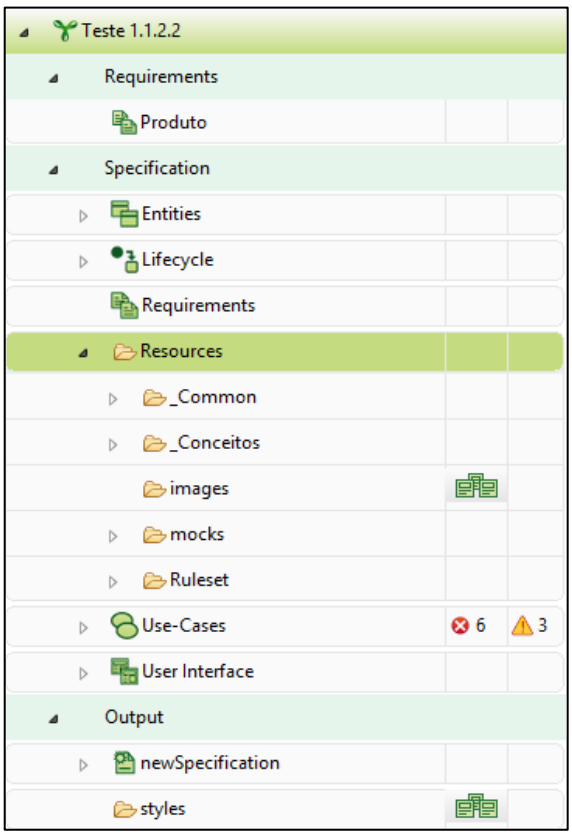


Figura 29 – Estrutura de um projeto Yakindu.

- É possível adicionar diversos requisitos que se podem relacionar entre eles e hierarquizar. Podem ser sincronizados de forma automática com uma ferramenta de gestão de projeto, como o *Jira* (Atlassian Pty Ltd, 2015) e possuem tipos personalizáveis (Qualidade, Funcional, entre outros) – Figura 30;

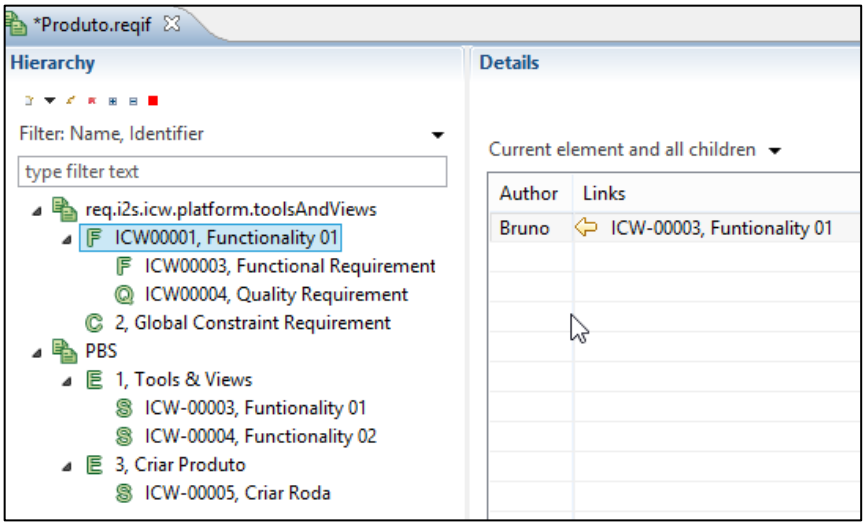


Figura 30 – Estrutura de requisitos e ligações com restante especificações no Yakindu.

- Possui também uma janela onde se definem as caraterísticas do requisito, o seu estado, última alteração, entre outros possíveis (Figura 31) usando a norma *reqif*;

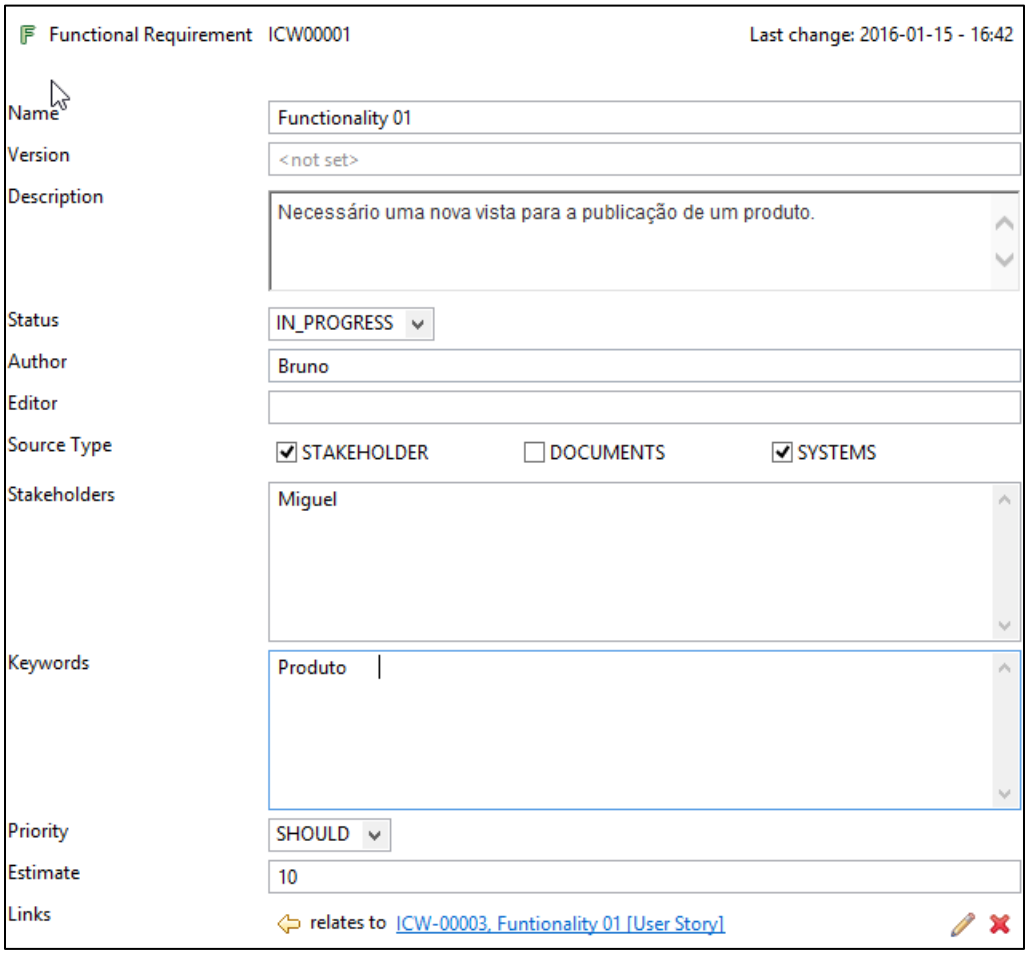


Figura 31 – Ecrã onde se pode especificar os requisitos no Yakindu.

- Permite controlo de versões através de um repositório git;
- A edição dos diagramas é através de texto, o que se torna extremamente produtivo realizando também as ligações para outros diagramas, ecrãs e requisitos (Figura 32);

```
Actors Customer as person “description of the customer”
                   System as system “system”
                   Organization as organization “organization”
                   Employee as person extends Customer “description”


        usecase UC001 “Register user”
                pages errorDialog
                goals “goal”
                entities Address, Bruno
                actors Customer , System
                requirements ICW00003 “Produto#_xtPNgfJcEeSn7vJMiwwKBA”,
                             ICW00004 “Produto#_yi5qMfJcEeSn7vJMiwwKBA”
                preconditions “precondition 2”
                labels “login”
                custom attributes
                 comment “”
                 priority high
                 source “Text”
```

```
            status New
            date 31-07-2015
        severity “3-major”
        author “Bruno”
    end attributes
        basic flow
            step 1 SP “Carregar língua do sistema” by System
            step 2 SP “Mostrar ecrã na língua selecionada” by System
             alternatives
                    if “login” then INR “Login” invoke usecase UC002 finalize flow
                                with postcondition “utilizador autenticado”
                    if “recuperar password” then INO “Recuperar Password”
                                invoke usecase UC003 finalize flow
                                with postcondition “Password enviada”
                    if “escolher língua” then RC “Escolher Língua” continue with step 2
            else if “cancelar” then
        end flow with postcondition “utilizador cancelou”
    end usecase

    usecase UC002 “Login user”
            actors Customer
    end usecase

    usecase UC003 “Reset password”
            actors Customer
            pages errorDialog(DefaultScreen)
            basic flow
                    step 1 “label” on DefaultScreen
            end flow with postcondition “postcondition”
    end usecase
```

Figura 32 – Texto utilizado para gerar os diagramas no Yakindu.

O texto indicado gera um diagrama de package (Figura 33), casos de uso (Figura 34), diagrama de interação entre casos de uso (Figura 35) e diagrama de fluxo do caso de uso (Figura 36);

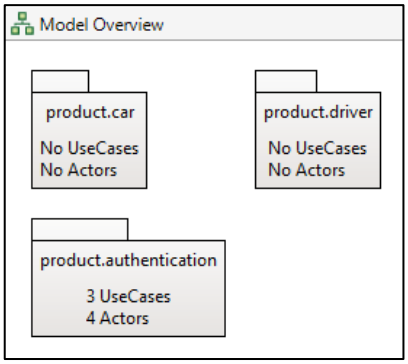


Figura 33 – Diagrama de package no Yakindu.

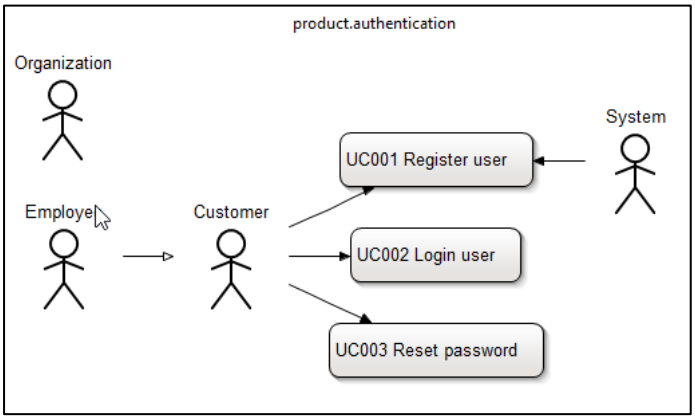


Figura 34 – Diagrama de casos de uso no Yakindu.

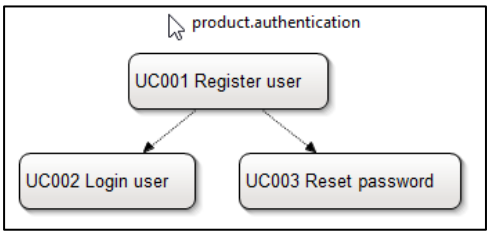


Figura 35 – Diagrama de interação no Yakindu.

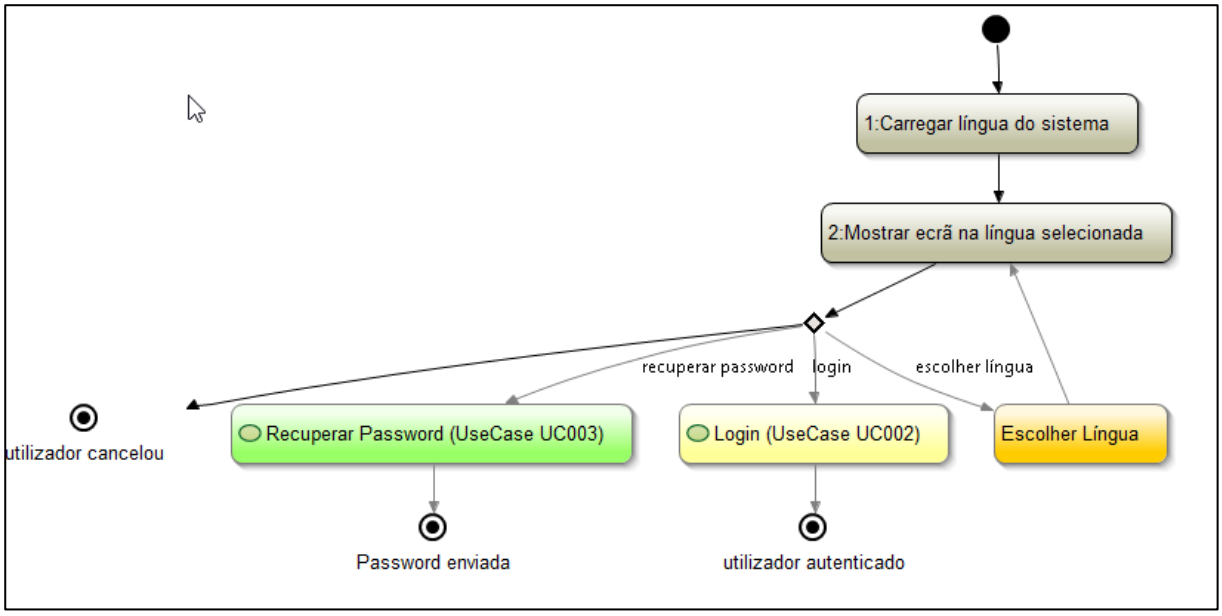


Figura 36 – Diagrama de fluxo do caso de uso login no Yakindu.

- Existem também outros diagramas complementares como diagrama de entidade (Figura 37), diagrama de ciclo de vida (Figura 38) e diagrama de interação de ecrã, onde é possível adicionar mock-ups, documentação, etc.

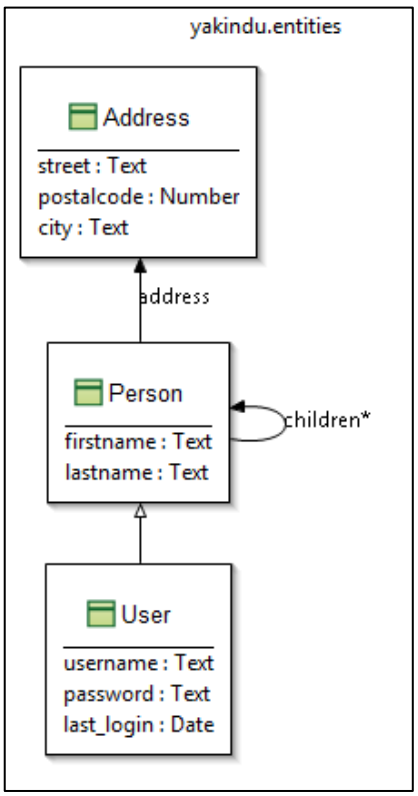


Figura 37 – Diagrama de entidades no Yakindu.

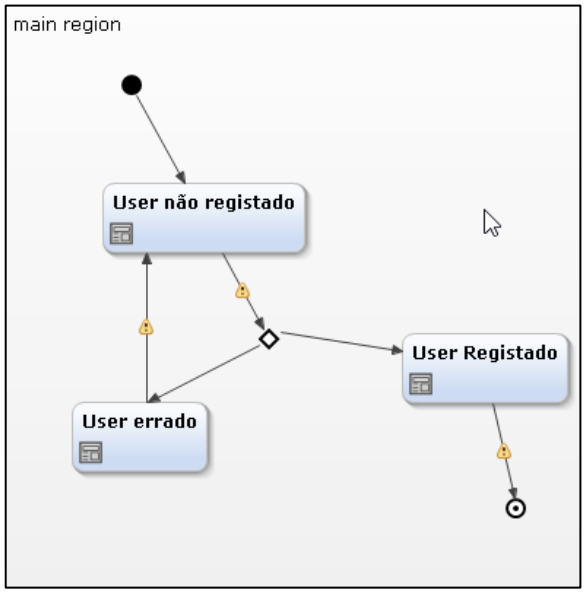


Figura 38 – Diagrama de ciclo de vida no Yakindu.

- Outra das vantagens do Yakindu é a geração de relatórios e manuais de utilizador em formato *pdf* e *html,* assim como geração de estatísticas de esforço necessário para desenvolver a funcionalidade especificada (Figura 39);

Use case	Steps	Comlexity	Actors	Entities	Pages	Grade	Rating	Estimation			PERT
								Min	Regular	Max	Average
uc.login											
Login	1	1	2	0	0	7	low	0	2	0	1,333333333
Registar	2	3	2	0	0	16	medium	1	0	0	0,166666667
Recuperar Password	19	26	2	0	0	125	really high	0	0	5	0,833333333

Figura 39 – Estimativa de esforço com base na especificação no Yakindu.

3. Visual Paradigm

O Exceler do Visual paradigm é uma ferramenta que surgiu muito recentemente (final de Janeiro de 2016) e que possui um processo de gestão de requisitos já muito completo, tentando responder às necessidades oriundas da análise de requisitos:

- Usa a premissa "**As a** <user role> **I want to** <action/behaviour> **so that** <benefit>" para identificação dos requisites (Figura 40):

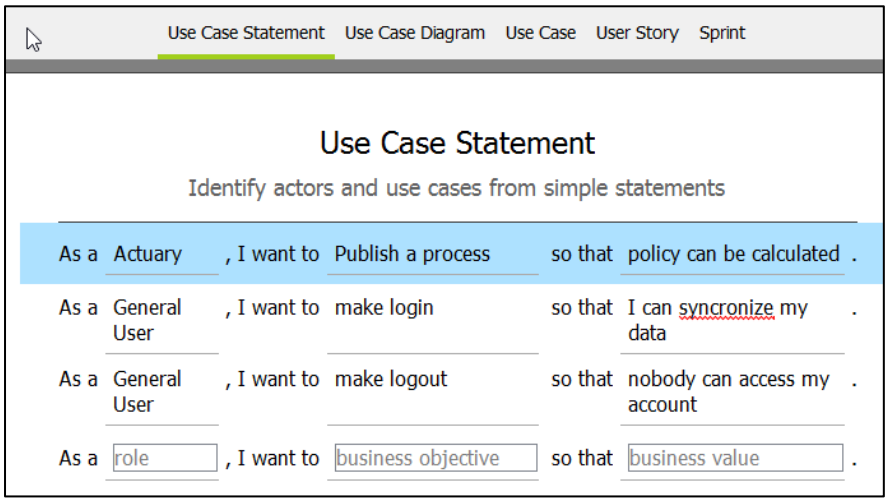


Figura 40 – Identificação de requisitos no Visual Paradigm.

- A partir desta identificação gera automaticamente os atores e ligação com o pretendido. Permite também aqui realizar alterações diretamente, apenas influenciando o caso anterior na alteração (Figura 41):

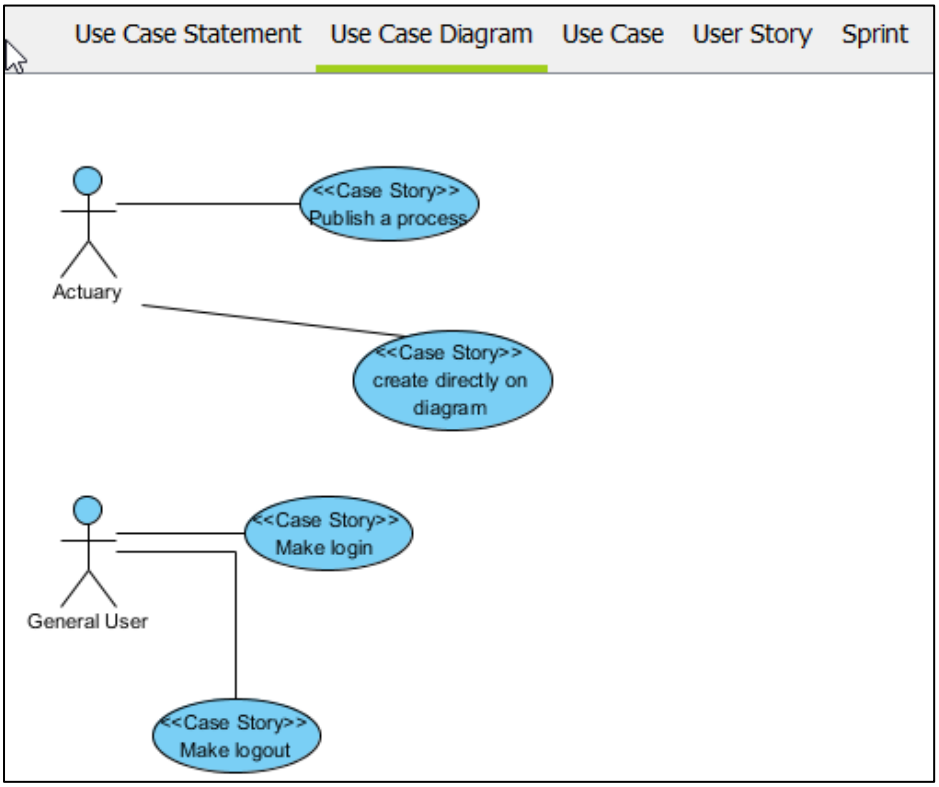


Figura 41 – Diagrama de casos de uso no Visual Paradigm.

A ferramenta permite analisar cada um dos requisitos preenchendo alguma informação de forma automatizada possuindo a possibilidade de priorizar, colocar o tipo (Figura 42), o autor, pressupostos, pré-condições e pós-condições (que podem ser uma ligação direta com um usecase), a complexidade, o seu estado em desenvolvimento (Figura 43).

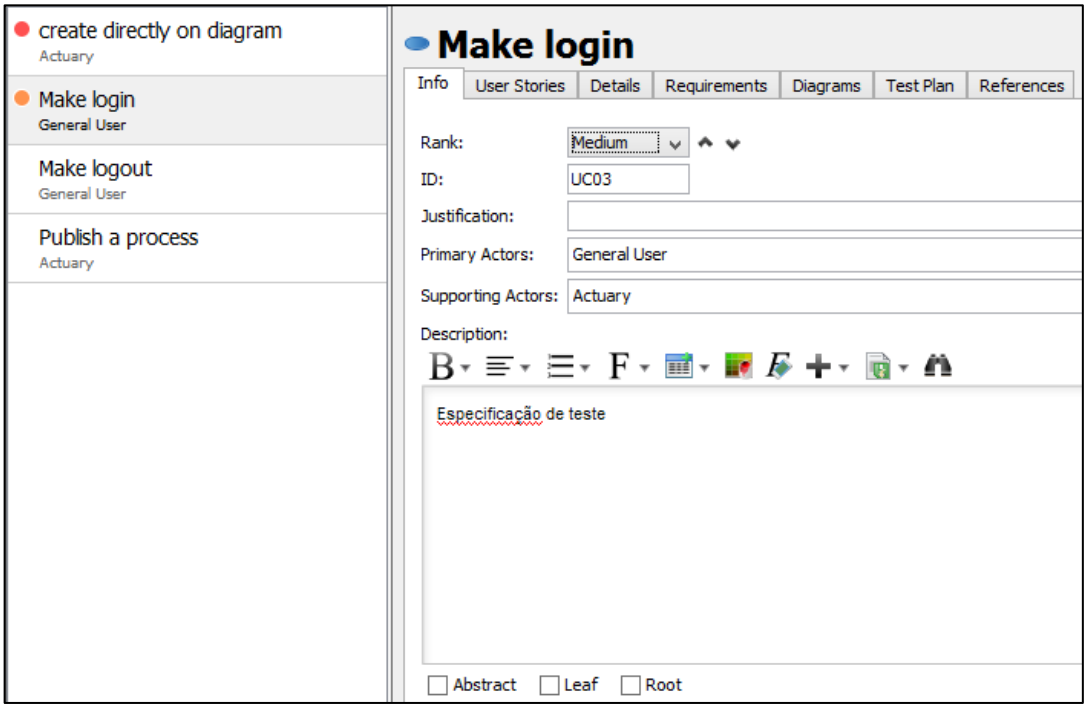


Figura 42 – Informação genérica da análise do requisito no Visual Paradigm.

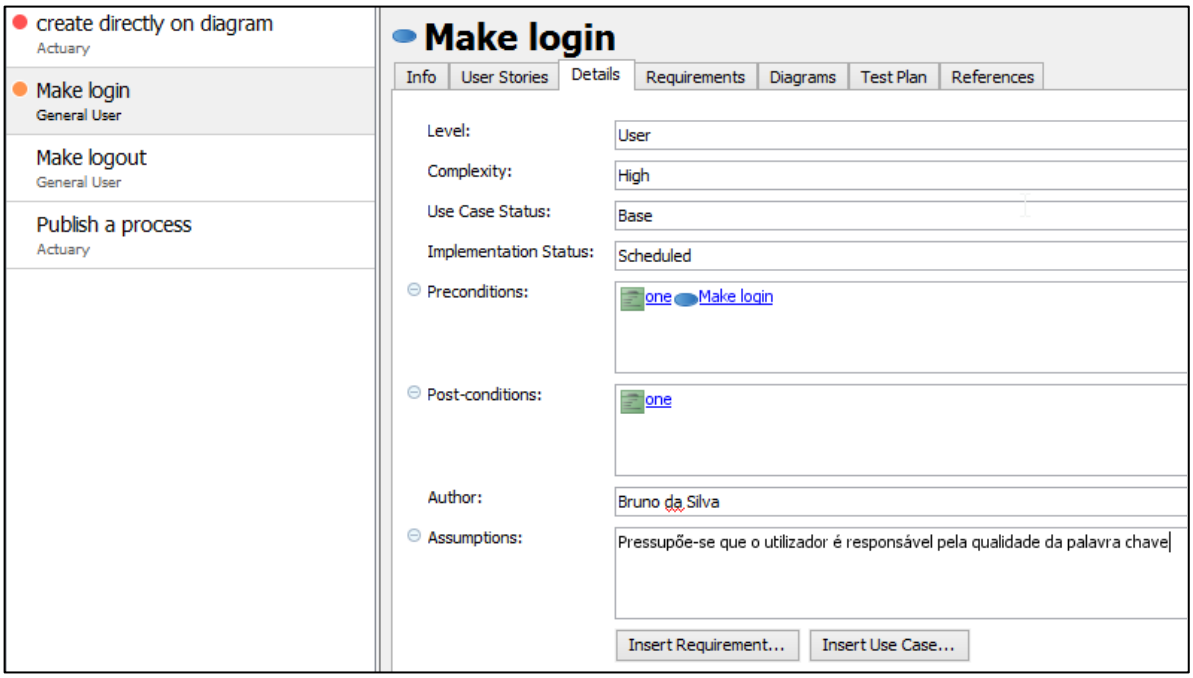


Figura 43 – Detalhes do requisito no Visual Paradigm.

- Permite também realizar a relação com outros requisitos, sendo as pré e pós-condições criadas automaticamente (Figura 44):

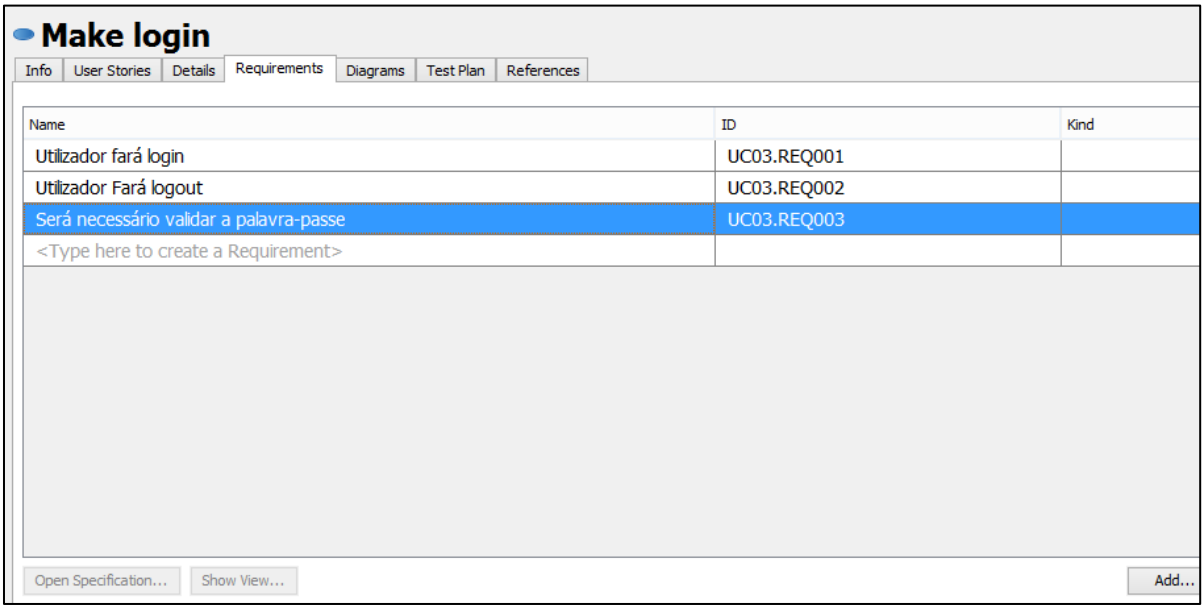


Figura 44 – Relação com outros requisitos no Visual Paradigm.

Gera o diagrama de fluxo, sendo possível alterar o mesmo (Figura 45), ligar diagramas existentes ou adicionar novos diagramas de requisitos, de estado, casos de uso, atividade, sequência ou uma análise (Figura 46).

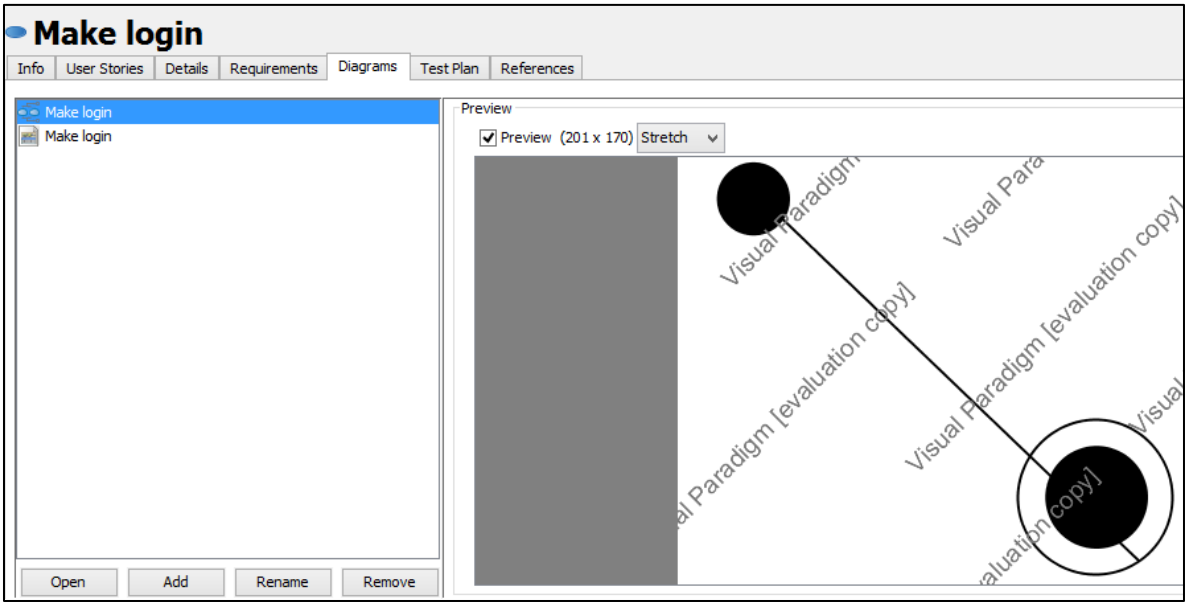


Figura 45 – Diagrama de fluxo no Visual Paradigm.

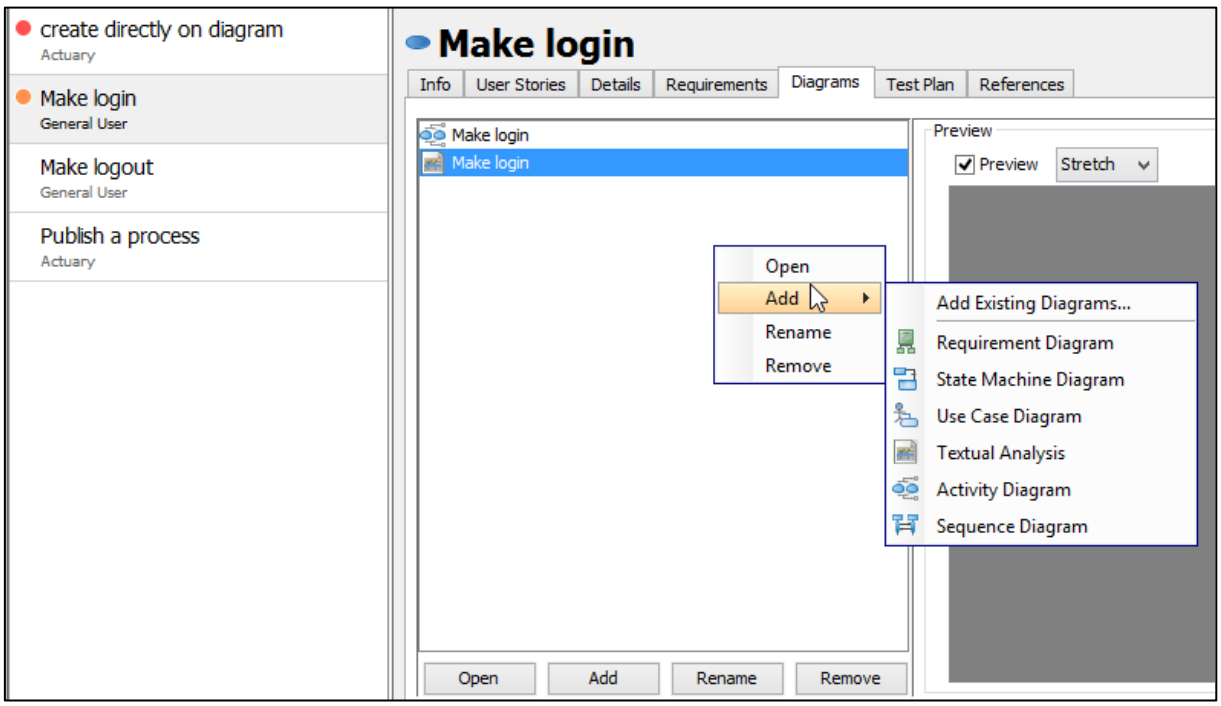


Figura 46 – Criação de novos diagramas no Visual Paradigm.

- É também possível adicionar qualquer referência usada no levantamento, como *websites*, diagramas, imagens, documentos, etc. (Figura 47);

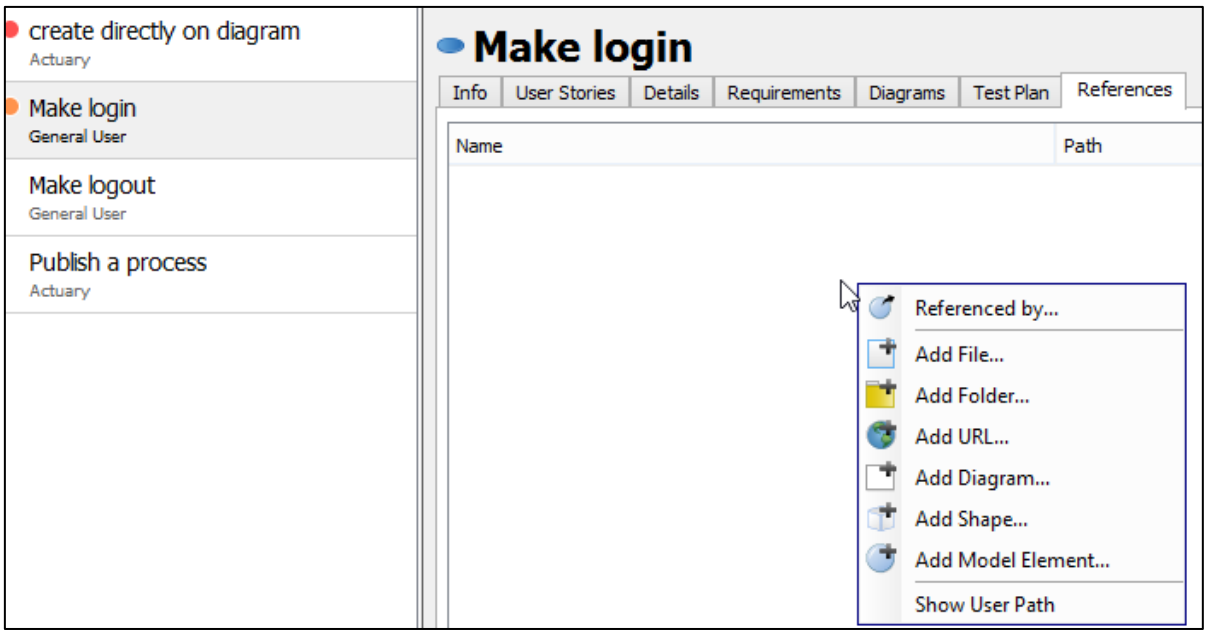


Figura 47 – Adição de referências usadas num requisito no Visual Paradigm.

- Permite também descrever o plano de testes e configurações necessárias (Figura 48);

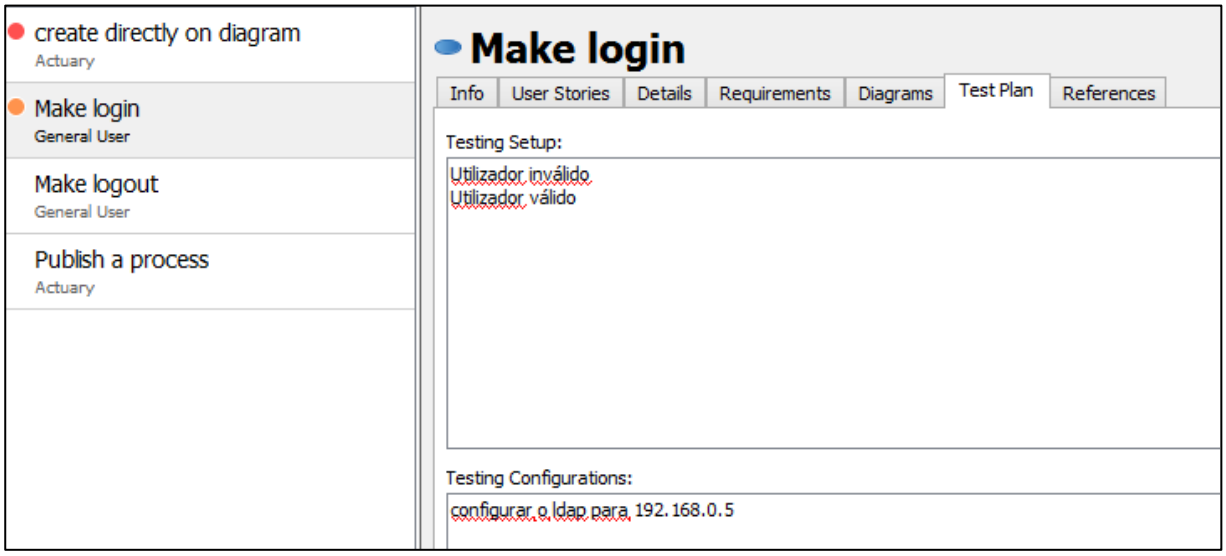


Figura 48 – Criação do plano de testes no Visual Paradigm.

- Permite também criar histórias de utilizador (*user storys*) para os diferentes requisitos, possibilitando a sua filtragem e organização (Figura 49) e permitindo colocar as mesmas em desenvolvimento de forma a permitir controlar (Figura 50);

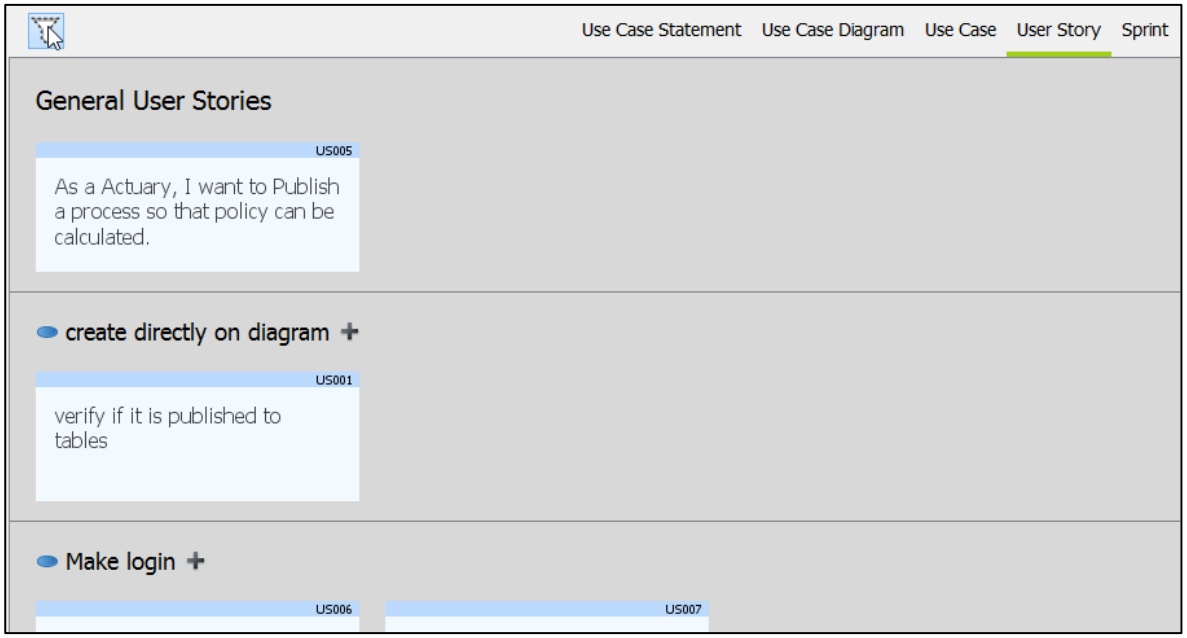


Figura 49 – Criação das user storys no Visual Paradigm.

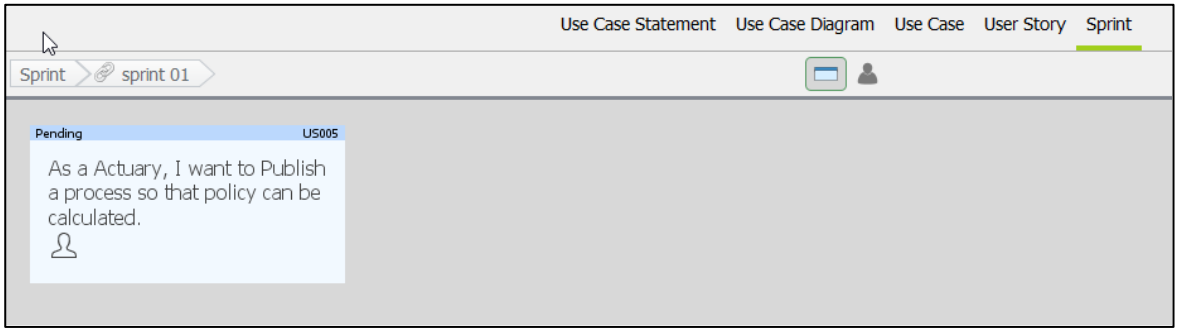


Figura 50 – Colocação de uma user story em desenvolvimento no Visual Paradigm.

- A possibilidade de realizar análises de impacto nas alterações (Figura 51 e Figura 52);

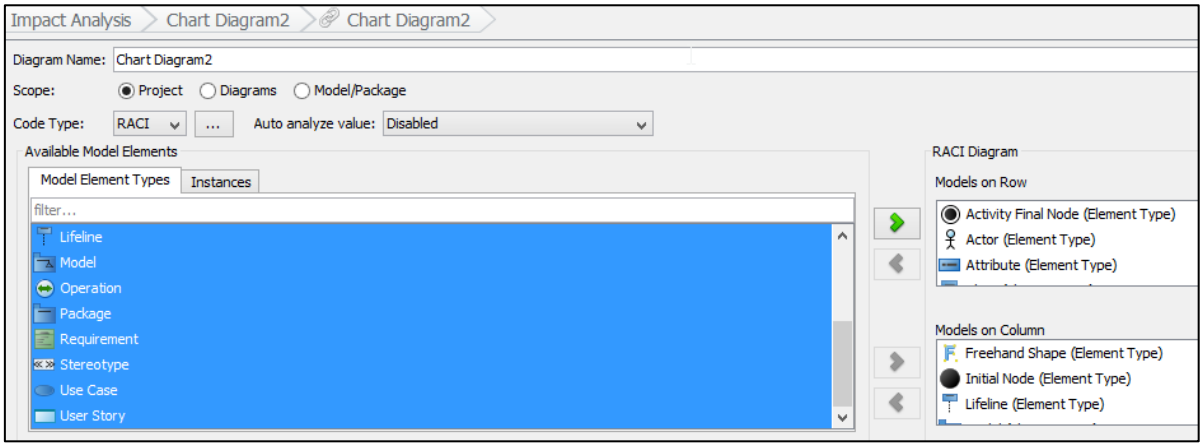


Figura 51 – Escolha dos elementos a analisar no Visual Paradigm.

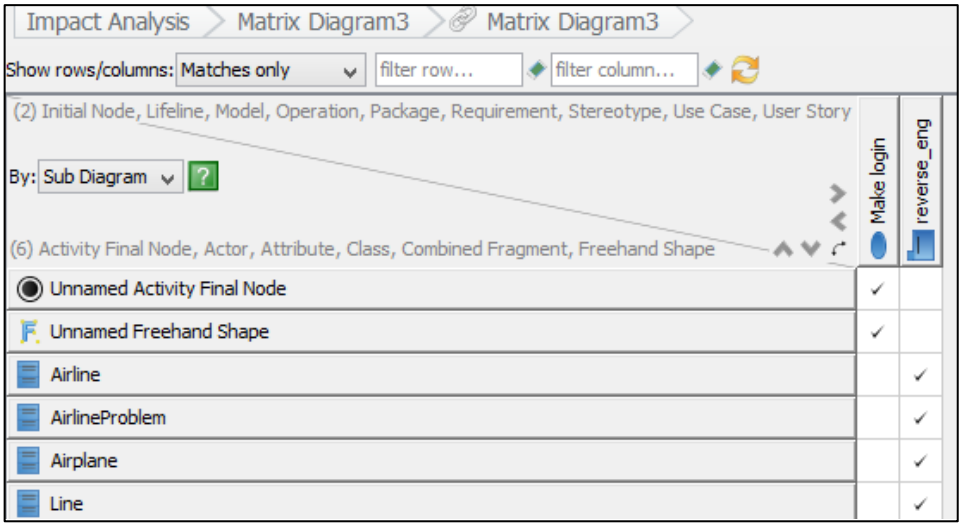


Figura 52 – Análise de impacto no Visual Paradigm.

- Possui um dicionário de termos para ajudar a reduzir ambiguidade entre intervenientes (Figura 53);

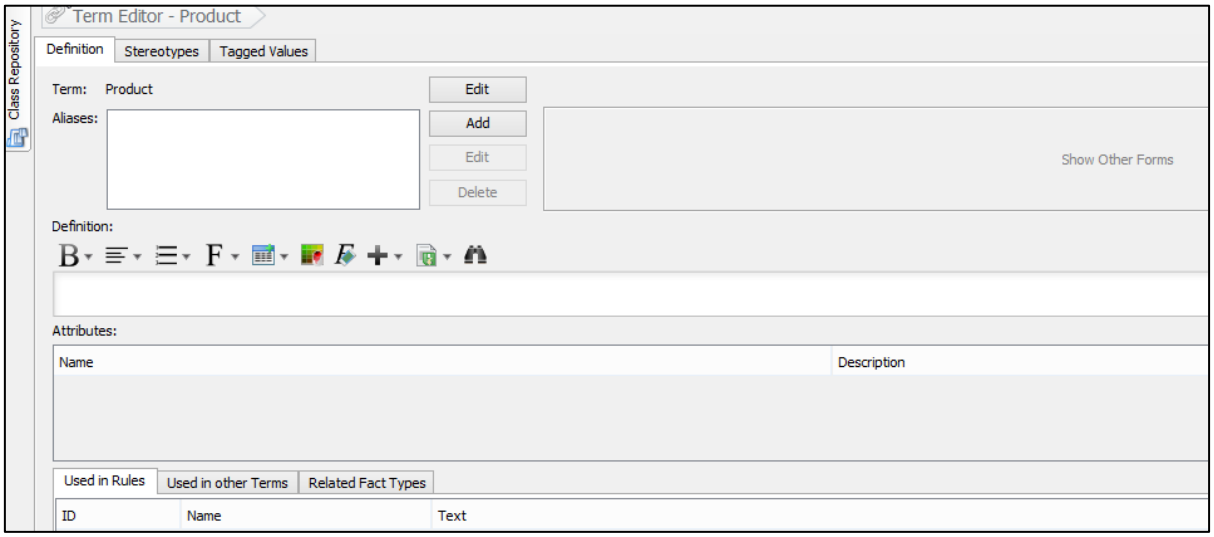


Figura 53 – Dicionário de termos no Visual Paradigm.

- Permite realizar também modelação UML (Diagramas ER, classes, sequência, etc.) e *reverse-engineering* de código (Figura 54);

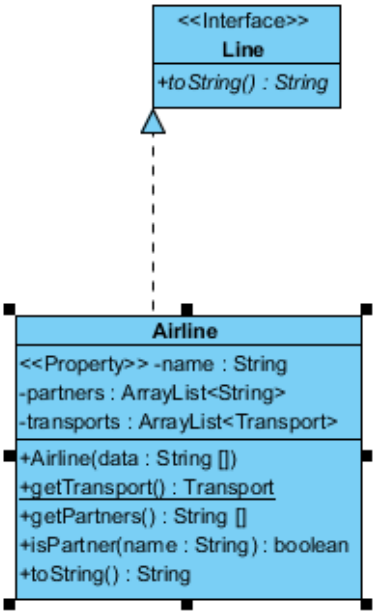


Figura 54 – Diagrama UML no Visual Paradigm.

- É possível visualizar diagramas por tipo e exportar os diagramas (Figura 55) e informações pretendidos, quer para ambiente *web* (Figura 56), quer para formato *pdf* (Figura 57);

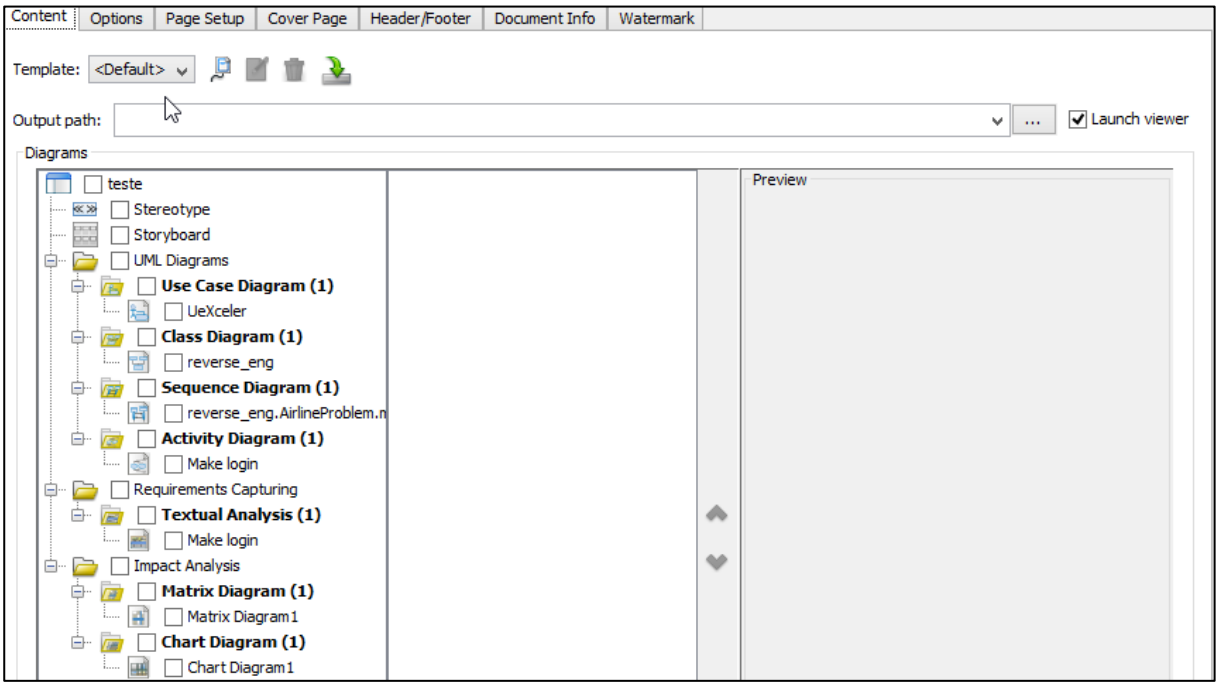


Figura 55 – Seleção dos elementos a exportar no Visual Paradigm.

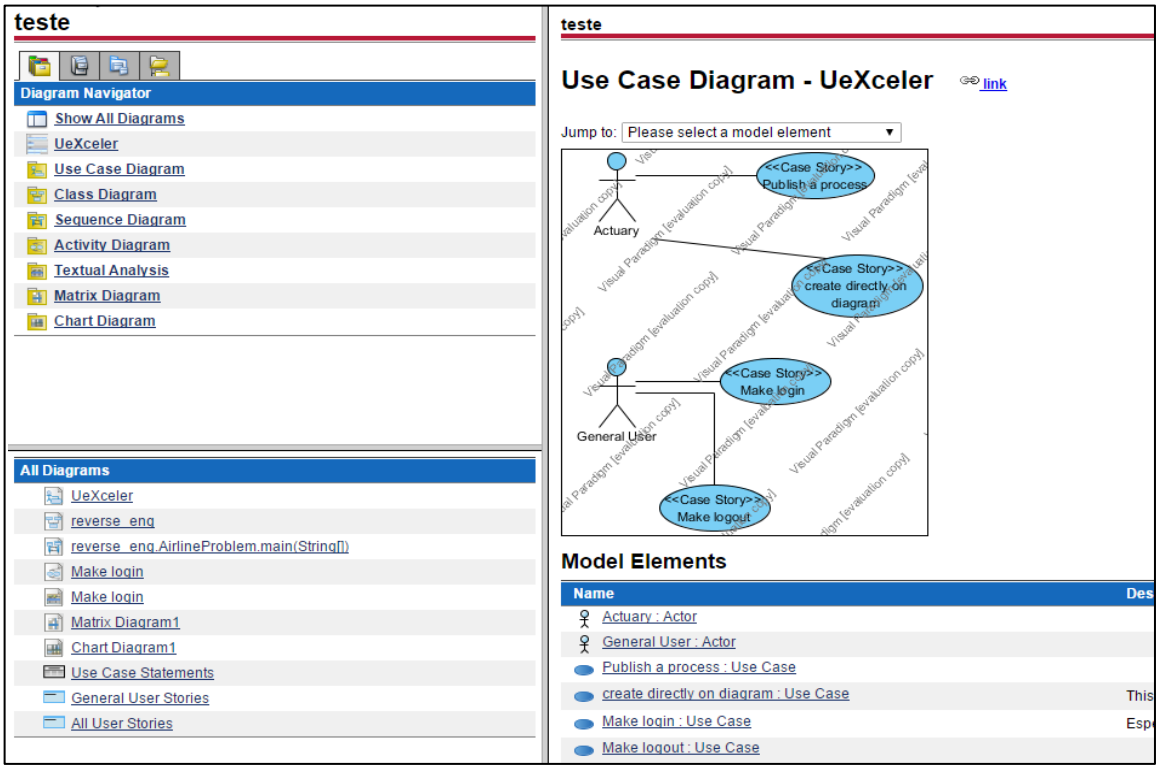


Figura 56 – Exportação para ambiente web no Visual Paradigm.

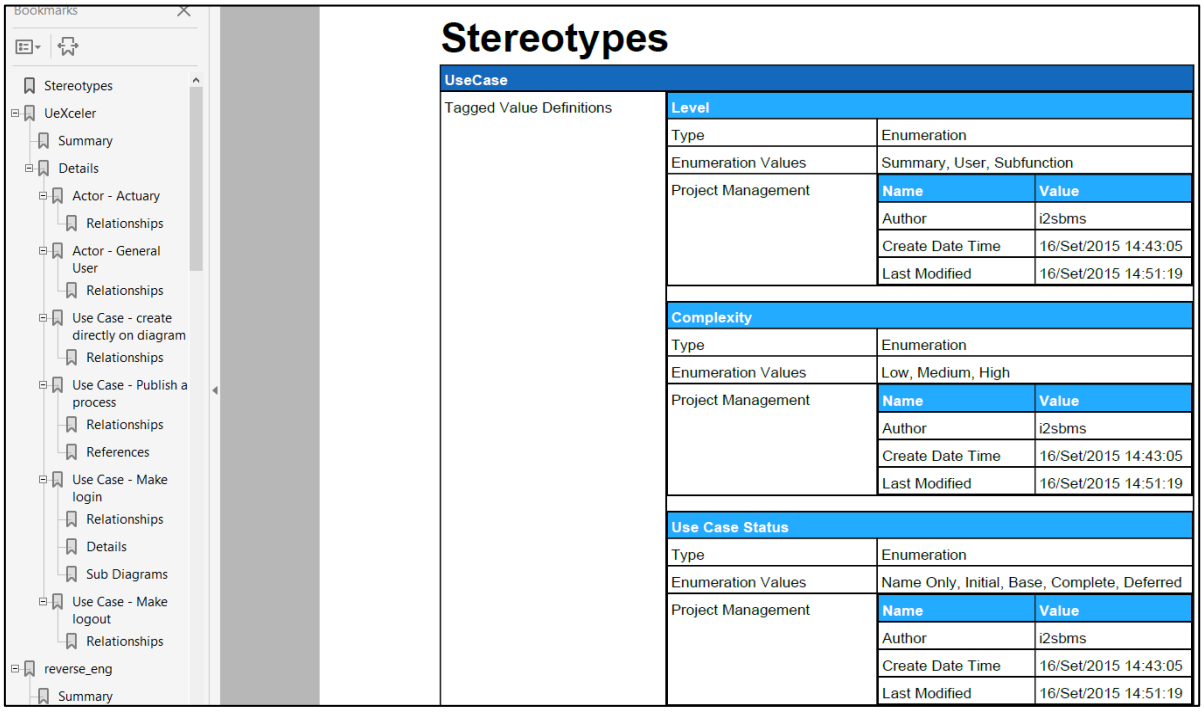


Figura 57 – Exportação para pdf no Visual Paradigm.

4. Blueprint Requirements Center

Blueprint (Blueprint Software Systems Inc., 2016) é uma solução baseada na *cloud* para realizar os requisitos de forma colaborativa, com suporte para todo o ciclo de vida dos requisitos. Possui também a geração de testes automáticos.

Outra ferramenta que foi avaliada é o *blueprint requirements center*. Esta ferramenta é muito completa, possuindo:

- Gestão de todos os documentos para a especificação de requisitos (fontes de informação, diagramas, ecrãs, *mock-ups*, casos de uso, glossário e documentos gerados) (Figura 58);

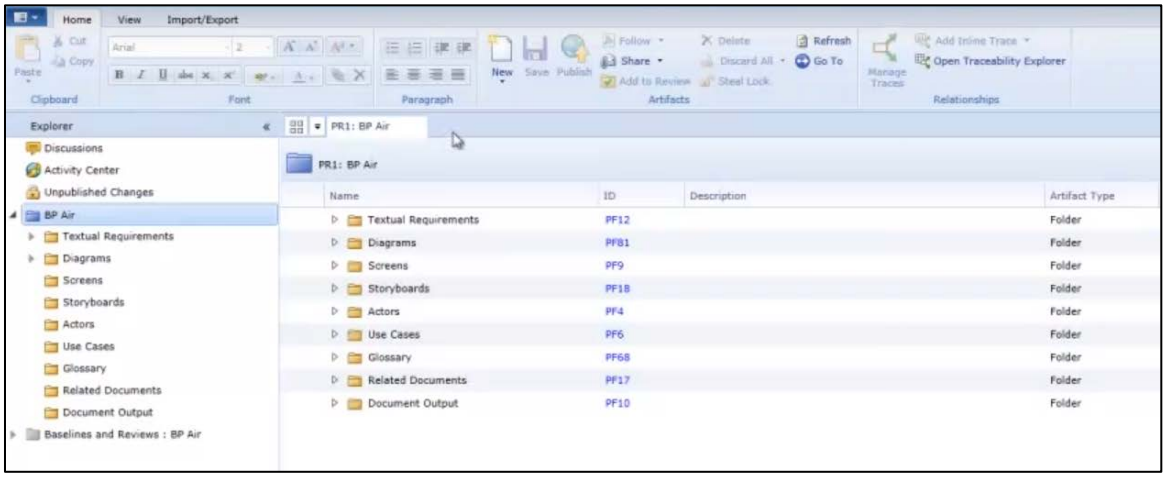


Figura 58 – Gestão dos diferentes documentos no Blueprint.

- Possui um glossário onde é possível definir um termo e sua definição (Figura 59);

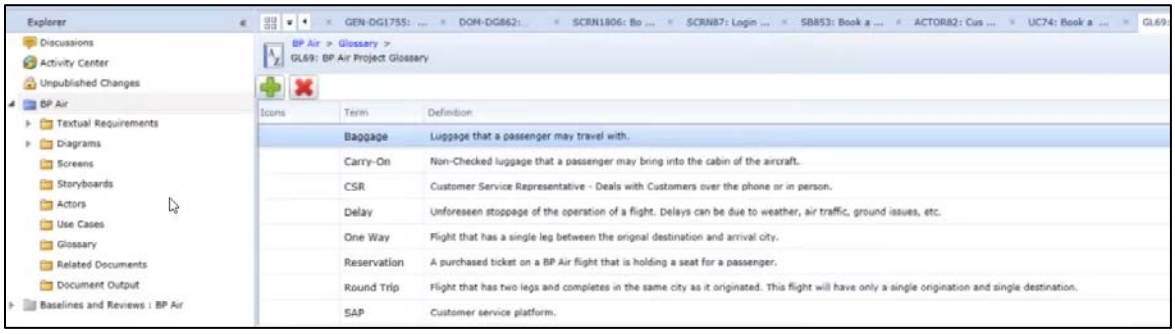


Figura 59 – Glossário de termos no Blueprint.

- Para cada requisito é possível catalogar o mesmo pelo seu tipo, definir a *release, stakeholder*, prioridade, risco e estado (Figura 60);

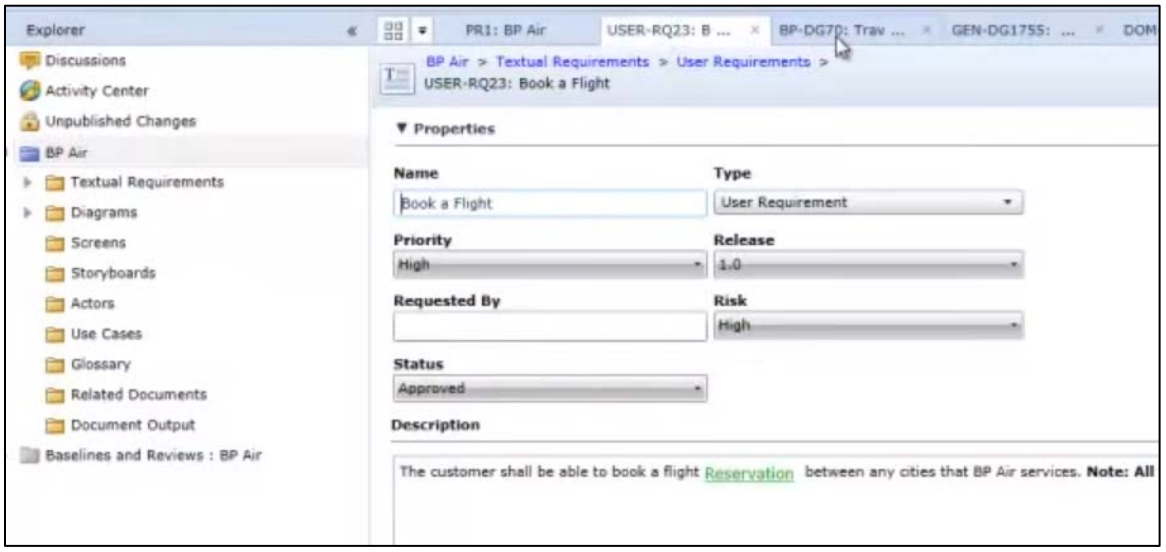


Figura 60 – Catalogação dos requisitos no Blueprint.

- Permite criar vários diagramas como o diagrama de casos de uso (Figura 61), *user stories* e respetivo diagrama com muita facilidade (Figura 62), diagramas de fluxo (Figura 63) e diagrama de entidade (Figura 64);

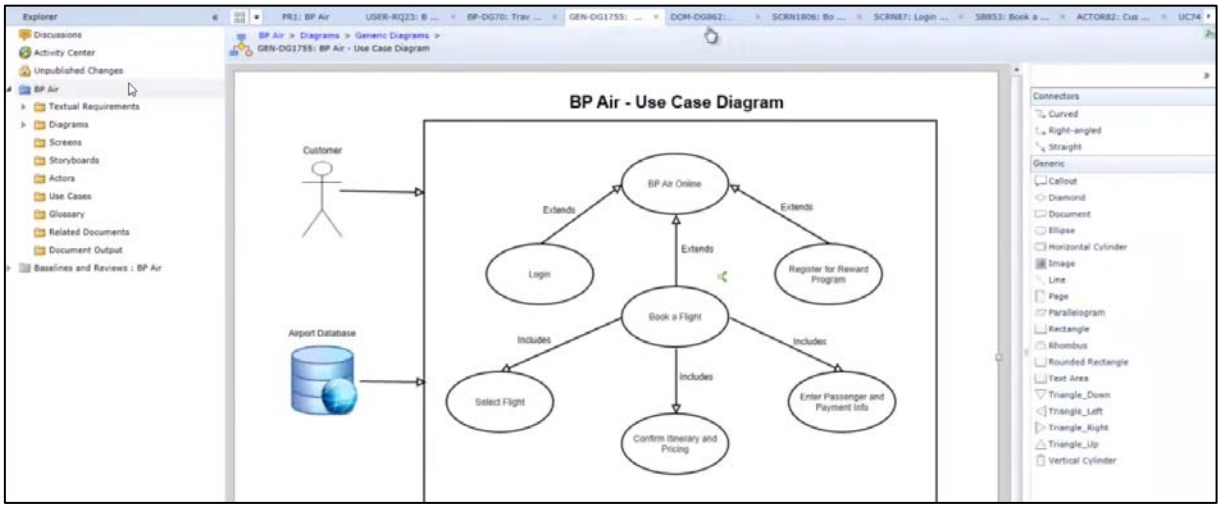


Figura 61 – Diagrama de casos de uso no Blueprint.

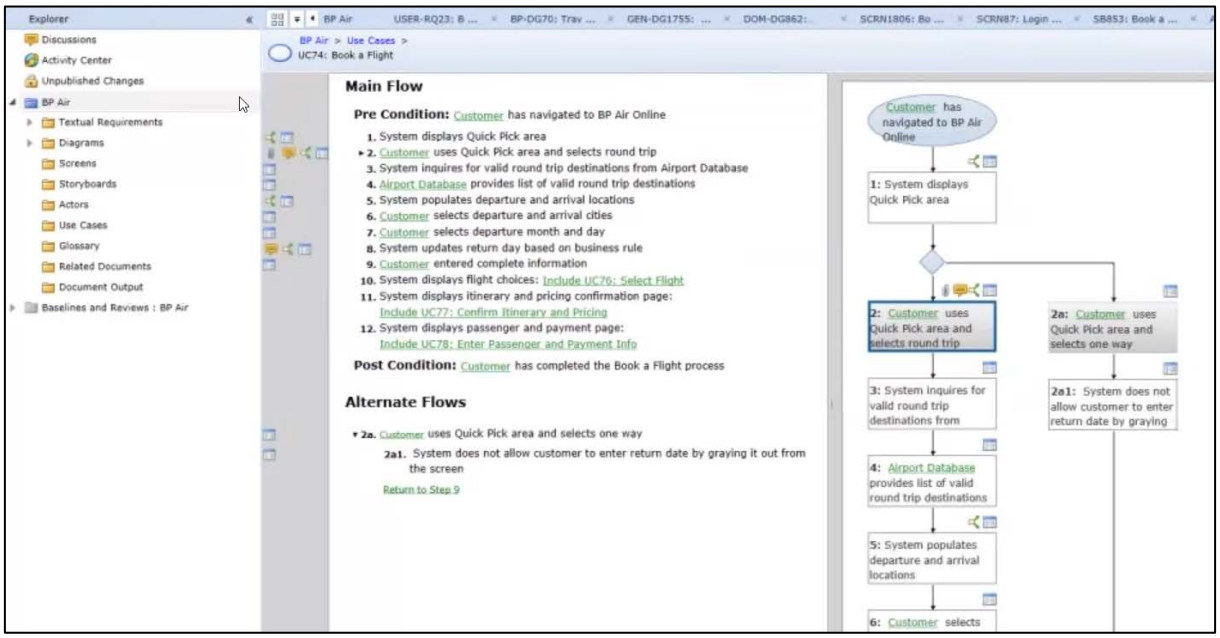


Figura 62 – User story e respetivo diagrama no Blueprint.

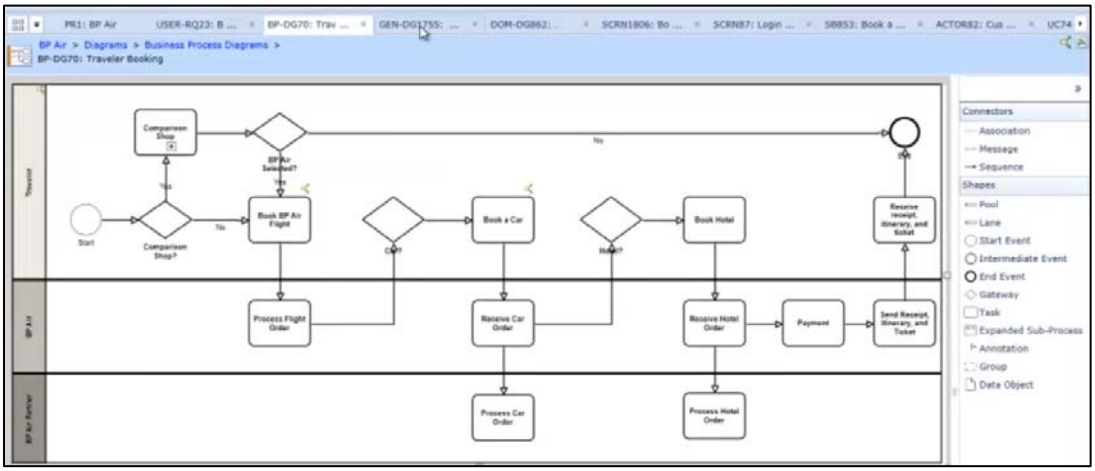


Figura 63 – Diagrama de fluxo no Blueprint.

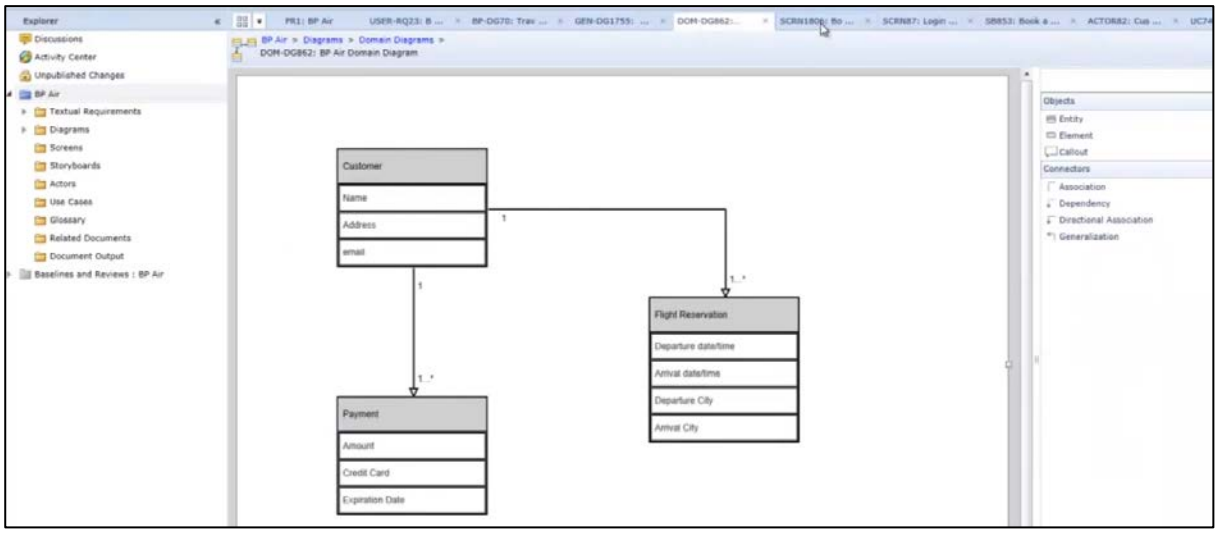


Figura 64 – Diagrama de Entidades no Blueprint.

- Gerir capturas de ecrã, criar *mock-ups* ou *storyboards* com fluxos entre capturas de ecrã e *mock-ups;*
- Permite realizar análises de impacto automatizadas (Figura 65) e visualizar diretamente o documento associado (Figura 66):

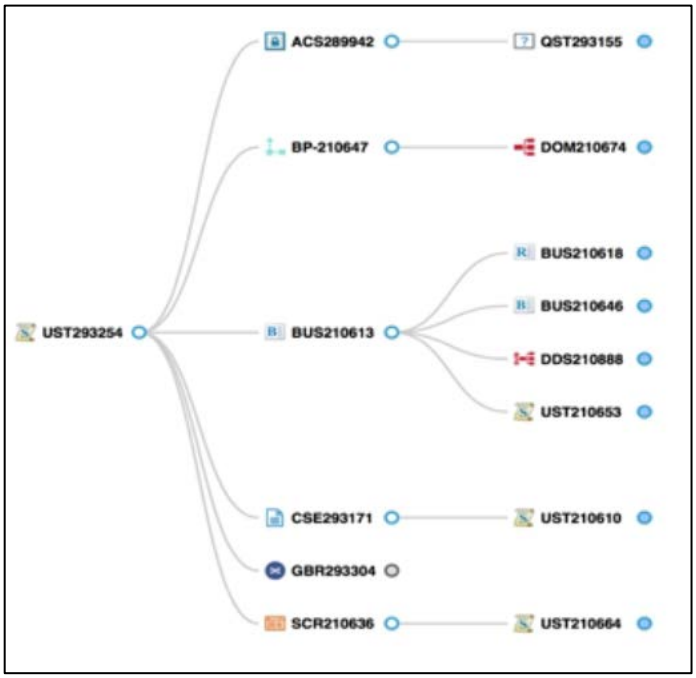


Figura 65 – Análise de impacto no Blueprint.

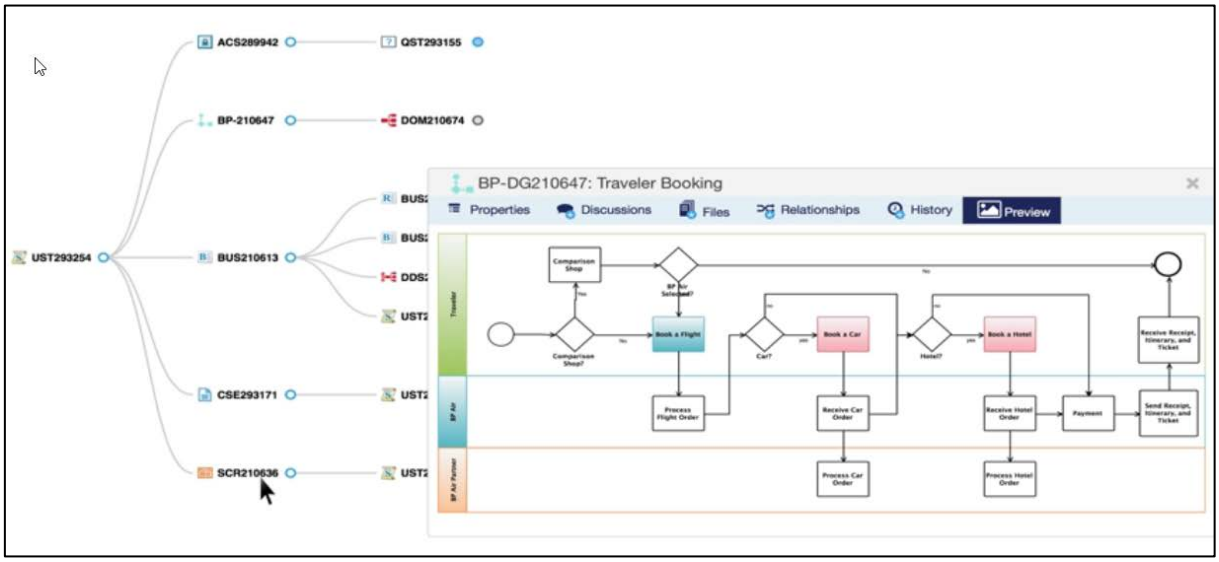


Figura 66 – Visualização de diagrama diretamente na análise de impacto no Blueprint.

- Possui um sistema de aprovação dos requisitos, o que permite gerir facilmente o que está pronto a implementar, tendo uma visão de pastas (Figura 67) e dos diferentes recursos (Figura 68):

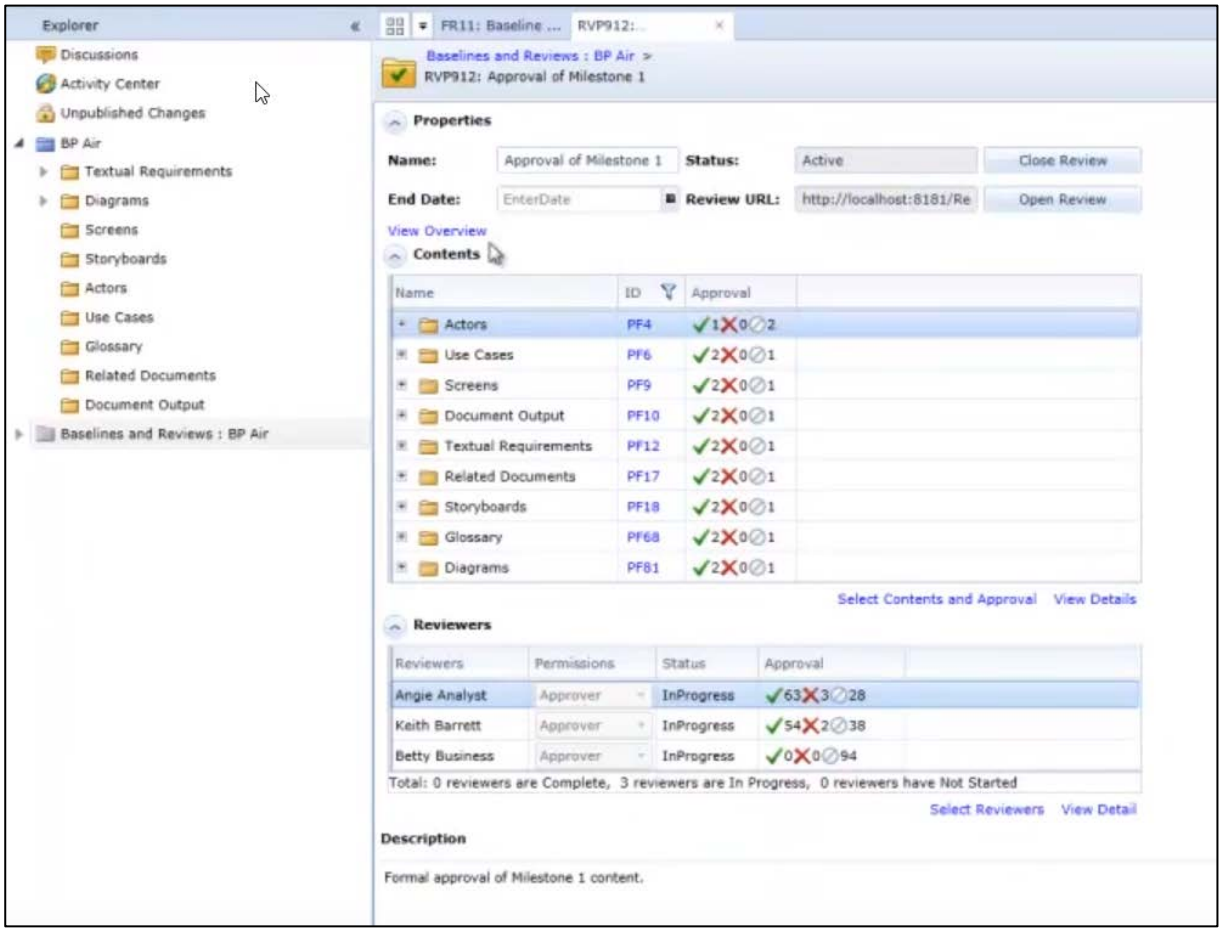


Figura 67 – Sistema de aprovação de requisitos e documentos no Blueprint – visão de pasta.

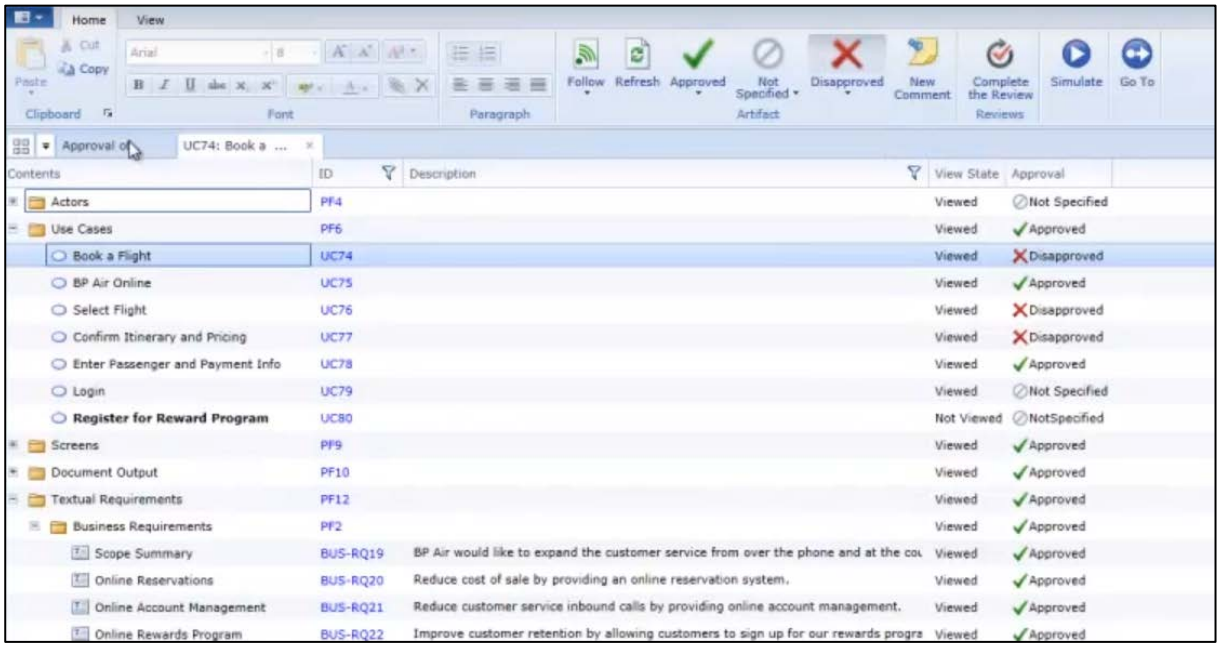


Figura 68 – Sistema de aprovação de requisitos e documentos no Blueprint – visão de documento.

- Permite também visualizar de forma gráfica o progresso do projeto em termos de requisitos (Figura 69):

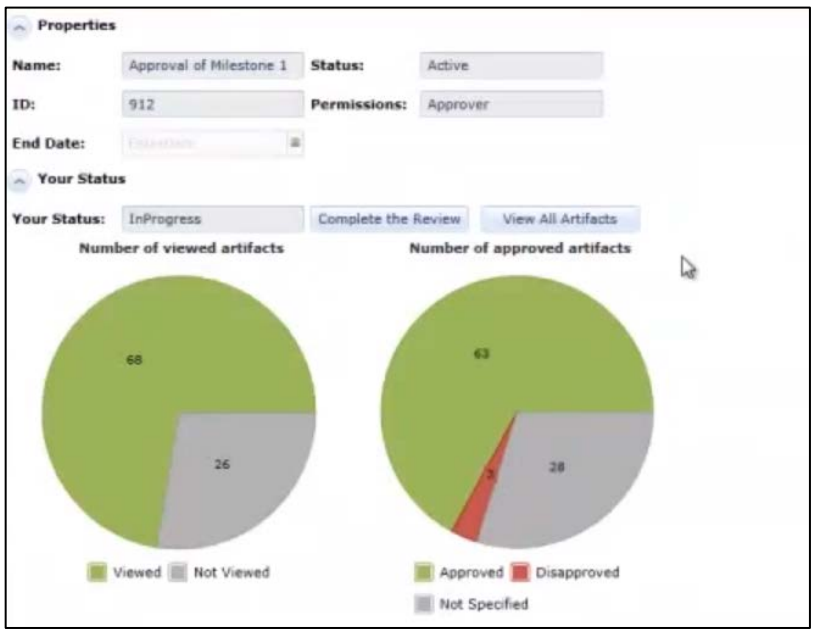


Figura 69 – Estatísticas da aprovação da especificação de requisitos do projeto no Blueprint.

- Possui uma área de gestão do requisito onde tem os seus detalhes, tarefas associadas, dependências, defeitos, discussões sobre o mesmo, revisões, gráficos, casos de teste, execução dos testes e alterações realizadas (Figura 70);

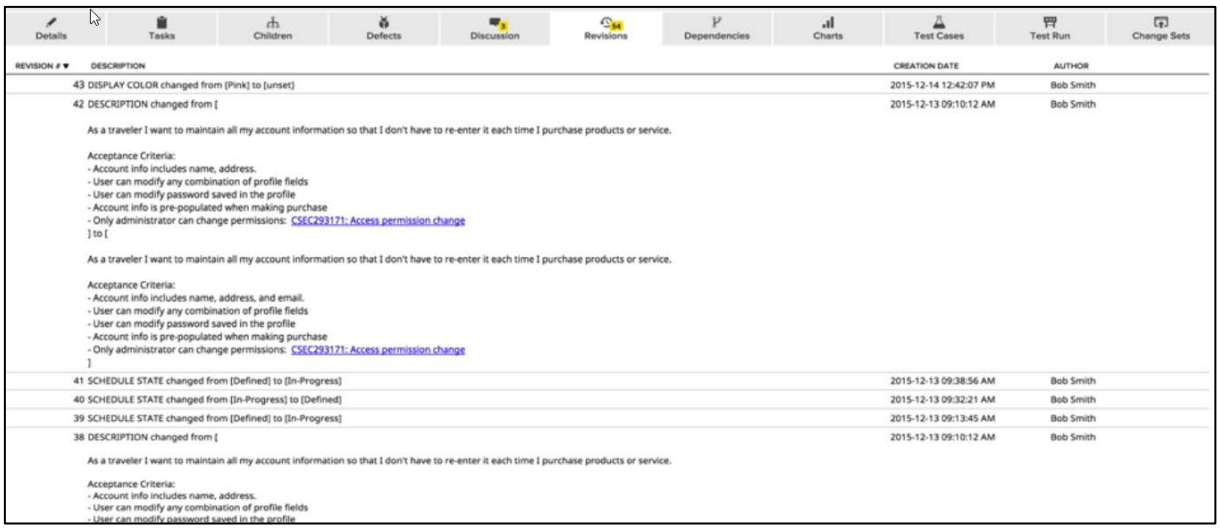


Figura 70 – Área de gestão do requisito no Blueprint.

Esta solução embora, bastante completa, não possui integração com ferramentas de projeto, o que torna muito difícil existir a ligação da especificação com o desenvolvimento. O seu preço é também bastante elevado, sendo apenas possível saber o valor exato através de negociação, no entanto, é superior a 1000 euros para uma empresa pequena, com um conjunto limitado de utilizadores.

B.Ferramentas de *Mock-ups*

1. Balsamiq

O *Balsamiq* é uma ferramenta para a realização de *mock-ups* com centenas de componentes (Figura 71) e bibliotecas gratuitas. Permite também fazer bibliotecas próprias para utilização.

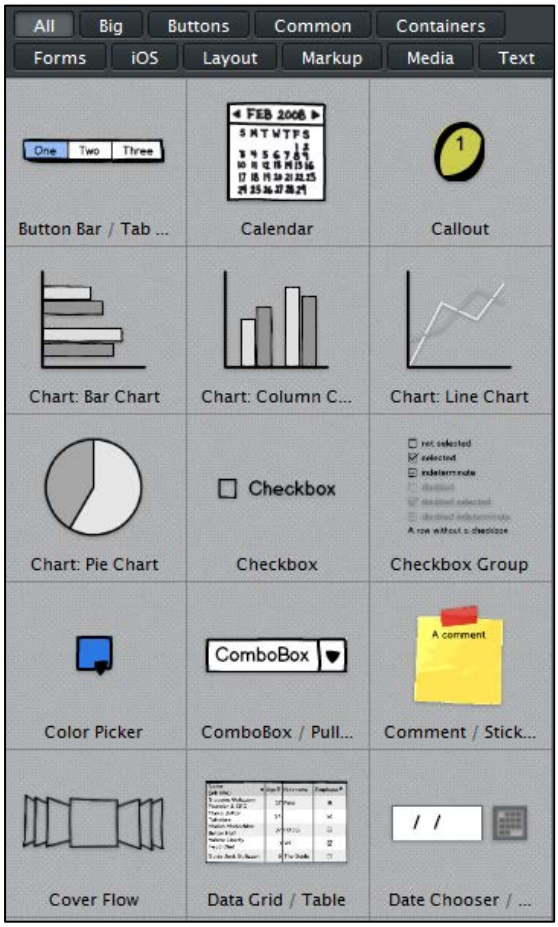


Figura 71 – Alguns dos componentes existentes no Balsamiq.

Possibilita a realização de *mock-ups* de forma muito rápida (Figura 72). Possui também diferentes bases de contexto como *smartphone*, *web*, *Tablet*, Aplicação de computador, etc.

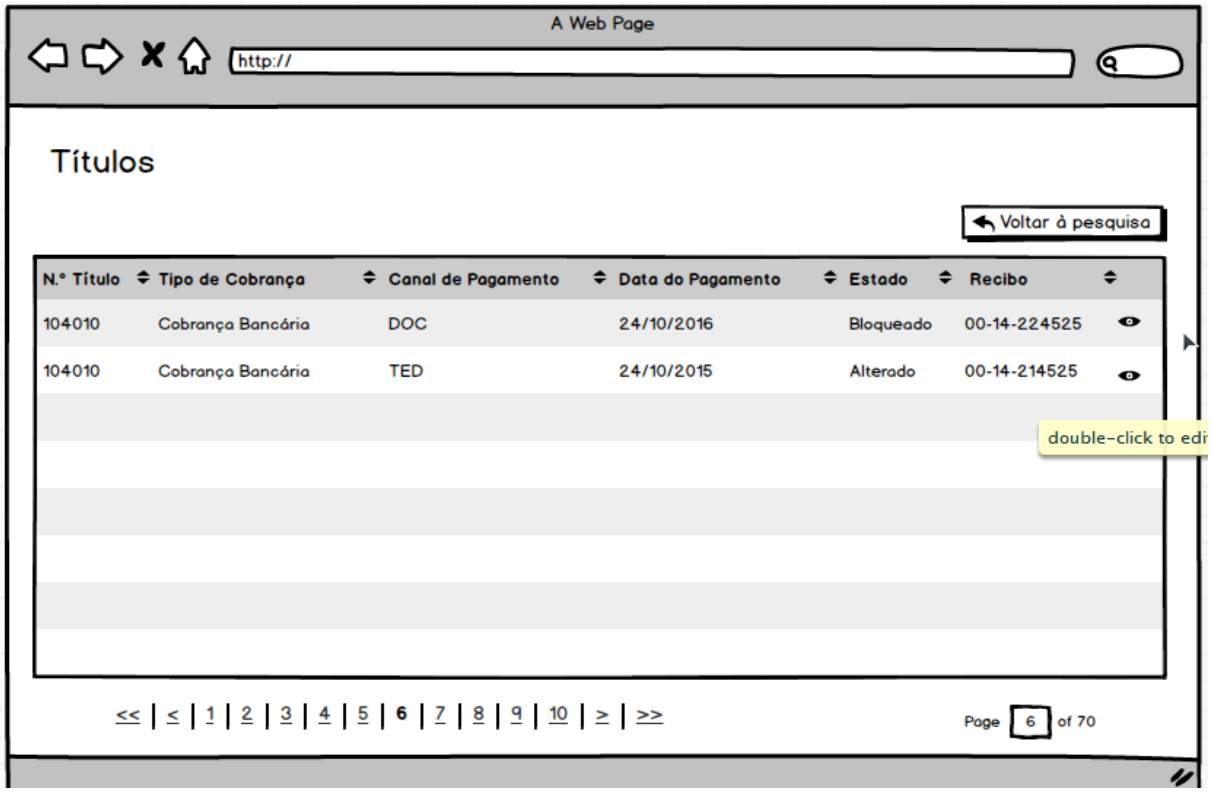


Figura 72 – Exemplo de um mock-up realizado no Balsamiq.

Permite reaproveitar blocos já realizados para usar como novos componentes e configurar os componentes base existentes (Figura 73).

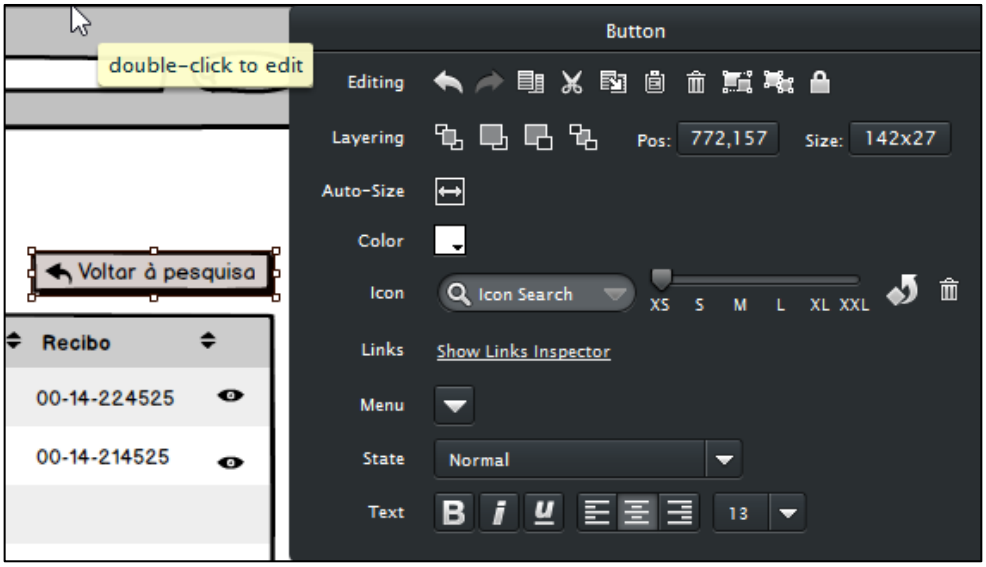


Figura 73 – Propriedades de um componente no Balsamiq – botão.

É também possível agregar vários mock-ups num projeto de forma a ter um agregador.

2. WireFrame Sketcher

O Wireframe Sketcher (WireframeSketcher, 2015) é uma ferramenta para a realização de *mock-ups* integrada com o eclipse. Ao criar um novo projeto é possível logo escolher o tipo de projeto (Figura 74);

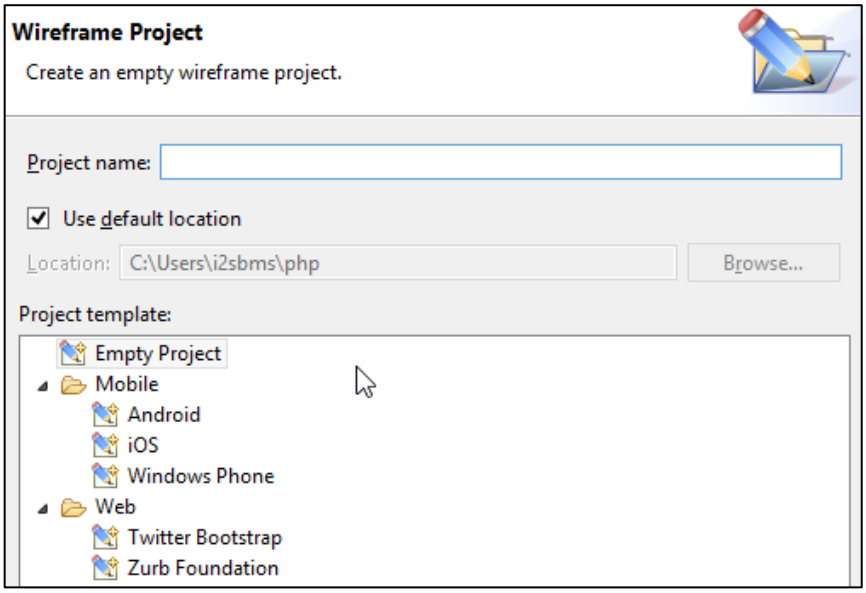


Figura 74 – Escolher o tipo de projeto no WireFrame Sketcher.

Possui imensos componentes para o desenvolvimento do *mock-up* (Figura 75) e permite configurar algumas propriedades por componente e gerir a ordem a ser apresentada – *outline* (Figura 76);

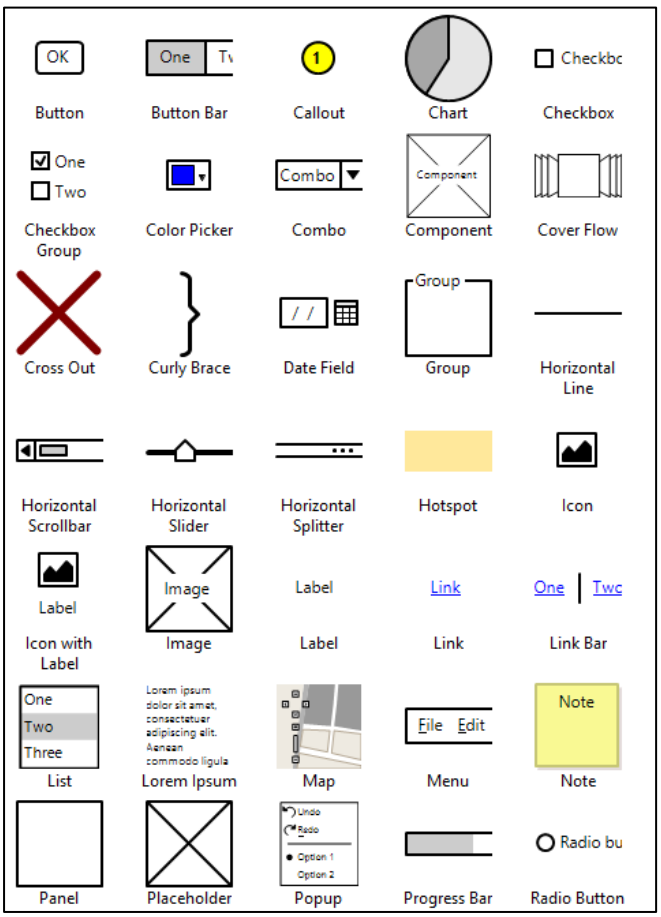


Figura 75 – Componentes no WireFrame Sketcher.

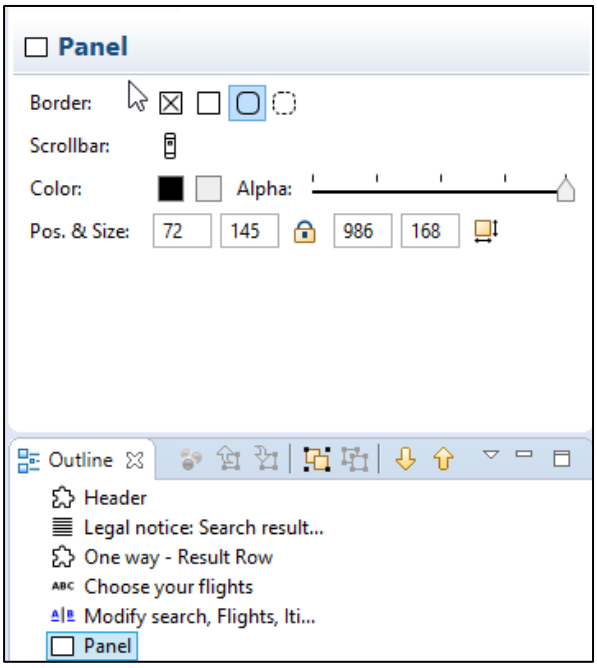
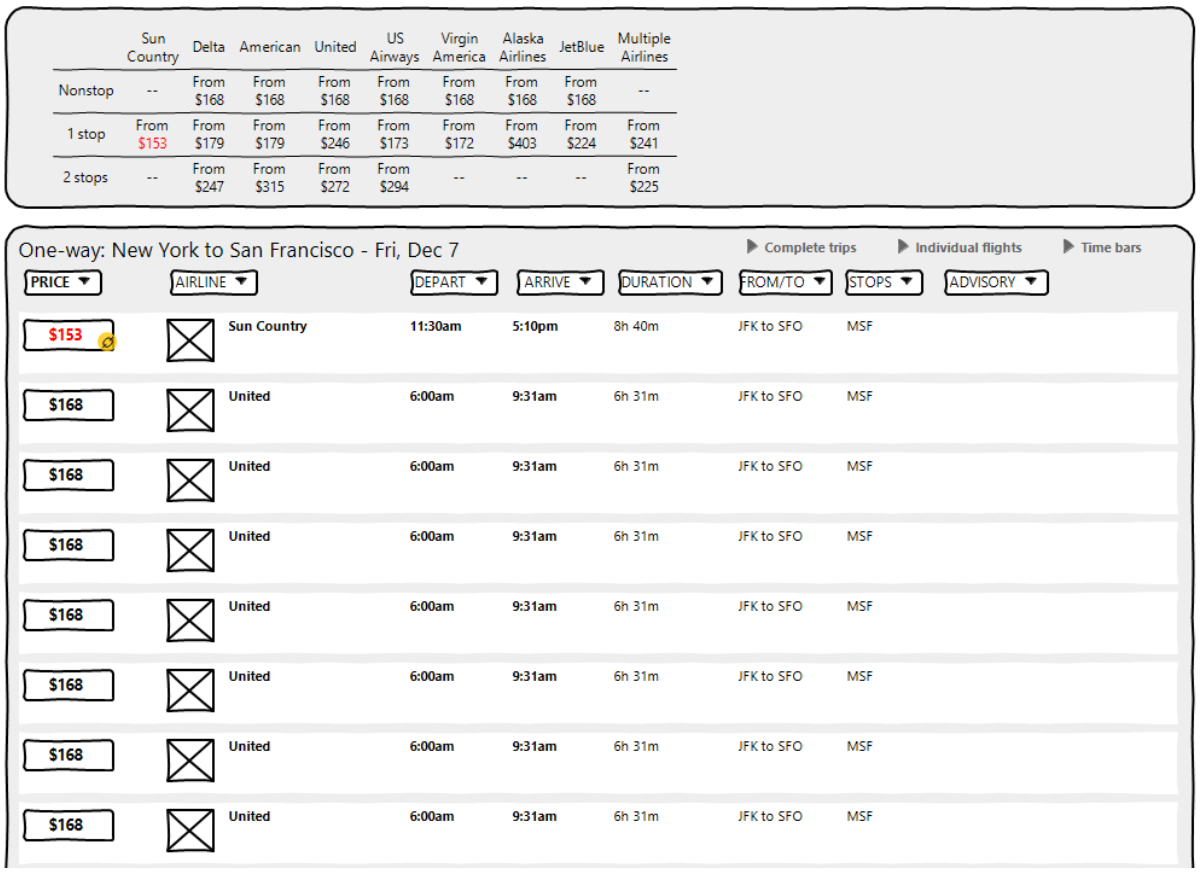


Figura 76 – Personalização de um componente e outline no WireFrame Sketcher.

Possui uma visão agregadora de diversos *mock-ups* (Figura 77) de forma a criar o *storyboard*. É possível ter vários *storyboards* por projeto usando os mesmos *mock-ups,* o que permite o seu reaproveitamento (Figura 78);



	Sun Country	Delta	American	United	US Airways	Virgin America	Alaska Airlines	JetBlue	Multiple Airlines
Nonstop	--	From $168	From $168	From $168	From $168	From $168	From $168	From $168	--
1 stop	From $153	From $179	From $179	From $246	From $173	From $172	From $403	From $224	From $241
2 stops	--	From $247	From $315	From $272	From $294	--	--	--	From $225

Figura 77 – Mock-up exemplo no WireFrame Sketcher.

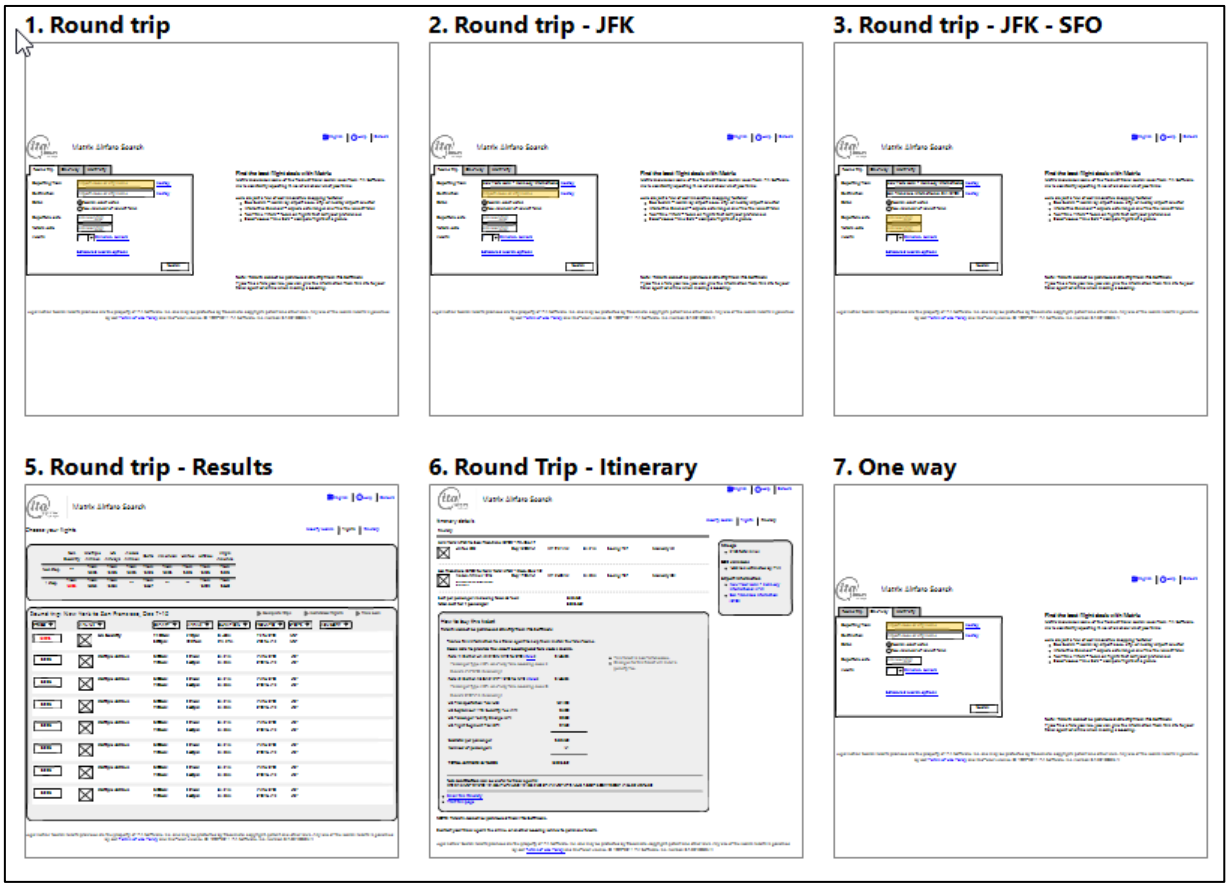


Figura 78 – Conjunto de mock-ups no WireFrame Sketcher.

3. Visual Paradigm

O Visual Paradigm (Visual Paradigm, 2016) possui uma ferramenta integrada que possui o conceito de *storyboard* que permite criar vários *mock-ups* para esse *storyboard* (Figura 79). No final da criação de vários permite uma apresentação sequencial do *storyboard*. É também possível utilizar o mesmo *mock-up* para diferentes *storyboards*.

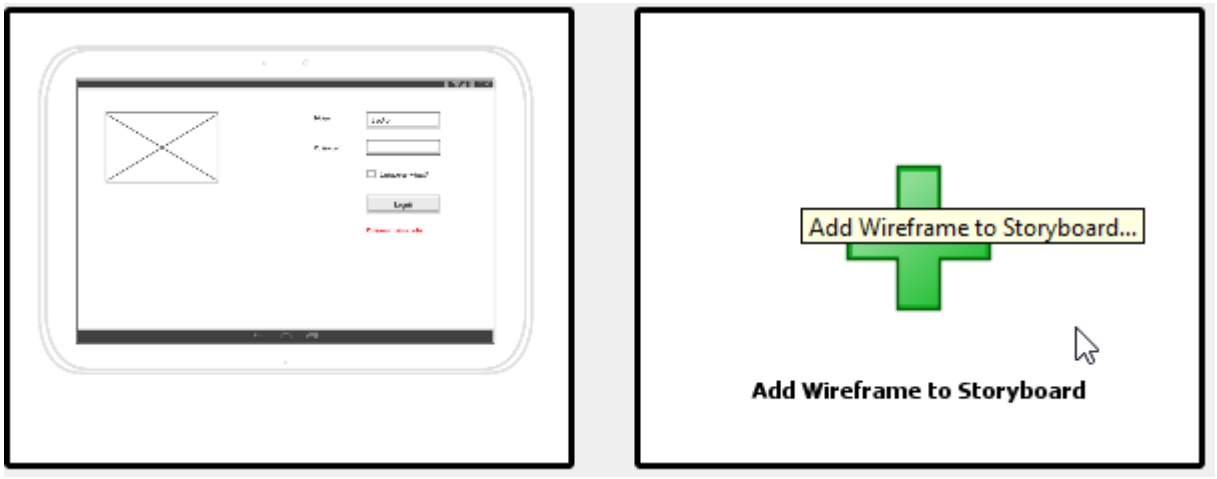


Figura 79 – Criação de mock-up para um storyboard no Visual Paradigm.

Permite realizar *mock-ups* para diferentes contextos (*web, tablet, smartphone*, aplicação de computador, etc.), possuindo componentes específicos para cada um dos mesmos.

Os *mock-ups* contruídos podem ser relacionados com os requisitos (Figura 80);

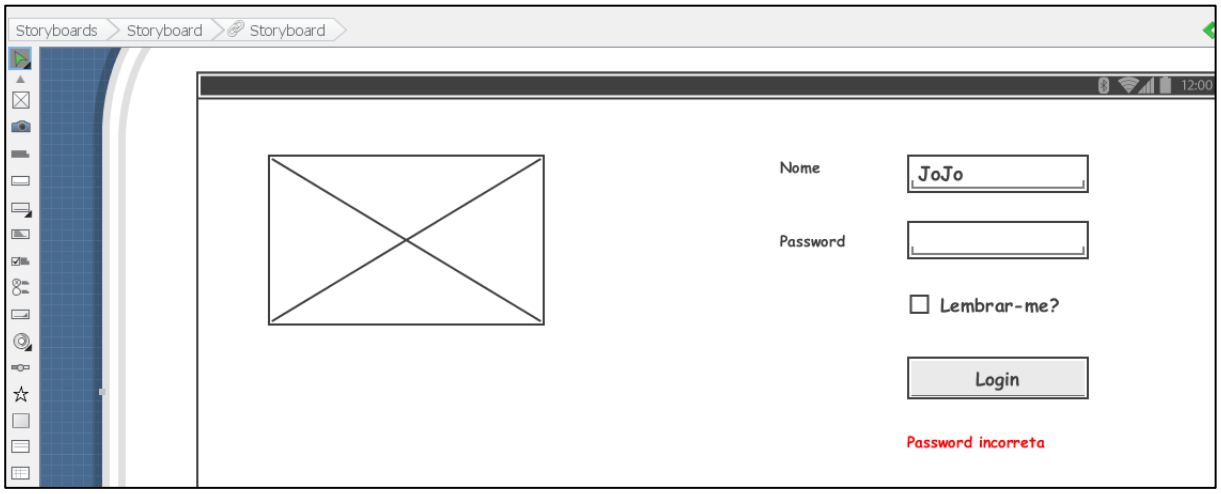


Figura 80 – Ferramenta para realização de mocks no Visual Paradigm.

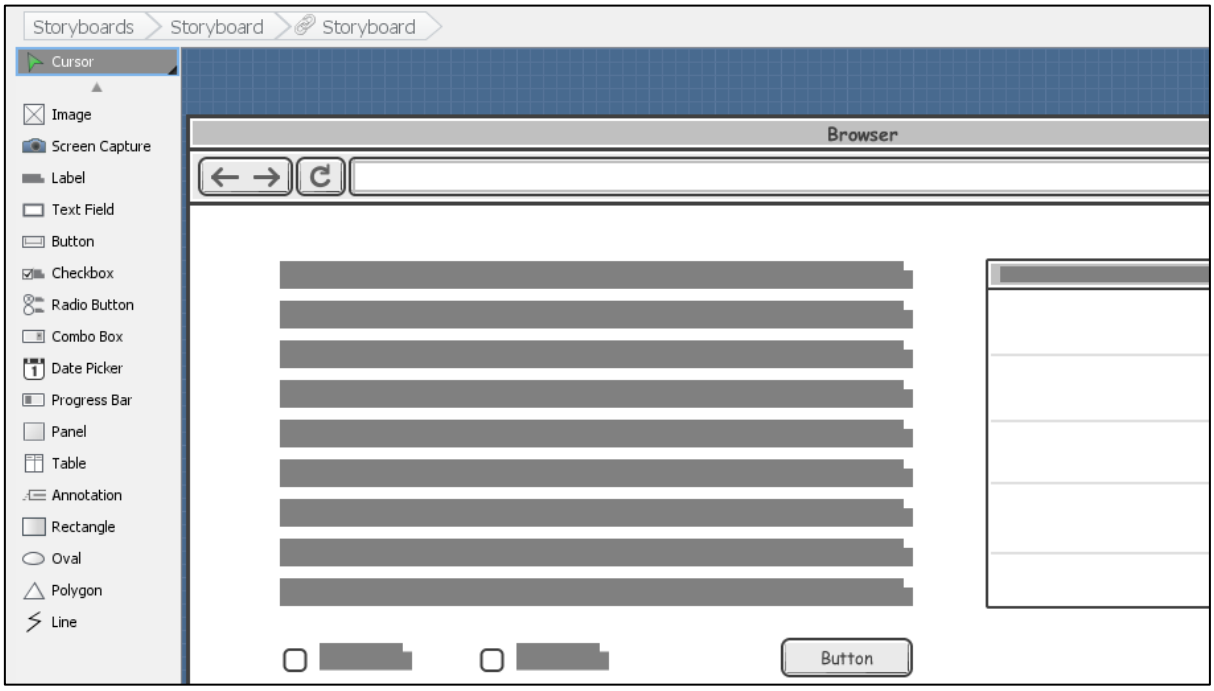


Figura 81 – Mock-up web realizado no visual paradigm.

4. Blueprint Requirements Center

Blueprint (Blueprint Software Systems Inc., 2016) possui uma ferramenta integrada para a realização de *mock-ups* e gestão de capturas de ecrã. Sobre esta área, esta aplicação permite:

- Criar *mock-ups* ou capturas de ecrã (Figura 82)*:*

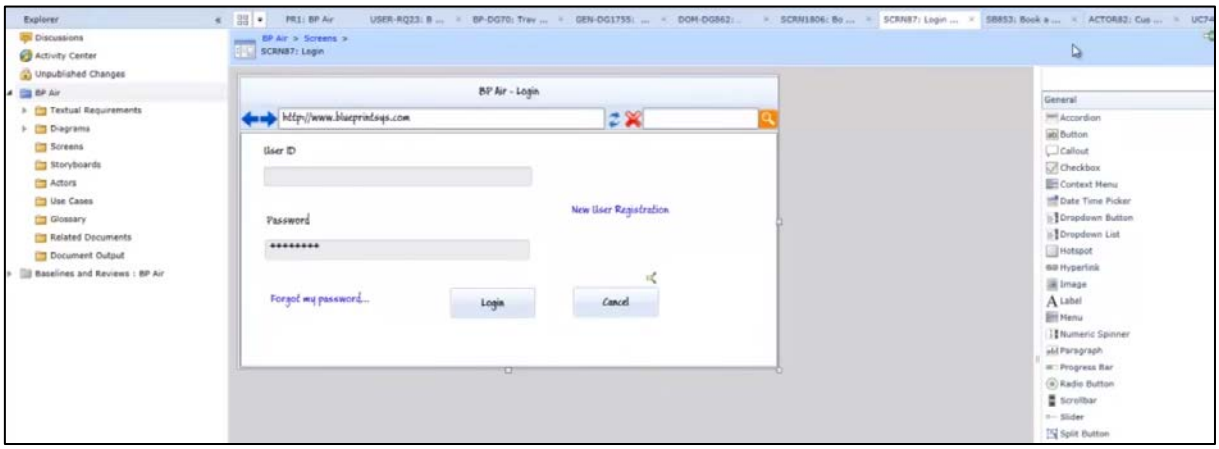


Figura 82 – Mock-up no Blueprint.

- Permite criar fluxos entre capturas de ecrã e *mock-ups* e, ao contrário das outras aplicações, esta permite mais que um só fluxo (Figura 83):

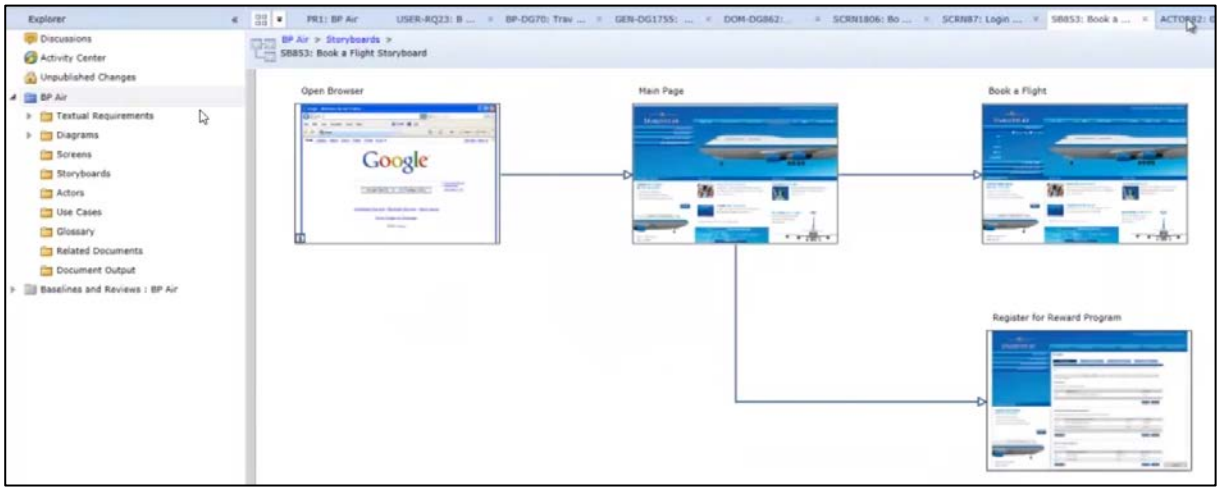


Figura 83 – Storyboards no Blueprint.

C.Ferramentas de Qualidade

1. SonarQube

Sonarqube (SonarQube, 2015) é um exemplo de uma ferramenta que verifica a qualidade de código baseado num conjunto de regras definidas, regras estas que possuem erros.

Já possui algumas destas regras por defeito, sendo possível adicionar regras personalizadas (Figura 84), catalogar as mesmas em termos de contexto e tipo – por exemplo "erro", "má-prática" ou "performance" – e mesmo priorizar as mesmas a nível de qualidade de código – erro bloqueante, crítico, grave, menor ou informativo (Figura 85).

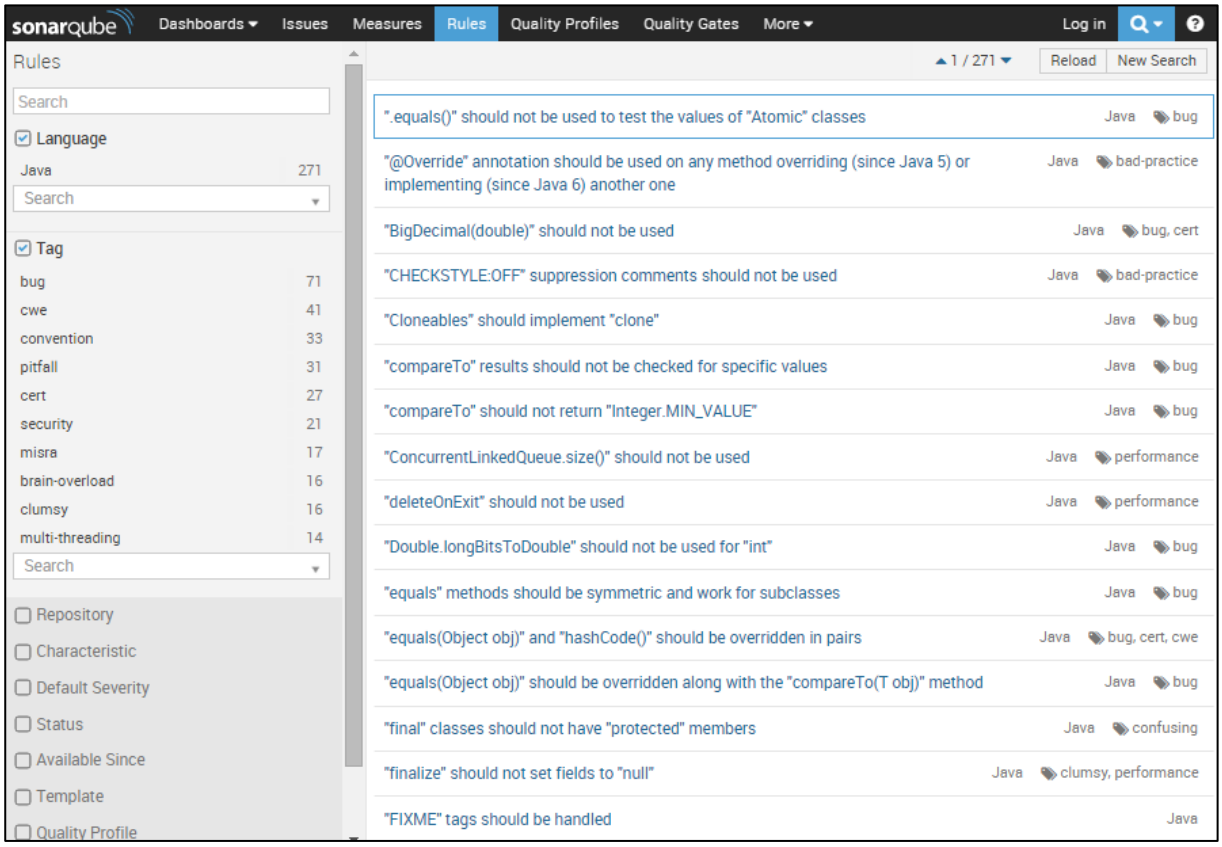


Figura 84 – Regras de validação de código com catalogação no SonarQube.

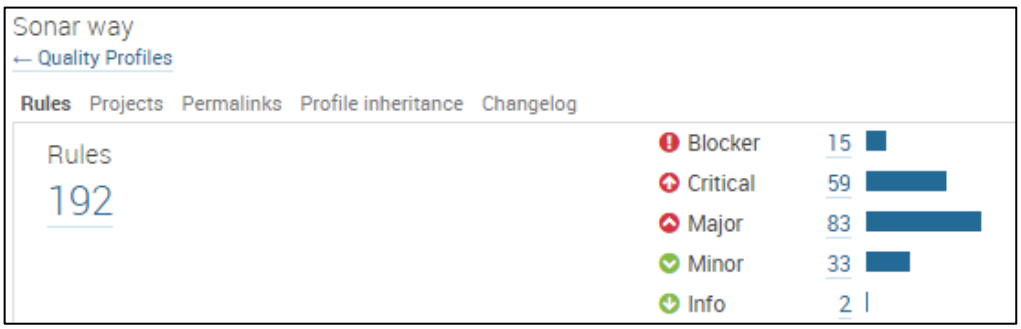


Figura 85 – Criticidade dos erros encontrados no SonarQube.

É possível ter vários projetos no sonar, existindo um ecrã com os resultados da análise por projeto. As métricas que são mostradas são configuráveis (Figura 86) e permitem obter dados muito interessantes do projeto como o número de erros, erros por tipo, linhas de código, ficheiros, funções, classes, percentagem de código duplicado, complexidade do código, etc. (Figura 87).

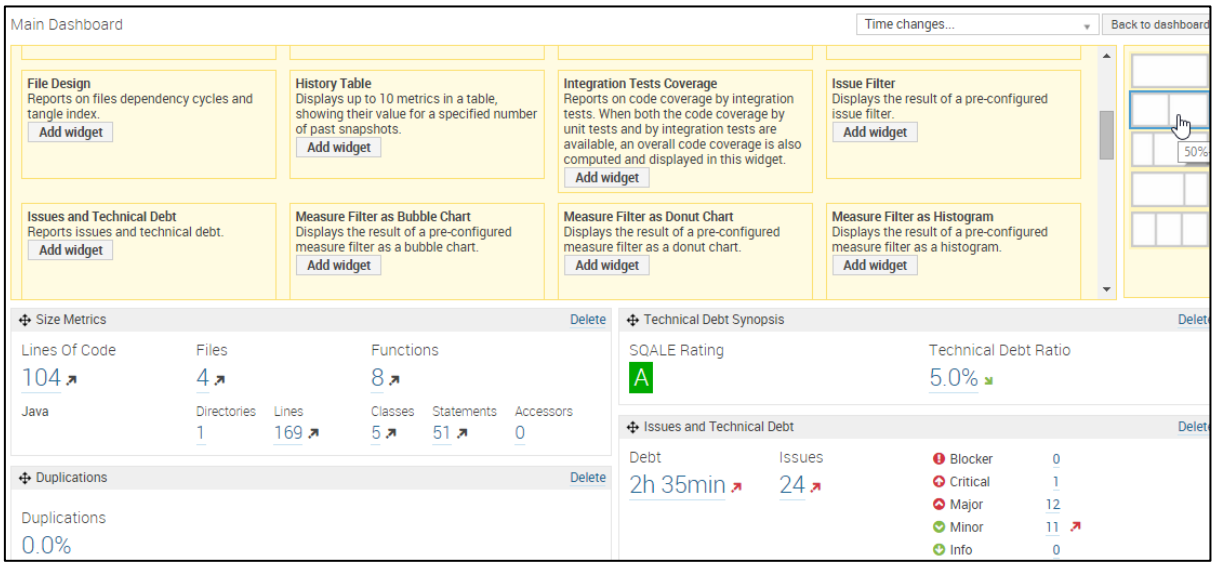


Figura 86 – Configuração das métricas a serem apresentadas no ecrã de análise no SonarQube.

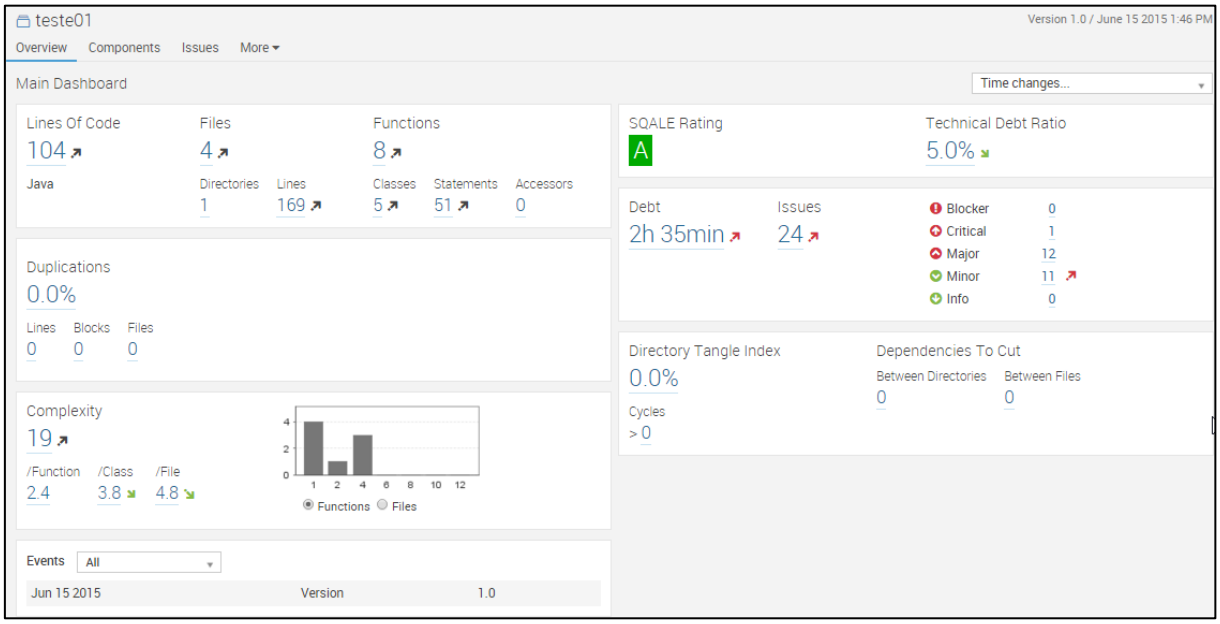


Figura 87 – Ecrã de análise com as métricas do projeto no SonarQube.

No ecrã de análise é possível ir selecionando as métricas de forma a ir refinando os resultados. Por exemplo, se for efetuado um clique nos erros, será apresentado a lista de erros do projeto (Figura 88). Se for efetuado um clique em erros “critical”, será apresentado os erros graves.

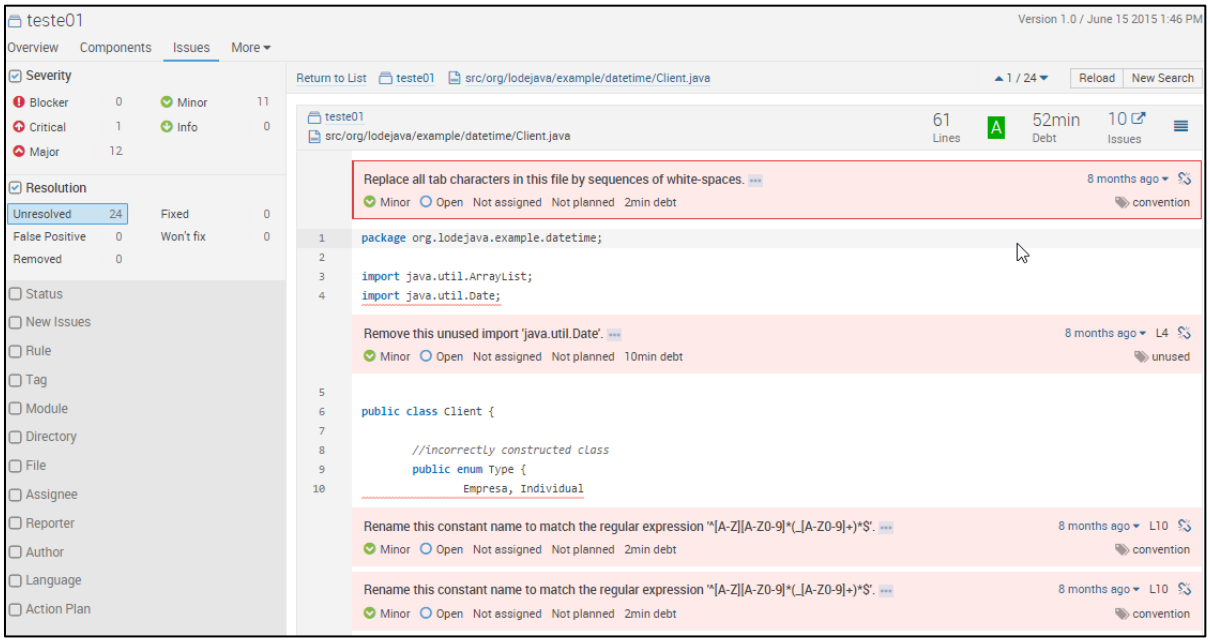


Figura 88 – Ecrã com os erros encontrados no projeto no SonarQube.

Referências Bibliográficas

Atlassian Pty Ltd, 2015. *Confluence.* [Online]
Available at: https://www.atlassian.com/software/confluence
[Acedido em 01 2015].

Atlassian Pty Ltd, 2015. *jira.* [Online]
Available at: https://www.atlassian.com/software/jira
[Acedido em 12 2015].

Aurum, A. & Wohlin, C., 2005. *Engineering and Managing Software Requirements.* New York, NY: Springer Science & Business Media.

Balsamiq Studios, LLC, 2015. *balsamiq.com.* [Online]
Available at: balsamiq.com
[Acedido em 15 11 2015].

Beatty, J., 2015. *PMI Business Analysis Virtual Conference.* s.l., s.n.

Berander, P. & Andrews, A., 2005. Requirements prioritization. Em: *Engineering and managing software requirements..* Heidelberg: Springer Berlin.

Bieg, D. P., 2015. *PMI Business Analysis Virtual Conference.* s.l., s.n.

Blueprint Software Systems Inc., 2016. *Blueprint.* [Online]
Available at: http://www.blueprintsys.com/
[Acedido em 02 2016].

Boehm, B. & Egyed, B., 1998. *Software requirements negotiation: some lessons learned.* Dunedin, New Zealand, s.n., pp. 503-506.

Boehm, B. W., 1991. Verifying and Validating Software Requirements and Design Specifications. *IEEE Software*, pp. 32-41.

Brinkkemper, S., 1996. Method Engineering: Engineering of Information Systems Development Methods. Volume 38, pp. 275-280.

Cannegieter, H. & Arensen, M., 2008. *Success met de Requirements!.* The Hague, Netherlands: Academic service.

Carlshamre, P., 2002. Release Planning in Market-Driven Software Product Development. Provoking an Understanding.. In: *Requirements Engineering.* 3 ed. Berlin: s.n., pp. 139-151.

Carlshamre, P., Sandahl, K., Lindvall, M. & Regnell, B., 2001. *An industrial survey of requirements interdependencies in software product release planning.* s.l., s.n., pp. 84-91.

Clegg, D. & Barker, R., 1994. *Case method fast-track: a RAD approach.* s.l.:Addison-Wesley Longman Publishing Co., Inc..

Codacy, 2016. *codacy.* [Online]
Available at: www.codacy.com
[Acedido em 20 01 2016].

Deneckere, R., Hug, C., Onderstal, J. & Brinkkemper, S., 2015. *Method Association Approach: Situational construction and evaluation of an implementation method for software products.* Athens, Greece, s.n., pp. 274-285.

Draw.io, 2015. *drawio.* [Online]
Available at: https://www.draw.io/
[Acedido em 10 2015].

Eclipse, 2015. *https://eclipse.org/.* [Online]
[Acedido em 11 2015].

Fayad, M., 2002. Accomplishing Software Stability. *Communications of the ACM*, 45(1), pp. 111-115.

Finkelstein, A. & Potts, C., 1985. *Evaluation of existing requirements extraction Strategies,* London: s.n.

Foundation for Critical Thinking, 2007. *The Thinker's Guide to Engineering Reasoning.* s.l.:s.n.

Gliffy, Inc., 2015. *Gliffy.* [Online]
Available at: https://www.gliffy.com/
[Acedido em 12 2015].

Gotel, O. C. & Finkelstein, A., 1994. *An analysis of the requirements traceability problem.* s.l., s.n., pp. 94-101.

Gottesdiener, E., 2015. *Agile Requirements and Business Analysis: Myths and Realities.* s.l., PMI Business Analysis Virtual Conference.

Günther, R. & Saliu, M. O., 2005. The art and science of software release planning. *IEEE Software*, 22(6), pp. 47-53.

Hickey, A. M. & Davis, A. M., 2004. A unified model of requirements elicitation. *Journal of Management Information Systems*, 20(4), pp. 65-84.

Inflectra Corporation, 2015. *SpiraTest.* [Online]
Available at: http://www.inflectra.com/SpiraTest/
[Acedido em 12 2015].

International Institute of Business Analysis, 2009. *A Guide to the Business AnalysisBody of Knowledge®(BABOK® Guide).* ISBN-13: 978-0-9811292-2-8 ed. Toronto: International Institute of Business Analysis.

Ionescu, P., 2015. *The 10 Most Common Application Attacks in Action.* [Online]
Available at: https://securityintelligence.com/the-10-most-common-application-attacks-in-action/

Itemis AG, 2015. *Yakindu Requirements.* [Online]
Available at: http://www.yakindu.com/requirements/
[Acedido em 12 2015].

Itemis, 2015. *http://www.yakindu.com/.* [Online].

Jive Software, 2015. *Producteev.* [Online]
Available at: https://www.producteev.com/
[Acedido em 10 2015].

Johnson, J., 2008. *The Chaos Report.* West Yarmouth, MA: The Standish Group.

Karlsson, J. & Ryan, K., 1997. A cost-value approach for prioritizing requirements. *IEEE Software*, 14(5), pp. 67-74.

Langer, A. M., 2008. System Development Life Cycle. Em: *Systems, Analysis and Design of Information.* London: Springer.

Luinenburg, L., Jansen, S., van de Weerd, I. & Brinkkemper, S., 2008. *Designing Web Content Management Systems Using the Method Association Approach.* Toulouse, France, s.n., pp. 106-120.

Marciniak, J. J. & Shumskas, A., 1994. *Acceptance Testing.* s.l.:John Wiley & Sons, inc..

Modelio Requirements, 2015. *http://archive.modeliosoft.com/en/modules/modelio-requirement-analyst.html.* [Online]
[Acedido em 12 2015].

Modelio, 2015. *Modelio Requirement Analyst.* [Online]
Available at: http://archive.modeliosoft.com/en/modules/modelio-requirement-analyst.html
[Acedido em 11 2015].

MyLin, 2015. *http://www.eclipse.org/mylyn/.* [Online]
[Acedido em 8 11 2015].

Nurmuliani, N., Zowghi, D. & Fowell, S., 2004. *Analysis of Requirements Volatility During Software Development Life Cycle.* Melbourne, Australia, s.n., pp. 28-37.

Nuseibeh, B. & Easterbrook, S., 2000. *Requirements Engineering: a Roadmap.* Limerick, Ireland, s.n., pp. 35-46.

Nuseibeh, B. & EasterBrook, S., 2000. *Requirements Engineering: a Roadmap.* Limerick, Ireland, s.n., pp. 35-46.

Open Web Application Security Project, 2015. *www.owasp.org.* [Online]
Available at: www.owasp.org
[Acedido em 01 2015].

Open Web Application Security Project, 2016. *OWASP Risk Rating Methodology.* [Online]
Available at: https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology

OpenSAMM Project, 2009. *Software Assurance Maturity Model (SAMM).* 1 ed. s.l.:s.n.

Paetsch, F., Eberlein, A. & Maurer, F., 2003. *Requirements engineering and agile software development.* Linz, Austria, s.n., pp. 308-313.

Paul Grünbacher, N. S., 2005. Engineering and Managing Software Requirements. Em: *Requirements Negotiation.* s.l.:Springer, pp. 143-162.

Project Management Institute, 2014. Requirements Management - A core competency for project and program success. *Pulse of the Profession®,* Agosto.

Project Management Institute, 2014. The High Cost of Low Performance. *Pulse of the Profession®,* Fevereiro.

Project Management Institute, 2015. *Business Analysis for practitioners: A practice guide.* Pennssylvania: Project Management Institute, Inc.

Project Management Institute, I., 2016. *Requirement Management: a practice guide.* Newtown Square, Pennsylvania: Project Management Institute, Inc..

Ralyté, J., Deneckère, R. & Rolland, C., 2003. *Towards a Generic Model for Situational Method Engineering..* Klagenfurt, Austria, s.n., pp. 95-110.

Ratchev, S. et al., 2003. Knowledge based requirement engineering for one-of-a-kind complex systems.. *Knowlege-Based systems*, 16(1), pp. 1-5.

Regnell, B., Beremark, P. & Eklundh, O., 1998. A Market-driven Requirements Engineering Process: Results From an Industrial Process Improvement Programme. Em: *Requirements Engineering.* 2 ed. Berlin, Germany: Springer-Verlag.

Regnell, B. & Brinkkemper, S., 2005. Market-driven Requirements Engineering for Software Products. *Engineering and Managing Software Requirements,* pp. 287-308.

RequirementONE Inc., 2015. *Requirement One.* [Online]
Available at: http://www.requirementone.com/
[Acedido em 10 2015].

Rolland, C. & Prakash, N., 1996. A Proposal for Context-specific Method Engineering. Em: *Method Engineering.* New York: Springer-Verlag, pp. 191-208.

Saaty, T. L., 1991. s.l., s.n.

Semmle Ltd, 2016. *semmle.com.* [Online]
Available at: semmle.com
[Acedido em 10 02 2016].

Software Freedom Conservancy, 2016. *https://git-scm.com/.* [Online]
Available at: https://git-scm.com/
[Acedido em 15 01 2016].

SonarQube, 2015. *www.sonarqube.org.* [Online]
Available at: www.sonarqube.org
[Acedido em 05 10 2015].

Sonarsource SA, 2016. *SonarQube.* [Online]
Available at: www.sonarqube.org/
[Acedido em 20 02 2016].

SQUORING Technologies, 2016. *SQuORE.* [Online]
Available at: http://www.squoring.com/en/produits/tableau-de-bord-squore/
[Acedido em 01 02 2016].

tracecloud, 2015. *tracecloud.* [Online]
Available at: www.tracecloud.com
[Acedido em 10 2015].

Van de Weerd, I., Brinkkemper, S., Souer, J. & Versendaal, J., 2006. A Situational Implementation Method for Web-based Content Management System-applications: Method Engineering and Validation in Practice. *Software Process: Improvement and Practice*, 11(5), pp. 521-538.

van Gurp, J., Bosch, J. & Svahnberg, M., 2001. *On the notion of variability in software product lines.* Amsterdam, Netherlands, s.n., pp. 45-54.

Vessey, I. & Conger, S. A., 1994. Requirements specification: learning object, process, and data methodologies. *Communications of the ACM*, 37(5), pp. 102-113.

Visual Paradigm, 2016. *https://www.visual-paradigm.com/.* [Online]
[Acedido em 02 2016].

Visual Paradigm, 2016. *Visual Paradigm Uexceler.* [Online]
Available at: http://www.visual-paradigm.com/features/uexceler-supports/
[Acedido em 02 2016].

Visure Solutions, S.L., 2016. *Visure Requirements.* [Online]
Available at: http://www.visuresolutions.com/requirements-engineering-tool
[Acedido em 01 2016].

Wiegers, K., 2015. *PMI Business Analysis Virtual Conference.* s.l., s.n.

Wiegers, K. & Beatty, J., 2013. *Software Requirements.* s.l.:Pearson Education.

WireframeSketcher, 2015. *wireframesketcher.com.* [Online]
Available at: wireframesketcher.com/
[Acedido em 12 2015].

www.phoenix.tc-ieee.org, 2016. *Methodology and Systems Management.* [Online]
Available at: http://www.phoenix.tc-ieee.org/015_Methodology_and_Systems_Management/SwE_workflow.html
[Acedido em 15 05 2016].

yWorks GmbH, 2015. *yWorks.* [Online]
Available at: https://www.yworks.com/
[Acedido em 11 2015].

isep Instituto Superior de Engenharia do Porto

Instituto Superior de Engenharia do Porto ⊙ Rua Dr. António Bernardino de Almeida, 431 ⊙ 4249-015 Porto
Tel. +351 228 340 500 ⊙ Fax +351 228 321 159 ⊙ mail@isep.ipp.pt ⊙ www.isep.ipp.pt

INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

isep

Business Analysis Process

BRUNO MIGUEL FERNANDES SILVA
Julho de 2016

INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

MESTRADO EM ENGENHARIA INFORMÁTICA

isep




Business Analysis Process

BRUNO MIGUEL FERNANDES SILVA
Julho de 2016

POLITÉCNICO
DO PORTO