FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Análise de Impacto das Alterações a Processos Descritos em BPMN

José Pedro Teles

Mestrado Integrado em Engenharia Informática e Computação

Orientador: Ana Cristina Ramada Paiva

26 de Fevereiro de 2019

Análise de Impacto das Alterações a Processos Descritos em BPMN

José Pedro Teles

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: João Carlos Pascoal Faria
Arguente: Nome do arguente do júri
Orientador: Ana Cristina Ramada Paiva

---

26 de Fevereiro de 2019

Resumo

A necessidade que dita a existência de uma enorme quantidade de soluções informáticas complexas, exige igualmente a necessidade de uma verificação a todas as suas atividades e tarefas durante atualizações. Uma das formas de realizar esta verificação é a realização da análise de impacto ao *software*, detetando implicações no *software* que possam inviabilizar o uso destas soluções, evitando assim problemas aos clientes. Contudo, a análise de impacto manual é um processo lento e, na maior parte dos casos, não consegue cobrir toda a amplitude de um projeto.

A Engenharia de Software mudou este paradigma com a introdução de metodologias de análise de impacto automático. De forma a gerir e simplificar estes processos, foram desenvolvidas metodologias e plataformas de análise de impacto.

O objectivo deste trabalho é criar uma plataforma que permita a conceção de mapas de análise de impacto de alterações, facilitando a rastreabilidade do processo de negócio. Para tal, definiu-se um conjunto de relações entre *Business Process Model and Notation* (BPMN) e artefactos, desenvolvidos nas diferentes etapas de desenvolvimento de *software*. Tendo em conta estas relações, desenvolveu-se uma plataforma que permite criar de uma forma automática estas relações e que, com um segundo modelo atualizado, realize uma análise de impacto. O resultado desta análise é apresentado de uma forma simples e clara ao utilizador de modo a conhecer as implicações das alterações.

Como trabalho futuro, esta plataforma poderia ser melhorada ao nível do aumento das relações com novos artefactos, assim como a criação de vistas onde fosse possível abri-los, permitindo uma visão mais alargada da análise de impacto.

Abstract

Nowadays, there is a need for an enormous amount of complex computer solutions, which demands the verification of all its activities and tasks. One of the ways to do this verification is to perform impact analysis to detect the implication of changes that may turn these solutions unfeasible, avoiding problems for customers. However, manual impact analysis is a slow process and, in most cases, cannot cover the full breadth of a project.

Software Engineering changed this paradigm with the introduction of automatic impact analysis. In order to manage and simplify these processes, management platforms and impact analysis have emerged.

Our goal is to create a platform which enables the generation of change impact analysis maps, to facilitate traceability of the business process. In order to do so, a set of relations between Business Process Model and Notation (BPMN) and artifacts, developed during the diferente stages of software development, was generated. Regarding these relations, a platform that enables the creation of these relations, in an automatic way, was created and a second model does the impact analysis. The results of this analysis are presented in a simple and clear form to the end user in order for him to know the implications of these changes.

Future work would involve further improvements of the solution at the level of creating new relations with new artifacts, as well as creating views where it would be possible to open these artifacts, enabling a larger view of the impact analysis.

Agradecimentos

Gostaria de agradecer à Professora Ana Paiva por todo o apoio que me prestou ao longo da realização desta dissertação. Quero agradecer também ao Professor Nuno Flores pela ajuda nesta fase final.

Agradeço a todos os meus amigos, que me acompanharam ao longo destes anos de universidade. Todas as sessões de estudo, trabalhos e de diversão que tivemos juntos. Vocês tornaram o meu percurso académico numa das melhores experiências da minha vida.

Gostaria de agradecer também aos meus pais por todo o esforço que fizeram para que eu pudesse chegar aqui e, ao meu irmão que tantas vezes me deu na cabeça para eu dar o meu melhor em tudo o que faço sem nunca desistir.

Por último, um obrigado especial à minha avó, que com as perguntas e dúvidas que fazia, me fez procurar saber sempre mais. Sem a sua sede de conhecimento não teria ganho o gosto que tenho em aprender sempre mais.

José Pedro Teles

"Nothing is more powerful than an idea whose time has come"

Victor Hugo

Conteúdo

1 Introdução **1**
- 1.1 Contexto . . . 1
- 1.2 Motivação e Objetivos . . . 2
- 1.3 Problema . . . 2
  - 1.3.1 Proposta . . . 2
- 1.4 Estrutura da Dissertação . . . 3

2 Revisão Bibliográfica **5**
- 2.1 Engenharia de Software . . . 5
  - 2.1.1 Testes de software . . . 5
  - 2.1.2 Modelação . . . 6
- 2.2 BPMN - Business Process Model and Notation . . . 7
  - 2.2.1 Introdução . . . 7
  - 2.2.2 Notação BPMN . . . 7
- 2.3 Testes baseados em Modelos . . . 8
  - 2.3.1 Introdução . . . 8
  - 2.3.2 Processos . . . 9
  - 2.3.3 Skyfire: MBT com Cucumber . . . 9
  - 2.3.4 Smartesting CertifyIT, MBT em projetos de grande escala . . . 10
  - 2.3.5 Pattern Based GUI Testing . . . 11
  - 2.3.6 Sepc Explorer . . . 12
  - 2.3.7 ModelJUnit . . . 12
  - 2.3.8 Comparação das ferramentas . . . 12
- 2.4 Análise de Impacto . . . 12
  - 2.4.1 Rastreabilidade de artefactos MBT . . . 13
  - 2.4.2 Rastreabilidade em processos de negócio . . . 14
- 2.5 ETA-Pro - End-to-end Test Automation Platform for Processes . . . 14
  - 2.5.1 Introdução . . . 14
  - 2.5.2 Plataforma . . . 15
  - 2.5.3 MBT no ETAP-Pro . . . 16
- 2.6 Conclusões . . . 16

3 Metodologia **19**
- 3.1 Introdução . . . 19
- 3.2 Meta Modelo . . . 19
- 3.3 Abordagem . . . 22
  - 3.3.1 Outros artefactos . . . 23
  - 3.3.2 Relação entre BPMN e artefactos . . . 27

3.4 Implementação 28
3.4.1 Visão Geral 28
3.4.2 Casos de uso 28
3.4.3 Protótipo da plataforma 29
3.4.4 Arquitetura 31
3.4.5 Tecnologias de desenvolvimento 33

4 Validação **35**
4.1 Caso base 35
4.2 Cenário 1 - alteração de uma tarefa 41
4.2.1 Resultados 43
4.3 Cenário 2 - eliminação de uma tarefa 43
4.3.1 Resultados 45
4.4 Cenário 3 - criação de uma tarefa 46
4.4.1 Resultados 46
4.5 Conclusões 47

5 Conclusões e Trabalho Futuro **49**
5.1 Considerações finais 49
5.2 Dificuldades 50
5.3 Trabalho Futuro 50

Referências **51**

A *Code snippet* **55**

Lista de Figuras

2.1 Elementos gerais da notação BPMN. Adaptado de [NF12] 8
2.2 Arquitetura Skyfire, retirado de [LEK16] 10
2.3 Processo de MBT com *Smartesting*. Retirado de [LB13] 11
2.4 Processo de MBT com *Smartesting*. Retirado de [LB13] 13
2.5 Processos de MBT no ETAP-Pro. Retirado de [PFFM18] 16

3.1 Metamodelo com as relações entre atividades de BPMN e artefactos relativos a um programa. 21
3.2 Exemplo de informação relativa a atores. 23
3.3 Exemplo de informação relativa a *user stories* 23
3.4 Exemplo de informação relativa a casos de teste 24
3.5 Exemplo de informação relativa a um diagrama de fluxo 24
3.6 Exemplo de informação relativa a um diagrama de classes 25
3.7 Casos de uso da plataforma 29
3.8 Página base da plataforma 29
3.9 Página com informação relativa ao BPMN e artefactos 30
3.10 Página com informação de análise de impacto 30
3.11 Arquitetura da plataforma 32

4.1 BPMN representativo de um processo de desenvolvimento de software 36
4.2 Exemplo de informação relativa a *user stories* 37
4.3 Exemplo de informação relativa a casos de teste 38
4.4 Exemplo de informação relativa a casos de teste da realização de comentários . . 39
4.5 Exemplo de informação relativa as classes afetas ao engenheiro de requisitos . . 39
4.6 Exemplo de informação relativa as classes afetas ao desenvolvimento do SRS . . 40
4.7 Exemplo de alteração de uma tarefa 42
4.8 Resultado da análise de impacto de alteração de uma tarefa 43
4.9 BPMN com remoção de uma tarefa 44
4.10 Exemplo de eliminação de uma tarefa 45
4.11 Exemplo de criação de uma tarefa 46
4.12 Resultado da análise de impacto de adição de uma tarefa 47

Abreviaturas e Símbolos

BPMN	Business Process Model and Notation
CSV	Comma-separated values
CSS	Cascading Style Sheets
EMF	Eclipse Modeling Framework
ETAP-PRO	End-to-end Test Automation Platform for Processes
EFSM	Extended FSM
FSM	Finite State Machine
HTML	Hyper Text Markup Language
ID	Identificador
ISTQB	International Software Testing Qualifications Board
JS	JavaScript
MBT	Model Based Testing
MVC	Model-View-Controller
PBGT	Pattern Base Gui Testing
QA	Quality and assurance
SRS	Software Requirements Specification
SUT	System Under Test
UITP	User Interface Test Patterns
UML	Unified Modeling Language
US	User Stories
XML	eXtensible Markup Language
XPDL	XML Process Definition Language
XSD	XML Schema Definition

Capítulo 1

Introdução

Esta dissertação insere-se na área de engenharia de software, mais especificamente na análise de impacto de alterações efetuadas a processos de negócio. Os testes a processos de negócio têm como grande objetivo testar um conjunto de atividades estruturadas que produzem um serviço ou que servem um objetivo particular. Isto é conseguido através da realização de testes ponta a ponta ao sistema. Este capítulo introdutório apresenta o contexto do problema que vai ser abordado, dando também um breve contexto técnico, a motivação e o objetivo principal a ser alcançado com esta dissertação. A última secção dá um panorama geral da estrutura do documento.

1.1 Contexto

Numa altura em que existe um aumento do número de soluções informáticas complexas e cada vez mais importantes no nosso quotidiano, torna-se necessário assegurar a qualidade das mesmas. Por isso, o teste de software tornou-se numa componente vital da Engenharia de Software [Cai11]. Testar manualmente todas as hipóteses e todos os cenários destes sistemas é um processo desafiante e com um elevado consumo de tempo.

Com a necessidade de automação dos testes para estes sistemas, surgiu o teste a processos de negócios. Esta nova abordagem visa validar todas as possíveis transições de ponta a ponta destes sistemas. Um dos procedimentos para a realização destes testes é o teste baseado em modelos. Um modelo de notação bastante utilizado na modelação de processos de negócios é o *Business Process Modeling and Notation* (BPMN). Esta notação permite uma fácil perceção interna do sistema de uma forma gráfica. Esta visualização gráfica do sistema permite também uma melhor comunicação entre as equipas de desenvolvimento e a parte interessada no produto final, *stakeholders*.

A plataforma End-to-end Test Automation Platform for Processes (ETAP-Pro) é uma plataforma que permite a criação automática de testes de ponta a ponta a partir de modelos BPMN.

Para além da criação destes testes, permite também gerir de forma integrada a automação de processos de negócio complexos, que assentam em diversos sistemas, aplicações e serviços existentes numa organização.

1.2 Motivação e Objetivos

A existência do aumento de eventuais alterações das soluções informáticas, torna essencial que estas possam ser adaptadas e atualizadas de uma maneira rápida e acessível. Por isso, um dos grandes desafios dos sistemas atuais é mantê-los robustos e operacionais de uma forma contínua [SSS06]. Para tal, é necessário assegurar que as plataformas de testes existentes tenham as ferramentas necessárias para irem ao encontro destas necessidades.

Esta dissertação tem como objetivo principal criar uma ferramenta que permita analisar o impacto das alterações nos processos de negócio. Esta análise tem como base a informação de rastreabilidade que vá permitir conhecer o impacto que alterações a processos de negócios irão ter nos diversos artefactos envolvidos (modelos, testes, relações de dependências, documentos, requisitos, entre outros) num processo de desenvolvimento de software.

1.3 Problema

O número de soluções informáticas tem vindo a aumentar assim como a sua complexidade e número de atualizações. Devido a estes aumentos, é necessário manter um registo de rastreabilidade que contenha toda a informação necessária à interpretação da solução, assim como do impacto que as modificações possam vir a ter sobre o produto. A dificuldade é encontrar uma ferramenta que faça isto a partir de ficheiros BPMN que representam a solução. Esta dificuldade existe, porque não há uma correlação entre o ficheiro BPMN e artefactos gerados durante o levantamento de requisitos, tornando a rastreabilidade manual destes ficheiros um processo muito exigente a nível de recursos humanos e financeiros.

1.3.1 Proposta

Existe a necessidade de ferramentas capazes de criar relações entre um BPMN e artefactos do mesmo projeto e que permitam um fácil entendimento do projeto e das implicações de modificações.

Com isto em vista, o objetivo deste projeto é encontrar uma solução para este problema. Assim, criou-se um conjunto de relações entre os diferentes artefactos de um BPMN com artefactos criados durante o processo de desenvolvimento de software. Deste modo, a ferramenta permite:

- retirar informações de um ficheiro BPMN relativas a participantes, a processos, a tarefas e ao fluxo de tarefas;
- retirar informações de diagramas, *user stories* (US), casos de testes, código fonte, entre outros, relativas a diagramas de classes, casos de uso e casos de testes;

- comparar dois BPMN e analisar o impacto de modificações nos artefactos;
- visualizar este impacto.

1.4 Estrutura da Dissertação

Para além da introdução, esta dissertação contém mais 4 capítulos.

No capítulo 2, é realizada uma revisão bibliográfica de engenharia de software, com especial foco em área de testes de software, modelação, notação BPMN, testes baseados em modelos e respetivas ferramentas e como análise de impacto. Neste capítulo, também é abordado a ferramenta ETAP-Pro.

No capítulo 3, é abordada a metodologia seguida para solucionar o problema.

No capítulo 4, é abordada a validação da contribuição realizada.

No capítulo 5, é apresentada a conclusão deste trabalho e sugestões de trabalho futuro.

Capítulo 2

Revisão Bibliográfica

Neste capítulo serão abordados alguns conceitos de Engenharia de Software imprescindíveis para esta dissertação (testes de software, modelação de sistemas informáticos, rastreabilidade e análise de impacto de alterações).

Neste capítulo também serão abordados conceitos relacionados com a notação BPMN. É necessário compreender os seus detalhes de notação de modo a compreender como esta pode ser aplicada nos testes a processos de negócio.

Neste capítulo vão ser abordados conceitos relacionados com testes a processos de negócio como Model-Based Testing (MBT), nomeadamente no que diz respeito a processos gerais, definições e limitações. São analisadas algumas metodologias de MBT relevantes para o estudo e também a análise de impacto quando ocorrem alterações nos modelos e consequentemente nos testes a serem feitos.

Este capítulo aborda também o estudo da plataforma ETAP-Pro que serve como modelo comparativo de criação e análise de testes de processos de negócio.

2.1 Engenharia de Software

A Engenharia de Software tornou-se numa componente essencial para o desenvolvimento de software. Com o seu desenvolvimento, a criação de software mudou e hoje em dia a criação de soluções informáticas segue um conjunto de princípios inicialmente utilizados na engenharia. Estes princípios trouxeram estrutura e disciplina para o desenvolvimento de software o que levou a um aumento da sua qualidade e eficácia. De entre as várias atividades que surgiram, o teste de software e a análise de impacto são as que apresentam uma maior relevância para esta dissertação.

2.1.1 Testes de software

Sistemas informáticos são uma realidade do nosso dia a dia, desde os telemóveis com as suas aplicações a sistemas altamente complexos e críticos, como software de aplicação hospitalar.

A dependência destes sistemas exigiu um aumento no seu controlo de qualidade e fiabilidade. Falhas nestes sistemas podem provocar sérias consequências que podem ser evitadas com o teste de software. O teste de software tem como objetivo principal assegurar a qualidade do produto final. Consoante o sistema, existe um conjunto de abordagens para definir esta qualidade. Para tal, são realizados um conjunto de testes que visam verificar o comportamento de soluções informáticas no maior número de casos possíveis. Com estes testes, a equipa de desenvolvimento poderá encontrar erros e deste modo corrigi-los. Permite também verificar se a solução desenvolvida, corresponde aos requisitos pedidos pelos stakeholders e a usabilidade dos mesmos.

2.1.1.1 Metodologias de Testes

Ao longo do ciclo de desenvolvimento de programas existem várias fases: o levantamento de requisitos, o design de software, a implementação, o teste e o lançamento do produto final. Nestas fases, existem diferentes tipos de testes: testes unitários, testes de integração, testes do sistema e testes de aceitação [AO08]. Os testes unitários correspondem ao teste individual de unidades ou grupos de unidades. Normalmente são desenvolvidos pela equipa de desenvolvimento para assegurar que as *features* funcionam corretamente, procurando falhas funcionais. Testes de integração averiguam se dois módulos interagem corretamente entre si. Teste de sistema corresponde ao teste do sistema completo de modo a verificar se cumpre com os requisitos especificados. Normalmente são realizados por uma equipa externa à equipa de desenvolvimento, QA (*quality and assurance*). Por fim, os testes de aceitação são realizados com os clientes ou *stakeholders* envolvidos no projeto. São utilizados para determinar se o sistema cumpre os critérios de aceitação do utilizador final do sistema.

2.1.1.2 Técnicas de conceção de testes

O design de técnicas de testes é utilizado para se encontrar um conjunto de casos de testes que cumprem os critérios de cobertura. Considerando que a cobertura de todas as possibilidades do sistema a testar (*System Under Test*, SUT) é difícil de alcançar, é preciso escrever o menor número de testes com a maior cobertura possível. Para se atingir estes objetivos existe um conjunto de técnicas de design que permitem a redução da quantidade de testes redundantes: *black box*, testes funcionais e *white box*, testes estruturais. Os testes funcionais focam-se na capacidade do programa realizar um conjunto de ações com base nos requisitos funcionais [Luo01]. Com este teste é possível verificar erros comportamentais do sistema como funcionalidades em falta ou incorretas, erros na interface e problemas de performance. Ao contrário dos testes funcionais, os testes estruturais focam-se em erros no código [Luo01]. Este tipo de testes verifica o código e a estrutura interna do programa.

2.1.2 Modelação

Modelos são utilizados para entender, especificar e desenvolver sistemas [VCG+08]. Na engenharia de software, são a descrição de sequência de atividades de um sistema informático. Mo-

delos são uma visão simplificada dos processos de software, sendo utilizados para exprimir de uma forma abstrata o comportamento de soluções informáticas ou, para representar estratégias de testes e ambientes de teste. A modelação de software traz um conjunto de vantagens nomeadamente, a fácil compreensão da implementação, a sua ampla utilização, a identificação de marcos importantes e o bom funcionamento para projetos de elevada complexidade e maturidade.

2.2 BPMN - Business Process Model and Notation

2.2.1 Introdução

BPMN é uma notação orientada a processo de negócios, desenvolvida pela BPMN e é agora mantida e gerida pelo *Object Management Group*. Foi desenvolvida com o propósito de abordar as necessidades da modelação de processos de negócios, tornando-se desde então numa referência neste tipo de modelação [PFFM18].

Esta permite a definição de modelos formais que expressam um certo grau de abstração e processos executáveis que visam todos os aspetos de processos de negócios [Wan06]. É baseada em fluxogramas, que podem ser bastante expressivos quanto às suas tarefas, eventos, subprocessos e condições. Permite também documentar o fluxo de dados em processos específicos através dos seus vários intervenientes.

2.2.2 Notação BPMN

A notação apresenta, para estes objetivos, um conjunto de elementos gráficos que a tornam mais intuitiva para todas pessoas envolvidas no projeto. Estes elementos podem ser divididos em quatro categorias [Wan06], como está demonstrado na figura 2.1.

Na primeira categoria inserem-se os objetos de fluxo. Estes objetos fazem referência a elementos que se conectam e formam um fluxo de processo. Fazem parte deles eventos, atividades e decisões. Os eventos são acontecimentos que podem ter impacto num processo de negócio, podem ser externos ou internos e são representados por um círculo. As atividades são uma forma representativa mais abstrata de ações que ocorrem em processos de negócios e são um de dois tipos de tarefas ou de subprocessos. As tarefas são funcionalidades que não podem ser subdivididas em partes mais pequenas ao passo que os subprocessos são funcionalidades que podem ser simplificadas até outro nível de detalhe, sendo representados por retângulos. As decisões são responsáveis por controlar o fluxo do processo de negócio. São representadas por losangos e num processo o resultado pode variar consoante certos fatores. É aí que as decisões entram, alterando o resultado caso existam decisões a tomar.

Outra categoria são os objetos de conexão, utilizados para conectar, mostrar mensagens trocadas ou associar elementos de fluxo. Graficamente, são representados por setas. Existem quatro tipos de elementos de fluxos: de sequência, de mensagem, associação de informação e associação. O primeiro conecta elementos de fluxo e mostra a sua ordem, sendo representados por setas

pretas. O fluxo de mensagem mostra as mensagens de comunicação entre elementos e são representados por uma seta a tracejado. Uma associação de informação é utilizada para mostrar o fluxo de informação entre atividades. Uma associação liga dois artefactos.

A terceira categoria são os artefactos. Estes são utilizados para modelar dados. Para isso utilizam objetos de dados que representam dados produzidos durante ou no fim dos processos. Artefactos também podem ser grupos que permitem agrupar atividades ou anotações de texto que servem para dar detalhes extra ao fluxo de objetos.

A última categoria são as *swimlanes*. Estes objetos são utilizados para representar os diferentes intervenientes num processo e deste modo organizar melhor o diagrama. Esta categoria tem dois objetos: as *pools*, que representam diferentes entidades num processos e as *lanes*, que são usadas para dividir as *pools*, por departamentos ou grupo de pessoas afetos a esses processos.

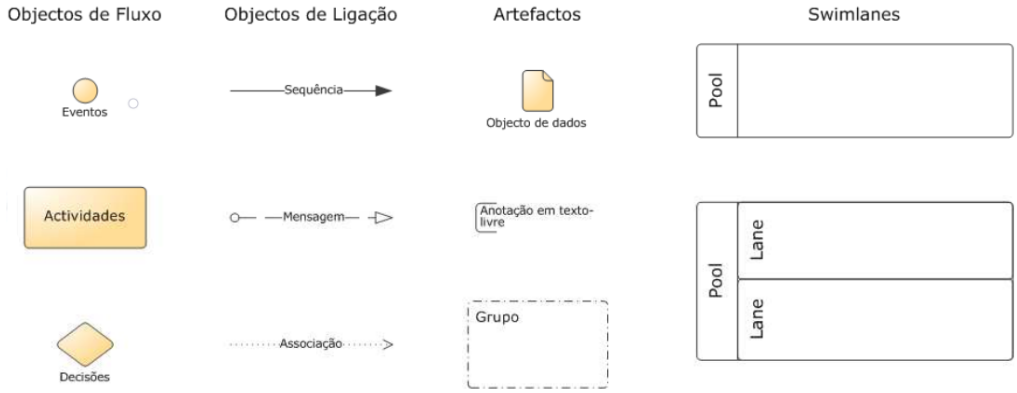

Figura 2.1: Elementos gerais da notação BPMN. Adaptado de [NF12]

Apesar de todos estes objetos, a notação BPMN ainda apresenta alguns aspetos negativos: a grande variedade de tipos de evento, a ausência de conceito de estado, a imposição de uma expressão externa de organizações em *swimlanes*, o que dificulta estruturas organizacionais independentes da descrição do processo e a existência de conceitos com definições ambíguas [RZ14].

2.3 Testes baseados em Modelos

2.3.1 Introdução

Hoje em dia as empresas enfrentam desafios devido ao aumento da complexidade dos negócios, das exigências dos *stakeholders*, da evolução tecnológica e das soluções informáticas [PFFM18] Por isso, um desafio constante é a manutenção da robustez e qualidade dos processos de negócio aquando da implementação e da evolução dos mesmos [SSS06]. Uma das formas de ultrapassar esse desafio é através de testes, mais concretamente testes baseados em modelos.

Como foi abordado na secção de modelação, modelos permitem ter uma visão abstrata do sistema e podem ser aplicados em qualquer parte do ciclo de vida dos mesmos, desde levantamento de requisitos, escrita de código à geração de testes [SSS06].

Permite também determinar se os testes passam ou se ocorrem falhas. De acordo com o *International Software Testing Qualifications Board* (ISTQB), a qualidade e a eficiência dos testes passam por realizar um modelo compreensivo utilizando ferramentas com base nos objetivos do projeto e fornecendo um modelo com especificação de testes. Este modelo tem de incluir um grande número de informação formal que permita gerar casos de testes automáticos.

2.3.2 Processos

De acordo com o ISTQB [Int11] os processos fundamentais para testes baseados em modelos MBT durante o ciclo de produção de software devem seguir certos passos de forma iterativa e incremental. Em primeiro lugar, a conceção do modelo de testes para o SUT deve ser feita a partir dos requisitos ou então de especificações, devendo delinear que testes vão ser realizados e com que métricas. De seguida, geração e gestão de testes, seguidos por implementação e execução de testes criados para o efeito. De seguida, avaliação de resultado dos testes, incluindo as métricas de cobertura, terminando com a criação de bibliotecas que possam ser reutilizadas em projetos futuros ou aquando da evolução do projeto.

Seguir estes passos permite uma melhor qualidade de testes sendo mais eficaz porque modelar é um processo que requer comunicação com o cliente, modelos gráficos que permitem uma melhor colaboração com os clientes, modelos que permitem adaptações a qualquer momento, o nível de abstração que permite uma fácil identificação das áreas mais problemáticas e a criação e análise de casos de testes antes da conceção do sistema. É também mais eficiente uma vez que a modelação nos estágios iniciais permite evitar erros nos momentos iniciais do desenvolvimento, uma vez que os clientes podem ter acesso e dar a sua opinião sobre o mesmo. Modelos de projetos anteriores podem ser utilizados para desenvolvimento de testes, suportando a automatização na criação de testes e reduzindo os erros do teste manual. Diferentes testes podem ser gerados a partir do mesmo modelo sendo adaptados mais eficazmente e podem ser também utilizados para cobrir diferentes níveis e tipos de testes durante a fase de testes. Por fim, reduzem os custos da manutenção aquando da alteração dos requisitos.

Ao mesmo tempo, existem limitações e desvantagens neste tipo de testes. MBT não substitui o ciclo completo de testes e são uma extensão não sendo substitutos. Requerem também ferramentas de modo a produzirem testes com impacto para a avaliação dos sistemas. Se um modelo não for corretamente elaborado os casos de testes vão dar resultados incorretos. Por último, quando são aplicados casos de testes combinatórios, é gerado um elevado número de casos de testes, algo que não se pretende.

2.3.3 Skyfire: MBT com Cucumber

Skyfire é uma extensão criada com objetivo de tornar o *Cucumber* numa plataforma que consiga realizar MBT [LEK16]. Estes testes vão ser criados em duas fases distintas. Numa primeira fase realizada no *Skyfire*, como se pode ver na figura 2.2 o modelo vai ser interpretado. Todos os elementos necessários para a realização de testes vão ser identificados. Esta fase é seguida

pela conversão do modelo criado na *Eclipse Modeling Framework* (EMF) num gráfico mais abstrato, que inclui uma secção com os critérios de cobertura. Depois são criados testes abstratos, que consistem em estados, transições e condições através do mapeamento dos casos de teste do EMF-based. Por fim, existe a conversão dos testes abstratos em cenários de teste do *Cucumber*. Na segunda e última fase, os testes criados no *Skyfire* são lidos e o *Cucumber* gera os testes propriamente ditos. Os testes são executados no *Cucumber* assim como a exibição dos resultados.

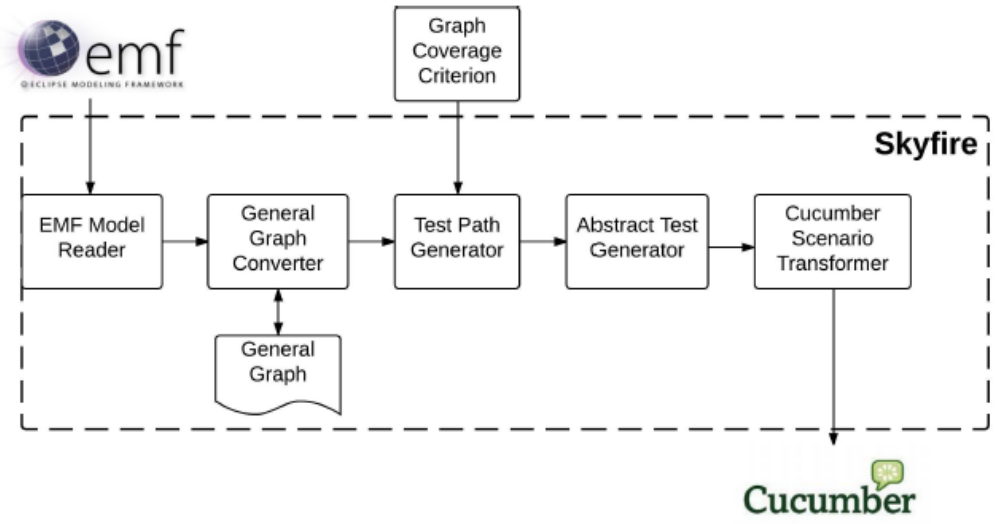

Figura 2.2: Arquitetura Skyfire, retirado de [LEK16]

2.3.4 Smartesting CertifyIT, MBT em projetos de grande escala

Smartesting CertifyIT é utilizada atualmente para o teste de projetos de grande escala que têm um grande volume de dados e um sistema orientado a processos [LB13]. Como tal, os processos utilizados para MBT tiveram que ser adaptados para ir de encontro às seguintes necessidades:

- dar suporte tanto ao fluxo de negócio como às regras de negócio;
- permitir a geração automática e completa de testes e também executáveis;
- dar suporte a testes automáticos e manuais;
- permitir a reutilização dos modelos e testes;
- permitir a realização de testes funcionais;
- integração com o ciclo de testes.

A imagem 2.3 apresenta os processos de MBT integrados com o *Smartesting*. Inicialmente, existe a criação dos modelos de teste, seguindo-se a seleção dos critérios de cobertura. O processo seguinte é a criação automática de testes e a sua colocação em ferramentas de teste. Os testes são então executados nesta mesma ferramenta. No fim, existe a manutenção dos testes aquando da

evolução do projeto. Como é possível verificar nesta metodologia apresentam apenas diretrizes para os testes, não sugerindo nenhuma ferramenta para os mesmos.

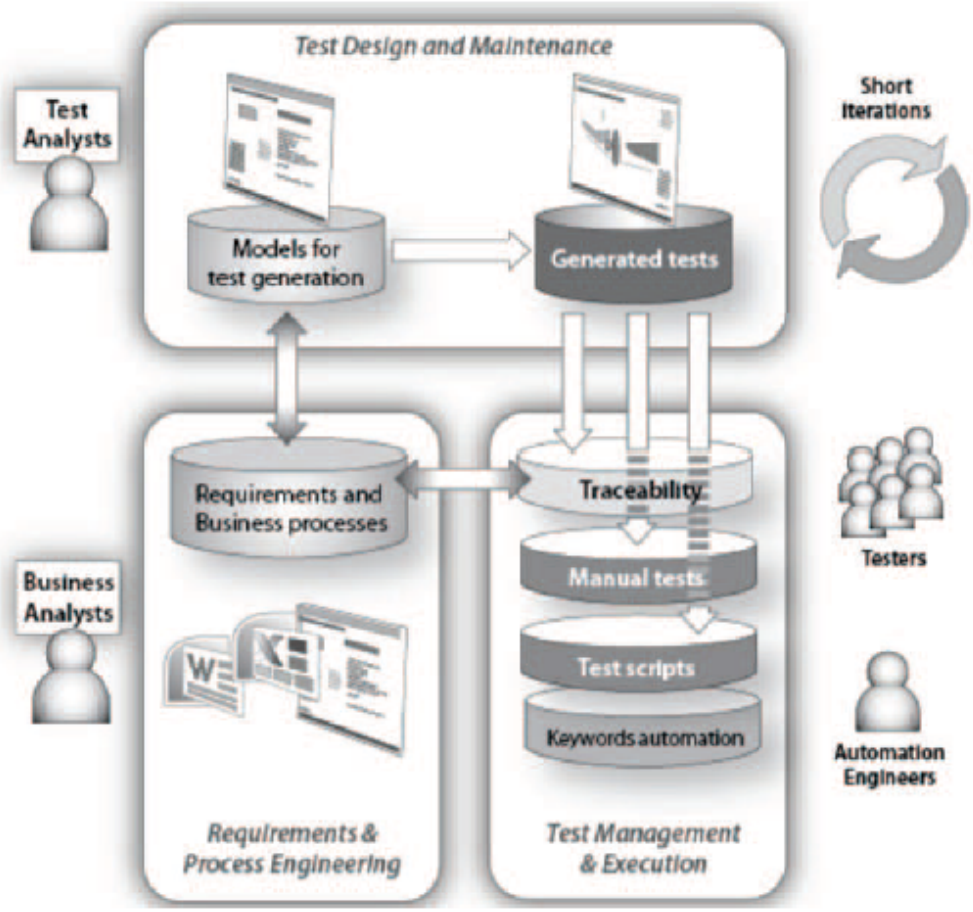

Figura 2.3: Processo de MBT com *Smartesting*. Retirado de [LB13]

2.3.5 Pattern Based GUI Testing

Outra metodologia de MBT é baseada em *Pattern Based GUI Testing* (PBGT) [NP14]. Esta abordagem surgiu devido ao aumento da importância das interfaces nos dias de hoje [MP14a]. Estes modelos apresentam como vantagens a reutilização e custos reduzidos, esforço reduzido, baixa manutenção, liberdade nos dados de teste e suporte de ferramentas [MPNM17]. Estes modelos são escritos numa linguagem específica para o domínio, PARADIGM, e são compostos por *User Interface Test Patterns* (UITP) [PV17]. Esta linguagem apresenta elementos e conectores, os elementos podem ser do tipo *Init* (representam o inicio do modelo), *End* (representam o fim do modelo). Os elementos estruturais, permitem a estruturação do modelo em diferentes níveis de abstração e comportamento, que descrevem o comportamento de teste. Por fim, os conectores fazem a conexão dos elementos [MP13]. O padrão tem como objetivo aumentar o nível de abstração dos modelos de teste e aumentar a sua reutilização. [MP14b] Esta abordagem usa o PARADIGM-ME, uma ferramenta de modelação e testes que permite a criação de modelos de

testes [RMLMM13]. Após a criação dos modelos de teste, a ferramenta vai calcular todos os caminhos existentes nesse grafo. Após o cálculo dos caminhos é implementado um filtro que vai restringir o número de caminhos a testar. O filtro é baseado em 3 condições: ciclos que podem ser restringidos no número de vezes que são executados, elementos obrigatórios a testar e a aleatoriedade. Depois desta filtragem, os caminhos são decompostos em vários casos de teste, que também vão sofrer uma filtragem. As métricas desta filtragem são configurações específicas, fornecidas pela equipa de testes e pela aleatoriedade, dado que um número aleatório de testes são escolhidos para serem executados. Por fim, permite também a exportação da informação dos testes para um ficheiro *Extensible Markup Languag*e (XML).

2.3.6 Sepc Explorer

Esta ferramenta, desenvolvida pela Microsoft, é uma ferramenta de uso não comercial utilizada para MBT. A sua principal funcionalidade é fornecer uma ferramenta integrada com desenvolvimento, exploração e validação de modelos, com geração de testes a partir desses modelos e, por fim, que os execute e permita visualizar os resultados dos mesmos. Esta ferramenta pega no modelo do sistema e cria um diagrama de máquina de estados. Através da exploração de todos os estados, os testes são então gerados. Para além disto, esta ferramenta permite a exportação das transições entre os estados e os testes gerados para um ficheiro XML.

2.3.7 ModelJUnit

ModelJUnit é uma biblioteca de Java que estende o Junit para suportar testes baseados em modelos. Nesta ferramenta, os modelos estão escritos em Java num formato de *Finite State Machine* (FSM) ou *Extended FSM* (EFSM). A partir destes modelos, os testes são criados através de um de 3 algoritmos disponíveis: *Random walk*, *Greedy walk* ou, *Look ahead walk*. Como os modelos estão escritos em Java, é necessário incluir Modeljunit.jar e junit.jar no *path* da classe.

2.3.8 Comparação das ferramentas

Através da utilização e de pesquisa destas duas ferramentas é possível fazer algumas comparações entre as duas. Ambas as ferramentas permitem a exportação das informações geradas para ficheiros XML, não permitem simulação de modelos e permitem a geração de testes de uma forma automática. Relativamente à usabilidade, ambas apresentam uma GUI de fácil compreensão e têm também uma manutenção dos testes. A principal diferença está na eficiência uma vez que a ferramenta da Microsoft apresenta uma eficiência melhor em relação à de ModelJUnit.

2.4 Análise de Impacto

A análise de impacto pode ser utilizada em situações de planeamento de mudanças, sua realização, rastreamento de mudanças e consequentes efeitos [OAH03]. Assim, serve para determinar as alterações que a evolução natural do software, possam vir a ter ao que já foi desenvolvido até

então. Relativamente à análise de impacto em MBT, não basta estar apenas atento aos diagramas, sendo também necessário dar atenção aos casos de testes e à informação de rastreabilidade entre os diferentes artefactos. Com as novas alterações, estes casos de testes podem-se tornar obsoletos, complemente novos ou podem ficar iguais. De acordo com E. Erlikh, em Leveraging legacy system dollars for E-business, 85-90% do orçamento é utilizado em operações e em manutenção [GKVS14]. Por isso, é de todo relevante realizar análise de impacto de modo a reduzir este custo.

Relativo à análise de impacto nos diagramas, Knethen e Grund dizem que numa fase inicial, os elementos que vão ser influenciados pelas alterações são identificados [KG03]. Após esta identificação, são divididos em 3 patamares: impacto primário, impacto secundário e impacto terciário. No primeiro patamar, são colocados os elementos que vão ser alterados; no segundo os que tem grande probabilidade de serem alterados e por fim, os que podem vir a ser alterados. Fica ao critério pessoal o que corresponde a probabilidade elevada e como esta é determinada.

L.C. Briand e equipa apresentam uma metodologia mais elaborada no sentido em que propõem um conjunto de passos que permitem uma análise completa e cuidada [BLOS06]. Inicialmente é verificada a consistência das alterações dos diagramas. Esta análise é feita através de ferramentas que apliquem regras da OMG (*Object Management Group*) [OMG17]. A deteção de alterações é também realizada através de ferramentas próprias para o efeito. A realização da análise de impacto é feita seguindo 97 regras que vão determinar se os elementos do modelo vão ser influenciados direta ou indiretamente [BLOS06]. A figura 2.4 mostra o modelo conceptual deste passo. Por fim a priorização dos resultados é feita através da distância entre os elementos alterados e o potencial elemento a ser influenciado, assumindo que quanto maior a distância, menos provável é o elemento ser influenciado.

Relacionado com testes baseados em modelos, Briand desenvolveu um método grosseiro de análise de impacto que classifica os testes em obsoletos ou reutilizáveis, com base num modelo *Unified Modeling Language* (UML) com classes e diagramas de sequências [BLOS06].

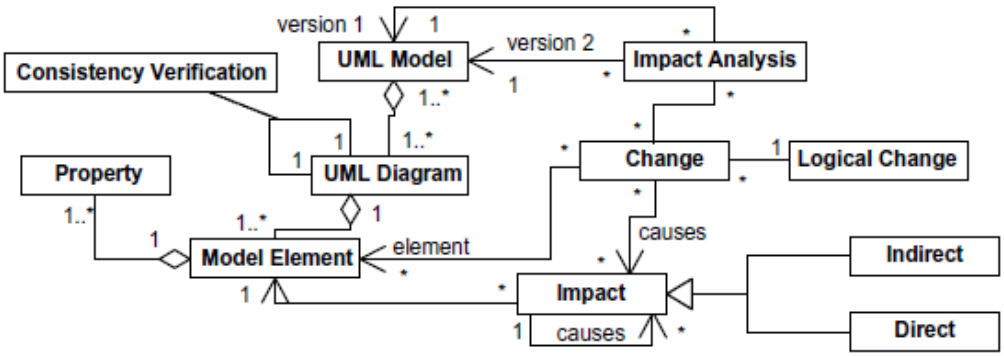

Figura 2.4: Processo de MBT com *Smartesting*. Retirado de [LB13]

2.4.1 Rastreabilidade de artefactos MBT

Esta abordagem apresenta quatro passos distintos com o intuito de criar a rastreabilidade dos artefactos dos testes baseados em modelos [NZR07]. No passo inicial, os autores utilizam um

plugin baseado na framework de modelação do Eclipse para a criação de diagramas de sequência em UML 2.0 Metamodel. De seguida, existe a criação de um modelo de controlo de fluxo, através da criação de um conjunto de caminhos que alcançam uma cobertura especificada. Estes caminhos apresentam os valores de entrada que determinam escolha de um caminho. Para a geração de casos de teste, Leila Naslavsky e equipa aplicam a ideia de uma hierarquia de testes [NZR07]. Utilizam o meta modelo TGH.ecore que suporta relações pais/filhos. No seu modelo, cada raiz representa um valor de entrada para o diagrama de sequência. Como os testes são baseados no diagrama de sequência, as relações entre mensagens de uma sequência e os nódulos são diretas. Através da utilização de uma ferramenta do grupo ATLAS, as relações entre os três artefactos (diagrama de sequência, modelo de controlo de fluxo e hierarquia de testes) são extraídos e criam um modelo de rastreabilidade.

2.4.2 Rastreabilidade em processos de negócio

Wietze Spijkerman e equipa fizeram duas assunções relativas aos BPMN que permitem que a rastreabilidade seja possível. Numa fase inicial, foi reelizado o levantamento dos requisitos e um BPMN criado por um analista com base nestes requisitos [GKVS14] De seguida houve a transformação deste modelo numa aplicação. Estas duas condições permitem, segundo os autores, a rastreabilidade do projeto, assim como avaliar o impacto de mudanças no mesmo. Caso estas condições sejam cumpridas, existem quatro etapas que permitem a obtenção de um relatório com o impacto das alterações. A primeira etapa é a exportação da informação encontrada no BPMN para um ficheiro XML com o auxílio de uma ferramenta própria para o efeito, sugerindo Visual Paradigm. Para além do ficheiro XML, também é preciso fornecer informação sobre o objetivo dos requisitos e do negócio, casos de uso, requisitos funcionais, diagrama de classes e o programa desenvolvido. De seguida, através do uso de uma ferramenta, existe a extração das atividades e outros dados estruturais do BPMN e mapeada a respetiva funcionalidade no programa. O terceiro passo é a deteção das alterações entre os modelos. Aqui, é feita a comparação de duas versões do modelo através do uso de algoritmos de deteção de padrões. Estes algoritmos são capazes de encontrar automaticamente a maioria destas alterações à exceção de quatro (troca de atividades; melhoria de atividades; conjunto de atividades e conjunto parcial de atividades), uma vez que os algoritmos não têm conhecimento semântico sobre as atividades necessárias, sendo realizada esta parte com o auxilio de um utilizador. Por fim, existe a criação de um relatório com a análise de impacto. A ferramenta produz um mapa com as atividades e as funcionalidades afetadas pelas alterações para futura análise.

2.5 ETA-Pro - End-to-end Test Automation Platform for Processes

2.5.1 Introdução

A execução de testes automáticos assumiu um papel importante nas equipas de QA de empresas da área de informática. A repetição dos casos de teste, a redução do tempo de execução de

testes complexos, a eliminação de tarefas e o aumento da cobertura de testes são algumas tarefas e objetivos que estas equipas de QA querem atingir quando estão a assegurar-se da qualidade de produtos. O ETAP-Pro foi desenvolvido para ajudar as equipas de QA a satisfazer estas necessidades.

2.5.2 Plataforma

O ETAP-Pro disponibiliza um conjunto de funcionalidades que visam ajudar as equipas de QA na realização das suas competências. Para tal, existem quatro módulos funcionais: gestão de testes, automação de casos de teste, execução de testes e por fim o módulo de suporte. O módulo de gestão de testes permite criar baterias e casos de teste sendo que para cada processo é possível registar um conjunto de testes. Permite também especificar os respetivos passos e resultados expectáveis de forma a facilitar a leitura e compreensão dos objetivos dos testes. Permite ainda importar descrições de processos de negócio, em formato XPDL e visualizar graficamente estes processos na plataforma na notação BMPN. Por fim, permite gerir requisitos e defeitos, ou seja, permite a rastreabilidade entre requisitos e os casos de testes que os validam. De igual forma permite a gestão dos defeitos associados aos casos de teste executados sem sucesso garantindo assim a sua rastreabilidade.

O modelo de automação de casos de teste permite, em primeiro lugar e como o próprio nome indica, automatizar casos de teste através de um editor onde o utilizador, através de uma meta linguagem da plataforma, pode organizar uma série de componentes de automação que vão permitir automatizar os testes e assim simular um fluxo do processo de negócio. De seguida, vai permitir a criação de componentes de automação o que permite criar e gerir os componentes referidos anteriormente. Por último, vai permitir a associação de etapas de um processo aos passos de um caso de teste. Após a automação, a plataforma permite ao utilizador associar os passos do caso de teste às respetivas etapas do processo de negócio. Esta associação permite aquando da execução dos testes, saber quais as etapas de um processo de negócio que foram testadas com sucesso e as que não foram.

O módulo de execução dos casos de testes permite ao utilizador correr e visualizar os resultados dos testes que executou. Em caso de insucesso de um caso de teste a plataforma marca na representação BPMN quais as etapas ou atividades do processo que foram executadas com sucesso, as que não foram executadas com sucesso e as que não foram de todo executadas. Para além disso, apresenta informação relativa ao teste falhado, referenciando qual o componente que não foi executado com sucesso, o resultado obtido e o esperado.

O módulo de suporte permite também gerir todos os conceitos manipulados pela plataforma, casos de teste, processos, etapas, componentes, entre outros, e regista as relações destes conceitos entre si.

2.5.3 MBT no ETAP-Pro

O processo de testes a modelos de negócios, na plataforma é baseado numa abordagem *keyword-driven*. Como se pode ver na figura 2.5 para a realização deste tipo de testes existem três etapas: criação de casos de teste, construção do *script* de testes e por fim a execução do *script* criado [PFFM18]. Na primeira etapa, a ferramenta, a partir do BPMN, gera um XPDL que gera casos de teste através de uma pesquisa em profundidade de todos os fluxos de processo. Cada um dos fluxos está associado a uma *keyword*. Os casos de teste são gerados na meta linguagem Gherkin, e cada processo está numa lista processos e cada cenário de teste tem uma lista de passos a realizar durante o teste.

Na segunda etapa, os *scripts* são escritos pelos QA, uma vez que a plataforma apenas cria o esqueleto do mesmo a partir da meta linguagem. No passo final, que corresponde à execução do *script* de testes, os resultados podem ser analisados através de estatísticas e registo de execução ou erros. Existe também um diagrama que permite ver os vários fluxos de execução mostrando o resultado do caso de teste assim como as *keywords* correspondentes. Permite ainda ver as relações que as *keywords* têm entre si, percebendo assim os testes que são afetados durante uma alteração.

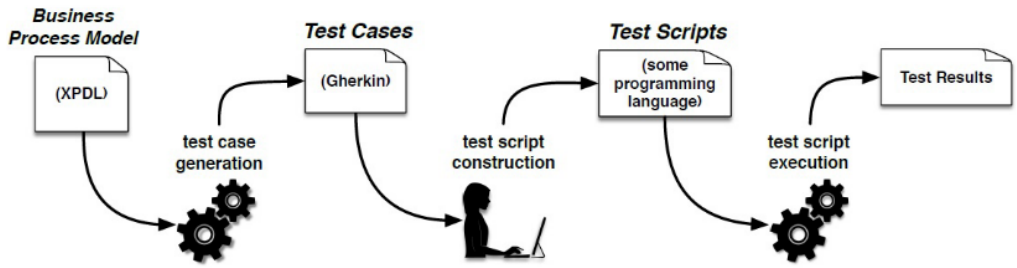

Figura 2.5: Processos de MBT no ETAP-Pro. Retirado de [PFFM18]

2.6 Conclusões

Todo o software é passível de alterações, por isso, é preciso que os processos de negócio estejam adaptados para facilmente serem alterados. Um grande desafio para este tipo de sistemas, que estão em constante mudança, é a manutenção da sua robustez e da sua operacionalidade aquando das mudanças [SSS06]. Por isso, é preciso aplicar formas de os validar, através de testes, que permitem às equipas de desenvolvimento encontrar erros, falhas ou vulnerabilidades nos seus projetos. A modelação de software vem facilitar a interpretação dos sistemas de software por toda a gente envolvida no mesmo, desde engenheiros a clientes, *testers* e outras equipas que possam estar a trabalhar no projeto.

A notação BPMN é utilizada na modelação de processos de negócio, sendo uma das notações mais utilizadas para este efeito. Inicialmente foram mostrados os objetivos e as suas diferentes

finalidades. De seguida, explanou-se sobre os elementos base da notação e quais as suas funcionalidades e respetivas representações gráficas. Com isto ficou-se a conhecer a notação e o potencial que esta apresenta assim como as limitações da sua utilização na engenharia de software.

Este capítulo apresenta e explora a técnica de testes baseada em modelos. São apresentados alguns conceitos inerentes assim como processos, as suas vantagens e desvantagens e ferramentas utilizadas neste contexto. O uso de MBT quando o modelo é criado e implementado conforme as especificações introduz sistematização e automatização no processo de testes. Potencia também uma fase de testes mais curta, menos dispendiosa, com mais cobertura e com mais dados para a mais fácil resolução de falhas.

Um dos assuntos também abordado foi a análise de impacto. É apresentada uma definição geral de análise de impacto na engenharia de software, abordando-se posteriormente duas metodologias direcionadas a modelos. Ambas partem de um ponto em comum que é a exportação da informação dos modelos nomeadamente do fluxo dos processos que representam, existindo diferenças no processo que lhes permite a realização de análise de impacto.

Neste capítulo apresentou-se a ferramenta de teste ETAP-Pro assim como as suas diversas funcionalidades que a tornam numa boa opção para a realização de testes baseados em modelos. Abordou-se também a forma de realizar MBT e as *features* associadas a ela. De salientar que a componente de análise de resultados não permite a comparação entre dois modelos, em diferentes níveis de desenvolvimento,o que acaba por impossibilitar a realização da análise de impacto. Numa realidade em que as atualizações são uma constante a análise de impacto, de soluções informáticas, é uma componente essencial. Por isso, é de todo pertinente criar uma ferramenta que permita a realização de análise de impacto para BPMN e dos seus respectivos artefactos.

Capítulo 3

Metodologia

Neste capítulo irá ser abordada a metodologia utilizada para resolver o problema apresentado no capítulo 1. É abordado o meta modelo com as relações existentes entre BPMN e artefactos, o método utilizado na rastreabilidade entre ficheiros e por fim, a ferramenta criada.

3.1 Introdução

Existe a necessidade de uma ferramenta que permita ter visão de uma solução informática, assim como permitir visualizar o impacto de possíveis alterações. Neste sentido, é proposto um meta modelo onde estão especificadas as relações que um ficheiro BPMN tem com os diferentes artefactos gerados aquando do desenvolvimento de software. De seguida, com estas relações, é possível ter uma ideia do local onde vão ser sentidas estas possíveis alterações.

3.2 Meta Modelo

Numa fase inicial, foi realizada uma análise dos dados de um sistema informático que um ficheiro BPMN consegue representar. A seguinte lista representa os elementos de um ficheiro BPMN que podem ter uma representação gráfica:

- fluxo de partes do sistema assim como do sistema como um todo;
- diagrama de classes;
- atores;
- casos de teste;
- casos de uso;
- esquema da base de dados.

De seguida, foi realizado um levantamento do tipo de artefactos desenvolvidos durante o levantamento de requisitos, desenvolvimento de código e fase de testes. A maioria destes artefactos têm representação gráfica num BPMN. É importante encontrar a relação entre os mesmos.

Na figura 3.1 estão representadas algumas destas relações. Numa fase inicial do projeto, os artefactos criados durante o levantamento de requisitos são, na sua maioria, diagramas de fluxo, classes e atores. São também gerados casos de uso, casos de testes, requisitos funcionais e não funcionais e modelos de interfaces. Grande parte destes elementos gerados estão representados num BPMN. Num BPMN, um conjunto de tarefas que se encontram relacionadas entre si, representam o fluxo de uma parte ou do programa na sua íntegra. Pode-se então concluir que uma *lane* que agrupa um conjunto de atividades interligadas entre sim, pode ser representada por um diagrama de fluxos. As atividades de uma forma isolada podem ser associadas a um diagrama de classes, uma vez, que os últimos são constituídos por um conjunto de métodos que permitem a realização de uma tarefa. As *lanes* contém informação relativa à pessoa ou grupo de pessoas que realizam as atividades nelas contidas, por isso, é possível afirmar que representam os atores do software.

Na fase de desenvolvimento de software, o principal artefacto que é gerado é o código fonte do projeto onde, dependendo da organização do código do projeto, pode ter todas as tarefas num package de lógica e cada tarefa num ficheiro específico.

Na fase de testes, são gerados casos de testes para numa fase seguinte criarem os testes representativos dos mesmos. O BPMN diretamente não consegue representar esta informação, mas a partir de ferramentas externas é possível gerar casos de teste através dos mesmos, como foi visto nos capítulos anteriores. Estes casos de teste estão então relacionados com o BPMN.

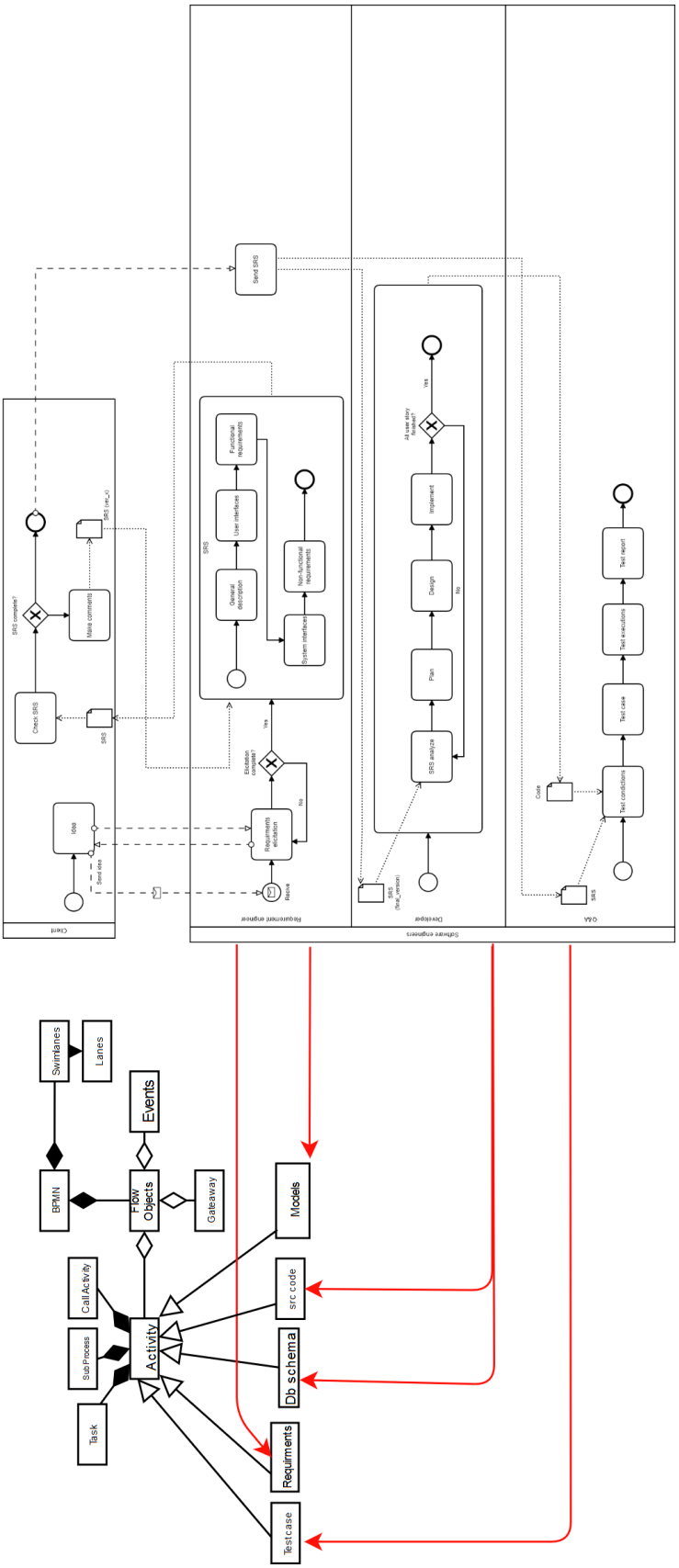

Figura 3.1: Metamodelo com as relações entre atividades de BPMN e artefactos relativos a um programa.

3.3 Abordagem

Após o estudo destas relações, surgiu a necessidade de se estudar a forma de as utilizar para desenvolver uma ferramenta capaz de permitir análise de rastreabilidade de um sistema representado em BPMN e dos seus artefactos. Para tal, foi preciso arranjar um conjunto de dados que deveriam ser retirados de cada um destes ficheiros, para que depois fosse possível ter a informação necessária para se encontrar a relação entre ficheiros e das implicações que uma alteração do BPMN teria nestes ficheiros.

Conclui-se, com base no *code snippet* 3.1, que de um ficheiro BPMN deverá ser retirada a lista de participantes que se encontra dentro da tag <participants>. As atividades são nós filhos do nó *process*. Dentro do nó *task* existe informação referente à sua origem e destino. Os objetos de conexão, visualmente são representados por setas que indicam com a sua orientação o sentido do fluxo do programa.

```
1  <?XML version="1.0" encoding="UTF-8"?>
2  <definitions XMLns="http://www.omg.org/spec/BPMN/20100524/MODEL" XMLns:BPMNdi="http://www.omg.org/
       spec/BPMN/20100524/DI" XMLns:omgdc="http://www.omg.org/spec/DD/20100524/DC" XMLns:omgdi="http://
       www.omg.org/spec/DD/20100524/DI" XMLns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       targetNamespace="" xsi:schemaLocation="http://www.omg.org/spec/BPMN/20100524/MODEL http://www.
       omg.org/spec/BPMN/2.0/20100501/BPMN20.xsd">
3    <collaboration id="sid-c0e745ff-361e-4afb-8c8d-2a1fc32b1424">
4      <participant id="sid-87F4C1D6-25E1-4A45-9DA7-AD945993D06F" name="Customer" processRef="sid-
           C3803939-0872-457F-8336-EAE484DC4A04" />
5    </collaboration>
6    <process id="sid-C3803939-0872-457F-8336-EAE484DC4A04" name="Customer" processType="None" isClosed=
         "false" isExecutable="false">
7      <extensionElements />
8      <laneSet id="sid-b167d0d7-e761-4636-9200-76b7f0e8e83a">
9        <lane id="sid-57E4FE0D-18E4-478D-BC5D-B15164E93254">
10         <flowNodeRef>sid-52EB1772-F36E-433E-8F5B-D5DFD26E6F26</flowNodeRef>
11         <flowNodeRef>sid-E49425CF-8287-4798-B622-D2A7D78EF00B</flowNodeRef>
12         <flowNodeRef>SCAN_OK</flowNodeRef>
13       </lane>
14     </laneSet>
15     <task id="sid-52EB1772-F36E-433E-8F5B-D5DFD26E6F26" name="Scan QR code">
16       <incoming>sid-4DC479E5-5C20-4948-BCFC-9EC5E2F66D8D</incoming>
17       <outgoing>sid-EE8A7BA0-5D66-4F8B-80E3-CC2751B3856A</outgoing>
18     </task>
19     <task id="sid-E49425CF-8287-4798-B622-D2A7D78EF00B" name="Open product information in mobile app
           ">
20       <incoming>sid-8B820AF5-DC5C-4618-B854-E08B71FB55CB</incoming>
21       <outgoing>sid-57EB1F24-BD94-479A-BF1F-57F1EAA19C6C</outgoing>
22     </task>
23     <exclusiveGateway id="SCAN_OK" name="Scan successful?&amp;#10;">
24       <incoming>sid-EE8A7BA0-5D66-4F8B-80E3-CC2751B3856A</incoming>
25       <outgoing>sid-8B820AF5-DC5C-4618-B854-E08B71FB55CB</outgoing>
26       <outgoing>sid-337A23B9-A923-4CCE-B613-3E247B773CCE</outgoing>
27     </exclusiveGateway>
28     <sequenceFlow id="sid-337A23B9-A923-4CCE-B613-3E247B773CCE" name="Yes" sourceRef="SCAN_OK"
           targetRef="sid-5134932A-1863-4FFA-BB3C-A4B4078B11A9" />
29     <sequenceFlow id="sid-4DC479E5-5C20-4948-BCFC-9EC5E2F66D8D" sourceRef="sid-5134932A-1863-4FFA-
           BB3C-A4B4078B11A9" targetRef="sid-52EB1772-F36E-433E-8F5B-D5DFD26E6F26" />
30     <sequenceFlow id="sid-8B820AF5-DC5C-4618-B854-E08B71FB55CB" name="No" sourceRef="SCAN_OK"
           targetRef="sid-E49425CF-8287-4798-B622-D2A7D78EF00B" />
31     <sequenceFlow id="sid-57EB1F24-BD94-479A-BF1F-57F1EAA19C6C" sourceRef="sid-E49425CF-8287-4798-
           B622-D2A7D78EF00B" targetRef="sid-E433566C-2289-4BEB-A19C-1697048900D2" />
32     <sequenceFlow id="sid-EE8A7BA0-5D66-4F8B-80E3-CC2751B3856A" sourceRef="sid-52EB1772-F36E-433E-8
           F5B-D5DFD26E6F26" targetRef="SCAN_OK" />
```

```
33  </process>
```

Listing 3.1: *Code snippet* do XML representativo de um ficheiro BPMN

3.3.1 Outros artefactos

3.3.1.1 Atores

O ficheiro referente aos atores tem uma estrutura simples onde apenas consta uma listagem de atores e respetiva descrição em texto simples (figura 3.2). Neste caso específico, só interessa guardar a informação relativa aos autores.

```
Client
Requirement engineer
Developer
Q&A
```

Figura 3.2: Exemplo de informação relativa a atores.

3.3.1.2 Casos de uso

Este ficheiro já apresenta uma estrutura mais complexa, como pode ser verificado na figura 3.3 uma vez que é composto por um conjunto de casos de uso, contendo um identificador, uma descrição e a prioridade. O tipo de ficheiro escolhido foi um *Comma-separated values* (CSV) uma vez que esta informação é geralmente apresentada em tabelas sendo também acessível a sua construção.

Identificator	Name	Priority	Description
1-01	Requirements elicitation	High	As a Requirement engineer I want to meet with the client to best know what kind of product he wants.
1-02	General description	Low	As a Requirement engineer I want to describe the client product based on elicitation
1-03	User interfaces	Medium	As a Requirement engineer I want to create user interfaces based on elicitation
1-04	Functional requirements	High	As a Requirement engineer I want to create a list of funcional requirements based on elicitaton
1-05	System interfaces	Medium	As a Requirement engineer I want to create sistem interfaces based on elicitation
1-06	Non-functional requirements	High	As a Requirement engineer I want to create a list of non funcional requirements based on elicitaton
1-07	Send	High	As a Requirement engineer I want to send the document created for client evaluation

Figura 3.3: Exemplo de informação relativa a *user stories*

3.3.1.3 Casos de teste

Este ficheiro é em texto simples, mas já apresenta uma estrutura um pouco mais complexa. A estrutura definida foi baseada no resultado do uso da ferramenta ETAP-Pro, sendo, numa fase inicial um ficheiro por cada tarefa. Casos de teste têm o nome do cenário de uma tarefa a realizar, as condições, caso de teste em específico seguido da ação a ser realizada. No exemplo da figura 3.4 temos um cenário para o início da criação de um SRS onde as condições são a ideia e as notas. O cenário a testar é, caso não haja falta de informação, inicia-se a criação do SRS.

```
Scenario TC1 - No missing information
Given Idea and Interview notes
When Is there missing information? and no
Then Start srs and end

Scenario TC2 - Missing information detected
Given Idea and Interview notes
When Is there missing information? and yes
Then Talk to client and add new notes and end
```

Figura 3.4: Exemplo de informação relativa a casos de teste

3.3.1.4 Diagramas de fluxo

Este ficheiro será do tipo *XML Schema Definition* (XSD), porque o Enterprise Architect permite a criação de diagramas de fluxo e a sua exportação em ficheiros XSD. Estes ficheiros têm toda a informação do diagrama num XML sendo este outro motivo para a sua escolha. A informação retirada é relativa às atividades, à sua origem e ao seu destino. Na figura 3.5 temos um exemplo de Diagramas de fluxo e no *listing* 3.2 a representação interna do mesmo diagrama.

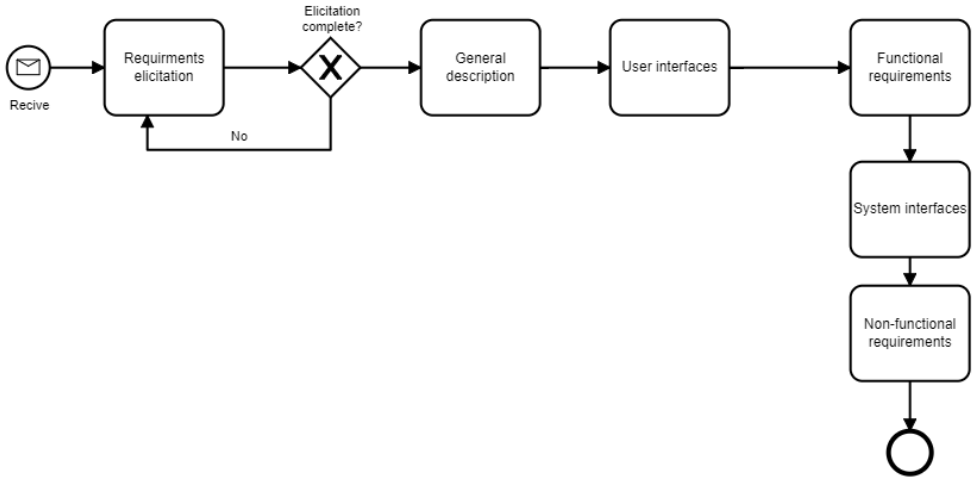

Figura 3.5: Exemplo de informação relativa a um diagrama de fluxo

```
1   <flux:startEvent id="Receive" name="Recive">
2     <flux:outgoing>SequenceFlow_1fou076</flux:outgoing>
3     <flux:messageEventDefinition id="MessageEventDefinition_0llwm2t" />
4   </flux:startEvent>
5   <flux:task id="requirements_elicitation" name="Requirments elicitation">
6     <flux:incoming>SequenceFlow_1fou076</flux:incoming>
7     <flux:incoming>SequenceFlow_0r6gqhc</flux:incoming>
8     <flux:outgoing>SequenceFlow_14g3p5o</flux:outgoing>
9   </flux:task>
10  <flux:exclusiveGateway id="elicitation_complete" name="Elicitation complete?">
11    <flux:incoming>SequenceFlow_14g3p5o</flux:incoming>
12    <flux:outgoing>SequenceFlow_0r6gqhc</flux:outgoing>
13    <flux:outgoing>SequenceFlow_0a8qr75</flux:outgoing>
14  </flux:exclusiveGateway>
```

```
15    <flux:sequenceFlow id="SequenceFlow_1fou076" sourceRef="Receive" targetRef="
          requirements_elicitation" />
16    <flux:sequenceFlow id="SequenceFlow_0r6gqhc" name="No" sourceRef="elicitation_complete" targetRef
          ="requirements_elicitation" />
17    <flux:sequenceFlow id="SequenceFlow_14g3p5o" sourceRef="requirements_elicitation" targetRef="
          elicitation_complete" />
18    <flux:endEvent id="end">
19      <flux:incoming>SequenceFlow_0o2cfqv</flux:incoming>
20    </flux:endEvent>
21    <flux:task id="non_functional_requir" name="Non-functional requirements">
22      <flux:incoming>SequenceFlow_0fxwl13</flux:incoming>
23      <flux:outgoing>SequenceFlow_0o2cfqv</flux:outgoing>
24    </flux:task>
25    <flux:task id="general_desciption" name="General description">
26      <flux:incoming>SequenceFlow_0a8qr75</flux:incoming>
27      <flux:outgoing>SequenceFlow_0qtuu66</flux:outgoing>
28    </flux:task>
29    <flux:task id="functional_requirements" name="Functional requirements">
30      <flux:incoming>SequenceFlow_1yzl2kk</flux:incoming>
31      <flux:outgoing>SequenceFlow_1ti9p0d</flux:outgoing>
32    </flux:task>
33    <flux:task id="user_interfaces" name="User interfaces">
34      <flux:incoming>SequenceFlow_0qtuu66</flux:incoming>
35      <flux:outgoing>SequenceFlow_1yzl2kk</flux:outgoing>
36    </flux:task>
37    <flux:task id="systems_interfaces" name="System interfaces">
38      <flux:incoming>SequenceFlow_1ti9p0d</flux:incoming>
39      <flux:outgoing>SequenceFlow_0fxwl13</flux:outgoing>
40    </flux:task>
41    <flux:sequenceFlow id="SequenceFlow_0o2cfqv" sourceRef="non_functional_requir" targetRef="end" />
42    <flux:sequenceFlow id="SequenceFlow_0fxwl13" sourceRef="systems_interfaces" targetRef="
          non_functional_requir" />
43    <flux:sequenceFlow id="SequenceFlow_0qtuu66" sourceRef="general_desciption" targetRef="
          user_interfaces" />
44    <flux:sequenceFlow id="SequenceFlow_1yzl2kk" sourceRef="user_interfaces" targetRef="
          functional_requirements" />
45    <flux:sequenceFlow id="SequenceFlow_1ti9p0d" sourceRef="functional_requirements" targetRef="
          systems_interfaces" />
46    <flux:sequenceFlow id="SequenceFlow_0a8qr75" sourceRef="elicitation_complete" targetRef="
          general_desciption" />
```

Listing 3.2: *Code snippet* do XML representativo de um diagrama de fluxo

3.3.1.5 Diagramas de classes

Este ficheiro também é do tipo XSD porque é possível ser obtido através da exportação de um UML (figura 3.6). O *listing* 3.3 tem toda a informação de um diagrama de classes nomeadamente métodos e variáveis.

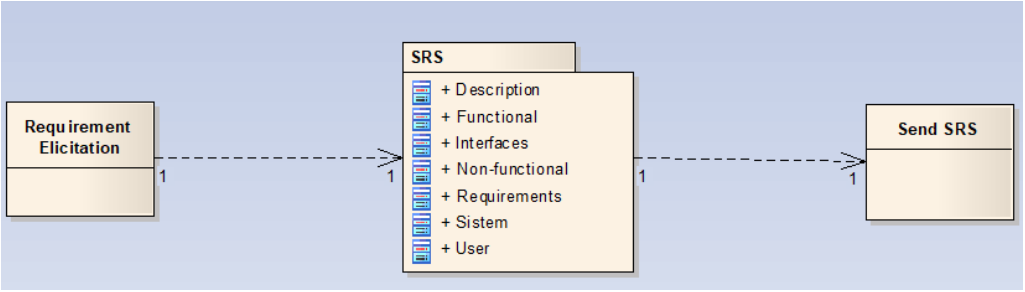

Figura 3.6: Exemplo de informação relativa a um diagrama de classes

```
<?XML version="1.0" encoding="ISO-8859-1"?>
<xs:schema XMLns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Description" type="Description"/>
  <xs:complexType name="Description">
    <xs:sequence>
      <xs:element name="Interfaces" type="Interfaces" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="Requirements" type="Requirements" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="Requirements" type="Requirements"/>
  <xs:complexType name="Requirements">
    <xs:sequence>
      <xs:element name="Interfaces" type="Interfaces" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="Functional" type="Functional"/>
  <xs:complexType name="Functional">
    <xs:complexContent>
      <xs:extension base="Requirements">
        <xs:sequence/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:element name="Non-functional" type="Non-functional"/>
  <xs:complexType name="Non-functional">
    <xs:complexContent>
      <xs:extension base="Requirements">
        <xs:sequence/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:element name="Interfaces" type="Interfaces"/>
  <xs:complexType name="Interfaces">
    <xs:sequence/>
  </xs:complexType>
  <xs:element name="User" type="User"/>
  <xs:complexType name="User">
    <xs:complexContent>
      <xs:extension base="Interfaces">
        <xs:sequence/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:element name="Sistem" type="Sistem"/>
  <xs:complexType name="Sistem">
    <xs:complexContent>
      <xs:extension base="Interfaces">
        <xs:sequence/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:schema>
```

Listing 3.3: *Code snippet* do XML representativo de um diagrama de classes

3.3.1.6 Esquema de base de dados

Este ficheiro vai seguir a mesma lógica dos dois anteriores, tipo XSD, uma vez que é exportado do *Enterprise Architect*. Estes ficheiros têm toda a informação do diagrama num XML. A sua estrutura está presente no listing *listing* 3.4 A informação relevante deste ficheiro é o elemento

xs:complexType sendo o resto informação relacionada com os tipos de atributos e as relações entre as tabelas.

```
1  <?XML version="1.0"?>
2  <xs:schema XMLns:xs="http://www.w3.org/2001/XMLSchema">
3    <xs:element name="Utilizador" type="Utilizador"/>
4    <xs:complexType name="Utilizador">
5      <xs:sequence>
6        <xs:element name="password" type="xs:string" minOccurs="1" maxOccurs="1"/>
7        <xs:element name="username" type="xs:string" minOccurs="1" maxOccurs="1"/>
8      </xs:sequence>
9    </xs:complexType>
10   <xs:element name="Cliente" type="Cliente"/>
11   <xs:complexType name="Cliente">
12     <xs:sequence>
13       <xs:element name="contribuinte" type="xs:int" minOccurs="1" maxOccurs="1"/>
14       <xs:element name="data de nascimento" type="xs:date" minOccurs="1" maxOccurs="1"/>
15       <xs:element name="morada" type="xs:string" minOccurs="1" maxOccurs="1"/>
16     </xs:sequence>
17   </xs:complexType>
18 </xs:schema>
```

Listing 3.4: *Code snippet* do XML representativo de um esquema de base de dados

3.3.2 Relação entre BPMN e artefactos

As relações entre artefactos e tarefas, participantes, processos objetos de fluxo e do próprio BPMN em si têm que ser guardadas. De um lado existe o BPMN e todos os artefactos que estão relacionados a ele, do outro existe a informação necessária à análise de impacto de alterações. O BPMN e os artefactos estão relacionados diretamente, havendo um conjunto de ficheiros para um BPMN. É o caso da informação relativa a atores, diagramas de classes da solução. A parte mais complexa é guardar as relações entre artefactos e tarefas concretas. A escolha final foi guardar, para cada artefacto, todas as tarefas a ele afetas. Escolheu-se este mecanismo porque permite um rápido e fácil acesso à informação quando se quer fazer uma análise de impacto. Ao saber a tarefa sobre a qual se incide as alterações, no segundo BPMN, obtêm-se logo os ficheiros que se relacionam com ela.

Com esta informação, no momento de comparação entre dois BPMN (um do projeto no seu estado inicial e outro com alterações), é possível ver as implicações destas alterações. Para tal é preciso saber o tipo de alterações, de modo descobrir que artefactos externos ao BPMN são influenciados.

Em alterações de permissões ou de atores, para cada atividade do BPMN inicial é necessário confirmar se a mesma atividade existe no segundo BPMN. Se existir, é necessário averiguar na lista de participantes qual deles é que tem o identificador (ID) do processo onde ela se encontra inserida. Se o participante ao qual ela está atribuída sofrer uma alteração, então houve uma alteração de permissões o que implica que o ficheiro referente aos casos de uso pode ter sofrido alterações. Para confirmar se sofreu alterações é necessário examinar que casos de uso tem e se correspondem a algum modificado.

Em alterações do fluxo, caso haja alguma mudança, o diagrama de fluxo associado àquela atividade sofreu alterações. Para além do diagrama de fluxo existem outros ficheiros que podem ser influenciados, nomeadamente o de casos de teste e casos de uso.

Por último, em alterações de atividades, para cada processo é preciso verificar se as tarefas que fazem parte do seu nó descendente sofreram algum tipo de eliminação ou alteração de ID. Ao mudar de ID assumimos que o teor da tarefa foi modificado, que a origem e o destino se mantêm, o diagrama de classe sofreu alterações assim como os casos de uso e casos de teste.

3.4 Implementação

3.4.1 Visão Geral

A plataforma proposta é capaz de recolher, num único sítio, informação de rastreabilidade de um ficheiro BPMN e dos seus artefactos e apresentar esta informação de forma intuitiva ao utilizador. Assim sendo foram definidas as seguintes funcionalidades:

- informação de ficheiro BPMN;
- *upload* dos artefactos de um BPMN;
- *upload* de uma versão do ficheiro BPMN, sobre o qual se quer saber a análise de impacto;
- comparar dois ficheiros BPMN;
- visualizar quais os artefactos implicados com as alterações no BPMN;
- visualizar a análise de impacto.

A plataforma desenvolvida apresenta algumas caraterísticas que contribuem para a fácil perceção da informação contida num BPMN e da sua análise de impacto. De entre os aspetos mais importantes da plataforma, destacam-se os padrões funcionais e de usabilidade. Estas questões são abordadas de forma mais pormenorizada em seguida.

3.4.2 Casos de uso

Os casos de uso desta plataforma são referentes a um utilizador uma vez que não existem diferentes permissões 3.7.

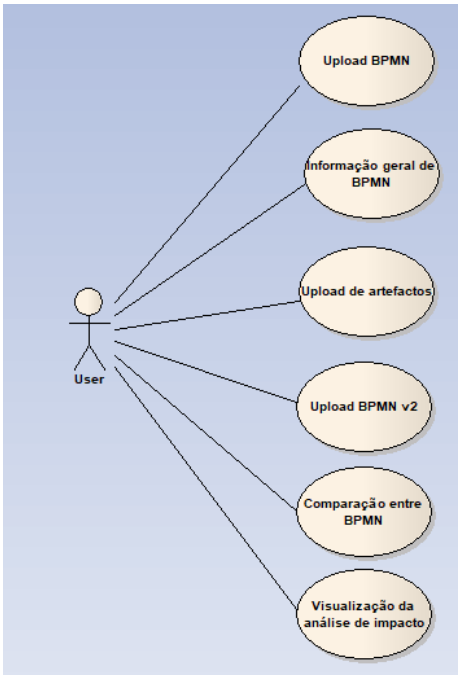

Figura 3.7: Casos de uso da plataforma

3.4.3 Protótipo da plataforma

Os protótipos da *user interface* foram desenhados com o intuito de facilitar o desenvolvimento da plataforma. A vantagem da existência destes protótipos bem definidos desde o início é ter uma base de trabalho que facilite a implementação da plataforma. Em seguida são apresentados os *mockups* desenvolvidos em primeira instância.

A figura 3.8 representa a página base do protótipo com três zonas predefinidas. Na zona do canto inferior esquerdo, o utilizador pode realizar *drop down* de ficheiro BPMN no seu estado inicial. Os artefactos que estão relacionados com ele podem ser *uploaded* no *drop down* à direita. O canto direito permite o upload da versão com alterações do BPMN.

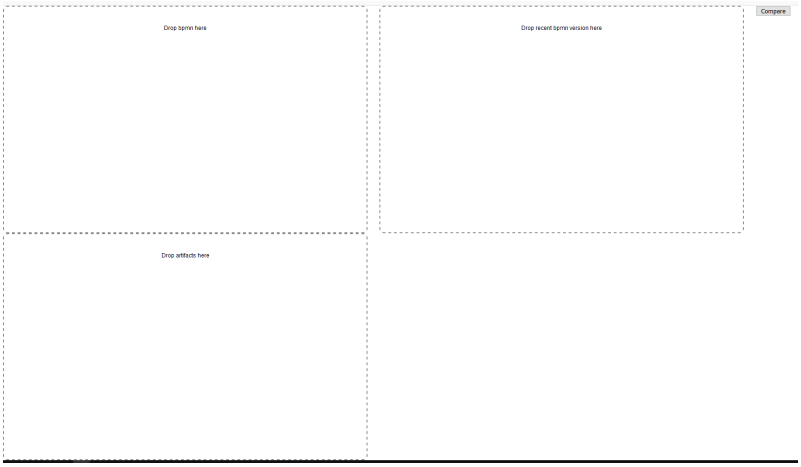

Figura 3.8: Página base da plataforma

A figura 3.9 apresenta a plataforma após o *input* de ficheiros pelo utilizador e pode-se verificar desde logo a existência de uma pré-visualização dos BPMN e uma listagem com os artefactos.

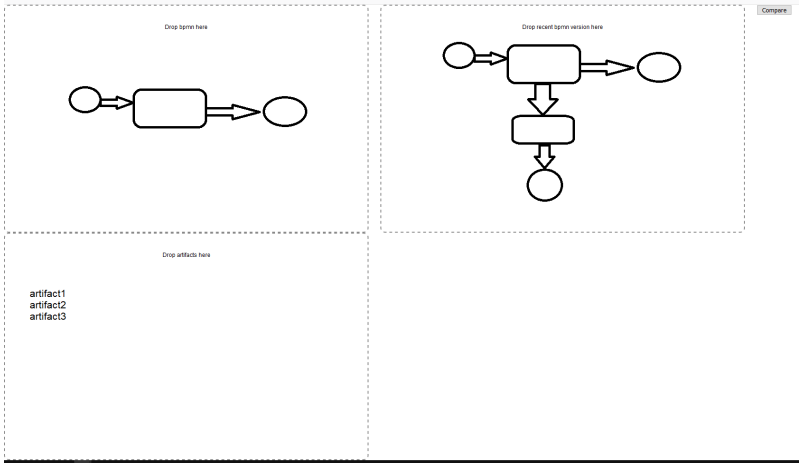

Figura 3.9: Página com informação relativa ao BPMN e artefactos

A figura 3.10 apresenta a plataforma após a comparação de ficheiros pelo utilizador permitindo uma pré-visualização das alterações assinaladas e uma listagem com os artefactos. Do lado direito, é apresentada uma lista de artefactos que podem ter sido afetados pelas alterações.

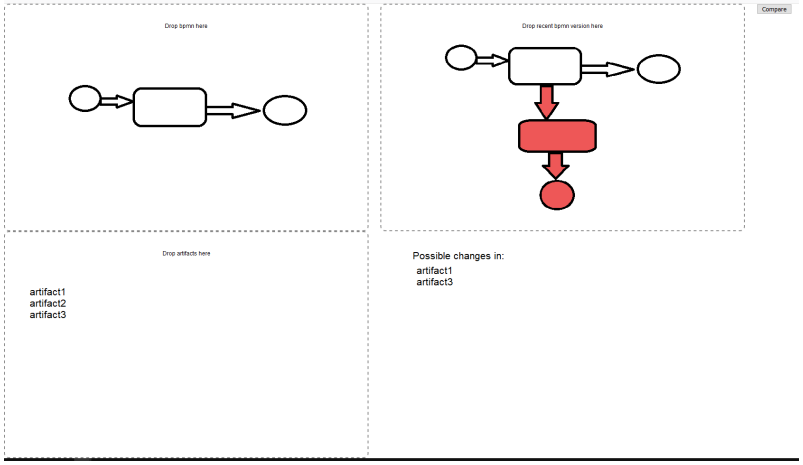

Figura 3.10: Página com informação de análise de impacto

3.4.4 Arquitetura

A plataforma web tem como principal função disponibilizar informação ao utilizador. Para tal, a arquitetura é baseada num padrão MVC (*Model-View-Controller)*. Este padrão permite a separação da representação da informação da interação do utilizador com o sistema, permitindo assim reutilização de código e desenvolvimento paralelo.

Na figura 3.11, é possível visualizar a arquitetura da plataforma desenvolvida.

A plataforma permite a inserção de dados relativos a BPMN e é conseguida através de acessos a uma base de dados. As tabelas criadas que permitem o funcionamento da plataforma, apresentam nomes sugestivos com base na sua utilidade. A seguinte lista, apresenta as tabelas criadas com uma pequena descrição do seu conteúdo:

- BPMN - contém informação relativa ao BPMN;
- *actors* - contém informação relativa aos atores;
- *caseTest* - contém informação relativa aos casos de teste;
- *diagrams* - contém informação relativa aos diagramas;
- *useCase* - contém informação relativa aos casos de uso;
- *relations* - contém informação relativa às relações entre o BPMN e os seus componentes e artefactos desenvolvidos aquando do desenvolvimento de software.

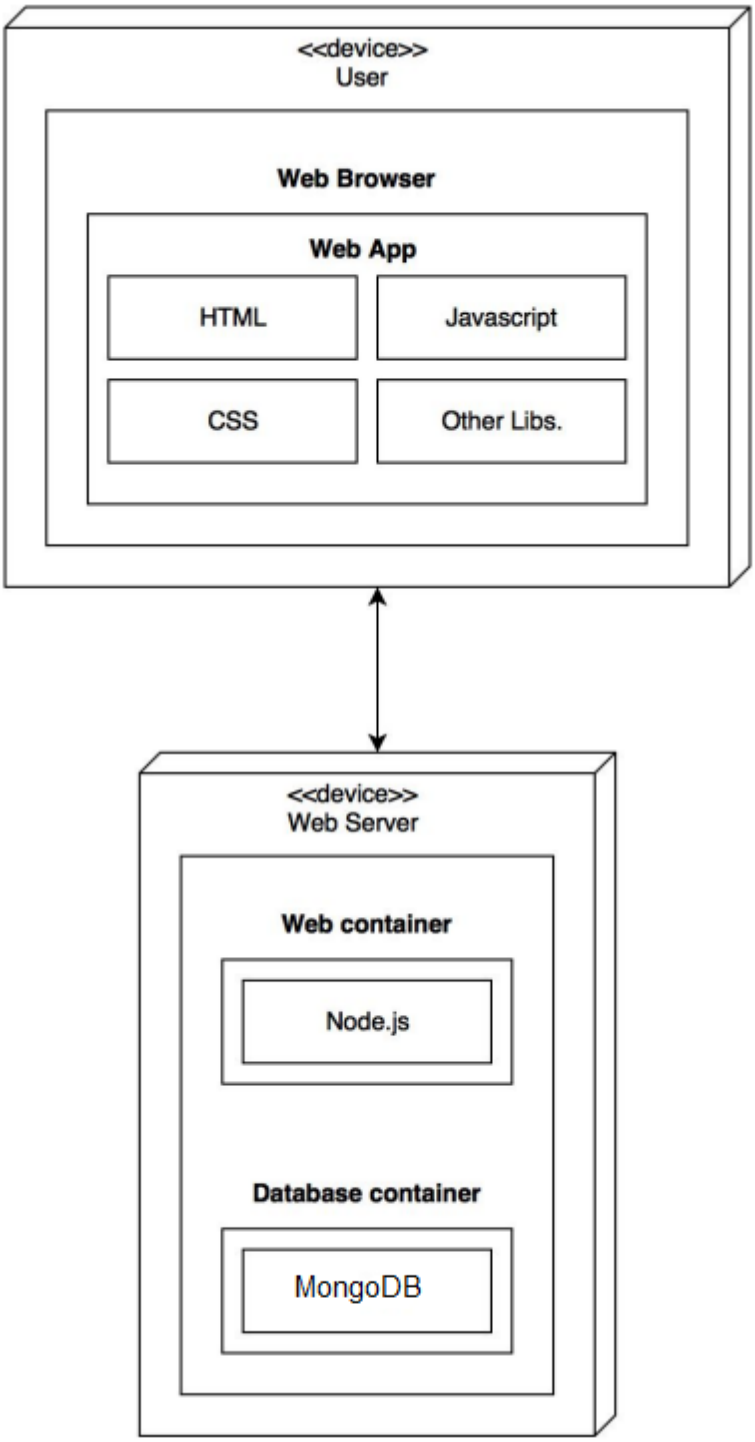

Figura 3.11: Arquitetura da plataforma

3.4.5 Tecnologias de desenvolvimento

O desenvolvimento da plataforma foi feito recorrendo a várias tecnologias *web*, de modo a criar uma plataforma robusta, de fácil desenvolvimento, implementação e atualização.

3.4.5.1 *Front-end*

HTML5

HTML (*Hyper Text Markup Language*) é uma linguagem para estruturação e apresentação de conteúdo de uma página *web*. Na última versão, existem novas funcionalidades a nível de semântica e acessibilidade que permite o uso de novos recursos. Permitem também manipular toda a estrutura e conteúdo de modo a construir aplicações complexas.

CSS3

CSS (*Cascading Style Sheets*) 3 permite adicionar estilo aos elementos HTML. É utilizado para separar o formato do HTML do seu conteúdo, sendo o seu principal objetivo tornar as aplicações visualmente mais agradáveis.

JavaScript

Esta é uma linguagem de programação dinâmica, tipificada, que permite a herança e multi paradigma. Foi idealizada para a execução de *scripts* do lado do cliente e interagir com o cliente sem a necessidade de serem executados no servidor.

jQuery

É uma biblioteca de JavaScript (JS) que permite a manipulação do conteúdo da página *web*. Permite uma escrita fácil e simplificada de scripts de manipulação.

Handlebars

É uma biblioteca que permite a construção de *user interfaces*. É focado numa camada de visualização e permite uma recolha de dados acessível.

3.4.5.2 *Back-end*

Nodejs

É um interpretador de código JS que permite a criação de servidores e ferramentas de conexão. O objetivo principal é criar módulos que comuniquem entre sim e façam todas as operações do lado do servidor, desde fluxos de dados, criptografia a leitura e escrita de ficheiros. Uma grande vantagem que apresenta é o facto de as suas funções conseguirem correr ao mesmo tempo e o uso de *callbacks* para sinalizar sucesso ou insucesso.

MongoDB

É uma base de dados que permite guardar informação de uma forma flexível, uma vez que permite uma fácil alteração da estrutura a qualquer momento. Permite também o mapeamento para objetos do lado de cliente e vice-versa e fácil acesso à informação.

Capítulo 4

Validação

Neste capítulo é apresentado um exemplo de um diagrama BPMN, ficheiros que se relacionam com ele e que são desenvolvidos aquando do desenvolvimento da solução. De seguida são apresentados casos de estudo onde se faz a validação da ferramenta criada para esta dissertação.

4.1 Caso base

O diagrama BPMN sobre o qual incide este caso de estudo é um BPMN que retrata a produção de software. A figura 4.1 apresenta o seu modelo representativo. Este modelo apresenta uma *swimlane* do cliente e as tarefas que lhe são adjacentes (a ideia, a verificação do SRS, a realização de comentários).

Na *pool* referente aos engenheiros de software, existem três *swimlanes* que correspondem a elementos de várias equipas. Cada uma destas *swimlanes* tem tarefas que incidem sobre os participantes. No caso de um engenheiro responsável pelo levantamento de requisitos, este tem como tarefas o levantamento de requisitos. Quando esta tarefa está completa, passa para a tarefa de criação de um documento de requisitos. Esta tarefa é composta por várias pequenas tarefas (elaboração de uma descrição geral do programa e descrição das interfaces de utilizadores, requisitos funcionais, interfaces de sistema e requisitos funcionais). Após a conclusão desta tarefa, existe o envio do documento para o cliente para validação e futuro envio para a equipa de desenvolvimento. A *swimlane* referente aos *developers* tem as tarefas que são realizadas no processo de desenvolvimento de software de uma metodologia *agile*. Aquando da receção do SRS, existe a sua análise, planeamento, design e implementação. Este processo é repetido até todos os requisitos serem concluídos. Quando isto acontecer, os membros da equipa de qualidade começam a realizar o controlo de qualidade. Têm como tarefas delinear as condições de teste, os casos de teste, executar estes testes e realizar um relatório. Como é percetível, este BPMN apresenta um conjunto de elementos diversos, como *swimlanes*, tarefas, artefactos, estados iniciais e finais, fluxo (figura 4.1). Com estes elementos, existe uma quantidade de informação que torna a análise da sua rastreabilidade pertinente. Esta informação está guardada na base de dados como o *listing* A.1 demonstra. Numa primeira instância todo o conteúdo de um BPMN está guardado. Todos

os participantes estão dentro de um *array* de participantes com informação relativa ao ID, nome e processo que lhe está atribuído. De seguida estão os processos, onde se guarda o ID. Dentro dos processos existem *array* de *arrays* onde está guardada a informação das tarefas e noutro, a tarefa de origem e de destino.

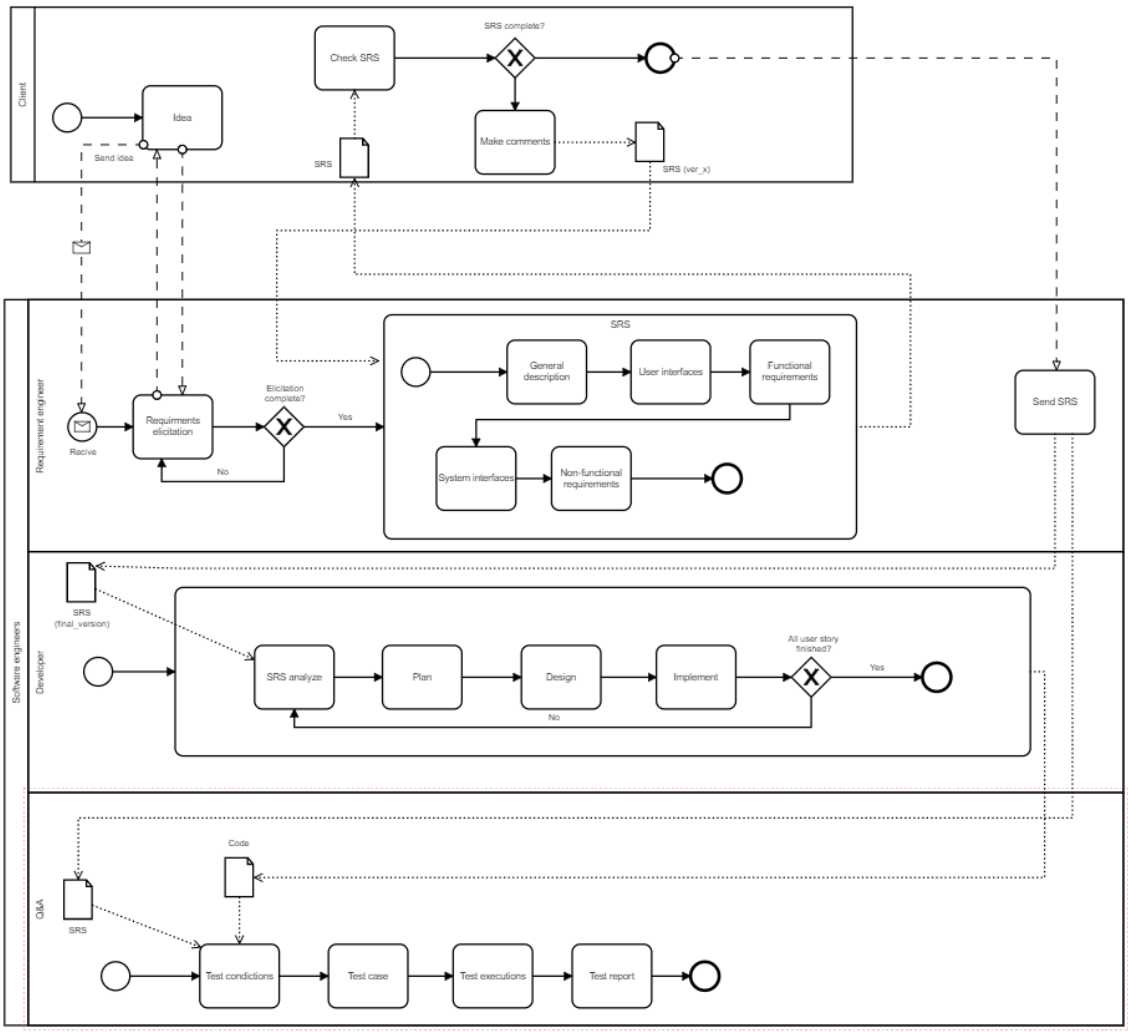

Figura 4.1: BPMN representativo de um processo de desenvolvimento de software

Para além do BPMN, apresentam-se também vários artefactos relacionados a este BPMN. Os artefactos desenvolvidos são atores, *user stories*, casos de teste, diagramas de classes e um diagrama de fluxo. O artefacto dos atores, *listing* 4.4 tem o nome dos quatros atores presentes neste BPMN: cliente, engenheiro de requisitos, programador e QA.

```
{
    "_id": {
        "$oid": "5c3e6753e7179a7d1245fc92"
    },
    "Actors": "Client, Requirement engineer, Developer, Q&A",
    "BPMNId": {
        "$oid": "5c0137f5e7179a6ca07fd978"
```

```
8       }
9   }
```

Listing 4.1: *Code snippet* da estrutura de atores guardada na base de dados

No artefacto de US existe informação relativa ao ator que realiza e o caso de uso. Na figura 4.2, o documento é referente às US do engenheiro de requisitos. Nele estão as sete tarefas que este tem de realizar: levantamento de requisitos, elaborar descrição geral do problema, elaborar requisitos de utilizador, interfaces de utilizador, requisitos não funcionais, interfaces de sistema e enviar o SRS à equipa de desenvolvimento. Esta informação é guardada na base de dados de acordo com o *listing* 4.2.

```
1   {
2       "_id": {
3           "$oid": "5c013d26e7179a6ca07fdc69"
4       },
5       "Scenario 1": {
6           "1-01": "As a Requirement engineer I want to meet with the client to best know what kind of
                product he wants.",
7           "1-02": "As a Requirement engineer I want to describe the client product based on elicitation
                .",
8           "1-03": "As a Requirement engineer I want to create user interfaces based on elicitation.",
9           "1-04": "As a Requirement engineer I want to create a list of funcional requirements based on
                elicitaton.",
10          "1-05": "As a Requirement engineer I want to create sistem interfaces based on elicitation.",
11          "1-06": "As a Requirement engineer I want to create a list of funcional requirements based on
                elicitaton.",
12          "1-07": "As a Requirement engineer I want to create a list of funcional requirements based on
                elicitaton."
13      },
14      "BPMNId": {
15          "$oid": "5c0137f5e7179a6ca07fd978"
16      }
17  }
```

Listing 4.2: *Code snippet* da estrutura de *user stories* guardada na base de dados

Identificator	Name	Priority	Description
1-01	Requirements elicitation	High	As a Requirement engineer I want to meet with the client to best know what kind of product he wants.
1-02	General description	Low	As a Requirement engineer I want to describe the client product based on elicitation
1-03	User interfaces	Medium	As a Requirement engineer I want to create user interfaces based on elicitation
1-04	Functional requirements	High	As a Requirement engineer I want to create a list of funcional requirements based on elicitaton
1-05	System interfaces	Medium	As a Requirement engineer I want to create sistem interfaces based on elicitation
1-06	Non-functional requirements	High	As a Requirement engineer I want to create a list of non funcional requirements based on elicitaton
1-07	Send	High	As a Requirement engineer I want to send the document created for client evaluation

Figura 4.2: Exemplo de informação relativa a *user stories*

Na figura 4.3, estão exemplificados casos de teste para a tarefa de levantamento de requisitos. Para esta tarefa ser realizada, é necessário que exista uma ideia do cliente e notas do levantamento de requisitos. Caso não haja informação em falta, pode-se dar início ao desenvolvimento do SRS. Noutro cenário, onde é necessária a ideia e notas do levantamento de requisitos podem ser necessárias, mas a informação não está completa. Neste caso, fala-se novamente com o cliente de modo a adicionar novas notas. Esta informação vai ser guardada na base de dados de acordo com o *listing* 4.3.

```
{
    "_id": {
        "$oid": "5c013a6ce7179a6ca07fdad9"
    },
    "Task": "Elicitaion",
    "Scenario 1": {
        "Given": "Idea and Interview notes and Review notes ",
        "When": "Is there missing information? and no",
        "Then": "Start srs and end"
    },
    "Scenario 2": {
        "Given": "Idea and Interview notes",
        "When": "Is there missing information? and yes",
        "Then": "Talk to client and add new notes and end"
    },
    "BPMNId": {
        "$oid": "5c0137f5e7179a6ca07fd978"
    }
}
```

Listing 4.3: *Code snippet* da estrutura de casos de teste do levantamento de requisitos guardada na base de dados

```
Scenario TC1 – No missing information
Given Idea and Interview notes
When Is there missing information? and no
Then Start srs and end

Scenario TC2 – Missing information detected
Given Idea and Interview notes
When Is there missing information? and yes
Then Talk to client and add new notes and end
```

Figura 4.3: Exemplo de informação relativa a casos de teste

NNo segundo caso de teste a tarefa é a realização de comentários pela parte do cliente (figura 4.4). Para a realização desta tarefa é necessário um ficheiro de SRS. Caso o documento esteja completo, não é preciso realizar comentários, porque o SRS está completo. Se por outro lado existir a necessidade de realizar comentário, o cliente pode adicionar comentário.

```
{
    "_id": {
        "$oid": "5c4ad978e7179a090e09d5f8"
    },
    "BPMNId": {
        "$oid": "5c0137f5e7179a6ca07fd978"
    },
    "Task": "Make_ciomments",
    "Scenario 1": {
        "Given": "SRS",
        "When": "Is there the need to add comments? and no",
        "Then": "SRS is complete and end"
    },
    "Scenario 2": {
        "Given": "SRS",
        "When": "Is there the need to add comments? and yes",
        "Then": "Add comments and end"
    }
```

```
}
```

Listing 4.4: *Code snippet* da estrutura de casos de teste da realização de comentários guardada na base de dados

```
Scenario TC1 - No missing information
Given SRS
When Is there the need to add comments? and no
Then SRS is complete and end

Scenario TC2 - Missing information detected
Given SRS
When Is there the need to add comments? and yes
Then Add comments and end
```

Figura 4.4: Exemplo de informação relativa a casos de teste da realização de comentários

Neste caso, existem dois diagramas de classes (figuras 4.5 e 4.6). O primeiro tem as classes relativas ao levantamento de requisitos e de envio do SRS, tendo também um package relativo ao desenvolvimento do SRS. O outro diagrama tem as classes do desenvolvimento do SRS. Esta informação é guardada na base de dados da maneira indicada nas *listing* 4.5 e 4.6

```
{
    "_id": {
        "$oid": "5c4acf56e7179a090e09d205"
    },
    "ReqEngClass": [
        {
            "0": "Description",
            "1": "check_SRS"
        }
    ],
    "BPMNId": {
        "$oid": "5c0137f5e7179a6ca07fd978"
    }
}
```

Listing 4.5: *Code snippet* da estrutura das classes afetas ao engenheiro de requisitos

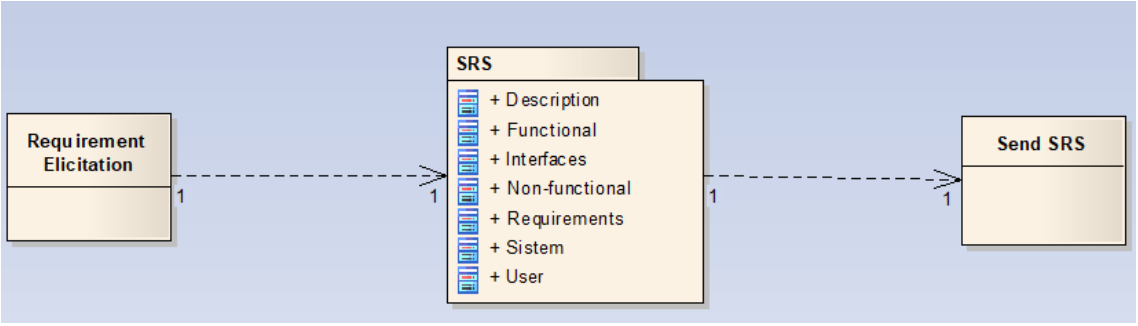

Figura 4.5: Exemplo de informação relativa as classes afetas ao engenheiro de requisitos

```
{
    "_id": {
        "$oid": "5c4acfeae7179a090e09d240"
    },
    "srs_processClass": [
        {
            "0": "Description",
            "1": "Requirements",
            "2": "Functional",
            "3": "Non-funtrional",
            "4": "Interfaces",
            "5": "UserInterfaces",
            "6": "SystemInterfaces"
        }
    ],
    "BPMNId": {
        "$oid": "5c0137f5e7179a6ca07fd978"
    }
}
```

Listing 4.6: *Code snippet* da estrutura das classes afetas à criação do SRS

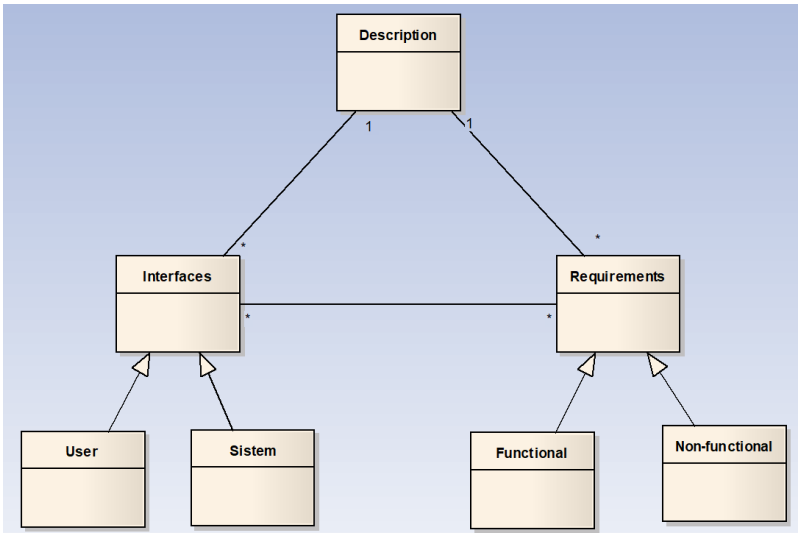

Figura 4.6: Exemplo de informação relativa as classes afetas ao desenvolvimento do SRS

As relações entre estes artefactos está guardada numa tabela de relações. Nesta tabela a informação está guardada como o *listing* 4.7 demonstra. Cada entrada corresponde a um ficheiro BPMN e é guardado o ID dos outros artefactos, como chave estrangeira. Cada ficheiro tem informação referente às tarefas que lhe estão afetas.

```
{
    "_id": {
        "$oid": "5c3faa38e7179a7d1246dc5d"
    },
    "BPMN": {
        "id": "5c0137f5e7179a6ca07fd978"
    },
    "Actors": {
        "id": "5c3e6753e7179a7d1245fc92"
```

```
10      },
11      "CaseTests": {
12          "Elicitaion": {
13              "id": "5c013a6ce7179a6ca07fdad9",
14              "task": "Elicitaion"
15          },
16          "Make_ciomments": {
17              "id": "5c4ad978e7179a090e09d5f8",
18              "task": "Make_cioments"
19          }
20      },
21      "US": {
22          "id": "5c013d26e7179a6ca07fdc69",
23          "name": "ReqEngUS",
24          "Taks": {
25            "0": "Elicitaion",
26            "1": "general_description",
27            "2": "user_interfaces",
28            "3": "functional_requirements",
29            "4": "system_interfaces",
30            "5": "non-functional",
31            "6": "Task_1gcojbg"
32          }
33      },
34      "Diagram": {
35          "ReqEngClass": {
36              "id": "5c4acf56e7179a090e09d205",
37              "name": "ReqEngClass",
38              "Tasks": {
39                  "0": "Elicitaion",
40                  "1": "Task_1gcojbg"
41              }
42          },
43          "srs_processClass": {
44              "id": "5c4acfeae7179a090e09d240",
45              "name": "srs_processClass",
46              "Tasks": {
47                  "0": "general_description",
48                  "1": "user_interfaces",
49                  "2": "functional_requirements",
50                  "3": "system_interfaces",
51                  "4": "non-functional"
52              }
53          }
54      }
55  }
```

Listing 4.7: *Code snippet* da estrutura da tabela de relações

4.2 Cenário 1 - alteração de uma tarefa

Neste primeiro cenário simula-se a alteração da tarefa relativa à realização de comentários por parte do cliente, como é percetível pela figura 4.7. Neste caso, ao colocar o BPMN original e esta nova versão é de esperar como resultado, uma lista de ficheiros que estão relacionados com a tarefa *make comments* do BPMN inicial. Neste caso, o ficheiro que está relacionado com a tarefa de realizar comentário é o ficheiro de casos de teste (figura 4.4).

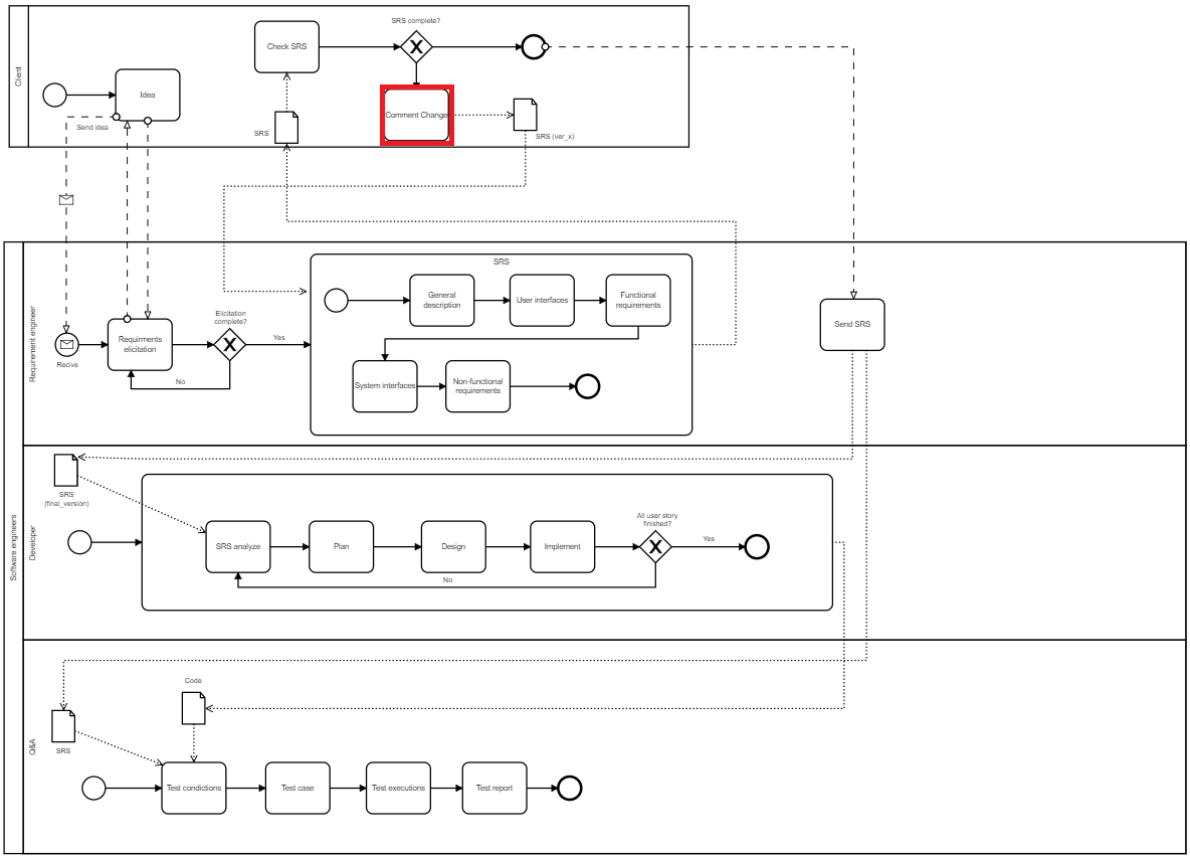

Figura 4.7: Exemplo de alteração de uma tarefa

4.2.1 Resultados

Como é possível ver na figura 4.8 o protótipo mostra um alerta com o processo e o nome da tarefa que foi alterada. Assim, como uma listagem com os nomes dos ficheiros assinalados, ao carregar no OK, o aviso desaparece e é possível ao utilizador navegar pelo BPMN para ver a tarefa implicada. Neste caso, o artefacto assinalado pela plataforma é o caso de teste referente à tarefa do cliente de realização de comentários. A lista também aparece no canto inferior direito, de modo à informação não ficar perdida.

Com base na informação apresentada na secção anterior, o único artefacto relacionado com esta tarefa é de facto relativo aos casos de teste, podendo então concluir-se que a informação representada está correta.

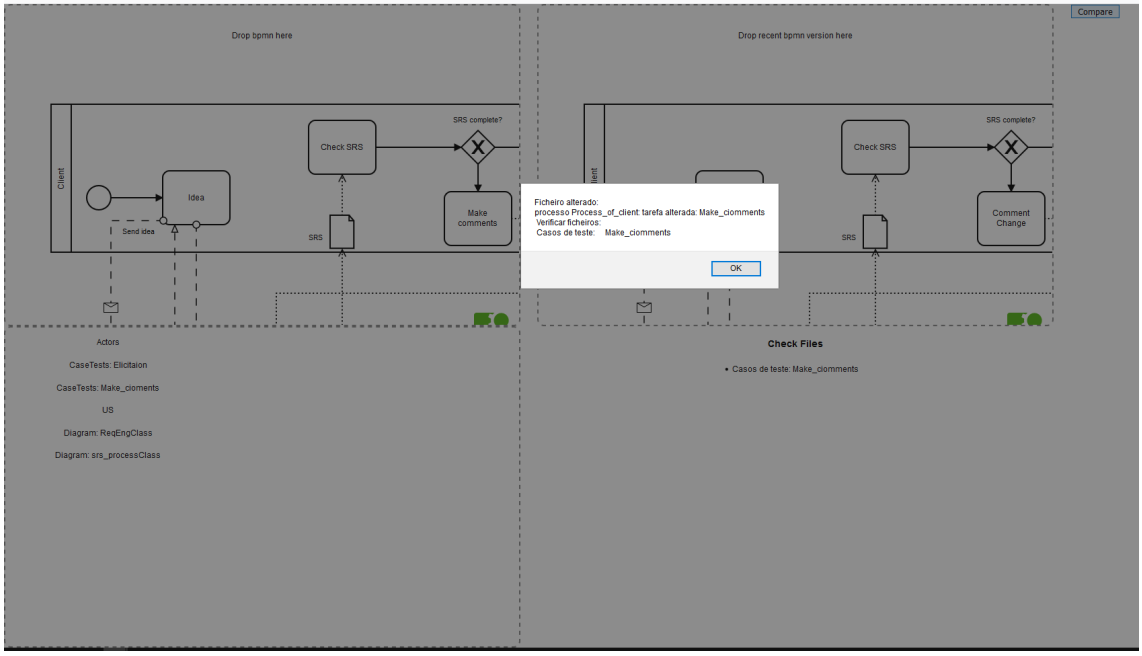

Figura 4.8: Resultado da análise de impacto de alteração de uma tarefa

4.3 Cenário 2 - eliminação de uma tarefa

Neste cenário, simulou-se a eliminação de uma tarefa referente ao *swimlane* do ator engenheiro de requisitos. Apagou-se a sua primeira tarefa, levantamento de requisitos 4.9). Esta alteração vai ter impacto em todos os ficheiros que se relacionam com esta atividade: o caso de testes (*listing* 4.3), de *user stories* (*listing* 4.2), o diagrama de classes do engenheiro de requisitos (figura 4.5) e do diagrama de classes do desenvolvimento de SRS (figura 4.6)

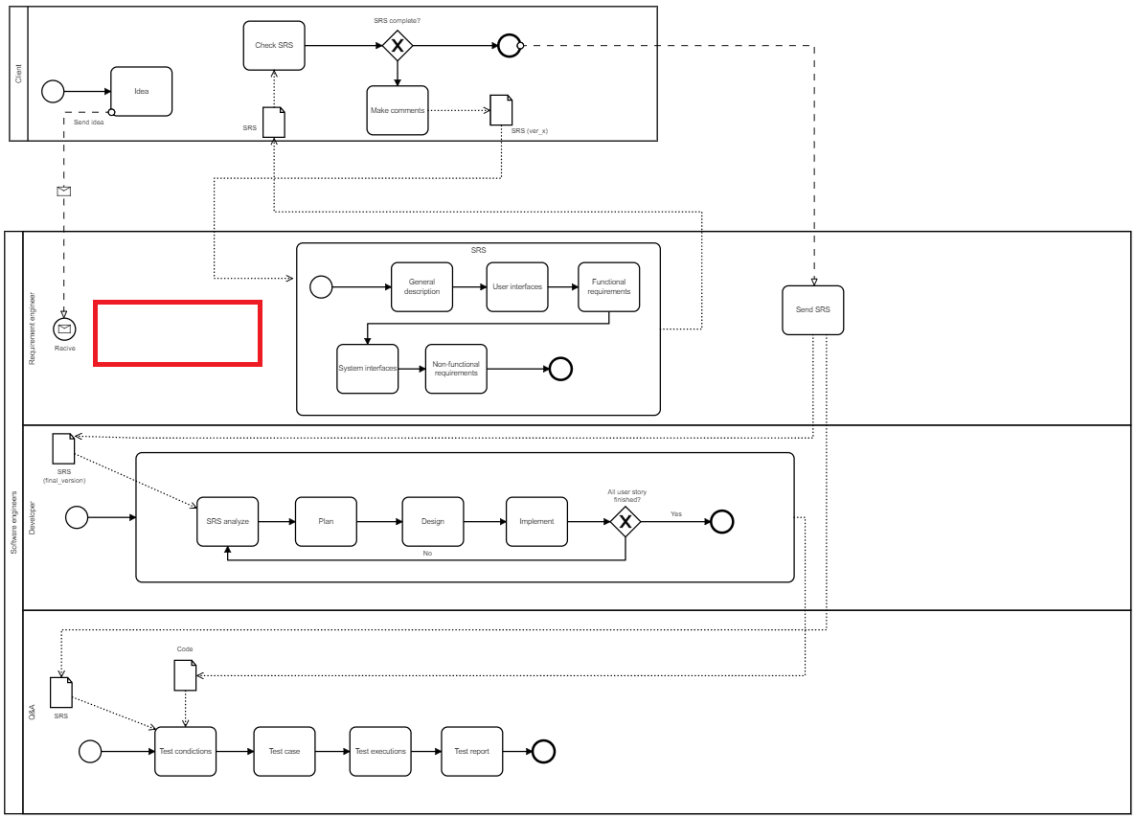

Figura 4.9: BPMN com remoção de uma tarefa

4.3.1 Resultados

Como é possível ver na figura 4.10 o protótipo mostra um alerta com o processo e o nome da tarefa que foi alterada, assim como uma listagem com os nomes dos ficheiros assinalados. Ao carregar no OK, o aviso desaparece e é possível ao utilizador navegar pelo BPMN para ver a tarefa implicada. Neste caso o artefacto assinalado pela plataforma é o caso de teste referente à do engenheiro de requisitos. A lista também aparece no canto inferior direito, de modo à informação não ficar perdida.

Com base na informação apresentada, os artefactos relacionados com esta tarefa são de facto relativos aos casos de uso do engenheiro de requisitos, o diagrama de classes também do engenheiro de requisitos e por fim o caso de testes da tarefas, podendo também concluir-se que a informação representada está correta.

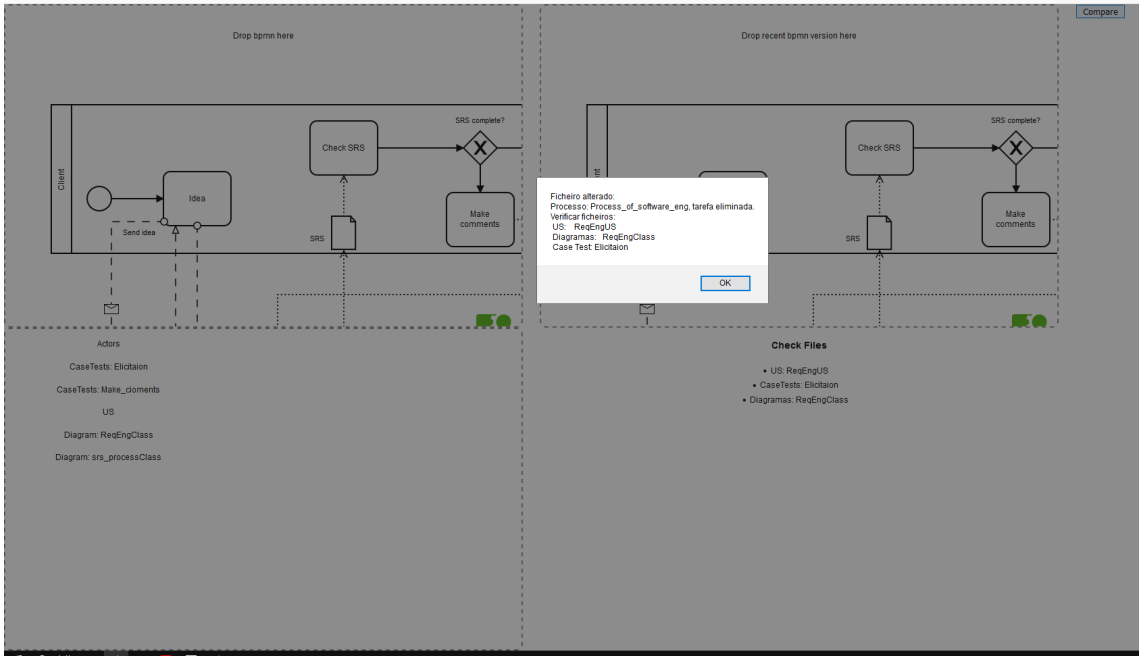

Figura 4.10: Exemplo de eliminação de uma tarefa

4.4 Cenário 3 - criação de uma tarefa

Neste cenário simulou-se a criação de uma tarefa referente ao *swimlane* do ator engenheiro de requisitos. Criou-se a tarefa escrita de documento para representar a escrita do documento de SRS (figura 4.11). Esta alteração vai ter impactos em todos os ficheiros que referem esta *swimlane*. Neste caso será o diagrama de classes do desenvolvimento (listing 4.6).

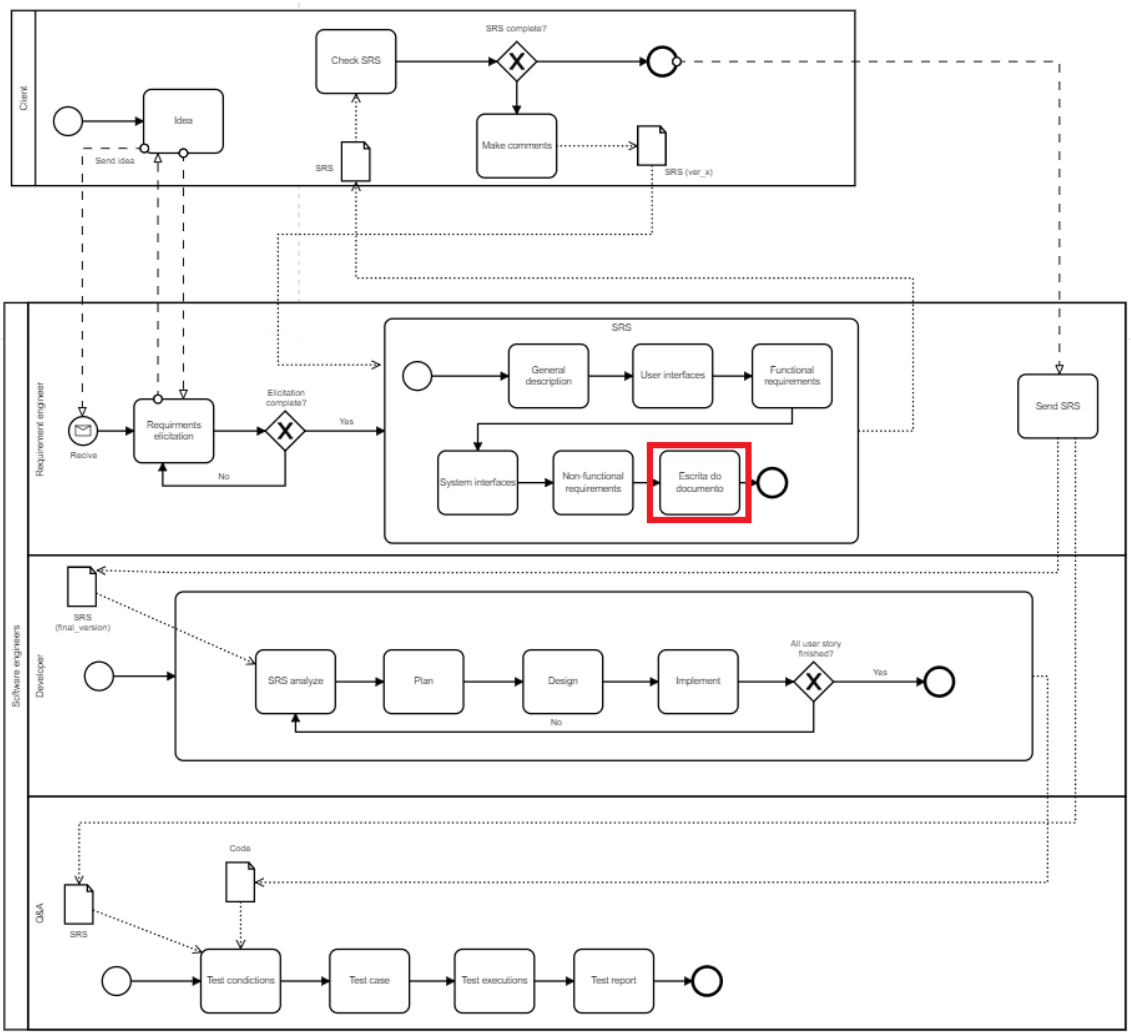

Figura 4.11: Exemplo de criação de uma tarefa

4.4.1 Resultados

Como é possível ver na figura 4.12 o protótipo mostra um alerta com o processo e o nome da tarefa que foi alterada, assim como uma listagem com os nomes dos ficheiros assinalados. Ao carregar no OK, o aviso desaparece e é possível ao utilizador navegar pelo BPMN para ver a tarefa implicada. Neste caso o artefacto assinalado pela plataforma é o diagrama de classes relativo ao engenheiro de requisitos. A lista também aparece no canto inferior direito.

Com base na informação apresentada, o artefacto relacionado com esta tarefa é de facto relativo ao diagrama de classes do engenheiro de requisitos, podendo também concluir que a informação representada está correta.

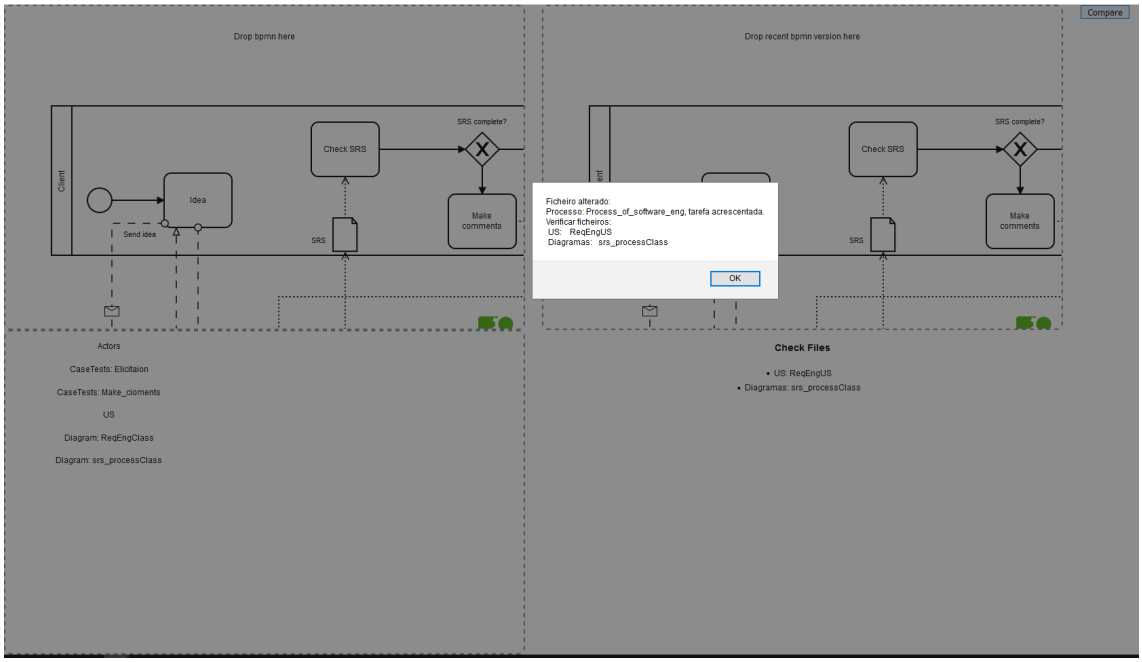

Figura 4.12: Resultado da análise de impacto de adição de uma tarefa

4.5 Conclusões

O caso base sobre o qual se incidiu para validação da plataforma apresenta um conjunto variado de artefactos e um BPMN com algum grau de complexidade. Isto permitiu a validação da plataforma tendo por base três cenários que representam os três tipos de operações que um BPMN pode sofrer (modificação, adição ou remoção de componentes). Para cada uma das experiências efetuadas os resultados obtidos vão ao encontro do esperado. Assim, podemos concluir que a ferramenta desenvolvida é capaz de analisar o conteúdo de um BPMN e criar relação entre o BPMN e os seus artefactos. Também é capaz de realizar uma análise de impacto apresentando o resultado de uma forma clara.

Capítulo 5

Conclusões e Trabalho Futuro

5.1 Considerações finais

A análise de impacto é um fator importante na atualização de *software*, sendo assim importante a existência de plataformas que permitam a realização desta tarefa. Esta dissertação teve como objetivo criar relações entre ficheiro BPMN e artefactos, realizados aquando do desenvolvimento de *software*, e desenvolver um protótipo funcional para a realização de análise de impacto.

Nesta dissertação, começou-se por abordar o problema de falta de relações entre BPMN e artefacto, criando-se um metamodelo que tentasse cobrir o maior número possível de ficheiros. De seguida, implementamos a plataforma que recebia BPMN e artefactos e criava relações entre os artefactos e o BPMN e permita a análise de impacto.

Na conceção das relações considerou-se um número de artefactos, passíveis de criação num projeto informático. No entanto, não foram considerados todos os artefactos, uma vez que são em grande número e numa primeira fase, o objetivo era averiguar se era possível. Com estas relações criadas, passou-se à implementação do protótipo. Esta ferramenta foi desenvolvida ao longo do semestre duma forma iterativa e modular. Isto permite uma fácil compreensão e atualização do *software*. O protótipo permite então visualizar informação de um BPMN, associar a ele artefactos e associar uma versão diferente do mesmo, de maneira a fazer-se uma análise de impacto das alterações.

A validação desta ferramenta, e consequentemente da dissertação, foi feita com base em casos de uso. Estes casos de uso permitiram testar a ferramenta em três casos possíveis: criação, eliminação e atualização. Verificou-se que, para os artefactos testados o protótipo da ferramenta teve um comportamento correto, mostrando a informação relativa à análise de impacto.

5.2 Dificuldades

As principais dificuldades encontradas no desenvolvimento desta dissertação, foram numa altura inicial encontrar as relações entre BPMN e os artefactos. Depois de se encontrar estas relações, encontrar uma maneira de guardar esta informação sem recorrer a estruturas com um elevado grau de complexidade também foi desafiante. No entanto, o protótipo não ficou longe das expectativas e os resultados podem ser considerados positivos.

5.3 Trabalho Futuro

Apesar do trabalho realizado, a ferramenta criada é apenas um protótipo precisando, por isso, de trabalho futuro. Pode-se começar por expandir as relações dos artefactos aos restantes tipos: design e arquitetura, código fonte e manual de utilizador. Definir uma estrutura adequada para cada um deles e futuramente integrar com a ferramenta também são passos seguintes lógicos deste trabalho. Outra componente passível de ser adicionada à plataforma, é adicionar vistas para cada um dos artefactos associados a um BPMN, permitindo assim ao utilizador abrir, por exemplo, um diagrama de classes e ver o diagrama e as possíveis implicações. Por fim, num caso mais avançado, seria interessante que a ferramenta pudesse fazer interpretação de linguagem natural e com base nela criar o mapa de rastreabilidade.

Referências

[AO08] Paul Ammann e Jeff Offutt. *Introduction to Software Testing*. 2008. URL: http://www.cse.hcmut.edu.vn/{~}hiep/KiemthuPhanmem/Tailieuthamkhao/IntroductiontoSoftwareTesting.pdf{%}0Ahttp://ebooks.cambridge.org/ref/id/CBO9780511809163, doi:10.1017/CBO9780511809163.

[BLOS06] L. C. Briand, Y. Labiche, L. O'Sullivan e M. M. Sówka. Automated impact analysis of UML models. *Journal of Systems and Software*, 79(3):339–352, 2006. doi:10.1016/j.jss.2005.05.001.

[Cai11] L Cai. A business process testing sequence generation approach based on test cases composition. *1st ACIS/JNU International Conference on Computers, Networks, Systems, and Industrial Engineering, CNSI 2011*, pages 178–185, 2011. URL: http://www.scopus.com/inward/record.url?eid=2-s2.0-80051655758{&}partnerID=40{&}md5=9255976cf19f5162dcd4295ed1be1c99, doi:10.1109/CNSI.2011.12.

[GKVS14] Arda Goknil, Ivan Kurtev, Klaas Van Den Berg e Wietze Spijkerman. Change impact analysis for requirements: A metamodeling approach. *Information and Software Technology*, 56(8):950–972, 2014. URL: http://dx.doi.org/10.1016/j.infsof.2014.03.002, doi:10.1016/j.infsof.2014.03.002.

[Int11] International Software Testing Qualifications Board. Certified Tester Foundation Level Syllabus. page 85, 2011.

[KG03] Antje von Knethen e Mathias Grund. Quatrace: A tool environment for (semi-) automatic impact analysis based on traces. In *Proceedings of the International Conference on Software Maintenance*, ICSM '03, pages 246–, Washington, DC, USA, 2003. IEEE Computer Society. URL: http://dl.acm.org/citation.cfm?id=942800.943589.

[LB13] Bruno Legeard e Arnaud Bouzy. Smartesting CertifyIt: Model-based testing for enterprise IT. *Proceedings - IEEE 6th International Conference on Software Testing, Verification and Validation, ICST 2013*, 1(1):391–397, 2013. doi:10.1109/ICST.2013.55.

[LEK16] Nan Li, Anthony Escalona e Tariq Kamal. Skyfire: Model-Based Testing with Cucumber. *Proceedings - 2016 IEEE International Conference on Software Testing, Verification and Validation, ICST 2016*, pages 393–400, 2016. doi:10.1109/ICST.2016.41.

[Luo01] Lu Luo. Software testing techniques. *Institute for software research international Carnegie mellon university Pittsburgh, PA*, 15232(1-19):19, 2001.

[MP13] Tiago Monteiro e Ana C R Paiva. Pattern based GUI testing modeling environment. *Proceedings - IEEE 6th International Conference on Software Testing, Verification and Validation Workshops, ICSTW 2013*, pages 140–143, 2013. doi:10.1109/ICSTW.2013.24.

[MP14a] M. L. M. Rodrigo Moreira e Ana C. R. Paiva. Towards a Pattern Language for Model-Based GUI Testing. *19th European Conference on Pattern Languages of Programs (EuroPLoP 2014)*, 2014. doi:10.1145/2721956.2721972.

[MP14b] Rodrigo M. L. M. Moreira e Ana C. R. Paiva. A GUI modeling DSL for pattern-based GUI testing - PARADIGM. *9th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*, pages 126–135, 2014. URL: https://doi.org/10.5220/0004880601260135, doi:10.5220/0004880601260135.

[MPNM17] Rodrigo M.L.M. Moreira, Ana Cristina Paiva, Miguel Nabuco e Atif Memon. Pattern-based GUI testing: Bridging the gap between design and quality assurance. *Software Testing Verification and Reliability*, 27(3), 2017. URL: https://doi.org/10.1002/stvr.1629, doi:10.1002/stvr.1629.

[NF12] O M G Document Number e Associated Schema Files. Business Process Model and Notation. 125(January), 2012. URL: http://link.springer.com/10.1007/978-3-642-33155-8, doi:10.1007/978-3-642-33155-8.

[NP14] Miguel Nabuco e Ana C. Paiva. Model-based test case generation for web applications. *14th International Conference on Computational Science and Its Applications; ICCSA 2014 - Volume 8584*, pages 248–262, 2014. URL: https://doi.org/10.1007/978-3-319-09153-2_19.

[NZR07] Leila Naslavsky, Hadar Ziv e Debra J. Richardson. Towards traceability of model-based testing artifacts. In *Proceedings of the 3rd international workshop on Advances in model-based testing - A-MOST '07*, pages 105–114, 2007. URL: http://portal.acm.org/citation.cfm?doid=1291535.1291546, doi:10.1145/1291535.1291546.

[OAH03] Alessandro Orso, Taweesup Apiwattanapong e Mary Jean Harrold. Leveraging field data for impact analysis and regression testing. *Proceedings of the 9th European software engineering conference held jointly with 10th ACM SIGSOFT international symposium on Foundations of software engineering - ESEC/FSE '03*, page 128, 2003. URL: http://portal.acm.org/citation.cfm?doid=940071.940089, doi:10.1145/940071.940089.

[OMG17] OMG. Unified Modeling Language Specification. (December):796, 2017. URL: https://www.omg.org/spec/UML/.

[PFFM18] Ana C.R. Paiva, Nuno H. Flores, João P. Faria e José M.G. Marques. End-to-end Automatic Business Process Validation. *Procedia Computer Science*, 130:999–1004, jan 2018. URL: https://www.sciencedirect.com/science/article/pii/S1877050918304666, doi:10.1016/J.PROCS.2018.04.104.

REFERÊNCIAS

[PV17] Ana C.R. Paiva e Liliana Vilela. Multidimensional test coverage analysis: PARADIGM-COV tool. *Cluster Computing*, 20(1):633–649, 2017. doi:10.1007/s10586-017-0728-4.

[RMLMM13] Atif Memon Rodrigo M. L. M. Moreira, Ana C. R. Paiva. A pattern-based approach for GUI modeling and testing. *24th International Symposium on Software Reliability Engineering (ISSRE)*, pages 288–297, 2013. URL: https://doi.org/10.1109/ISSRE.2013.6698881, doi:10.1109/ISSRE.2013.6698881.

[RZ14] Václav Repa e Ondrej Zeleznik. Methodological limitations of modeling languages BPMN and ARIS. *Proceedings of the 2014 15th International Carpathian Control Conference, ICCC 2014*, pages 507–512, 2014. doi:10.1109/CarpathianCC.2014.6843657.

[SSS06] J Schiefer, G Saurer e A Schatten. Testing event-driven business processes. *Journal of Computers (Finland)*, 1(7):69–80, 2006. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84871575300{&}partnerID=40{&}md5=52fe43e700f39767e4ce703204b95f37.

[VCG+08] Margus Veanes, Colin Campbell, Wolfgang Grieskamp, Wolfram Schulte, Nikolai Tillmann e Lev Nachmanson. Model-based testing of object-oriented reactive systems with spec explorer. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4949 LNCS:39–76, 2008.

[Wan06] A Comparison of Business Process Modeling Methods. *IEEE International Conference on Service Operations and Logistics, and Informatics*, pages 1136–1141, 2006. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4125745.

REFERÊNCIAS

Anexo A

Code snippet

```
{
    "_id": {
        "$oid": "5c0137f5e7179a6ca07fd978"
    },
    "bpmnInformation": {
        "bpmn": [
            "  <semantic:message id=\"message_idea\" />\n  <semantic:collaboration id=\"
                Collaboratis_analyze\">\n    <semantic:participant id=\"Client\" name=\"Client\"
                processRef=\"Process_of_client\" />\n    <semantic:participant id=\"
                software_engineers\" name=\"Software engineers\" processRef=\"
                Process_of_software_eng\" />\n    <semantic:messageFlow id=\"idea_transmission\"
                name=\"Send idea\" sourceRef=\"idea\" targetRef=\"engineer_start\" messageRef=\"
                message_idea\" triso:userConstraints=\"true\" />\n    <semantic:messageFlow id=\"
                MessageFlow_165u97y\" sourceRef=\"Elicitaion\" targetRef=\"idea\" />\n    <semantic:
                messageFlow id=\"MessageFlow_1q9q5zz\" sourceRef=\"idea\" targetRef=\"Elicitaion\"
                />\n    <semantic:messageFlow id=\"MessageFlow_17e4t0h\" sourceRef=\"
                EndEvent_1hk4stl\" targetRef=\"Task_1gcojbg\" />\n  <\/semantic:collaboration>\n  <
                semantic:process id=\"Process_of_client\" isExecutable=\"true\">\n    <semantic:
                startEvent id=\"client_start\">\n      <semantic:outgoing>conect_client_start <\/
                semantic:outgoing>\n    <\/semantic:startEvent>\n    <semantic:task id=\"idea\" name
                =\"Idea\">\n      <semantic:incoming>conect_client_start <\/semantic:incoming>\n
                <\/semantic:task>\n    <semantic:task id=\"Check_SRS\" name=\"Check SRS\">\n      <
                semantic:outgoing>SequenceFlow_0b8wnpv <\/semantic:outgoing>\n      <semantic:
                property id=\"Property_1rr8gw7\" name=\"__targetRef_placeholder\" />\n      <
                semantic:dataInputAssociation id=\"DataInputAssociation_081r59h\">\n        <
                semantic:sourceRef>DataObjectReference_srs <\/semantic:sourceRef>\n        <semantic:
                targetRef>Property_1rr8gw7 <\/semantic:targetRef>\n      <\/semantic:
                dataInputAssociation>\n    <\/semantic:task>\n    <semantic:task id=\"Make_ciomments
                \" name=\"Make comments\">\n      <semantic:incoming>SequenceFlow_0zoiw1p <\/semantic
                :incoming>\n      <semantic:dataOutputAssociation id=\"DataOutputAssociation_1veg43l
                \">\n        <semantic:targetRef>DataObjectReference_17op23t <\/semantic:targetRef>\n
                    <\/semantic:dataOutputAssociation>\n    <\/semantic:task>\n    <semantic:
                sequenceFlow id=\"SequenceFlow_0zoiw1p\" sourceRef=\"ExclusiveGateway_1c11kak\"
                targetRef=\"Make_ciomments\" />\n    <semantic:sequenceFlow id=\"
                SequenceFlow_0b8wnpv\" sourceRef=\"Check_SRS\" targetRef=\"ExclusiveGateway_1c11kak
                \" />\n    <semantic:sequenceFlow id=\"conect_client_start\" sourceRef=\"
                client_start\" targetRef=\"idea\" />\n    <semantic:dataObjectReference id=\"
                DataObjectReference_srs\" name=\"SRS\" dataObjectRef=\"DataObject_1v2vvr2\" />\n
                <semantic:dataObject id=\"DataObject_1v2vvr2\" />\n    <semantic:dataObjectReference
                 id=\"DataObjectReference_17op23t\" name=\"SRS (ver_x)\" dataObjectRef=\"
                DataObject_09w1816\" />\n    <semantic:dataObject id=\"DataObject_09w1816\" />\n
                <semantic:exclusiveGateway id=\"ExclusiveGateway_1c11kak\" name=\"SRS complete?\">\n
                    <semantic:incoming>SequenceFlow_0b8wnpv <\/semantic:incoming>\n      <semantic:
                outgoing>SequenceFlow_0zoiw1p <\/semantic:outgoing>\n      <semantic:outgoing>
                SequenceFlow_02nppwt <\/semantic:outgoing>\n    <\/semantic:exclusiveGateway>\n    <
                semantic:sequenceFlow id=\"SequenceFlow_02nppwt\" sourceRef=\"
```

Code snippet

```
ExclusiveGateway_1c11kak\" targetRef=\"EndEvent_1hk4stl\" />\n    <semantic:endEvent
 id=\"EndEvent_1hk4stl\">\n      <semantic:incoming>SequenceFlow_02nppwt <\/semantic:
incoming>\n    <\/semantic:endEvent>\n  <\/semantic:process>\n  <semantic:process id
=\"Process_of_software_eng\" isExecutable=\"false\">\n    <semantic:laneSet id=\"
LaneSet_1nuh8od\">\n      <semantic:lane id=\"Lane_14a0gkx\" name=\"Requirement
engineer\">\n        <semantic:flowNodeRef>engineer_start <\/semantic:flowNodeRef>\n
      <semantic:flowNodeRef>Elicitaion <\/semantic:flowNodeRef>\n        <semantic:
flowNodeRef>ExclusiveGateway_0c11chy <\/semantic:flowNodeRef>\n        <semantic:
flowNodeRef>Task_1gcojbg <\/semantic:flowNodeRef>\n        <semantic:flowNodeRef>
srs_process <\/semantic:flowNodeRef>\n      <\/semantic:lane>\n      <semantic:lane
id=\"Lane_00oyka8\" name=\"Developer\">\n        <semantic:flowNodeRef>
SubProcess_03u5bq0 <\/semantic:flowNodeRef>\n        <semantic:flowNodeRef>
StartEvent_dev <\/semantic:flowNodeRef>\n      <\/semantic:lane>\n      <semantic:
lane id=\"Lane_11g0uwu\" name=\"Q&#38;A\">\n        <semantic:flowNodeRef>
StartEvent_1178x47 <\/semantic:flowNodeRef>\n        <semantic:flowNodeRef>
Task_1dk5o93 <\/semantic:flowNodeRef>\n        <semantic:flowNodeRef>Task_11zbn01 <\/
semantic:flowNodeRef>\n        <semantic:flowNodeRef>Task_1gzqf1h <\/semantic:
flowNodeRef>\n        <semantic:flowNodeRef>Task_04bjijv <\/semantic:flowNodeRef>\n
      <semantic:flowNodeRef>EndEvent_0hwer4x <\/semantic:flowNodeRef>\n      <\/
semantic:lane>\n    <\/semantic:laneSet>\n    <semantic:startEvent id=\"
engineer_start\" name=\"Recive\">\n      <semantic:outgoing>SequenceFlow_1oh7f1c <\/
semantic:outgoing>\n      <semantic:messageEventDefinition messageRef=\"message_idea
\" />\n    <\/semantic:startEvent>\n    <semantic:task id=\"Elicitaion\" name=\"
Requirments elicitation \">\n      <semantic:incoming>SequenceFlow_1oh7f1c <\/
semantic:incoming>\n      <semantic:incoming>SequenceFlow_0i9ge2s <\/semantic:
incoming>\n      <semantic:outgoing>SequenceFlow_0jkwt9n <\/semantic:outgoing>\n
<\/semantic:task>\n    <semantic:exclusiveGateway id=\"ExclusiveGateway_0c11chy\"
name=\"Elicitation complete?\">\n      <semantic:incoming>SequenceFlow_0jkwt9n <\/
semantic:incoming>\n      <semantic:outgoing>SequenceFlow_02l4bwr <\/semantic:
outgoing>\n      <semantic:outgoing>SequenceFlow_0i9ge2s <\/semantic:outgoing>\n
<\/semantic:exclusiveGateway>\n    <semantic:sequenceFlow id=\"SequenceFlow_0i9ge2s
\" name=\"No\" sourceRef=\"ExclusiveGateway_0c11chy\" targetRef=\"Elicitaion\" />\n
   <semantic:sequenceFlow id=\"SequenceFlow_02l4bwr\" name=\"Yes\" sourceRef=\"
ExclusiveGateway_0c11chy\" targetRef=\"srs_process\" />\n    <semantic:sequenceFlow
id=\"SequenceFlow_0jkwt9n\" sourceRef=\"Elicitaion\" targetRef=\"
ExclusiveGateway_0c11chy\" />\n    <semantic:sequenceFlow id=\"SequenceFlow_1oh7f1c
\" sourceRef=\"engineer_start\" targetRef=\"Elicitaion\" />\n    <semantic:
subProcess id=\"srs_process\" name=\"SRS\">\n      <semantic:incoming>
SequenceFlow_02l4bwr <\/semantic:incoming>\n      <semantic:property id=\"
Property_15k4xbm\" name=\"__targetRef_placeholder\" />\n      <semantic:
dataInputAssociation id=\"DataInputAssociation_01fxzxv\">\n        <semantic:
sourceRef>DataObjectReference_17op23t <\/semantic:sourceRef>\n        <semantic:
targetRef>Property_15k4xbm <\/semantic:targetRef>\n      <\/semantic:
dataInputAssociation>\n      <semantic:dataOutputAssociation id=\"
DataOutputAssociation_0fmjrr7\">\n        <semantic:targetRef>
DataObjectReference_srs <\/semantic:targetRef>\n      <\/semantic:
dataOutputAssociation>\n      <semantic:task id=\"system_interfaces\" name=\"System
interfaces\">\n        <semantic:incoming>SequenceFlow_1nwmtnw <\/semantic:incoming>\
n        <semantic:outgoing>SequenceFlow_177qn9w <\/semantic:outgoing>\n      <\/
semantic:task>\n      <semantic:task id=\"user_interfaces\" name=\"User interfaces
\">\n        <semantic:incoming>SequenceFlow_1gth9p9 <\/semantic:incoming>\n      <
semantic:outgoing>SequenceFlow_1svtfoi <\/semantic:outgoing>\n      <\/semantic:task
>\n      <semantic:task id=\"functional_requirements\" name=\"Functional
requirements\">\n        <semantic:incoming>SequenceFlow_1svtfoi <\/semantic:incoming
>\n        <semantic:outgoing>SequenceFlow_1nwmtnw <\/semantic:outgoing>\n      <\/
semantic:task>\n      <semantic:task id=\"general_description\" name=\"General
description\">\n        <semantic:incoming>SequenceFlow_0zbcd98 <\/semantic:incoming
>\n        <semantic:outgoing>SequenceFlow_1gth9p9 <\/semantic:outgoing>\n      <\/
semantic:task>\n      <semantic:task id=\"non-functional\" name=\"Non-functional
requirements\">\n        <semantic:incoming>SequenceFlow_177qn9w <\/semantic:incoming
>\n        <semantic:outgoing>SequenceFlow_0f2x53e <\/semantic:outgoing>\n      <\/
semantic:task>\n      <semantic:startEvent id=\"StartEvent_srs\">\n        <semantic
:outgoing>SequenceFlow_0zbcd98 <\/semantic:outgoing>\n      <\/semantic:startEvent>\n
    <semantic:sequenceFlow id=\"SequenceFlow_0zbcd98\" sourceRef=\"StartEvent_srs
\" targetRef=\"general_description\" />\n      <semantic:sequenceFlow id=\"
SequenceFlow_1gth9p9\" sourceRef=\"general_description\" targetRef=\"user_interfaces
\" />\n      <semantic:sequenceFlow id=\"SequenceFlow_1svtfoi\" sourceRef=\"
user_interfaces\" targetRef=\"functional_requirements\" />\n      <semantic:
```

Code snippet

```
sequenceFlow id=\"SequenceFlow_1nwmtnw\" sourceRef=\"functional_requirements\"
targetRef=\"system_interfaces\" />\n      <semantic:sequenceFlow id=\"
SequenceFlow_177qn9w\" sourceRef=\"system_interfaces\" targetRef=\"non-functional\"
/>\n      <semantic:endEvent id=\"EndEvent_srs\">\n        <semantic:incoming>
SequenceFlow_0f2x53e <\/semantic:incoming>\n      <\/semantic:endEvent>\n      <
semantic:sequenceFlow id=\"SequenceFlow_0f2x53e\" sourceRef=\"non-functional\"
targetRef=\"EndEvent_srs\" />\n    <\/semantic:subProcess>\n    <semantic:task id=\"
Task_1gcojbg\" name=\"Send SRS\">\n      <semantic:dataOutputAssociation id=\"
DataOutputAssociation_0uhji0n\">\n        <semantic:targetRef>
DataObjectReference_0bzha3l <\/semantic:targetRef>\n      <\/semantic:
dataOutputAssociation>\n      <semantic:dataOutputAssociation id=\"
DataOutputAssociation_1aymnes\">\n        <semantic:targetRef>
DataObjectReference_1qsfryz <\/semantic:targetRef>\n      <\/semantic:
dataOutputAssociation>\n    <\/semantic:task>\n    <semantic:dataObjectReference id
=\"DataObjectReference_0bzha3l\" name=\"SRS (final_version)\" dataObjectRef=\"
DataObject_158es7x\" />\n    <semantic:dataObject id=\"DataObject_158es7x\" />\n
<semantic:startEvent id=\"StartEvent_1l78x47\">\n      <semantic:outgoing>
SequenceFlow_0xwa7k6 <\/semantic:outgoing>\n    <\/semantic:startEvent>\n    <
semantic:task id=\"Task_1dk5o93\" name=\"Test condictions\">\n      <semantic:
incoming>SequenceFlow_0xwa7k6 <\/semantic:incoming>\n      <semantic:outgoing>
SequenceFlow_1ppw3q5 <\/semantic:outgoing>\n      <semantic:property id=\"
Property_0f32nvf\" name=\"__targetRef_placeholder\" />\n      <semantic:
dataInputAssociation id=\"DataInputAssociation_1k0npz5\">\n        <semantic:
sourceRef>DataObjectReference_1qsfryz <\/semantic:sourceRef>\n        <semantic:
targetRef>Property_0f32nvf <\/semantic:targetRef>\n      <\/semantic:
dataInputAssociation>\n      <semantic:dataInputAssociation id=\"
DataInputAssociation_0g37i7u\">\n        <semantic:sourceRef>
DataObjectReference_1ty9x2j <\/semantic:sourceRef>\n        <semantic:targetRef>
Property_0f32nvf <\/semantic:targetRef>\n      <\/semantic:dataInputAssociation>\n
  <\/semantic:task>\n    <semantic:task id=\"Task_1lzbn01\" name=\"Test case\">\n
    <semantic:incoming>SequenceFlow_1ppw3q5 <\/semantic:incoming>\n      <semantic:
outgoing>SequenceFlow_0uxj5a0 <\/semantic:outgoing>\n    <\/semantic:task>\n    <
semantic:task id=\"Task_1gzqf1h\" name=\"Test executions\">\n      <semantic:
incoming>SequenceFlow_0uxj5a0 <\/semantic:incoming>\n      <semantic:outgoing>
SequenceFlow_0k63alt <\/semantic:outgoing>\n    <\/semantic:task>\n    <semantic:task
 id=\"Task_04bjijv\" name=\"Test report\">\n      <semantic:incoming>
SequenceFlow_0k63alt <\/semantic:incoming>\n      <semantic:outgoing>
SequenceFlow_10rsy19 <\/semantic:outgoing>\n    <\/semantic:task>\n    <semantic:
endEvent id=\"EndEvent_0hwer4x\">\n      <semantic:incoming>SequenceFlow_10rsy19 <\/
semantic:incoming>\n    <\/semantic:endEvent>\n    <semantic:dataObjectReference id
=\"DataObjectReference_1qsfryz\" name=\"SRS\" dataObjectRef=\"DataObject_0s4msuh\"
/>\n    <semantic:dataObject id=\"DataObject_0s4msuh\" />\n    <semantic:
sequenceFlow id=\"SequenceFlow_0xwa7k6\" sourceRef=\"StartEvent_1l78x47\" targetRef
=\"Task_1dk5o93\" />\n    <semantic:sequenceFlow id=\"SequenceFlow_1ppw3q5\"
sourceRef=\"Task_1dk5o93\" targetRef=\"Task_1lzbn01\" />\n    <semantic:sequenceFlow
 id=\"SequenceFlow_0uxj5a0\" sourceRef=\"Task_1lzbn01\" targetRef=\"Task_1gzqf1h\"
/>\n    <semantic:sequenceFlow id=\"SequenceFlow_0k63alt\" sourceRef=\"Task_1gzqf1h
\" targetRef=\"Task_04bjijv\" />\n    <semantic:sequenceFlow id=\"
SequenceFlow_10rsy19\" sourceRef=\"Task_04bjijv\" targetRef=\"EndEvent_0hwer4x\" />\
n    <semantic:subProcess id=\"SubProcess_03u5bq0\">\n      <semantic:incoming>
SequenceFlow_0edafyc <\/semantic:incoming>\n      <semantic:dataOutputAssociation id
=\"DataOutputAssociation_0co8rut\">\n        <semantic:targetRef>
DataObjectReference_1ty9x2j <\/semantic:targetRef>\n      <\/semantic:
dataOutputAssociation>\n      <semantic:task id=\"Task_srs_analyse\" name=\"SRS
analyze\">\n        <semantic:incoming>SequenceFlow_134kdlu <\/semantic:incoming>\n
      <semantic:outgoing>SequenceFlow_1gq4r46 <\/semantic:outgoing>\n        <
semantic:property id=\"Property_0zic7rn\" name=\"__targetRef_placeholder\" />\n
      <semantic:dataInputAssociation id=\"DataInputAssociation_09cimj6\">\n
        <semantic:sourceRef>DataObjectReference_0bzha3l <\/semantic:sourceRef>\n
        <semantic:targetRef>Property_0zic7rn <\/semantic:targetRef>\n        <\/
semantic:dataInputAssociation>\n      <\/semantic:task>\n      <semantic:task id=\"
Task_0vzbpyd\" name=\"Plan\">\n        <semantic:incoming>SequenceFlow_1gq4r46 <\/
semantic:incoming>\n        <semantic:outgoing>SequenceFlow_0x8iw9o <\/semantic:
outgoing>\n      <\/semantic:task>\n      <semantic:task id=\"Task_1yr5tqh\" name=\"
Design\">\n        <semantic:incoming>SequenceFlow_0x8iw9o <\/semantic:incoming>\n
      <semantic:outgoing>SequenceFlow_1xzlzvq <\/semantic:outgoing>\n      <\/
semantic:task>\n      <semantic:task id=\"Task_1t8wps6\" name=\"Implement\">\n
      <semantic:incoming>SequenceFlow_1xzlzvq <\/semantic:incoming>\n        <
```

Code snippet

```
                semantic:outgoing>SequenceFlow_13sp3ni<\/semantic:outgoing>\n        <\/semantic:task
                >\n      <semantic:exclusiveGateway id=\"ExclusiveGateway_0t3zzdh\" name=\"All user
                story finished?\">\n        <semantic:incoming>SequenceFlow_13sp3ni<\/semantic:
                incoming>\n        <semantic:outgoing>SequenceFlow_134kdlu<\/semantic:outgoing>\n
                        <semantic:outgoing>SequenceFlow_0jq7aqz<\/semantic:outgoing>\n      <\/
                semantic:exclusiveGateway>\n      <semantic:endEvent id=\"EndEvent_0uwdnvs\">\n
                        <semantic:incoming>SequenceFlow_0jq7aqz<\/semantic:incoming>\n      <\/
                semantic:endEvent>\n      <semantic:sequenceFlow id=\"SequenceFlow_134kdlu\" name=\"
                No\" sourceRef=\"ExclusiveGateway_0t3zzdh\" targetRef=\"Task_srs_analyse\" />\n
                     <semantic:sequenceFlow id=\"SequenceFlow_1gq4r46\" sourceRef=\"Task_srs_analyse
                \" targetRef=\"Task_0vzbpyd\" />\n      <semantic:sequenceFlow id=\"
                SequenceFlow_0x8iw9o\" sourceRef=\"Task_0vzbpyd\" targetRef=\"Task_1yr5tqh\" />\n
                     <semantic:sequenceFlow id=\"SequenceFlow_1xzlzvq\" sourceRef=\"Task_1yr5tqh\"
                targetRef=\"Task_1t8wps6\" />\n      <semantic:sequenceFlow id=\"
                SequenceFlow_13sp3ni\" sourceRef=\"Task_1t8wps6\" targetRef=\"
                ExclusiveGateway_0t3zzdh\" />\n      <semantic:sequenceFlow id=\"
                SequenceFlow_0jq7aqz\" name=\"Yes\" sourceRef=\"ExclusiveGateway_0t3zzdh\" targetRef
                =\"EndEvent_0uwdnvs\" />\n    <\/semantic:subProcess>\n    <semantic:startEvent id
                =\"StartEvent_dev\">\n      <semantic:outgoing>SequenceFlow_0edafyc<\/semantic:
                outgoing>\n    <\/semantic:startEvent>\n    <semantic:sequenceFlow id=\"
                SequenceFlow_0edafyc\" sourceRef=\"StartEvent_dev\" targetRef=\"SubProcess_03u5bq0\"
                 />\n    <semantic:dataObjectReference id=\"DataObjectReference_1ty9x2j\" name=\"
                Code\" dataObjectRef=\"DataObject_0tuygye\" />\n    <semantic:dataObject id=\"
                DataObject_0tuygye\" />\n  <\/semantic:process>"
    ],
    "participants": [
        {
            "id": "Client",
            "name": "Client",
            "processRef": "Procces_of_client"
        },
        {
            "id": "Software_engineers",
            "name": "Software_engineers",
            "processRef": "Procces_of_software_eng"
        }
    ],
    "Process": [
        [
            {
                "id": "Process_of_client",
                "isExecutable": true
            },
            {
                "id": "Process_of_software_eng",
                "isExecutable": false
            }
        ],
        [
            [
                [
                    {
                        "id": "idea",
                        "name": "idea"
                    },
                    {
                        "id": "Check_SRS",
                        "name": "Check_SRS"
                    },
                    {
                        "id": "Make_ciomments",
                        "name": "Make comments"
                    }
                ],
                [
                    {
                        "id": "Elicitaion",
                        "name": "Requirments elicitation "
```

```
52                      },
53                      {
54                          "id": "system_interfaces",
55                          "name": "System interfaces"
56                      },
57                      {
58                          "id": "user_interfaces",
59                          "name": "User interfaces"
60                      },
61                      {
62                          "id": "functional_requirements",
63                          "name": "Functional requirements"
64                      },
65                      {
66                          "id": "general_description",
67                          "name": "General description"
68                      },
69                      {
70                          "id": "non-functional",
71                          "name": "Non-functional requirements"
72                      },
73                      {
74                          "id": "Task_1gcojbg",
75                          "name": "Send SRS"
76                      },
77                      {
78                          "id": "Task_1dk5o93",
79                          "name": "Test condictions"
80                      },
81                      {
82                          "id": "Task_1lzbn01",
83                          "name": "Test case"
84                      },
85                      {
86                          "id": "Task_1gzqf1h",
87                          "name": "Test executions"
88                      },
89                      {
90                          "id": "Task_04bjijv",
91                          "name": "Test report"
92                      },
93                      {
94                          "id": "Task_srs_analyse",
95                          "name": "SRS analyze"
96                      },
97                      {
98                          "id": "Task_0vzbpyd",
99                          "name": "Plan"
100                     },
101                     {
102                         "id": "Task_1yr5tqh",
103                         "name": "Design"
104                     },
105                     {
106                         "id": "Task_1t8wps6",
107                         "name": "Implement"
108                     }
109                 ]
110             ],
111             [
112                 [
113                     [
114                         "connect_client_start"
115                     ],
116                     {},
117                     [
118                         "SequenceFlow_0zoiw1p"
119                     ],
```

Code snippet

```
                    [
                        "SequenceFlow_1oh7f1c",
                        "SequenceFlow_0i9ge2s"
                    ],
                    [
                        "SequenceFlow_1nwmtnw"
                    ],
                    [
                        "SequenceFlow_1gth9p9"
                    ],
                    [
                        "SequenceFlow_1svtfoi"
                    ],
                    [
                        "SequenceFlow_0zbcd98"
                    ],
                    [
                        "SequenceFlow_177qn9w"
                    ],
                    [],
                    [
                        "SequenceFlow_0xwa7k6"
                    ],
                    [
                        "SequenceFlow_1ppw3q5"
                    ],
                    [
                        "SequenceFlow_0uxj5a0"
                    ],
                    [
                        "SequenceFlow_0k63alt"
                    ],
                    [
                        "SequenceFlow_134kdlu"
                    ],
                    [
                        "SequenceFlow_1gq4r46"
                    ],
                    [
                        "SequenceFlow_0x8iw9o"
                    ],
                    [
                        "SequenceFlow_1xzlzvq"
                    ]
                ],
                [
                    {},
                    [
                        "SequenceFlow_0b8wnpv"
                    ],
                    {},
                    [
                        "SequenceFlow_0jkwt9n"
                    ],
                    [
                        "SequenceFlow_177qn9w"
                    ],
                    [
                        "SequenceFlow_1svtfoi"
                    ],
                    [
                        "SequenceFlow_1nwmtnw"
                    ],
                    [
                        "SequenceFlow_1gth9p9"
                    ],
                    [
                        "SequenceFlow_0f2x53e"
```

```
188                         ],
189                         [],
190                         [
191                             "SequenceFlow_1ppw3q5"
192                         ],
193                         [
194                             "SequenceFlow_0uxj5a0"
195                         ],
196                         [
197                             "SequenceFlow_0k63alt"
198                         ],
199                         [
200                             "SequenceFlow_10rsy19"
201                         ],
202                         [
203                             "SequenceFlow_1gq4r46"
204                         ],
205                         [
206                             "SequenceFlow_0x8iw9o"
207                         ],
208                         [
209                             "SequenceFlow_1xzlzvq"
210                         ],
211                         [
212                             "SequenceFlow_13sp3ni"
213                         ]
214                     ]
215                 ]
216             ]
217         ]
218     }
219 }
```

Listing A.1: *Code snippet* da estrutura da informação de um ficheiro BPMN guardada na base de dados