Universidade do Minho
Escola de Engenharia
Departamento de Sistemas de Informação

Carina Daniela Batoca Campos

Estudo de Interdependências e Rastreabilidade no Processo RUP: Análise do Micro-Processo V-Model e do Método 4SRS

Janeiro de 2013

Universidade do Minho
Escola de Engenharia
Departamento de Sistemas de Informação

Carina Daniela Batoca Campos

Estudo de Interdependências e Rastreabilidade no Processo RUP: Análise do Micro-Processo V-Model e do Método 4SRS

Dissertação de Mestrado
Mestrado em Engenharia e Gestão de Sistemas de Informação

Trabalho efetuado sob a orientação do
Professor Doutor Ricardo J. Machado

Janeiro de 2013

DECLARAÇÃO

Nome: Carina Daniela Batoca Campos

Endereço Eletrónico: carina.campos13@gmail.com **Telefone:** +351 966532728

Nº. do Bilhete de Identidade: 13496957

Título da Dissertação de Mestrado: Estudo de Interdependências e Rastreabilidade no Processo RUP: Análise do Micro-Processo V-Model e do Método 4SRS

Orientador: Professor Doutor Ricardo J. Machado

Ano de conclusão: 2013

Designação do Mestrado: Mestrado em Engenharia e Gestão de Sistemas de Informação

Universidade do Minho, _____ / _____ / _____

Assinatura: _____

Agradecimentos

A realização deste trabalho não teria sido possível sem o apoio e o contributo de algumas pessoas. Assim sendo, agradeço a todos aqueles que contribuíram para a concretização desta dissertação, especialmente:

Ao meu orientador, Professor Doutor Ricardo J. Machado, pela orientação, partilha de conhecimentos e revisão do trabalho.

À minha mãe, Amélia Baixo, que sempre me acompanhou ao longo da minha vida, pelo apoio incondicional, pela sua paciência e pela confiança depositada em mim.

Ao Pedro Veras, pela constante motivação, carinho e apoio prestado ao longo do desenvolvimento deste trabalho.

Finalmente, um obrigado a todos os meus familiares, amigos, professores e colegas que me ajudaram a ultrapassar todas as dificuldades ao longo deste percurso académico.

Obrigada a todos...

Resumo

Dado que a maioria dos artefactos desenvolvidos durante o processo de engenharia de requisitos se relacionam e se afetam mutuamente de diversas formas, é emergente a necessidade de identificar as interdependências entre os artefactos, de forma a conhecer detalhadamente como estes se relacionam ao longo do desenvolvimento de *software*.

Sendo o *Rational Unified Process* (RUP) um processo que fornece as melhores práticas e orientações para o desenvolvimento de *software* bem sucedido, conhecer pormenorizadamente como os seus elementos se relacionam traz bastante utilidade. Este trabalho de investigação tem como propósito efetuar uma sistematização das interdependências existentes no processo RUP e na sua utilização para estudar um micro-processo (*V-Model*) e um método (*Four Step Rule Set* - 4SRS) do ponto de vista da cobertura que garantem relativamente às recomendações do RUP.

Desta forma, primeiramente pretende-se estudar os diversos elementos que constituem o RUP, assim como, os seus relacionamentos. E em seguida, com base nesse conhecimento, pretende-se demonstrar como é vantajoso conhecer as interdependências e a rastreabilidade entre os diversos elementos que constituem o RUP, quando se compara um determinado método/modelo de processo com o RUP.

Palavras-chave: engenharia de requisitos, interdependências, rastreabilidade, artefactos, RUP, *V-Model*, 4SRS.

Abstract

Given that most artifacts developed during the requirements engineering process are related and affect each other in different ways, it is emergent the need to identify the interdependencies between artifacts, in order to know in detail how they relate during software development.

Being the Rational Unified Process (RUP) a process that provides best practices and guidelines for successful software development, to know in detail how its elements are related bring enough utility. This research work aims to make a systematization of existing interdependencies in the RUP process and in its use to study a microprocess (V-Model) and a method (Four Step Rule Set - 4SRS) from the viewpoint of ensuring coverage of the recommendations of RUP.

Thus, first intend to study the different elements that constitute the RUP, as well as their relationships. Then, based on that knowledge, it is intended to demonstrate how it is advantageous to know the interdependencies and the traceability between different elements that constitute the RUP, when comparing a given method/process model with RUP.

Keywords: requirements engineering, interdependencies, traceability, artifacts, RUP, V-Model, 4SRS.

Índice

Índice de Figuras

Índice de Tabelas

Acrónimos

AEs *Architectural Elements* – Elementos Arquiteturais

EBT *Event Based Traceability* – Rastreabilidade Baseada em Eventos

FB *Feature-Model Based Approach* – Abordagem Baseada em Modelos de Atributos

FORT *Feature Oriented Requirements Tracing* – Rastreabilidade de Requisitos Orientada a Atributos

4SRS *Four Step Rule Set*

IR *Information Retrieval* – Recuperação da Informação

IREQ *Inter-Requirement*

OCs *Organizational Configurations* – Configurações Organizacionais

RB *Rule Based Approach* – Abordagem Baseada em Regras

RTOM *Requirement-To-Object-Model*

RUP *Rational Unified Process* – Processo Unificado da *Rational*

TI Tecnologias de informação

UML *Unified Modeling Language*

VBRT *Value Based Requirements Traceability* – Rastreabilidade de Requisitos Baseada em Valor

XML Extensible Markup Language

1. Introdução

Este capítulo inicialmente apresenta uma contextualização do tema da dissertação, onde se pretende demonstrar qual a relevância do mesmo. Posteriormente descrevem-se os objetivos que se pretendem atingir com a realização deste trabalho de investigação, assim como, a abordagem metodológica adotada. Por último, procede-se à descrição da organização do documento, onde se apresentam os principais assuntos abordados em cada um dos capítulos que compõem a presente dissertação.

1.1 Contextualização

A Engenharia de requisitos é um processo da engenharia de *software* crucial para o sucesso de qualquer projeto de *software*. A maioria dos requisitos individuais desenvolvidos durante o processo da engenharia de requisitos relacionam-se e afetam-se mutuamente de diversas formas, não podendo assim, ser tratados isoladamente (Carlshamre, Sandahl, Lindvall, Regnell, & Natt och Dag, 2001; Regnell, et al., 2001).

É importante notar que os requisitos não são independentes uns dos outros, pois, há requisitos que só podem ser implementados se outros requisitos forem implementados previamente. O facto dos requisitos se relacionarem e se afetarem mutuamente faz com que seja necessário identificar e controlar as interdependências de requisitos, de forma a evitar erros potencialmente caros ao longo do desenvolvimento do sistema. As interdependências de requisitos não são um problema por si, mas influenciam o número de atividades de desenvolvimento e as decisões feitas durante o processo de engenharia de *software* (Dahlstedt & Persson, 2005).

A rastreabilidade é a base para estudar as interdependências de requisitos ao longo do processo de desenvolvimento (Dahlstedt & Persson, 2003), uma vez que, permite identificar e justificar os artefactos que implementam os requisitos inicialmente formalizados. A rastreabilidade de requisitos apoia a compreensão de por que razão um determinado objeto foi criado, modificado e evoluído (Ramesh & Jarke, 2001). Negligenciar a rastreabilidade pode levar a *software* menos sustentável e a falhas devido às inconsistências e omissões (Winkler & Pilgrim, 2010).

No desenvolvimento de *software* são produzidos vários tipos de artefactos, por exemplo, documentos de especificação de requisitos, descrições da arquitetura, código fonte e casos de

teste (Heindl & Biffl, 2005). Os artefactos, tal como os requisitos, não existem de forma isolada uns dos outros, em vez disso, relacionam-se e afetam-se mutuamente (Heindl & Biffl, 2005).

Durante o desenvolvimento das soluções e também durante a fase de exploração, por questões de manutenção, surge a necessidade de introduzir diversas alterações às decisões de projeto previamente estabelecidas. Essas mudanças devem ser facilmente identificadas, de forma a garantir que os artefactos envolvidos nas alterações sejam conhecidos. Isto faz com que seja importante possuir conhecimento sobre como os diversos artefactos se relacionam, uma vez que, facilita a identificação dos artefactos afetados.

O *Rational Unified Process* (RUP) é um processo abrangente que cobre quase todos os aspetos dos projetos de desenvolvimento de *software* (Hirsch, 2002). Este fornece as melhores práticas e orientações para o desenvolvimento de *software* bem sucedido (Rational Unified Process). Ao longo deste trabalho de investigação será estudado o processo RUP, com o intuito de analisar a rastreabilidade e as interdependências que podem ocorrer entre os diversos elementos, durante os projetos de desenvolvimento de *software*.

Quando as técnicas de análise de interdependências e de rastreabilidade entre requisitos são aplicadas a um processo de desenvolvimento como é o caso do RUP, passam a ser considerados os *work products.* Estes resultam dos diversos cruzamentos entre as fases e as disciplinas do RUP e vão estar interligados em diversos momentos ao longo de todo o processo de desenvolvimento.

O propósito deste trabalho consiste na sistematização das interdependências existentes no processo RUP e na sua utilização para estudar um micro-processo (*V-Model)* e um método (4SRS) do ponto de vista da cobertura que garantem relativamente às recomendações do RUP.

Neste trabalho de investigação pretende-se, numa primeira fase, efetuar uma análise ao RUP com o objetivo de estudar os diversos elementos que o constituem. E posteriormente, com base nessa análise, pretende-se demonstrar que os diversos *work products* que constituem o RUP são interdependentes de várias tarefas, atividades, disciplinas e fases.

Numa segunda fase, e depois de analisadas as diversas interdependências que podem ocorrer no RUP, pretende-se demonstrar que os conhecimentos adquiridos ao longo de todo o estudo de interdependências e rastreabilidade são vantajosos quando se compara um determinado método/modelo de processo com o RUP. Para tal, serão elaboradas tabelas para analisar a cobertura do micro-processo *V-Model* e do método 4SRS nos diversos elementos que constituem

o RUP, permitindo verificar se as recomendações do RUP estão a ser seguidas pelo método/modelo de processo adotado.

1.2 Objetivos

Conforme se referiu anteriormente, o propósito deste trabalho consiste na sistematização das interdependências existentes no processo RUP e na sua utilização para estudar um micro-processo (*V-Model)* e um método (4SRS) do ponto de vista da cobertura que garantem relativamente às recomendações do RUP.

O presente trabalho tem um carácter teórico e aplicativo. Sobre o ponto de vista teórico, será efetuada uma análise ao RUP, com o objetivo de estudar os diferentes elementos que o constituem e perceber de que forma eles se relacionam entre si. Em relação ao ponto de vista aplicacional, serão elaboradas tabelas para analisar a cobertura do micro-processo *V-Model* e do método 4SRS nos diversos elementos que constituem o RUP. Essas tabelas serão elaboradas com base nas diversas análises efetuadas ao RUP e com a finalidade de demonstrar as vantagens de possuir conhecimento sobre as interdependências e a rastreabilidade, quando se avalia a cobertura de um determinado método/modelo de processo com o RUP.

Assim, este trabalho tem como objetivos:

1. Analisar os diferentes elementos do RUP de forma a sistematizar o relacionamento entre os seus artefactos ao longo das fases e pertencentes às diversas disciplinas, recorrendo a técnicas de interdependência e rastreabilidade.

2. Com base na sistematização obtida relativamente ao RUP, proceder a uma análise de cobertura RUP do micro-processo *V-Model* e do método 4SRS.

Atingidos todos os objetivos, o resultado esperado para este trabalho de investigação consiste em demonstrar como é vantajoso conhecer as interdependências e a rastreabilidade entre os diversos elementos que constituem o RUP, quando se compara um determinado método/modelo de processo com o RUP.

1.3 Abordagem de investigação

Nos projetos de investigação é fundamental utilizar-se uma abordagem metodológica adequada, uma vez que, a mesma vai contribuir para a concretização dos objetivos definidos para o projeto.

Segundo Nunamaker, Chen e Purdin (1991) uma metodologia de investigação consiste na combinação de processos, métodos e ferramentas que são usados na realização da investigação sobre um determinado tema.

Para o sucesso e conclusão de um projeto de investigação é importante escolher métodos apropriados e sistemáticos (Berndtsson, Hansson, Olsson, & Lundell, 2008). Segundo Berndtsson, Hansson, Olsson e Lundell (2008) um método refere-se a uma abordagem organizada para a resolução de problemas que inclui: (1) recolher dados, (2) formular uma hipótese ou proposição, (3) testar a hipótese, (4) interpretar os resultados e (5) tirar conclusões.

Existem diferentes abordagens metodológicas que podem ser adotados nos projetos de investigação, a sua escolha efetua-se com base na natureza do problema e no que se pretende realizar (Berndtsson, Hansson, Olsson, & Lundell, 2008).

A escolha de uma abordagem metodológica adequada vai permitir executar de forma mais conseguida o estudo de investigação, servindo de guia para dar resposta à questão de investigação a que o trabalho pretende responder.

Com base na natureza do problema e no que se pretende realizar, a abordagem metodológica utilizada para este trabalho de investigação é a *Design Science Research*, uma vez que, se pretende com o artefacto criado a resolução de um problema real e uma contribuição para a teoria em causa.

O princípio fundamental da *Design Science Research* é que "o conhecimento e a compreensão de um problema e a sua solução são adquiridos na construção e aplicação de um artefacto inovador" (Hevner & Chatterjee, 2010).

Design Science Research em sistemas de informação, "cria e avalia artefactos de TI que são planeados para resolver problemas organizacionais identificados" (Hevner, March, Park, & Ram, 2004).

Segundo Vaishnavi e Jr. (2008) esta metodologia encontra-se dividida em cinco fases. Em seguida serão brevemente descritas cada uma das fases.

A primeira fase diz respeito à **consciencialização do problema**. Nesta fase efetua-se uma descrição do problema encontrado, assim como, justifica-se a importância de criar um artefacto para resolver o problema em causa. O *output* desta fase é uma proposta para um novo esforço de investigação (Vaishnavi & Jr., 2008).

A segunda fase consiste na **sugestão**, esta fase ocorre a seguir à proposta resultante da primeira fase, e está intimamente ligada a ela. A sugestão é uma fase criativa essencial, em que o novo artefacto é profetizado com base nos conhecimentos/teorias existentes (Vaishnavi & Jr., 2008).

A terceira fase, o **desenvolvimento**, consiste no desenvolvimento e implementação da sugestão mencionada na fase anterior, resultando assim um artefacto. Os artefactos desenvolvidos podem ser construtos, modelos, métodos e instanciações (Hevner & Chatterjee, 2010).

A quarta fase é a **avaliação**, uma vez construído o artefacto este é avaliado para verificar se o mesmo é a solução para o problema mencionado.

Por fim, a quinta fase, as **conclusões**. Esta fase diz respeito aos resultados obtidos com o artefacto, assim como, os conhecimentos adquiridos ao longo do processo de desenvolvimento do mesmo.

Neste trabalho foram adotadas as cinco fases desta metodologia (Vaishnavi & Jr., 2008) que serviram de guias na resolução do problema apresentado.

Este trabalho, como já foi referido anteriormente, pretende resolver um problema real que ocorre durante o desenvolvimento das soluções e também durante a fase de exploração. O problema advém pelo facto de frequentemente ocorrerem diversas mudanças nos projetos e da maioria dos artefactos se relacionarem e se afetarem mutuamente. O que faz com que seja necessário identificar tais mudanças, de forma a garantir que os artefactos envolvidos nas alterações sejam conhecidos. Isto evidencia a importância de possuir conhecimento sobre como os diversos artefactos se relacionam, uma vez que, facilita a identificação dos artefactos afetados.

A sugestão apresentada neste trabalho para resolver o problema mencionado, consiste em efetuar uma sistematização das interdependências existentes entre os diversos elementos que constituem o processo RUP. Mostrando assim, que ao longo do processo de desenvolvimento de *software* existem vários artefactos interligados que requerem cuidados especiais ao sofrerem alterações. Com este trabalho pretende-se também demonstrar como é vantajoso possuir

conhecimento sobre as diversas interdependências existentes no processo RUP, quando se analisa a cobertura de um determinado método/modelo de processo com o RUP. Através da análise de cobertura do micro-processo *V-Model* e do método 4SRS nos diversos elementos que constituem o RUP será demonstrada a vantagem da sistematização das interdependências existentes no RUP.

O facto da criação dos artefactos, nesta metodologia, se basear em teorias existentes que são aplicadas, testadas, modificadas e ampliadas através da experiência, criatividade, intuição e capacidade para resolver problemas (Hevner, March, Park, & Ram, 2004), faz com que a concretização do 1º objetivo deste trabalho seja bastante importante. Uma vez que, é através desse objetivo que é reunido grande parte do conhecimento necessário para criar as tabelas pretendidas para este estudo. Visto a relevância desse objetivo foi necessário efetuar diversas análises ao RUP com o intuito de perceber quais os elementos que o constituem e como estes se relacionam entre si. Para minimizar possíveis riscos, as diversas pesquisas efetuadas para este trabalho de investigação foram realizadas com base em artigos científicos, teses, dissertações, atas de conferências e livros. Ao longo da pesquisa houve sempre a preocupação de verificar se os documentos eram de qualidade, verificando sempre o número de citações e o local onde foram publicados.

Para concretizar o 1º objetivo desta dissertação foi efetuada uma sistematização das interdependências existentes no RUP, com base nas diversas análises efetuadas aos elementos do RUP e recorrendo a técnicas de interdependência e rastreabilidade. As tabelas resultantes dessa sistematização permitem avaliar a cobertura de um determinado método/modelo de processo com o RUP.

Com base nos resultados do 1º objetivo procedeu-se a uma análise de cobertura RUP do micro-processo *V-Model* e do método 4SRS, concretizando-se assim, o 2º objetivo desta dissertação.

Por fim, com base nos resultados obtidos com a concretização do 2º objetivo, foram discutidas as respetivas conclusões. De forma, a demonstrar como é vantajoso possuir conhecimento sobre as interdependências e a rastreabilidade entre os diversos elementos que constituem o RUP, quando se analisa a cobertura de um determinado método/modelo de processo com o RUP.

1.4 Organização do documento

Nesta secção descreve-se a organização deste documento e sintetizam-se os cinco capítulos que o constituem.

Neste primeiro capítulo apresenta-se um breve enquadramento do estudo, identificando-se o problema e propósito de investigação, assim como os objetivos que se pretendem alcançar com a dissertação, a abordagem de investigação adotada e a organização do documento.

O segundo capítulo apresenta a revisão da literatura que identifica e descreve os principais assuntos relacionados com o tema em estudo.

No terceiro capítulo, inicialmente efetua-se uma síntese do processo RUP com o propósito de estudar os diversos elementos que o constituem. E em seguida, apresentam-se as tabelas e os esquemas desenvolvidos para mostrar a rastreabilidade e as diversas interdependências que se podem verificar entre os elementos do RUP.

No quarto capítulo apresentam-se diversas tabelas, elaboradas com base nas tabelas do capítulo anterior. Sendo que, inicialmente efetua-se uma síntese do micro-processo *V-Model* e do método 4SRS e posteriormente analisa-se a cobertura deste método/modelo de processo com o RUP, através de tabelas elaboradas para esse propósito.

No quinto capítulo efetua-se uma síntese do trabalho desenvolvido, assim como, uma discussão dos resultados obtidos e por fim, apresentam-se algumas sugestões de trabalhos futuros a desenvolver.

2. Revisão da literatura

2.1 Introdução

Este capítulo apresenta a revisão da literatura que identifica e descreve os principais assuntos que fundamentam este trabalho de investigação, no que respeita a técnicas de rastreabilidade e interdependência. Apesar da esmagadora maioria das técnicas descritas na literatura serem dedicadas à análise de requisitos, mais à frente nesta dissertação algumas das técnicas serão aplicadas a artefactos do RUP. A revisão da literatura desta dissertação está estruturada em três assuntos fundamentais. A secção 2.2 refere-se à engenharia de requisitos. A secção 2.3 à rastreabilidade de requisitos. E por fim, a secção 2.4 que se refere às interdependências de requisitos.

2.2 Engenharia de Requisitos

Antes de se proceder à definição do conceito de engenharia de requisitos, considera-se relevante a definição de requisitos. Os requisitos constituem a primeira fase do ciclo de vida do desenvolvimento de *software* (Kotonya & Sommeiville, 1996). Estes "são declarações que refletem as necessidades dos clientes e dos utilizadores de um sistema" (Kotonya & Sommeiville, 1996).

Segundo Rzepka (1985, citado por Kotonya & Sommeiville, 1996) os requisitos podem ser divididos em duas categorias principais: requisitos funcionais e não-funcionais. Os requisitos funcionais "são requisitos diretamente ligados à funcionalidade do *software,* descrevem as funções que o *software* deve executar" (Ávila & Spínola, 2007). Os requisitos não-funcionais "expressam condições que o *software* deve atender ou qualidades específicas que o *software* deve ter. Em vez de informar o que o sistema fará, os requisitos não-funcionais colocam restrições no sistema" (Ávila & Spínola, 2007).

A engenharia de requisitos é o ramo da engenharia de *software* (ZAVE, 1997) que se foca em identificar, analisar, especificar, verificar e gerir os requisitos do sistema (Kuloor & Eberlein, 2003).

A engenharia de requisitos é "um processo essencial no ciclo de vida do desenvolvimento de *software*" (Assawamekin, Sunetnanta, & Pluempitiwiriyawej, 2010). Esta preocupa-se com a identificação dos objetivos a serem alcançados pelo sistema previsto, com a operacionalização de tais

objetivos em serviços e restrições e com a atribuição de responsabilidades pelos requisitos resultantes (Lamsweerde, 2000).

Segundo Nogueira (2009) a engenharia de requisitos fornece um mecanismo adequado para entender o que o cliente deseja, analisar as necessidades, avaliar a viabilidade, negociar uma solução razoável, especificar a solução de maneira não-ambígua, validar a especificação e administrar os requisitos à medida que eles são transformados num sistema em operação.

Paetsch, Eberlein e Maurer (2003) referem que o objetivo da engenharia de requisitos é ajudar a saber o que construir antes do desenvolvimento do sistema começar, a fim de, prevenir trabalhos extras dispendiosos. A necessidade do desenvolvimento de *software* tem como base a resolução de um problema, efetuar um levantamento dos requisitos na fase inicial do projeto vai proporcionar um maior entendimento sobre esse mesmo problema, assim como, antever possíveis dificuldades que possam ocorrer ao longo do desenvolvimento do *software*.

O processo da engenharia de requisitos consiste em cinco atividades principais (Kotonya & Sommerville, 1997, citado por Paetsch, Eberlein, & Maurer, 2003): levantamento, análise e negociação, documentação, validação e gestão.

A atividade de **levantamento** tenta descobrir os requisitos e identificar os limites do sistema ao consultar os *stakeholders* (por exemplo: clientes, programadores, utilizadores) (Paetsch, Eberlein, & Maurer, 2003). Os limites do sistema definem o contexto do sistema. "Compreender o domínio de aplicação, as necessidades do negócio, as restrições do sistema, os *stakeholders* e o problema em si é essencial para compreender o sistema a ser desenvolvido" (Paetsch, Eberlein, & Maurer, 2003).

A atividade de **análise e negociação** "verifica se os requisitos estão de acordo com as necessidades, verifica a consistência (os requisitos não devem ser contraditórios), a integridade (nenhum serviço ou restrição está a faltar) e a viabilidade (os requisitos são viáveis no contexto do orçamento e do tempo disponível para o desenvolvimento do sistema) " (Paetsch, Eberlein, & Maurer, 2003). Esta atividade resolve ainda conflitos nos requisitos (Paetsch, Eberlein, & Maurer, 2003).

O objetivo da atividade de **documentação** é o de "comunicar os requisitos entre os *stakeholders* e os programadores" (Paetsch, Eberlein, & Maurer, 2003). Os requisitos devem ser documentados, a fim de servir de base para o restante processo de desenvolvimento. Essa documentação deve ser feita de forma consistente e seguindo-se um padrão que permita

demonstrar, em vários níveis de detalhe, a especificação dos requisitos levantados (Genvigir, 2009). "Um bom documento de requisitos é inequívoco, completo, correto, compreensível, consistente, conciso e viável" (Paetsch, Eberlein, & Maurer, 2003).

A atividade de **validação** tem como propósito "certificar que os requisitos são uma descrição aceitável do sistema a ser implementado" (Paetsch, Eberlein, & Maurer, 2003). O processo de validação tem como *inputs* os documentos de requisitos, os padrões organizacionais e o conhecimento organizacional. E como *outputs* tem uma lista que contém os problemas relatados com os documentos de requisitos e as ações necessárias para lidar com os problemas relatados (Paetsch, Eberlein, & Maurer, 2003).

O objetivo da atividade de **gestão** é "capturar, armazenar, divulgar e gerir informação" (Paetsch, Eberlein, & Maurer, 2003). Esta atividade deve atender à manutenção da evolução dos requisitos ao longo do processo de desenvolvimento, isto é, inclui todas as atividades preocupadas com a mudança e controlo de versões e rastreamento de requisitos (Paetsch, Eberlein, & Maurer, 2003; Genvigir, 2009). A rastreabilidade de requisitos está incluída nesta atividade (Paetsch, Eberlein, & Maurer, 2003).

Os sistemas de *software* estão sujeitos a contínuas mudanças. Deixar de atender a essas mudanças pode afetar negativamente o sucesso do sistema (Jadallah, Galster, Moussavi, & Ruhe, 2009). É necessário compreender as características do sistema, e como elas estão relacionadas umas com as outras, a fim de perceber qual o impacto de uma simples mudança no sistema (Jadallah, Galster, Moussavi, & Ruhe, 2009).

A fim de evitar falhas ou erros nas fases finais de desenvolvimento é importante entender completamente os requisitos do sistema desde o início do desenvolvimento, bem como, analisa-los cuidadosamente (Kuloor & Eberlein, 2003). Uma compreensão dos requisitos e uma análise adequada, não só tornam a fase de conceção mais simples, como também ajudam a reduzir as falhas que podem ocorrer nas fases posteriores de desenvolvimento (Kuloor & Eberlein, 2003). Durante o desenvolvimento, também é essencial ter conhecimento das relações existentes entre os requisitos e os outros artefactos do sistema, a fim de, evitar erros potencialmente caros (Dahlstedt & Persson, 2005) que podem por em causa o sucesso do sistema. Todos estes fatos mencionados anteriormente, fazem com que a engenharia de requisitos seja tão importante ao longo do desenvolvimento de *software*.

2.3 Rastreabilidade de Requisitos

A rastreabilidade, tal como mostra a Figura 1 e como já foi referido anteriormente, está situada no ramo da engenharia de *software* mais precisamente na atividade de gestão do processo da engenharia de requisitos.

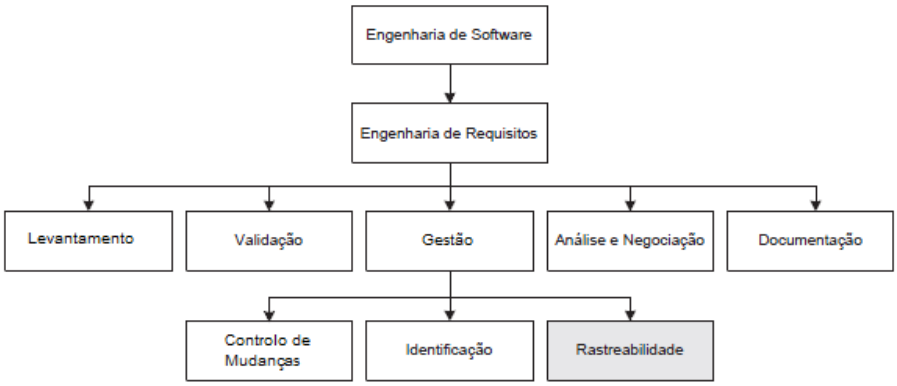


Figura 1 – Alocação da rastreabilidade dentro da engenharia de software

Fonte: Adaptado de (Genvigir, 2009)

A rastreabilidade é considerada por Winkler e Pilgrim (2010) um assunto muito interessante para os programadores de *software* e para os engenheiros de requisitos, uma vez que, através da rastreabilidade estes poderão seguir a história dos artefactos de *software*.

Gotel e Finkelstein (1994) definem a rastreabilidade de requisitos como "a capacidade de descrever e seguir a vida de um requisito, em ambas as direções para a frente e para trás (*forward and backward direction*), isto é, desde as suas origens, através do seu desenvolvimento e especificação, à sua subsequente implementação e utilização". De entre as várias definições encontradas na literatura para o conceito de rastreabilidade, esta definição é a mais referenciada pelos diversos investigadores.

García, Santos e Windels (2008) definem a rastreabilidade como "a capacidade de acompanhar os requisitos, a sua evolução e transformação nos diferentes componentes relacionados com o processo de engenharia, bem como, a gestão das relações entre esses componentes".

Ramesh e Jarke (2001) referem que a rastreabilidade de requisitos visa "garantir o alinhamento contínuo entre os requisitos dos *stakeholders* e a evolução do sistema". Permitindo que esta "sustente a compreensão de por que razão um determinado objeto foi criado, modificado e evoluído" (Ramesh & Jarke, 2001). Estes mencionam a rastreabilidade de requisitos como "uma

característica do sistema, na qual, os requisitos estão claramente ligados às suas fontes e aos artefactos criados durante o ciclo de vida de desenvolvimento do sistema".

Segundo Genvigir (2009) e Zou, Settimi e Cleland-Huang (2010) a rastreabilidade está intimamente associada ao processo de produção de *software*, especificamente aos requisitos e à capacidade de estabelecer vínculos entre esses requisitos e outros artefactos (modelos, documentos, código fonte, sequências de testes ou executáveis) que os satisfaçam.

Sánchez, Alonso, Rosique, Álvarez e Pastor (2011) mencionam que a rastreabilidade de requisitos destina-se a ajudar a determinar o impacto das mudanças na fase de conceção de *software*, a suportar a sua integração, a preservar o conhecimento e a assegurar a qualidade e correção do sistema global.

Como pode ser verificado pelas várias definições encontradas na literatura, a rastreabilidade permite gerir as relações existentes entre os vários artefactos do sistema, uma vez que, através da mesma é possível saber toda a história dos requisitos permitindo assim, uma maior gestão e controlo das mudanças que vão ocorrendo no sistema.

A rastreabilidade de requisitos tem sido identificada na literatura como um fator de qualidade (Ramesh & Jarke, 2001; Winkler & Pilgrim, 2010; Dahlstedt & Persson, 2005; Spanoudakis & Zisman, 2005). Suportar ativamente a rastreabilidade nos projetos de desenvolvimento de *software* pode ajudar a assegurar outras qualidades do *software*, tais como, adequação e compreensibilidade (Winkler & Pilgrim, 2010). Por outro lado, negligenciar a rastreabilidade pode levar a *software* menos sustentável e a falhas devido às inconsistências e omissões (Winkler & Pilgrim, 2010). Dömges e Pohl (1998) referem que negligenciar a rastreabilidade ou capturar insuficientes e/ou não estruturados rastreamentos leva a uma diminuição na qualidade do sistema, provocando revisões e consequentemente, um aumento nos custos do projeto e no tempo.

De acordo com Winkler e Pilgrim (2010), a maior parte da investigação sobre rastreabilidade tem sido feita nas últimas duas décadas pela comunidade da engenharia de requisitos. Ao longo dos últimos anos, a rastreabilidade tem ganho importância, e temas sobre rastreabilidade tornaram-se assunto para investigação em muitas outras áreas de desenvolvimento de *software* (Winkler & Pilgrim, 2010).

Winkler e Pilgrim (2010) também aludem que as práticas de rastreabilidade, em geral, estão longe de ser maduras, pois, ainda é necessário muita investigação nesta área. O que é um

desafio, não só, pela dificuldade das questões de investigação, mas também, pelo facto dos investigadores no campo da rastreabilidade pertencerem a diferentes comunidades de investigação (tais como, engenharia de requisitos, modelação, compreensão do programa, entre outras) onde há pouca comunicação entre elas (Winkler & Pilgrim, 2010).

Panis (2010) refere que "no mundo da engenharia de requisitos parece ser uma conclusão inevitável de que a rastreabilidade é fundamental para o processo de desenvolvimento do produto". Facto, que faz com que esta deva ser incluída e tratada ao longo do processo de desenvolvimento.

Aizenbud-Reshef , Nolan , Rubin e Shaham-Gafni (2006) referem que da perspetiva da gestão de requisitos, a rastreabilidade facilita interligar os requisitos às suas origens e razões, capturando a informação necessária para compreender a evolução dos requisitos e verificar que requisitos foram cumpridos. Uma rastreabilidade completa vai permitir obter custos mais precisos, assim com, determinar listas de mudanças, sem necessitar de depender do programador para conhecer todas as áreas que serão afetadas por essas mudanças (Aizenbud-Reshef , Nolan , Rubin , & Shaham-Gafni , 2006). Todas estas razões fazem com que seja tão importante implementar práticas de rastreabilidade ao longo do desenvolvimento de *software*.

Ao longo dos anos, vários termos relacionados com a rastreabilidade de requisitos foram estabelecidos. Os mais comuns são: rastrear para a frente (*forward traceability*), rastrear para trás (*backward traceability*), pré-rastreabilidade, pós-rastreabilidade, rastreabilidade horizontal e rastreabilidade vertical (Winkler & Pilgrim, 2010).

A capacidade de rastrear um requisito para a frente (*forward traceability*) significa "seguir as ligações de rastreabilidade para os artefactos que foram derivados do artefacto em consideração" (Winkler & Pilgrim, 2010). Rastrear um requisito para trás (*backward traceability*) "refere-se à capacidade de seguir as ligações de rastreabilidade de um artefacto específico de volta às suas origens de onde foi derivado" (Winkler & Pilgrim, 2010). Estas duas capacidades devem estar presentes em todos os tipos de rastreabilidade (Genvigir, 2009).

A Figura 2 apresenta os dois tipos de rastreabilidade introduzidos por Gotel e Finkelstein (1994) pré-rastreabilidade e pós-rastreabilidade.

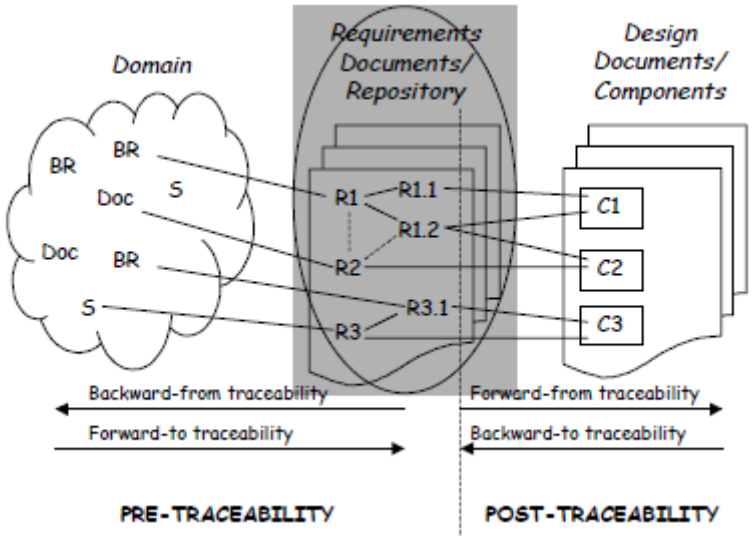


Figura 2 – Pré e Pós-rastreabilidade

Fonte: (Dahlstedt & Persson, 2003)

Pré-rastreabilidade – "refere-se aos aspetos da existência de um requisito antes de ser incluído na especificação de requisitos" (Gotel & Finkelstein, 1994) e "está focada em permitir uma melhor compreensão dos requisitos" (Dahlstedt & Persson, 2005). Dahlstedt e Persson (2003) referem que a pré-rastreabilidade "preocupa-se com a produção de requisitos e foca-se no domínio com o qual se interage quando os requisitos são desenvolvidos e em que o sistema está a ser instalado". Os requisitos devem estar relacionados à sua origem, por exemplo, ao *stakeholder* (S), às regras de negócios (BR), à documentação anterior (Doc) e também a outros requisitos, por exemplo, através da decomposição de requisitos (Dahlstedt & Persson, 2005). "A pré-rastreabilidade é a base para gerir a evolução de um sistema, porque permite o levantamento das partes de especificação que são afetadas por uma mudança específica no pedido suscitado" (Dahlstedt & Persson, 2005).

Pós-rastreabilidade – "refere-se aos aspetos da existência de um requisito a partir do momento em que foi incluído na especificação de requisitos" (Gotel & Finkelstein, 1994) e "está focada em permitir uma melhor compreensão e aceitação do atual *software* de sistema" (Dahlstedt & Persson, 2005). A pós-rastreabilidade preocupa-se em assegurar que todos os requisitos são cumpridos pelo sistema, através da conceção e implementação do sistema, ao relacionar os requisitos ao componente (C) que ajuda a satisfazer aquele requisito específico (Dahlstedt & Persson, 2005). Dahlstedt e Persson (2003) descrevem que "a pós-rastreabilidade preocupa-se com

o desenvolvimento dos requisitos e foca-se no *software* que é desenvolvido com base nos requisitos".

Ramesh e Edwards (1993) introduziram a rastreabilidade horizontal e vertical. A Figura 3 mostra esses dois tipos de rastreabilidade.

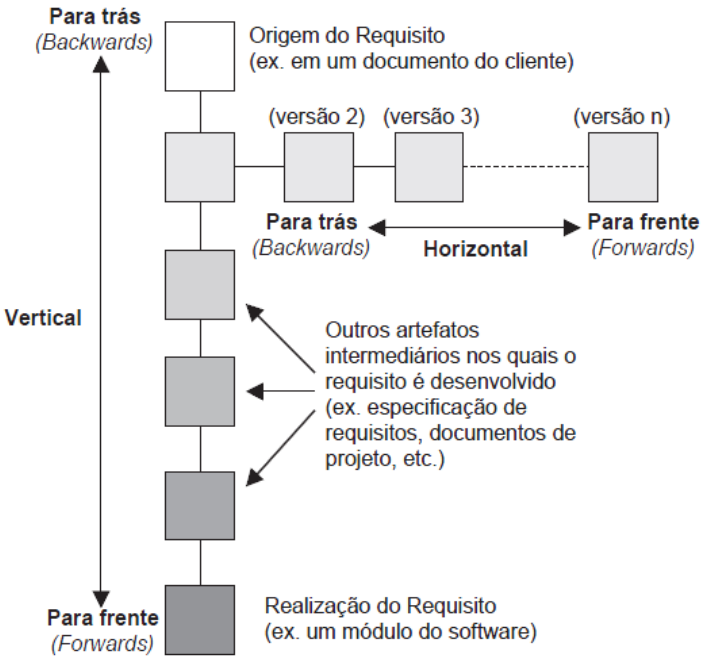


Figura 3 – Rastreabilidade horizontal e vertical

Fonte: (Genvigir, 2009)

Rastreabilidade Horizontal – trata de relacionar versões ou variantes do mesmo tipo de informação, por exemplo, entre requisitos ou entre componentes do sistema (Dahlstedt & Persson, 2005; Winkler & Pilgrim, 2010)

Rastreabilidade Vertical – preocupa-se em rastrear informação entre anteriores e subsequentes fases no processo de desenvolvimento, isto é, entre objetos de informação de diferentes tipos (Dahlstedt & Persson, 2005; Winkler & Pilgrim, 2010). Por exemplo, uma relação entre um requisito e um elemento da conceção (Winkler & Pilgrim, 2010).

Wieringa (1995) mencionou alguns dos benefícios da rastreabilidade, os quais foram baseados em Ramesh et al. (1993). Em seguida apresentam-se alguns desses benefícios agrupados de acordo com as necessidades dos *stakeholders* no processo de desenvolvimento (Wieringa, 1995).

• *Gestão de projetos* – a rastreabilidade vai proporcionar à gestão de projetos vários benefícios, tais como:

-Estimar o impacto de uma mudança nos requisitos;

-Descobrir mais cedo os conflitos entre os requisitos, podendo assim, evitar atrasos inesperados;

-Os requisitos ainda não satisfeitos pela implementação podem ser recolhidos e o trabalho a ser feito para satisfazer esses requisitos restantes pode ser estimado;

-Os sistemas futuros terão o tempo de desenvolvimento reduzido e os esforços, uma vez que, as decisões de implementação passadas podem ser reutilizadas.

Em geral, um sistema de informações de rastreamento ajuda a gestão de projetos a acompanhar o estado do projeto.

• *Cliente* – se o sistema de informações de rastreamento registar que requisitos são satisfeitos e por que partes da implementação e quais os testes que devem ser realizados para verificar a presença de um requisito, então os benefícios para o cliente são:

-Permitir avaliar a qualidade do produto em relação aos requisitos do utilizador;

-Permitir verificar que adicionar funcionalidades a um sistema que não foram solicitadas pelos utilizadores tem de ser evitado, porque todos os componentes da implementação podem ser rastreados a pelo menos um requisito;

-Os requisitos do utilizador são ligados aos requisitos de conceção, de modo que, o pessoal de desenvolvimento possa manter o seu foco nos requisitos do utilizador que eles estão a tentar satisfazer.

• *Designer* - o sistema de informações de rastreamento deve registar os resultados da fase de conceção, a justificação dos resultados, as alternativas consideradas e as suposições feitas numa decisão. Se estes forem registados juntamente com as ligações entre os requisitos e a conceção, então os benefícios para o *designer* são:

-Puder verificar mais facilmente se uma conceção satisfaz os requisitos;

-Estimar o impacto de uma mudança nos requisitos;

-Compreender a razão pela qual uma certa conceção foi aceite e outra foi rejeitada, mesmo quando a conceção foi produzida há muito tempo por um outro *designer* não presente;
-Puder reutilizar os componentes de conceção em outros projetos, porque os pressupostos em que o componente irá trabalhar são registados.
O sistema de informações de rastreamento é um tipo de "memória corporativa" que pode ser usado para acelerar a tomada de decisões em projetos de desenvolvimento futuros.

• *Gestor de manutenção (Maintainer)* - a rastreabilidade vai proporcionar ao gestor de manutenção os seguintes benefícios:
-Estimar o impacto de uma mudança nos requisitos em outros requisitos (descoberta de conflitos, dependências);
-Estimar o impacto de uma mudança nos requisitos na implementação.

Dados os benefícios que a rastreabilidade pode proporcionar ao longo do processo de desenvolvimento de *software*, esta não deve ser negligenciada, muito pelo contrário. Esta deve ser incluída e tratada ao longo de todo o processo de desenvolvimento, pois, vai trazer bastantes benefícios para o mesmo.

Ao longo dos últimos anos, as comunidades da engenharia de sistemas e de *software* têm desenvolvido várias abordagens e técnicas para tratar diversos aspetos da rastreabilidade (Spanoudakis & Zisman, 2005; Zou, Settimi, & Cleland-Huang, 2010). O desenvolvimento e o uso de técnicas para rastrear requisitos tiveram início em 1970 (Aizenbud-Reshef , Nolan , Rubin , & Shaham-Gafni , 2006).

O primeiro método usado para expressar e manter a rastreabilidade foi as referências cruzadas (Aizenbud-Reshef , Nolan , Rubin , & Shaham-Gafni , 2006). Segundo Winkler e Pilgrim (2010) uma referência cruzada na maioria das suas formas triviais pode ser incorporada e representada numa linguagem natural, como por exemplo, "ver a revisão 1.2 da especificação dos requisitos de *software*, secção 3.14" (Winkler & Pilgrim, 2010), ou anotada usando características das referências cruzadas utilizadas no processador de texto (Winkler & Pilgrim, 2010). As referências cruzadas podem ser intuitivamente compreendidas por qualquer leitor. Contudo, a visão sobre as relações de rastreabilidade é muito restrita na sua representação (Winkler & Pilgrim, 2010).

Desde essa altura, muitas outras técnicas têm sido utilizadas para representar as relações de rastreabilidade. Algumas dessas abordagens/técnicas são:

- **Meta-Modelo de Ramesh e Jarke**

Ramesh e Jarke (2001) criaram um meta-modelo para a rastreabilidade de requisitos, Figura 4. Esse meta-modelo resultou de uma série de estudos empíricos e fornece os princípios fundamentais da linguagem para categorizar e descrever os modelos de rastreabilidade em mais detalhe (Ramesh & Jarke, 2001).

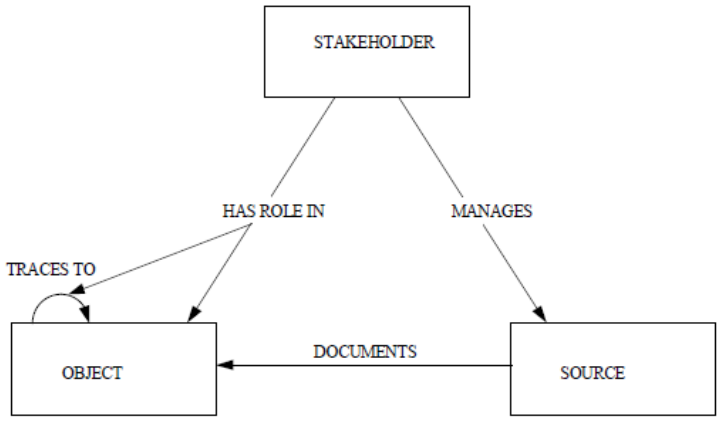


Figura 4 – Meta-modelo para a rastreabilidade de requisitos proposto por Ramesh e Jarke

Fonte: (Ramesh & Jarke, 2001)

Cada entidade e ligação no meta-modelo pode ser especializada e instanciada para criar modelos de organização ou projetos específicos de rastreabilidade (Ramesh & Jarke, 2001).

Segundo Ramesh e Jarke (2001) o meta-modelo pode ser utilizado para representar as três dimensões da engenharia de requisitos (objetos, *stakeholders* e fontes) e as seis dimensões da informação de rastreabilidade (Qual?; Quem?; Onde?; Como?; Porque? e Quando?). Em seguida descrevem-se as dimensões da informação de rastreabilidade de acordo com Ramesh e Jarke (2001).

1. **Qual** a informação que é representada – incluindo atributos importantes ou características da informação?

   No modelo, os **objetos** (*objects*) representam os *inputs* e *outputs* do processo de desenvolvimento do sistema. Os objetos podem ser de vários tipos: requisitos, hipóteses, componentes de sistema, alternativas, fatores de sucesso, entre outros (Ramesh & Jarke, 2001).

Estes representam os principais elementos conceptuais entre os quais a rastreabilidade é mantida durante as várias fases do ciclo de vida. A rastreabilidade sobre os vários objetos é representada pela relação *Traces-to.*

2. **Quem** são os *stakeholders* que desempenham diferentes papéis na criação, manutenção e utilização dos vários objetos e relações de rastreabilidade entre eles?

   No modelo, os ***stakeholders*** representam os agentes envolvidos no desenvolvimento do sistema e na manutenção das atividades do ciclo de vida. Os *stakeholders* podem ser, por exemplo, gestores de projetos, analistas de sistemas, *designers*, entre outros. Estes *stakeholders* atuam em diferentes papéis ou capacidades na criação e utilização dos vários objetos conceptuais e nas relações de rastreabilidade (Ramesh & Jarke, 2001).

3. **Onde** é representada – em termos de fontes?

   Todos os objetos são documentados por **fontes** *(sources)* que podem ser meios físicos, tais como, documentos ou coisas intangíveis, tais como, referências a pessoas ou políticas não documentadas e procedimentos (Ramesh & Jarke, 2001). São os *stakeholders* que criam, mantêm e usam as fontes.

4. **Como** esta informação é representada?

   As fontes, como mencionado anteriormente, podem ser físicas ou intangíveis. Além disso, podem ser representadas em diferentes níveis de formalidade. Por exemplo, as especificações de requisitos podem ser documentos de texto, enquanto que outros documentos de conceção podem ser representados em múltiplos formatos, tais como, gráficos e texto (Ramesh & Jarke, 2001).

5. **Porque** um certo objeto conceptual foi criado, modificado e evoluído?

   A razão por trás da criação, modificação e evolução dos vários objetos conceptuais pode ser representada como uma especialização do objeto meta-classe (Ramesh & Jarke, 2001). Então, ela pode ser ligada ao objeto conceptual (usando uma especialização da ligação *Traces-to*).

6. **Quando** esta informação foi capturada, modificada e evoluída?

   Informações temporais relevantes sobre qualquer uma das entidades ou ligações no modelo podem ser representadas como seus atributos. Por exemplo, a frequência ou o tempo/duração em que um requisito foi criado, modificado, revisto ou justificado por uma razão específica (Ramesh & Jarke, 2001).

- **Rastreabilidade de requisitos baseada em valor -** *Value Based Requirements Traceability (VBRT)*

Esta abordagem fornece suporte técnico para executar o rastreamento dos requisitos, bem como, ter em conta considerações de valor e custo (Rochimah, Kadir, & Abdullah, 2007).

O objetivo do VBRT é "identificar rastreamentos baseados em requisitos priorizados e assim identificar que rastreamentos são mais importantes e valiosos que outros" (Heindl & Biffl, 2005).

Este processo é constituído por cinco etapas (Heindl & Biffl, 2005):

1. Definição de requisitos - nesta etapa é atribuído a cada requisito um identificador único.
2. Priorização de requisitos - nesta etapa é estimado o valor, o risco e os esforços de cada requisito.
3. Empacotamento de requisitos - nesta etapa são identificados grupos de requisitos.
4. Ligação de requisitos - nesta etapa são estabelecidas ligações de rastreabilidade entre os requisitos e outros artefactos.
5. Avaliação - nesta etapa podem-se usar os rastreamentos gerados para vários fins, tais como, estimar o impacto de mudar determinados requisitos.

Heindl e Biffl (2005) mencionam que o VBRT é um bom passo para resolver problemas de rastreabilidade de requisitos que exigem elevados esforços e elevada complexidade.

- **Rastreabilidade de requisitos orientada a *atributos* -** *Feature Oriented Requirements Tracing (FORT)*

O objetivo da rastreabilidade de requisitos orientada a atributos é identificar relações valiosas de rastreabilidade baseadas em requisitos priorizados, considerando o custo e os esforços (Ahn & Chong, 2006).

Os atributos são as características principais dos produtos (Ahn & Chong, 2006). Estes atributos "podem ser classificados em termos de capacidades, domínio de tecnologias, técnicas de implementação e ambientes operacionais" (Ahn & Chong, 2006).

Em seguida apresenta-se o processo FORT segundo Ahn e Chong (2006). Este é composto por cinco fases:

-Definição de requisitos - esta fase é composta por três atividades: análise da especificação de requisitos, identificação de requisitos individuais e atribuição de identificadores para cada requisito. O resultado desta fase é uma lista de requisitos e os seus identificadores.

-Modelação de atributos - esta fase também é composta por três atividades: identificação de categorias e atributos, organização de diagramas de atributos e atribuição de requisitos relacionados com atributos. O resultado desta fase é uma lista de atributos e diagramas de atributos.

-Priorização de atributos - esta fase consiste em duas atividades: estimativa do valor dos requisitos e ordenação da lista de atributos. O resultado desta fase é uma lista ordenada de atributos que são classificados em três níveis de prioridade (baixa, média e alta).

-Ligação de requisitos - esta fase é constituída por três atividades: atribuição de artefactos a atributos relacionados, fracionamento de elementos de implementação por nível e estabelecimento de ligações de rastreabilidade de requisitos. O resultado desta fase consiste numa lista global de ligações de rastreabilidade.

-Avaliação das ligações de rastreabilidade - esta fase é composta por duas atividades: usabilidade das ligações de rastreabilidade no processo de desenvolvimento e refinamento das ligações de rastreabilidade. Nesta fase, as ligações de rastreabilidade são usadas para análises de mudança de impacto, análises de conflitos de requisitos e verificação de consistência no processo de desenvolvimento.

Ahn e Chong (2006) mencionam que o FORT reduz os esforços de criar ligações de rastreabilidade.

- **Rastreabilidade baseada em eventos -** *Event Based Traceability (EBT)*

Cleland-Huang e Chang (2003) propuseram uma abordagem para atualização e manutenção de relacionamentos de rastreabilidade. Estes definem as relações de rastreabilidade como relações editor-subscritor, na qual os requisitos e os outros promotores de mudança desempenham o papel de editores e os artefactos dependentes atuam como subscritores (Cleland-Huang & Chang, 2003). Quando ocorrem mudanças nos requisitos, é publicada uma mensagem de evento e as notificações relacionadas são enviadas para todos os subscritores dependentes (Cleland-Huang & Chang, 2003).

Segundo Rochimah, Kadir e Abdulla (2007), o método envolve três componentes principais:

1. Gestor de requisitos - é responsável pela gestão dos requisitos e pela publicação de mensagens de eventos de mudanças para o servidor de eventos.

2. Servidor de eventos - é responsável por manipular subscrições de entidades dependentes, por escutar as notificações de eventos do gestor de requisitos e por encaminhar as mensagens de eventos para subscritores relevantes.
3. Gestor de subscritores - é responsável por escutar, em nome dos subscritores que gere, as notificações de eventos transmitidas pelo servidor de eventos.

Cleland-Huang e Chang (2003) referem que o EBT reduz a necessidade de uma sincronização estrita entre os programadores, porporcionando um nível elevado de visibilidade para a gestão de projetos.

• **Recuperação da informação -** *Information Retrieval (IR)*

Esta abordagem foca-se em automatizar a produção de relações de rastreabilidade, através da comparação de semelhanças entre dois tipos de artefactos (Rochimah, Kadir, & Abdullah, 2007). Segundo Rochimah, Kadir e Abdullah (2007) os dois modelos de IR que geralmente são usados na produção da rastreabilidade são modelos probabilisticos e espaço vetoriais. Os metodos de IR segundo Rochimah, Kadir e Abdullah (2007) incluem 3 etapas gerais:

1. Pré-processamento, isto é, remoção de palavras irrelevantes e ou resultantes.
2. Análise e indexação de uma coleção de documentos recebida, seguida pela construção de uma representação de cada documento e depois seu arquivamento.
3. Análise e representação de uma *query* (consulta) de entrada e utilização de um algoritmo de correspondência ou classificação para determinar que representações de documentos são semelhantes à representação da *query*.

Raja e Kamran (2008) mencionam que os métodos de IR reduzem significativamente os esforços necessários para a criação de relações de rastreabilidade entre os artefactos, mas por outro lado, ainda requerem esforços significativos por parte do analista.

• **Abordagem baseada em regras -** *Rule Based Approach (RB)*

O propósito desta abordagem é gerar automaticamente vínculos de rastreabilidade utilizando regras (Raja & Kamran, 2008). São usadas duas regras de rastreabilidade, a regra de rastreabilidade de requisitos para modelos de objetos (RTOM rule) e a regra de rastreabilidade inter-requisito (IREQ rule) (Rochimah, Kadir, & Abdullah, 2007; Raja & Kamran, 2008). Segundo Rochimah, Kadir e Abdullah (2007) as regras são implementadas em três tipos específicos de documentos, ou seja,

em documentos de declarações de requisitos, documentos de casos de uso e análises de modelos de objetos. As regras RTOM são usadas para rastrear os documentos de declarações de requisitos e os documentos de casos de uso para uma análise de modelos de objetos, enquanto que, as regras de IREQ são usadas para rastrear entre documentos de declarações de requisitos e documentos de casos de uso (Rochimah, Kadir, & Abdullah, 2007). Nesta abordagem todos os tipos de documentos e regras são apresentados em formato baseado em XML (Raja & Kamran, 2008). Esta abordagem segundo Rochimah, Kadir e Abdullah (2007) consiste em quatro etapas: codificação gramatical dos artefactos; conversão dos artefactos codificados em representações XML; geração de relações de rastreabilidade entre artefactos; e geração de relações de rastreabilidade entre diferentes partes dos artefactos (Raja & Kamran, 2008). Esta abordagem aborda os documentos de declarações de requisitos, os documentos de casos de uso e as análises de modelos de objetos como os objetos de rastreamento (Rochimah, Kadir, & Abdullah, 2007).

- **Abordagem baseada em modelos de atributos -** *Feature-Model Based Approach* (FB)

  Nesta abordagem utiliza-se a modelação de atributos que descreve os requisitos como uma visão global e modela a variabilidade de uma linha de produto (Rochimah, Kadir, & Abdullah, 2007). Um modelo de atributos consiste num gráfico com nós e arestas, sendo que, os nós são os atributos e as arestas são as relações dos atributos (Raja & Kamran, 2008). Cada atributo representa uma propriedade do produto do ponto de vista do cliente (Raja & Kamran, 2008). Há três categorias de atributos: atributos funcionais; atributos de interface e atributos de parâmetros (Rochimah, Kadir, & Abdullah, 2007).

  Os atributos são estruturados por relações hierárquicas (Rochimah, Kadir, & Abdullah, 2007). De acordo com Rochimah, Kadir e Abdullah (2007) as relações dos atributos podem ser classificadas em três categorias: i) *Relações hierárquicas* que descrevem a sequência de decisões de produtos. ii) *Relações de refinamento* que descrevem relações de generalização e especialização, bem como, agregação. iii) *Relações de exigência ou exclusão que* definem restrições entre os atributos variáveis que podem influenciar a sequência de decisões de produtos.

- **Matrizes de rastreabilidade -** *Traceability Matrices*

  As matrizes de rastreabilidade consistem numa grelha bidimensional que representa as ligações que existem entre os artefactos (Winkler & Pilgrim, 2010). As linhas e as colunas da grelha representam os artefactos e as marcas nas intersecções dos artefactos significam que existe uma ligação entre eles.

  Figura 5 mostra uma matriz de rastreabilidade. As matrizes no campo da rastreabilidade de requisitos são a forma de visualização mais tradicional (Winkler & Pilgrim, 2010). As matrizes de rastreabilidade são fáceis de compreender e de utilizar, no entanto, em projetos de alguma dimensão e complexidade tornam-se muito grandes e ilegíveis (Winkler & Pilgrim, 2010).

	Req 1	Req 2	Req 3	Class 1
Req 1		■	■	■
Req 2			■	
Req 3		■		
Class 1				

Figura 5 – Matriz de rastreabilidade

Fonte: (Winkler & Pilgrim, 2010)

As diversas técnicas/abordagens, analisadas anteriormente, permitem adquirir conhecimentos sobre aspetos importantes que podem ser usados para tratar a rastreabilidade. Algumas técnicas têm mais relevância que outras no trabalho pretendido, no entanto, todas elas podem contribuir com algum tipo de conhecimento.

O facto de se efetuar um estudo pormenorizado sobre a rastreabilidade e as suas técnicas/abordagens, vai permitir adquirir conhecimentos que vão ser aplicados posteriormente, no desenvolvimento dos artefactos a que este trabalho se propõe.

2.4 Interdependência de Requisitos

Dahlstedt e Persson (2003) mencionam a rastreabilidade como "uma base para abordar as interdependências de requisitos".

Durante o processo de engenharia de requisitos, a maioria dos requisitos individuais desenvolvidos não podem ser tratados isoladamente (Dahlstedt & Persson, 2005), uma vez que, se relacionam e se afetam mutuamente de diversas formas (Carlshamre, Sandahl, Lindvall, Regnell, & Natt och Dag, 2001; Regnell, et al., 2001). Dahlstedt e Persson referem no seu trabalho (2003) que os requisitos são interdependentes. Este facto faz com que seja necessário um cuidado especial ao efetuar diversas mudanças no sistema, uma vez que, outros requisitos ou artefactos podem ser afetados.

Zhang, Mei e Zhao (2005) referem que as dependências são elementos essenciais entre os requisitos de um sistema de *software* real, por causa da coesão do sistema. Estes mencionam que a coesão é uma qualidade básica de um sistema, uma qualidade que é necessária para um sistema ser um sistema e para alcançar determinados objetivos pretendidos pelos clientes. Estes mesmos autores referem ainda que, um conjunto de requisitos de um sistema complexo pode também ter dependências negativas entre eles, isto é, conflitos ou inconsistências. Estas dependências negativas são causadas pelo facto dos requisitos frequentemente terem origem de *stakeholders* com diferentes ou contraditórias perspetivas (Zhang, Mei, & Zhao, 2005).

Durante o processo de desenvolvimento de *software*, várias atividades, tais como, coesão, controlo e gestão de mudanças podem ser influenciadas pelo facto dos requisitos serem dependentes uns dos outros (Goknil, Kurtev, Berg, & Veldhuis, 2009). Isto implica uma grande necessidade de ter em consideração as interdependências entre os requisitos e os diversos artefactos do sistema, a fim de, evitar decisões inconscientes durante o processo de desenvolvimento. Negligenciar essas mesmas interdependências pode levar a consequências bastante negativas para o processo de desenvolvimento.

Um requisito pode afetar outros quando: "forçar como outros requisitos podem ser concebidos ou implementados; afetar o custo de implementação de outros requisitos; ou aumentar ou diminuir a satisfação dos clientes de outros requisitos" (Dahlstedt & Persson, 2005).

Carlshamre, Sandahl, Lindvall, Regnell e Natt och Dag (2001) mostraram através do seu estudo de interdependências que apenas uma pequena percentagem de requisitos são verdadeiramente independentes, isto é, não se relacionam nem influenciam quaisquer outros requisitos. Salientando assim, a necessidade para explorar as interdependências de requisitos e também a necessidade de apoiar a identificação e gestão de interdependências (Carlshamre, Sandahl, Lindvall, Regnell, & Natt och Dag, 2001).

O facto dos requisitos se relacionarem e se afetarem mutuamente faz com que seja essencial identificar e controlar as interdependências que ocorrem ao longo do desenvolvimento do sistema, de forma a evitar erros potencialmente caros. Como já foi referido anteriormente, através da rastreabilidade é possível gerir essas mesmas interdependências, daí a rastreabilidade ser considerada tão fundamental no processo de desenvolvimento.

Ao longo do desenvolvimento das soluções e também durante a fase exploração, por questões de manutenção/evolução, surge a necessidade de introduzir diversas alterações às decisões de projeto previamente estabelecidas. Essas alterações devem ser facilmente identificadas, de forma a garantir que os artefactos envolvidos nas alterações sejam conhecidos e posteriormente tratados. Quando avaliado o impacto de uma mudança nos requisitos, negligenciar as interdependências pode resultar em custos mais elevados (Dahlstedt & Persson, 2005).

O propósito de sistematicamente lidar com interdependências de requisitos permite melhorar as decisões feitas durante o desenvolvimento de *software* e também detetar antecipadamente potenciais problemas que possam surgir, pelo facto dos requisitos serem interdependentes (Dahlstedt & Persson, 2005). Gerir interdependências de requisitos consiste em "identificar, armazenar e manter informações sobre como os requisitos se relacionam e se afetam uns aos outros" (Dahlstedt & Persson, 2005).

Manter a rastreabilidade das interdependências de requisitos é essencial, a fim de, apoiar várias situações e atividades no processo de desenvolvimento do sistema (Dahlstedt & Persson, 2003).

Apesar de ser uma área bastante inexplorada (Dahlstedt & Persson, 2003), foram encontrados na literatura alguns trabalhos que abordam o assunto das interdependências de requisitos. Em seguida apresentam-se alguns dos mesmos.

Carlshamre, Sandahl, Lindvall, Regnell e Natt och Dag (2001) basearam-se em trabalhos anteriores e criaram um conjunto preliminar de cinco tipos de interdependências. Em seguida descrevem-se esses mesmos tipos de interdependências e as suas representações de acordo com os autores.

AND – um requisito requer outro requisito para poder ser executado e esse outro requisito requer o primeiro para também poder ser executado. R_1 AND R_2

REQUIRES – um requisito requer outro requisito para poder ser executado, no entanto, esse outro requisito não necessita do primeiro para ser executado. R_1 REQUIRES R_2

TEMPORAL – quando dois requisitos dependem temporalmente um do outro, isto é, um requisito só pode ser executado quando outro requisito já foi executado anteriormente. R_1 TEMPORAL R_2

CVALUE – quando um requisito aumenta ou diminui o valor de outros requisitos para o cliente. R_1 CVALUE R_2

ICOST – quando um requisito aumenta ou diminui o custo de implementar outros requisitos. R_1 ICOST R_2

OR – apenas um dos requisitos precisa de ser implementado. R_1 OR R_2

Dahlstedt e Persson (2003) no seu estudo tiveram como objetivo fornecer uma visão global dos tipos existentes de interdependências de requisitos apresentados na literatura, assim como, sintetiza-los num modelo de tipos de interdependências fundamentais.

Dahlstedt e Persson (2003) focaram-se no modelo de Pohl (como citado em Dahlstedt & Persson, 2003). Este desenvolveu um *framework* de rastreabilidade que incluía um modelo de dependências, Figura 6, que definia 18 tipos diferentes de possíveis relações de dependências. O modelo de Pohl descreve os tipos de dependências que podem existir entre qualquer tipo de objeto rastreado usado no processo de engenharia de requisitos (Dahlstedt & Persson, 2003).

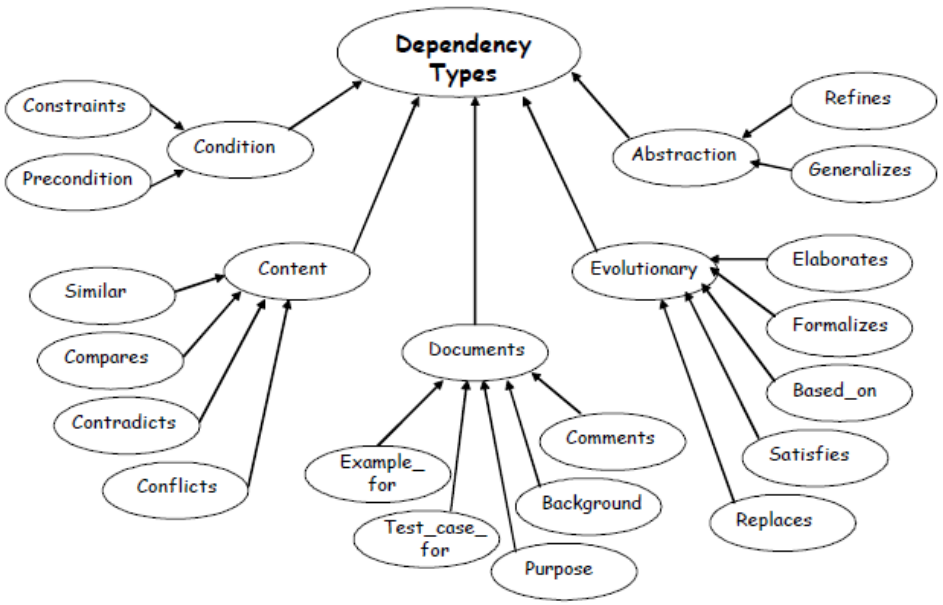


Figura 6 – Modelo de dependências de Pohl

Fonte: (Dahlstedt & Persson, 2003)

Dahlstedt e Persson (2003) referem o modelo de dependências de Pohl como um valioso ponto de partida na sua investigação. No entanto mencionam ser necessário adaptar e especializar o modelo de Pohl para as interdependências de requisitos, uma vez que, o modelo possui alguns tipos de dependências que claramente não existem entre os requisitos (Dahlstedt & Persson, 2003).

As categorias e os tipos de dependência apresentados no modelo de Pohl, segundo Dahlstedt e Persson (2003) são por vezes difíceis de distinguir, claramente, uns dos outros. E segundo as mesmas autoras, existem tipos de interdependências de requisitos adicionais encontrados na literatura subsequente que devem ser incluídos no modelo (Dahlstedt & Persson, 2003).

Assim sendo, com base na literatura e também num estudo de entrevista, Dahlstedt e Persson desenvolveram um modelo de tipos fundamentais de interdependências de requisitos, Figura 7. Dahlstedt e Persson (2005) referem no entanto que, os tipos de interdependências presentes nesse modelo podem ser mais elaborados devido às necessidades especificas dos projetos, necessitando assim de mais investigação, e mais categorias e tipos de interdependências podem ser encontrados.

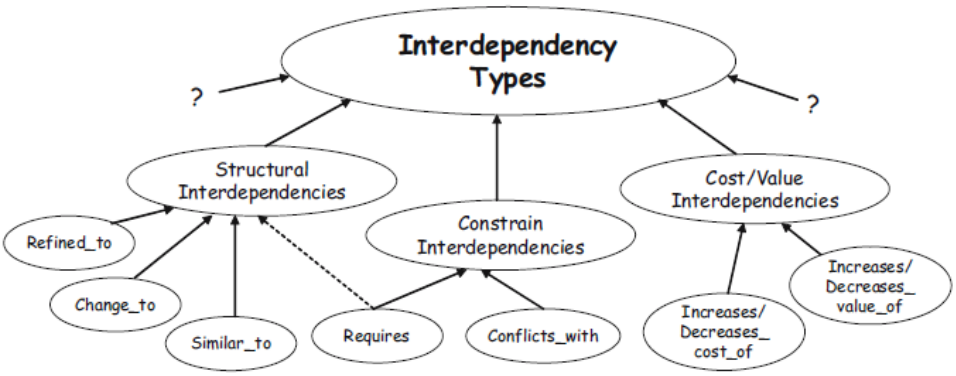


Figura 7 – Classificação dos tipos fundamentais de interdependência

Fonte: (Dahlstedt & Persson, 2005)

Dahlstedt e Persson (2005) no seu modelo de tipos fundamentais de interdependências de requisitos identificaram três categorias de interdependências: interdependências estruturais, interdependências constritivas e interdependências de custo/valor. Em seguida apresenta-se uma breve descrição das mesmas segundo as autoras.

• *Interdependências estruturais (structural interdependencies)*

As interdependências estruturais estão preocupadas com o facto de que dado um conjunto específico de requisitos, eles podem ser organizados numa estrutura, onde as relações são de uma estrutura hierárquica, bem como, de uma estrutura de natureza transversal. Os requisitos de negócio de alto nível são gradualmente decompostos em requisitos de *software* mais detalhados, formando uma hierarquia. A esta categoria pertencem os seguintes tipos de interdependência:

Refined_to – um requisito de nível mais elevado é refinado por um número de requisitos mais específicos.

Change_to – um requisito muda para outro requisito se uma nova versão desse requisito é desenvolvida que substitui a antiga.

Similar_to – um requisito é semelhante ou sobreposto com outro, em termos de como ele é expressado ou em termos de uma ideia subjacente semelhante de que o sistema deverá ser capaz de executar.

• *Interdependências constritivas (constrain interdependencies)*

As interdependências constritivas descrevem como os requisitos podem constranger uns os outros, isto é, há requisitos que apenas podem ser executados quando outros requisitos forem executados antes e requisitos em que a sua execução impede a execução de outros. Dahlstedt e Persson (2005) apesar de terem incluído esta categoria no seu modelo referem que a mesma necessita de mais investigação. Nesta categoria estão identificados dois tipos de interdependências:

Requires – a realização de um requisito depende da realização de outro requisito.

Conflicts_with – um requisito está em conflito com outro requisito, se aumentar a satisfação de um requisito diminui a satisfação de outro requisito, ou se eles não podem existir ao mesmo tempo.

• *Interdependências de custo/valor (cost/value interdependencies)*

As interdependências de custo/valor referem-se aos custos envolvidos na implementação de um requisito em relação ao valor que a realização desse requisito fornecerá ao cliente/utilizador afetado. Esta categoria possui dois tipos de interdependências:

Increases/decreases_cost_of – se um requisito é escolhido para ser implementado então o custo de implementar outro requisito aumenta ou diminui.

Increases/decreases_value_of – se um requisito é escolhido para ser implementado então o valor para o cliente de outro requisito aumenta ou diminui.

Ao longo dos últimos anos, têm sido desenvolvidas algumas técnicas e abordagens para identificar e gerir as interdependências de requisitos. Algumas dessas abordagens/técnicas são:

• Abordagem orientada a atributos

Esta abordagem foi apresentada por Zhang, Mei e Zhao (2005) e consiste numa abordagem orientada a atributos para modelação de dependências de requisitos. A característica mais importante desta abordagem, segundo os autores, é que ela não utiliza requisitos individuais quando explora as dependências de requisitos mas atributos como entidades básicas.

Os atributos mencionados por esta abordagem são definidos pelos autores como sendo, um conjunto de fortes requisitos relacionados com o ponto de vista dos utilizadores/clientes. Estes atributos servem como um módulo que encapsula um conjunto de requisitos relacionados e podem ser operacionalizados num conjunto de responsabilidades (Zhang, Mei, & Zhao, 2005).

A Figura 8 ilustra o processo de operacionalização e atribuição de responsabilidades dos atributos.

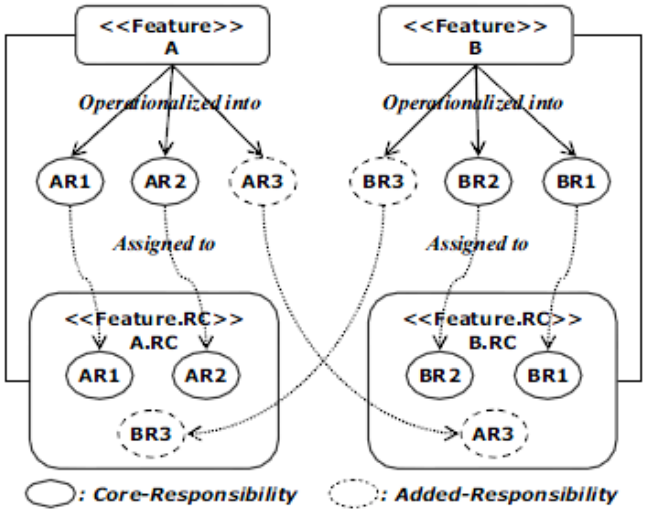


Figura 8 – Atribuição de responsabilidades

Fonte: (Zhang, Mei, & Zhao, 2005)

Segundo Zhang, Mei e Zhao (2005), neste exemplo, o atributo A é operacionalizado por três responsabilidades: AR1, AR2 e AR3. AR1 e AR2 são atribuídas ao A.RC, representado assim, as responsabilidades essenciais de A. AR3 é atribuído ao B.RC do atributo B, representando assim, uma das responsabilidades adicionais de B. O processo do atributo B é similar.

Zhang, Mei e Zhao (2005) alegam no seu estudo que o número de possíveis dependências entre atributos é significativamente menor que o dos requisitos. Estes introduziram quatro importantes tipos de dependências de atributos nomeadamente refinamento, restrição, influência e interação. Em cada um destes tipos de dependência existem subtipos.

Zhang, Mei e Zhao (2005) investigaram as possíveis ligações entre os quatro tipos de dependências identificados, mostrando assim, que uma importante função destas ligações é que elas facilitam a identificação das dependências dos atributos.

- **Meta-modelo de requisitos**

Goknil, Kurtev, Berg e Veldhuis (2009) focaram-se nos requisitos e nas relações dos requisitos, a partir de uma perspetiva de rastreabilidade e propuseram um meta-modelo de requisitos. Esse meta-modelo pode ser visualizado na Figura 9.

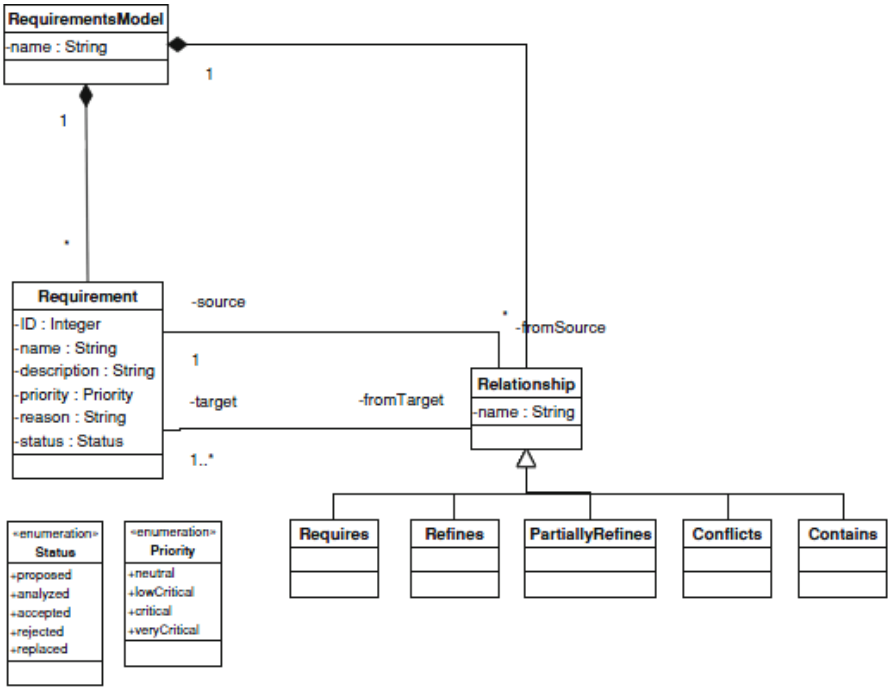


Figura 9 – Meta-modelo de requisitos

Fonte: (Goknil, Kurtev, Berg, & Veldhuis, 2009)

Neste meta-modelo de requisitos, os requisitos são capturados num modelo de requisitos, sendo que, este modelo possui os requisitos e as suas relações (Goknil, Kurtev, Berg, & Veldhuis, 2009).

Segundo Goknil, Kurtev, Berg e Veldhuis (2009) os requisitos têm um identificador único (ID), um nome, uma descrição textual, uma prioridade, uma razão e um estado. Os autores identificaram cinco tipos de relações. Estas relações são definidas por Goknil, Kurtev, Berg e Veldhuis (2009) da seguinte forma:

Relação *requires* – um requisito R_1 requer um requisito R_2. R_1 apenas é cumprido quando R_2 for comprido.

Relação *refines* – um requisito R_1 refina um outro requisito R_2. R_1 é derivado de R_2 por adicionar mais detalhes às suas propriedades.

Relação *partially refines* – um requisito R_1 refina parcialmente um requisito R_2. R_1 é derivado de R_2 por adicionar mais detalhes às propriedades de R_2 e excluir propriedades não refinadas de R_2. Nesta relação R_2 pode ser decomposto em outros requisitos e R_1 refina um subconjunto desses requisitos decompostos. Esta relação é descrita como uma combinação especial de decomposição e refinamento.

Relação *conflicts* - um requisito R1 entra em conflito com um requisito R2. O cumprimento do R1 exclui o cumprimento do R2 e vice-versa.

Relação *contains* – um requisito R1 contem requisitos R2, ..., Rn. R2, ..., Rn são partes de R1.

O objetivo deste meta-modelo é melhorar as relações dos requisitos por atribuição de tipos de relações e definição da sua semântica (Goknil, Kurtev, Berg, & Veldhuis, 2009).

• Abordagem de Chitchyan e Rashid

Chitchyan e Rashid (2006), através do seu trabalho, mencionaram como as dependências de requisitos podem ser deduzidas, a partir da semântica de requisitos e como a sua rastreabilidade pode ser mantida. Nesta abordagem a composição é utilizada para assegurar consistência nos requisitos e para reduzir a redundância de repetir requisitos (Chitchyan & Rashid, 2006).

Os autores derivaram um conjunto de operadores de ação, Figura 10, que refletem os tipos gerais de dependências de requisitos.

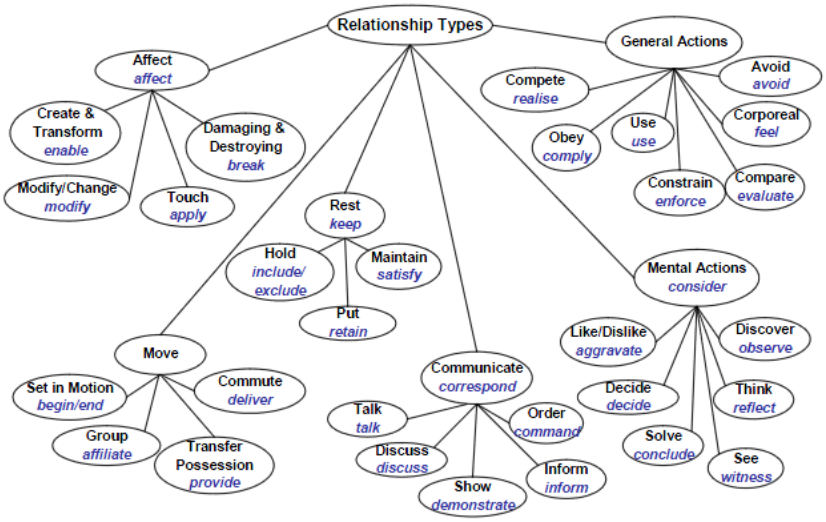


Figura 10 – Composição dos operadores de ação

Fonte: (Chitchyan & Rashid, 2006)

Chitchyan e Rashid (2006) através da Figura 11 ilustraram como a rastreabilidade pode ser mantida no processo de composição.

Nesta abordagem as dependências entre os requisitos de duas preocupações são mapeadas por um operador, sendo que, o operador reflete o tipo de dependência existente entre os requisitos.

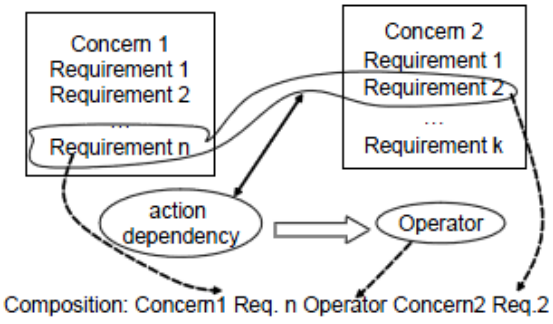


Figura 11 – Rastreabilidade para os operadores de ação

Fonte: (Chitchyan & Rashid, 2006)

Esta abordagem pode ser utilizada como uma sistemática forma de derivar operadores de ação, a partir das semânticas das relações de requisitos e manter a rastreabilidade (Chitchyan & Rashid, 2006).

Algumas das técnicas/abordagens apresentadas anteriormente, na rastreabilidade de requisitos, tais como, rastreabilidade de requisitos baseada em valor, rastreabilidade de requisitos orientada

a atributos, abordagem baseada em regras, abordagem baseada em modelos de atributos e as matrizes de rastreabilidade para além de permitirem tratar a rastreabilidade, também permitem identificar e gerir as interdependências de requisitos.

As diversas técnicas e abordagens, analisadas anteriormente, permitem adquirir conhecimentos sobre aspetos importantes que podem ser usados para identificar e gerir interdependências de requisitos. Sendo que, alguns desses conhecimentos vão ser aplicados posteriormente, no desenvolvimento dos artefactos pretendidos para este trabalho.

Chitchyan e Rashid (2006) referem que o problema das interdependências de requisitos resultar em conflitos e compromissos é à muito conhecido na comunidade da engenharia de requisitos. No entanto, a área das interdependências de requisitos é bastante inexplorada, a julgar pela quantidade relativamente reduzida de literatura a discutir este assunto e mais investigação nesta área é necessária (Dahlstedt & Persson, 2003). Contudo, como se pode verificar pela revisão da literatura, este assunto é extremamente importante, uma vez que, pode por em causa todo o processo de desenvolvimento do sistema.

2.5 Conclusões

Através da revisão da literatura efetuada pode-se concluir que a rastreabilidade de requisitos, assim como, o conhecimento sobre as várias interdependências existentes é fundamental, ao longo do processo de desenvolvimento.

A rastreabilidade deve ser incluída e tratada ao longo dos projetos de desenvolvimento representado assim, uma mais valia para o sucesso dos mesmos. O facto de se conhecer toda a história dos artefactos, assim como, as suas interdependências vai possibilitar identificar e gerir mais facilmente as interdependências existentes, desde as fases iniciais de desenvolvimento. Permitindo assim, minimizar possíveis problemas que possam advir pelo facto dos requisitos se relacionarem uns com os outros e conhecer pormenorizadamente como os vários artefactos se relacionam.

3. Interdependências e rastreabilidade no RUP

3.1 Introdução

O *Rational Unified Process* (RUP) é um processo de engenharia de *software* que tem como finalidade garantir a produção de *software* de qualidade, que atenda às necessidades e expectativas dos seus utilizadores num custo e prazo previsíveis (Kruchten, 2002). O RUP fornece as melhores práticas e orientações para o desenvolvimento de *software* (Kruchten, 2002) e é constituído por vários elementos, tais como, atividades, tarefas, *roles* (papéis) e *work products.*

Ao longo de todo o processo de desenvolvimento, os elementos que constituem o RUP vão estar interligados em diversos momentos, o que faz com que uma simples mudança num elemento provoque vários ajustes posteriores em outros. Este facto faz com que o conhecimento sobre as interdependências existentes entre os diversos elementos seja bastante útil, uma vez que, permite identificar mais facilmente quais os elementos que serão afetados aquando de uma mudança.

Sendo o RUP um processo que fornece as melhores práticas e orientações para o desenvolvimento de *software,* será bastante útil conhecer as interdependências que podem existir entre os diversos elementos do RUP. O facto de se conhecer tais interdependências pode evitar decisões inconscientes durante o processo de desenvolvimento, assim como, detetar antecipadamente potenciais problemas que possam surgir devido às interdependências existentes. O conhecimento sobre as interdependências existentes pode ainda, permitir conhecer pormenorizadamente como o processo RUP é constituído.

Através da rastreabilidade dos diversos elementos do RUP é possível identificar e gerir mais facilmente as interdependências que podem ocorrer entre os elementos. Tal como Dahlstedt e Persson (2003) aludem é essencial manter a rastreabilidade das interdependências, uma vez que, permite conhecer pormenorizadamente como os elementos se relacionam e apoiar várias situações e atividades no processo de desenvolvimento de *software.*

Neste capítulo pretende-se, numa primeira fase, efetuar uma análise ao RUP com o objetivo de estudar os diversos elementos que o constituem. E posteriormente, com base nessa análise, pretende-se sistematizar a rastreabilidade e as interdependências que se podem verificar entre

os elementos que constituem o RUP, recorrendo a técnicas de interdependências e rastreabilidade.

3.2 Síntese do RUP

O RUP é um processo de desenvolvimento de *software* (Manzoni & Price, 2003), inicialmente desenvolvido e comercializado pela *Rational Software Corporation,* sendo que, em 2003 passou a fazer parte da IBM (Kruchten, 2004).

O RUP descreve o conjunto de atividades necessárias para transformar requisitos do utilizador num sistema de *software* (Manzoni & Price, 2003). Este processo fornece uma abordagem disciplinada para a atribuição de tarefas e responsabilidades numa organização de desenvolvimento (Kruchten, 2002). O objetivo do RUP segundo Kruchten (2002) é garantir a produção de *software* de alta qualidade, que atenda às necessidades dos seus utilizadores, segundo um custo e um prazo previsíveis.

Este processo captura muitas das melhores práticas de desenvolvimento de *software* moderno, de uma forma que seja adquada para uma vasta gama de projetos e organizações (Kruchten, 2002). Segundo a Rational Software (2001) o RUP permite melhorar a produtividade da equipa, uma vez que, fornece a cada membro da equipa um fácil acesso a uma base de conhecimento com orientações, templates e mentores de ferramentas para todas as actividades de desenvolvimento criticas. É de referir que, os mentores de ferramentas explicam como usar uma ferramenta específica para criar parte de um *work product,* quer no contexto de uma tarefa ou actividade ou independentemente (Rational Unified Process).

O RUP é suportado por ferramentas que automatizam grande parte do processo. Estas são usadas para criar e manter os diversos artefactos do processo de engenharia de *software* (Rational Software, 2001). A *Unified Modeling Language* (UML) é usada como a principal notação para os vários modelos que são construidos durante o desenvolvimento (Kruchten, 2002).

Este processo promove o desenvolvimento iterativo e organiza o desenvolvimento de *software* em quatro fases, cada uma consistindo de uma ou mais iterações executáveis do *software* nessa fase de desenvolvimento (IBM).

A Figura 12 mostra as duas dimensões principais do RUP que são as fases e as disciplinas.

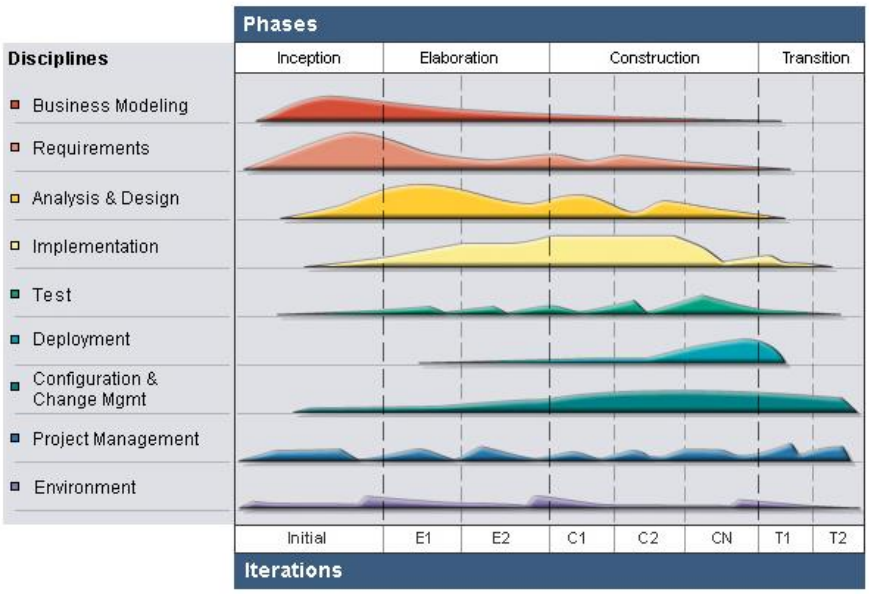


Figura 12 – Dimensões principais do RUP

Fonte: (Rational Unified Process)

As disciplinas representam as áreas de trabalho num projeto de desenvolvimento de *software.* Segundo Hirsch (2002) o RUP possui nove disciplinas, são elas:

-**Modelação de negócios** (*Business Modeling*) - descreve os processos de negócio e a estrutura interna de um negócio a fim de: 1) compreender melhor o negócio e 2) ser capaz de chegar com os requisitos adequados para os sistemas de software a serem construidos para o negócio em mão.

-**Requisitos** (*Requirements*) – esta disciplina refere-se à gestão dos requisitos, ou seja, ao levantamento, organização e documentação dos requisitos.

- **Análise e conceção** (*Analysis and Design*) – cria a arquitetura e a conceção do sistema de *software*.

-Implementação *(Implementation)* – tem como função escrever e depurar (*debugging*) o código fonte, testes de unidade e criar gestão.

-Teste (*Test*) – fornece orientações sobre como avaliar e analisar a qualidade do produto.

-Entrega *(Deployment)* – tem as funções de empacotar/produtizar o *software,* criar *scripts* de instalação, escrever a documentação para os utilizadores e outras tarefas necessárias para tornar o *software* disponível para os seus utilizadores.

-Configuração e gestão de mudança *(Configuration and Change Management)* – abrange todas as tarefas relacionadas com: 1) gestão de lançamentos e versões e 2) gestão de mudanças de requisitos.

-Gestão de projetos (*Project Management*) – refere-se ao planeamento e monitorização do projeto.

-Ambiente (Environment) – tem a função de adaptar o processo às necessidades de um projeto (ou uma organização), selecionar, introduzir e apoiar ferramentas de desenvolvimento.

Para cada disciplina, o RUP define um conjunto de artefactos, atividades e *roles* (Hirsch, 2002).

Os artefactos são fragmentos de informação que são produzidos, modificados ou usados por um processo (Manzoni & Price, 2003). Estes são um produto de trabalho (*work product*), tal como, um documento, código fonte, ou um modelo de objetos expresso em UML (Hirsch, 2002). Os artefactos são utilizados como entrada (*input*) pelos trabalhadores para executar uma atividade e são o resultado ou saída (*output*) dessa atividade (Rational Software, 2001).

Uma atividade é uma descrição detalhada de uma pequena unidade de trabalho que cria, modifica, acrescenta ou revê um artefacto (Hirsch, 2002). As atividades têm artefactos de entrada (*input*) e de saída (*output*) (Manzoni & Price, 2003).

Um *role* é uma responsabilidade que uma ou mais pessoas assumem num projeto, tal como, gestor de projetos, arquiteto de *software* ou *designer* de teste.

O RUP divide um projeto em quatro fases são elas (Hirsch, 2002):

-Conceção (*Inception*) – define os objetivos do projeto, incluindo o caso de negócio.

-Elaboração (*Elaboration*) - cria e valida a arquitetura do sistema de *software*, capturando os requisitos mais importantes e críticos, e planeia e estima o resto do projeto.

-Construção (*Construction*) - implementa o sistema baseado numa arquitetura executável, criada na fase de elaboração.

-Transição (*Transition*) – testa a versão beta do sistema e prepara lançamentos candidatos.

Cada fase é dividida em uma ou mais iterações. Cada iteração baseia-se nos resultados das iterações anteriores, sendo que, a duração e os objetivos de uma iteração são planeados antes da iteração começar (Hirsch, 2002). Quando uma iteração é concluída, é feita uma avaliação completa da iteração, a fim de, possibilitar ações corretivas no caso das mesmas serem necessárias (Hirsch, 2002).

3.3 Análise de atividades e tarefas

O RUP, tal como já foi mencionado anteriormente, é constituído por várias disciplinas e fases. Porém, este estudo incide-se, tal como mostra a Figura 13, na transição da disciplina *Business Modeling* fase *Inception* para a disciplina *Requirements* fase *Inception* e na transição da disciplina *Requirements* fase *Inception* para a disciplina *Requirements* fase *Elaboration*, uma vez que, é nessa área que o micro-processo *V-Model* e o método 4SRS se posicionam.

Neste estudo são analisadas as atividades, as tarefas e os *work products* respeitantes á área delimitada anteriormente, com o intuito de estudar as interdependências e a rastreabilidade entre esses elementos no RUP.

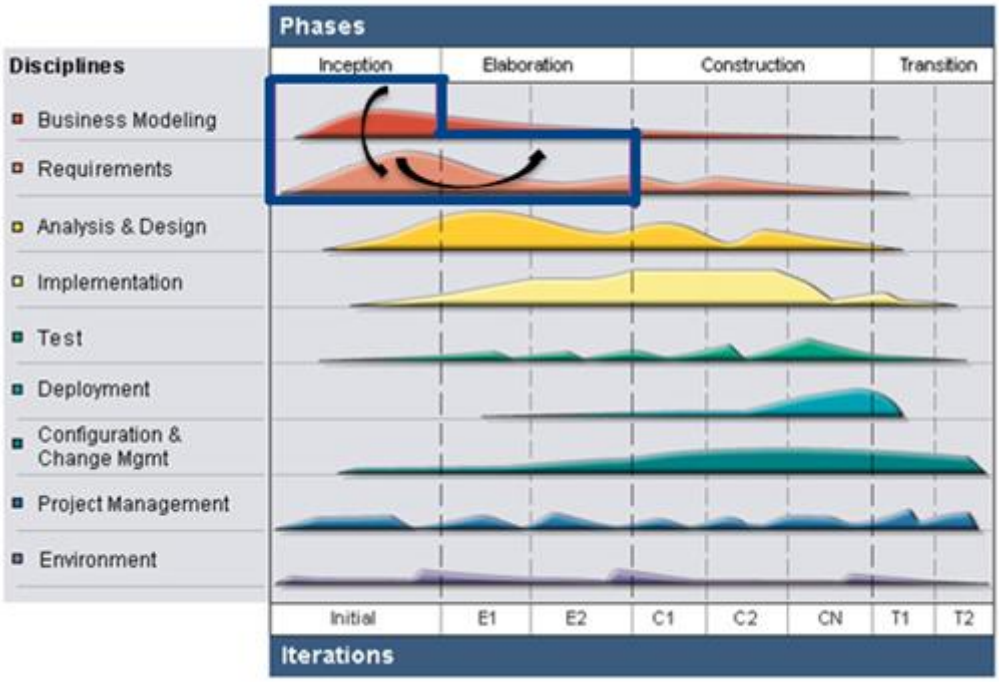


Figura 13 – Posicionamento do estudo no RUP

Numa primeira fase foram elaboradas a Tabela 1 e a Tabela 2 com o propósito de analisar todas as tarefas das disciplinas *Business Modeling* e *Requirements*, assim como, as respetivas atividades, fases e *roles*. A Tabela 1 e a Tabela 2 foram elaboradas tendo como base as matrizes de rastreabilidade, apresentadas anteriormente no capítulo da revisão da literatura.

Para realizar estas duas tabelas (Tabela 1 e Tabela 2), inicialmente foram analisadas, através do RUP, todas as atividades referentes às duas disciplinas em estudo Figura 14 e Figura 15, bem como, as tarefas associadas às mesmas Figura 16 e Figura 17.

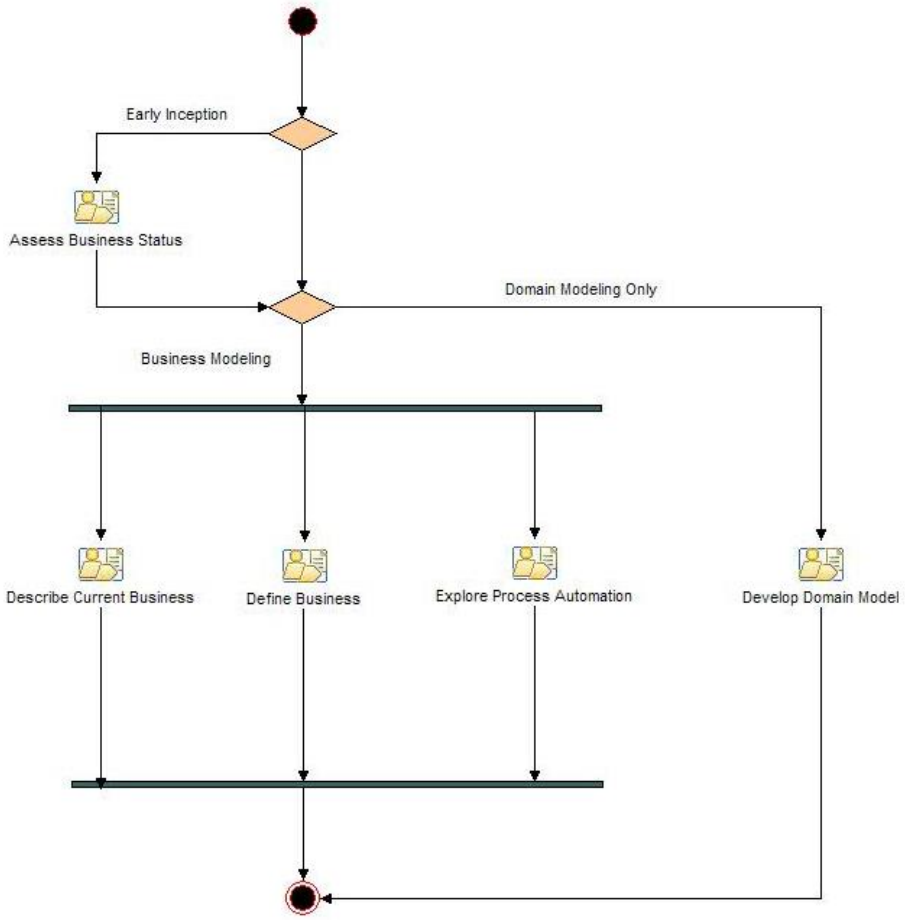


Figura 14 – Atividades da disciplina *Business Modeling*

Fonte: (Rational Unified Process)

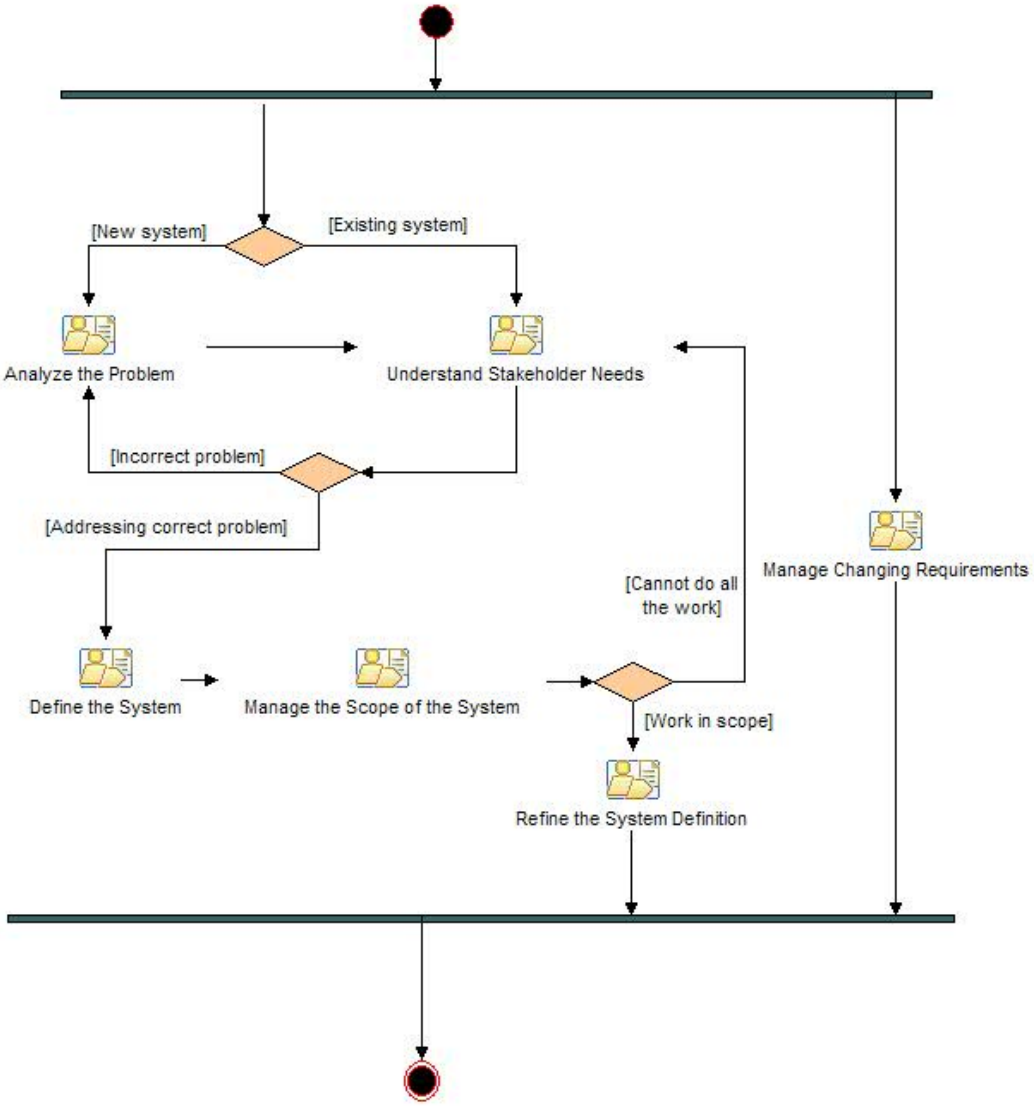
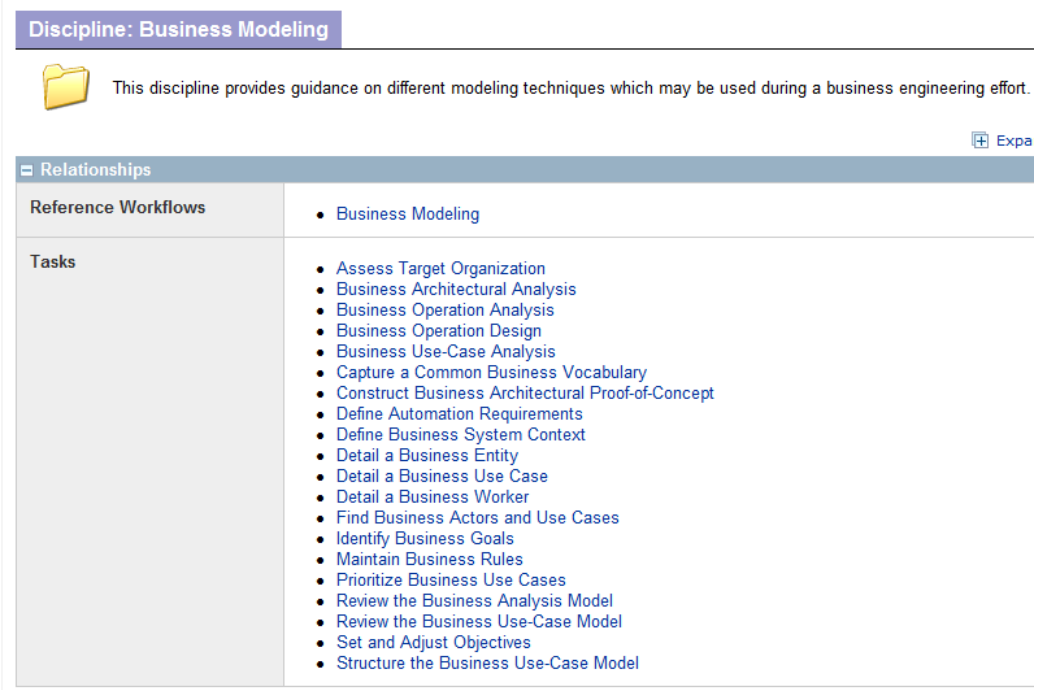


Figura 15 – Atividades da disciplina *Requirements*

Fonte: (Rational Unified Process)



Discipline: Business Modeling

This discipline provides guidance on different modeling techniques which may be used during a business engineering effort.

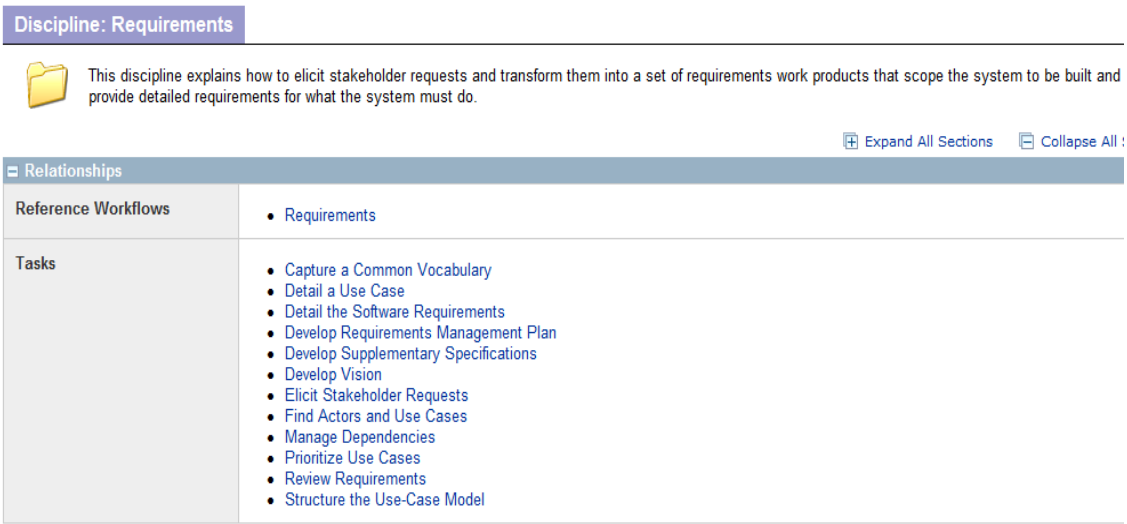
Expa

Relationships

Reference Workflows	• Business Modeling
Tasks	• Assess Target Organization • Business Architectural Analysis • Business Operation Analysis • Business Operation Design • Business Use-Case Analysis • Capture a Common Business Vocabulary • Construct Business Architectural Proof-of-Concept • Define Automation Requirements • Define Business System Context • Detail a Business Entity • Detail a Business Use Case • Detail a Business Worker • Find Business Actors and Use Cases • Identify Business Goals • Maintain Business Rules • Prioritize Business Use Cases • Review the Business Analysis Model • Review the Business Use-Case Model • Set and Adjust Objectives • Structure the Business Use-Case Model

Figura 16 – Tarefas da disciplina *Business Modeling*

Fonte: (Rational Unified Process)



Discipline: Requirements

This discipline explains how to elicit stakeholder requests and transform them into a set of requirements work products that scope the system to be built and provide detailed requirements for what the system must do.

Expand All Sections Collapse All

Relationships

Reference Workflows	• Requirements
Tasks	• Capture a Common Vocabulary • Detail a Use Case • Detail the Software Requirements • Develop Requirements Management Plan • Develop Supplementary Specifications • Develop Vision • Elicit Stakeholder Requests • Find Actors and Use Cases • Manage Dependencies • Prioritize Use Cases • Review Requirements • Structure the Use-Case Model

Figura 17 – Tarefas da disciplina *Requirements*

Fonte: (Rational Unified Process)

Com base nestas primeiras análises foi possível verificar todas as atividades executadas nas disciplinas em estudo, assim como, as tarefas pertencentes a essas mesmas atividades.

Em seguida, analisou-se em que fases as diversas tarefas eram executadas. O RUP tem dois processos que podem ser seguidos, o *Classic RUP Lifecycle* e o *Business Modeling Lifecycle.* O primeiro representa um exemplo do ciclo de vida do RUP clássico e o segundo representa um exemplo de um processo focado apenas na modelação de negócios (Rational Unified Process). O facto do RUP ter estes dois processos faz com que seja necessário analisar todas as fases em cada um dos processos, uma vez que, são diferentes.

A Figura 18 apresenta a análise efetuada à fase *Inception* no processo *Classic RUP Lifecycle*. Com o cruzamento da informação obtida através desta análise e com a lista das atividades pertencentes às disciplinas em estudo, Figura 14 e Figura 15, é possível identificar quais as atividades que são executadas na fase em análise. Assim sendo, verifica-se que no processo *Classic RUP Lifecycle* na fase *Inception* apenas são executadas duas atividades (*Assess Business Status* e *Develop Domain Model)* da disciplina *Business Modeling* e quatro atividades (*Analyze the Problem*, *Understand Stakeholder Needs*, *Define the System* e *Manage the Scope of the System*) da disciplina *Requirements.* É de referir que a atividade *Analyze the Problem* e a *Understand Stakeholder Needs* aparecem agrupadas no RUP numa atividade denominada *Develop Initial Vision.*

Para cada fase dos dois processos foi realizada uma análise semelhante à apresentada anteriormente, para assim, se conseguir reunir a informação necessária para se verificar em que fases as atividades são executadas.

Através destas primeiras análises verifica-se que a mesma tarefa pode ser executada em várias atividades e fases diferentes.

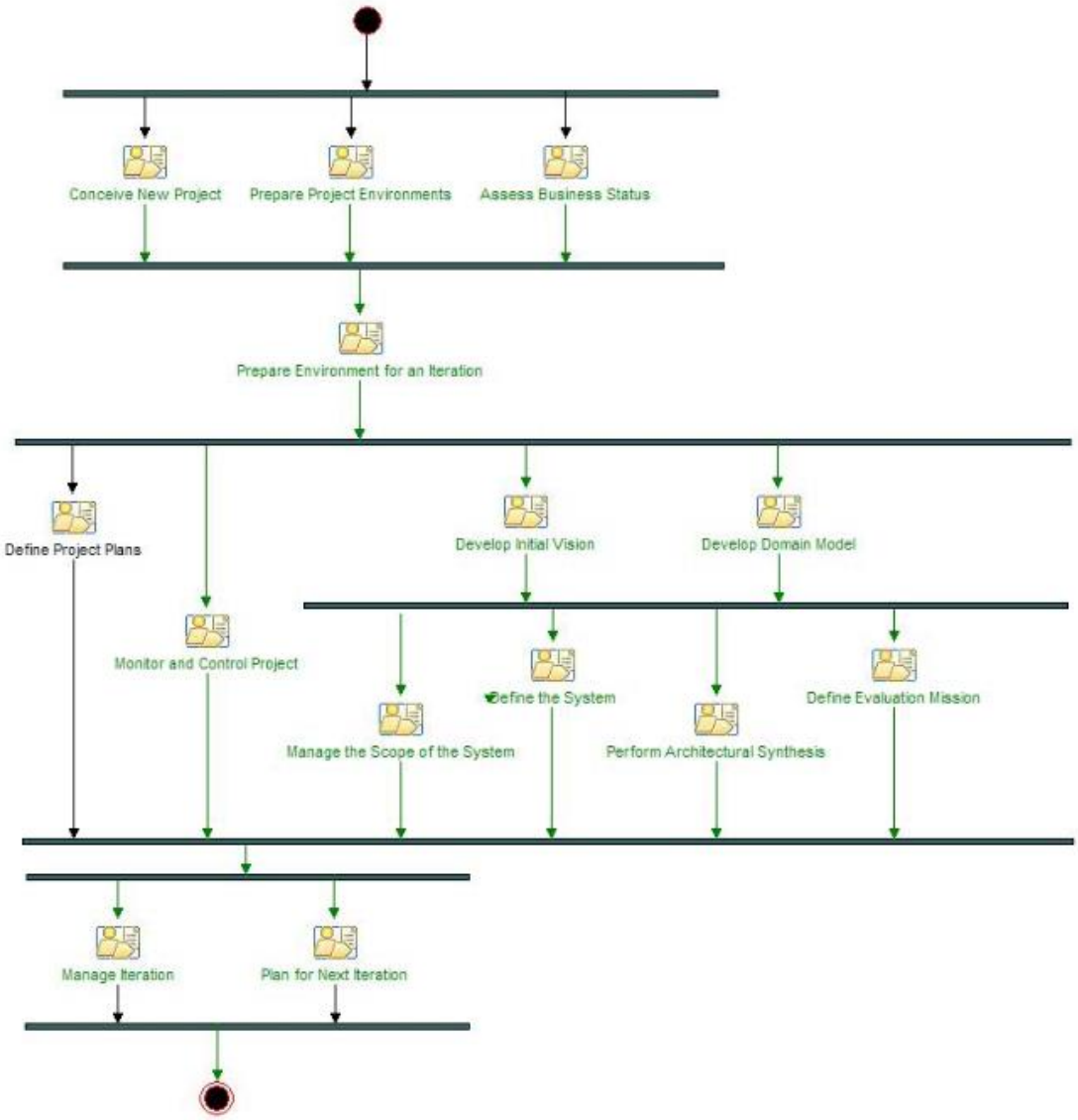


Figura 18 – Atividades da fase *Inception*

Fonte: (Rational Unified Process)

Ao aceder ao conteúdo das atividades da Figura 14 e da Figura 15 tem-se acesso às diferentes tarefas que estão associadas às atividades, aos *work products* consumidos e produzidos e aos *roles* responsáveis por realizar as tarefas. A Figura 19 mostra exatamente a informação que é apresentada quando uma atividade, neste caso, a atividade *Assess Business Status* da disciplina *Business Modeling* é expandida.

Ao longo deste estudo, foram analisadas todas as atividades das disciplinas em análise, para assim, se ter acesso às tarefas que estão associadas a essas atividades, aos *work products* e aos *roles* responsáveis.

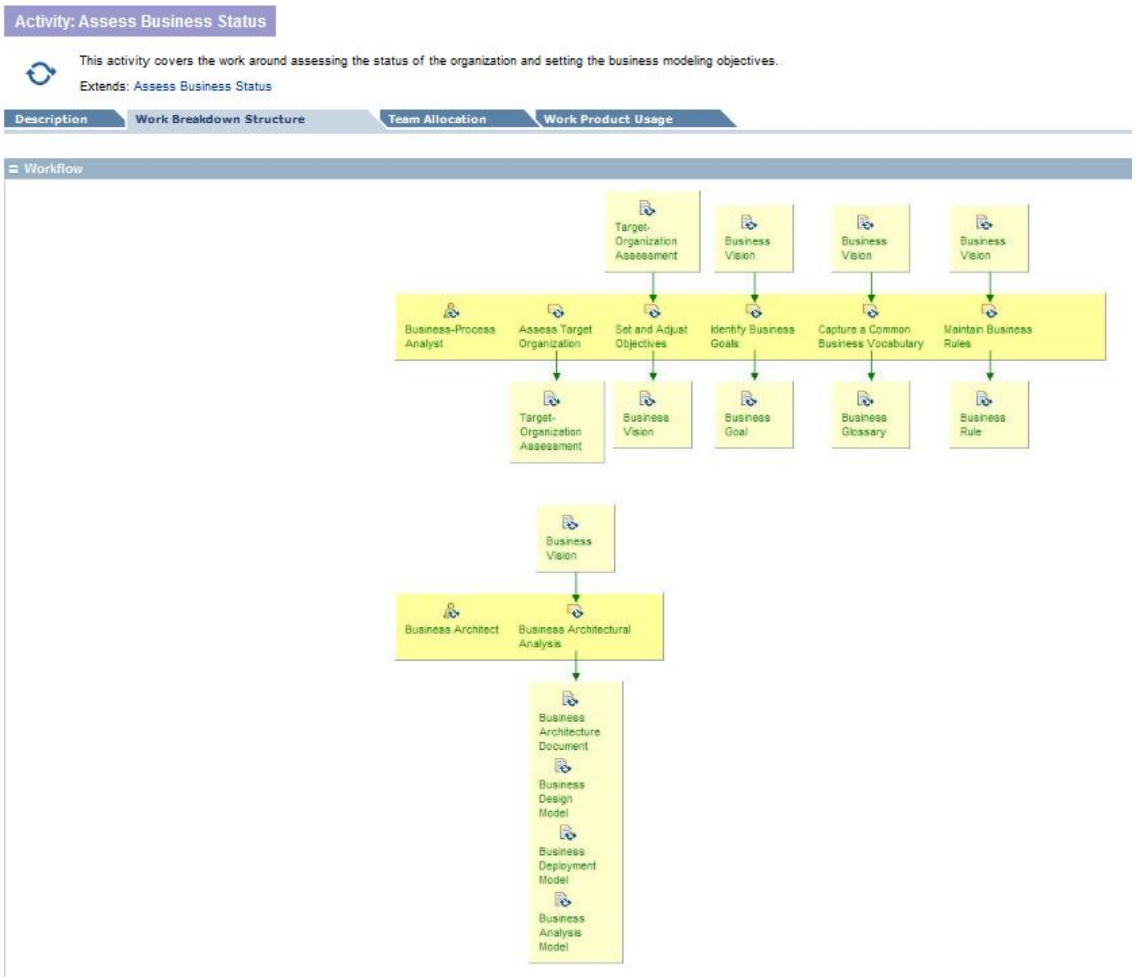


Figura 19 – Expansão da atividade *Assess Business Status*

Fonte: (Rational Unified Process)

Através destas análises, apresentadas anteriormente, foi possível reunir a informação necessária para dar início a este estudo. Estas análises permitiram reunir para as disciplinas/fases em estudo todas as atividades executadas, assim como, as suas tarefas e *roles* associados.

A Tabela 1, que foi realizada com base na informação analisada anteriormente, é referente às atividades, às tarefas, às fases e aos *roles* da disciplina *Business Modeling.*

Na coluna ***Activities*** apresentam-se as cinco atividades praticadas nesta disciplina, o facto das atividades serem praticadas em processos distintos faz com que seja necessário identificar em que processo as atividades são executadas. As atividades que são executadas no processo *Classic RUP Lifecycle* são representadas na tabela por um *****, as atividades que são executadas

no processo *Business Modeling Lifecycle* são representadas por um***'** e as atividades que são praticadas em ambos os processos são representadas na tabela por um ****'**.

A coluna ***Tasks*** expõe todas as tarefas praticadas na disciplina *Business Modeling*, é de referir que apenas as tarefas com fundo cinzento foram estudadas, uma vez que, as outras posicionam-se em fases que estão fora do âmbito deste estudo.

A intersecção da coluna ***Activities*** com a coluna ***Tasks*** é representada por um 'x' e refere-se em que atividades as tarefas são praticadas.

Na coluna ***Phases*** apresenta-se em que fases são praticadas as diferentes tarefas e atividades. Note-se que são usadas as abreviaturas **B1, B2, B3, B4** e **B5** (referentes às diversas atividades) para associar as tarefas e suas atividades às fases onde são praticadas.

A coluna ***Role main*** refere quem são os *roles* responsáveis por realizar as diversas tarefas.

As atividades com fundo azul (*Assess Business Status*, *Describe Current Business* e *Develop Domain Model*) e a fase (*Inception*) referem-se às atividades e à fase que foram estudadas na disciplina *Business Modeling.*

Através desta Tabela 1 pode-se, por exemplo, verificar que a tarefa *Detail a Business Entity – DBE* é executada em duas atividades. Na atividade *Define Business* do processo *Business Modeling Lifecycle* nas fases *construction* e *transition* e na atividade *Develop Domain Model* do processo *Classic RUP Lifecycle* na fase *inception.* E o *role* responsável por executar essa tarefa é o *Business Designer.*

Tabela 1 – Atividades, tarefas, fases e *roles* da disciplina *Business Modeling*

Business Modeling													
Activities						Tasks	Phases				Role main
B1 Assess Business Status**	B2 Describe Current Business**	B3 Define Business**	B4 Explore Process Automation *	B5 Develop Domain Model*		Inception	Elaboration	Construction	Transition	
x					Assess Target Organization - **ATO**	B1				Business-Process Analyst
x	x	x		x	Business Architectural Analysis - **BAA**	B1, B2, B5	B2, B3	B3	B3	Business Architect
		x			Business Operation Analysis		B3		B3	Business Designer
		x			Business Operation Design		B3	B3	B3	Business Designer
		x			Business Use-Case Analysis		B3	B3	B3	Business Designer
x	x	x		x	Capture a Common Business Vocabulary - **CCBV**	B1, B2, B5	B2, B3	B3		Business-Process Analyst
			x		Construct Business Architectural Proof-of-Concept		B4			Business Architect
			x		Define Automation Requirements		B4	B4	B4	Business Designer
		x			Define Business System Context		B3			Business-Process Analyst
		x		x	Detail a Business Entity - **DBE**	B5		B3	B3	Business Designer
		x			Detail a Business Use Case			B3		Business Designer
		x			Detail a Business Worker			B3	B3	Business Designer
	x	x			Find Business Actors and Use Cases - **FBAUC**	B2	B2, B3	B3		Business-Process Analyst
x	x	x			Identify Business Goals - **IBG**	B1, B2	B3			Business-Process Analyst
x	x	x		x	Maintain Business Rules - **MBR**	B1, B2, B5	B2, B3	B3	B3	Business-Process Analyst
		x			Prioritize Business Use Cases		B3	B3		Business Architect
		x		x	Review the Business Analysis Model - **RBAM**	B5	B3	B3	B3	Technical Reviewer
		x			Review the Business Use-Case Model		B3	B3		Technical Reviewer
x	x	x			Set and Adjust Objectives - **SAO**	B1, B2	B3			Business-Process Analyst
		x			Structure the Business Use-Case Model		B3	B3		Business-Process Analyst

Activities	
Classic RUP Lifecycle	*
Business Modeling Lifecycle	**
Classic RUP Lifecycle + Business Modeling Lifecycle	***

A Tabela 2, que também foi elaborada com base na informação analisada inicialmente, é referente às atividades, às tarefas, às fases e aos roles da disciplina *Requirements*.

Na coluna ***Activities*** apresentam-se as seis atividades praticadas nesta disciplina, tal como já foi referido anteriormente, o facto das atividades serem praticadas em processos distintos faz com que seja necessário identificar em que processo as atividades são executadas. As atividades que são executadas no processo *Classic RUP Lifecycle* são representadas na tabela por um *****, as atividades que são executadas no processo *Business Modeling Lifecycle* são representadas por um***'** e as atividades que são praticadas em ambos os processos são representadas na tabela por um ****'**.

A coluna ***Tasks*** expõe todas as tarefas praticadas na disciplina *Requirements*, nesta disciplina todas as tarefas têm fundo cinzento, pois, todas elas se posicionam nas fases do âmbito deste estudo, e como tal, todas foram estudadas.

A intersecção da coluna ***Activities*** com a coluna ***Tasks*** é representada por um 'x' e refere-se em que atividades as tarefas são praticadas.

Na coluna ***Phases*** apresenta-se em que fases são praticadas as diferentes tarefas e atividades. Note-se que são usadas as abreviaturas **R1, R2, R3, R4, R5** e **R6** (referentes às diversas atividades) para associar as tarefas e suas atividades às fases onde estas são praticadas. Em algumas intersecções usa-se ainda a simbologia (***** -*Classic RUP Lifecycle,* ***'** -*Business Modeling Lifecycle e* ****'** -ambos os processos) que se refere ao processo onde as tarefas e suas atividades são executadas, uma vez que, há tarefas que são executadas em determinadas atividades e fases apenas num processo. Isto é, por exemplo, a tarefa *Develop Requirements Management Plan* - DRMP é executada na atividade *Analyze the Problem,* apesar de esta atividade ser executada em ambos os processos, esta tarefa apenas é executada na fase *inception* do processo *Classic RUP Lifecycle.*

A coluna ***Role main*** refere quem são os *roles* responsáveis por realizar as diversas tarefas.

As atividades com fundo azul (*Analyze the Problem*, *Understand Stakeholder Needs*, *Define the System*, *Manage the Scope of the System*, *Refine the System Definition* e *Manage Changing Requirements*) e as fases (*Inception* e *Elaboration*) referem-se às atividades e às fases que foram estudadas na disciplina *Requirements.*

Através desta Tabela 2 pode-se, por exemplo, verificar que a tarefa *Develop Supplementary Specifications – DSS* é executada em três atividades. Na fase *inception* é executada na atividade *Understand Stakeholder Needs* no processo *Classic RUP Lifecycle* e na atividade *Define the System* no processo *Classic RUP Lifecycle.* Na fase *elaboration* é executada na atividade *Refine the System Definition* no processo *Classic RUP Lifecycle.* E na fase *transition* é executada na atividade *Understand Stakeholder Needs* no processo *Business Modeling Lifecycle.* Verifica-se ainda, através desta tabela que o *role* responsável por executar esta tarefa é o *System Analyst.*

Tabela 2 – Atividades, tarefas, fases e *roles* da disciplina *Requirements*

Requirements	Activities							Tasks	Phases				Role main
	R1 Analyze the Problem**	R2 Understand Stakeholder Needs**	R3 Define the System*	R4 Manage the Scope of the System*	R5 Refine the System Definition*	R6 Manage Changing Requirements*			Inception	Elaboration	Construction	Transition	
	X	X	X					Capture a Common Vocabulary - **CCV**	R1*, R2*, R3			R1*, R2*	System Analyst
					X			Detail a Use Case - **DUC**		R5			Requirements Specifier
					X			Detail the Software Requirements - **DSR**		R5			Requirements Specifier
	X							Develop Requirements Management Plan - **DRMP**	R1*				System Analyst
		X	X		X			Develop Supplementary Specifications - **DSS**	R2*, R3	R5		R2*	System Analyst
	X	X	X	X				Develop Vision - **DV**	R1*, R2*, R3, R4			R1*, R2*	System Analyst
		X						Elicit Stakeholder Requests - **ESR**	R2*			R2*	System Analyst
	X	X	X					Find Actors and Use Cases - **FAUC**	R1*, R2*, R3			R1*, R2*	System Analyst
		X	X	X		X		Manage Dependencies - **MDep**	R2*, R3, R4	R6	R6	R6	System Analyst
				X				Prioritize Use Cases - **PUC**	R4				Software Architect
						X		Review Requirements - **RReq**		R6	R6	R6	Technical Reviewer
						X		Structure the Use-Case Model - **SUCM**		R6	R6	R6	System Analyst

Activities/ Phases	
Classic RUP Lifecycle	*
Business Modeling Lifecycle	*:
Classic RUP Lifecycle + Business Modeling Lifecycle	**:

Através da Tabela 1 e da Tabela 2 tem-se a possibilidade de visualizar quais as diferentes tarefas das disciplinas de *Business Modeling* e de *Requirements*, quais as atividades associadas a essas mesmas tarefas, em que fases são realizadas e quem são os roles responsáveis por as realizar.

A informação disponibilizada nestas tabelas vai ser útil ao longo do desenvolvimento de *software*, pois, vai permitir perceber como se relacionam as atividades, as tarefas e os *roles* de uma determinada disciplina e fase.

3.4 Análise de *Work Products* e tarefas

Com base no estudo efetuado para elaborar a Tabela 1 e a Tabela 2, e o qual permitiu perceber quais as tarefas e as atividades abrangidas no âmbito deste estudo, procedeu-se à elaboração da Tabela 3 e da Tabela 4. A Tabela 3 e a Tabela 4 foram elaboradas tendo como base as matrizes de rastreabilidade, apresentadas anteriormente no capítulo da revisão da literatura.

Estas tabelas (Tabela 3 e Tabela 4) foram elaboradas com o propósito de permitir interligar todos os *work products* das disciplinas em estudo às suas respetivas tarefas. Para elaborar estas tabelas foi necessário analisar, através do RUP, quais os *work products* que fazem parte das duas disciplinas em estudo. A Figura 20 refere-se aos *work products* consumidos e produzidos na disciplina *Business Modeling* e a Figura 21 refere-se aos *work products* consumidos e produzidos na disciplina *Requirements.*

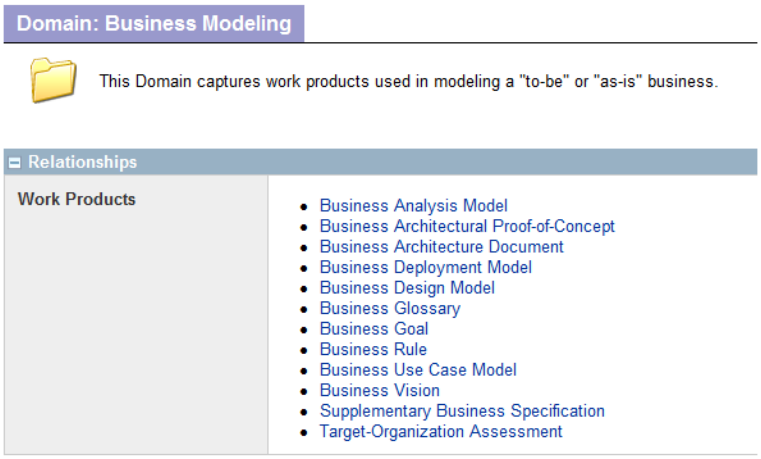


Figura 20 – *Work products* da disciplina *Business Modeling*

Fonte: (Rational Unified Process)

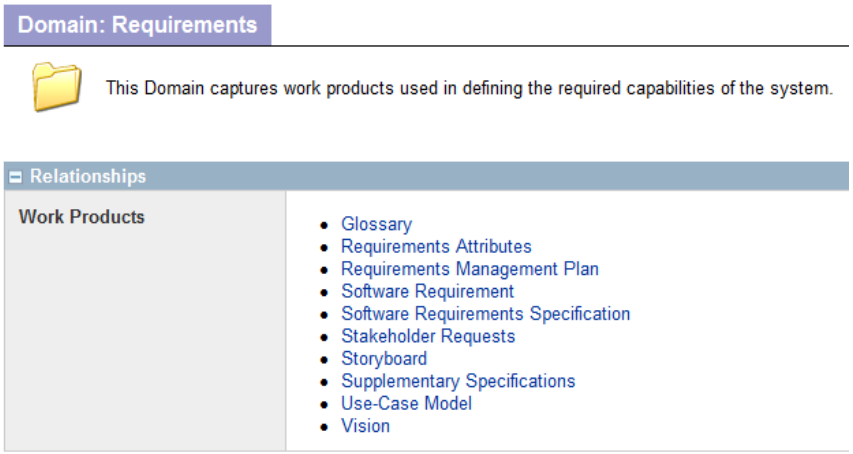


Figura 21 – *Work products* da disciplina *Requirements*

Fonte: (Rational Unified Process)

Ao aceder ao conteúdo de cada tarefa tem-se acesso à informação dos *work products* que são consumidos e produzidos pela tarefa. A Figura 22 mostra a informação disponibilizada pelo RUP quando a tarefa *Capture a Common Business Vocabulary* é acedida. Através da imagem verifica-se que esta tarefa consome o *work product Business Vision* e produz o *work product Business Glossary.*

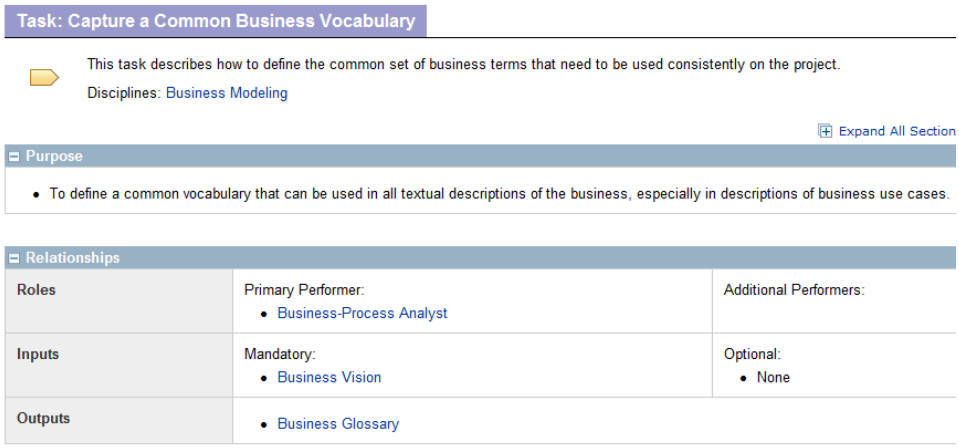


Figura 22 – Conteúdo da tarefa *Capture a Common Business Vocabulary*

Fonte: (Rational Unified Process)

Através do cruzamento da listagem dos *work products* (Figura 20 e Figura 21) com a informação disponibilizada no conteúdo de cada tarefa é possível associar os *work products* às tarefas onde são consumidos e produzidos. Este cruzamento entre os *work products* e as tarefas é feito através do uso de três termos.

O termo OUT é usado para representar os *work products* que são produzidos pela tarefa em questão. O termo IN é utilizado para referir que os *work products* são consumidos pela tarefa a que estão associados. E o termo I/O é usado para representar que os *work products* tanto são consumidos pela tarefa em questão como são produzidos pela mesma.

A Tabela 3 na primeira coluna apresenta todos os *work products* da disciplina *Business Modeling* e na segunda coluna expõe todas as tarefas. É de referir, que apenas as tarefas com fundo cinzento foram analisadas, pois tal como já foi referido anteriormente, as outras tarefas encontram-se em fases que não estão no âmbito deste estudo. A intersecção destas duas colunas retrata quais os *work products* que são consumidos e produzidos nas diversas tarefas a que estão associados, e é representada na tabela através dos termos OUT, IN e I/O, os quais já foram definidos anteriormente. Para além destes termos, usa-se também o termo IN*. Este termo refere que os *work products* são uma entrada opcional da tarefa a que estão associados, ou seja, os *work product*s não são obrigatoriamente consumidos na tarefa a que estão associados.

De forma a facilitar a identificação dos termos OUT, IN e I/O representaram-se os mesmos por cores. O termo OUT é representado pela cor vermelha, o IN pela verde e o I/O pela amarela tornando assim a informação mais percetível.

Através desta Tabela 3, pode-se por exemplo, verificar que o *work product Business Analysis Model* é produzido na tarefa *Business Architectural Analysis* e na *Business Use-Case Analysis*. E é consumido na tarefa *Business Operation Design* e na *Review the Business Analysis Model.* E nas tarefas *Business Operation Analysis* e *Define Business System Context* tanto é consumido como produzido.

Tabela 3 – *Work products* das tarefas da disciplina *Business Modeling*

Discipline	Business Modeling																				
Tasks / Work Products	Assess Target Organization – **ATO**	Business Architectural Analysis – **BAA**	Business Operation Analysis	Business Operation Design	Business Use-Case Analysis	Capture a Common Business Vocabulary – **CCBV**	Construct Business Architectural Proof-of-Concept	Define Automation Requirements	Define Business System Context	Detail a Business Entity – **DBE**	Detail a Business Use Case	Detail a Business Worker	Find Business Actors and Use Cases – **FBAUC**	Identify Business Goals – **IBG**	Maintain Business Rules – **MBR**	Prioritize Business Use Cases	Review the Business Analysis Model – **RBAM**	Review the Business Use-Case Model	Set and Adjust Objectives – **SAO**	Structure the Business Use-Case Model
Business Analysis Model – **BAM**		OUT	I/O	IN	OUT				I/O								IN			
Business Architectural Proof-of-Concept – **BAP-C**		IN *			IN *		OUT													
Business Architecture Document – **BAD**		OUT			IN		IN									I/O				
Business Deployment Model – **BDM**		OUT		OUT			IN													
Business Design Model – **BDesM**		OUT		OUT			IN													
Business Glossary – **BGl**						OUT		IN												
Business Goal – **BG**											IN			OUT						
Business Rule – **BR**															OUT					
Business Use Case Model – **BUCM**			I/O	IN	IN				I/O				OUT			I/O		IN		I/O
Business Vision – **BV**		IN				IN							IN	IN	IN	IN			OUT	
Supplementary Business Specification – **SBS**					IN				IN		OUT		OUT							
Target-Organization Assessment – **TOA**	OUT							IN											IN	

OUT	Output
IN	Input
I/O	Input/Output
IN *	O work product é um input opcional da tarefa a que está associado

A Tabela 4 na primeira coluna apresenta todos os *work products* da disciplina *Requirements* e na segunda coluna expõe todas as tarefas. É de referir, que todas as tarefas desta disciplina foram analisadas, pois tal como já foi referido anteriormente, todas as tarefas encontram-se em fases que estão no âmbito deste estudo. A intersecção destas duas colunas retrata quais os *work products* que são consumidos e produzidos nas diversas tarefas a que estão associados, e é representada na tabela através dos termos OUT, IN e I/O, os quais já forma definidos anteriormente.

De foram a facilitar a identificação dos termos OUT, IN e I/O representaram-se os mesmos por cores. O termo OUT é representado pela cor vermelha, o IN pela verde e o I/O pela amarela tornando assim a informação mais percetível.

Através desta Tabela 4, pode-se por exemplo, verificar que o *work product Software Requirement* é produzido na tarefa *Detail the Software Requirements* e na *Prioritize Use Cases*. E é consumido na tarefa *Review Requirements*.

Tabela 4 – *Work products* das tarefas da disciplina *Requirements*

Discipline	Requirements											
Tasks / Work Products	Capture a Common Vocabulary - **CCV**	Detail a Use Case - **DUC**	Detail the Software Requirements - **DSR**	Develop Requirements Management Plan - **DRMP**	Develop Supplementary Specifications - **DSS**	Develop Vision - **DV**	Elicit Stakeholder Requests - **ESR**	Find Actors and Use Cases - **FAUC**	Manage Dependencies - **MDep**	Prioritize Use Cases - **PUC**	Review Requirements - **RReq**	Structure the Use-Case Model - **SUCM**
Glossary - **Gl**	OUT											OUT
Requirements Attributes - **ReqA**		OUT	OUT		OUT	OUT	OUT	OUT	OUT	OUT		OUT
Requirements Management Plan - **RMP**				OUT					I/O			
Software Requirement - **SofR**			OUT							OUT	IN	
Software Requirements Specification - **SRS**			OUT									
Stakeholder Requests - **SR**					IN	IN	OUT	IN				
Storyboard - **St**							OUT					
Supplementary Specifications - **SS**					OUT							OUT
Use-Case Model - **UCM**								OUT		IN		I/O
Vision - **Vi**			IN			OUT			OUT			

Através da Tabela 3 e da Tabela 4 tem-se a possibilidade de visualizar quais os *work products* que são consumidos e produzidos nas diferentes tarefas.

A informação disponibilizada nestas tabelas (Tabela 3 e Tabela 4) vai ser útil ao longo do desenvolvimento de *software*, pois, vai permitir identificar quais as interdependências existentes entre as tarefas e os *work products* que são produzidos e consumidos.

Para tornar a informação das tabelas, anteriormente elaboradas, mais percetível foram elaborados dois esquemas, Figura 23 e Figura 24, os quais permitem visualizar graficamente todas as interdependências existentes entre as atividades, as tarefas e os *work products* de uma determinada disciplina e fase. Os dois esquemas foram construídos tendo como base o processo de operacionalização e atribuição de responsabilidades da abordagem orientada a atributos, mencionado na revisão da literatura e a informação das tabelas elaboradas anteriormente. Estes esquemas permitem analisar as várias interdependências existentes ao longo do processo de desenvolvimento, permitindo assim uma melhor gestão das mesmas.

O esquema da Figura 23 foi elaborado com base na informação recolhida na Tabela 1 e na Tabela 3. Este esquema refere-se à disciplina *Business Modeling* na fase *Inception.* Inicialmente apresentam-se as 5 atividades pertencentes a esta disciplina. Estas atividades estão interligadas às suas tarefas, sendo que duas dessas atividades não possuem tarefas associadas, pois como já foi referido anteriormente, encontram-se fora do âmbito deste estudo. Cada tarefa tem os seus *work products* associados. Estes tanto podem ser *work products* consumidos na tarefa a que estão associados (*inputs,* representados graficamente pela seta verde) como podem ser produzidos por essa mesma tarefa (*outputs,* representados graficamente pela seta vermelha). Os *work products* representados a amarelo referem-se aos *work products* pertencentes à disciplina *Business Modeling.* E os *work products* representados a laranja apesar de serem *work products* consumidos e produzidos nesta disciplina|fase não pertencem diretamente aos *work products* definidos pelo RUP para esta disciplina. Para estes *work products*, representados a laranja, descreve-se por baixo dos mesmos a disciplina e a fase a que pertencem, é de referir que alguns não têm disciplina associada, pois, não pertencem a nenhuma em concreto.

Este esquema da Figura 23 permite, por exemplo, visualizar que a tarefa *Capture a Common Business Vocabulary* (CCBV) é executada nas atividades *Assess Business Status* (ABS), *Describe Current Business* (DCB) e *Develop Domain Model* (DDM). E que tem como *input* o *work product Business Vision* (BV) e como *output* o *work product Business Glossary* (BGl). Este esquema é bastante útil ao longo do processo de desenvolvimento de *software*, uma vez que, permite visualizar mais facilmente todas as interdependências existentes entre as diversas atividades, tarefas e *work products* possibilitando assim, uma melhor gestão das mesmas.

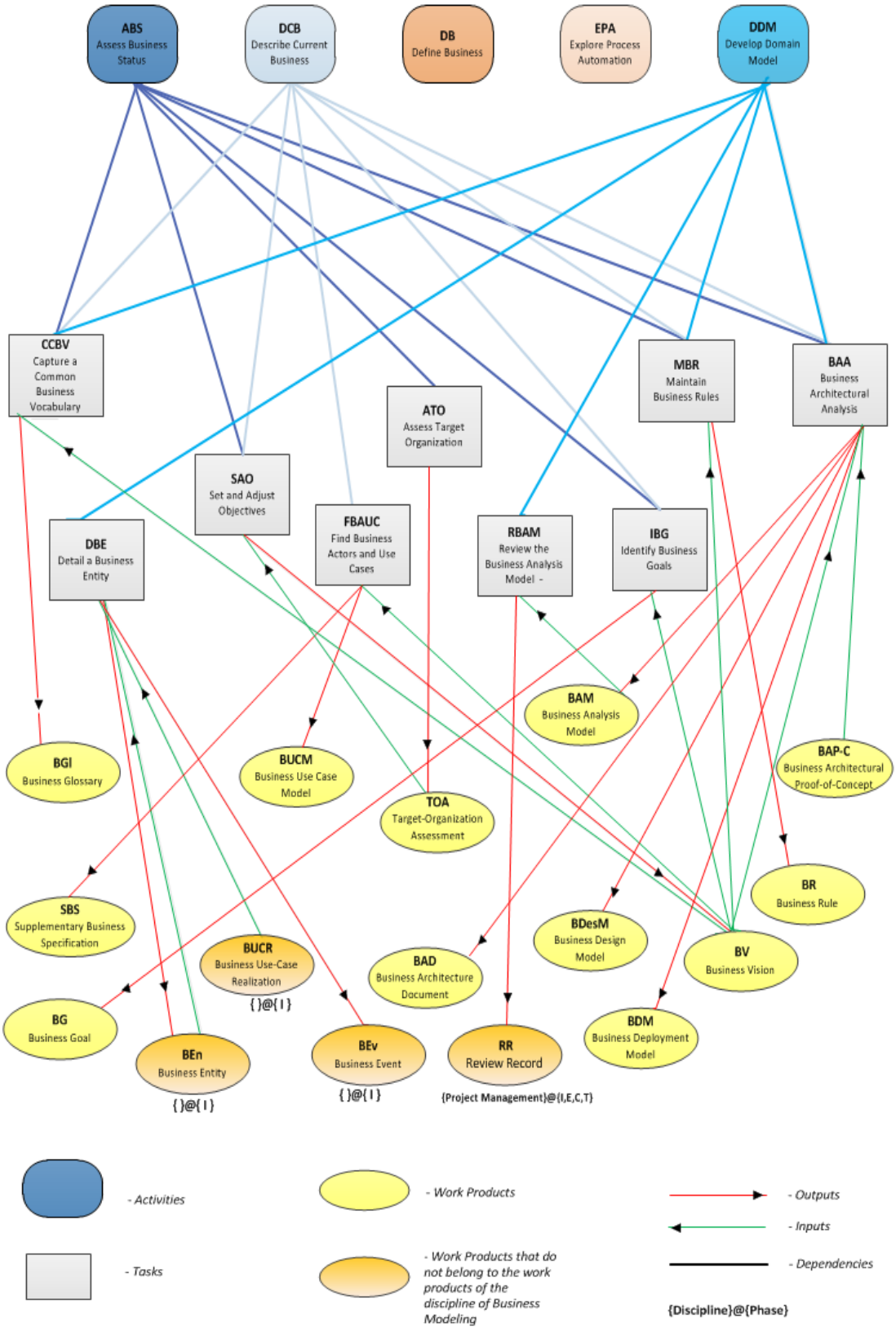


Figura 23 – Esquema *Business Modeling@Inception*

O esquema da Figura 24 foi elaborado com base na informação reunida na Tabela 2 e na Tabela 4. Este esquema refere-se à disciplina *Requirements* na fase *Inception* e *Elaboration.* Inicialmente apresentam-se as 6 atividades pertencentes a esta disciplina. Estas atividades estão interligadas às suas tarefas, todas estas atividades possuem tarefas associadas, pois como já foi referido anteriormente, todas elas se encontram no âmbito deste estudo. Cada tarefa tem os seus *work products* associados. Estes tanto podem ser *work products* consumidos na tarefa a que estão associados (*inputs,* representados graficamente pela seta verde) como podem ser produzidos por essa mesma tarefa (*outputs,* representados graficamente pela seta vermelha). Os *work products* representados a amarelo referem-se aos *work products* pertencentes à disciplina *Requirements.* Os *work products* representados a laranja apesar de serem *work products* consumidos e produzidos nesta disciplina|fase não pertencem diretamente aos *work products* definidos pelo RUP para esta disciplina. Para estes *work products*, representados a laranja, descreve-se por baixo dos mesmos a disciplina e a fase a que pertencem, é de referir que alguns não têm disciplina associada, pois, não pertencem a nenhuma em concreto. Na Figura 24 estão apresentadas duas manchas diferentes, uma representa a fase *Inception* e a outra a *Elaboration.* Pode-se verificar através destas manchas que existem duas tarefas que são tratadas em ambas as fases, a tarefa *Develop Supplementary Specification* (DSS) e a *Manage Dependencies* (MDep), verificando-se assim que existem interdependências entre as fases.

Este esquema da Figura 24 permite, por exemplo, visualizar que a tarefa *Develop Supplementary Specifications* (DSS) é executada tanto na fase *inception* como na fase *elaboration*. Esta tarefa encontra-se interligada a três atividades à *Understand Stakeholder Needs* (USN), à *Define the System* (DS) e à *Refine the System Definition* (RSD). Como *inputs* esta tarefa possui os *work products Stakeholder Requests* (SR) e *Iteration Plan* (IP). E como *outputs* possui os *work products Supplementary Specifications* (SS) e *Requirements Attributes* (ReqA). Este esquema é bastante útil ao longo do processo de desenvolvimento de *software*, uma vez que, permite visualizar mais facilmente todas as interdependências existentes entre as diversas atividades, tarefas e *work products* possibilitando assim, uma melhor gestão das mesmas.

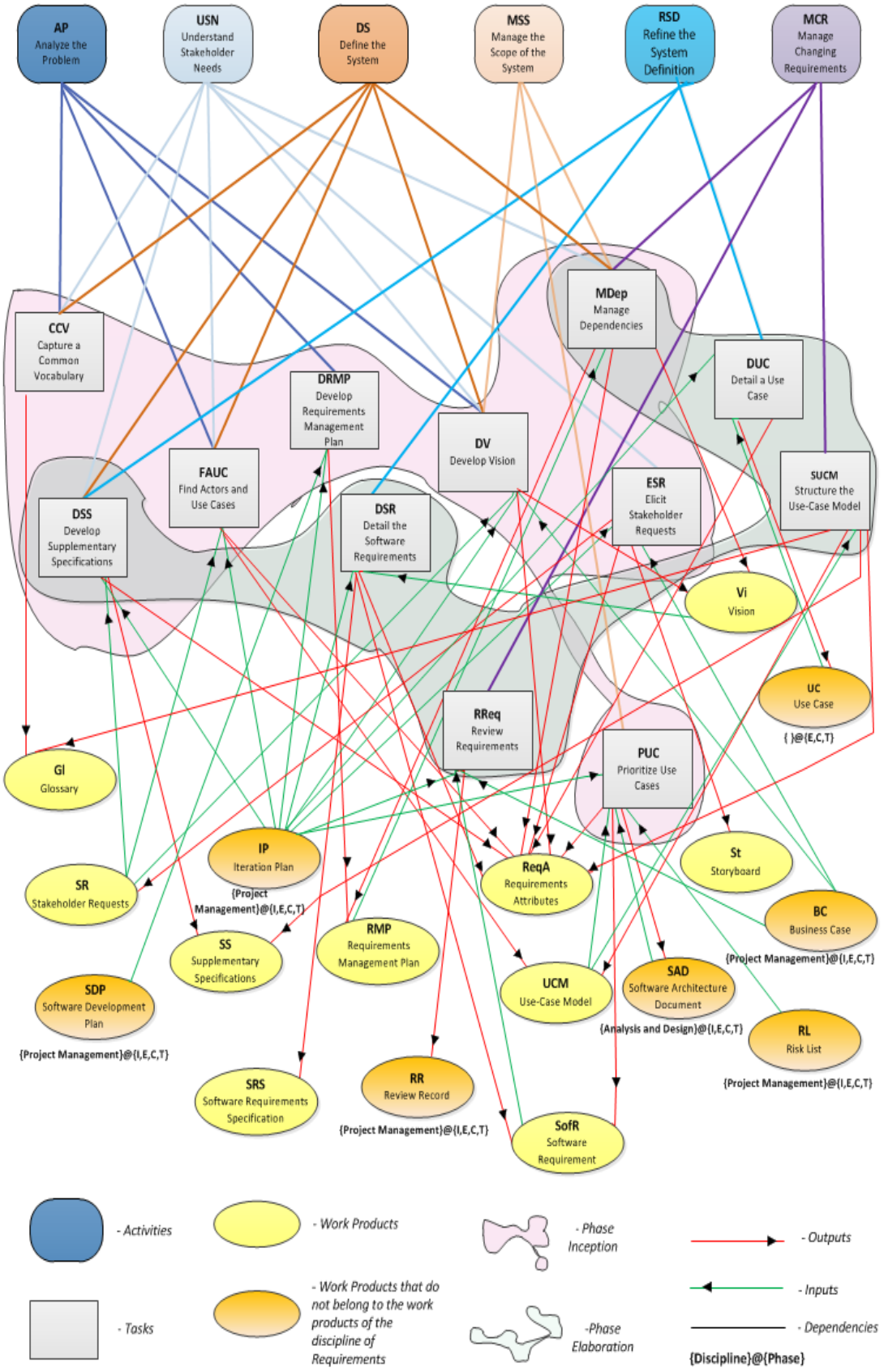


Figura 24 – Esquema *Requirements@Inception, Elaboration*

3.5 Conclusões

Conforme se analisou neste capitulo o RUP é constituído por vários elementos. Esses elementos, como já foi referido anteriormente, vão-se relacionar e se influenciar uns aos outros, isto porque são interdependentes. Este facto faz com que uma simples mudança num elemento provoque vários ajustes posteriores em outros elementos.

Ao longo de todo o processo de desenvolvimento de *software,* torna-se essencial conhecer antecipadamente quais as interdependências existentes, de forma a identificar e a gerir mais facilmente os elementos afetados aquando de uma mudança.

Através das várias analises efetuadas neste capítulo foi possível elaborar tabelas que permitem analisar as interdependências existentes entre as atividades, as tarefas, as fases e os *roles* de uma determinada disciplina, assim como, tabelas que permitem analisar as interdependências existentes entre as tarefas e os *work products* das disciplinas em análise.

Estas tabelas, assim como os esquemas que foram elaborados a partir das mesmas, permitem analisar a rastreabilidade dos diversos elementos do RUP, assim como, identificar facilmente todas as interdependências exististes entre esses mesmos elementos. Isto vai ser bastante útil, uma vez que, permite conhecer detalhadamente como os diversos elementos do processo RUP se relacionam.

Através da informação disponibilizada nestas tabelas e esquemas é possível evitar decisões inconscientes, durante o processo de desenvolvimento, assim como, detetar antecipadamente potenciais problemas que possam surgir devido às interdependências existentes.

4. Análise de cobertura do *V-Model* /4SRS no RUP

4.1 Introdução

No capítulo anterior foram criadas tabelas e esquemas que permitem analisar a rastreabilidade e as interdependências existentes entre os diversos elementos que constituem o RUP. É com base na informação recolhida através dessas tabelas, que neste capítulo são elaboradas outras tabelas.

As tabelas criadas neste capítulo têm como finalidade permitir avaliar a cobertura de um determinado método/modelo de processo com o RUP, com base no conhecimento adquirido no capítulo anterior, de como os diversos elementos se relacionam. Através destas tabelas é analisada a cobertura do micro-processo *V-Model* e do método 4SRS nos diversos elementos que constituem o RUP.

O *V-Model* tem como finalidade derivar modelos de arquitetura lógica para executar nas diferentes camadas da *cloud*, a partir de uma perspectiva de nível de processo, em vez da perspectiva tradicional de nível de produto (Ferreira N. , Santos, Machado, Fernandes, & Gašević, 2011). Esta abordagem *V-Model* engloba a definição inicial dos objetivos do projeto, expressos através das configurações organizacionais, as análises e a concepção de artefactos que resulta numa perspectiva de nível de processo da arquitetura lógica do sistema (Ferreira N. , Santos, Machado, Fernandes, & Gašević, 2011).

O método 4SRS é principalmente baseado no mapeamento de diagramas de casos de uso para diagramas de objetos (Machado R. J., Fernandes, Monteiro, & Rodrigues, 2006). Os casos de uso atuam como entrada para o método 4SRS sendo que, na perspetiva de nível de processo 4SRS representam as atividades (processos) executadas pelas pessoas ou máquinas no âmbito do sistema e na perspetiva de nível de produto representam as características (requisitos) dos produtos destinados a serem desenvolvidos (Ferreira N. , Santos, Machado, & Gašević, 2012).

Com as tabelas criadas neste capítulo pretende-se demonstrar que o conhecimento da rastreabilidade e das interdependências é fundamental quando se analisa a cobertura de um determinado método/modelo de processo com o RUP. Estas tabelas vão permitir verificar se o

que o RUP recomenda está a ser seguido pelo método/modelo de processo adotado e como está a ser seguido.

4.2 Síntese do micro-processo *V-Model* e do método 4SRS

Quando os requisitos não são devidamente levantados e há insuficientes entradas (*inputs*) para uma abordagem de produto para o levantamento requisitos, uma perspetiva de nível de processo é uma forma alternativa para alcançar os requisitos principais pretendidos para a conceção lógica (Ferreira N. , Santos, Machado, & Gašević, 2012). É de referir que, na perspetiva de nível de processo os casos de uso representam as atividades (processos) executadas pelas pessoas ou máquinas no âmbito do sistema. E na perspetiva de nível de produto os casos de uso representam as características (requisitos) dos produtos destinados a serem desenvolvidos (Ferreira N. , Santos, Machado, & Gašević, 2012).

Ferreira N. et al. (2011) em vez de conceberem uma arquitetura de *cloud computing* baseada nos requisitos do utilizador, defenida tradicionalmente numa perspetiva de nível de produto, propõem o uso de uma perspetiva de nível de processo para a definição de requisitos e conceção do modelo lógico da arquitetura do sistema. Estes referem que usar uma perspetiva de nível de processo, em vez de uma perspetiva de nível de produto, contribui para uma definição mais precisa dos requisitos do produto e uma melhor compreensão do âmbito do projeto (Ferreira N. , Santos, Machado, Fernandes, & Gašević, 2011) .

A abordagem V-*Model* é constituida, tal como apresenta a Figura 25, por configurações organizacionais (OCs), diagramas de sequência do tipo A e tipo B, modelos de casos de uso e um diagrama da arquitetura lógica a nível de processo (Ferreira N. , Santos, Machado, Fernandes, & Gašević, 2011).

Uma configuração organizacional (OC) modela uma possível relação inter-organizacional num muito alto nível de abstração e não considera processos de baixo nível e/ou atores envolvidos na relação. Uma caracterização OC deve conter informação sobre as atividades realizadas, os vários perfis dos profissionais (atores) que participam na execução da atividade e também a troca de informação ou artefactos (Ferreira N. , Santos, Machado, Fernandes, & Gašević, 2011).

Os diagramas de sequência do tipo A são utilizados para reunir e formalizar as intenções dos *stakeholders* principais, os quais fornecem uma instrumentação e uma sequência de algumas

atividades previstas. Os diagramas de sequência do tipo A relacionam os *roles* apresentados nas OCs e instanciam as suas relações com atividades (Ferreira N. , Santos, Machado, Fernandes, & Gašević, 2011).

Os diagramas de sequência o tipo B modelam a troca de informação entre os atores e os elementos arquiteturais lógicos, portanto, eles ainda são modelados a nível do sistema. Estes podem ser usados para validar a arquitetura lógica derivada, através da deteção de elementos arquiteturais em falta e/ou associações, a fim de executar um determinado processo na arquitetura lógica derivada (Ferreira N. , Santos, Machado, Fernandes, & Gašević, 2011).

Na abordagem *V-Model* os artefactos são gerados com base na racionalidade e na informação existente em artefactos previamente definidos, isto é, os diagramas de sequência do tipo A são baseados nas configurações organizacionais, o modelo de casos de uso é baseado nos diagramas de sequência do tipo A, a arquitetura lógica é baseada no modelo de casos de uso, e os diagramas de sequência do tipo B cumprem a arquitetura lógica (Ferreira N. , Santos, Machado, Fernandes, & Gašević, 2011).

No *V-Model*, tal como apresenta a Figura 25, os artefactos colocados no lado esquerdo da representação estão adequadamente alinhados com os artefactos colocados no lado direito, isto é, os diagramas de sequência do tipo B são alinhados com os diagramas de sequência do tipo A e a arquitetura lógica esta alinhada com os modelos de casos de uso (Ferreira N. , Santos, Machado, Fernandes, & Gašević, 2011). O alinhamento entre o modelo de casos de uso e a arquitetura lógica é assegurado pela correta aplicação do método 4SRS. A representação do *V-Model* promove o alinhamento entre os artefactos no domínio do problema e os artefactos no domínio da solução (Ferreira N. , Santos, Machado, Fernandes, & Gašević, 2011).

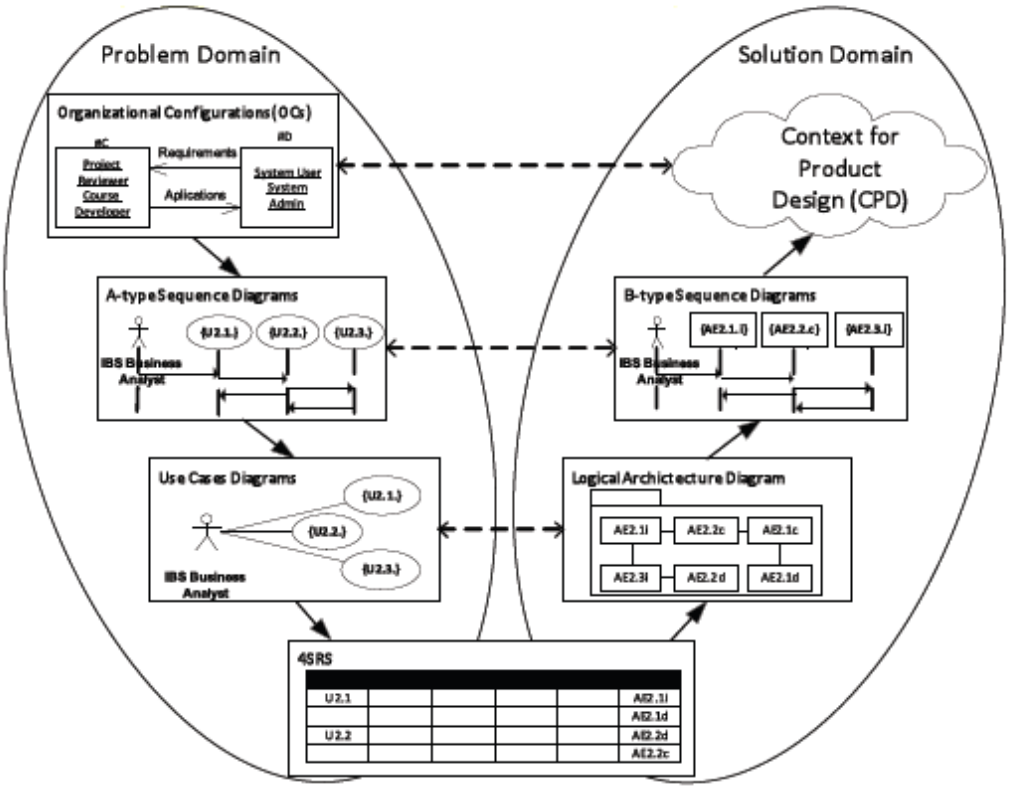


Figura 25 – Representação do *V-Model*

Fonte: (Ferreira N. , Santos, Machado, Fernandes, & Gašević, 2011).

O método 4SRS permite a transformação de requisitos de utilizador numa representação do modelo arquitetural e é tradicionalmente aplicado numa perspetiva de nível de produto (Machado R. J., Fernandes, Monteiro, & Rodrigues, 2006).

Ferreira N. et al. (2011) apresentaram uma extensão do uso tradicional do método 4SRS (perspetiva de nível de produto) para permitir a sua aplicação numa perspetiva de nível de processo, apoiando a criação de contexto para o levantamento de requisitos de nível de produto. Essa aplicação difere da tradicional ao definir um conjunto de regras que devem ser observadas, aquando o raciocínio sobre a execução das etapas do método. A extensão do método define também micro passos adicionais aos existentes. É de referir que, será essa extensão do método 4SRS a analisada neste trabalho.

O método 4SRS recebe como entrada (*input*) um conjunto de casos de uso que descrevem os requisitos para os processos específicos que abordam o problema inicial, sendo que, esses casos de uso são refinados através de sucessivas iterações 4SRS (Ferreira N. , Santos, Machado, Fernandes, & Gašević, 2011).

A aplicação do método 4SRS requer a criação de elementos arquiteturais (AEs). A natureza dos AEs varia de acordo com o tipo de sistema em estudo e também com o contexto onde é aplicado. No contexto específico de arquiteturas lógicas, o termo elemento arquitetural refere-se às partes a partir das quais a arquitetura lógica final pode ser construída.

A execução dos passos de transformação do método 4SRS pode ser suportada numa representação em tabela como detalhado em (Machado R. J., Fernandes, Monteiro, & Rodrigues, 2006).

Em seguida, apresentam-se resumidamente as extensões feitas por Ferreira N. et al. (2011) à perspetiva de nível de processo do método 4SRS e os micro passos adicionais (os passos originais do 4SRS de nível de produto são mencionados em (Machado R. J., Fernandes, Monteiro, & Rodrigues, 2006)).

Passo 1 – Criação de elementos arquiteturais

Neste passo, cada caso de uso deve ser transformado em três AEs (*interface*, *dados* e controlo). Os estereótipos *i-type*, *d-type* e *c-type* são adicionados a cada AE e os seus nomes são precedidos com "AE". Nenhuma razão particular ou decisão é necessária neste passo, uma vez que, diz respeito principalmente à transformação de um caso de uso em três AEs específicos.
Uma adição a este passo é a identificação de elementos cola (*glue*) resultantes da descrição textual associada com os casos de uso em análise. Se o caso de uso retrata pré- ou pós-condições em forma de validações pode ser expresso neste passo como uma AE cola. Estes AEs têm o estereótipo *c-type*, uma vez que, requerem decisões a serem efetuadas com suporte computacional, isto é, devem ser suportadas pela arquitetura do sistema a ser representada.
Este passo é representado na 1ª e 2ª coluna da Tabela 5.

Tabela 5 – Passo 1 do método 4SRS

Fonte: (Ferreira N. , Santos, Machado, Fernandes, & Gašević, 2011).

Step 1 -architectural element creation	
Use Case	Description
{UI.9.}	Send info to IBS
{AE1.9.c2}	Glue AE
{AE1.9.i}	Generated AE

Passo 2 – Eliminação de elementos arquiteturais

O arquiteto de sistema neste passo decide quais dos três AEs (i, c, d) criados no passo 1 e mais alguns elementos cola são mantidos ou eliminados, tendo em conta todo o sistema.

O passo 2 original do método 4SRS está dividido em sete micro-passos, nesta versão foi adicionado um novo micro-passo, o 2viii - especificação dos elementos arquiteturais. Através deste novo micro-passo, o passo 2 tornou-se mais robusto e detalhado.

-Micro-passo 2i: Classificação dos casos de uso

Neste micro-passo, cada caso de uso é classificado de acordo com a natureza dos seus AEs criados anteriormente no passo 1. A 2ª coluna da Tabela 6 corresponde à execução deste micro-passo. A natureza de um AE é definida de acordo com o sufixo que o AE foi marcado. Cada tipo de AE deve ser interpretado como se segue:

- *i-type* – refere-se à *interface*. E representa as *interfaces* do processo com os utilizadores, o *software* ou outros processos.
- *c-type* – refere-se ao controlo. E representa um processo com foco na tomada de decisão e essa decisão deve ter um suporte computacional, dado a partir do sistema global pretendido.
- *d-type* – refere-se a repositórios de decisão (dados), não suportados computacionalmente a partir do sistema global pretendido.

-Micro-passo 2ii: Eliminação local

Neste micro-passo determina-se quais os AEs que devem ser eliminados no contexto de um caso de uso, garantindo a sua representação completa. Isto é necessário, pois o micro-passo 2i desconsidera quaisquer preocupações de representatividade.

A 3ª coluna da Tabela 6 corresponde à execução deste micro-passo. As células são preenchidas com "T" ou "F". Sendo que "T" significa que o AE vai ser eliminado e "F" que o AE mantem-se vivo.

-Micro-passo 2iii: Nomenclatura dos elementos arquiteturais

Neste micro-passo (4ª coluna da Tabela 6), os AEs que não foram eliminados no micro-passo anterior devem receber um nome adequado que reflita tanto o papel do AE na totalidade do caso de uso, como o caso de uso do qual o AE foi originado.

-Micro-passo 2iv: Descrição dos elementos arquiteturais

Cada AE nomeado resultante do micro-passo anterior deve ser descrito e os requisitos que ele representa devem ser abordados na perspetiva de nível de processo. Este micro-passo é onde a transição do domínio do problema para o domínio da solução é feita, assim, as descrições devem detalhar em termos de processo, como, porquê, quando e por quem o AE vai ser executado. Este micro-passo é representado na 5ª coluna da Tabela 6.

Tabela 6 – Micro-passos 2i ao 2iv do método 4SRS

Fonte: (Ferreira N. , Santos, Machado, Fernandes, & Gašević, 2011).

	Step 2 - architectural element elimination			
	2i - use case classification	2ii - local elimination	2iii - architectural element naming	2iv - architectural element description
{U1.9.}	i			
{AE1.9.c2}		F	Validate Remote Business Program	Execute the necessary verification procedures to ensure that the Remote Business Program is …
{AE1.9.i}		F	Send Commands to IBS	Send commands and associated information to the IBS in order to process a business request…

-Micro-passo 2v: Representação dos elementos arquiteturais

Este micro-passo tem como objetivo eliminar a redundância de AEs no processo global. Todos os AEs são considerados e comparados, a fim de identificar se um AE é representado por qualquer outro. Este micro-passo é considerado o mais crítico do método 4SRS, porque a eliminação da redundância assegura uma coerência semântica da arquitetura lógica e descobre anomalias no modelo de casos de uso.

A execução deste micro-passo é representada na Tabela 7 na 2ª e 3ªcolunas. A 2ª coluna "*represented by*" (representado por) armazena a referência do AE que irá representar o AE a ser analisado. A 3ª coluna "*represents*" (representa) armazena a referência dos objetos que o AE analisado irá representar.

Tabela 7 – Micro-passos 2v ao 2viii do método 4SRS

Fonte: (Ferreira N. , Santos, Machado, Fernandes, & Gašević, 2011).

	Step 2 - architectural element elimination				
	2v - architectural element representation		2vi - global elimination	2vii - architectural element renaming	2viii - architectural element specification
	represented by	represent			
{U1.9.}					
{AE1.9.c2}	(AE1.9.c2)	(AE1.1.c2)	F	Validate Platform Access	Execute the necessary verification procedures to ensure that subscribed ISOFIN Customers…
{AE1.9.i}	(AE1.9.i)		F	Send Commands to IBS	

-Micro-passo 2vi: Eliminação global

Neste micro-passo determina-se quais os AEs que devem ser eliminados no contexto do modelo global e é apresentado na 4ª coluna da Tabela 7. Os AEs representados por si só ou que representam outros AEs são mantidos, e como tal, preenche-se a célula da tabela com "F". Os restantes (isto é, os AEs que são representados por outros AEs) são eliminados, preenchendo-se assim a célula da tabela com "T". Este micro-passo é completamente "automático", uma vez que, é baseado nos resultados anteriores.

-Micro-passo 2vii: Renomeação dos elementos arquiteturais

O objetivo deste micro-passo, representado na 5ª coluna da Tabela 7, é renomear os AEs que não foram eliminados no micro-passo anterior. Nos casos em que o AE em análise resulta da representação de mais de um AE, o nome deve refletir a execução global do AE no contexto do projeto.

-Micro-passo 2viii: Especificação dos elementos arquiteturais

Este micro-passo, representado na 6ª coluna da Tabela 7, nunca foi considerado em versões anteriores do método tradicional 4SRS. E pretende descrever AEs que no micro-passo 2v, foram considerados para representar outros AEs. Este micro-passo surgiu devido à necessidade de definir claramente o comportamento adequado do "novo" AE, de forma que seja claro para os arquitetos de sistema. A especificação dos AEs deve incluir as pré-condições dos AEs base, para que possa adequadamente apoiar as associações a serem estabelecidas no passo 4.

A especificação deve clarificar os arquitetos de sistema de que modo os AEs são executados e como a sua execução representa um AE eliminado.

Passo 3 - Empacotamento e agregação

Este passo é representado na 2ª coluna da Tabela 8. Neste passo os AEs que se mantiveram depois da execução do passo 2 devem dar origem a agregações ou pacotes de AEs semanticamente consistentes. Este passo apoia a construção de um modelo de nível de processo verdadeiramente coerente.

A técnica de empacotamento contribui para uma obtenção temporária de um modelo de processos mais abrangente e compreensível.

A agregação é usada quando há uma parte do sistema que constitui um sub-sistema legado, ou quando a conceção tem uma arquitetura de referência pré-definida que restringe o modelo.

Tabela 8 – Passo 3 do método 4SRS

Fonte: (Ferreira N. , Santos, Machado, Fernandes, & Gašević, 2011).

	Step 3 - packaging & aggregation
{U1.9.}	
{AE1.9.c2}	{P6} ISOFIN Platform Management
{AE1.9.i}	{P2.4} IBS

Passo 4 – Associação dos elementos arquiteturais

As decisões relativas à identificação de associações entre AEs podem ser baseadas em informações contidas no modelo de casos de uso e no micro-passo 2i. Nesta nova versão do 4SRS, o passo 4 foi dividido em dois micro-passos: o micro-passo 4i) associações diretas e o 4ii) associações de casos de uso.

O micro micro-passo *4i) associações diretas* é representado na 2ª coluna da Tabela 9. As associações diretas são as que derivam dos AEs originados pelo mesmo caso de uso. Estas associações são representadas a partir da classificação dada no micro-passo 2i do método.

O micro-passo *4ii) associações do modelo de casos de uso* é representado na 3ª coluna da Tabela 9. As associações do modelo de casos de uso são as que podem ser inferidas a partir das descrições textuais de casos de uso, ou seja, quando uma descrição do caso de uso se refere implicitamente ou explicitamente a outro caso de uso, as associações inferidas implicam que os casos de uso sejam conectados.

Tabela 9 – Passo 4 do método 4SRS

Fonte: (Ferreira N. , Santos, Machado, Fernandes, & Gašević, 2011).

	Step 4 - architectural element association	
	4i - Direct Associations	4ii - UC Model Associations
{U1.9.}		
{AE1.9.c2}	{AE1.1.i}, {AE1.9.c1}, {AE1.9.i}.	{AE3.3.i}.
{AE1.9.i}	{AE1.9.c1}, {AE1.9.c2}.	{AE1.7.i}, {AE2.9.i}, {AE3.3.i}.

4.3 Tabelas de cobertura elementar

Com base no estudo efetuado anteriormente, o qual permitiu analisar as diversas interdependências existentes entre as atividades, as tarefas, os *roles* e os *work products* de uma determinada disciplina e fase do RUP, foram elaboradas 32 tabelas elementares, as quais se encontram em anexo (Anexo I – Anexo XXXII). A Tabela 10 e a Tabela 11 são exemplares dessas tabelas.

Estas tabelas de cobertura elementar têm o propósito de permitir comparar cada *work product* do RUP com os vários elementos que constituem um determinado método/modelo de processo. Permitindo assim, verificar se o que o RUP recomenda ao longo do desenvolvimento de *software* está ou não a ser seguido pelo método/modelo de processo adotado. Esta análise só é possível devido aos conhecimentos adquiridos com a sistematização das interdependências existentes no processo RUP. Com estas tabelas de cobertura elementar vai ser possível verificar se o que o RUP recomenda é muito coberto, pouco coberto ou não é coberto no método/modelo de processo adotado.

Para elaborar estas tabelas foi necessário recorrer à informação disponibilizada nas tabelas apresentadas no capítulo anterior e ainda escolher um método/modelo de processo para comparar com o que o RUP recomenda ao longo do desenvolvimento de *software*. Tal como já foi referido anteriormente, o método/modelo de processo escolhido para este estudo foi o micro-processo *V-Model* e o método 4SRS.

Ao longo das tabelas de cobertura elementar os diversos *work products* do RUP são comparados com os diversos elementos que constituem o *V-Model*// 4SRS. A cada elemento *V-Model*// 4SRS é atribuída uma avaliação. A avaliação resulta da comparação da descrição que o RUP faz dos diversos *work products* com a descrição dos elementos *V-Model*// 4SRS, analisando-se assim, se o que o RUP recomenda esta a ser coberto pelo método/modelo de processo adotado.

Essa avaliação consiste na atribuição de uma das três terminologias apresentadas na Figura 26:

-*High Coverage* (**HC**, representada a cor verde) - significa que o *work product* que está a ser avaliado possui uma elevada cobertura no *V-Model*// 4SRS.

-*Low Coverage* (**LC**, representada a cor amarela) – representa que o *work product* que está a ser alvo de avaliação apresenta uma baixa cobertura no *V-Model*// 4SRS.

-*Not Covered* (**NC**, representada a cor vermelha) – indica que o *work product* em avaliação não é coberto no *V-Model/4SRS*.

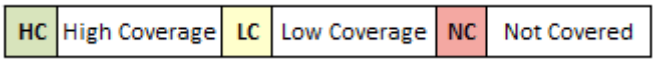


Figura 26 – Terminologias usadas para avaliar os *work products*

Depois de comparar o *work product* com cada um dos elementos que constituem o método/modelo de processo, resulta uma avaliação final a qual revela se o *work product* em questão é ou não coberto pelo *V-Model*/4SRS. É de referir que, a avaliação final é baseada nas avaliações dadas aos diversos elementos.

Como se pode verificar pela Tabela 10 e Tabela 11, estas tabelas são constituídas por três colunas.

A primeira coluna destas tabelas é para identificar o nome correspondente ao *work product* que está a ser avaliado, assim como, a avaliação final resultante das restantes avaliações.

A segunda coluna corresponde à avaliação do *work product* com os diversos elementos que constituem o método/modelo de processo. É com base nestas avaliações que resulta a avaliação final do *work product* na primeira coluna.

A terceira coluna permite descrever alguma observação sobre o *work product* que está a ser avaliado.

Tabela 10 – Tabela de cobertura elementar do *work product Glossary* (exemplar 1)

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Glossary **Gl**	HC	HC	LC	HC	HC	HC	LC	

Tabela 11 – Tabela de cobertura elementar do *work product Business Architecture Document* (exemplar 2)

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Business Architecture Document **BAD**	HC	NC	NC	NC	HC	HC	HC	

Através da Tabela 10 pode-se, por exemplo, verificar que o *work product Glossary* possui uma elevada cobertura no *V-Model//* 4SRS, apesar de ser avaliado em dois elementos como pouco coberto, na totalidade este *work product* no *V-Model//* 4SRS tem uma elevada cobertura. O *work product Business Architecture Document* possui também uma elevada cobertura no *V-Model//* 4SRS, tal como apresenta a Tabela 11, apesar deste *work product* ser avaliado em três elementos como não coberto, na sua totalidade tem uma elevada cobertura no *V-Model//* 4SRS.

A avaliação efetuada através destas tabelas de cobertura elementar permite comparar todos os *work products* do RUP com um determinado método/modelo de processo, possibilitando assim, concluir se o método/modelo de processo adotado cobre ou não o que o RUP recomenda nas suas boas práticas de desenvolvimento de *software*.

4.4 Tabelas de cobertura Fase|Disciplina

As tabelas de cobertura fase|disciplina foram elaboradas com o propósito de permitir comparar as atividades e as tarefas do RUP com o método/modelo de processo adotado. É de referir que a avaliação dada às atividades e às tarefas é efetuada com base nas avaliações resultantes das tabelas de cobertura elementar, referidas anteriormente.

Estas tabelas (Tabela 12, Tabela 13 e Tabela 14) foram elaboradas com base nas várias informações obtidas ao longo de todo o estudo e permitem identificar facilmente as diversas interdependências existentes no RUP, assim como, rastrear os vários *work products.*

Estas tabelas de cobertura fase|disciplina são constituídas por nove colunas, sendo que, a coluna ***phase*** e a coluna ***discipline*** apresentam a fase e a disciplina do RUP em estudo. A

coluna ***Role main*** indica o *role* responsável por executar a atividade. A coluna ***Activity*** designa a atividade que se está a avaliar. Estas tabelas apresentam três colunas com o nome **Evaluation** sendo que, a primeira destas apresenta a avaliação final que é dada à atividade, a qual indica se a atividade do RUP é ou não coberta no *V-Model*// 4SRS. Esta avaliação resulta da avaliação das tarefas que a atividade tem associadas e que é efetuada na segunda coluna com o nome *Evaluation* e da avaliação dos *work products* que é realizada na última coluna com o nome *Evaluation*. Sendo que, esta última resulta da avaliação das tabelas de cobertura elementar do *work product* a que está associado. A avaliação, tal como nas tabelas de cobertura elementar, é representada através das três terminologias e respetivas cores, descritas anteriormente. A coluna ***Task*** indica todas as tarefas associadas à atividade. E a coluna ***Work Products*** apresenta todos os *work products* que são consumidos e produzidos nas diversas tarefas. Todos estes *work products* são comparados detalhadamente com o método/modelo de processo adotado nas tabelas de cobertura elementar.

Através destas tabelas pode-se verificar a cobertura das diversas atividades que constituem o RUP, no *V-Model*// 4SRS, assim como, a cobertura das tarefas e dos *work products* associados a essas mesmas atividades. Estas tabelas permitem também verificar as várias interdependências existentes entre as atividades e as suas tarefas e os *work products*. E permitem ainda rastrear os diversos *work products* do RUP indicando quem foi a tarefa que lhe deu origem e quais as outras interligações que possui.

A Tabela 12 refere-se à avaliação da fase *Inception*|disciplina *Business Modeling* do RUP comparativamente com o micro-processo *V-Model* e o método 4SRS. Esta Tabela 12 permite comparar as diferentes atividades pertencentes a esta fase|disciplina do RUP, assim como, as suas tarefas e *work products* com o método/modelo de processo adotado.

Com a Tabela 12 pode-se, por exemplo, verificar que a atividade *Assess Business Status* (ABS) que pertence à disciplina Business Modeling, fase *Inception* e que possui o *Business-Process Analyst* e o *Business Architect* como *role* principal foi avaliada como pouco coberta no *V-Model*// 4SRS. Esta avaliação resultou da avaliação dada às várias tarefas e *work products* que fazem parte da atividade em questão. Note-se que a avaliação dos diversos *work products* é realizada detalhadamente nas tabelas de cobertura elementar e é com base nessa avaliação que resultam todas as outras.

De uma forma geral, através da Tabela 12 é possível verificar que o *V-Model*// 4SRS na fase *Inception*| disciplina *Business Modeling* quando comparado com o RUP possui duas atividades pouco cobertas, a *Assess Business Status* (ABS) e a *Develop Domain Model* (DDM) e uma atividade muito coberta a *Describe Current Business* (DCB).

Tabela 12 – *Inception* | *Business Modeling Matrix for V-Model*/4SRS - *RUP Alignment*

Phase	Discipline	Role main	Activity	Evaluation	Task	Evaluation	Work Products	Evaluation
Inception	Business Modeling	Business-Process Analyst	**ABS** Assess Business Status ••	LC	Assess Target Organization **ATO**	LC	**INPUTS:**	---
							---	
							OUTPUTS:	LC
							Target-Organization Assessment - **TOA**	
					Set and Adjust Objectives **SAO**	LC	**INPUTS:**	LC
							Target-Organization Assessment - **TOA**	
							OUTPUTS:	LC
							Business Vision - **BV**	
					Identify Business Goals **IBG**	LC	**INPUTS:**	LC
							Business Vision - **BV**	
							OUTPUTS:	LC
							Business Goal - **BG**	
					Capture a Common Business Vocabulary **CCBV**	HC	**INPUTS:**	LC
							Business Vision - **BV**	
							OUTPUTS:	HC
							Business Glossary - **BGl**	
					Maintain Business Rules **MBR**	LC	**INPUTS:**	LC
							Business Vision - **BV**	
							OUTPUTS:	LC
							Business Rule - **BR**	
		Business Architect			Business Architectural Analysis **BAA**	HC	**INPUTS:**	LC
							Business Vision - **BV**	
							Business Architectural Proof-of-Concept*- **BAP-C**	HC
							OUTPUTS:	HC
							Business Architecture Document - **BAD**	
							Business Design Model - **BDesM**	HC
							Business Deployment Model - **BDM**	NC
							Business Analysis Model - **BAM**	HC
		Business-Process Analyst	**DDM** Develop Domain Model •	LC	Capture a Common Business Vocabulary **CCBV**	HC	**INPUTS:**	LC
							Business Vision - **BV**	
							OUTPUTS:	HC
							Business Glossary - **BGl**	
					Maintain Business Rules **MBR**	LC	**INPUTS:**	LC
							Business Vision - **BV**	
							OUTPUTS:	LC
							Business Rule - **BR**	
		Business Architect			Business Architectural Analysis **BAA**	HC	**INPUTS:**	LC
							Business Vision - **BV**	
							Business Architectural Proof-of-Concept - **BAP-C**	HC
							OUTPUTS:	HC
							Business Architecture Document - **BAD**	
							Business Design Model - **BDesM**	HC
							Business Deployment Model - **BDM**	NC
							Business Analysis Model - **BAM**	HC
		Business Designer			Detail a Business Entity **DBE**	LC	**INPUTS:**	HC
							Business Entity - **BEn**	
							Business Use-Case Realization - **BUCR**	HC
							OUTPUTS:	NC
							Business Event - **BEv**	
							Business Entity - **BEn**	HC
		Technical Reviewer			Review the Business Analysis Model **RBAM**	LC	**INPUTS:**	HC
							Business Analysis Model - **BAM**	
							OUTPUTS:	LC
							Review Record - **RR**	
		Business-Process Analyst	**DCB** Describe Current Business •	HC	Set and Adjust Objectives **SAO**	LC	**INPUTS:**	LC
							Target-Organization Assessment - **TOA**	
							OUTPUTS:	LC
							Business Vision - **BV**	
					Identify Business Goals **IBG**	LC	**INPUTS:**	LC
							Business Vision - **BV**	
							OUTPUTS:	LC
							Business Goal - **BG**	
					Find Business Actors and Use Cases **FBAUC**	HC	**INPUTS:**	LC
							Business Vision - **BV**	
							OUTPUTS:	HC
							Business Use Case Model - **BUCM**	
							Supplementary Business Specification - **SBS**	HC
					Capture a Common Business Vocabulary **CCBV**	HC	**INPUTS:**	LC
							Business Vision - **BV**	
							OUTPUTS:	HC
							Business Glossary - **BGl**	
					Maintain Business Rules **MBR**	LC	**INPUTS:**	LC
							Business Vision - **BV**	
							OUTPUTS:	LC
							Business Rule - **BR**	
		Business Architect			Business Architectural Analysis **BAA**	HC	**INPUTS:**	LC
							Business Vision - **BV**	
							Business Architectural Proof-of-Concept - **BAP-C**	HC
							OUTPUTS:	HC
							Business Architecture Document - **BAD**	
							Business Design Model - **BDesM**	HC
							Business Deployment Model - **BDM**	NC
							Business Analysis Model - **BAM**	HC

A Tabela 13 refere-se à avaliação da fase *Inception*|disciplina *Requirements* do RUP comparativamente com o micro-processo *V-Model* e método 4SRS. Esta Tabela 13 permite comparar as diferentes atividades pertencentes a esta fase|disciplina do RUP, assim como, as suas tarefas e *work products* com o método/modelo de processo adotado.

Com a Tabela 13 pode-se, por exemplo, verificar que a atividade *Analyze the Problem* (AP) que pertence à disciplina Requirements, fase *Inception* e que possui o *System Analyst* como *role* principal foi avaliada como muito coberta no *V-Model*/4SRS. Esta avaliação resultou da avaliação dada às várias tarefas e *work products* que fazem parte da atividade em questão. Note-se que a avaliação dos diversos *work products* é realizada detalhadamente nas tabelas de cobertura elementar e é com base nessa avaliação que resultam todas as outras.

De uma forma geral, através da Tabela 13 é possível verificar que o *V-Model*/4SRS na fase *Inception*|disciplina *Requirements* quando comparado com o RUP possui duas atividades muito cobertas, a *Analyze the Problem* (AP) e a *Define the System* (DS) e duas atividades pouco cobertas a *Understand Stakeholder Needs* (USN) e a *Manage the Scope of the System* (MSS).

Tabela 13 – *Inception* | *Requirements Matrix for V-Model*/4SRS - *RUP Alignment*

Phase	Discipline	Role main	Activity	Evaluation	Task	Evaluation	Work Products	Evaluation
Inception	Requirements	System Analyst	**AP** Develop Initial Vision -> *Analyze the Problem*	HC	Capture a Common Vocabulary **CCV**	HC	**INPUTS:**	---
							---	
							OUTPUTS:	HC
							Glossary - **Gl**	
					Find Actors and Use Cases **FAUC**	HC	**INPUTS:**	HC
							Stakeholder Requests - **SR**	
							Iteration Plan - **IP**	LC
							OUTPUTS:	HC
							Use-Case Model - **UCM**	
							Requirements Attributes - **ReqA**	LC
					Develop Vision **DV**	LC	**INPUTS:**	HC
							Stakeholder Requests - **SR**	
							Business Case - **BC**	NC
							Iteration Plan - **IP**	LC
							OUTPUTS:	HC
							Vision - **Vi**	
							Requirements Attributes - **ReqA**	LC
					Develop Requirements Management Plan **DRMP**	LC	**INPUTS:**	HC
							Software Development Plan - **SDP**	
							Iteration Plan - **IP**	LC
							OUTPUTS:	LC
							Requirements Management Plan - **RMP**	
			USN Develop Initial Vision -> *Understand Stakeholder Needs*	LC	Capture a Common Vocabulary **CCV**	HC	**INPUTS:**	---
							---	
							OUTPUTS:	HC
							Glossary - **Gl**	
					Elicit Stakeholder Requests **ESR**	LC	**INPUTS:**	NC
							Business Case - **BC**	
							Iteration Plan - **IP**	LC
							OUTPUTS:	HC
							Stakeholder Requests - **SR**	
							Storyboard - **St**	LC
							Requirements Attributes - **ReqA**	LC
					Develop Vision **DV**	LC	**INPUTS:**	HC
							Stakeholder Requests - **SR**	
							Business Case - **BC**	NC
							Iteration Plan - **IP**	LC
							OUTPUTS:	HC
							Vision - **Vi**	
							Requirements Attributes - **ReqA**	LC
					Find Actors and Use Cases **FAUC**	HC	**INPUTS:**	HC
							Stakeholder Requests - **SR**	
							Iteration Plan - **IP**	LC
							OUTPUTS:	HC
							Use-Case Model - **UCM**	
							Requirements Attributes - **ReqA**	LC
					Develop Supplementary Specifications **DSS**	HC	**INPUTS:**	LC
							Iteration Plan - **IP**	
							Stakeholder Requests - **SR**	HC
							OUTPUTS:	HC
							Supplementary Specifications - **SS**	
							Requirements Attributes - **ReqA**	LC
					Manage Dependencies **MDep**	LC	**INPUTS:**	LC
							Requirements Management Plan - **RMP**	
							OUTPUTS:	LC
							Requirements Management Plan - **RMP**	
							Requirements Attributes - **ReqA**	LC
							Vision - **Vi**	HC
			MSS Manage the Scope of the System	LC	Develop Vision **DV**	LC	**INPUTS:**	HC
							Stakeholder Requests - **SR**	
							Business Case - **BC**	NC
							Iteration Plan - **IP**	LC
							OUTPUTS:	HC
							Vision - **Vi**	
							Requirements Attributes - **ReqA**	LC
					Manage Dependencies **MDep**	LC	**INPUTS:**	LC
							Requirements Management Plan - **RMP**	
							OUTPUTS:	LC
							Requirements Management Plan - **RMP**	
							Requirements Attributes - **ReqA**	LC
							Vision - **Vi**	HC
		Software Architect			Prioritize Use Cases **PUC**	HC	**INPUTS:**	LC
							Iteration Plan - **IP**	
							Use-Case Model - **UCM**	HC
							Risk List - **RL**	NC
							Software Architecture Document - **SAD**	HC
							OUTPUTS:	HC
							Software Architecture Document - **SAD**	
							Software Requirement - **SofR**	HC
							Requirements Attributes - **ReqA**	LC
		System Analyst	**DS** Define the System	HC	Develop Vision **DV**	LC	**INPUTS:**	HC
							Stakeholder Requests - **SR**	
							Business Case - **BC**	NC
							Iteration Plan - **IP**	LC
							OUTPUTS:	HC
							Vision - **Vi**	
							Requirements Attributes - **ReqA**	LC
					Capture a Common Vocabulary **CCV**	HC	**INPUTS:**	---

							OUTPUTS:	HC
							Glossary - **Gl**	
					Find Actors and Use Cases **FAUC**	HC	**INPUTS:**	HC
							Stakeholder Requests - **SR**	
							Iteration Plan - **IP**	LC
							OUTPUTS:	HC
							Use-Case Model - **UCM**	
							Requirements Attributes - **ReqA**	LC
					Develop Supplementary Specifications **DSS**	HC	**INPUTS:**	LC
							Iteration Plan - **IP**	
							Stakeholder Requests - **SR**	HC
							OUTPUTS:	HC
							Supplementary Specifications - **SS**	
							Requirements Attributes - **ReqA**	LC
					Manage Dependencies **MDep**	LC	**INPUTS:**	LC
							Requirements Management Plan - **RMP**	
							OUTPUTS:	LC
							Requirements Management Plan - **RMP**	
							Requirements Attributes - **ReqA**	LC
							Vision - **Vi**	HC

A Tabela 14 refere-se à avaliação da fase *Elaboration*|disciplina *Requirements* do RUP comparativamente com o micro-processo *V-Model* e o método 4SRS. Esta Tabela 14 permite comparar as diferentes atividades pertencentes a esta fase|disciplina do RUP, assim como, as suas tarefas e *work products* com o método/modelo de processo adotado.

Com a Tabela 14 pode-se, por exemplo, verificar que a atividade *Manage Changing Requirements* (MCR) que pertence à disciplina *Requirements,* fase *Elaboration* e que possui o *System Analyst* e o *Technical Reviewer* como *role* principal foi avaliada como pouco coberta no *V-Model*/4SRS. Esta avaliação resultou da avaliação dada às várias tarefas e *work products* que fazem parte da atividade em questão. Note-se que a avaliação dos diversos *work products* é realizada detalhadamente nas tabelas de cobertura elementar e é com base nessa avaliação que resultam todas as outras.

De uma forma geral, através da Tabela 14 é possível verificar que o *V-Model*/4SRS na fase *Elaboration*|disciplina *Requirements* quando comparado com o RUP possui uma atividade muito coberta, a *Refine the System Definition* (RSD) uma atividade pouco coberta a *Manage Changing Requirements* (MCR).

Tabela 14 – *Elaboration* | *Requirements Matrix for V-Model*/4SRS - *RUP Alignment*

Phase	Discipline	Role main	Activity	Evaluation	Task	Evaluation	Work Products	Evaluation
Elaboration	Requirements	System Analyst	**MCR** Ongoing Management and Support -> *Manage Changing Requirements*	LC	Structure the Use-Case Model **SUCM**	HC	**INPUTS:**	
							Use-Case Model - **UCM**	HC
							OUTPUTS:	
							Glossary - **Gl**	HC
							Supplementary Specifications - **SS**	HC
							Use-Case Model - **UCM**	HC
							Requirements Attributes - **ReqA**	LC
					Manage Dependencies **MDep**	LC	**INPUTS:**	
							Requirements Management Plan - **RMP**	LC
							OUTPUTS:	
							Requirements Management Plan - **RMP**	LC
							Requirements Attributes - **ReqA**	LC
							Vision - **Vi**	HC
		Technical Reviewer			Review Requirements **RReq**	LC	**INPUTS:**	
							Iteration Plan - **IP**	LC
							Business Case - **BC**	NC
							Software Requirement - **SofR**	HC
							OUTPUTS:	
							Review Record - **RR**	LC
		Requirements Specifier	**RSD** Refine the System Definition	HC	Detail a Use Case **DUC**	HC	**INPUTS:**	
							Use Case - **UC**	HC
							Iteration Plan - **IP**	LC
							OUTPUTS:	
							Use Case - **UC**	HC
							Requirements Attributes - **ReqA**	LC
					Detail the Software Requirements **DSR**	HC	**INPUTS:**	
							Vision - **Vi**	HC
							Iteration Plan - **IP**	LC
							OUTPUTS:	
							Software Requirements Specification - **SRS**	HC
							Software Requirement - **SofR**	HC
							Requirements Attributes - **ReqA**	LC
		System Analyst			Develop Supplementary Specifications **DSS**	HC	**INPUTS:**	
							Iteration Plan - **IP**	LC
							Stakeholder Requests - **SR**	HC
							OUTPUTS:	
							Supplementary Specifications - **SS**	HC
							Requirements Attributes - **ReqA**	LC

Através dos resultados obtidos nas diversas tabelas de cobertura Fase|Disciplina, chega-se à conclusão que o micro-processo *V-Model* e o método 4SRS possuem uma cobertura bastante boa relativamente às recomendações do RUP. Tal como apresenta a Tabela 15, estes possuem uma elevada cobertura em quatro atividades do RUP e uma baixa cobertura em cinco atividades, no entanto todas as atividades analisadas são cobertas.

Tabela 15 – Resumo da cobertura *V-Model* /4SRS relativamente às atividades do RUP

Fase|Disciplina	Actividades do RUP	Cobertura V-Model/4SRS
Inception|Business Modeling	*Assess Business Status* (ABS)	LC
	Develop Domain Model (DDM)	LC
	Describe Current Business (DCB)	HC
Inception|Requirements	*Analyze the Problem* (AP)	HC
	Understand Stakeholder Needs (USN)	LC
	Manage the Scope of the System (MSS)	LC
	Define the System (DS)	HC
Elaboration|Requirements	*Manage Changing Requirements* (MCR)	LC
	Refine the System Definition (RSD)	HC

Esta avaliação tão pormenorizada da cobertura do micro-processo *V-Model* e do método 4SRS relativamente às recomendações do RUP, só foi possível realizar devido à sistematização, efetuada inicialmente, das interdependências existentes no processo RUP. É de referir que, sem essa sistematização não seria possível efetuar esta avaliação tão detalhadamente, uma vez que, não se possuía conhecimento suficiente acerca do processo RUP para a puder realizar.

O facto de se ter um conhecimento prévio sobre como os diferentes elementos se relacionam permite conhecer mais detalhadamente o processo. O que permite obter uma avaliação mais detalhada ao analisar a cobertura de um determinado método/modelo de processo relativamente às recomendações do RUP.

4.5 Conclusões

Através deste capítulo conclui-se que é fundamental possuir conhecimento acerca dos vários elementos que constituem o RUP, assim como, da rastreabilidade e das interdependências existentes. As tabelas elaboradas neste capítulo só foram possíveis de desenvolver devido ao conhecimento prévio de como os vários elementos do RUP se relacionam entre si, daí ser importante e vantajoso estudar as interdependências e a rastreabilidade no RUP.

As tabelas desenvolvidas neste capítulo são bastante úteis quando se compara um determinado método/modelo de processo com o RUP, uma vez que, é através das mesmas que se conhece detalhadamente como RUP é constituído.

Estas tabelas reúnem o conhecimento da rastreabilidade e das interdependências existentes entre os diversos elementos RUP e permitem avaliar pormenorizadamente se o que o RUP recomenda está a ser seguido no método/modelo de processo adotado e como está a ser seguido.

5. Conclusões

5.1 Síntese

O processo RUP é, de acordo com o que foi apresentado ao longo deste documento, um processo que fornece as melhores práticas e orientações para o desenvolvimento de *software.* Contudo, este não disponibiliza nenhum tipo de informação que permita identificar facilmente a rastreabilidade e as interdependências existentes entre os vários elementos que o constituem. Ao longo do desenvolvimento de *software,* isto pode-se tornar um problema, uma vez que, não se possui qualquer informação de como os elementos se relacionam nem quais os elementos que são afetados no caso de uma determinada atividade ou tarefa não ser realizada.

O propósito deste trabalho, como já foi referido anteriormente, consiste na sistematização das interdependências existentes no processo RUP e na sua utilização para estudar um micro-processo (*V-Model)* e um método (4SRS) do ponto de vista da cobertura que garantem relativamente às recomendações do RUP.

Assim sendo, ao longo deste trabalho foram elaboradas diversas tabelas e esquemas com a finalidade de mostrar como os diversos elementos do RUP se relacionam. Estas tabelas e esquemas permitem desde as fases iniciais de desenvolvimento identificar mais facilmente as várias interdependências e a rastreabilidade entre os elementos, assim como, fornecer um conhecimento mais aprofundado acerca da constituição do RUP. Isto é bastante vantajoso, uma vez que, com base no conhecimento fornecido pelas tabelas é possível comparar um determinado método/modelo de processo com o RUP, analisando a sua cobertura nos diversos elementos que constituem o RUP e concluindo se o que o RUP recomenda é coberto e como é coberto pelo método/modelo de processo adotado.

As tabelas e os esquemas elaborados nesta dissertação trazem bastante utilidade, pois, permitem conhecer detalhadamente os vários elementos que constituem o RUP e ainda analisar pormenorizadamente a cobertura de um determinado método/modelo de processo no RUP.

5.2 Discussão

No primeiro capítulo foram definidos os dois objetivos desta dissertação, os quais permitiram alcançar o resultado esperado para este trabalho de investigação.

Assim sendo, a realização do primeiro objetivo foi fundamental na concretização deste trabalho, uma vez que, foi com base nele que o segundo objetivo foi realizado. O primeiro objetivo permitiu conhecer os vários elementos do RUP e perceber como os mesmos se relacionam entre si. As tabelas e os esquemas desenvolvidos neste objetivo apresentam a rastreabilidade e as interdependências existentes entre os vários elementos do RUP. O conhecimento sobre a rastreabilidade e as interdependências entre os vários elementos do RUP é essencial, uma vez que, é através desse conhecimento que se pode avaliar pormenorizadamente a cobertura de um determinado método/modelo de processo no RUP.

O segundo objetivo foi concretizado através da utilização das tabelas elaboradas no primeiro objetivo. Com essas tabelas foi possível analisar a cobertura do micro-processo *V-Model* e do método 4SRS nos diferentes elementos do RUP e verificar se o que o RUP recomenda é seguido pelo método/modelo de processo adotado e como é seguido. A concretização deste objetivo só foi possível devido às tabelas elaboradas para realizar o primeiro objetivo, pois, só através dessas tabelas é que se conhece pormenorizadamente como os elementos que constituem o RUP se relacionam.

Após a concretização dos dois objetivos deste trabalho concluiu-se que é bastante útil e vantajoso conhecer a rastreabilidade e as interdependências existentes entre os diversos elementos que constituem o RUP, pois, quando se compara um determinado método/modelo de processo com o RUP é extremamente importante conhecer detalhadamente como o RUP é constituído. É de referir que, sem a sistematização das interdependências não seria possível efetuar esta avaliação tão detalhadamente do método/modelo de processo adotado, uma vez que, não se possuía conhecimento suficiente acerca do processo RUP para a puder realizar.

Através da sistematização das interdependências existentes no processo RUP fica-se a conhecer pormenorizadamente como os diferentes elementos que constituem o RUP se relacionam. E pode-se ainda utilizar essa sistematização para avaliar minuciosamente se o que o RUP recomenda está a ser seguido no método/modelo de processo adotado e como está a ser seguido.

5.3 Trabalho futuro

No que diz respeito a trabalho futuro, sugere-se que o estudo da rastreabilidade e das interdependências entre os vários elementos do processo RUP seja alargado a todas as disciplinas e fases que compõem este processo. O facto de se expandir o estudo vai permitir conhecer minuciosamente como os vários elementos se relacionam ao longo de todo o processo, o que permitirá avaliar a cobertura de qualquer método/modelo de processo no RUP.

Referências Bibliográficas

Ahn, S., & Chong, K. (2006). A Feature-Oriented Requirements Tracing Method: A Study of Cost-benefit Analysis. *Proceedings of the 2006 International Conference on Hybrid Information Technology (ICHIT'06).* IEEE.

Aizenbud-Reshef , N., Nolan , B. T., Rubin , J., & Shaham-Gafni , Y. (2006). Model traceability. *IBM Systems Journal, 45*(3).

Assawamekin, N., Sunetnanta, T., & Pluempitiwiriyawej, C. (2010). Ontology-based multiperspective requirements traceability framework. Springer.

Ávila, A. L., & Spínola, R. O. (2007). Introdução à Engenharia de Requisitos. *Qualidade de Software*. Engenharia de Software Magazine, DevMedia.

Berndtsson, M., Hansson, J., Olsson, B., & Lundell, B. (2008). *Thesis Projects: A Guide for Students in Computer Science and Information Systems* (2ª ed.). Springer.

Carlshamre, P., Sandahl, K., Lindvall, M., Regnell, B., & Natt och Dag, J. (2001). An Industrial Survey of Requirements Interdependencies in Software Product Release Planning. *Proc. of the Fifth International Symposium on Requirements Engineering* (pp. 84-91). Toronto, Canada: IEEE.

Chitchyan, R., & Rashid, A. (2006). Tracing Requirements Interdependency Semantics. *Workshop on Early Aspects (AOSD 06).* Bonn, Germany.

Cleland-Huang, J., & Chang, C. K. (2003). Event-Based Traceability for Managing Evolutionary Change. *29(9)*, pp. 796-810. IEEE Transactions on Software Engineering.

Dahlstedt, Å. G., & Persson, A. (2003). Requirements Interdependencies - Moulding the State of Research into a Research Agenda. *Proc. of the Ninth International Workshop on Requirements Engineering: Foundation for Software Quality*, (pp. 55-64). Klagenfurt/Velden, Austria.

Dahlstedt, Å. G., & Persson, A. (2005). Requirements Interdependencies– State of the Art and Future Challenges. In Aybüke Aurum, & Claes Wohlin, *Engineering and Managing Software Requirements* (pp. 95-116). Springer.

Dömges, R., & Pohl, K. (1998). Adapting Traceability Environments to Project-Specific Needs. *Communications of the ACM, 41*(12), 54-62.

Ferreira, N., Santos, N., Machado, R. J., & Gašević, D. (2012). Derivation of Process-Oriented Logical Architectures: An Elicitation Approach for Cloud Design. In O. Dieste, A. Jedlitschka, & N. Juristo (Ed.), *PROFES 2012, LNCS 7343* (pp. 44-58). Springer-Verlag Berlin Heidelberg.

Ferreira, N., Santos, N., Machado, R. J., Fernandes, J. E., & Gašević, D. (2011). A V-Model Approach for Business Process Requirements Elicitation in Cloud Design. Springer-Verlag Berlin Heidelberg.

García, H., Santos, E., & Windels, B. (2008). Traceability Management Architectures Supporting Total Traceability in the Context of Software Engineering. *Sixth International Conference on Information Research and Applications – i.Tech 2008*, (pp. 17-23). Bulgaria.

Genvigir, E. C. (2009). *Um modelo para rastreabilidade de requisitos de software baseado em generalização de elos e atributos .* São José dos Campos: Instituto Nacional de Pesquisas Espaciais .

Goknil, A., Kurtev, I., Berg, K. V., & Veldhuis, J. W. (2009). Semantics of trace relations in requirements models for consistency checking and inferencing. *Springer*, 31-54.

Gotel, O. C., & Finkelstein, A. C. (1994). *An Analysis of the Requirements Traceability Problem.* IEEE.

Heindl, M., & Biffl, S. (2005). A Case Study on Value-based Requirements Tracing. *Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering* (pp. 60-69). ACM.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly Vol. 28 No. 1*, pp. 75-105.

Hevner, A., & Chatterjee, S. (2010). *Design Research in Information Systems - Theory and Practice* (Vol. 22). Springer.

Hirsch, M. (2002). Making RUP agile. *OOPSLA '02.* ACM.

IBM. (s.d.). *IBM Rational Unified Process (RUP)*. Obtido em 30 de Junho de 2012, de http://www-01.ibm.com/software/awdtools/rup/

Jadallah, A., Galster, M., Moussavi, M., & Ruhe, G. (2009). Balancing Value and Modifiability when Planning for the Next Release. *Proc. ICSM 2009.* Edmonton, Canada: IEEE.

Kotonya, G., & Sommeiville, I. (1996). Requirements engineering with viewpoints. *Software Engineering Journal*.

Kruchten, P. (2002). Tutorial: Introduction to the Rational Unified Process®. *24th International Conference on Software Engineering (ICSE '02)* (p. 703). USA: ACM.

Kruchten, P. (2004). *The Rational Unified Process: An Introduction* (3º ed.). Addison-Wesley.

Kuloor, C., & Eberlein, A. (2003). Aspect-Oriented Requirements Engineering for Software Product Lines. *Proceedings of the 10 th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS'03).* IEEE.

Lamsweerde, A. V. (2000). Requirements Engineering in the Year 00: A Research Perspective. *Proceedings of the 22nd international conference on Software engineering.* ACM.

Machado, R. J., Fernandes, J. M., Monteiro, P., & Rodrigues, H. (2006). Refinement of Software Architectures by Recursive Model Transformations. J. Münch and M. Vierimaa (Eds.). PROFES 2006. Springer-Verlag.

Manzoni, L. V., & Price, R. T. (2003). Identifying Extensions Required by RUP (Rational Unified Process) to Comply with CMM (Capability Maturity Model) Levels 2 and 3. *IEEE Transactions on Software Engineering , 29*(2), 181-192.

Nogueira , M. (2009). *Engenharia de software: um framework para a gestão de riscos em projeto de software.* Rio de Janeiro: Ciência Moderna.

Nunamaker, J. F., Chen , M., & Purdin , T. D. (1991). Systems development in information systems research. *Journal of Management Information Systems, 7*(3), 89-106.

Paetsch, F., Eberlein, A., & Maurer, F. (2003). Requirements Engineering and Agile Software Development. *Proceedings of the Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'03).* IEEE.

Panis, M. C. (2010). Successful Deployment of Requirements Traceability in a Commercial Engineering Organization...Really. *18th IEEE International Requirements Engineering Conference.* IEEE.

Raja, U. A., & Kamran, K. (2008). *Framework for Requirements Traceability - TLFRT supporting pre-RS & post-RS traceability.* Dissertação de Mestrado, Instituto de Tecnologia de Blekinge, Escola de Engenharia, Suécia.

Ramesh, B., & Edwards, M. (1993). Issues in the Development of a Requirements Traceability Model. *Proceedings of IEEE International Symposium on Requirements Engineering* (pp. 256-259). IEEE.

Ramesh, B., & Jarke, M. (2001). *Toward Reference Models for Requirements Traceability.* IEEE Transactions on Software Engineering.

Ramesh, B., Harrington, G., Rondeau, K., & Edwards, M. (1993). A model of requirements traceability to support systems development. Technical Report NPS-SM-93-017, Naval Surface Warfare Center Dahlgren Division, 10901 New Hampshire Avenue, Silver Spring, Maryland 20903-5000, U.S.A. .

Ramesh, B., Powers, T., Stubbs, C., & Edwards, M. (1993). A study of current practices of requirements traceability in systems development. Technical Report NPS-SM-93-018, Naval Surface Warfare Center Dahlgren Division, 10901 New Hampshire Avenue, Silver Spring, Maryland 20903-5000, U.S.A.

Rational Software. (2001). Rational Unified Process: Best Practices for Software Development Teams.

Rational Unified Process. (s.d.). Rational Method Composer, Version 7.1 , IBM.

Regnell, B., Paech, B., Aurum, A., Wohlin, C., Dutoit, A., & Natt och Dag, J. (2001). Requirements Mean Decisions! – Research issues for understanding and supporting decision-making in Requirements Engineering. *First Swedish Conference on Software Engineering Research and Practice (SERP'01) .* Ronneby, Sweden.

Rochimah, S., Kadir, W. M., & Abdullah, A. H. (2007). An Evaluation of Traceability Approaches to Support Software Evolution. *International Conference on Software Engineering Advances(ICSEA 2007).* IEEE.

Sánchez, P., Alonso, D., Rosique, F., Álvarez, B., & Pastor, J. A. (2011). Introducing Safety Requirements Traceability Support in Model-Driven Development of Robotic Applications. *60(8).* IEEE.

Spanoudakis, G., & Zisman, A. (2005). Software Traceability: A Roadmap. *Vol. III: Recent Advancements*. S K Chang ; World Scientific Publishing.

Vaishnavi, V. K., & Jr., W. K. (2008). *Design Science Research Methods and Patterns - Innovating Information and Communication Technology.* Auerbach.

Wieringa, R. (1995). *An Introduction to Requirements Traceability.* University of Twente Publications .

Winkler, S., & Pilgrim, J. V. (2010). A survey of traceability in requirements engineering and model-driven development. *Software and Systems Modeling, 9(4)*, 529-565. Springer.

ZAVE, P. (1997). Classification of Research Efforts in Requirements Engineering. *29, 4*. ACM Computing Surveys.

Zhang, W., Mei, H., & Zhao, H. (2005). A Feature-Oriented Approach to Modeling Requirements Dependencies. *Proceedings of the 2005 13th IEEE International Conference on Requirements Engineering (RE'05).*

Zou, X., Settimi, R., & Cleland-Huang, J. (2010). Improving automated requirements trace retrieval: a study of term-based enhancement methods. *Empirical Software Engineering, 15(2)*, 119-146. Springer.

Anexos

Anexo I - Tabela de cobertura elementar do *work product Target-Organization Assessment*

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Target-Organization Assessment **TOA**	**LC**	**LC**	**NC**	**NC**	**NC**	**NC**	**NC**	

HC	High Coverage	LC	Low Coverage	NC	Not Covered

Anexo II - Tabela de cobertura elementar do *work product Business Vision*

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Business Vision **BV**	**LC**	**LC**	**NC**	**NC**	**NC**	**NC**	**NC**	

HC	High Coverage	LC	Low Coverage	NC	Not Covered

Anexo III - Tabela de cobertura elementar do *work product Business Goal*

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Business Goal **BG**	LC	LC	LC	LC	LC	NC	NC	

HC	High Coverage	LC	Low Coverage	NC	Not Covered

Anexo IV - Tabela de cobertura elementar do *work product Business Glossary*

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Business Glossary **BGl**	HC	HC	HC	HC	HC	HC	HC	

HC	High Coverage	LC	Low Coverage	NC	Not Covered

Anexo V - Tabela de cobertura elementar do *work product Business Rule*

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Business Rule **BR**	**LC**	**LC**	**LC**	**LC**	**LC**	**LC**	**LC**	

HC	High Coverage	LC	Low Coverage	NC	Not Covered

Anexo VI - Tabela de cobertura elementar do *work product Business Architectural Proof-of-Concept*

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Business Architectural Proof-of-Concept **BAP-C**	HC	NC	NC	NC	HC	HC	HC	

HC	High Coverage	LC	Low Coverage	NC	Not Covered

Anexo VII - Tabela de cobertura elementar do *work product Business Architecture Document*

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Business Architecture Document **BAD**	HC	NC	NC	NC	HC	HC	HC	

HC	High Coverage	LC	Low Coverage	NC	Not Covered

Anexo VIII - Tabela de cobertura elementar do *work product Business Design Model*

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Business Design Model **BDesM**	HC	NC	HC	HC	NC	NC	NC	

HC	High Coverage	LC	Low Coverage	NC	Not Covered

Anexo IX - Tabela de cobertura elementar do *work product Business Deployment Model*

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Business Deployment Model **BDM**	NC	NC	NC	NC	NC	NC	NC	

HC	High Coverage	LC	Low Coverage	NC	Not Covered

Anexo X - Tabela de cobertura elementar do *work product Business Analysis Model*

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Business Analysis Model **BAM**	HC	NC	HC	HC	NC	NC	NC	

HC	High Coverage	LC	Low Coverage	NC	Not Covered

Anexo XI - Tabela de cobertura elementar do *work product Business Entity*

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Business Entity **BEn**	HC	NC	HC	HC	NC	NC	HC	

HC	High Coverage	LC	Low Coverage	NC	Not Covered

Anexo XII - Tabela de cobertura elementar do *work product Business Use-Case Realization*

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Business Use-Case Realization **BUCR**	HC	NC	HC	HC	NC	NC	NC	

HC	High Coverage	LC	Low Coverage	NC	Not Covered

Anexo XIII - Tabela de cobertura elementar do *work product Business Event*

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Business Event **BEv**	NC	NC	NC	NC	NC	NC	NC	

HC	High Coverage	LC	Low Coverage	NC	Not Covered

Anexo XIV - Tabela de cobertura elementar do *work product Review Record*

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Review Record **RR**	LC	NC	NC	NC	NC	LC	LC	

HC	High Coverage	LC	Low Coverage	NC	Not Covered

Anexo XV - Tabela de cobertura elementar do *work product Business Use Case Model*

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Business Use Case Model **BUCM**	HC	NC	HC	HC	NC	NC	NC	

HC	High Coverage	LC	Low Coverage	NC	Not Covered

Anexo XVI - Tabela de cobertura elementar do *work product Supplementary Business Specification*

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Supplementary Business Specification **SBS**	HC	NC	NC	NC	HC	NC	NC	

HC	High Coverage	LC	Low Coverage	NC	Not Covered

Anexo XVII - Tabela de cobertura elementar do *work product Glossary*

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Glossary **Gl**	HC	HC	LC	HC	HC	HC	LC	

HC	High Coverage	LC	Low Coverage	NC	Not Covered

Anexo XVIII - Tabela de cobertura elementar do *work product Stakeholder Requests*

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Stakeholder Requests **SR**	HC	NC	HC	HC	HC	NC	NC	

HC	High Coverage	LC	Low Coverage	NC	Not Covered

Anexo XIX - Tabela de cobertura elementar do *work product Iteration Plan*

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Iteration Plan **IP**	LC	NC	NC	NC	LC	NC	NC	

HC	High Coverage	LC	Low Coverage	NC	Not Covered

Anexo XX - Tabela de cobertura elementar do *work product Use-Case Model*

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Use-Case Model **UCM**	HC	NC	HC	HC	HC	NC	NC	

HC	High Coverage	LC	Low Coverage	NC	Not Covered

Anexo XXI - Tabela de cobertura elementar do *work product Requirements Attributes*

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Requirements Attributes **ReqA**	LC	NC	NC	NC	LC	NC	NC	

HC	High Coverage	LC	Low Coverage	NC	Not Covered

Anexo XXII - Tabela de cobertura elementar do *work product Business Case*

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Business Case **BC**	NC	NC	NC	NC	NC	NC	NC	

HC	High Coverage	LC	Low Coverage	NC	Not Covered

Anexo XXIII - Tabela de cobertura elementar do *work product Vision*

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Vision **Vi**	HC	NC	NC	NC	HC	NC	NC	

HC	High Coverage	LC	Low Coverage	NC	Not Covered

Anexo XXIV - Tabela de cobertura elementar do *work product Software Development Plan*

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Software Development Plan **SDP**	HC	NC	NC	NC	HC	HC	NC	

HC	High Coverage	LC	Low Coverage	NC	Not Covered

Anexo XXV - Tabela de cobertura elementar do *work product Requirements Management Plan*

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Requirements Management Plan **RMP**	LC	NC	NC	NC	LC	NC	NC	

HC	High Coverage	LC	Low Coverage	NC	Not Covered

Anexo XXVI - Tabela de cobertura elementar do *work product Storyboard*

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Storyboard **St**	LC	NC	NC	NC	LC	LC	NC	

HC	High Coverage	LC	Low Coverage	NC	Not Covered

Anexo XXVII - Tabela de cobertura elementar do *work product Supplementary Specifications*

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Supplementary Specifications **SS**	HC	NC	NC	NC	HC	HC	HC	

HC	High Coverage	LC	Low Coverage	NC	Not Covered

Anexo XXVIII - Tabela de cobertura elementar do *work product Risk List*

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Risk List **RL**	NC	NC	NC	NC	NC	NC	NC	

HC	High Coverage	LC	Low Coverage	NC	Not Covered

Anexo XXIX - Tabela de cobertura elementar do *work product Software Architecture Document*

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Software Architecture Document **SAD**	HC	NC	NC	NC	HC	HC	NC	

HC	High Coverage	LC	Low Coverage	NC	Not Covered

Anexo XXX - Tabela de cobertura elementar do *work product* Software Requirement

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Software Requirement **SofR**	HC	NC	NC	NC	HC	NC	NC	

HC	High Coverage	LC	Low Coverage	NC	Not Covered

Anexo XXXI - Tabela de cobertura elementar do *work product Use Case*

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Use Case **UC**	HC	NC	HC	HC	LC	NC	NC	

HC	High Coverage	LC	Low Coverage	NC	Not Covered

Anexo XXXII - Tabela de cobertura elementar do *work product Software Requirements Specification*

Elementary Matrix for V-Model - RUP Alignment

RUP		V-Model process at process-level						Observations
		Problem Domain				*Solution Domain*		
Work Product	**Evaluation**	**Organizational Configurations (OCs)**	**A-type Sequence Diagrams**	**Use Cases Diagrams**	**4SRS**	**Logical Architecture Diagrams**	**B-type Sequence Diagrams**	
Software Requirements Specification **SRS**	HC	NC	NC	NC	HC	HC	HC	

HC	High Coverage	LC	Low Coverage	NC	Not Covered