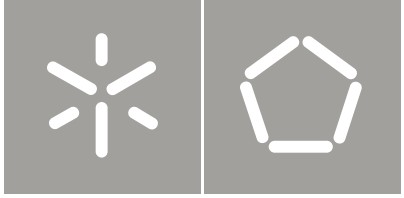
Universidade do Minho
Escola de Engenharia

Jorge Delfim Ferreira Mendes

Computação em Nuvem em Sistemas Logísticos de Transportes: Análise de Um Caso Prático

Computação em Nuvem em Sistemas Logísticos de Transportes: Análise de Um Caso Prático
Jorge Delfim Ferreira Mendes
UMinho | 2013

outubro de 2013



Universidade do Minho
Escola de Engenharia

Jorge Delfim Ferreira Mendes

Computação em Nuvem em Sistemas Logísticos de Transportes: Análise de Um Caso Prático

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

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

outubro de 2013

DECLARAÇÃO

Nome: Jorge Delfim Ferreira Mendes

Endereço eletrónico: jorgedfmendes@gmail.com

Telefone: 937516584

Número do Bilhete de Identidade: 13381280

Título dissertação/tese

Computação em Nuvem em Sistemas Logísticos de Transportes: Análise de Um Caso Prático

Orientador: Professor 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 foi apenas possível com a contribuição e apoio de muitas pessoas.

Agradeço ao Professor Ricardo J. Machado por ter desempenhado as suas funções de orientador na perfeição durante a realização deste trabalho.

Agradeço ao Carlos Salgado, aluno de doutoramento, por ter acompanhado e validado os resultados alcançados durante a aplicação do processo V+V.

Agradeço ao António Pereira, mestre de comunicações, por ter partilhado o seu trabalho e a sua experiência na aplicação de modelos de referência de computação em nuvem em arquiteturas lógicas de sistemas de software.

Por fim, mas não menos importante, agradeço a toda a minha família e amigos por todo o apoio que me deram, em especial aos meus pais, António Mendes e Rosa Mendes, aos meus irmãos, Luís Mendes e Mário Mendes e aos meus amigos, Pedro Fidalgo e José Gonçalves.

RESUMO

A computação em nuvem está a provocar uma mudança na forma como os recursos e os serviços de Tecnologias de Informação (TI) são consumidos e oferecidos. Por um lado, está a criar novas alternativas para reduzir custos e acrescentar valor ao negócio do consumidor. Por outro, está a pressionar o fornecedor a adotar novos modelos de negócio de desenvolvimento de software e novas alternativas de modelos de entrega de software.

Deste modo, com o propósito de auxiliar o fornecedor a adaptar-se a esta nova realidade, este trabalho apresenta uma sistematização de abordagens para o Desenvolvimento de Sistemas de Informação (DSI) baseados no modelo de computação em nuvem. Para além disso, de modo a demonstrar e exemplificar a utilização de um conjunto de abordagens deste tipo, este trabalho apresenta um caso prático para o domínio dos Sistemas Logísticos de Transportes (SLT).

O resultado final parece apenas beneficiar os fornecedores, que oferecem ou pretendem oferecer soluções entregues como um serviço para o domínio dos SLT. No entanto, espera-se que a base de conhecimento criada e as decisões estratégicas efetuadas durante a realização deste trabalho ajudem também todos os outros fornecedores, que oferecem ou pretendem oferecer soluções entregues como um serviço para qualquer domínio, a colmatar os problemas associados à seleção, adaptação e utilização de abordagens para o DSI baseados no modelo de computação em nuvem.

Palavras-chave: Computação em Nuvem, Recursos e Serviços de Tecnologia de Informação, Arquiteturas de Sistemas de Informação, Fornecedores de Software e Sistemas Logísticos de Transportes.

ABSTRACT

Cloud Computing in Logistic Systems of Transports: A Practical Case Analysis

Cloud computing is changing the way resources and services of Information Technology are consumed and delivered. On one hand, is creating new alternatives to reduce costs and to add value to the business of consumer. On the other hand, cloud computing is also pushing suppliers to adopt new business models for software development and software delivery.

Therefore, in order to help supplier to adapt into this new reality, this work presents a systematization of approaches for develop cloud-based information systems. In additionally, with the purpose to demonstrate and exemplify the use of a set of approaches for this type, this paper presents a practical case in the field of Transports Logistic Systems.

The result seems to benefit only the suppliers that offer or intend to offer solutions *as a service* for the field of Transports Logistic Systems. However, it is expected that the knowledge created and the strategic decisions made during this work also help all other suppliers, which offer or intend to offer solutions *as a service* in any domain, to address the problems associated with the selection, adaptation and use of approaches for the development of cloud-based information systems.

keywords: Cloud Computing, Information Technology Resources and Services, Information Systems Architectures, Software Vendors and System Logistics of Transports.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

SIGLAS

Neste documento são apresentados e utilizados acrónimos localmente no âmbito de cada um dos capítulos. As siglas, enquanto abreviaturas de designações comuns, apresentadas na sua primeira utilização e empregues ao longo de todo o documento são:

- TI – Tecnologias de Informação
- SI – Sistemas de Informação
- DSI – Desenvolvimento de Sistemas de Informação
- CSI – Construção de Sistemas Informáticos
- SLT – Sistema Logístico de Transportes

CAPÍTULO 1 – INTRODUÇÃO

1.1 Contextualização

As inovações tecnológicas têm moldado a indústria das Tecnologias de Informação (TI), desde o seu início (Sahoo 2009). Atualmente, uma das inovações que promete revolucionar o negócio da indústria das TI é o conceito de computação em nuvem (Armbrust et al. 2010).

A computação em nuvem promete, principalmente, a entrega de serviços de TI com as mesmas ou até mesmo novas funcionalidades de serviços de TI e, ao mesmo tempo, reduzir os custos iniciais que impedem muitas organizações de implementarem muitos serviços de TI avançados (Marston et al. 2011). No entanto, com o modelo de computação em nuvem, o software passa a ser operado pelo fornecedor de software e entregue ao cliente como um serviço (Stuckenberg, Fielt e Loser 2011).

Deste modo, e se a computação em nuvem está a alcançar o seu potencial, é preciso compreender claramente as diferentes questões envolvidas, tanto do ponto de vista do consumidor como do ponto de vista do fornecedor (Marston et al. 2011). Por um lado, com as soluções de computação em nuvem, o consumidor, de repente, começou a ter novas alternativas para reduzir custos e acrescentar valor ao seu negócio, através de uma melhor utilização dos recursos e dos serviços de TI (Böhm et al. 2011). Por outro lado, como a computação em nuvem começa a ser vista, cada vez mais, como um substituto viável para a infraestrutura de TI de uma organização, o fornecedor acaba por ser pressionado a adotar novos modelos de negócio de desenvolvimento de software e novas alternativas de modelos de entrega de software (Francis 2009).

Para o caso do fornecedor, os desafios não estão só relacionados com a tecnologia mas também com o negócio (Böhm et al. 2011). Por isso, antes de iniciar a mudança para o novo modelo de computação, o fornecedor deve avaliar todas as questões envolvidas relativas à adoção do modelo de computação em nuvem. Identificadas e analisadas estas questões, se o fornecedor optar pela adoção do novo modelo de computação, este deverá definir uma estratégia adequada à sua realidade de forma a evitar problemas onerosos.

É nesta perspetiva que se justifica a realização do presente trabalho de investigação. Este deve ser considerado como um contributo para a atividade de definição estratégica do fornecedor de

recursos e serviços de TI, na medida em que permite identificar um conjunto de abordagens que podem ser utilizadas para a adoção do modelo de computação em nuvem, facilita a escolha da melhor estratégia, conduzindo, consequentemente, à obtenção de melhores resultados.

Além disso, este trabalho apresenta um caso prático, no contexto dos Sistemas Logísticos de Transportes (SLT), que apresenta a aplicação de um conjunto de abordagens para o Desenvolvimento de Sistemas de Informação (DSI) baseados no modelo de computação em nuvem.

O resultado final parece apenas beneficiar os fornecedores (i.e. empresas fornecedoras de software e departamentos de TI), que oferecem ou pretendem oferecer soluções entregues como um serviço para o domínio dos SLT. No entanto, espera-se que a base de conhecimento criada e as decisões estratégicas efetuadas durante a realização deste trabalho ajudem também todos os outros fornecedores, que oferecem ou pretendem oferecer soluções entregues como um serviço para qualquer domínio, a colmatar os problemas associados à seleção, adaptação e utilização de abordagens para o DSI baseados no modelo de computação em nuvem.

1.2 Objetivos

O mercado de serviços em nuvem é, claramente, um sector de alto crescimento dentro do mercado global de TI. Contudo, e antes de se assumirem ações de mudança, os fornecedores de serviços e recursos de TI devem compreender claramente as várias questões envolvidas, tanto ao nível de negócio como de tecnologia.

Assim, e com um maior ênfase sobre as questões relacionados com a tecnologia, este problema deu origem ao principal objetivo da realização desta dissertação, a sistematização de abordagens para o DSI baseados em computação em nuvem. Para além disso, e uma vez que já detinha, antes da realização do presente trabalho, experiência pessoal no desenvolvimento de soluções para o domínio dos SLT, decidiu-se que outro objetivo passaria pela aplicação dos conhecimentos adquiridos a um caso prático.

Deste modo, formulou-se a seguinte questão de investigação: “Como conceber, na perspetiva do fornecedor de serviços e recursos de TI, uma solução baseada em computação em nuvem para o suporte a Sistemas Logísticos de Transportes?”.

Com vista a responder a esta questão de investigação, foi definido um conjunto de três objetivos a cumprir. Estes objetivos encontram-se reunidos na Tabela 1 e serão alvo de uma descrição mais detalhada nos parágrafos subsequentes.

Tabela 1 – Objetivos do trabalho de investigação.

Identificador do objetivo	Descrição do objetivo
O1	Rever fundamentos e literatura.
O2	Identificar, analisar e selecionar abordagens existentes para a conceção de um sistema de software baseado no modelo de computação em nuvem para o suporte a SLT.
O3	Exemplificar a utilização de abordagens para a conceção de um sistema de software alinhado com as necessidades de computação em nuvem e dos SLT.

O primeiro objetivo (O1 - rever fundamentos e literatura) envolveu a revisão e análise de um conjunto de documentos bibliográficos relacionados com o conceito de computação em nuvem. Esta análise foi realizada com o intuito de construir uma base teórica de conhecimento capaz de suportar e apoiar todo o trabalho de investigação.

O segundo objetivo (O2 - identificar, analisar e selecionar abordagens existentes para a conceção de um sistema de software baseado no modelo de computação em nuvem para o suporte a SLT) constituiu a primeira contribuição concreta para o cumprimento da finalidade deste trabalho. A realização deste objetivo envolveu três atividades principais: a identificação das diferentes abordagens ou dos diferentes métodos existentes para a conceção de sistemas de software baseados no modelo de computação em nuvem; perante as abordagens identificadas, analisar as mesmas para averiguar as vantagens e as desvantagens da utilização de cada uma delas; e a seleção das abordagens mais adequadas para o caso prático dos SLT.

O terceiro e último objetivo (O3 - exemplificar a utilização de abordagens para a conceção de um sistema de software alinhado com as necessidades de computação em nuvem e dos SLT) envolveu a exemplificação da utilização de abordagens para a conceção de um sistema de software baseado no modelo de computação em nuvem para o suporte a SLT. Este objetivo permitiu validar a utilidade dos métodos selecionados no segundo objetivo e alcançar uma solução real e concreta, que se acredita de valor para as empresas com processos de SLT e/ou para empresas fornecedoras de software, que oferecem ou pretendem oferecer soluções para o domínio dos SLT. Este objetivo foi realizado com base no conhecimento produzido nos objetivos

anteriores e com base na perspetiva do autor relativamente às características, particularidades e conhecimento pessoal sobre o domínio dos SLT e sobre a atividade de DSI. O resultado deste objetivo seria a exemplificação de todo o processo de construção de um sistema de software, no entanto, devido à complexidade da atividade de DSI, foi apenas possível criar uma base ou contexto para compreensão do problema e para conceção de um sistema de software de nível de produto.

1.3 Abordagem Metodológica

Consciente de que a realização de um trabalho de investigação exige um cuidado especial na sua condução, de modo a que os objetivos inicialmente propostos sejam alcançados dentro do tempo estipulado, antes de iniciar a investigação, foi realizada uma reflexão sobre a forma como todo o estudo deveria decorrer.

Esta reflexão iniciou com um estudo sobre o conceito e a importância das metodologias de investigação. Estudo que, para além de fomentar a consciencialização pessoal sobre a importância das metodologias de investigação, também permitiu verificar a existência de várias metodologias de investigação. De seguida, foi realizada uma análise às várias metodologias identificadas, terminando com a seleção da metodologia que se considerou mais adequada à natureza do problema que foi definido.

Como resultado desta reflexão, foi deliberado um conjunto de três atividades sequenciais, baseadas na metodologia Design Science Research (DSR), que formalizou a estratégia adotada para a realização dos objetivos deste trabalho de dissertação. A Figura 1 sintetiza estas atividades, assim como as técnicas utilizadas e os resultados esperados para cada uma delas, e ajuda a compreender a forma real como decorreu o processo de investigação, desde o estabelecimento da temática em estudo até à entrega da dissertação. Os fluxos de conhecimento representam o novo conhecimento adquirido a partir do ato específico de conceção e sempre que foi necessário voltar à primeira atividade, novas restrições de compreensão de teorias foram aplicadas.

Apresenta-se de seguida uma descrição sucinta de cada uma das atividades identificadas.

Na primeira atividade (definir o problema e profetizar uma solução) a ênfase foi colocada na consciencialização do problema e na sugestão de uma solução. O principal objetivo desta

atividade foi responder a um conjunto de questões genéricas, como por exemplo, qual o tema a abordar, qual o problema a resolver, como proceder para resolver o problema, quais os objetivos a atingir e quais os resultados esperados. Para realizar este objetivo, foi definido que uma revisão de literatura sobre a área do problema deveria ser efetuada com base nas sugestões de Webster e Watson (2002). O resultado desta atividade foi um(a) projeto de dissertação/pré-dissertação, que foi alvo de avaliação por um professor doutorado da Universidade do Minho antes de iniciar as atividades seguintes.

A realização desta atividade revelou-se importante por vários motivos: permitiu clarificar o que se pretendia alcançar com o estudo; permitiu delinear a estratégia a seguir no decorrer de toda a investigação; permitiu estabelecer um ponto de partida estável para a realização da dissertação e para a escrita do relatório de dissertação; e permitiu elaborar uma estratégia de gestão de riscos que contribuiu para o sucesso deste trabalho. Para além desses motivos, esta atividade também contribuiu consideravelmente para a realização do primeiro objetivo de investigação (O1), definido na secção anterior.

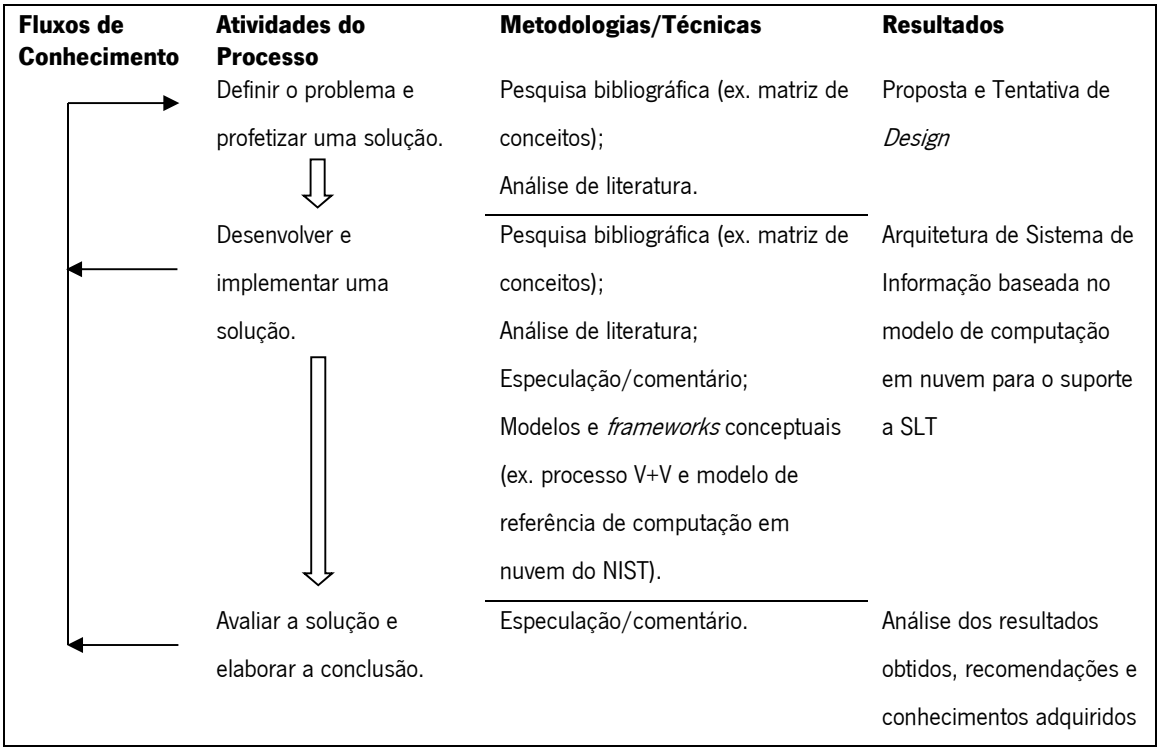


Figura 1 - Metodologia utilizada no âmbito deste trabalho de dissertação (adaptado de Vaishnavi e Kuechler (2004)).

Na segunda atividade (desenvolver e implementar uma solução) o foco foi para a concretização da proposta definida na primeira atividade. Esta proposta determinava a realização dos objetivos apresentados na secção anterior (O1, O2 e o O3). Para a realização do primeiro (O1) foi definida a realização de uma revisão de literatura, sobre o conceito de computação em nuvem, com base

nas sugestões de Webster e Watson (2002). Para o segundo objetivo (O2), inicialmente, foi apenas definido seguir as sugestões de Carvalho (1996) relativamente à atividade de DSI, todavia, com a realização do mesmo, foi também definido realizar uma outra revisão de literatura sobre o problema dos SLT e as abordagens de engenharia de requisitos. Relativamente ao último objetivo (O3), após a conclusão do segundo objetivo (O2), foi definido que deveriam ser utilizadas como ferramentas auxiliares, o modelo V+V de Ferreira, Santos, Machado, et al. (2013) e o modelo de referência de computação em nuvem do NIST (Liu et al. 2011), para a conceção de um sistema de software baseado no modelo de computação em nuvem e alinhado com as necessidades dos SLT. O resultado esperado passava pela conceção de uma arquitetura de SI de nível de produto baseada no modelo de computação em nuvem para o suporte a SLT. Contudo, e pelo facto do problema se revelar mais complexo do que o inicialmente esperado, foi apenas possível especificar o contexto (i.e. uma arquitetura de SI de nível de processo e uma arquitetura de computação em nuvem SLT) que pode servir de *input* para a conceção de uma arquitetura de nível de produto.

Por fim, a terceira atividade - avaliar a solução e elaborar a conclusão - envolveu a realização de uma análise crítica da solução alcançada e de todo o trabalho efetuado, combinando os resultados esperados com os resultados obtidos, daí resultando um conjunto de recomendações.

Como resultado final de todas estas atividades, foi desenvolvido o presente documento de dissertação.

1.4 Organização do Documento

Este documento constitui o produto final de todo o trabalho de investigação realizado e é fruto de uma postura de análise, exposição e originalidade com o intuito de conseguir expor verdadeiramente o assunto em estudo. De seguida descreve-se a organização deste documento e sintetizam-se os cinco capítulos que o constituem.

O capítulo 1 inicia com a contextualização do tema de 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.

O capítulo 2 está estruturado em três secções, onde se tenta, ao longo das sucessivas secções, restringir o assunto na direção do problema inicialmente definido. A primeira secção apresenta três grandes áreas do conhecimento, Computação, Gestão Organizacional e Gestão de Sistemas de Informação, que ajudam a definir a área de estudo desta investigação. A segunda secção apresenta, desde a origem até o estado atual, o principal objeto em estudo, a computação em nuvem. Por fim, a última secção apresenta e discute como a computação em nuvem pode ser útil para a sociedade, quer na perspetiva do fornecedor, quer na perspetiva do utilizador.

O capítulo 3 está organizado em cinco secção, onde se apresenta o problema dos SLT e as abordagens de DSI que podem ajudar a resolver o mesmo. A primeira e a última secção dizem respeito à introdução e às conclusões, respetivamente. A segunda secção apresenta um estudo sobre o estado atual e as necessidades das organizações com processos de SLT, permitindo ao leitor identificar a temática específica deste trabalho de investigação. De seguida, na terceira secção, são identificados e analisadas as principais abordagens de engenharia de requisitos para a conceção de SI. Por fim, a quarta secção descreve o modelo de referência de computação em nuvem do NIST que será utilizado no caso prático para identificar requisitos de computação em nuvem.

O capítulo 4 também está estruturado em cinco secções e apresenta um caso prático de como conceber um sistema de software baseado no modelo de computação em nuvem e alinhado com os SLT. A primeira e a última secção dizem respeito à introdução e às conclusões, respetivamente. A segunda secção exemplifica a utilização do processo V+V durante as fases de modelação das necessidades de nível de processo do domínio do problema dos SLT. A terceira secção, exemplifica a utilização das restantes fases do processo V+V até à criação de contexto para a conceção de um sistema de software de nível de produto. A terceira secção exemplifica a utilização do modelo de referência de computação em nuvem do NIST para a identificação de requisitos de computação em nuvem.

O capítulo 5, e último, sintetiza as conclusões mais relevantes deste trabalho, justificando a sua importância, a aprendizagem daí resultante, e as dificuldades principais e problemas enfrentados durante a sua realização. Antes de terminar são ainda formuladas algumas propostas de trabalho futuro.

A Tabela 2 relaciona cada um dos capítulos da dissertação com os objetivos propostos para este trabalho de investigação.

Para além dos cinco capítulos que constituem o corpo deste documento, são ainda incluídos dois anexos e a lista de referências bibliográficas. O primeiro anexo apresenta a matriz de conceitos resultante das sugestões de Webster e Watson (2002) para a revisão de literatura. O segundo anexo descreve os resultados alcançados durante a execução do processo V+V para o caso prático dos SLT.

Tabela 2 – Matriz capítulos *versus* objetivos.

	01	**02**	**03**
Capítulo 1	X		
Capítulo 2	X		
Capítulo 3		X	
Capítulo 4			X
Capítulo 5	X	X	X

CAPÍTULO 2 – COMPUTAÇÃO EM NUVEM

2.1 Introdução

Com o intuito de apresentar e clarificar alguns conceitos essenciais no âmbito da investigação a desenvolver, foi realizada uma revisão de literatura inicial cujo resultado é aqui descrito. Este resultado possibilitou a criação de uma base sólida para permitir o progresso do conhecimento, a realização do primeiro objetivo (O1) e servir de ponto de partida para a realização dos restantes objetivos (O2 e O3).

De forma a estruturar a informação recolhida durante a revisão de literatura, as sugestões de Webster e Watson (2002) foram seguidas e a matriz de conceitos resultante, destas sugestões, pode ser consultada em "Anexo A – Matriz de Conceitos".

Na secção 2.2 são apresentados os conceitos que se creem basilares para o trabalho e que clarificam a área em estudo. Na secção 2.3, analisa-se o conceito de computação em nuvem. Na secção 2.4, destaca-se a relevância da computação em nuvem para os fornecedores e para os utilizadores de serviços e recursos de TI. Por fim, na secção 2.5, são apresentadas as ideias principais mencionadas nas secções anteriores.

2.2 Conceitos Basilares

Esta secção apresentada os conceitos fundamentais deste contributo de investigação e atenta-se sobre os conceitos de Computação, de Gestão Organizacional e de Gestão de Sistemas de Informação.

2.2.1 Computação

Seguindo Philipson (2003), é possível constatar que a história da computação é bastante longa e que começou muito antes do hardware de computação e da tecnologia moderna de computação. Mas, antes de iniciar uma perspetiva histórica sobre a computação, crê-se relevante apresentar as definições do conceito de computação, de tecnologia e de plataforma de computação.

A palavra computação pode ser definida como o "ato ou efeito de computar" (Infópedia 2012), em que "computar" é "avaliar por meio de cálculo" (Infópedia 2012). A sua primeira utilização conhecida ocorreu no século XV (Dictionary.com 2012). Por sua vez, o termo tecnologia tem sido definido como as "implementações práticas da inteligência" (March e Smith 1995, p. 252). Por fim, o nome de plataforma de computação pode ser atribuído quando várias tecnologias formam uma infraestrutura para a computação, independentemente do seu propósito (Grossman 2012).

Acredita-se, então, que desde que o Homem começou a contar, dispositivos mecânicos (tecnologia) foram utilizados para o ajudar a somar e multiplicar (computar). O *antikythera, counting boards*, e *abacus* são exemplos de uns dos primeiros dispositivos de cálculo mais conhecidos.

Contudo, a computação moderna teve início apenas no século XVII com a invenção do *Napier's Bones* e do logaritmo de John Napier. A partir daqui e até aos anos 50, uma série de inovações pulularam, como por exemplo:

- Com a influência do logaritmo de Napier, foram desenvolvidas as primeiras calculadoras mecânicas, como a *calculating clock* de Wilhelm Schickard (1623) e a *pascaline* de Blaise Pascal (1642).
- Com a influência do *pascaline*, Gottfried Wilhelm von Leibniz criou o *stepped reckoner*, dispositivo que era de tal forma sofisticado que a capacidade da tecnologia existente era insuficiente para se poder utilizar com frequência.
- Como as extensões de melhoria aos dispositivos de Pascal e Leibniz eram apenas calculadoras e não podiam ser programadas, no século XIX, Charles Babbage inventou a máquina diferencial e a máquina analítica. Esta última é hoje considerada o primeiro computador de todos os tempos e a Augusta Ada Byron, que conheceu Charles Babbage em 1833, é hoje considerada a primeira programadora de computadores da história.
- Mais tarde, George Boole (1815-1864) inventou a álgebra booleana que é a base para a forma de como os computadores modernos processam as instruções e os dados.
- Em 1880, com a elaboração dos censos dos Estados Unidos, Herman Hollerith concebeu um sistema (com *punched cards*) capaz de calcular os resultados muito mais rapidamente do que os sistemas existentes. A empresa de Hollerith iria mais tarde dar origem à tão conhecida International Business Machines (IBM).

- Durante a segunda guerra mundial, Alan Turing desenvolveu os símbolos lógicos e a "máquina de Turing", que descreve como os computadores modernos leem, processam e gravam os dados.
- Em 1942, John Mauchly começou a projetar o primeiro computador eletrónico, o Electronic Numerator, Integrator, Analyzer and Computer (ENIAC). Fascinado pelo sucesso do ENIAC, John Von Neumann teorizou sobre o que um computador eletrónico era e como devia funcionar. E, por volta dos anos 50, começaram a ser concebidos, cada vez mais, computadores com propósitos de negócio, como o Universal Automatic Computer (UNIVAC).

A partir dos anos 50, o desenvolvimento da computação continuou a progredir e nunca mais parou. Surgiram os microprocessadores, os computadores pessoais, a internet e atualmente a era da informação continua a influenciar a forma como se computa.

Assim, e analisando o passado, é possível verificar que ainda antes de existirem os computadores modernos, em 1930, Alan Turing escreveu precisamente os limites entre o que uma "máquina de computação" pode fazer e o que não pode fazer (Hopcroft, Motwani e Ullman 1979). Isto significa que, ainda hoje, uma tecnologia de computação, como o computador ou o *personal digital assistant* (PDA), não consegue resolver todos os problemas de cálculo.

Ainda assim, esta tecnologia tem um papel importante na vida das pessoas (Martin 2003). A tecnologia de computação pode servir, não só para a computação, mas também para melhorar a eficiência de processos de negócio. Ou seja, podem existir diferentes visões de computação e diferentes propósitos de utilização de computação (Grossman 2012).

Shackelford et al. (2006) identificaram seis formas diferentes de observar a computação com um propósito diferente:

- Ciências da Computação. Os profissionais desta área preocupam-se principalmente com a conceção e implementação de software, a conceção de novas formas de se utilizar computadores e o desenvolvimento de formas efetivas para resolver problemas de computação.
- Engenharia Informática. Os profissionais deste campo preocupam-se especialmente com a conceção e construção de computadores e sistemas baseados em computador.
- Sistemas de Informação. Os profissionais desta disciplina concentram-se na integração de soluções de tecnologia de informação e processos de negócio para atender as

necessidades de informação do negócio e de outras empresas, permitindo às empresas alcançar os objetivos de uma forma eficaz e eficiente. Para além de especificarem, conceberem e implementarem sistemas de informação, também estão envolvidos na conceção de sistemas organizacionais baseados em tecnologia e de sistemas de colaboração.

- Tecnologia de Informação. Comparativamente com os sistemas de informação, a ênfase é mais sobre a tecnologia em si do que a informação que é transmitida. Os especialistas de tecnologias de informação assumem a responsabilidade para selecionar os produtos de hardware e software apropriados para uma organização, assumem a responsabilidade para integrar esses produtos com as necessidades organizacionais e de infraestrutura e assumem a responsabilidade para instalar, personalizar e manter essas aplicações para os utilizadores da organização.
- Engenharia de Software. É a disciplina que se preocupa em desenvolver e manter sistemas de software fiáveis, eficientes, económicos para desenvolvimento e manutenção e que satisfaça todos os requisitos dos clientes. No mundo profissional esta área pode ser vista como uma pessoa programadora de computadores ou alguém que gere projetos de software grandes, complexos e/ou críticos.

Neste trabalho, são abordados vários aspetos de todas as disciplinas de computação. Contudo os aspetos relacionados com a disciplina de Sistemas de Informação, Tecnologia de Informação e Engenharia de Software terão uma maior ênfase.

Os objetivos deste trabalho estão principalmente relacionados com a criação de valor de negócio, para dois tipos de organizações distintas, através da utilização do modelo de computação em nuvem (Tecnologias de Informação).

As organizações fornecedoras de recursos e serviços de TI, em princípio, poderão alcançar uma vantagem competitiva através do desenvolvimento, manutenção e entrega de sistemas de software baseados em computação em nuvem (Engenharia de Software). Contudo, antes de se avançar para a implementação de um projeto deste tipo, é necessário que as organizações compreendam a complexidade de gestão destes tipos de ações estratégicas (consultar subsecção 2.2.2).

As organizações com sistemas logísticos de transportes, em princípio, poderão alcançar uma vantagem competitiva através de um bom aproveitamento, exploração e integração da

computação em nuvem com os seus processos de negócio (Sistemas de Informação). A atividade inerente ao sucesso dos sistemas de informação é a atividade de gestão de sistemas de informação (consultar subsecção 2.2.3).

2.2.2 Gestão Organizacional

Antes de alcançar uma explicação sobre o que engloba a atividade de gestão organizacional, esta secção inicia com a apresentação dos conceitos de organização e de gestão.

Uma organização consiste num grupo de duas ou mais pessoas que trabalhem em conjunto, de forma estruturada, para atingir um objetivo ou um conjunto de objetivos específicos. A sua estrutura pode ser mais formal ou mais informal, no entanto todos os membros da organização devem procurar os benefícios de trabalhar em conjunto para alcançar os objetivos da organização. Para alcançar estes objetivos, embora seja mais evidente numas organizações do que em outras, todas as organizações têm gestores e caso estes não realizem uma gestão eficaz, provavelmente as organizações irão fracassar (Stoner, Freeman e Gilbert 1996).

Os contributos para o desenvolvimento da gestão foram-se sucedendo ao longo dos séculos (Soares 1998). Contudo, desde o final do século XIX, é costume definir a gestão em relação a quatro funções específicas dos gestores (Stoner, Freeman e Gilbert 1996):

- Planear: atividade relacionada com a determinação dos objetivos de uma organização e o estabelecimento de estratégias adequadas para os alcançar;
- Organizar: atividade que engloba a distribuição e delegação de tarefas, de recursos e de autoridade entre membros de uma organização, de modo a que os membros possam atingir os objetivos da organização;
- Liderar: atividade que envolve dirigir, influenciar e motivar os funcionários a realizarem tarefas essenciais;
- Controlar: atividade com o objetivo de assegurar a realização das tarefas conforme o que foi planeado. Implica o estabelecimento de normas de desempenho, a medição dos resultados atingidos, a comparação destes resultados com as normas estabelecidas e a tomada de ações corretivas quando se detetam desvios entre os resultados e as normas.

Assim, a atividade de gestão pode ser definida como “o processo de planear, organizar, liderar e controlar as atividades dos membros da organização e o uso de todos os outros recursos

organizacionais com o propósito de atingir os objetivos estabelecidos para a organização" (Stoner, Freeman e Gilbert 1996, p. 11). Um modelo representativo desta atividade frequentemente utilizado é o ilustrado na Figura 2.

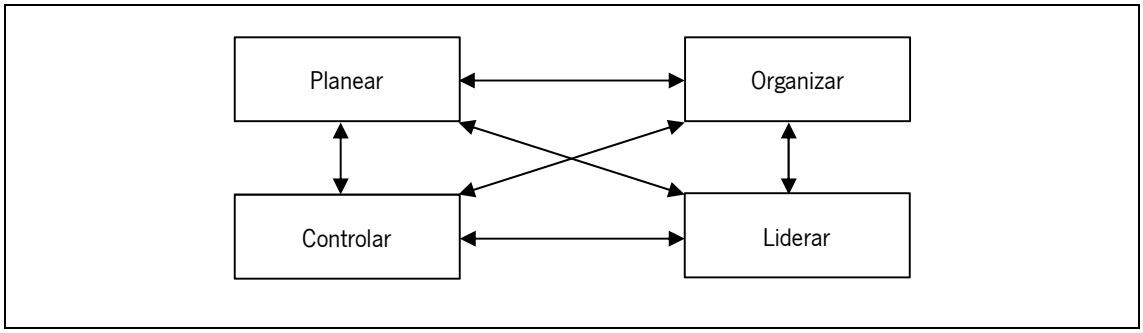


Figura 2 – Modelo do processo de gestão (retirado de Soares (1998)).

Mas, para além do reconhecimento da atividade de gestão, também se considera relevante o reconhecimento dos diferentes tipos e estruturação das decisões envolvidas e as diferentes exigências informacionais (Soares 1998), uma vez que os gestores podem executar estas quatro atividades em diferentes níveis de gestão organizacional (Stoner, Freeman e Gilbert 1996). Neste contexto, e conforme se ilustra na Figura 3, Anthony (1965) identificou três níveis de gestão nas organizações:

- Gestão estratégica: é o processo de assegurar que os recursos sejam obtidos e utilizados de forma eficaz e eficiente na concretização dos objetivos da organização;
- Gestão tática: é o processo de decidir as mudanças nos objetivos da organização, nos recursos que estão a ser utilizados para atingir esses objetivos, e nas políticas que ajudam a governar a aquisição e a utilização desses recursos;
- Gestão operacional: o processo de assegurar a aquisição e utilização eficiente dos recursos, no que diz respeito às atividades para as quais a relação ótima entre os *outputs* e os recursos pode ser determinada aproximadamente.

Deste modo, e a título de exemplo, a informação acerca do ambiente revela-se mais importante para os níveis superiores de gestão do que para os inferiores. Por outro lado, a informação relativa às operações internas da organização interessam, sobretudo, aos gestores do nível da gestão operacional (Soares 1998).

Para este projeto de investigação, a informação que se refere ao nível de gestão estratégica é a mais abordada, apesar de se reconhecer que cada uma destas atividades de gestão assume

uma importância significativa no seio da organização e que os gestores devem ser os responsáveis pelo cumprimento das mesmas a todos os níveis de gestão organizacional.

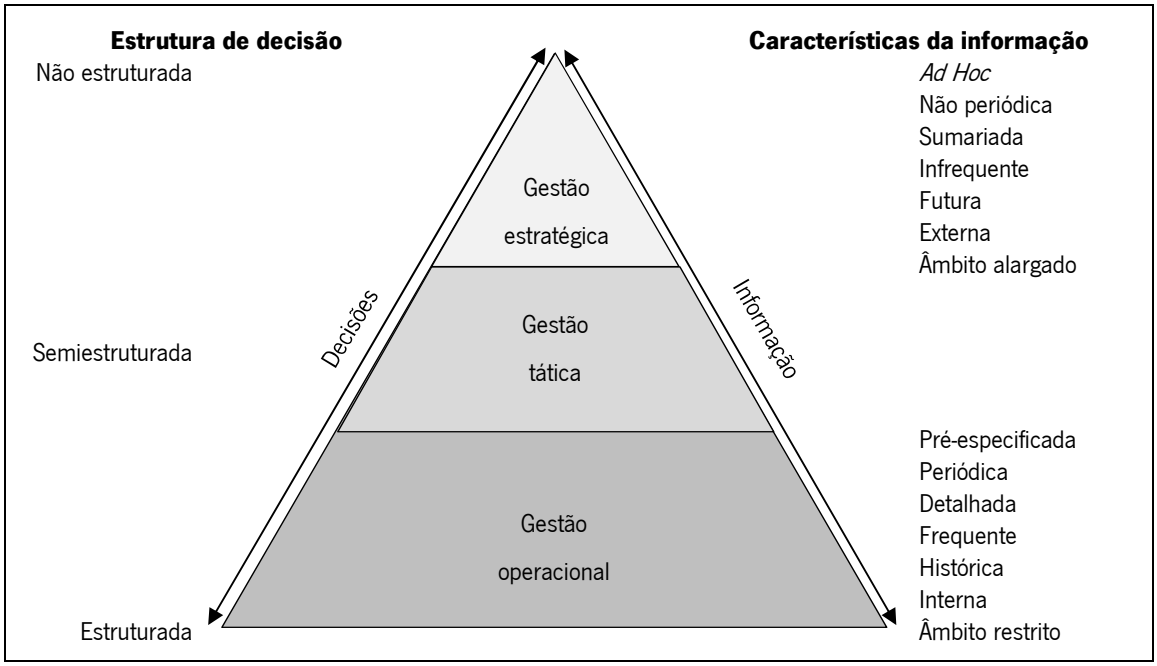


Figura 3 – Os diferentes níveis de gestão (retirado de O'Brien (1993)).

Nesta dissertação, a importância das atividades de gestão estratégia prende-se com o facto de ser crítico analisar se a definição de uma nova estratégia, para a criação de uma solução de computação em nuvem, no caso das organizações fornecedoras de recursos e serviços de TI, ou para a utilização de uma solução de computação em nuvem, no caso das organizações com processos de SLT, poderá ser ou não uma iniciativa inovadora que contribua para competitividade/sustentabilidade das organizações. Para além disso, neste contributo, são objetos de análise métodos e técnicas que permitam, de alguma forma, ajudar estas organizações a estabelecerem um plano estratégico, para a adoção e implementação de uma solução de computação em nuvem, e a decidirem pela aceitação ou recusa desse plano.

Assim sendo, de seguida, procede-se a uma descrição mais detalhada sobre o campo e o processo de gestão estratégica e uma explicação da sua importância.

O campo da gestão estratégica pode ser referido como o campo que "analisa as principais iniciativas desejadas e emergentes assumidas pela gestão de topo em nome dos proprietários, envolvendo a utilização de recursos, para melhorar o desempenho das empresas nos seus ambientes externos" (Nag, Hambrick e Chen 2007, p. 944).

O processo de gestão estratégica é descrito como um processo dinâmico no qual as empresas devem evolui continuamente com novas entradas estratégicas. O primeiro passo consiste em

analisar os ambientes externos e internos para determinar os recursos, as capacidades e as competências essenciais. Com essa informação, a empresa desenvolve a visão e missão e formula a sua estratégia. E, de forma a implementar a estratégia, a empresa assume compromissos, decisões e ações para alcançar competitividade estratégica e obter retornos acima da média[1] (Hitt, Ireland e Hoskisson 2012).

Aprender a competir com sucesso no mundo globalizado é um dos desafios mais importantes para as empresas no século XXI, de acordo com Hitt, Ireland e Hoskisson (2012). Os gestores devem adotar uma nova mentalidade em que os valores e os desafios evoluem a partir de uma constante evolução das condições. Por isso, desenvolver e implementar uma estratégia contínua deve ser um elemento importante para o sucesso das organizações. Isto permitirá que ações estratégicas sejam planeadas e as ações ocorram quando as condições ambientais são as apropriadas. Para além disso, também ajudará a coordenar as estratégias desenvolvidas pelas unidades de negócio em que a responsabilidade de competir em mercados específicos é descentralizada.

Tendo consciência da realidade atual, as empresas deverão produzir continuamente produtos inovadores. Contudo, para alcançar esta capacidade de inovação, as empresas precisam de ter ou procurar uma estrutura organizacional adequada e os recursos apropriados. Por exemplo, empresas como a Procter & Gamble e a Johnson & Johnson, ao longo do tempo, foram realizando esforços para alterar a sua cultura e os seus modelos de negócios com o objetivo de aumentarem a sua produção de inovação e permanecer competitivas no contexto atual (Hitt, Ireland e Hoskisson 2012).

2.2.3 Gestão de Sistemas de Informação

De facto, a importância que as TI apresentam para as organizações é inquestionável. Contudo, os responsáveis das organizações devem questionar sempre a forma como os SI são utilizados (Soares 1998). Por isso, julga-se pertinente apresentar uma explicação geral sobre TI, SI e GSI e, relativamente a estas, justificar para onde converge o interesse de investigação nesta dissertação.

[1] Retornos acima da média são os retornos em excesso que um investidor espera ganhar de outros investimentos com uma quantidade similar de risco (Hitt, Ireland e Hoskisson 2012).

Relativamente ao termo "sistemas de informação", existe mais do que um possível significado (Carvalho 2000). Todavia, uma definição comum para SI é a proposta por Buckingham (1987): "Sistema de Informação é um sistema que reúne, guarda, processa e faculta informação relevante para a organização (...), de modo que a informação é acessível e útil para aqueles que a querem utilizar, incluindo gestores, funcionários, clientes, (...). Um SI é um sistema de atividade humana (social) que pode envolver ou não a utilização de computadores" (Amaral 1994, p. 24).

Sobre o conceito de TI, numa perspetiva estritamente tecnológica, pode-se dizer que este é "o conjunto de equipamentos e suportes lógicos (hardware e software), que permitem executar tarefas como aquisição, transmissão, armazenamento, recuperação e exposição de dados" (Amaral 1994, p. 25).

Embora os SI possam existir sem a participação de computadores, as TI desempenham um papel cada vez mais importante nas organizações. Atualmente a maioria dos SI tem presentes as TI porque permite a eficiência das operações, bem como a eficácia da gestão (Watson 2007).

Porém, muitas organizações não conseguem alcançar o valor esperado do investimento nas TI (Henderson e Venkatraman 1993). Em virtude disso, os SI devem suportar adequadamente os objetivos estratégicos e as necessidades de gestão da organização, beneficiando a sobrevivência e sucesso da mesma (Soares 1998). Assim, a preocupação fundamental das organizações deve passar pela procura evolutiva e dinâmica de um alinhamento que permita a aplicação das TI, de forma adequada e oportuna, em harmonia com as estratégias, objetivos e necessidades das organizações (Luftman 2000; Henderson e Venkatraman 1993).

Desta forma, as organizações necessitam de poderem ou deverem contar com alguém responsável pela gestão da mudança que resulta da adoção das TI (Amaral 2005). Neste contexto, a atividade de GSI constitui uma tarefa essencial para os responsáveis das organizações (Soares 1998).

A atividade de GSI pode ser definida como "a gestão do recurso informação e de todos os recursos envolvidos no planeamento, desenvolvimento, exploração e manutenção do SI" (Amaral 1994, p. 36), conforme se sugere na Figura 4.

Esta atividade é reconhecida por muitos autores como uma atividade de enorme complexidade (Soares 1998). De acordo com Amaral (2005), é uma atividade que está perante um domínio amplo e complexo. Amplo por envolver e interferir em todos os aspetos da organização, em

qualquer dos seus níveis de gestão e operação e em qualquer das suas áreas funcionais (alguns destes aspetos estão apresentados genericamente na subsecção 2.2.2). Complexo porque reflete naturalmente toda complexidade da estrutura e funcionamento da organização, nomeadamente a que resulta da componente humana ou natureza sociotécnica.

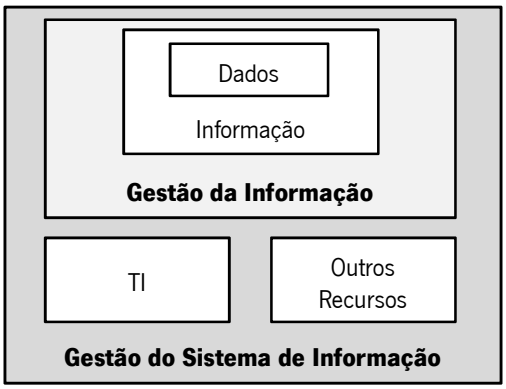


Figura 4 – Da Gestão da Informação à Gestão do Sistema de Informação (retirado de Amaral (1994)).

De forma a estabelecer limites e conteúdos para as atividades que compõem a GSI e propor um referencial que permita perspetivar a "posição" relativa das atividades de planeamento e desenvolvimento Organizacional e do SI em todo o espaço da gestão organizacional (Amaral 2005), Carvalho e Amaral (1993) formularam uma representação matricial, conforme se apresenta na Figura 5. Nessa representação combinam-se:

- No eixo vertical as atividades de natureza holística que contemplam a natureza sistémica do objeto de estudo, versus as atividades construtivas, incrementais, associadas às suas implementações. Genericamente rotuladas de atividades de "Planeamento" e de "Desenvolvimento";
- No eixo horizontal os dois níveis de abrangência daquelas atividades: toda a "Organização" ou apenas o seu "Sistema de informação".

Identificam-se assim, em todo o espaço da gestão organizacional, quatro atividades distintas (Amaral 2005):

- Planeamento de Sistemas de Informação: atividade de conceção global do Sistema de Informação;
- Desenvolvimento de Sistemas de Informação: atividade de obtenção e/ou construção dos diversos componentes do sistema de informação;

- Planeamento Organizacional: atividade de planeamento estratégico e global da organização;
- Desenvolvimento Organizacional: atividades de desenho, reestruturação e racionalização da organização não limitadas ao sistema de informação.

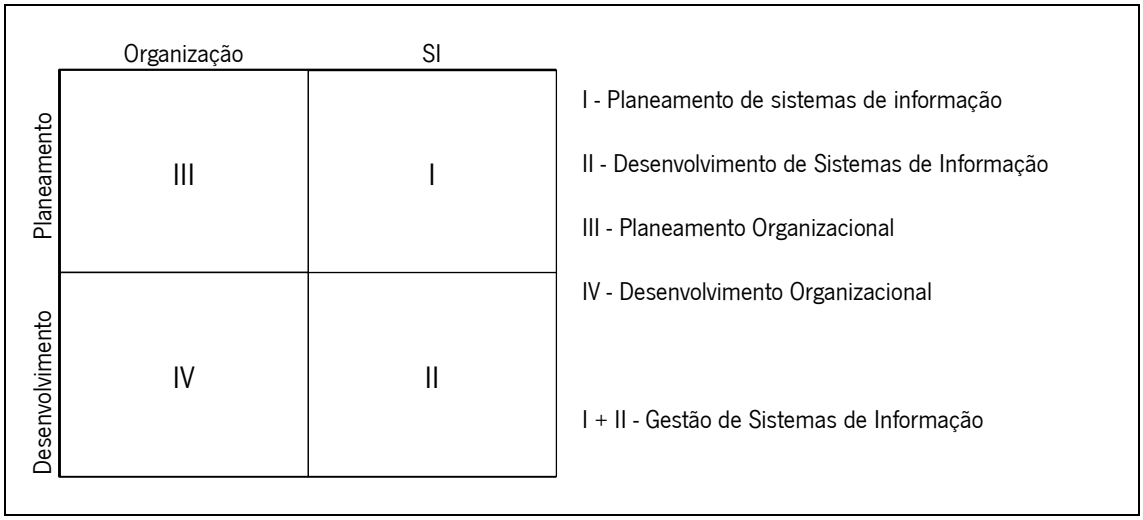


Figura 5 – Matriz de Atividades de planeamento e desenvolvimento organizacional e do SI (retirado de Amaral (2005)).

No âmbito deste trabalho, a atividade de Desenvolvimento de Sistemas de Informação é a que terá uma maior enfâse, numa perspetiva de uma organização que pretende fornecer uma solução de TI com o propósito de melhorar o SI do consumidor. Ou seja, procura-se que uma organização (i.e. departamento ou empresa fornecedora de recursos e serviços de TI) seja capaz de conceber um componente de SI baseado no modelo de computação em nuvem, tendo em consideração as necessidades de gestão e de SI, que beneficie a sobrevivência e o sucesso da organização consumidora (i.e. uma organização com sistemas logísticos de transportes).

2.3 Conceito de Computação em Nuvem

Recentemente, o conceito de computação em nuvem disparou em popularidade (Bahl et al. 2012). Contudo, a frequente utilização do conceito, em particular para fins comerciais, gerou uma ideia errada sobre o seu propósito real (Böhm et al. 2011; Vaquero et al. 2008). Nesta secção, pretende-se fornecer um entendimento comum sobre este conceito e em particular sobre os seus aspetos essenciais através de uma perspetiva de invocação.

2.3.1 A Origem e a Inovação de Computação em Nuvem

Começando pela origem, de acordo com Francis (2009), o conceito de computação em nuvem surgiu nos anos 60 como uma forma de computação organizada de utilidade pública e derivou da representação gráfica comum, para os recursos de computação baseados na Internet, dos diagramas de sistemas.

Atualmente, segundo Böhm et al. (2011), este conceito é utilizado, por vezes, para se referir a um novo paradigma ou uma nova tecnologia que de forma flexível oferece recursos e serviços de TI através da Internet.

De um ponto de vista tecnológico, este conceito é um avanço de computação, que historicamente pode ser rastreado a partir do século XVII e que revela o desenvolvimento de uma tendência de máquinas de computação locais, de *mainframe* centrais, através de computadores e dispositivos móveis para uma tendência de centralização seminova, que a computação em nuvem revela (Böhm et al. 2011).

De forma a compreender esta tendência, a análise de Grossman (2012), sobre as suas primeiras quatro Eras de Computação Digital, pode ser útil. Mas antes de partir para a análise do autor, resumidamente, a evolução da computação digital pode ser apresentada, assim:

1. Na era do *mainframe* (1965 - 1985), a computação estava centralizada, uma equipa especialista em computação controlava o sistema e as empresas geriam informação ao nível organizacional.
2. Na era do computador pessoal (1980-2000), a computação era baseada no modelo cliente-servidor. As pessoas poderiam comprar, instalar e usar aplicações *desktop* sem o envolvimento de um departamento de computação centralizado e as organizações começaram a lidar com informação gerada, pelos colaboradores, ao nível do departamento.
3. Na era da Web (1995-2015), o software passou a residir, muitas vezes, em máquinas remotas disponíveis através de um navegador Web sobre a rede. Sobre a informação, apesar de parecer um contrassenso, uma vez que de repente está tudo acessível, esta é raramente organizada o suficiente para que possa ser útil.
4. A era do dispositivo (2005-2025) é sobre a transição de aplicações baseadas em navegadores Web para dispositivos, de redes de cobre para redes sem fios e de fibra, de um modelo de software baseado em aplicação para um modelo de software baseado em

serviços. A quarta era é cheia de dispositivos de computação que fornecem nuvens de serviços através de uma rede IP sempre presente e ubíqua.

A análise de Grossman (2012) assinala que nas duas primeiras eras, o computador estava no centro do modelo e anexados ao computador estavam vários dispositivos periféricos, tais como: terminais, impressoras e unidades de disco. Na terceira era, a rede começou a mover-se em direção ao centro e que, para o fim da quarta, esta estará no centro e os CPUs, as unidades de disco e os dispositivos sem fio estarão como periféricos.

Cruzando as opiniões de Böhm et al. (2011) e Grossman (2012), a computação em nuvem parece estar representada na quarta era e que de facto existiu uma evolução tecnológica até se identificar este conceito.

Enquadradas com esta opinião, existem outras de vários autores. Por exemplo, Youseff, Butrico, e Da Silva (2008) consideram que o conceito de computação em nuvem é baseado numa coleção de velhos e novos conceitos em vários áreas de investigação, como *Service-Oriented Architectures* (SOA), computação distribuída, computação em Grid e virtualização. Outro exemplo, é a opinião de Rimal, Choi, e Lumb (2009). Estes autores são da opinião que a Computação em Nuvem herda *legacy systems*[2] e aborda o próximo passo evolutivo da computação distribuída. Ainda a este respeito, também Vaquero et al. (2008) elaborou apreciações nesse sentido. Os autores alegaram que algumas das tecnologias existentes que a nuvem se baseia (como virtualização, computação de utilidade ou computação distribuída) não são novas e algumas delas regressam das raízes da computação.

Para além da visão tecnológica, o conceito de computação em nuvem também pode ser visto como uma inovação na perspetiva de prestação de serviços de TI. Acredita-se essencialmente que este tem o potencial de revolucionar o modo de desenvolvimento e implementação de recursos de computação e aplicações, abrindo espaço para novos modelos de negócio para as empresas fornecedoras de software (Böhm et al. 2011; Stuckenberg, Fielt e Loser 2011; Youseff, Butrico e Da Silva 2008).

No âmbito deste contributo de investigação, apenas a perspetiva tecnológica de inovação, apresentada nesta subsecção, é a mais relevante uma vez que engloba os principais aspetos

[2] Um *legacy system* é um método, tecnologia, sistema de computação ou programa de software antigo que continua a ser utilizado, normalmente porque ainda cumpre necessidades dos utilizadores, apesar de já existirem novas tecnologias ou métodos mais eficientes para realizar uma tarefa (TheFreeDictionary.com 2012).

necessários para a conceção de sistemas de software baseados no modelo de computação em nuvem. No entanto, a perspetiva de prestação de serviços de TI também deverá, de forma semelhante, ser útil num trabalho futuro para a construção de um modelo de negócio baseado no modelo de computação em nuvem para as empresas fornecedoras de software.

2.3.2 Definição de Computação em Nuvem

Atualmente é extremamente fácil encontrar uma definição sobre o conceito de computação em nuvem, no entanto a tarefa parece complicar-se quando é necessário encontrar uma definição precisa, enquadrada com um âmbito de investigação.

Youseff, Butrico, e Da Silva (2008) estiveram entre os primeiros que tentaram fornecer uma definição completa de computação em nuvem. Estes autores consideram a computação em nuvem como "um novo paradigma de computação que permite aos utilizadores utilizar temporariamente infraestrutura de computação em rede, fornecidas como um serviço pelo fornecedor de Nuvem a um ou mais níveis de abstração" (Youseff, Butrico e Da Silva 2008, p. 1). Estes níveis de abstração referem-se à ontologia, desenvolvida pelos mesmos, que é descrita na subsecção 2.3.3.

Outros autores que estiveram à frente para fornecer uma definição globalmente aceite do que é a computação em nuvem foram Vaquero et al. (2008). Neste estudo de 2008, é referido que o conceito de computação em nuvem ainda estava a ser construído, mas que podia ser entendido como "um grande conjunto de recursos virtualizados facilmente utilizáveis e acessíveis. Esses recursos são dinamicamente reconfigurados a uma carga variável que permite a utilização ideal dos recursos. Este conjunto de recursos é normalmente explorado por um modelo de *pay-per-use*[3] em que as garantias são oferecidas pelo fornecedor de infraestrutura por meio de *service level agreements*[4] personalizados" (Vaquero et al. 2008, p. 2).

Entretanto, uma definição informal amplamente utilizada surgiu, a definição de National Institute of Standards and Technology (NIST) elaborada por Mell e Grance (2011). Esta definição caracteriza o conceito como "um modelo para permitir acesso a rede de forma ubíqua,

[3] *Pay-as-you-go* (ou *pay-per-use*) é um modelo de pagamento de serviços de acordo com a utilização efetiva em serviços (TechTarget 2005).

[4] *Service level agreement* é parte de um contrato de serviço, onde o nível de serviço é definido formalmente. Na prática, este é por vezes utilizado para se referir ao prazo de entrega ou desempenho contratado (TheFreeDictionary.com 2012).

conveniente e sob procura a um conjunto compartilhado de recursos computacionais configuráveis que podem ser rapidamente fornecidos e substituídos com mínimo esforço de gestão ou mínima interação com o prestador de serviços" (Mell e Grance 2011, p. 6).

Além destas, outros investigadores preferiram definir o seu próprio conceito no âmbito da sua própria investigação, como a definição de Armbrust et al. (2009), Buyya et al. (2009) e Böhm et al. (2011).

Frequentemente utilizada na literatura, é a definição de Armbrust et al. (2009, p. 1). De acordo com os autores, a computação em nuvem "refere-se tanto às aplicações entregues como serviços através da Internet, como ao hardware e sistemas de software dos centros de dados que prestam esses serviços. Os próprios serviços são referidos como *software as a service*. O hardware e o software de centros de dados são a nuvem. Quando uma Nuvem é disponibilizada numa forma *pay-as-you-go* ao público em geral, dá-se o nome de nuvem pública, e o serviço que está a ser vendido é *utility computing*. Nuvem privada refere-se a centros de dados internos de uma organização, que não está disponibilizada ao público em geral. Assim, a computação em nuvem é a soma de *software as a service* e *utility computing*, mas não inclui nuvens privadas." Nesta definição, os autores entendem a computação em nuvem como um termo coletivo, que abrange conceitos de computação pré-existentes, tais como *software as a service* e *utility computing*.

Semelhante à definição Vaquero et al. (2008) e com base na observação da essência do que as nuvens prometem ser, Buyya et al. (2009, p. 3) propõe a seguinte definição: "uma nuvem é um tipo de sistema paralelo e distribuído que consiste num conjunto de computadores interligados e virtualizados que são dinamicamente fornecidos e apresentados como um ou mais recursos de *unified computing*[5] com base em *service level agreements* estabelecidos através de negociação entre o serviço fornecedor e consumidor".

Por fim, a última definição é a de Böhm et al (2011, p. 7). O autor estruturou o conceito com o objectivo de fornecer uma definição holística, tanto na perspectiva de aplicação como de infra-estrutura. Sendo apresentada como "um modelo de implementação de TI, baseado em virtualização, onde os recursos, em termos de infraestrutura, aplicações e dados são implementados através da Internet como um serviço distribuído por um ou vários prestadores de

[5] Um *unified computing system* é uma arquitetura de centro de dados que integra recursos de computação, redes e armazenamento para aumentar a eficiência e permitir a gestão centralizada (TechTarget 2009).

serviços. Estes serviços são adaptáveis sob procura e podem ser taxados a um preço baseado em *pay-per-use*".

Tendo em conta a revisão de literatura efetuada, as definições de Mell e Grance (2011) e Youseff, Butrico, e Da Silva (2008) parecem ser as definições mais amplamente aceites. Para este trabalho de dissertação a que melhor se enquadra é a de Mell e Grance (2011) por duas razões. Primeiro porque é mais esclarecedora que a de Youseff, Butrico, e Da Silva (2008), talvez por se tratar de uma definição mais recente. Segundo porque foca os aspetos de infraestrutura e os aspetos de aplicação que são essenciais para elaboração de arquiteturas de SI baseadas no modelo de computação em nuvem.

2.3.3 Descrição de Computação em Nuvem

Sobre o conceito de computação em nuvem, vários modelos de representação do conhecimento foram desenvolvidos, como o de Youseff, Butrico, e Da Silva (2008), Rimal, Choi, e Lumb (2009), Katzan Jr (2010), Schubert, Jeffery, e Neidecker-Lutz (2010) e Mell e Grance (2011). Por isso, parece suficiente apenas apresentar uma das primeiras representações concretas que descreve a computação em nuvem.

Tal como nas definições, uma das primeiras representações de toda a área de computação em nuvem foi elaborada por Youseff, Butrico, e Da Silva (2008). Os autores propuseram uma ontologia que, conforme a Figura 6, traduz a interdependência e agregação entre cinco camadas: Aplicação em Nuvem, Ambiente de Software em Nuvem, Infraestrutura de Software em Nuvem, Software Kernel e Firmware/Hardware. Uma camada é constituída por um ou mais serviços com níveis equivalentes de abstração evidentes para os seus utilizadores alvo. E quando uma camada se encontra num nível superior a outra, esta tem um ou mais serviços que podem ser compostos a partir de outros serviços de uma camada subjacente.

Assim, e sinteticamente, pode descrever-se a camada de Aplicação em Nuvem como a mais visíveis para os utilizadores finais da nuvem. Estes utilizadores acedem aos serviços, normalmente, através de portais e o trabalho computacional das infraestruturas dos utilizadores passa para centros de dados externos de fornecedores, onde as Aplicações em Nuvem são instaladas. Estas aplicações podem ser desenvolvidas através da utilização dos serviços da camada Ambiente de Software em Nuvem ou Infraestrutura de Software em Nuvem. Além disso, utilizando os conceitos de SOA, estas aplicações podem ser compostas como um serviço de

outros serviços oferecidos por outros sistemas de Nuvem. Sobre os serviços desta camada, estes são referidos geralmente como um modelo de *Software as a Service* (SaaS).

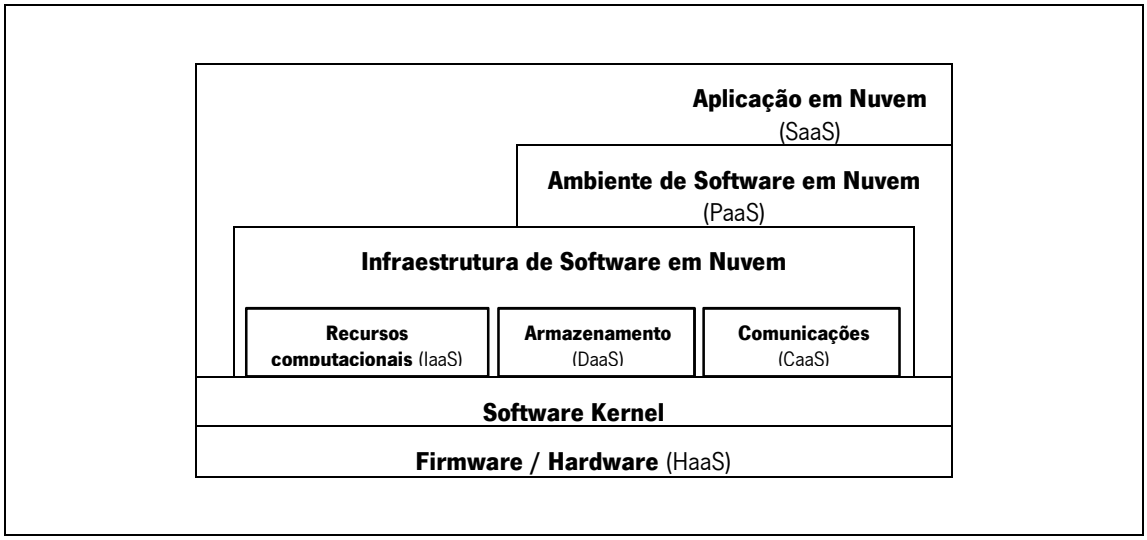


Figura 6 - Ontologia de Computação em Nuvem (retirado de Youseff, Butrico e Da Silva (2008)).

A segunda camada é a Ambiente de Software em Nuvem ou Plataforma de Software. Os utilizadores desta camada são os programadores de aplicações em Nuvem, para implementar aplicações e instalar aplicações na Nuvem. Nestes sistemas de Nuvem, os serviços prestados são vulgarmente mencionados como *Platform as a Service* (PaaS).

Por sua vez, a terceira camada, Infraestrutura de Software em Nuvem, fornece os recursos fundamentais às camadas superiores, sem que estas se preocupem com a complexidade desta camada, para a construção de novos Ambientes de Software em Nuvem ou Aplicações em Nuvem. A interface de utilizador é diferente de sistema para sistema, no entanto dois exemplos de protocolos de interface é o *Simple Object Access Protocol* (SOAP) e o *Representational State Transfer* (REST). Relativamente os serviços, estes podem ser classificados como:

- *Infrastructure as a Service* (IaaS) - permite fornecer recursos computacionais, geralmente, através de máquinas virtuais (VMs).
- *Data-Storage as a Service* (DaaS) - possibilita armazenar e aceder a dados remotamente a qualquer momento a partir de qualquer lugar.
- *Communication as a Service* (CaaS) - pretende garantir a qualidade de serviço (QoS) para comunicação em rede entre sistemas de Nuvem.

A camada Software Kernel fornece a gestão de software base para os servidores físicos que compõem a nuvem. A este nível pode ser implementado um sistema operacional *kernel*, um *hypervisor*, um *clustering middleware*, entre outros.

A última camada, Firmware/Hardware, refere-se ao hardware físico real e aos *switches* que constituem o suporte principal da Nuvem. Os serviços prestados pelo fornecedor desta camada tem como nome *Hardware as a Service* (Haas) e inclui a operação, gestão e atualização de hardware para um período de aluguer definido com o utilizador.

2.3.4 A Computação em Nuvem no Mundo

Pretende-se com esta subsecção dar uma visão geral da tendência atual do conceito de computação em nuvem em todo o mundo e da sua evolução. Para efeito de estudo atual e evolutivo, serão utilizados os dados do relatório The Essential CIO de 2011 da IBM, os dados do levantamento realizado pela WikiCFP até 30 de Dezembro de 2012, as previsões de Gartner de 2012, os dados do relatório de Garner's Hype Cycle de 2012 e os resultados do Google Trends referentes aos anos entre 2004 e 2012.

De acordo com as previsões de Gartner (2012), o mercado de serviços em nuvem é claramente um sector de alto crescimento no mercado global de TI, como se apresenta na Figura 7. Gartner prevê que o mercado de serviços em nuvem irá crescer 17.7% até 2016, de $75.6 mil milhões em 2010 para $206.6 mil milhões em 2016 (Gartner Group 2012). Por isso, e para aproveitar este crescimento, Gartner considera que as empresas de TI devem ajustar as suas estratégias.

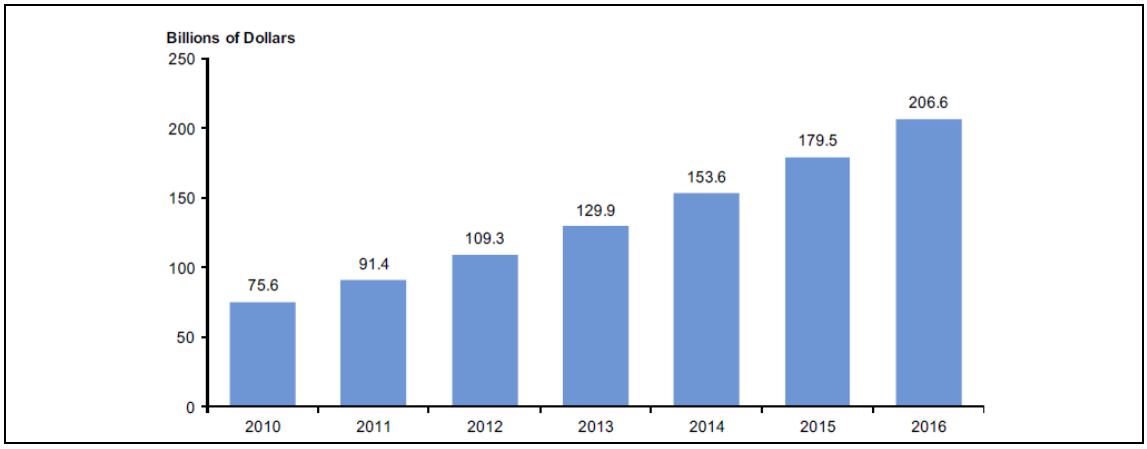


Figura 7 – Previsões do crescimento do mercado de serviços em nuvem (retirado de Gartner Group (2012)).

Contudo, existem tipos de serviços em nuvem com mais mercado do que outros. Segundo as previsões de Gartner Group (2012), atualmente os serviços de nuvem do tipo *Business Process as a Services* (BPaaS) representam cerca de 77% do mercado total. No futuro, este tipo de serviço irá continuar a ser o maior segmento, mas a sua percentagem irá diminuir para os restantes serviços, principalmente para o IaaS e SaaS, conforme se ilustra na Figura 8.

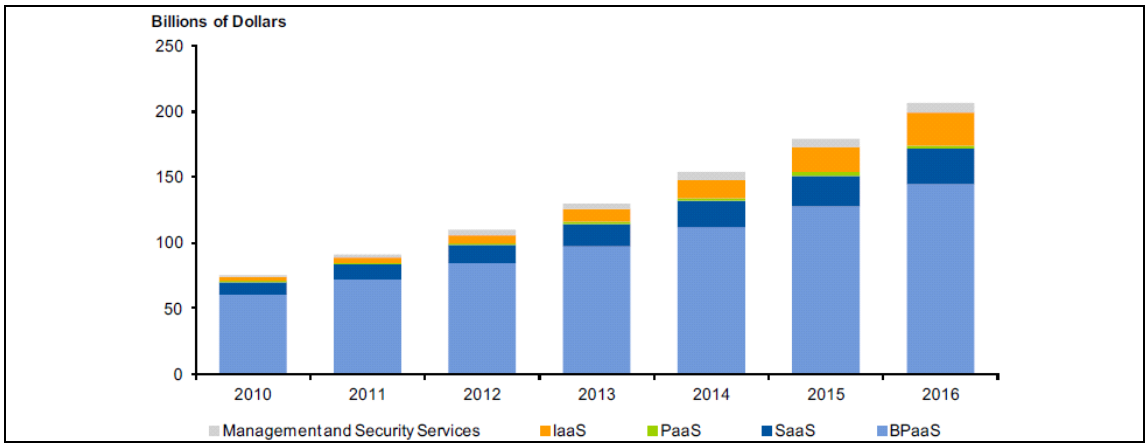


Figura 8 - Tamanho do mercado de serviços em nuvem públicos por segmento (retirado de (Gartner Group 2012)).

A IBM realizou um estudo a mais de 3000 Chief Information Officer (CIO) sobre como estes estão a pensar ajudar as suas organizações a adaptar-se à mudança acelerada e à complexidade que marca o panorama competitivo e económico dos dias de hoje (Horan 2011). Os resultados indicam que para aumentar a competitividade das organizações, nos próximos três a cinco anos, 83% dos CIOs têm planos estratégicos que incluem projetos de *Business Intelligence and Analytics*, 74% de *Mobility Solutions*, 68% de *Virtualization* e 60% de Computação em Nuvem, tal como se ilustra na Figura 9. Identifica-se assim, que desde do estudo de 2009, a computação em nuvem disparou em prioridade, saltando para a quarta posição com a mesma importância do *Business Process Mangament*.

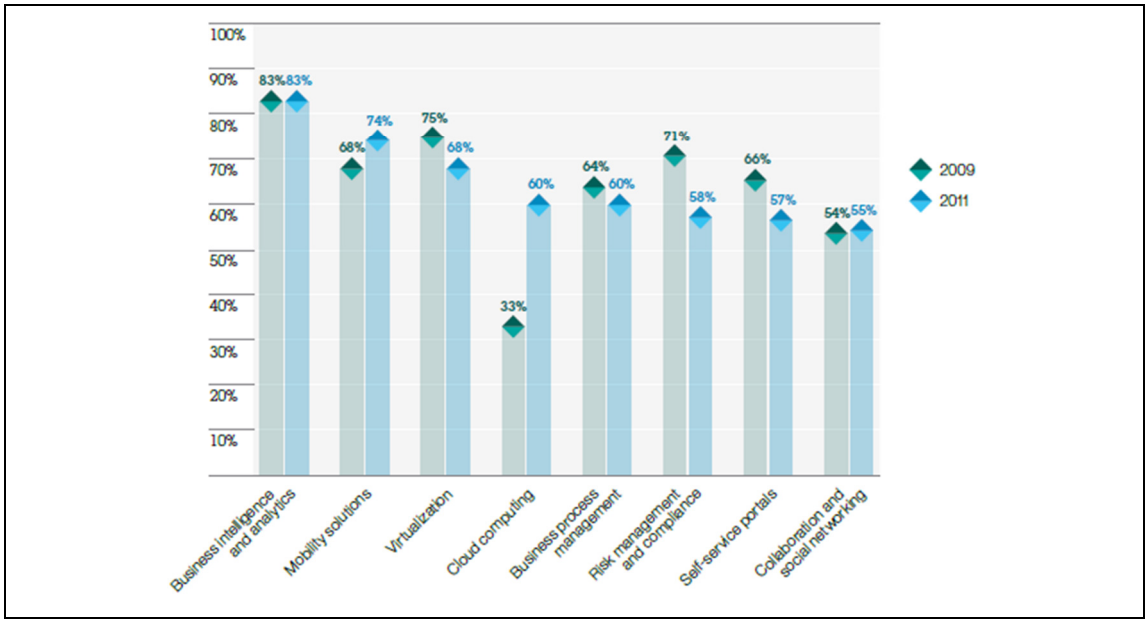


Figura 9 – Reajuste estratégico dos CIO para 2011 (retirado de Horan (2011)).

Sobre a WikiCFP, esta é uma *wiki* semântica de "Calls For Papers" para áreas científicas e tecnológicas (Wikicfp.com 2012). De acordo com o levantamento da WikiCFP, e como se pode

analisar na Tabela 3, até ao momento, o termo computação em nuvem encontra-se no trigésimo segundo lugar, com um total de 296 eventos dedicados a submissões de artigos sobre esta área.

Tabela 3 – Algumas categorias populares de "Calls For Papers" (adaptado de Wikicfp.com (2012)).

	Categoria	**Número de CFPs**
1°	*artificial intelligence*	1326
2°	*communications*	1022
3°	*software engineering*	901
32°	*cloud computing*	296
297°	*wireless communication*	26

Se for ignorado o ano 2013 que ainda não terminou, e os eventos sem data associada é possível verificar que o número de eventos "Call For Papers", apresentados na Tabela 4, aumentou de 2009 até 2012, no entanto, o seu crescimento tem diminuído de ano para ano. Perante estes dados, pode-se induzir que o interesse para investigar o "objeto" computação em nuvem continua a aumentar e que existe ainda muito conhecimento para se descobrir.

Tabela 4 – Eventos "Calls For Papers" sobre Computação em Nuvem no mundo (adaptado de Wikicfp.com (2012).

Resultados	**2009**	**2010**	**2011**	**2012**	**2013**	**N/A**	**Total**
Total de Eventos	4	45	77	93	43	34	296
Crescimento	-	41	32	16	(-50)	-	89

Relativamente ao relatório de Garner's Hype Cycle (2012), este é uma proposta de caracterização de como um *hype* sobre uma tecnologia evolui "de entusiasmo excessivo e um período de desilusão para uma eventual compreensão da relevância da tecnologia e do papel num mercado ou domínio". Segundo este relatório, a computação em nuvem é um dos termos tecnológicos que se desloca rapidamente pelas fases do ciclo de *hype* das tecnologias emergentes. Atualmente está a iniciar a terceira fase, rotulada de "Vale de Desilusão" ou "*hype* negativo", e deverá alcançar o "Planalto de Produtividade" ou o "eventual *hype* real" entre 2 a 5 anos, como se apresenta na Figura 10.

Estes dados reforçam a ideia que ainda subsiste uma confusão geral sobre o paradigma de computação em nuvem e as suas potencialidades. Este conceito ainda está numa fase de amadurecimento, após a fase positiva, é esperado que as espectativas excessivamente positivas desvaneçam e que as dúvidas sobre as capacidades reais da computação em nuvem surjam.

Apesar disso, e como se verificou em subsecções anteriores, é possível identificar imensas ideias recorrentes sobre o termo de computação em nuvem.

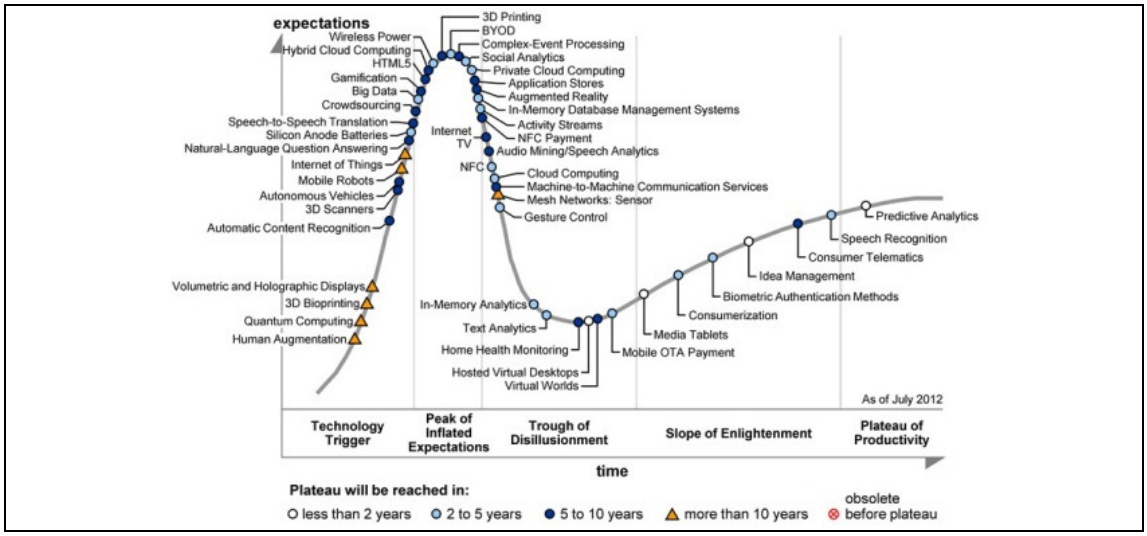


Figura 10 - O ciclo hype para as tecnologias emergentes (retirado de Gartner Group (2012)).

Por fim, e perante a situação que o termo computação em nuvem é por vezes comparado com outros paradigmas de computação, foi utilizada a ferramenta Google Trends que permite identificar o interesse ao longo do tempo sobre os termos *cloud computing*, *grid computing*, *utility computing* e *virtualization*. A Figura 11 apresenta os resultados obtidos. Eles revelam uma tendência que a computação em nuvem continua de facto a ser do interesse mundial das pessoas e que esse interesse tem diminuído (facto que talvez se deva à fase negativa de *hype* onde o termo se encontra, tal como apresentado no relatório de Garner's Hype Cycle). Além disso, esta tendência aponta no sentido que a computação em nuvem é um dos conceitos de computação que teve o maior pico de disseminação nos últimos tempos.

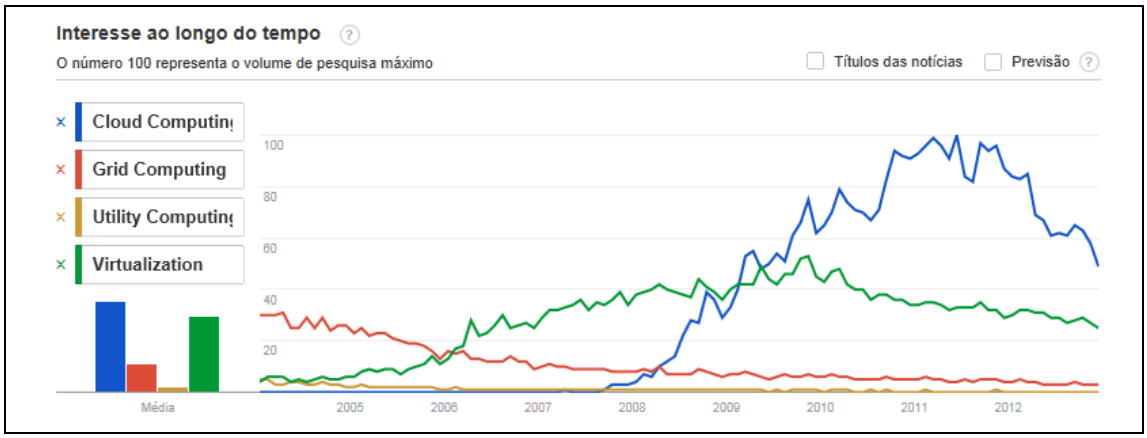


Figura 11 - Comparação da tendência de utilização entre os termos *grid*, *utility* e *cloud Computing* e *virtualization* (gerada e retirada do Google Trends em Dezembro de 2012).

2.4 Computação em Nuvem para Fornecedores e Utilizadores

A computação em nuvem pode fornecer oportunidades positivas, segundo Armbrust et al. (2010), para as pessoas utilizadoras de SaaS, fornecedores de SaaS ou utilizadoras de computação utilitária e fornecedores de computação utilitária. Seguindo esta opinião, para além destes três diferentes papeis, é também possível identificar dois tipos de serviços de computação em nuvem que podem trazer vantagens tanto para as pessoas fornecedores como as pessoas utilizadores desses serviços: serviços de computação utilitária e serviços de aplicação como um serviço. A Figura 12 apresenta a relação entre os utilizadores-finais e fornecedores de computação em nuvem.

Figura 12 – Utilizadores-finais e Fornecedores de Computação em Nuvem (adaptada de Armbrust (2009)).

Neste capítulo serão abordados os benefícios e fatores condicionadores da adoção da computação em nuvem, na perspetiva dos fornecedores e utilizadores de serviços de computação utilitária (subsecção 2.4.1) e na perspetiva dos fornecedores e dos utilizadores de serviços de aplicação como um serviço (subsecção 2.4.2). Para além disso, esta secção servirá de base para a discussão sobre a importância da criação de um sistema de informação baseado no modelo de computação em nuvem para o suporte a sistemas logísticos de transportes (capítulo 3).

Assim sendo, e antes de se avançar com os assuntos propostos nesta secção, julga-se essencial destacar que a utilização das palavras cliente, consumidor e utilizador referem-se de igual forma para as empresas de TI que sejam clientes de entidades que prestam serviço de computação utilitária e a palavra utilizador-final refere-se aos utilizadores finais do serviço (os clientes de entidades que fornecem serviços de SaaS).

2.4.1 Serviços de Computação Utilitária

Acredita-se que é cada vez maior a perceção que a computação se tornará num utilitário, como a água, a eletricidade, o gás e as telecomunicações (Buyya, Yeo e Venugopal 2008). As TI, em vez de serem um ativo que as próprias empresas detêm em forma de computadores, software e um miríade de componentes relacionados, poderão ser um serviço (público) que as empresam compram de fornecedores de utilitários (Carr 2005).

A computação utilitária, como todos os outros utilitários existentes, irá fornecer o nível básico de serviço de computação que é considerado essencial para realizar as necessidades diárias da comunidade em geral (Buyya, Yeo e Venugopal 2008).

Durante a história da computação, vários paradigmas de computação prometeram a entrega de computação utilitária, como a computação em *cluster* e a computação em *grid*. A computação em nuvem é o paradigma mais recente (Buyya, Yeo e Venugopal 2008).

Os serviços de computação utilitária de computação em nuvem (ou serviços de nuvem) focam-se nos *trade-offs* entre a conveniência, flexibilidade e portabilidade do programador, no ponto de vista do fornecedor e utilizador de computação utilitária (Armbrust et al. 2009).

Existem vários casos de sucesso e insucesso na implementação de serviços de computação utilitária (Armbrust et al. 2009). No caso dos que foram mal sucedidos temos, por exemplo, a Intel Computing Services. Quanto aos bem-sucedidos temos o caso da Amazon, Google, Salesforce, IBM, Microsoft e Sun Microsystems.

Para comercializar este tipo de serviço, Armbrust et al. (2009) consideram que é necessário primeiro perceber bem a economia de escala proporcionada pela aquisição e/ou construção de centros de dados extremamente grandes. Por exemplo, vários centros de dados de computação em nuvem foram construídos em locais aparentemente inesperados, como Quincy, Washington, San Antonio e Texas. A motivação por trás desta escolha são os custos de eletricidade, refrigeração, de trabalho, custo de aquisição de propriedade e os impostos que são geograficamente variáveis.

Assim sendo, uma das características necessárias, mas não suficiente, para uma empresa se poder tonar num fornecedor de computação utilitária é ter orçamento suficiente não apenas para centro de dados enormes, mas também para infraestruturas de software de grande escala e competências operacionais indispensáveis para operá-las. Dadas estas condições, uma

variedade de fatores podem influenciar uma empresa a se tornar num fornecedor de serviços de Nuvem (Armbrust et al. 2009):

1. Ganhar muito dinheiro. Baseado nas estimativas de James Hamilton, Armbrust et al. alegam que grandes centros de dados podem comprar hardware, largura de banda e energia com uma diferença de 1/5 a 1/7 dos preços oferecidos para os centros de dados médios. Além disso, os custos fixos de desenvolvimento e implementação de software podem ser amortizados com mais máquinas. Assim uma empresa suficientemente grande poderia oferecer um serviço bem abaixo dos custos de uma empresa média, gerando ainda um lucro considerável.
2. Aproveitar o investimento. Adicionar serviços de SaaS em cima de uma infraestrutura que existe fornece um novo fluxo de receitas com um baixo custo incremental, ajudando a amortizar os investimentos de grandes centro de dados.
3. Defender um *franchise*[6]. Como as aplicações convencionais abraçam a computação em nuvem, os vendedores com uma ligação de *franchise* estabelecida com essas aplicações vão estar motivados a fornecer a sua própria opção de Nuvem. Por exemplo, Microsoft Azure fornece um caminho facilitado para migrar as aplicações empresariais Microsoft para um ambiente em nuvem.
4. Ser diferenciador. Uma empresa com recursos de centro de dados e de software pode querer estabelecer um produto forte antes que surja uma organização poderosa. Por exemplo, a Google AppEngine fornece uma alternativa para a implementação de nuvens onde rege os seus automatismos de muitas capacidades de escalonamento e balanceamento de carga que, de outra forma, teriam de ser implementadas pelos próprios programadores/empresas.
5. Estimular o relacionamento com os clientes. Organizações de serviços de TI, como a IBM Global Services, têm relações longas com os seus clientes através das suas ofertas de serviços. Proporcionar uma oferta de computação em nuvem dá aos clientes uma oportunidade de migração de livre vontade que permite preservar os investimentos das duas partes na relação.

[6] *Franchising* é um sistema contratual de distribuição de bens e serviços pelo qual uma das partes (o *franchisor*) concede a outra parte (o *franchisee*) o direito de distribuir ou vender certos bens ou serviços. *Franchise* é o privilégio, muitas vezes exclusivo, concedido a um distribuidor ou revendedor por um *franchisor* para vender produtos do *franchisor* dentro de um território específico (American Marketing Association 2012).

6. Tornar o produto numa plataforma. Tal como o Facebook permite que *plug-ins* sejam desenvolvidos e alocados nos seus servidores e disponibilizados na sua aplicação, pode-se usar a nuvem para permitir aos utilizadores a criação dos seus próprios serviços.

O objetivo destas plataformas de computação em nuvem é melhorar a utilização de recursos distribuídos para alcançar e oferecer aos utilizadores-finais um maior rendimento e uma melhor resolução de problemas de computação de grande escala (Rimal, Choi e Lumb 2009). Apesar de todos os benefícios, existem vários problemas que impedem a sua ampla adoção por parte dos fornecedores de serviços de computação em nuvem e consequentemente por parte dos utilizadores-finais (Youseff, Butrico e Da Silva 2008). A Tabela 5 apresenta uma série de obstáculos técnicos que podem ser críticos para o crescimento da computação em nuvem, sendo que cada obstáculo está associado a possíveis soluções.

Tabela 5 – Top 10 dos obstáculos e oportunidades para o crescimento da computação em nuvem (retirado de Armbrust et al. (2009)).

	Obstáculos	**Oportunidades**
1º	Disponibilidade do serviço	Utilizar múltiplos fornecedores de nuvem; Utilizar a elasticidade para prevenir *Distributed Denial-of-Service* (DDoS).
2º	Dados bloqueados	Normalizar APIs; Compatibilizar software para permitir computação hibrida[a].
3º	Confidencialidade e autenticidade dos dados	Implementar encriptação, VLANs e Firewalls.
4º	*Bottlenecks* de transferência de dados	Criar *backup* dos dados; FedExing Disks; Higher BW Switches.
5º	Imprevisibilidade do desempenho	Melhorar o suporte às máquinas virtuais; Flash Memory.
6º	Armazenamento escalável	Inventar armazenamento escalável.
7º	*Bugs* em sistemas distribuídos de grande escala	Inventar *debuggers* que se baseiem em máquinas virtuais distribuídas.
8º	Dimensionamento rápido	Inventar auto dimensionamento que se baseie em *programming language* (ML); *Snapshots* para conservação.
9º	Reputação de *fate sharing*[b]	Oferecer serviços que protegem a reputação como têm os serviços de emails.
10º	Licenciamento de software	Licenças *pay-for-use*;

[a] Computação hibrida permite que os programadores enviem apenas os dados suficientes para a nuvem e esta devolva um resultado aceitável, ou semelhantemente permite o carregamento de recursos de outras nuvens quando a capacidade local está temporariamente excedida (Rean 2009).

[b] Reputação de *fate sharing* refere que o comportamento de um único cliente de nuvem pode afetar a reputação dos outros clientes. Por exemplo, se forem apreendidos computadores de clientes inocentes pelas autoridades (Fox 2009).

Devido a estes obstáculos e às limitações inevitáveis em serviços de computação em nuvem, os fornecedores destes serviços são obrigados a estabelecerem *Service Level Agreement* (SLA) que indiquem o nível de serviço oferecido através de parâmetros de qualidade de serviço (QoS) (Buyya, Yeo e Venugopal 2008).

Como os dados, as aplicações e os recursos de computação deixam de estar sob o controlo do utilizador-final, os utilizadores-finais precisam de exigir aos fornecedores de serviços de computação em nuvem ferramentas de transparência e mecanismos de monitorização e controlo sobre o estado dos seus dados, das suas aplicações e dos seus recursos de computação (Rimal, Choi e Lumb 2009).

Problemas de segurança (confidencialidade, disponibilidade e integridade) e privacidade são as preocupações mais importantes para os utilizadores-finais de computação em nuvem (Rimal, Choi e Lumb 2009). Antes que os fornecedores possam persuadir os fornecedores de SaaS e os utilizadores-finais a migrarem as suas aplicações, dados e recursos para a nuvem, os fornecedores de computação utilitária e/ou os fornecedores de SaaS precisam de abordar as preocupações dos utilizadores-finais e fornecer SLA confiáveis para os seus serviços em nuvem (Youseff, Butrico e Da Silva 2008).

Atualmente e devido ao espantoso crescimento dos serviços da Web, muitas empresas multinacionais já têm infraestruturas de software escaláveis e experiência suficiente para blindar os seus centros de dados contra potenciais ataques físicos e eletrónicos (Armbrust et al. 2009), garantindo assim melhores níveis de serviço de computação em nuvem. Para além disso, estas empresas também começaram a estabelecer novos centros de dados que permite hospedar futuras aplicações de computação em nuvem em diversos locais no mundo de forma a fornecer redundância e garantir a segurança em caso de falhas dos serviços (Buyya, Yeo e Venugopal 2008).

De forma a ajudar os fornecedores de serviços SaaS na escolha dos serviços de computação utilitária mais adequado, tanto a taxonomia de Rimal, Choi, e Lumb (2009) como as classes de Armbrust et al. (2009) poderão ser utilizadas.

Rimal, Choi, e Lumb (2009) desenvolveram uma taxonomia baseada nas ideias principais dos sistemas distribuídos para processamento de grandes quantidades de dados. Esta taxonomia foca sobre a arquitetura da nuvem, a gestão de virtualização, os serviços, a tolerância a falhas, o balanceamento de carga, a interoperabilidade e a escalabilidade do armazenamento de dados e

pode ajudar a identificar as forças, fraquezas e desafios técnicos dos serviços de computação utilitária.

Outra forma possível pode passar pela utilização das classes de computação utilitária definidas por Armbrust et al. (2009). Como qualquer aplicação precisa de um modelo de computação, um modelo de armazenamento e, assumindo que é trivialmente distribuída, um modelo de comunicação as três classes são: modelo de computação (maquinas virtuais), modelo de armazenamento e modelo de rede. Os autores acreditam que as ofertas de computação utilitária distinguem-se com base no nível de abstração apresentada ao utilizador-final (programador) e no nível de gestão dos recursos.

2.4.2 Serviços de Aplicação como um Serviço

A popularidade do conceito de computação em nuvem e a sua incorreta utilização não gerou apenas uma ideia errada sobre o seu propósito real (secção 2.2), mas também uma ideia errada de outros conceitos relacionados, como por exemplo o conceito de *Software as a Service* (SaaS).

Existem vários conceitos que parecem similares à computação em nuvem, mas, na verdade, são complementares a este, como é o caso do conceito SaaS. Se estes conceitos forem comparados, fica claro que a computação em nuvem é um elemento essencial para que um fornecedor consiga entregar uma oferta de SaaS escalável (Plummer et al. 2008).

SaaS, por vezes referido como *Application Service Provider* (ASP) ou *on-demand software* (Chou e Chou 2007), foi anunciado como a nova tendência para a distribuição de software (SIIA 2000), a próxima geração de ASP (Sääksjärvi, Lassila e Nordström 2005) e o modelo que estende a ideia do modelo ASP (Pathak 2012).

O modelo ASP surgiu na década de 90 com a expansão da Internet e como uma nova forma de computação centralizada. As empresas fornecedoras de software começaram a fornecer um serviço de alojamento e gestão de aplicações de negócios, com o objetivo de reduzir os custos através de uma administração central e através da especialização do fornecedor para soluções numa área de negócio em particular (Pathak 2012).

Apesar das semelhanças entre SaaS e ASP, segundo Chou e Chou (2007), podem existir diferenças no modelo de entrega, nas taxas de serviço e nas receitas. Contudo, e talvez por isso,

o modelo SaaS pode ser mais utilizado em determinados contextos do que em outros (Pathak 2012).

Na computação em nuvem, SaaS é considerado como parte integrante da sua nomenclatura (Pathak 2012) e refere-se mais precisamente à camada de Aplicação (como um serviço) que engloba os modelos de serviços SaaS, PaaS, IaaS, DaaS e CaaS (Youseff, Butrico e Da Silva 2008).

As definições de SaaS estão constantemente a mudar devido à criação constante de novos modelos de negócios e tecnologias que as empresas de TI utilizam para entregar a sua visão de SaaS (SIIA 2000). Por exemplo, o mercado é inundado com siglas em que cada uma representa uma abordagem um pouco diferente: ASP, *Application Infrastructure Providers* (AIPs) *Internet Business Service* (IBS), *Business Service Provider* (BSP), *Solutions Service Provider* (SSP), entre outros. Contudo, o entendimento comum sobre o conceito de SaaS amadureceu durante os últimos anos (Stuckenberg, Fielt e Loser 2011).

De acordo com Sääksjärvi, Lassila e Nordström (2005), vários artigos descrevem SaaS como uma forma nova e amigável para o cliente realizar *outsourcing* de TI, onde o fornecedor irá possuir tanto o software como a infraestrutura de TI necessária para o serviço online e onde ambas as partes irão beneficiar da lógica de receita simples e atraentes, uma vez que a lógica deixa de ser baseada no investimento de desenvolvimento de aplicações.

Todavia, e apesar das diferentes definições existentes sobre o conceito SaaS, Stuckenberg, Fielt e Loser (2011) consideram, com base no estudo de Mäkilä et al. (2010), que os critérios-chave do conceito são compartilhados. Assim, e em geral, SaaS pode ser visto como um modelo de distribuição de software de fornecedores de software que se diferencia das abordagens tradicionais, como *on-premises software*[7], em três áreas (Stuckenberg, Fielt e Loser 2011):

- Propriedade de serviço: Os serviços podem ser definidos através de três critérios: imaterialidade, *uno-actu-principle*[8] e a existência de um fator externo na fase do cumprimento, realização ou entrega. O conceito SaaS está em conformidade com estes

[7] *On-premises software* ou *shrink-wrapped software* refere-se ao *software* que é instalado e executado em computadores no local da pessoa ou organização que utiliza o *software* (Wikipedia contributors 2013).
[8] *Uno-actu-principle*, é uma característica dos serviços que indica que a produção e consumo coincidem no tempo (Wikipedia contributors 2012).

três critérios, resultando num relacionamento (serviço) contínuo entre o fornecedor de SaaS e o cliente.

- Modelo de implementação: O modelo de implementação alterado é uma consequência direta da propriedade de serviço. As soluções SaaS são operadas/exploradas e mantidas pelo fornecedor do software. O fornecedor assume atividades que anteriormente viviam sob a responsabilidade do cliente e estende a sua área de responsabilidade, além das atividades de desenvolvimento para as operações/exploração e manutenção. As características típicas e implementações técnicas ligadas a este modelo é a utilização de uma arquitetura *multi-tenancy*[9], *web browsers* para aceder ao software, e a adoção de arquiteturas orientadas a serviços.
- Modelo de atribuição de preços: este terceiro ponto de diferenciação baseia-se na relação de serviço contínuo entre o cliente e o fornecedor. Ao contrário do software tradicional, a definição de preços para os serviços SaaS é baseada em métricas dependentes do tempo ou utilização. A manutenção contínua é cobrada previamente e as taxas de suporte são, em geral, incorporadas numa taxa de inscrição/subscrição. O dever de pagamento persiste enquanto uma empresa está inscrita no serviço.

Para este tipo de serviços, como para qualquer outro, existem pontos positivos e negativos tanto para o fornecedor (SaaS) como para o consumidor (utilizador-final). Sääksjärvi, Lassila e Nordström (2005) realizaram um estudo que resultou numa sintetize dos benefícios e dos riscos, presentes em vários artigos e relatórios, para os principais *stakeholders* do conceito de SaaS. Este estudo aponta que a literatura parece ser demasiado otimista para o utilizadores-final, quando comparados os benefícios e os riscos de ambos. Porém, muitos dos benefícios para os utilizadores só serão efetivos se os fornecedores de SaaS conseguirem superar determinados riscos. Estes riscos são os seguintes (os quatro primeiros foram os mais mencionados):

- Dificuldade em gerir a complexa rede de fornecedores, que é necessária para integrar os negócios de produtos e serviços;
- Mudar para o modelo de SaaS reduz o volume de receitas iniciais, em vez de vendas de licenças e taxas de manutenção tem-se taxas de serviços;

[9] *Multi-tenancy* refere-se a um princípio da arquitetura de *software*, onde uma única instância de *software* é executada num servidor, servindo múltiplas organizações de clientes (*tenants*). *Multi-tenancy* contrasta com uma arquitetura *multi-instance* onde independentes instâncias de *software* (ou sistemas de hardware) são configuradas para organizações clientes diferentes (Wikipedia contributors 2013).

- Possíveis problemas de desempenho e escalabilidade que depende da solução técnica utilizada;
- Investimentos elevados para iniciar o negócio SaaS: construir e manter a infraestrutura de TI necessária e comprar software;
- A personalização de aplicações SaaS normalmente requer custos adicionais;
- Requer compromisso com um ciclo mais frequente de atualizações e novas versões de software.

Portanto, para o sucesso do modelo SaaS, segundo Sääksjärvi, Lassila e Nordström (2005), é necessário, tanto uma rede eficaz de fornecedores como uma lógica justa de receitas inovadoras, para permitir um fluxo contínuo de inovações de software partilhado como a transformação do negócio de produtos de software em negócio de serviços de software. Para Wu, Garg e Buyya (2011), o que estas empresas precisam é a realização de esforços para obter lucro a partir da margem entre o custo operacional da infraestrutura e da receita gerada pelos clientes, sem afetar o cumprimento de um nível de serviço mínimo para os utilizadores-finais.

Uma ajuda para alcançar estes requisitos pode ser o recurso a fornecedores de computação utilitária (Armbrust et al. 2010). Com os serviços de computação utilitária os benefícios possíveis para os fornecedores de SaaS e utilizadores-finais não se alteram, mas dá mais opções aos fornecedores de software de implementarem o seu produto como SaaS sem terem que configurar um centro de dados. Assim, da mesma forma que os SaaS permitem aos utilizadores-finais transferirem alguns problemas para os fornecedores de SaaS, o fornecedor de SaaS pode agora transferir também alguns desses problemas para os fornecedores de computação utilitária. Assim, as empresas, que pretendam fornecer serviços de SaaS a clientes, podem decidir manter o seu próprio centro de dados ou alugá-lo a partir de fornecedores de computação utilitária.

Ultrapassadas estas barreiras e riscos, existem várias razões para um empresa se tornar fornecedora de SaaS. De acordo com Sääksjärvi, Lassila e Nordström (2005), os benefícios podem ser os seguintes:

- Economia de escala tanto para os custos de produção como para os custos de distribuição;
- Fluxos de receita mais previsíveis do que nas vendas de licenças de software tradicionais;

- Expansão de número de clientes potencial;
- Reduzido ciclo de vendas do que o ciclo tradicional;
- Redução de custos de gestão de versões e de manutenção de software;
- Obtenção de uma forte vantagem competitiva, se o fornecedor conseguir integrar bem os produtos e serviços numa oferta de SaaS.

De acordo com Armbrust et al. (2009), as vantagens de SaaS para os fornecedores de SaaS estão bem definidas. Os fornecedores de SaaS beneficiam de uma maior simplicidade de instalação, manutenção de software e controlo de versões.

Relativamente aos utilizadores-finais, existem vários autores a tentar enumerar as vantagens e desvantagens da utilização de serviços SaaS (SIIA 2000; Miller 2008; Grossman 2009), sendo que alguns deles parecem querer demonstrar que existem mais vantagens que desvantagens (Sääksjärvi, Lassila e Nordström 2005), e que as desvantagens existentes não são razão suficiente para que uma organização não adote estes serviços, pois são facilmente ultrapassadas (Armbrust et al. 2009).

Seguindo o estudo de Sääksjärvi, Lassila e Nordström (2005), é possível verificar doze benefícios significativos para os utilizadores-finais. Alguns destes benefícios ([1, 3]) são benefícios tradicionais do *outsourcing* de TI, outros ([4, 6] e [10, 12]) estão mais diretamente relacionados com o modelo SaaS e os restantes ([7, 9]) referem-se à forma única (em principio) para obter acesso ao software:

1. SaaS permite que o cliente se concentre mais nas suas competências essenciais;
2. SaaS facilita o acesso a conhecimento especializado;
3. SaaS permite um mais amplo e flexível método de pagamento (custos previsíveis e/ou inferiores);
4. SaaS proporciona um menor tempo de implementação da aplicação;
5. SaaS facilita a gestão de versões de software (atualizações gratuitas, a tecnologia não fica obsoleta, etc.);
6. Possibilidade de obter do mesmo fornecedor um pacote agregado de software de várias fontes tornando a oferta de serviços mais completa.
7. SaaS permite que o cliente tenha acesso a aplicações que eram demasiado caras para serem compradas;
8. SaaS torna possível aceder ao software independentemente da localização e da hora;

9. Com SaaS os investimentos e os custos iniciais são muito mais baixos;
10. Com SaaS, o cliente pode ter acesso a uma infraestrutura de TI melhor em aspetos de segurança e escalabilidade;
11. SaaS amplia a seleção de potenciais aplicações disponíveis para o cliente;
12. SaaS aumenta as opções de personalização de aplicação para o cliente.

Porém, e como se verificou anteriormente, estes benefícios podem não ser facilmente percetíveis para os utilizadores-finais porque alguns destes benefícios podem constituir grandes riscos para os fornecedores de SaaS. Para além disso, os utilizadores-finais precisam de ter em atenção riscos associados à utilização de SaaS:

- Menor opção de adaptação/personalização da aplicação e integração de aplicações;
- Maior probabilidade para perder dados críticos de negócios;
- Maior probabilidade para obter problemas relacionados com o desempenho do serviço;
- Em troca de preços menores, o cliente normalmente compromete-se a um contrato de longo termo (custos de mudança).

Embora SaaS possa parecer uma vantagem clara para a utilização do produto por meio de um modelo de serviço, este pode não ser o caso para todas as aplicações (SIIA 2000). Os utilizadores-finais devem analisar as suas necessidades atuais e futuras, assim como a estrutura de preços que os fornecedores oferecem com o objetivo de encontrar a melhor solução para as suas necessidades de TI. Os fornecedores de SaaS devem ter em conta todos os custos diretos e indiretos na prestação de um serviço com preço justo e valor verdadeiro para o cliente.

2.5 Conclusões

De forma a clarificar a relevância deste trabalho de investigação, esta secção apresenta uma informação estruturada, resultante de uma revisão de literatura, que evidencia a relevância do tema.

A primeira secção inicia com a apresentação de três grandes áreas do conhecimento, Computação, Gestão de Organizações e Gestão de Sistemas de Informação, e delimita a área de estudo desta dissertação.

Invocar as origens da Computação permitiu não só identificar as limitações inerentes às tecnologias de computação, mas também destacar a importância das tecnologias de

computação na vida das pessoas. Para além disso, também foi possível constatar que, hoje em dia, a computação pode ser observada de formas diferentes com propósitos diferentes, daí existir um conjunto de disciplinas relacionadas com a computação. Para concluir, foram apresentadas quais as disciplinas, Tecnologias de Informação, Sistemas de Informação e Engenharia de Software, que tiveram uma maior ênfase na realização desta dissertação.

Identificar alguns aspetos da estrutura e do funcionamento de uma organização, em qualquer um dos seus níveis de gestão, possibilitou destacar a complexidade das atividades dos gestores, de forma a contribuírem para a sustentabilidade e melhoria das suas organizações. Verificou-se também que a conjuntura atual obriga as empresas a produzirem e a criarem continuamente produtos inovadores, representando assim uma necessidade para as organizações. Terminou com a indicação que nesta dissertação se estuda a possibilidade estratégica das organizações iniciarem esforços para adoção do modelo de computação em nuvem.

Reconhecer a complexidade inerente das atividades de Gestão de Sistemas de Informação, permitiu verificar que estas atividades são fatores fundamentais para alcançar o valor esperado da utilização das TI. Foi também referido que a atividade de Desenvolvimento de Sistemas de Informação é a que tem uma maior enfâse nesta dissertação, uma vez que se estudou como uma organização fornecedora de serviços e recursos de TI pode conceber uma solução de TI baseada no modelo de computação em nuvem com o propósito de melhorar o SI do consumidor.

A segunda secção apresenta a origem, os principais aspetos e o estado atual do principal objeto em estudo, a Computação em Nuvem. Nesta secção, foi possível verificar que, hoje em dia, existe a ideia errada sobre o real propósito do conceito de computação em nuvem e que existem várias definições e representações para o conceito de computação em nuvem, onde umas focam mais aspetos técnicos em detrimento dos aspetos de negócio, e vice-versa. Contudo, e no âmbito desta dissertação, foi escolhida uma definição que engloba/foca todos os aspetos de computação em nuvem em geral e foi apresentada uma representação do conceito de computação em nuvem. Para terminar, foram exibidos dados que comprovam a pertinência para continuar a estudar este objeto.

A terceira e última secção, permitiu fazer o afunilamento relativamente ao problema que se pretende resolver no âmbito desta dissertação e destacar as principais oportunidades e desafios que tanto os fornecedores de recursos e serviços de TI como os utilizadores-finais de soluções de

computação em nuvem deverão ter que considerar antes da sua adoção. Verificou-se que na computação em nuvem podem ser identificados dois tipos de serviços diferentes e que reconhecer os principais *trade-offs* entre os *stakeholders* destes serviços é um fator essencial para alcançar o benefício coletivo da utilização do modelo de computação em nuvem.

A consideração de muitos dos fatores aqui apresentados revelou-se importante, para a análise e conceção de uma componente de SI baseada no modelo de computação em nuvem para o suporte a SLT, que poderá ser útil tanto para os fornecedores de SaaS como para os utilizadores-finais. Todo o processo que permitiu alcançar esta componente de SI é apresentado nós próximos dois capítulos.

CAPÍTULO 3 – ABORDAGENS DE ENGENHARIA DE REQUISITOS PARA SISTEMAS LOGÍSTICOS DE TRANSPORTES

3.1 Introdução

Como se verificou no capítulo anterior, os benefícios do modelo de computação em nuvem podem ser vários, tanto para o fornecedor como para o consumidor. No entanto, conceber uma componente de sistema de informação (i.e. um sistema informático ou aplicação informática) para esse contexto pode ser uma tarefa complicada.

Consciente deste facto e de que a atividade de Desenvolvimento de Sistemas de Informação (DSI) é uma atividade fundamental para a obtenção e/ou construção de componentes de sistema de informação, decidiu-se seguir as sugestões de Carvalho (1996) para iniciar o processo de construção de uma solução informática para o domínio dos SLT.

Segundo Carvalho (1996), o processo de desenvolvimento de DSI passa por três fases. Na primeira (perceção) os agentes que conduzem o processo procuram compreender o universo com que são confrontados, identificar, delimitar e formular o problema. Na segunda (conceção) são geradas e avaliadas soluções alternativas e é feita a escolha da solução que parece mais apropriada. Na última (implementação) serão executados os planos que conduzirão à implementação da solução escolhida.

Contudo, o termo desenvolvimento de sistemas de informação pode ter três interpretações diferentes e o foco da atenção em cada uma das fases do processo deve ser diferente (Carvalho 1996). A construção de sistemas informáticos (CSI), desenvolvimento de sistemas de informação (DSI) e redefinição de processos organizacionais (RPO) são as três diferentes interpretações de DSI. E cada uma delas introduz mudança a níveis organizacionais diferentes: organização, sistema de informação ou sistema informático.

Como resultado da análise às sugestões de Carvalho (1996), foi decidido seguir o processo de CSI, uma vez que corresponde a uma visão técnica desenvolvida em áreas próximas da engenharia de software e uma vez que vai de encontro com o que é pretendido realizar nesta dissertação, um sistema informático baseado no modelo de computação em nuvem para o suporte a SLT. A Figura 13 procura ilustrar este processo de CSI.

Nesta perspetiva (CSI), a intervenção é feita essencialmente ao nível dos sistemas informáticos e o principal resultado é uma aplicação informática. Inicia com a especificação dos requisitos do sistema informático na primeira fase. De seguida, as aplicações informáticas são concebidas, especificadas e implementadas. Terminada a implementação, as aplicações informáticas são disponibilizadas para integração no sistema de informação da organização. Esta integração, no entanto, não é normalmente contemplada nos modelos do processo de CSI (Carvalho 1996).

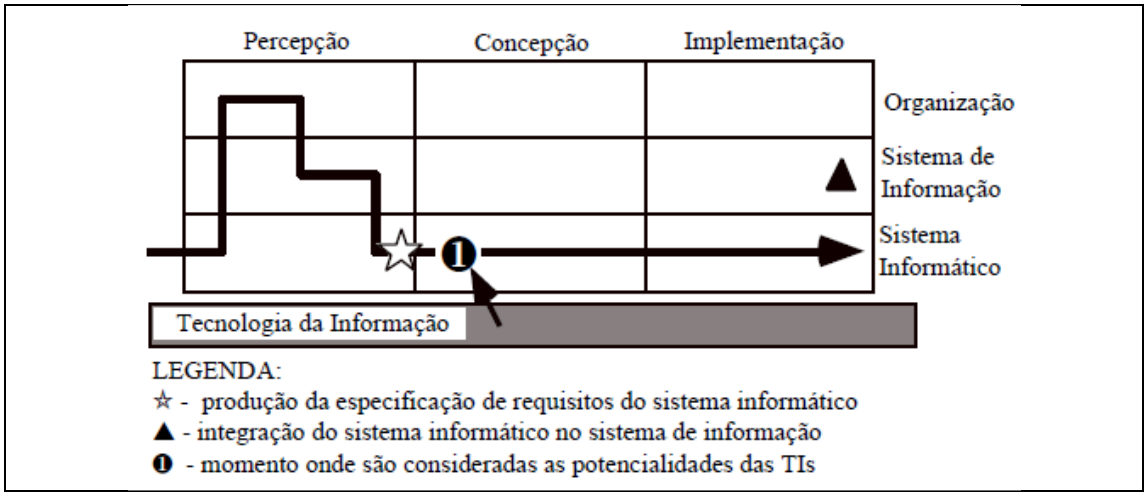


Figura 13 – Construção de sistemas informáticos ou aplicações informáticas (retirado de Carvalho (1996)).

Devido à complexidade que normalmente está associada à realização das atividades do processo de CSI, o principal resultado do mesmo, uma aplicação informática, não foi alcançado nesta dissertação, impossibilitando a realização total do terceiro objetivo desta dissertação. No entanto, para o domínio dos SLT, foram especificados os requisitos de um sistema informático de nível de processo e os requisitos associados ao modelo de computação em nuvem. Ou seja, foi construída uma base inicial que poderá permitir a realização do resultado principal do processo CSI.

Se os resultados alcançados forem mapeados com o processo de CSI, da Figura 15, pode ser mencionado que foi elaborada uma parte considerável da primeira fase (perceção) do processo de CSI. Mas antes de apresentar a forma como foram alcançados esses resultados, foi essencial ainda compreender a primeira fase (perceção) do processo de CSI.

A primeira fase (perceção) do processo de CSI é considerada como uma atividade autónoma, designada normalmente por análise de sistemas ou por engenharia de requisitos, crucial para o sucesso dos sistemas informáticos, uma vez que os erros cometidos nesta fase são responsáveis por grande parte das correções, muitas delas onerosas, a efetuar ao sistema informático (Carvalho 1996).

O principal objetivo passa por compreender as necessidades da organização envolvendo as três atividades seguintes. A primeira é a análise do modo como a organização funciona. A segunda é a análise do modo como o sistema de informação suporta a organização. E a terceira é a indagação junto dos futuros utilizadores do sistema informático que suporte é que pretendem ter por parte das aplicações informáticas (Carvalho 1996). O principal resultado é uma especificação dos requisitos do sistema informático de nível de produto, no entanto, e como já foi referido anteriormente, o resultado alcançado foi uma especificação dos requisitos do sistema informático de nível de processo.

Assim, e nunca negligenciando as sugestões de Carvalho (1996), foi definida uma abordagem de três fases para a conceção de um sistema de software baseado no modelo de computação em nuvem e alinhado com o domínio dos SLT. A primeira fase compreende a identificação, delimitação e formulação do problema das organizações com SLT (capítulo 3, secção 3.2). A segunda fase envolve a seleção de abordagens adequadas de engenharia de requisitos para a conceção de um sistema de software que resolva o problema dos SLT (capítulo 3, secção 3.3 e 3.4). A terceira fase consiste na utilização das abordagens selecionadas de engenharia de requisitos para a conceção do sistema de software pretendido (capítulo 4).

Este capítulo inicia o processo de construção de um sistema de informação baseado no modelo de computação em nuvem para o suporte a SLT. Na secção 3.2 é apresentado um estudo sobre as organizações com processos de SLT. Na secção 3.3, é apresentado um estudo de abordagens de requisitos que podem ajudar a resolver as necessidades das organizações com SLT e é selecionada uma dessas abordagens. Na secção 3.4, é descrito um modelo para a identificação de requisitos de computação em nuvem. Por fim, na secção 3.5, são apresentadas as ideias principais debatidas nas secções anteriores.

3.2 Sistemas Logísticos de Transportes (SLT)

De modo a identificar, delimitar e formular o problema das organizações com SLT, foi realizado um estudo sobre os conceitos relacionados com o termo sistema logístico de transportes e sobre o estado atual do mercado para sistemas como aquele que é estudado nesta dissertação.

No âmbito desta dissertação, um sistema logístico de transportes refere-se a todas as atividades da gestão da cadeia de abastecimento de um nó (i.e. centros de produção ou centros de

distribuição) relacionadas com o transporte de materiais. Este sistema, para além de incluir os processos produtivos e os fluxos de informação necessários para um "sistema de gestão de cadeia de abastecimento" monitorizar todas as atividades de um nó, também inclui o planeamento detalhado dos fluxos de materiais dentro de um nó (normalmente a logística não está normalmente associada ao planeamento detalhado dos fluxos de materiais de um centro de produção ou montagem (Ghiani, Laporte e Musmanno 2004)).

Os conceitos que estão diretamente relacionados com o termo SLT são: o conceito de logística, sistema de logística, de cadeia de abastecimento, de gestão logística e de gestão de cadeira de abastecimento.

Sem pretender tecer considerações detalhadas, pode-se descrever logística como "o processo de gestão estratégica da aquisição, movimentação e armazenamento de materiais, inventário de produtos de produção e produtos acabados (e os fluxos de informação relacionados) através da organização e dos seus canais de marketing [distribuição] de tal forma que a rentabilidade atual e futura seja maximizada através da relação custo-benefício da realização de encomendas" (Christopher 2011, p. 2).

O objetivo de gerir as atividades de logística passa por ligar o mercado com as atividades operacionais do negócio de modo a que os clientes tenham melhores serviços a um custo menor (Karpio, Orłowski e Przybyć 2010).

Um sistema de logística é um conjunto de *facilities*[10] ligadas por serviços de transporte[11] e um conjunto de três atividades principais: processamento de encomendas, gestão de inventários e gestão de transporte (Ghiani, Laporte e Musmanno 2004).

Por sua vez, a "cadeia de abastecimento" pode ser considerado como um "sistema de logística complexo" (Ghiani, Laporte e Musmanno 2004, p. 3) e definido como uma "rede de organizações ligadas e interdependentes que mutuamente e cooperativamente trabalham em conjunto para controlar, gerir e melhorar o fluxo de materiais e informação desde os fornecedores aos utilizadores-finais" (Christopher 2011, p. 4).

---

[10] *Facilities* são locais onde os materiais são, por exemplo, fabricados, armazenados, classificados, vendidos ou consumidos e inclui centros de produção e montagem, armazéns, centros de distribuição, terminais de transporte, lojas, centros de triagem, lixeiras, entre outros (Ghiani, Laporte e Musmanno 2004, p. 2).

[11] Serviço de transporte é considerado o que move materiais entre as *facilities* utilizando viaturas e equipamentos, como camiões, tratores, reboques, paletes, contentores, carros e comboios (Ghiani, Laporte e Musmanno 2004, p. 2).

De forma a facilitar a compreensão de "cadeia de abastecimento", a Figura 14 apresenta uma "cadeia de abastecimento" típica, no qual os sistemas de produção e distribuição são constituídos por duas fases cada. No sistema de produção, componentes e peças semiacabados são produzidas em dois centros de produção, enquanto os produtos acabados são montados numa instalação diferente. O sistema de distribuição consiste em dois centros de distribuição centrais (CDC) fornecidos diretamente pelo centro de montagem, que por sua vez reabastecem dois centros de distribuição regionais (RDC). Cada uma das ligações de transporte pode ser uma linha de transporte simples (por exemplo, uma linha de camiões) ou um processo mais complexo de transporte que envolve instalações adicionais (ex. terminais portuários) e empresas (ex. transportadoras de camiões) (Ghiani, Laporte e Musmanno 2004).

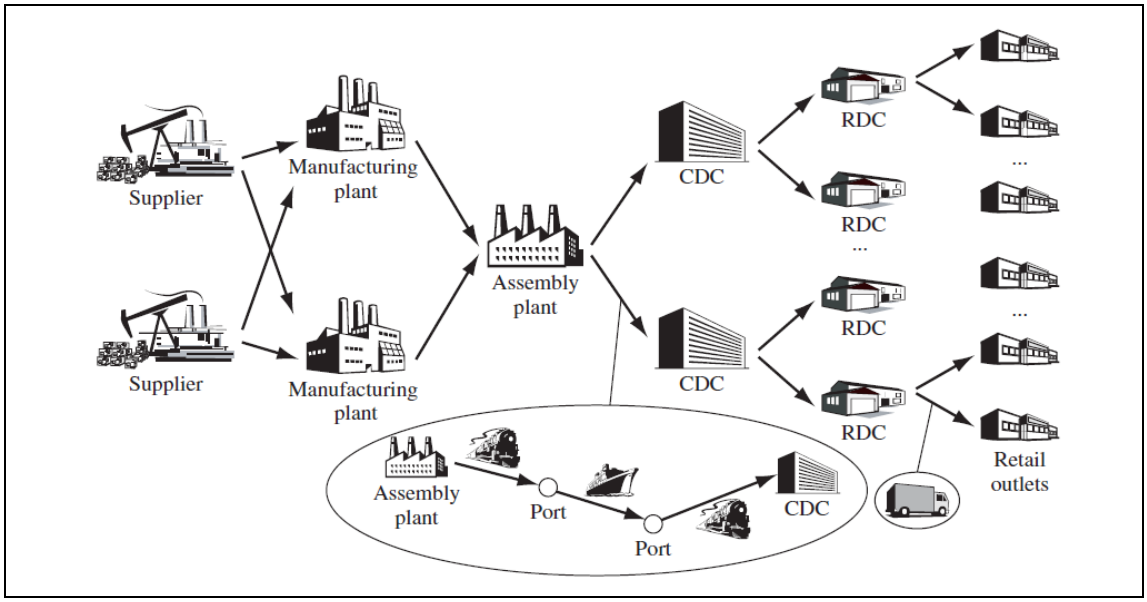


Figura 14 – Uma cadeia de abastecimento (retirado de Ghiani, Laporte e Musmanno (2004)).

Relativamente ao conceito "gestão logística" e "gestão da cadeia de abastecimento" é possível identificar algumas incongruências entre diferentes autores, contudo o conceito de "gestão logística" é amplamente considerado como um subconjunto da "gestão da cadeia de abastecimento" (Sweeney 2005).

Uma das definições frequentemente utilizadas para "gestão logística" e "gestão da cadeia de abastecimento" são as seguintes:

- Gestão Logística: "é a parte da gestão da cadeia de abastecimento responsável pelo planeamento, implementação e controlo, de forma eficiente e eficaz, dos fluxos diretos e inversos e o armazenamento de produtos e toda a informação associada, desde o ponto

de origem ao ponto de consumo, de forma a satisfazer os requisitos do serviço a clientes" (Council of Supply Chain Management Professionals 2013).

- Gestão da cadeia de abastecimento: "engloba o planeamento e a gestão de todas as atividades envolvidas na aquisição e distribuição, conversão, e todas as atividades da gestão de logística. Inclui a coordenação e a colaboração com os parceiros, que podem ser fornecedores, intermediários, prestadores de serviços e clientes. Em essência, a gestão da cadeia de abastecimento integra a gestão do fornecimento e da procura interna e através de empresas" (Council of Supply Chain Management Professionals 2013).

A logística é geralmente vista no domínio de uma empresa, ainda que trate fluxos com o exterior (empresas fornecedoras e clientes). A gestão da cadeia de abastecimento inclui não só os fluxos de logística, mas também a gestão de encomendas de clientes, os processos de produção e os fluxos de informação necessários para monitorizar todas as atividades de cada nó da cadeia de abastecimento (Sweeney 2005).

Para melhorar os sistemas de logística, nos últimos anos, uma grande quantidade de ferramentas de software foram desenvolvidas (Ghiani, Laporte e Musmanno 2004). Isto deve-se provavelmente à forma como hoje em dia as organizações encaram as atividades da logística e os seus custos associados.

A logística deixou de ser uma questão militar para ser uma das preocupações fundamentais das organizações de todas indústrias e de todos os setores (Karpio, Orłowski e Przybyć 2010). E atualmente quase todas as organizações enfrentam o problema de conseguir os materiais certos, no local certo e à hora certa. Facto que está a tornar cada vez mais imperativa a gestão eficiente de sistemas logísticos (Ghiani, Laporte e Musmanno 2004).

Para ter uma melhor noção da significância dos custos normalmente associados à logística, é pertinente a apresentação do relatório elaborado por Council of Supply Chain Management Professionals (Wilson 2012) para os Estados Unidos da América.

Os dados deste relatório indicam que, nos últimos anos, a evolução dos custos associados à logística é inconstante, conforme também se pode observar na Figura 15. No entanto continua a

ser considerada uma quantidade significativa. Em 2011 atingiu o valor de $1.28 bilhões, $79 mil milhões acima de 2010, representando cerca de 8.5% do Gross Domestic Product (GDP)[12].

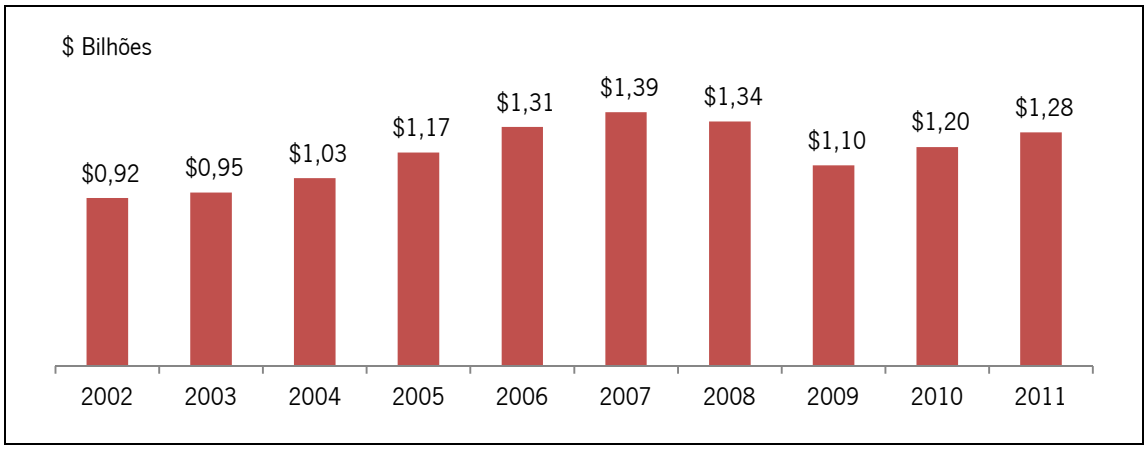


Figura 15 – Custos de negócio da logística dos Estados Unidos da América (retirado de (Wilson 2012)).

Sobre os custos de transporte, estes subiram 6.2% em 2011 por causa de taxas de transporte e representam a maior quantia comparativamente com os restantes custos associados à logística. O maior segmento de transportes, segundo este relatório, é o segmento de transportes rodoviários que compreende cerca de 77% do mercado.

Sendo que as TI podem ter um forte impacto sobre a logística (Ghiani, Laporte e Musmanno 2004) e que os seus custos ainda continuam significativos (Wilson 2012), considera-se que ainda existe espaço para a criação de novas soluções de TI e/ou melhoria das soluções existentes para os sistemas logísticos e em particular para os sistemas logísticos de transportes.

De acordo com Gartner Group (2012), algumas destas soluções de TI para o suporte a sistemas logísticos já são entregues como um serviço. Segundo Gartner Group (2012), os serviços de nuvem que são e continuarão a ser o maior segmento de mercado (subsecção 2.3.4) correspondem aos serviços provenientes de implementações exclusivas orientadas para indústrias específicas, designados por *Business Process as a Services* (BPaaS). Esta sigla representa uma abordagem diferente para a entrega de SaaS (subsecção 2.4.2) e inclui soluções de Customer Management, Finance and Accounting, Supply Chain Management, entre outros.

Os serviços associados ao BPaaS Supply Chain Management (SCM), que o relatório apresenta, parecem ser os que mais se adequam às soluções de sistemas logísticos de transportes que esta dissertação trata. Apesar de esta ser um dos tipos de BPaaS com previsões de volume de negócio menores, comparativamente com os restantes conforme se ilustra na Figura 16, esta

[12] Gross Domestic Product (GDP) é o equivalente em Portugal ao Produto Interno Bruto (PIB).

tem sido amplamente adotado para determinadas funções dentro da ampla definição de SCM. Em pormenor, o relatório indica que BPaaS SCM é mais generalizado no processo de logística do que nos processos de distribuição e aquisição (Gartner Group 2012).

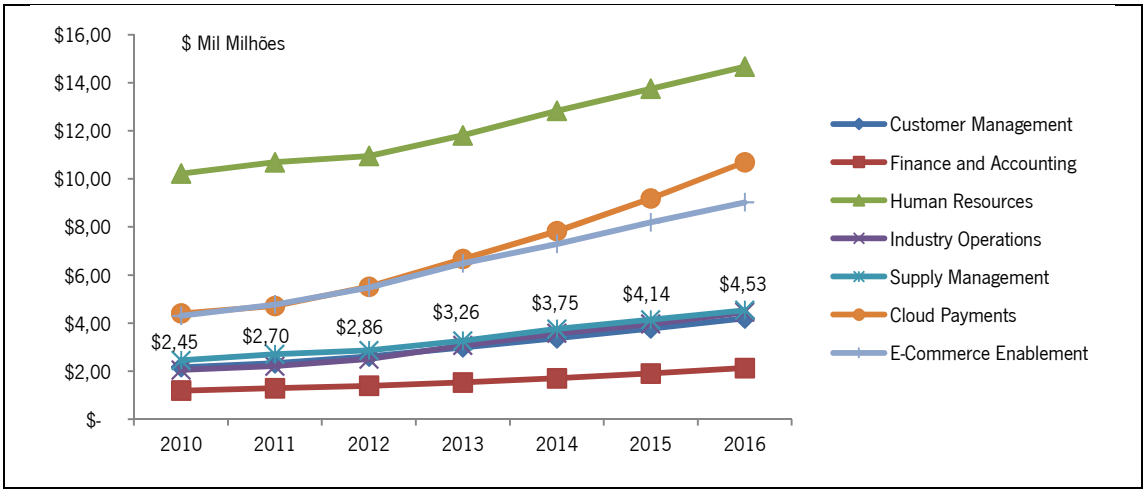


Figura 16 – Tamanho do mercado de serviços BPaaS, exceto o serviço Cloud Advertising (retirado de (Gartner Group 2012)).

Estes dados comprovam a tendência que as empresas fornecedoras de serviços e recursos de TI preferem, relativamente às ofertas de SaaS, apostar em sistemas mais complexos, em detrimento de sistemas mais simples e de uso pessoal. Já em 2000, segundo SIIA (2000), o mercado sobre as ofertas de SaaS indicava que a maioria dos fornecedores, como a SAP e a Oracle, procuravam oferecer aplicações de SaaS complexas como os Enterprise Resource Planning (ERP), Customer Relationship Management (CRM), Supply Chain Management (SCM), Decision Support Systems (DSS), entre outros. Poucos fornecedores eram os que se focavam em serviços de TI relativamente simples e para uso pessoal.

Assim, e tendo em consideração todos estes fatores, julga-se que muitas das necessidades, das organizações com processos de SLT, poderão, de facto, ser resolvidas com a criação ou melhoria de soluções de software entregue como um serviço. Uma solução a ser desenvolvida para os SLT (capítulo 4) seria uma que automatizasse os processos de negócio relacionados com o transporte de mercadorias de um nó e integrasse estes processos com outros desse nó e da cadeia de abastecimento. O seu objetivo seria tornar mais eficiente e eficaz tanto a logística interna (de um nó) como toda a cadeia de abastecimento.

3.3 Abordagens de Formalização de Requisitos

Para satisfazer as principais necessidades dos utilizadores das organizações da cadeia de abastecimento com processos de SLT (secção 3.2) através de uma solução de software entregue como um serviço, é necessário primeiro definir um conjunto adequado de requisitos para essa solução. Assim sendo, e de modo a alcançar uma especificação da solução de software (i.e. um documento que contém uma descrição completa do que o software vai fazer, independente dos detalhes de implementação (Kotonya e Sommerville 1996)), considerou-se que era importante a realização de um estudo sobre a área da engenharia de requisitos e, após o estudo, selecionar um conjunto de abordagens adequadas para uma formalização de requisitos precisa e alinhada com as necessidades de negócio das organizações com SLT.

Este estudo envolveu, então, o entendimento dos conceitos de requisitos e engenharia de requisitos e a descoberta de abordagens de engenharia de requisitos.

Os requisitos podem ser entendidos como "declarações de necessidade, destinadas a transmitir entendimento sobre um resultado desejado, independentemente da sua efetiva realização" (Kotonya e Sommerville 1996, p. 5) e refletem "as necessidades dos clientes e utilizadores de um sistema" (Kotonya e Sommerville 1996, p. 6).

A engenharia de requisitos pode ser considerada um processo para a compreensão clara dos requisitos do sistema pretendido (Kotonya e Sommerville 1996). O seu objetivo é ajudar a saber o que construir antes do desenvolvimento do sistema começar, de forma a prevenir trabalhos extras dispendiosos (Paetsch, Eberlein e Maurer 2003).

O processo da engenharia de requisitos ajuda a identificar, analisar e documentar os requisitos do sistema e consiste em cinco atividades principais (Paetsch, Eberlein e Maurer 2003):

1. Levantamento: atividade de descoberta dos requisitos e identificação dos limites do sistema através da consulta aos *stakeholders* (ex. clientes, programadores, utilizadores).
2. Análise e negociação: atividade de verificação de requisitos e resolução de conflitos. Para todos os requisitos verifica-se a sua necessidade, consistência (os requisitos não devem ser contraditórios), a integridade (nenhum serviço ou restrição está a faltar) e viabilidade (os requisitos são viáveis no contexto do orçamento e do tempo disponível para o desenvolvimento do sistema). Os conflitos de requisitos que possam surgir são resolvidos por meio de negociação de priorização com os *stakeholders*.

3. Documentação: atividade de criação de documentos com o objetivo de comunicar os requisitos entre os *stakeholders* e os programadores. A documentação gerada deve servir de base para o restante processo de desenvolvimento de sistemas de informação, para validação do produto final e para a gestão das mudanças.
4. Validação: atividade de certificação de requisitos que são uma descrição aceitável do sistema a ser implementado. 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 identificados com os documentos de requisitos e as ações necessárias para lidar com os problemas identificados.
5. Gestão: atividade de captura, armazenamento, divulgação e gestão de informação. Inclui todas as atividades preocupadas com a mudança e controlo de versões e rastreamento de requisitos.

Na área da engenharia do software, a engenharia de requisitos desempenha um papel fundamental na especificação e evolução de sistemas de software e no alinhamento entre negócio e os SI/TI (Salgado, Machado e Suzana 2013).

Vários estudos demonstraram que o potencial impacto dos requisitos de software pobremente formulados é substancial. A maioria dos erros cometidos na formalização dos requisitos é muitas vezes encontrada nas fases de testes e entrega sendo o custo daí resultante normalmente considerável (Kotonya e Sommerville 1996).

A falha do software para satisfazer as reais necessidades dos utilizadores/clientes é a manifestação mais visível dos problemas associados ao estabelecimento de um conjunto adequado de requisitos para um sistema de software (Kotonya e Sommerville 1996) ou associados à execução do processo de engenharia de requisitos.

Não existe uma solução visível ou determinista para alinhar o software com as necessidades específicas de domínio (Ferreira, Santos, Machado, et al. 2013), todavia para potenciar o estabelecimento de um conjunto adequado de requisitos de software, várias técnicas podem ser utilizadas durante o processo de engenharia de requisitos (Paetsch, Eberlein e Maurer 2003).

Para a fase de levantamento de requisitos algumas técnicas a utilizar são: as entrevistas, cenários/casos de uso, observação e análise social, *focus groups*, *brainstorming* e prototipagem. Para a análise de requisitos sáo: a *joint application development* (JAD), priorização e modelação. Para a especificação de requisitos não foram encontradas técnicas específicas, no entanto

existem sugestões, como as de Berry (2003), para tornar o documento claro, completo, correto, compreensível, consistente e conciso. Para a validação de requisitos as técnicas utilizadas são as revisões e testes de requisitos. E para a atividade de gestão de requisitos não foram encontradas técnicas (Paetsch, Eberlein e Maurer 2003).

Para além destas técnicas, existem várias abordagens que ajudam a assegurar um levantamento preciso de requisitos quando não há acordo sobre ou contexto definido para o levantamento de requisitos (Ferreira et al. 2012), como por exemplo, a metodologia *soft systems* (Checkland 2000), o método de sistema de trabalho (Alter 2002) e o processo V+V (Ferreira, Santos, Soares, et al. 2013).

A Metodologia Soft Systems (SSM) é uma abordagem sistémica para lidar com situações problemáticas do mundo real onde não há uma definição formal dos problemas (Checkland 2000). Esta metodologia serve particularmente para lidar com situações complexas, pois ajuda, através de um processo iterativo de aprendizagem, a observar uma organização em várias perspetivas sistémicas diferentes, a determinar o problema que precisa ser tratado e a identificar as inter-relações entre os vários aspetos da situação problemática e, consequentemente, ao desenvolvimento da compreensão de um determinado problema. A Figura 17 apresenta as sete fases da metodologia soft systems de Checkland.

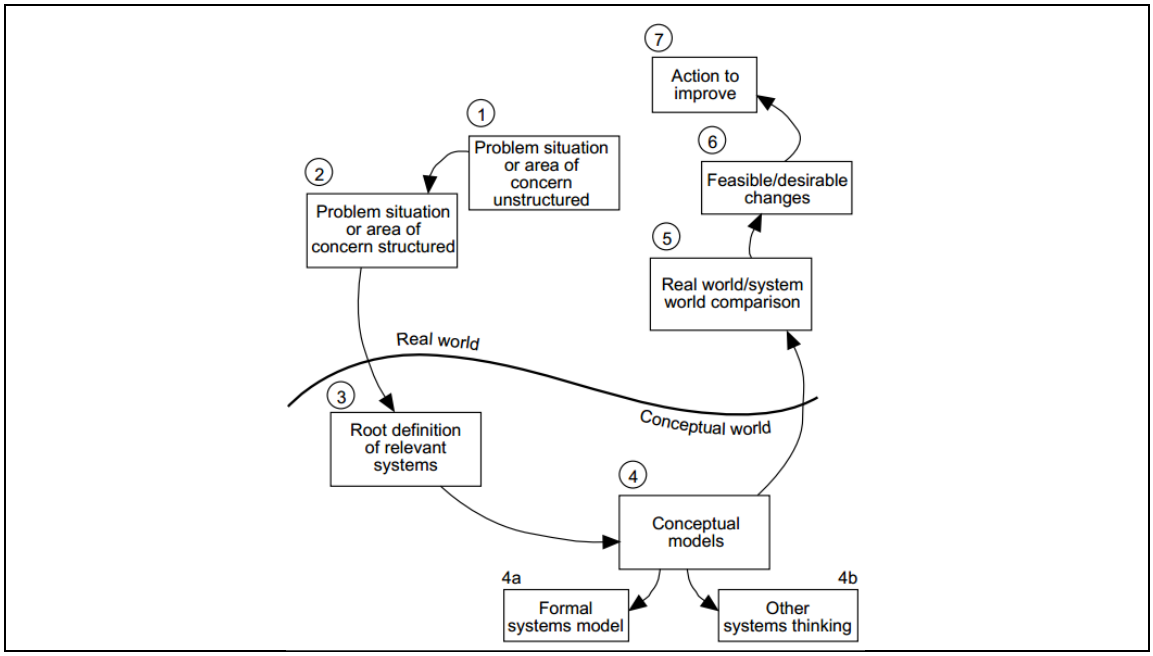


Figura 17 – Metodologia Soft Systems de sete fases de Checkland (retirado de (Patel 1995)).

O método de sistema de trabalho combina uma visão estática do sistema atual em operação ou sistema proposto e uma visão dinâmica da evolução do sistema ao longo do tempo (Alter 2002).

Este método tem como principal foco o "sistema de trabalho" para ajudar à compreensão, analise e melhoria dos sistemas nas organizações e é mais amplamente aplicável do que as técnicas concebidas para especificar requisitos de software. A Figura 18 apresenta a *framework* de sistema de trabalho (lado esquerdo), que pode ser utilizada para sintetizar e analisar qualquer sistema de trabalho, e o modelo de ciclo de vida de sistema de trabalho (lado direito), que resume a forma como um sistema de trabalho evolui.

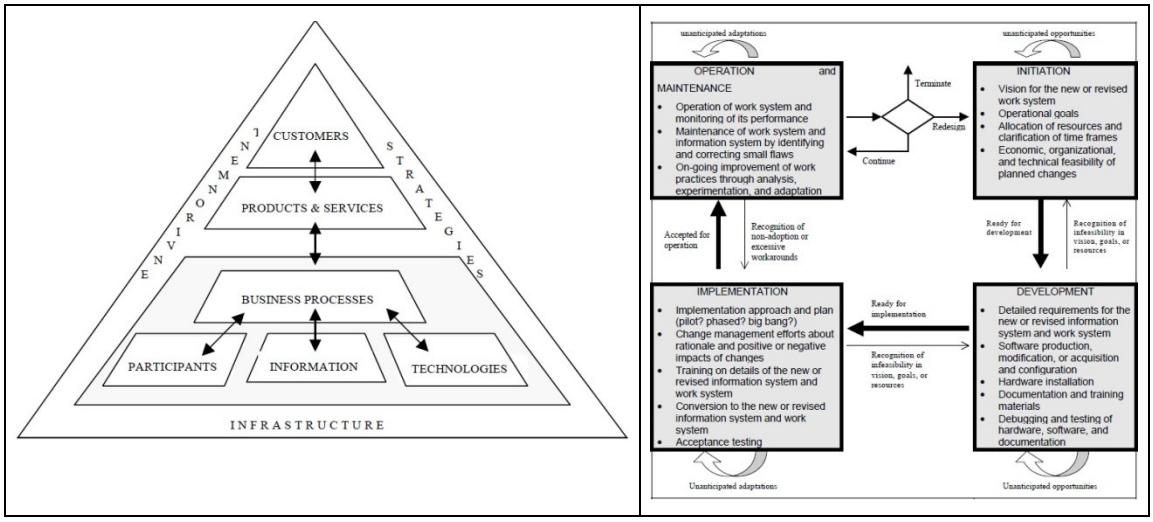


Figura 18 – *Framework* e modelo de ciclo de vida de sistema de trabalho (retirado de (Alter 2002)).

O processo V+V é uma abordagem baseada no modelo de processo "Vee" [3] que sugere um fluxo para a conceção do produto (software) com base nas necessidades do negócio identificadas numa fase de análise inicial (Ferreira, Santos, Soares, et al. 2013). Esta abordagem engloba dois modelos V, um trata a perspetiva de nível de processo, outro de produto, e requer a identificação das necessidades do negócio. Após a identificação das necessidades do negócio, por derivação sucessiva de artefactos, é possível transitar de uma perspetiva de nível de negócio para uma perspetiva de nível de TI, e ao mesmo tempo, alinhar os requisitos com os artefactos de TI derivados. Além disso, na fase de análise, esta abordagem assegura a transição entre as necessidades do negócio e o levantamento de requisitos. A Figura 19 procura ilustrar este processo.

Como resultado deste estudo, foi decido utilizar processo V+V, como abordagem de engenharia de requisito para a formalização dos requisitos de software, uma vez que se preocupa, ao contrário das outras duas abordagens, tanto com os aspetos de negócio como de produto e assegura a transição entre as duas perspetivas.

O método de sistema de trabalho comparativamente com a SSM apenas se foca numa parte das questões. Contudo, é uma metodologia mais prescritiva e útil para a descrição do sistema a ser estudado, identificação de problemas e oportunidades, descrição de possíveis alterações e monitorização dos possíveis impactos em outras partes do sistema.

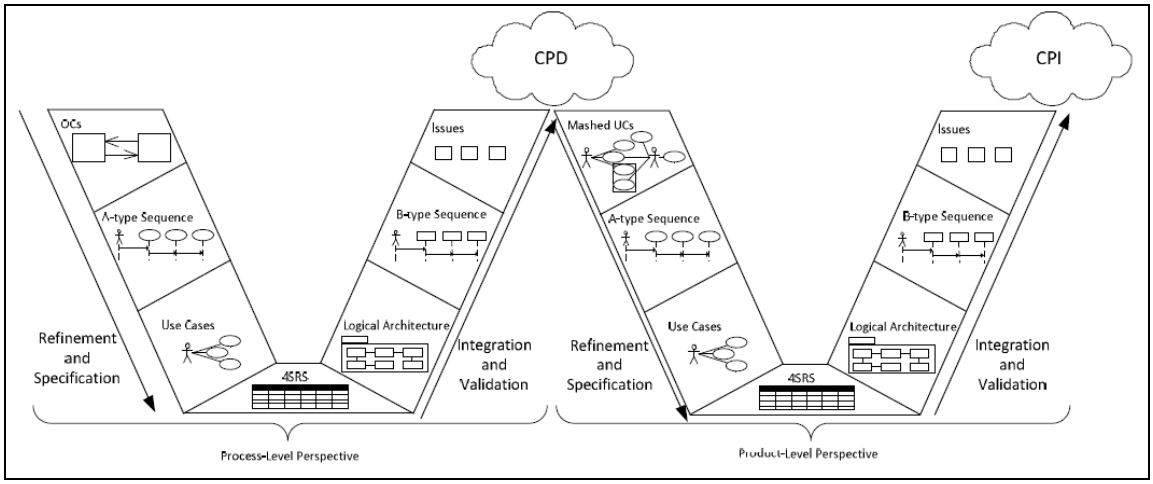


Figura 19 – Abordagem de processo V+V (retirado de (Ferreira, Santos, Soares, et al. 2013)).

O processo V+V comparativamente com o método de sistema de trabalho, para além de englobar muitos dos elementos do método de sistema de trabalho para a derivação de uma arquitetura lógica de nível de processo, também inclui os aspetos para a formalização de requisitos de produto, com a derivação da arquitetura lógica de nível de produto. O método de sistema de trabalho descarta a perspetiva de nível de produto.

Uma arquitetura lógica pode ser considerada uma visão de um sistema (ex. de software) composto por um conjunto de abstrações de problemas específicos que suportam requisitos funcionais (Azevedo et al. 2009). Uma arquitetura lógica de nível de processo pode ser definida como uma representação de uma disposição de atividades e suas interfaces num processo e tem em consideração alguns requisitos não-funcionais, como desempenho e disponibilidade. Ou seja, os requisitos para a execução do processo podem ser representados numa arquitetura lógica (Ferreira et al. 2012).

Se porventura, no caso prático desta dissertação (capítulo 4) tivesse sido realizado todo o processo V+V poder-se-ia dizer que uma iteração da fase de perceção do processo CSI teria sido também realizada. Contudo, e mapeando o caso prático com o processo V+V, da Figura 19, pode ser mencionado que foi realizada uma iteração do primeiro V.

3.4 Requisitos de Computação em Nuvem

Como se verificou na secção 2.3, uma solução de software para ser baseada no modelo de computação em nuvem deve incluir um conjunto de funcionalidades e características particulares, além das funcionalidades/características exigidas pelo negócio do consumidor (secção 3.2). Muitas destas funcionalidades e características "adicionais", para além de afetar positiva ou negativamente o negócio do consumidor, também influencia positiva ou negativamente o negócio do fornecedor. Por isso, e de forma a melhorar o contexto para a conceção do produto para os SLT e para as organizações fornecedoras de recursos e serviços de TI, julgou-se que em cada iteração do primeiro V do processo V+V, poderia ser utilizada a informação do modelo de referência de computação em nuvem do NIST.

A utilização de modelos de referência de computação em nuvem para a identificação e validação de requisitos de nível de processo não é uma prática muito discutida na literatura. No entanto, Pereira (2013) conseguiu demonstrar, através de dois casos de demonstração, que o alinhamento das arquiteturas de negócio (nível de processo) e de serviços para a nuvem (nível de produto) pode ser assegurado com a utilização do modelo de referência de computação em nuvem do NIST.

No primeiro caso, o modelo do NIST foi aplicado ao nível dos casos de uso que descrevem as principais funcionalidades de um sistema ERP, através do cruzamento das descrições dos casos de uso com as descrições arquiteturais dos componentes do modelo do NIST. No segundo caso, o modelo do NIST foi aplicado ao nível dos elementos arquiteturais da arquitetura lógica de nível de processo, que especifica as principais atividades e necessidades de um sistema que suporta a disponibilidade de serviços baseados em nuvem, através do cruzamento das descrições dos elementos arquiteturais da arquitetura lógica de nível de processo com as descrições dos componentes do modelo de referência do NIST.

Em ambos os casos, de acordo com Pereira (2013), a aplicação do modelo do NIST possibilitou a identificação e correção de incoerências semânticas, a descoberta e definição de requisitos, o auxilio para a conceção de arquiteturas, a definição de serviços e a sistematização de normas e protocolos para as respetivas arquiteturas de computação em nuvem.

O modelo NIST é o mais reconhecido e utilizado pela indústria para discutir e analisar as atividades e funções da computação em nuvem (Pereira 2013) e resulta de uma ampla

participação da indústria, de académicos, de organizações de desenvolvimento de normas e de entidades do setor público e privado (Liu et al. 2011).

A Figura 20 apresenta uma visão geral do modelo NIST, que identifica os principais atores, suas atividades e funções em computação em nuvem. Este diagrama é uma representação de um modelo conceptual genérico de alto nível que tem como objetivo facilitar a compreensão dos requisitos, formas de utilização, características e padrões de computação em nuvem.

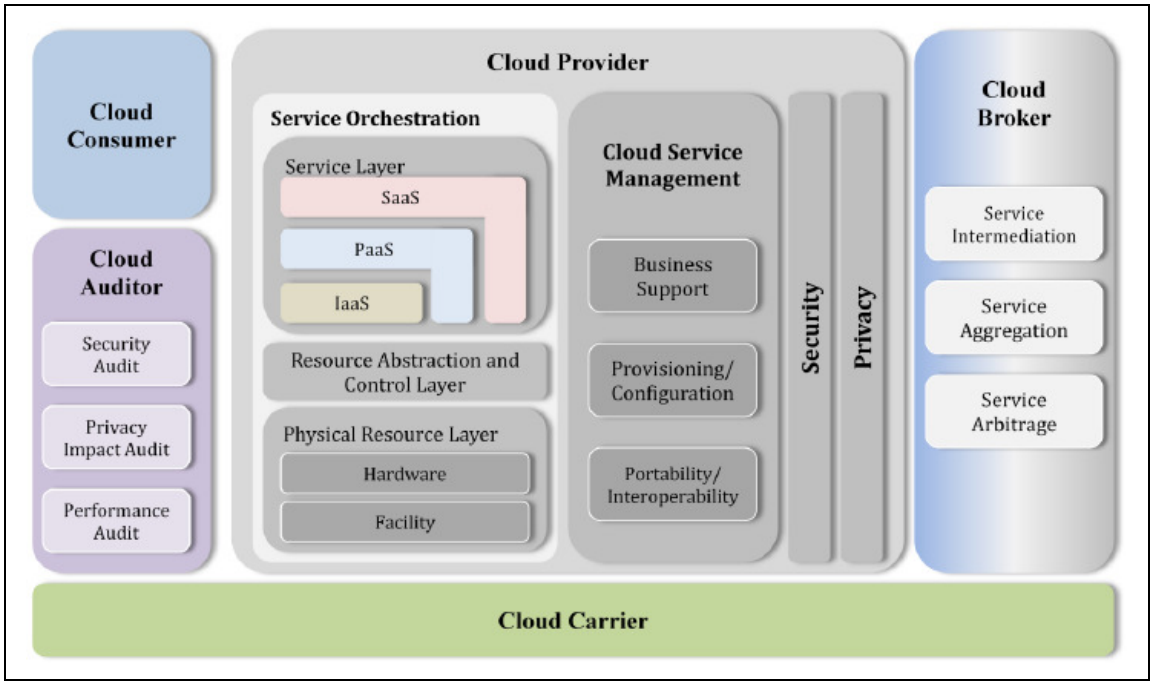


Figura 20 – Arquitetura de referência de computação em nuvem NIST (retirado de (Liu et al. 2011)).

Como a Figura 20 apresenta, o modelo NIST define cinco principais atores: o consumidor de nuvem, fornecedor de nuvem, *broker* de nuvem, auditor de nuvem e *carrier* de nuvem. Cada ator é uma entidade (uma pessoa ou uma organização) que participa numa transação ou processo e/ou desempenha tarefas em computação em nuvem (Liu et al. 2011).

Um consumidor de nuvem é um indivíduo ou organização que adquire e utilize produtos e serviços em nuvem. O fornecedor de produtos e serviços de nuvem é o fornecedor de nuvem. Devido às diferentes possíveis ofertas de serviços (i.e. software, plataforma ou infraestrutura) permitidos pelo fornecedor de nuvem, haverá uma mudança no nível de responsabilidades para alguns aspetos do âmbito de controlo, segurança e configuração. O *broker* de nuvem atua como um intermediário entre o consumidor e o fornecedor de nuvem e ajuda o consumidor através da complexidade das ofertas de serviços em nuvem. Este ator pode também criar serviços em nuvem com valor acrescentado. O auditor de nuvem realiza avaliações de desempenho

independentes e monitoriza a segurança de serviços em nuvem. O *carrier* de nuvem é uma organização que tem a responsabilidade de transferir os dados.

Os componentes do modelo NIST descrevem os aspetos importantes da implementação, orquestração e gestão dos serviços em nuvem (Liu et al. 2011).

A implementação de uma infraestrutura de nuvem pode ser realizada de quatro formas diferentes (Liu et al. 2011). Numa nuvem pública a infraestrutura de nuvem e recursos de computação são disponibilizados para o público em geral através de uma rede pública. Numa nuvem privada uma única organização do consumidor de nuvem tem acesso exclusivo e utilização da infraestrutura e recursos computacionais. Uma nuvem de comunidade serve um grupo de consumidores de nuvem que partilham preocupações comuns. Uma nuvem híbrida é uma composição de duas ou mais nuvens (pública, privada ou comunidade) que permanecem como entidades distintas, mas estão unidas por tecnologia padronizada ou proprietária que permite a portabilidade de dados e aplicações.

A orquestração de serviços (Service Orchestration) refere-se à composição dos componentes do sistema que sustenta o aprovisionamento de serviços em nuvem para os consumidores de nuvem e diz respeito às atividades dos fornecedores de nuvem relativamente à disposição, coordenação e gestão de recursos de computação (Liu et al. 2011). Tem que ver com as cinco camadas de Youseff, Butrico, e Da Silva (2008) apresentadas na secção 2.3.3.

A gestão dos serviços da nuvem (Cloud Service Management) é reconhecida como um elemento importante da arquitetura. Mecanismos de suporte ao negócio (Business Support Management) são úteis para reconhecer as questões relacionadas com a gestão de clientes como contratos, forma de cobrança e preços e são vitais para a computação em nuvem. A discussão sobre aprovisionamento e configuração (Provision and Configuration) destaca os requisitos relacionados com a instalação, operação e manutenção dos serviços, incluindo a medição de serviços e gestão de SLA para os sistemas de nuvem. As questões de portabilidade e interoperabilidade (Portability and Interoperability) de dados, sistemas e serviços são fatores cruciais para os consumidores da nuvem. As preocupações de segurança e privacidade (Security and Privacy) precisam ser abordadas para criar uma atmosfera de aceitação na capacidade da nuvem para fornecer um sistema fiável e seguro.

Para o caso prático desta dissertação, decidiu-se utilizar o modelo NIST, juntamente com a arquitetura de nível de processo SLT, para descobrir novos requisitos de computação em nuvem

de nível de processo que não foram considerados durante a conceção da arquitetura SLT de nível de processo (secção 4.4).

3.5 Conclusões

O principal objetivo deste capítulo foi formular o problema dos SLT e definir, de forma justificada, uma estratégia para a conceção de uma solução entregue como um serviço que resolva o problema dos SLT. Para concretizar este objetivo foram realizados três estudos distintos.

O primeiro estudo foi sobre as organizações com processos de SLT e sobre o estado atual do mercado para as soluções de serviços como aquele que é estudado nesta dissertação. Este estudo permitiu uma melhor compreensão das necessidades das organizações com SLT e a comprovação da pertinência de conceber uma solução de software entregue como um serviço para o domínio dos SLT.

O segundo estudo foi sobre as abordagens de engenharia de requisitos. Este estudo possibilitou, de forma mais consciente, selecionar uma abordagem adequada para a conceção de uma solução de software alinhada com as necessidades dos SLT.

O terceiro consistiu no modelo de referência de computação em nuvem NIST. Este permitiu identificar as possíveis formas de aplicação do modelo NIST para auxiliar a conceção de uma arquitetura de software baseada no modelo e computação em nuvem.

Este capítulo revelou-se crucial na deliberação de uma estratégia para construção de um sistema de software baseado no modelo de computação em nuvem e alinhado com o domínio dos SLT. Para além disso, permitiu iniciar a execução da estratégia definida. É apresentada no próximo capítulo a exemplificação da utilização do processo V+V de Ferreira, Santos, Machado, et al. (2013) e do modelo de referência de computação em nuvem do NIST (Liu et al. 2011) para a conceção do sistema de software.

CAPÍTULO 4 – UM CASO PRÁTICO PARA OS SISTEMAS LOGÍSTICOS DE TRANSPORTES

4.1 Introdução

O capítulo anterior permitiu enquadrar a atividade de CSI, apresentar o problema dos SLT e identificar diferentes abordagens de engenharia de requisitos. A análise efetuada desse capítulo culminou na seleção do processo V+V de Ferreira, Santos, Machado, et al. (2013) e na decisão de utilizar o modelo de referência de computação em nuvem do NIST (Liu et al. 2011) para a construção de um sistema de software alinhado com as necessidades dos SLT e baseado no modelo de computação em nuvem. Este capítulo, o quarto, procura explicar os principais resultados alcançados e todas as decisões realizadas durante a execução do processo V+V e durante a aplicação do modelo do NIST.

O processo V+V, como já foi referido anteriormente, engloba dois modelos V. Um trata a perspetiva ao nível de processo, outro de produto. No entanto, e devido ao tempo disponível para a concretização deste trabalho de dissertação, só foi possível a realização de uma iteração do primeiro modelo V.

O principal objetivo da execução do primeiro modelo V, em diante designado de modelo V, diz respeito ao levantamento de requisitos de um nível de negócio de alto nível para criar contexto para a conceção do produto (Ferreira, Santos, Soares, et al. 2013). No modelo V são utilizadas configurações organizacionais (OC), diagramas de sequência do tipo A e B, modelos de caso de uso e um modelo de arquitetura logica de nível de processo. O seu vértice é suportado pelo método Four-Step-Rule-Set (4SRS) de nível de processo e a sua execução resulta na criação de uma arquitetura lógica validada (Azevedo et al. 2009).

Como apresenta a Figura 21, o modelo V promove o alinhamento entre os modelos do domínio do problema e os modelos do domínio da solução (Ferreira, Santos, Machado, et al. 2013). No lado descendente do modelo V (lado esquerdo do V), os modelos criados em sucessão representam o refinamento dos requisitos e a criação de especificações do sistema. No lado ascendente (o lado direito da V), os modelos representam a integração das partes lógicas descobertas e o seu envolvimento num esforço de validação cruzada.

Terminada a iteração do primeiro modelo V, foi utilizado o modelo do NIST com o objetivo de identificar requisitos de computação em nuvem. Decidiu-se apenas utilizar o modelo NIST no final da primeira iteração para que a arquitetura SLT possa ser aproveitada mais facilmente para outros contextos, além do contexto de computação em nuvem, em trabalhos futuros.

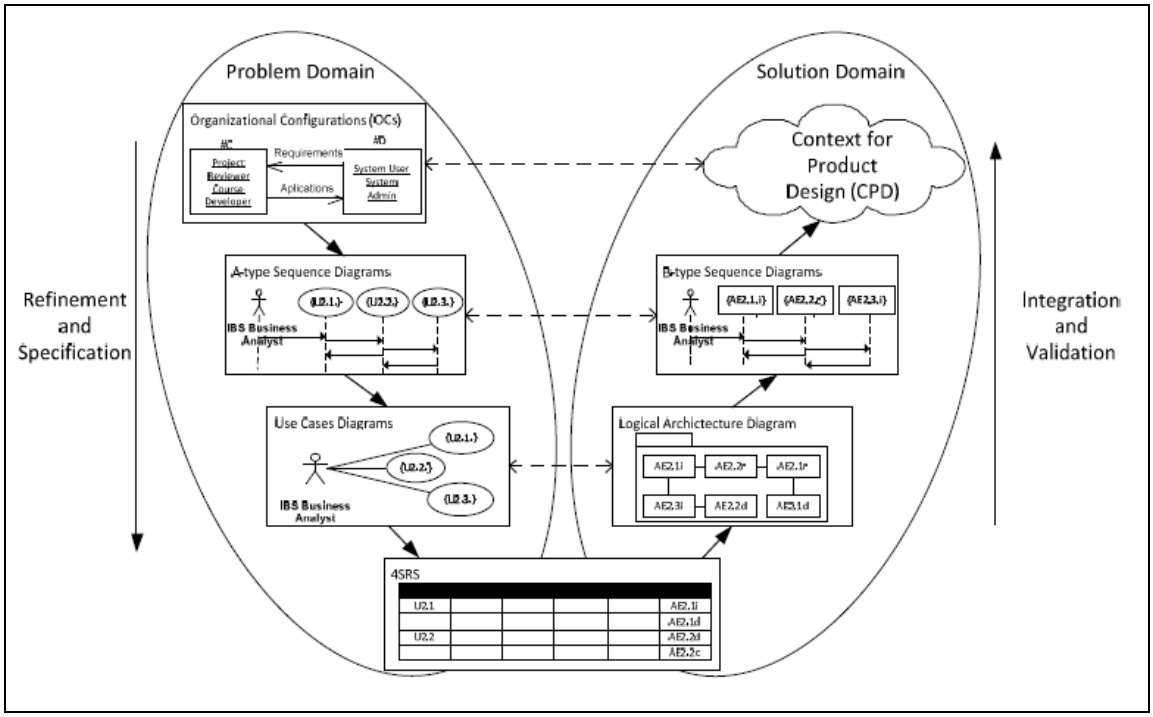


Figura 21 – Modelo V para alinhar o domínio e o software (retirado de (Ferreira, Santos, Machado, et al. 2013)).

A explicação deste caso prático está dividida em três secções. Na secção 4.2 explica-se como foi a realização do refinamento e da especificação dos requisitos do domínio do problema. Na secção 4.3 explica-se como foi a realização do restante processo do modelo V, derivação da arquitetura lógica de nível de processo e validação. Na secção 4.4 explica-se a utilização do modelo NIST para a identificação de requisitos de computação em nuvem que não foram considerados durante a primeira iteração do modelo V.

Assim sendo, e antes de se avançar com os assuntos propostos neste capítulo, julga-se relevante destacar que esta explicação apenas apresenta alguns dos resultados alcançados durante a execução do processo V+V (contudo, estes podem ser consultados em "Anexo B – Resultados do Modelo V") sendo que a utilização do termo nó ou *node*, neste capítulo, refere-se a uma organização interdependente, que coopera com outras para controlar, gerir e/ou melhorar o fluxo de materiais e informação desde os fornecedores aos utilizadores-finais.

4.2 Modelos de Domínio do Problema

A execução do modelo V requer a identificação das necessidades específicas do domínio para a conceção de uma arquitetura lógica de nível de processo. Isto quer dizer que antes de iniciar o processo de conceção de uma arquitetura de nível de processo, deve ser realizado um levantamento adequado de requisitos. Esta secção apresenta a forma como foram identificados, delimitados e formulados os requisitos de domínio dos SLT.

Uma vez que o modelo V não suporta qualquer técnica específica para levantamento de requisitos, decidiu-se utilizar as informações fornecidas por duas técnicas de levantamento de requisitos. A primeira técnica utilizada foi a observação e análise. A segunda foi os cenários de utilização.

Para além destas técnicas, de modo a facilitar a realização do levantamento de requisitos e tornar o conjunto de requisitos mais próximo das necessidades do mundo real, foi decidido também utilizar a informação de uma solução real relacionada com o domínio dos SLT. Esta solução é designada de SLV Cement, foi desenvolvida pela Cachapuz Bilanciai Group e tem-se vindo a afirmar na indústria cimenteira (Cachapuz Bilanciai Group).

Comparativamente com o conceito de SLT utilizado nesta dissertação, o conceito SLV Cement é menos amplo e genérico. Por exemplo, o SLV Cement é uma solução "de expedição e de controlo dos fluxos logísticos que automatiza todos os processos desde a chegada de um camião à fábrica até à expedição de cimento... garante toda a troca de informação entre os departamento logístico e o comercial... complementa por isso o software de gestão do cliente (ERP)" e não é baseada no modelo de computação em nuvem. Um sistema de software a ser desenvolvido para o domínio de SLT é uma solução de expedição e de controlo dos fluxos logísticos que automatiza todos os processos desde a chegada de um meio de transporte a um nó até à expedição do produto final (i.e. com *n* tarefas), garante toda a troca de informação entre departamentos de um nó e entre nós de uma cadeia de abastecimento e permite a interoperabilidade com outros sistemas de software (ex. ERP, CRM e SCM). Todavia, apesar das suas diferenças, a análise efetuada à solução SLV Cement foi fundamental para compreender as necessidades internas de um nó relativamente à sua cadeia de abastecimento.

Identificados os requisitos do domínio do SLT, foram criadas representações (i.e. configurações organizacionais e diagramas de sequência do tipo A e B) que caracterizam as relações organizacionais, os processos e as interações entre atores a diferentes níveis de abstração.

4.2.1 Configurações Organizacionais

Como ponto de partida para a definição do contexto de domínio SLT, foi modelado um conjunto de configurações organizacionais (COs) que dizem respeito às possíveis interações entre os atores do domínio e a solução SLT (secção B.1). Mas antes de apresentar alguns exemplos das COs definidas, julga-se relevante apresentar de forma breve o que é uma CO e o que esta deve conter.

Uma CO modela uma possível relação inter-organizacional a um nível alto de abstração e não considera os processos e/ou atores envolvidos na relação a um nível detalhado. Ou seja, o modelo CO é uma representação de alto nível das atividades (interações) que existem entre as entidades de nível de negócio de um determinado domínio. O modelo de CO deve conter informações sobre as atividades realizadas, os diversos perfis profissionais (atores e habilidades) que participam na execução da atividade e também a troca de informações ou artefactos (Ferreira, Santos, Machado, et al. 2013).

Não existe um número certo de COs que devem ser modeladas, contudo deve ser um número suficiente para descrever, pelo menos, as principais relações demonstradas pelas necessidades específicas do domínio dos *stakeholders* (Ferreira, Santos, Machado, et al. 2013).

Deste modo, antes de modelar as COs do SLT, foi necessário inicialmente definir os atores de domínio, que dizem respeito às funções existentes nos processos numa perspetiva de alto nível da solução SLT, e deliberar um conjunto de tipos de atividades, que representam o tipo de participação que os atores possuem no âmbito dos SLT.

Sobre os atores foram definidos três: SLT Consumer, SLT Solution e SLT Supplier.

O primeiro ator de domínio (SLT Consumer) é um que use diretamente os recursos da SLT Solution e que tem um papel de consumidor de serviço da SLT Solution. Normalmente representa uma entidade de negócio (i.e. o nó ou os clientes, fornecedores e transportadores de um nó). Quando a generalização do termo SLT Consumer não se aplica, este papel é representado pelo ator de sistema adequado (consultar secção 4.2.3 Tabela 6). Um SLT Consumer pode ser um utilizador de negócio (i.e. User, Programmer, Controller, Technician, Analyst, External Entity) ou um sistema de software externo de negócio (i.e. Other Software Systems).

O segundo ator de domínio (SLT Solution) não é uma entidade mas sim um ambiente de execução para as funcionalidades. A SLT Solution representa a execução do serviço, que está presente entre os pedidos do SLT Consumer e as respostas do SLT Supplier. O ator de negócio que executa no âmbito do SLT Solution é o administrador de sistemas (Admin).

Por fim, o terceiro (SLT Supplier) é um ator que oferece os serviços SLT para os consumidores e que tem um papel de fornecedor de serviço do SLT Solution. Os serviços fornecidos são disponibilizados no SLT Solution através das configurações do administrador de sistemas (Admin). O Admin pode ser uma pessoa contratualmente ligada a uma empresa fornecedora de software ou ligada a um nó.

Relativamente aos tipos de atividades foram determinados quatro: SLT Solution Services Activities (#A), Consumer Business Activities (#B), Supplier Services Management Activities (#C) e Infrastructure Support and Maintenance Activities (#D).

O primeiro tipo (#A: SLT Solution Services Activities) diz respeito às atividades executadas na SLT Solution. Este tipo inclui atividades (#A.1: Planning Activities) destinadas a criar as condições adequadas para o bom funcionamento das atividades de negócio de um nó (i.e. as atividades do tipo #A.2: Business Activities), através da programação das ordens de operações; atividades (#A2: Business Activities) relativas à execução de negócio de um nó, ou seja, todas as tarefas necessárias para receber matérias-primas e/ou expedir produtos finais (i.e. o que as organizações com processos SLT oferecem aos seus clientes); as atividades (#A.3: Business Analysis Activities) relacionadas com a monitorização do desempenho de negócio. O objetivo destas atividades é determinar soluções de melhoria para o desempenho das atividades de negócio (i.e. as atividades do tipo #A.2: Business Activities); as atividades (#A.4: Assistance Activities) proporcionam o suporte necessário aos utilizadores-finais para a realização, de forma correto e independente, das suas ordens de operações; e atividades (#A.5: Incidents Activities) referentes à deteção e comunicação de situações de anormalidade ou de incidente da Solution SLT.

O segundo tipo (#B: Consumer Business Activities) engloba as atividades relacionadas com a execução de negócio de uma entidade de domínio de negócio (i.e. clientes, fornecedores, transportadoras de um nó e o nó);

O terceiro tipo (#C: Supplier Services Management Activities) compreende as atividades relativas à instalação, operação e manutenção de serviços da SLT Solution.

Por fim, o último tipo (#D: Infrastructure Support and Maintenance Activities) inclui atividades típicas relacionadas com a administração, gestão e monitorização de infraestruturas de TI.

Definidos os atores de domínio e os tipos de atividades, de seguida foram criadas dois cenários (i.e. COs), compostos por uma descrição sobre a troca de informação, onde algumas interações são definidas.

Como exemplo apresenta-se de seguida o cenário "The SLT Consumer Requests an Existing Service Execution in the SLT Solution". Este cenário descreve as relações necessárias quando um utilizador de negócio (SLT Consumidor) solicita a execução de um serviço existente na SLT Solution. Como se pode observar na Figura 22, este tem dois tipos de atividade (#A e #B), cada uma com um papel, três interações e dois atores de domínio (SLT Consumer e SLT Solution).

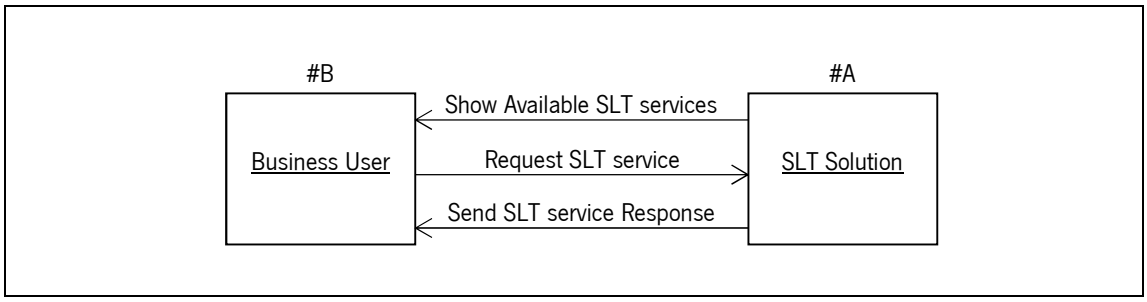


Figura 22 – Configuração Organizacional, cenário 1.

Ao analisar os tipos de atividades deste cenário, verifica-se que a utilização ou consumo de um serviço diz respeito ao tipo Consumer Business Activities (#B), ao passo que a disponibilização ou fornecimento de serviços está associada ao tipo SLT Solution Services Activities (#A).

4.2.2 Diagramas de Sequência do Tipo A

Uma vez modeladas as configurações organizacionais, é preciso definir as relações entre as atividades e os atores de forma detalhada (secção B.2). Esta subsecção apresenta quais as relações definidas para o domínio SLT e como estas foram modeladas.

No modelo V as relações entre as atividades e os atores são definidas por meio de interações sendo esta exibida num diagrama de sequência UML estereotipado (Ferreira, Santos, Machado, et al. 2013).

O modelo V contém dois tipos de diagramas de sequência estereotipados, o tipo A e o tipo B. Um diagrama de sequência do tipo A permite a derivação das sequências de processo representadas pelos fluxos entre os atores e os casos de uso e pode ser utilizado para a

modelação dos casos de uso. Um diagrama do tipo B permite o mesmo contudo a representação é com base nos fluxos entre os atores e as partes lógicas da arquitetura lógica e a sua utilização é realizada numa fase mais avançada do modelo V (subsecção 4.3.2). Ambos, para além de representarem as sequências, também modelam a troca de mensagens entre os atores e os casos de uso/partes lógicas.

Deste modo, e nesta fase do modelo V, importava então definir um conjunto de diagramas de sequência do tipo A com base na informação existente (i.e. informação sobre os SLT, SLV Cement e COs). Os diagramas de sequência do tipo A realizam os papéis apresentados numa CO e instancia-os em atividades. Estes enquadram a execução das atividades no tempo, permitem uma representação funcional da interação comportamental com o meio ambiente e fornecem informações para a definição e modelação de casos de uso numa perspetiva de nível de processo (Ferreira, Santos, Machado, et al. 2013). De acordo com Machado et al. (2007) são apropriados para ilustrar os requisitos dos utilizadores de fluxo de trabalho.

Assim, e depois de identificados cenários de interações do domínio SLT, foi elaborado um conjunto de diagramas de sequência do tipo A com a intenção de representar os fluxos de atividade num ambiente SLT (secção B.2). Este conjunto retrata as principais interações entre os atores, SLT Consumer e SLT Solution, e as atividades, #A e #B, definidas nas COs. As interações entre os atores, SLT Supplier e SLT Solution, e as atividades, #C, #D e #B, não foram tratadas de forma detalhada nesta fase, uma vez que a perceção sobre a sua verdadeira dimensão e complexidade deu-se na fase final deste trabalho, mais especificamente, aquando da aplicação do modelo do NIST (secção 4.4). Todavia, considera-se crucial que estas interações sejam analisadas num trabalho futuro, principalmente se for pretendido um sistema de software alinhado com as necessidades de computação em nuvem.

O diagrama (#1: Operation Orders Planning) foi criado para retratar as interações possíveis associadas ao processo de planeamento das ordens de operação de um nó. Os diagramas (#2: Parking, #3: Entrance/Exit, #4: Weighing, #5: Loading e #6: Unloading) foram elaborados com o intuito de apresentar as interações possíveis relacionadas com as tarefas que podem fazer parte do processo de realização de ordens de operação de um nó. Poderiam ser elaborados mais diagramas sobre outras tarefas, mas considerou-se que seriam suficientes estas cinco tarefas para o âmbito deste trabalho. Por fim, o diagrama (#7: Repair/Maintenance Assistance) foi

definido para destacar as interações possíveis relativas ao processo de reparação/manutenção do SLT Solution.

Como exemplo apresenta-se, de seguida, o diagrama "#2: Parking", conforme ilustra a Figura 23. Neste diagrama são representados fluxos sequenciais de casos de uso de nível de processo que se referem às atividades necessárias para a realização de *check-in* e de verificação de disponibilidade de um nó para a execução de ordens de operação. Estas atividades são executadas no âmbito das atividades do tipo #A, depois de receber o pedido de serviço e antes de responder a esse pedido (interações representadas nas CO da Figura 22).

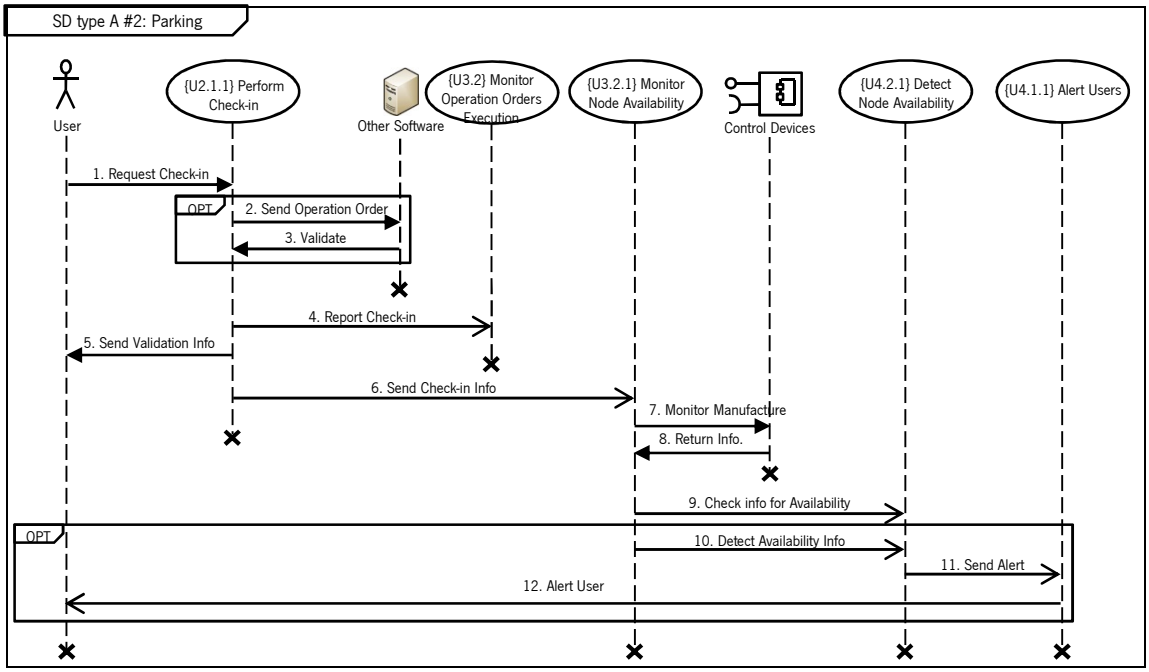


Figura 23 – Diagrama de sequência do tipo A, *parking*.

4.2.3 Diagramas de Casos de Uso

Modelados os diagramas de sequência do tipo A, de seguida estes podem ser utilizados como uma técnica de levantamento de requisitos para a modelação de casos de uso. Nesta subsecção apresenta-se de que forma foram refinados os requisitos de nível de processo do domínio dos SLT através de diagramas de casos de uso (secção B.3).

Uma vez que os cenários expressos nos diagramas de sequência do tipo A são construídos com a utilização de casos de uso na forma de atividades, os diagramas de casos de uso podem ser considerados refinamentos dos diagramas de sequência do tipo A (Ferreira, Santos, Machado, et al. 2013).

Nos diagramas de sequência do tipo A são utilizados casos de uso e em cada um deles são associados atores e outros casos de uso correspondentes. Nos diagramas de casos de uso, os casos de uso são organizados depois de ser eliminada a redundância e depois de ser atribuído um nome adequado a cada um deles.

O modelo de caso de uso específica as utilizações necessárias do sistema. Os casos de uso de nível de processo devem permitir uma melhor compreensão das atividades (processos) executadas por pessoas ou máquinas no âmbito do sistema. Por isso, para além dos diagramas de casos de uso, cada caso de uso deve ter uma descrição textual de forma a representar em plenitude o modelo funcional permitindo a identificação e descrição dos requisitos de processo.

Assim, e em relação aos casos de uso de nível de processo para o domínio dos SLT, o processo de modelação iniciou com a definição de um conjunto de atividades canónicas e atores de sistema. Na Figura 24, pode-se observar os principais atores e as suas principais interações com as atividades canónicas do SLT Solution do ponto de vista de processo. Na Tabela 6, pode-se verificar os atores do sistema identificados e especificados. Um ator de sistema representa um utilizador que interage com o SLT Solution.

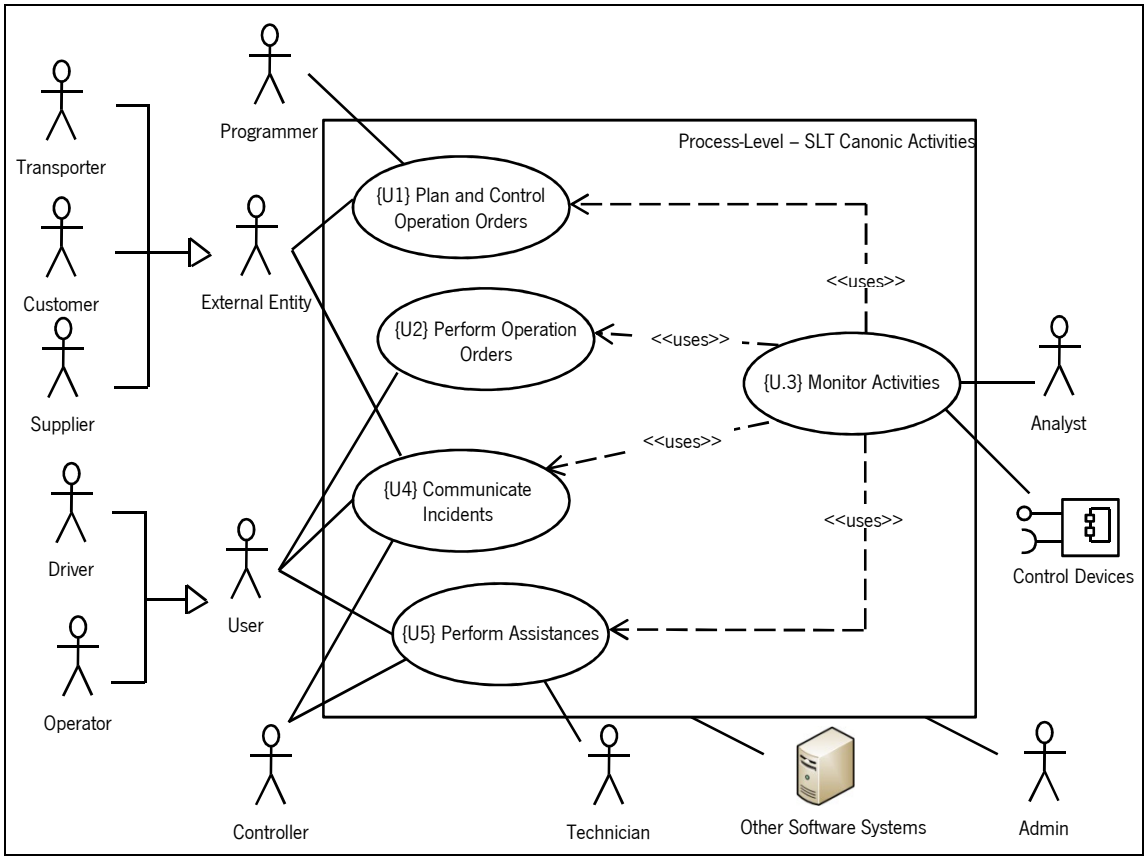


Figura 24 – Atividades Canónicas do SLT.

Como apresenta a Figura 24, numa vista de alto nível de processo, um conjunto de cinco atividades canónicas, que são diretamente relacionadas a uma atividade realizada pelo SLT Solution, foi definido: {U1} Plan and Control Operation Orders, {U2} Perform Operation Orders, {U.3} Monitor Activities, {U4} Communicate Incidents e {U5} Perform Assistances.

Sem pretender tecer considerações detalhadas, pode-se descrever a atividade {U1} Plan and Control Operation Orders como a atividade de planeamento e controlo de um conjunto de ordens de operação de um nó. Na perspetiva de um nó, uma ordem de operação consiste na receção de matéria-prima e/ou na expedição do produto final. Esta atividade corresponde à #A.1: Planning Activities apresentada na subsecção das COs.

A atividade {U2} Perform Operation Orders é relativa à execução de um conjunto de tarefas necessárias para garantir a automatização, o controlo e a segurança na realização de uma ordem de operação por parte de um utilizador (User). O número de tarefas necessárias pode variar consoante o contexto de aplicação do SLT. Por isso, o SLT deverá permitir ao administrador (Admin) a configuração das tarefas necessárias, assim como a sequência de execução dessas mesmas tarefas. Esta atividade corresponde à #A2: Business Activities apresentada na subsecção das COs.

Tabela 6 – Atores de Sistema.

Nome	Descrição
User	O User representa o ator sobre o qual o sistema em desenvolvimento se concentra. Este pode ser um Driver de uma External Entity ou um Operator de um nó e é apresentado neste contexto como um utilizador que direta ou indiretamente interage com os produtos e serviços do SLT.
Admin	O Admin é um ator com privilégios para configurar o comportamento dos Control Devices e para gerir a infraestrutura e os serviços SLT.
Analyst	Este ator é uma pessoa capaz de inferir a qualidade de serviço de um nó e reportar sugestões de melhoria.
Programmer	Um Programmer é uma pessoa capaz de gerir, definir e programar uma agenda de ordem de operações de um nó.
Controller	Um Controller é uma pessoa capaz de analisar os pedidos de assistência de um nó e contactar os Technicians responsáveis e adequados para cada situação de reparação do sistema. Para além disso, o Controller também pode alertar todas as External Entities para eventuais atrasos no processamento das ordens operações do nó.
Technician	Um Technician é uma pessoa capaz de prestar assistência aos Users e executar a manutenção dos equipamentos do sistema, tanto remotamente como no local.
External Entity	Este ator representa as entidades externas de um nó, como por exemplo, empresas clientes, empresas fornecedoras e empresas transportadoras. As External Entities podem planear e gerir com mais precisão as suas necessidades relativas a um nó.
Other Software Systems	Este ator representa o software que já está instalado e explorado dentro do contexto de cada nó da cadeia de abastecimento. Através deste software (ex. um ERP, CRM ou SCM) uma pessoa pode interagir com o SLT e explorar as funcionalidades do SLT.
Control Devices	Representa um conjunto de dispositivos de monitorização e controlo (ex. equipamentos de pesagem, sensores, postos e semáforos) que garantem a segurança da execução das ordens de operação de um nó.

Por sua vez, o caso de uso {U3} Monitor Activities está relacionado com a atividade de monitorização de todas as atividades do SLT Solution para que o analista (Analyst) consiga identificar áreas de melhoria e para que controle a realização das ordens de operação. Esta atividade corresponde à #A.3: Business Analysis Activities apresentada na subsecção das COs.

O caso de uso {U4} Communicate Incidents deteta incidentes, i.e. parâmetros anormais ou eventos relevantes, relativamente à utilização do SLT e permite o envio de alertas entre utilizadores. Quando uma anormalidade ou um evento é detetado, o sistema alerta os utilizadores especificados para essa anormalidade ou para esse evento. Esta especificação deverá ser realizada pelo administrador (Admin) do SLT. Esta atividade corresponde à #A.5: Incidents Activities apresentada na subsecção das COs.

Por fim, a atividade {U5} Perform Assistances representa o suporte e manutenção necessário para manter a correta e independente utilização do SLT. O utilizador (User) ou o sistema (Communicate Incidents) pode requisitar um pedido de assistência, o controlador (Controller) pode gerir os pedidos de assistência e o técnico (Technician) pode oferecer serviços de assistência remota e/ou local. Esta atividade corresponde à #A.4: Assistance Activities apresentada na subsecção das COs.

Posteriormente à definição das atividades canónicas de alto nível de processo, cada uma delas foi refinada por meio de decomposição (Cruz, Machado e Santos 2012). No final deste processo de refinamento de requisitos, obteve-se uma especificação detalhada de 49 casos de usos (consultar secção B.3), dos quais 39 (a cinzento na Figura 25) devem ser utilizados como *input* para o método 4SRS. A Figura 25 apresenta a estrutura em árvore que resultou do processo de decomposição dos requisitos de nível de processo.

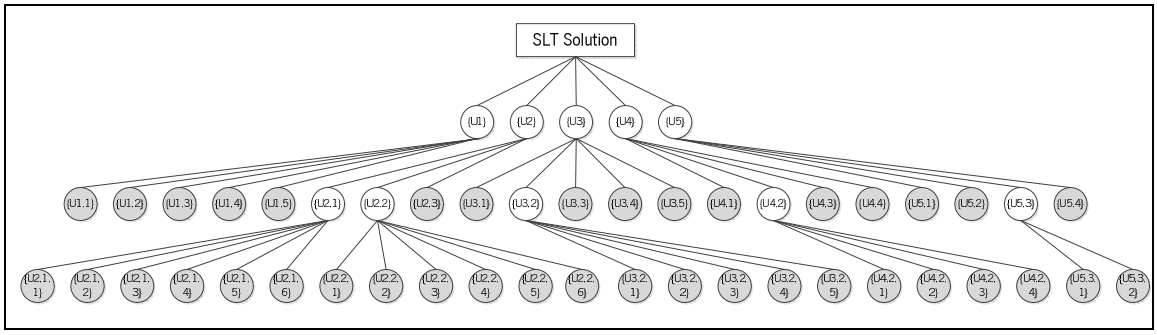


Figura 25 – A estrutura de árvore do SLT Solution.

4.3 Derivação da Arquitetura Lógica e Modelos de Domínio da Solução

Reunida a informação sobre as necessidades específicas do domínio, pode-se agora avançar para a conceção de uma arquitetura lógica de nível de processo alinhada com as necessidades do domínio. Esta secção inicia com a apresentação da forma como foi derivada a arquitetura lógica de nível de processo do sistema pretendido (subsecção 4.3.1) e como esta foi validada (subsecção 4.3.2).

4.3.1 Derivação da Arquitetura Lógica

De modo a assegurar o alinhamento da arquitetura lógica de nível de processo com as necessidades específicas de domínio ou com os modelos de casos de uso, o modelo V sugere a execução do método 4SRS.

Tradicionalmente, o método 4SRS é aplicado numa perspetiva de nível de produto, contudo recentemente surgiu um método adaptado e estendido do 4SRS que permite a aplicação numa perspetiva de nível de processo (Ferreira et al. 2012).

Nesta perspetiva de nível de processo, o 4SRS utiliza como *input* as representações (e correspondentes descrições textuais) dos casos de uso para transformar (recorrendo a transformações tabulares) as necessidades dos utilizadores (os casos de uso) em elementos arquiteturais (AEs) de forma a criar uma representação da arquitetura lógica do sistema.

A sua aplicação difere da tradicional pela definição de um conjunto de regras que devem ser consideradas quando se raciocina sobre a execução dos passos do método e pela existência de novos micro-passos. É também de destacar que a natureza do termo elemento arquitetural pode variar, mas no contexto específico de arquiteturas lógicas este refere-se às partes a partir do qual a arquitetura lógica final pode ser construída (Ferreira et al. 2012).

Uma vez que a descrição da aplicação do método 4SRS, numa perspetiva de nível de processo, é completamente realizada na literatura (Ferreira et al. 2012), e pode ser usado como descrito nessa obra, por razões de inteligibilidade, de seguida é apresentado apenas uma breve explicação da estrutura do método e da aplicação.

De forma a transformar os casos de uso em AEs, o método 4SRS está organizado em quatro passos que podem ser suportados por representações tabulares, onde cada coluna tem seu próprio significado e regras (consultar Figura 26) (Ferreira, Santos, Machado, et al. 2013). O

primeiro passo (criação de elementos arquiteturais) cria automaticamente três tipos de AEs para cada caso de uso: um do tipo i (interface), um do tipo c (controlo) e um do tipo d (dados). O segundo passo (eliminação de elementos arquiteturais) remove automaticamente a redundância dos requisitos herdados pelos casos de uso, e promove a descoberta de novos requisitos. O terceiro passo (agregação de elementos arquiteturais) agrupa semanticamente elementos arquiteturais em *packages* e permite representar agregações. Por fim, o passo 4 (associação de elementos arquiteturais) tem como objetivo representar as associações entre os elementos arquiteturais que "sobreviveram" aos passos anteriores.

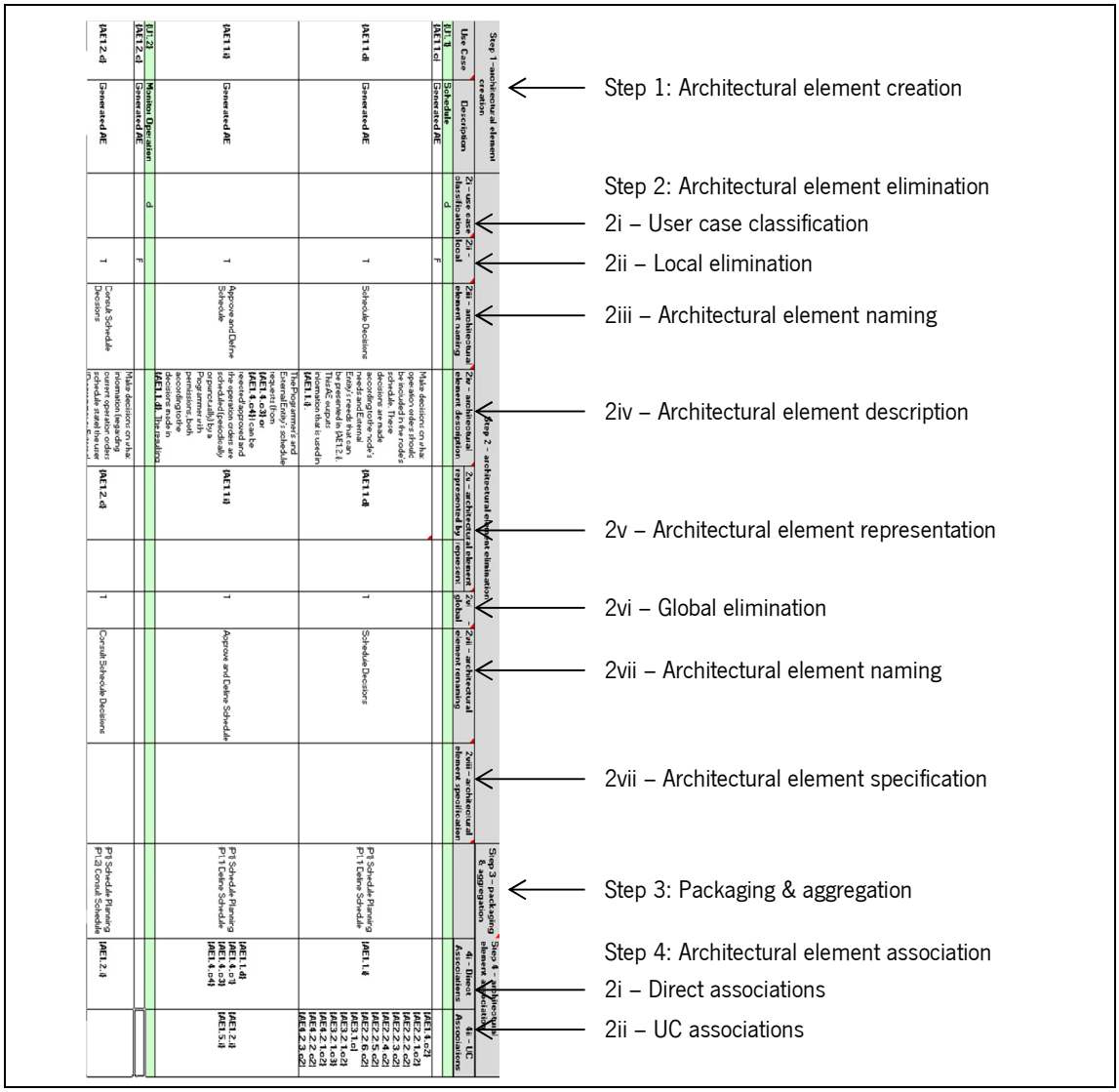


Figura 26 – Transformações tabulares do método 4SRS (adaptado de (Ferreira, Santos, Machado, et al. 2013)).

Para o caso prático deste trabalho, como *input* para o método 4SRS, foram utilizados os casos de uso correspondentes aos "folha" da estrutura em árvore do processo de decomposição (i.e. os casos de uso a cinzento na Figura 25), uma vez que representam a informação mais detalhada e completa alcançada do processo de identificação das necessidades de negócio.

Como resultado deste trabalho, foi criado um diagrama lógico que representa as funcionalidades da arquitetura lógica de nível processo da SLT Solution.

Após terminada a execução completa do 4SRS, algumas informações estatísticas sobre os AEs mantidos e eliminados foram possíveis de extrair das transformações tabulares. Conforme apresenta a Figura 27, os dados estatísticos revelam que os 39 casos de uso utilizados como *input* deram origem a 142 elementos arquiteturais, dos quais mantiveram-se 66 até o final das transformações tabulares. Estes dados que foram criados representam quase 40% de elementos arquiteturais a mais do que os casos de uso iniciais, ou seja, a execução do 4SRS permitiu uma representação mais detalhada e expressiva do sistema em estudo, ao mesmo tempo diminui a redundância e valida o diagrama de arquitetura lógica derivada.

		Total	Vivos	Mortos	% Mortos	% Vivos
2ii	Total de C's	59	41	18	31%	69%
	Total de D's	44	32	12	27%	73%
	Total de I's	39	28	11	28%	72%
		142	101	41	29%	71%
2vi	Total de C's	41	26	15	37%	63%
	Total de D's	32	20	12	38%	63%
	Total de I's	28	20	8	29%	71%
		101	66	35	35%	65%

Figura 27 – Dados estatísticos da execução do 4SRS.

No final da execução do 4SRS foi também construído uma representação das funcionalidades da arquitetura lógica de nível processo da SLT Solution. A arquitetura lógica derivada é composta pelos elementos da arquitetura que sobreviveram após a execução do passo 2. Os *packages* que resultaram da execução do passo 3 permitem a identificação dos principais processos. E as associações identificadas no passo 4 são representadas no diagrama pelas ligações entre os elementos da arquitetura.

Analisando a arquitetura lógica derivada (diagrama lógico da Figura 28), é possível concluir que existe uma interdependência visível entre os *macro-packages* da arquitetura lógica. Para além disso, é também verificada a existência de processos que apresentam uma dependência temporal ({P1} Schedule Planning, {P2} Operation Order Execution, {P3.2} Task Monitoring, {P4} Incident Communication e {P5} Assistances) e outros que são transversais a todo sistema ({P6} Configurations e {P3.1} Data Monitoring). Por um lado, a execução do processo transversal, {P6} Configurations, influência o comportamento de todos os outros processos do sistema. Por outro, um processo de dependência temporal, como o {P2} Operation Order Execution, é apenas uma

parte integrante de um processo que também estimulará o início de outros processos de dependência temporal, como o {P3.2} Task Monitoring e {P4.2} Detect Incidents.

Para os processos {P1} Schedule Planning, {P2} Operation Order Execution, {P3} Activity Monitoring, {P4} Incident Communication e {P5} Assistances a execução típica inicia com um *input* a partir de elementos arquiteturais referentes a decisões humanas, de seguida o fluxo de informação é tratado a partir dos elementos arquiteturais de decisões de sistema e finaliza com um *output* através de elementos arquiteturais de interfaces. Apenas, para o processo {P6} Configurations, a execução começa a partir das decisões humanas e vai diretamente para camada de apresentação, porque não há necessidade de qualquer decisão do SLT Solution numa perspetiva de nível de processo.

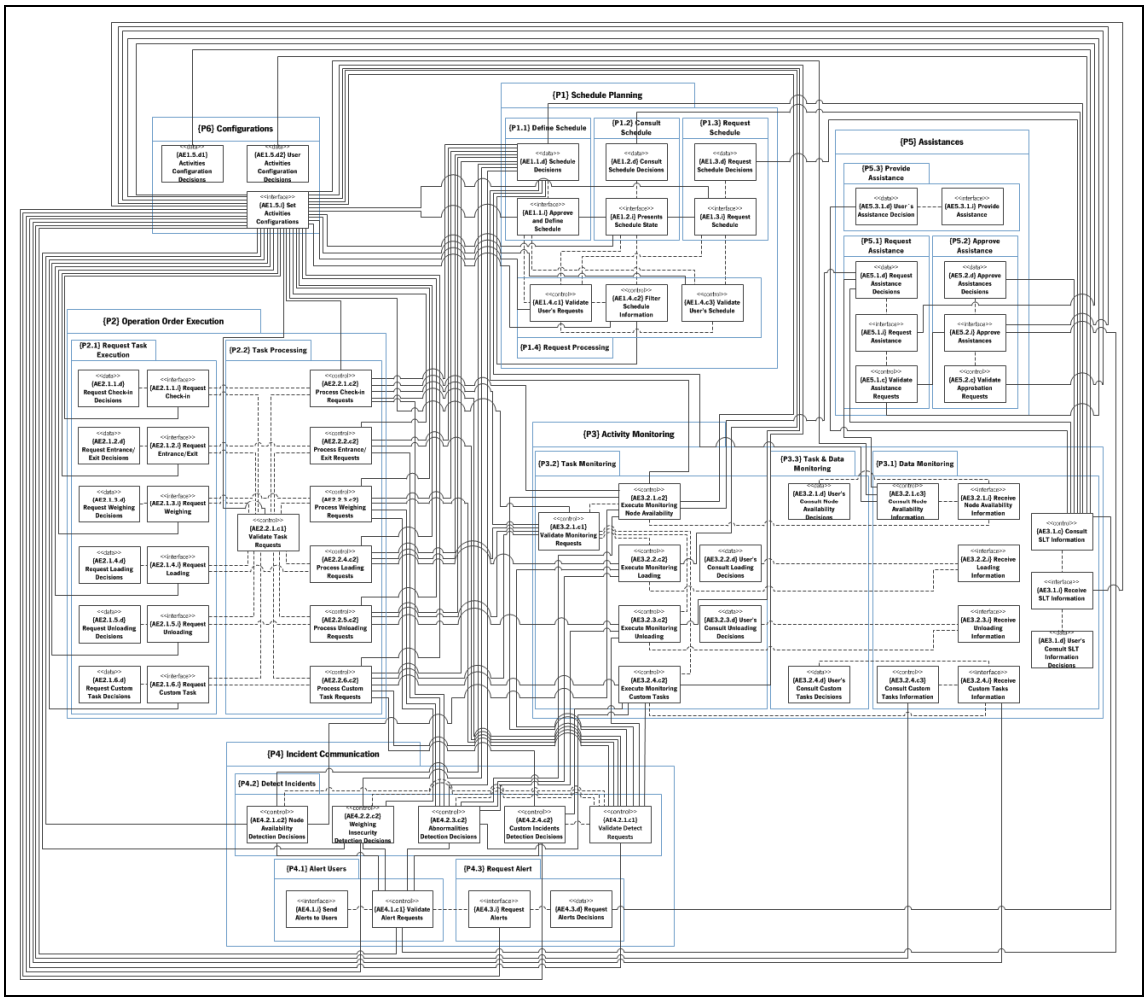


Figura 28 – Arquitetura lógica de nível-processo SLT.

4.3.2 Diagramas de Sequência do Tipo B

Nesta fase final do modelo V procede-se à definição dos processos relativos à arquitetura lógica derivada na secção 4.3.1. Na presente subsecção elucida-se então, o modo como foram abordados e definidos esses processos, e apresenta-se ainda o conjunto alcançado dos mesmos.

De modo a assegurar uma correta definição dos processos relativos à arquitetura lógica derivada, o modelo V sugere a realização de uma atividade de testes através da modelação de diagramas de sequência do tipo B.

Estes representam a troca de informações entre atores e elementos arquiteturais da arquitetura lógica e admitem duas distintas validações, nomeadamente: a validação da arquitetura lógica derivada e a validação das sequências, sendo estas últimas representadas pelos diagramas de sequência do tipo A. Estas validações compreendem, respetivamente, a deteção da falta de elementos arquiteturais e/ou associações para executar um determinado processo dentro de arquitetura lógica derivada (subsecção 4.3.1); e a verificação de uma relação conforme entre os fluxos de sequência associados aos casos de uso dos diagramas do tipo A e os fluxos de sequência associados aos elementos arquiteturais dos diagramas do tipo B.

Assim, durante o processo de modelação de diagramas do tipo B, devem-se considerar pelo menos três aspetos. Assinalam-se portanto as necessidades de assegurar que as sequências do processo modeladas para os diagramas do tipo B exponham os mesmos fluxos que as sequências de processo modeladas para os diagramas do tipo A; garantir a conformidade dos diagramas do tipo B com os elementos arquiteturais, *packages* e associações definidos na arquitetura lógica de nível de processo derivada; e por último, ter em conta o número de diagramas do tipo B a modelar, asseverando que todos os elementos arquiteturais são utilizados.

No âmbito do caso prático desta dissertação, foi modelado um conjunto de diagramas de sequência do tipo B com a intenção de representar as principais execuções de processos possíveis em relação à arquitetura lógica do SLT Solution (consultar secção B.4 em anexo). Estes diagramas de sequência foram modelados de forma estereotipada, utilizando os elementos arquiteturais, os atores e *packages* (i.e. diferem dos tradicionais).

A predileção pelos processos representados deriva dos cenários de execução previamente definidos, na secção 4.2.2, como críticos para a realização de negócio. Como exemplo, na Figura 29 são apresentadas as mesmas atividades (do diagrama de sequência do tipo A apresentado na subsecção 4.2.2) relacionadas com a realização de *check-in* e de verificação de

disponibilidade de um nó para a execução de ordens de operação, todavia num nível inferior de abstração.

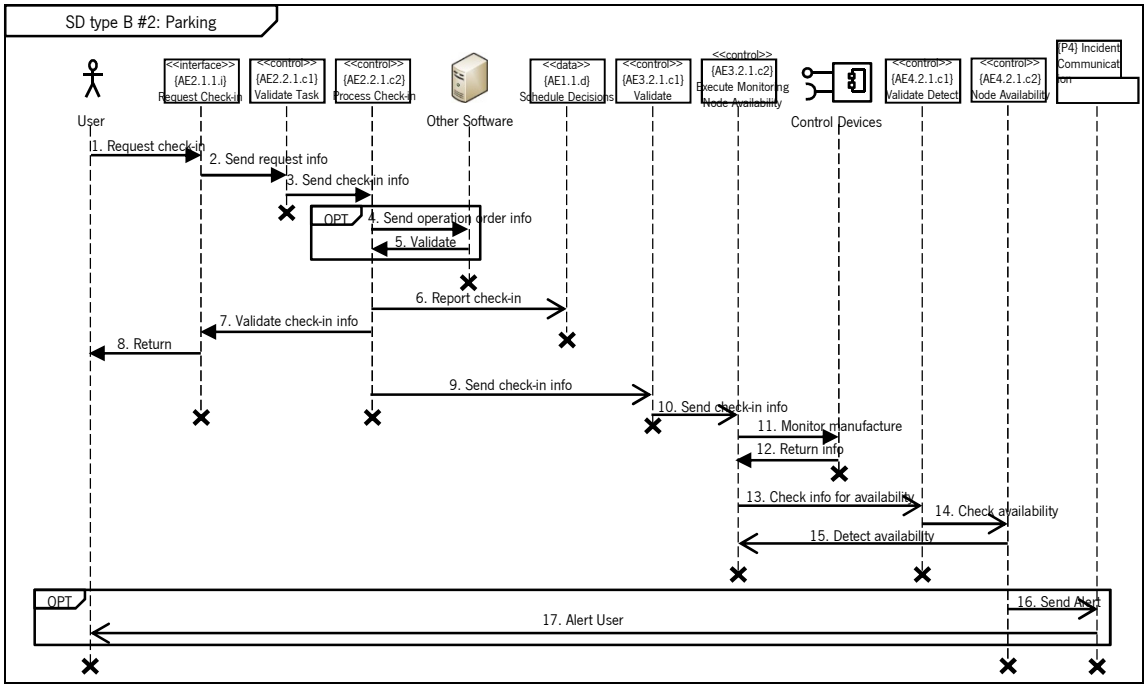


Figura 29 – Diagrama de sequência do tipo B, *parking*.

Com a conclusão desta fase, pululou a primeira versão do contexto para a conceção do sistema de software pretendido.

4.4 Aplicação do Padrão NIST

O principal objetivo da execução do modelo V (explicada nas últimas duas secções) foi a criação de um completo e preciso contexto para a conceção de um sistema de software alinhado com as necessidades dos SLT, independentemente da tecnologia de computação que suportará as funcionalidades do sistema. No entanto, no âmbito deste trabalho, o sistema de software pretendido deve ser baseado no modelo de computação, ou seja, este também deverá estar alinhado com as necessidades de computação em nuvem.

Assim, e com o objetivo de potenciar este último alinhamento, decidiu-se utilizar o modelo NIST (Liu et al. 2011), de forma semelhante à demonstrada por Pereira (2013), no seu contributo de investigação, com o intuito de descobrir os requisitos de computação em nuvem em falta nos modelos V do SLT. Pereira (2013) utilizou o modelo NIST ao nível dos casos de uso (i.e. no final da atividade de descrição das principais funcionalidades do sistema pretendido) e ao nível dos elementos arquiteturais (AEs) de uma arquitetura lógica de nível de processo (i.e. depois de

especificadas as principais atividades do sistema pretendido). Para este caso prático, foi decidido utilizar o modelo NIST ao nível das Configurações Organizacionais (COs) (secção 4.4.1) e ao nível dos AEs da arquitetura lógica de nível de processo (secção 4.4.2), depois de terminada a primeira iteração do modelo V. Decidiu-se aplicar o modelo NIST a estes dois modelos V (COs e AEs) por representarem os extremos em termos de detalhe sobre o que é o sistema pretendido. Durante esta secção será possível verificar a pertinência para a realização de uma nova iteração ao modelo V, utilizando a informação que resultou da utilização do modelo NIST, se for pretendido um sistema baseado no modelo de computação em nuvem.

4.4.1 Configurações Organizacionais SLT

De modo a iniciar o processo de identificação de requisitos de computação em nuvem (ao nível de processo), foi então realizada uma análise que seguiu uma lógica de cruzamento dos modelos V (COs e AEs) com os componentes do modelo do NIST. Este cruzamento foi possível através do mapeamento das descrições das COs com as descrições do modelo NIST. Esta análise foi efetuada individualmente a cada CO e teve como objetivo identificar características nas COs do SLT semelhantes às características do modelo NIST. A ilustração do resultado deste cruzamento pode ser visualizada na Figura 30.

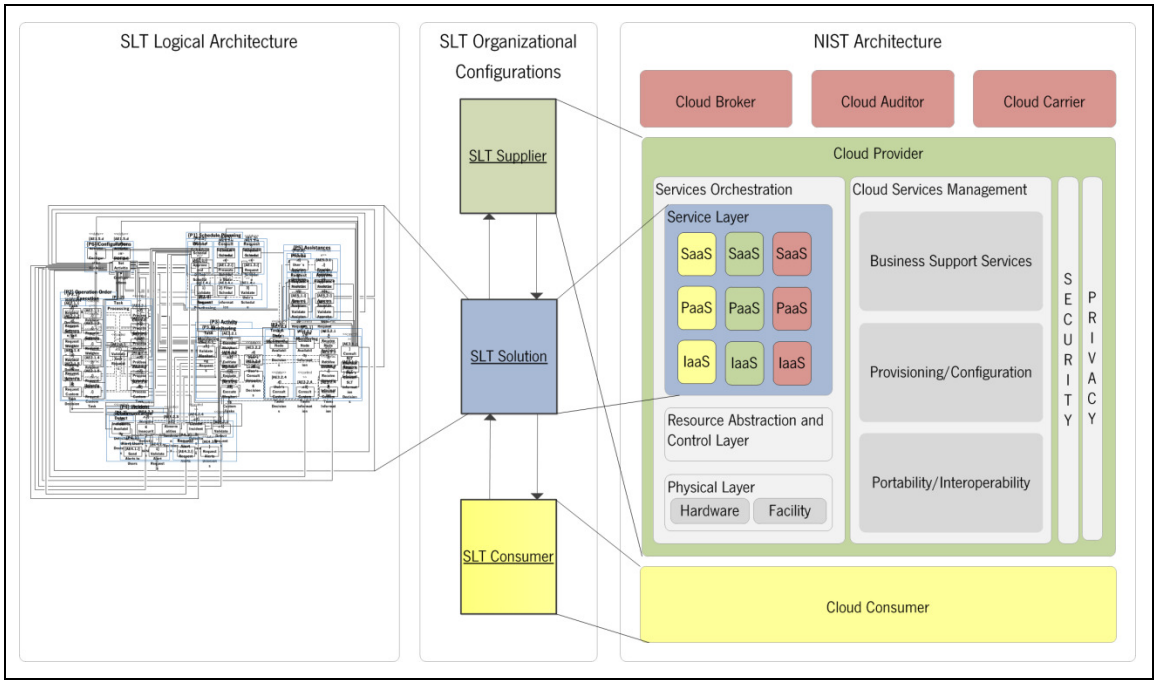


Figura 30 – Cruzamento Arquitetura Lógica SLT, Configurações Organizacionais SLT e Modelo NIST.

A Tabela 7 exemplifica em maior detalhe como foi efetuado o cruzamento das descrições das COs com as descrições do modelo NIST e apresenta o resultado da decisão de atribuição das atividades e funções dos atores do domínio SLT (COs) no modelo NIST. As descrições do modelo NIST, relacionadas com as atividades e funções dos atores de computação em nuvem, evidenciam as atividades e as funções relacionadas com os atores do domínio SLT. A identificação das COs do SLT, que se identificam com as características do modelo do NIST, é realizada com base nas descrições arquiteturais do modelo NIST. Com base nessas descrições é possível percorrer (um a um) os atores de domínio do SLT de forma a identificar a existência de atores de domínio do NIST com caraterísticas semelhantes. Por exemplo, para a associação Cloud Provider e SLT Supplier, é possível encontrar na respetiva descrição de Cloud Provider a frase "*entity responsible for making a service available to interested parties*", permitindo concluir que as atividades e funções deste ator estão relacionadas com as do SLT Supplier, podendo por isso ser mapeado no modelo do NIST. Seguindo esta lógica de mapeamento, os restantes atores de domínio SLT foram também submetidos a esta análise.

Tabela 7 – Modelo de Atribuição dos COs do SLT no modelo do NIST.

Atores de domínio do modelo NIST	Atores de domínio das COs
Cloud Provider - is the entity responsible for making a service available to interested parties.	**SLT Supplier** - provides the SLT services to consumers and has a role as a service supplier of the SLT Solution.
Cloud Consumer - represents a person or organization that maintains a business relationship with, and uses the service from a Cloud Provider.	**SLT Consumer** - directly uses the resources of the SLT Solution and has a role as a service consumer of the SLT Solution.
Service Layer - is where Cloud Providers define interfaces for Cloud Consumers to access the computing services	**SLT Solution** - represents the service execution, standing between the SLT Consumer's requests and the SLT Supplier's responses.

Como resultado desta análise, foi possível verificar que existem interações de computação em nuvem por tratar nas COs. O SLT Solution[13] (ou o Service Layer) inclui atividades de interação com os atores Cloud Provider e Cloud Consumer, mas não inclui para os atores Cloud Broker, Cloud Auditor e Cloud Carrier. Este facto permite, como se esperava, concluir que é necessária uma nova iteração ao modelo V se for pretendido um sistema alinhado com as necessidades do modelo de computação em nuvem.

[13] De destacar que o SLT Solution e seu correspondente (Service Layer) não se referem a uma entidade mas sim um ambiente (composto de componentes de sistema) para a execução de serviços em nuvem, que está entre o fornecedor e o consumidor de nuvem.

4.4.2 Arquitetura Lógica SLT

Uma vez identificadas as inconsistências das COs do SLT para o contexto de computação em nuvem, pode-se concluir imediatamente que, para alinhar o sistema pretendido com as necessidades de computação em nuvem, é necessária uma reestruturação das COs e, consequentemente, a iniciação de uma nova iteração ao modelo V. Contudo, antes de terminar esta análise às incoerências, decidiu-se ainda realizar uma outra com o intuito de perceber a adequação do SLT Solution relativamente às necessidades dos atores, Cloud Provider e Cloud Consumer, do modelo NIST (i.e. os atores considerados durante a execução do modelo V).

A arquitetura lógica SLT de nível de processo num contexto de computação em nuvem deve representar um conjunto completo de atividades (SaaS, PaaS e IaaS) para satisfazer as necessidades de todos os atores do modelo NIST. A análise realizada neste capítulo permitirá avaliar se esta arquitetura tem um conjunto de serviços adequado para as funções dos atores Cloud Provider e Cloud Consumer. A Figura 31 pretende esclarecer o intuito desta análise.

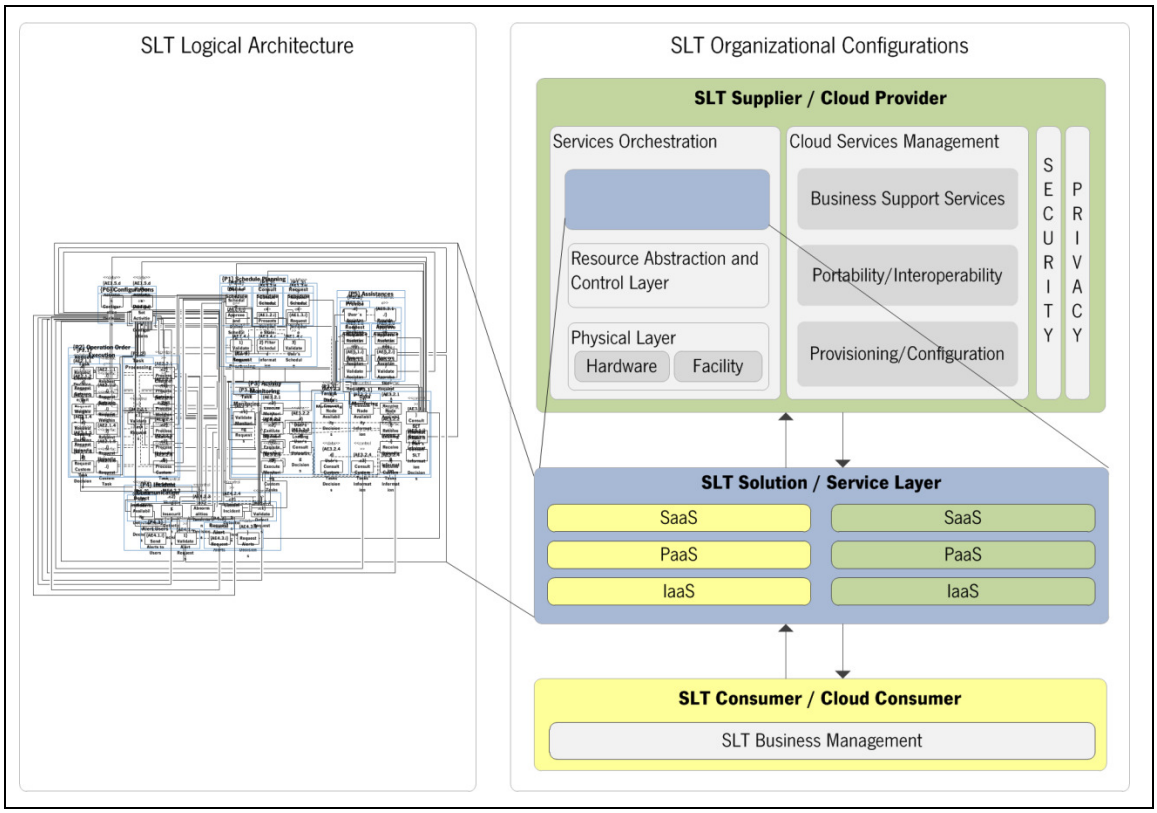


Figura 31 – Enquadramento Arquitetura Lógica SLT e Configurações Organizacionais SLT.

Assim, a análise realizada envolveu o estudo da arquitetura lógica SLT em função das atividades e funções consideradas pelo modelo de referência do NIST e seguiu uma lógica de cruzamento

da informação dada pelos elementos arquiteturais (AEs) da Arquitetura Lógica SLT, com a informação dada pelos elementos arquiteturais do modelo de referência do NIST.

O estudo iniciou com a atribuição dos AEs da arquitetura lógica SLT nos AEs do modelo de referência correspondentes, com semântica semelhante, de forma a verificar quais os AEs do SLT estão preparados para atuar em contextos de computação em nuvem, tendo em consideração o modelo de referência do NIST. O critério de atribuição foi suportado por uma análise baseada em tabelas com as descrições arquiteturais dos AEs do modelo de referência do NIST e dos AEs da Arquitetura Lógica SLT. Para além disso, a estas tabelas também foram incluídas a informação dos casos de uso do SLT para ter um maior detalhe do respetivo AE da arquitetura lógica SLT e consequentemente dar mais consistência à decisão de atribuir o AE da arquitetura lógica SLT no modelo de referência do NIST.

A Tabela 8 exemplifica o resultado da atribuição do AE {AE3.1.c} Consult SLT Information da arquitetura lógica SLT ao AE Reporting and Auditing do modelo de referência do NIST. Segundo as descrições arquiteturais do AE do modelo de referência do NIST, revela que este AE está relacionado com atividades de monitorização das operações do utilizador e de geração de relatórios.

Tabela 8 – Atribuição dos AE's da arquitetura lógica nos AE's do modelo NIST.

NIST Architectural Component	**NIST Architectural Element**	**SLT Architectural Element**	**SLT Related Use Case**
Business Support	**Reporting and Auditing** (Monitor user operations, generate reports, etc.)	**{AE3.1.c} Consult SLT Information** (Consult and make decisions regarding the SLT activities information in order to build reports.)	**{AE3.1.c}** Consult Schedule Planning Information; **{AE3.2.5.c}** Consult Operation Orders Information; **{AE3.3.c}** Consult Assistance Information; **{AE3.4.c}** Consult Incidents Information

Como resultado desta análise, foi possível verificar que a arquitetura lógica SLT quase na sua totalidade representa serviços (SaaS) de gestão de negócio SLT para os SLT Consumers e apenas tem uma correspondência semântica (a apresentada na Tabela 8) entre os AEs da arquitetura SLT e os AEs do modelo do NIST. Ou seja, muitos das funções de gestão e operação de serviços (exigidos e propostos pelos Cloud Consumers) associadas ao Cloud Provider não

foram consideradas durante a execução do modelo V. Para além disso, também se verificou que a arquitetura SLT não incorpora preocupações de segurança e privacidade relacionadas com a computação em nuvem. A falta de AEs relacionados com o modelo NIST poder ser justificada principalmente pela ingenuidade de considerar as relações entre o SLT Solution e o SLT Supplier como relações simples e diretas.

De forma a alinhar o SLT Solution com as necessidades de computação em nuvem associadas aos Cloud Providers, os requisitos de *Business Support* (i.e. gestão de clientes, gestão de contratos, gestão de inventário, cobrança e faturação), de *Provisioning and Configuration* (i.e. aprovisionamento rápido, monitorizar e reportar, mensuração, gestão de SLA e mudança de recursos), de *Portability and Interoperability* (i.e. portabilidade de dados, interoperabilidade de serviço e portabilidade de sistema), de segurança (i.e. autenticação, autorização, disponibilidade, confidencialidade, etc.) e de privacidade devem ser considerados numa futura iteração ao modelo V.

4.5 Conclusões

Este capítulo teve como principal objetivo esclarecer os principais resultados alcançados e todas as decisões realizadas durante o processo de formalização dos requisitos de nível de processo de um sistema de software alinhado com as necessidades dos SLT e computação em nuvem. De forma a concretizar este objetivo foi descrito o processo de utilização do primeiro V do processo V+V e do modelo de referência de computação em nuvem do NIST.

A descrição da utilização do primeiro V permite a compreensão do modo como foi alcançada uma arquitetura lógica de nível de processo de um sistema de software alinhado com as necessidades dos SLT. Esta descrição foi dividida em duas secções. A primeira envolveu a apresentação dos resultados obtidos e das decisões efetuadas durante a execução do modelo V, até à formalização das necessidades específicas dos SLT, ou seja, iniciou com a explicação do modo como foram identificadas as principais interações (i.e. configurações organizacionais) e os principais processos (i.e. diagramas de sequência do tipo A) do domínio dos SLT e terminou com a organização das atividades (i.e. diagramas de casos de uso) representados pelos mesmos diagramas de sequência do tipo A. A segunda secção também consistiu na explicação dos resultados obtidos e das decisões efetuadas, todavia para o restante processo de execução do primeiro V, até à criação de contexto para a conceção de produto. Esta segunda secção iniciou

com a explicação do modo como foi utilizado o método 4SRS para derivar uma arquitetura lógica SLT de nível de processo e terminou com a utilização dos diagramas de sequência do tipo B para assegurar uma correta definição dos processos relativos à arquitetura lógica SLT derivada. Ao analisar a utilização do primeiro V, é possível concluir que este método obriga a que cada fase seja verificada antes de avançar para a próxima, proporcionando uma definição mais correta. Os resultados da execução do primeiro V podem ser utilizados como uma base inicial para conceção de sistemas de software para contextos mais específicos do domínio dos SLT.

A descrição da utilização do modelo NIST potencia o entendimento da forma como foi aplicado o modelo NIST para identificar inconsistências nos modelos SLT relativamente às necessidades de computação em nuvem. Se não tivessem sido identificadas incoerências poder-se-ia assumir que a arquitetura lógica SLT derivada da execução do primeiro V estava alinhada com as necessidades de computação em nuvem, contudo não foi este o caso. A utilização do modelo NIST ao nível das configurações organizacionais permitiu identificar que nem todas as interações relativas ao modelo de computação foram consideradas. A aplicação do modelo NIST ao nível dos elementos arquiteturais da arquitetura lógica SLT permitiu identificar que esta também não retrata todas as atividades de computação em nuvem possíveis e úteis para os fornecedores de serviços de nuvem. A informação que resultou da aplicação do modelo NIST pode servir, num trabalho futuro, como *input* para a realização de uma nova iteração do primeiro V e, consequentemente, a criação de um contexto de conceção de produto mais preciso e completo de um sistema de software alinhado com os SLT e a computação em nuvem.

Este capítulo dá por concluída a explicação da execução de uma abordagem de CSI (definida e iniciada no capítulo 3). Acredita-se que a exemplificação da prossecução desta abordagem possa ser útil para organizações que oferecem ou pretendem oferecer soluções de software alinhadas com as necessidades de diversos domínios específicos e em particular para o domínio dos SLT e computação em nuvem.

CAPÍTULO 5 – CONCLUSÕES

Neste capítulo sintetiza-se as conclusões mais relevantes do presente trabalho, justificando a sua importância (secção 5.1), a aprendizagem daí resultante, e as dificuldades principais enfrentadas durante a sua realização (secção 5.2). Antes de terminar são ainda formuladas algumas propostas de trabalho futuro (secção 5.3).

5.1 Síntese

A computação em nuvem é uma das inovações da atualidade que está a revolucionar a indústria das TI. Nos últimos anos, tem vindo a aumentar o número de organizações fornecedoras de serviços e recursos de TI (i.e. empresas de software e departamentos de TI) a oferecer soluções baseadas no modelo de computação em nuvem (Stuckenberg, Fielt e Loser 2011). Ao lado das grandes empresas, como Amazon, Google, IBM ou Microsoft, emergem, cada vez mais, fornecedores de software que constroem as suas próprias aplicações ou serviços de consultoria sobre serviços de TI oferecidos por outros operadores no mercado (Böhm et al. 2011). Contudo, as organizações, que ainda não adotaram este novo conceito, não devem assumir ações estratégicas de mudança sem compreenderem, claramente, as várias questões envolvidas, tanto ao nível de negócio como de tecnologia.

Foi nesta perspetiva que se justificou a realização do presente trabalho de investigação. Este deve ser considerado como um contributo para a atividade de definição estratégica das organizações fornecedoras de recursos e serviços de TI, na medida em que permite identificar um conjunto de abordagens que podem ser utilizadas para o DSI alinhados com as necessidades de computação em nuvem, facilita a escolha da melhor estratégia, conduzindo, consequentemente, à obtenção de melhores resultados.

Com vista à prossecução da finalidade estabelecida para este projeto delineou-se uma estratégia de três fases, em consonância com os objetivos formulados para este projeto. A primeira fase envolveu a revisão e análise de um conjunto de documentos bibliográficos relacionados com o conceito de computação em nuvem, com o intuito de construir uma base teórica de conhecimento capaz de suportar e apoiar todo o trabalho de investigação. A segunda compreendeu a identificação de diferentes abordagens de DSI, para a construção de um sistema de software baseado no modelo de computação em nuvem, a análise das abordagens

identificadas e a seleção das mais adequadas para o âmbito deste projeto e em particular, para o caso prático dos SLT. Por fim, a última fase, consistiu na exemplificação da utilização de abordagens de DSI para construir um sistema de software alinhado com as necessidades dos SLT e as necessidades de computação em nuvem.

A base de conhecimento criada, a identificação de abordagens de DSI para contextos de computação em nuvem e a exemplificação de um caso prático para os SLT constituem a contribuição principal decorrente de toda a investigação efetuada ao longo desta dissertação. Este trabalho apresenta várias questões potenciais, tanto ao nível de negócio como de tecnologia, a ocorrerem na prossecução de estratégias de adotação de computação em nuvem.

5.2 Discussão

Com o propósito de satisfazer a finalidade identificada para este projeto, procurou-se o cumprimento dos três objetivos inicialmente estabelecidos, nomeadamente: (O1) rever fundamentos e literatura, (O2) identificar, analisar e selecionar abordagens existentes para a conceção de um sistema de software baseado no modelo de computação em nuvem para o suporte a SLT, e (O3) exemplificar a utilização de abordagens para a conceção de um sistema de software alinhado com as necessidades de computação em nuvem e dos SLT.

Para o cumprimento do primeiro objetivo (O1) procedeu-se à revisão e análise de um conjunto de documentos bibliográficos relevantes sobre o domínio de computação em nuvem, de que resultaram numerosas contribuições. Uma dessas contribuições foi a delimitação da área de estudo, através do esclarecimento de três grandes áreas do conhecimento (Computação, Gestão de Organizações e Gestão de Sistemas de Informação). Outra foi a compreensão dos principais aspetos e do estado atual do principal objeto em estudo, a computação em nuvem. A última foi a identificação das principais oportunidades e desafios para o processo de adoção do modelo de computação em nuvem, tanto para os consumidores como para os fornecedores de serviços e recursos de TI. Terminada a realização deste objetivo, considera-se este como fundamental para construção de uma base de conhecimento capaz de suportar e apoiar a realização dos restantes objetivos (O2 e O3). Apesar das dificuldades encontradas na revisão efetuada, decorrentes da forma algo desorganizada e incoerente como muitos dos termos e conceitos de computação em nuvem são utilizados, julga-se que o primeiro objetivo proposto para este trabalho foi plenamente alcançado.

O segundo objetivo foi concretizado seguindo as sugestões de Carvalho (1996) relativamente à atividade de DSI e através da revisão e análise de documentos bibliográficos sobre conceitos associados aos SLT e às abordagens de engenharia de requisitos. A utilização das sugestões de Carvalho (1996) permitiu a definição, de forma justificada, de uma estratégia para a construção de um sistema de software alinhado com as necessidades do domínio da computação em nuvem e do domínio dos SLT. Por sua vez, a revisão e análise de documentos bibliográficos sobre as organizações com processos de SLT e sobre o estado atual do mercado para as soluções entregues com um serviço para o contexto dos SLT possibilitaram a compreensão das necessidades das organizações com SLT e a comprovação da pertinência de conceber uma solução de software entregue como um serviço para o domínio dos SLT. Por fim, a revisão e análise de documentos bibliográficos sobre abordagens de engenharia de requisitos proporcionou a identificação, análise e seleção de diferentes abordagens para a conceção de sistemas de software alinhado com as necessidades dos SLT e de computação em nuvem. As abordagens selecionadas, e que se consideraram ser mais adequadas para a exemplificação do terceiro objetivo (O3), foi o processo V+V de Ferreira, Santos, Machado, et al. (2013) e o modelo de referência de computação em nuvem do NIST (Liu et al. 2011). As dificuldades encontradas na realização deste objetivo são as mesmas que foram apresentadas para o primeiro objetivo (O1) e apesar da ocorrência dessas, crê-se que este objetivo (O2) foi plenamente alcançado e a realização deste objetivo constituiu a primeira contribuição concreta para a satisfação da finalidade desta dissertação.

O terceiro e último objetivo estabelecido para esta dissertação foi parcialmente atingido, devido à complexidade inesperada na execução de todo o processo V+V. No início do projeto esperava-se conseguir exemplificar a utilização de todo o processo V+V, todavia foi apenas possível realizar a exemplificação da execução do primeiro V. Apesar desta dificuldade, foi possível tirar um conjunto de contribuições relativamente à utilização da execução do primeiro V e a utilização do modelo NIST. A utilização do primeiro V permitiu a compreensão do modo como pode ser alcançada uma arquitetura lógica de nível de processo de um sistema de software alinhado com as necessidades de um problema, podendo ser relativo a qualquer contexto, como por exemplo, a computação em nuvem e/ou os SLT. A utilização do modelo NIST possibilitou o entendimento sobre como pode ser potenciado o alinhamento das funcionalidades de um sistema de software com as necessidades de computação em nuvem, através da identificação de inconsistências nos modelos do processo V+V. Face ao trabalho realizado e aos resultados obtidos julga-se que o

terceiro objetivo, apesar de ser parcialmente concretizado, contribui de forma significativa para a finalidade deste projeto de investigação.

Após a concretização dos três objetivos deste trabalho concluiu-se que é bastante útil e vantajoso para as organizações fornecedoras de serviços e recursos de TI reconhecerem as várias questões envolvidas no processo de construção de um sistema de software baseado no modelo de computação em nuvem, pois a decisão de mudar para o novo modelo de computação pode por si só provocar onerosos problemas às mesmas.

5.3 Limitações e Trabalho Futuro

Este estudo tendo-se debruçado, predominantemente (ou por razões temporais ou logísticas, ou por não fazerem parte do enquadramento imposto pelos objetivos inicialmente estabelecidos), sobre a análise de questões técnicas relacionadas com a adoção e implementação de computação em nuvem, não abarcou outro tipo de assuntos e realidades considerados relevantes para a discussão da adotação de computação em nuvem.

Para além das questões, abordadas nesta dissertação, sobre a forma como podem ser construídas soluções baseadas no modelo de computação, há uma necessidade igualmente pertinente para a realização de estudos sobre outros assuntos, particularmente relacionados com o negócio da computação em nuvem (ex. questões sobre modelos de negócio de fornecedores de software, cultura organizacional, políticas de SI, legalização, regulamentação, etc.), de forma a permitir uma avaliação mais precisa dos riscos envolvidos na adoção do modelo de computação em nuvem.

Assim, como propostas de trabalho futuro podem ser sugeridas duas, uma que deriva da extensão natural ao estudo realizado e outra das limitações do próprio projeto. Relativamente à primeira proposta, de modo a dar continuidade a este contributo de investigação, sugere-se a continuação da execução da abordagem de CSI adotada neste projeto e a avaliação da sua utilidade para diferentes tipos de projetos de TI. Sobre a segunda proposta, sugere-se o desenvolvimento de metodologias para a avaliação dos riscos envolvidos na adoção de computação em nuvem.

REFERÊNCIAS BIBLIOGRÁFICAS

Alter, Steven. 2002. The Work System Method for Understanding Information Systems and Information Systems Research. *Communications of the AIS* 9 (6):90-104.

Amaral, Luis. 1994. PRAXIS: Um Referencial para o Planeamento de Sistemas de Informação. Dissertação de Doutoramento, Departamento de Sistemas de Informação, Universidade do Minho, repositoriUM.

Amaral, Luis. 2005. Da Gestão ao Gestor de Sistemas de Informação: Expectativas fundamentais no desempenho da profissão.

American Marketing Association. *Franchising e Franchise* 2012 (Acedido em 8 de Fevereiro de 2013) - http://www.marketingpower.com/_layouts/dictionary.aspx?dLetter=F.

Anthony, R.N. 1965. *Planning and Control: A Framework for Analysis*. Cambridge, MA: Harvard University Press.

Armbrust, M., A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin e I. Stoica. 2010. A View of Cloud Computing. *Communications of the ACM* 53 (4):50-58.

Armbrust, M., A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica e M. Zaharia. 2009. Above the Clouds: A Berkeley View of Cloud Computing. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28*.

Azevedo, Sofia, Ricardo J Machado, Dirk Muthig e Hugo Ribeiro. 2009. Refinement of Software Product Line Architectures through Recursive Modeling Techniques. *On the Move to Meaningful Internet Systems: OTM 2009 Workshops*.

Bahl, P., R. Y. Han, L. E. Li e M. Satyanarayanan. 2012. Advancing the State of Mobile Cloud Computing. *Proceedings of the third ACM workshop on Mobile cloud computing and services*.

Berry, Daniel M. 2003. From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity.

Buckingham, R. A., R. A. Hirschheim, F. F. Land e C. J. Tully. 1987. Information Systems Curriculum: a Basis for Course Design. *Information Systems Education: Recommendations and Implementation*.

Buyya, R., C.S. Yeo e S. Venugopal. 2008. Market-oriented cloud computing: Vision, hype e reality for delivering it services as computing utilities.

Buyya, R., C.S. Yeo, S. Venugopal, J. Broberg e I. Brandic. 2009. Cloud computing and emerging IT platforms: Vision, hype e reality for delivering computing as the 5th utility. *Future Generation Computer Systems* 25 (6):599-616.

Böhm, M., S. Leimeister, C. Riedl e H. Krcmar. 2011. Cloud Computing–Outsourcing 2.0 or a new Business Model for IT Provisioning? *Application Management*.

Cachapuz Bilanciai Group. *SLV Cement*. - http://www.slvcement.com/.

Carr, N. G. 2005. The end of corporate computing. *MIT Sloan Management Review* 46 (3):67-73.

Carvalho, J. A. 2000. Information system? Which one do you mean. *Information system concepts: an integrated discipline emerging: IFIP TC8/WG8. 1 International conference on information system concepts (4th 1999: University of Leiden). Kluwer Academic Publishers.*

Carvalho, J. A. e L. Amaral. 1993. Matriz de Actividades: Um enquadramento Conceptual para as Actividades de Planeamento e Desenvolvimento de Sistemas de Informação. *Sistemas de Informação* 1:37-48.

Carvalho, João Álvaro. 1996. Desenvolvimento de Sistemas de Informação: Da Construção de Sistemas Informáticos à Reengenharia Organizacional.

Checkland, Peter. 2000. Soft Systems Methodology: A Thirty Year Retrospective. *Systems Research and Behavioral Science* 17:S11-S58.

Chou, David C. e Amy Y. Chou. 2007. Software as a Service (SaaS) as an outsourcing model: An economic analysis.386-391.

Christopher, Martin. 2011. *Logistics and Supply Chain Management: Creating Value-Adding Networks*. Quarta Edição ed: Pearson Education.

Council of Supply Chain Management Professionals. *CSCMP Supply Chain Management* 2013 (Acedido em 17 de Fevereiro de 2013) - http://www.clm1.org/about-us/supply-chain-management-definitions.

Cruz, Estrela F., Ricardo J. Machado e Maribel Y. Santos. 2012. On the decomposition of use case diagrams for the refinement of requirements

Dictionary.com. *Computation* 2012 (Acedido em 26 de Dezembro de 2012) - http://dictionary.reference.com/browse/computation.

Ferreira, Nuno, Nuno Santos, Ricardo J Machado e Dragan Gašević. 2012. Derivation of Process-Oriented Logical Architectures: An Elicitation Approach for Cloud Design. In *Product-Focused Software Process Improvement*: Springer.

Ferreira, Nuno, Nuno Santos, Ricardo J Machado e Dragan Gašević. 2013. Aligning Domain-Related Models for Creating Context for Software Product Design. In *Software Quality. Increasing Value in Software and Systems Development*: Springer.

Ferreira, Nuno, Nuno Santos, Pedro Soares, Ricardo Machado e Dragan Gašević. 2013. Transition from Process-to Product-Level Perspective for Business Software. In *Enterprise Information Systems of the Future*: Springer.

Fox, Armando. *Cloud computing, law enforcement and business continuity* 2009 (Acedido em 8 de Fevereiro de 2012) - http://berkeleyclouds.blogspot.pt/2009/04/cloud-computing-law-enforcement-and.html.

Francis, L. 2009. Cloud Computing: Implications for Enterprise Software Vendors (ESV), System Design and Management Program, Massachusetts Institute of Technology.

Gartner Group. 2012. Forecast Overview: Public Cloud Services. *Gartner Research*.

Gartner Group. 2012. Gartner's Hype Cycle Special Report for 2012. *Gartner Research*.

Ghiani, Gianpaolo, Gilbert Laporte e Roberto Musmanno. 2004. *Introduction to Logistics Systems Planning and Control*: Wiley.

Grossman, Robert L. 2009. The Case for Cloud Computing. *IEEE Computer Society*:23-27.

Grossman, Robert L. 2012. *The Structure of Digital Computing: From Mainframes to Big Data*: Open Data Press LLC.

Henderson, J. C. e N. Venkatraman. 1993. Strategic alignment: Leveraging information technology for transforming organizations. *IBM systems journal* 32 (1).

Hitt, M. A., R. D. Ireland e R. E. Hoskisson. 2012. *Strategic Management: Competitiveness and Globalization*: South-Western Pub.

Hopcroft, J. E., R. Motwani e J. D. Ullman. 1979. *Introduction to Automata Theory, Languages, and Computation*. Vol. 2: Addison-wesley Reading, MA.

Horan, Jeanette A. 2011. The Essential CIO. IBM Corporation.

Infópedia. *Computar* 2012 (Acedido em 26 de Dezembro de 2012) - http://www.infopedia.pt/pesquisa-global/computar.

Infópedia. *Computação* 2012 (Acedido em 26 de Dezembro de 2012) - http://www.infopedia.pt/pesquisa-global/computa%C3%A7%C3%A3o.

Karpio, Krzysztof, Arkadiusz Orłowski e Mateusz Przybyć. 2010. Logistics in Information Systems. *Information Systems in Management IV*.

Katzan Jr, H. 2010. On An Ontological View of Cloud Computing. *Journal of Service Science (JSS)* 3 (1).

Kotonya, Gerald e Ian Sommerville. 1996. Requirements engineering with viewpoints. *Software Engineering Journal* 11 (1):5-18.

Kruchten, Philippe. 2004. *The Rational Unified Process: An Introduction*.

Liu, Fang, Jin Tong, Jian Mao, Robert Bohn, John Messina, Lee Badger e Dawn Leaf. 2011. NIST Cloud Computing Reference Architecture. *NIST special publication* 500:292.

Luftman, J. 2000. Assessing Business-IT Alignment Maturity. *Communications of the Association for Information Systems* 4.

Machado, Ricardo J, Kristian Bisgaard Lassen, Sérgio Oliveira, Marco Couto e Patrícia Pinto. 2007. Requirements Validation: Execution of UML Models with CPN Tools. *International Journal on Software Tools for Technology Transfer* 9 (3-4):353-369.

March, S. T. e G. F. Smith. 1995. Design and Natural Science Research on Information Technology. *Decision support systems* 15 (4):251-266.

Marston, S., Z. Li, S. Bandyopadhyay, J. Zhang e A. Ghalsasi. 2011. Cloud computing—The business perspective. *Decision Support Systems* 51 (1):176-189.

Martin, J. C. 2003. *Introduction to Languages and the Theory of Computation*. Fourth ed: McGraw-Hill.

Mell, P. e T. Grance. 2011. The NIST Definition of Cloud Computing. *NIST special publication* 800 (145).

Miller, Michael. 2008. *Cloud computing: Web-Based Applications That Change the Way You Work and Collaborate Online*: Que Publishing.

Mäkilä, Tuomas, Antero Järvi, Mikko Rönkkö e Jussi Nissilä. 2010. How to Define Software-as-a-Service–An Empirical Study of Finnish SaaS Providers. *Software Business*:115-124.

Nag, R., D. C. Hambrick e M. J. Chen. 2007. What is strategic management, really? Inductive derivation of a consensus definition of the field. *Strategic Management Journal* 28 (9):935-955.

O'Brien, J.A. 1993. Management Information Systems: A Managerial End User Perspective. edited by R. D. Irwin. Homewood, Illinois.

Paetsch, Frauke, Armin Eberlein e Frank Maurer. 2003. Requirements Engineering and Agile Software Development. *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on*.

Patel, Nandish V. 1995. Application of soft systems methodology to the real world process of teaching and learning. *International Journal of Educational Management* 9 (1):13-23.

Pathak, Rajesh. 2012. How IT Departments Benefited by SaaS. *International Journal of Research in Science And Technology*.

Pereira, Manuel. 2013. Conceção de Arquiteturas para Cloud Computing: Casos de Demonstração da Utilização do Modelo de Referência do NIST, Universidade do Minho.

Philipson, Graeme. 2003. Time Machine: A Brief History of Computing. *Australian Personal Computer*.

Plummer, D. C., T. J. Bittman, T. Austin, D. W. Cearley e D. M. Smith. 2008. Cloud Computing: Defining and Describing an Emerging Phenomenon. *Gartner, June* 17.

Rean. *Surge Computing/Hybrid Computing* 2009 (Acedido em 8 de Fevereiro de 2012) - http://berkeleyclouds.blogspot.pt/2009/05/surge-computing.html.

Rimal, B. P., E. Choi e I. Lumb. 2009. A Taxonomy and Survey of Cloud Computing Systems. *Fifth International Joint Conference on INC, IMS and IDC, 2009*.

Sahoo, M. 2009. IT Innovations: Evaluate, Strategize e Invest. *IT professional* 11 (6):16-22.

Salgado, Carlos, Ricardo J. Machado e Rita Suzana. 2013. Modeling the Alignment between Business and IS/IT: a Requirements Engineering Perspective.

Schubert, L., K. Jeffery e B. Neidecker-Lutz. 2010. *The Future of Cloud Computing: Opportunities for European Cloud Computing Beyond 2010*: European Commission, Information Society and Media.

Shackelford, Russell, H. James, Gordon Davies, John Impagliazzo, Reza Kamali, Richard LeBlanc, Barry Lunt, Andrew McGettrick, Robert Sloan e Heikki Topi. 2006. Computing Curricula 2005. ACM e IEEE.

SIIA, Software & Information Industry Association. 2000. Software as a Service: Strategic Backgrounder.

Soares, D. S. 1998. Planeamento de Sistemas de Informação: Estudo das Variáveis que Condicionam a Sua Estratégia de Execução, Departamento de Sistemas de Informação, Universidade do Minho, repositoriUM.

Stoner, J. A. F., R. E. Freeman e D. R. Gilbert. 1996. *Management*.

Stuckenberg, S., E. Fielt e T. Loser. 2011. The Impact Of Software-As-A-Service On Business Models Of Leading Software Vendors: Experiences From Three Exploratory Case Studies. *Proceedings of the 15th Pacific Asia Conference on Information Systems (PACIS 2011)*.

Sweeney, Edward. 2005. Perspectives on Supply Chain Management and Logistics Definitions.

Sääksjärvi, Markku, Aki Lassila e Henry Nordström. 2005. Evaluating the Software as a Service Business Model: From CPU Time-sharing to Online Innovation Sharing. *IADIS International Conference e-Society*.

TechTarget. *Metered services (pay-per-use)* 2005 (Acedido em 26 de Dezembro de 2012) - http://searchcio.techtarget.com/definition/metered-services.

TechTarget. *Unified Computing System* 2009 (Acedido em 29 de Dezembro de 2012) - http://searchdatacenter.techtarget.com/definition/unified-computing-system-UCS.

TheFreeDictionary.com. *Legacy system* 2012 (Acedido em 29 de Dezembro de 2012) - http://encyclopedia.thefreedictionary.com/Legacy+system.

TheFreeDictionary.com. *Service Level Agreement* 2012 (Acedido em 29 de Dezembro de 2012) - http://encyclopedia.thefreedictionary.com/Service+Level+Agreement.

Vaishnavi, V. K. e W. Kuechler. *Design Science Research in Information Systems*, 30 de Setembro de 2011 2004. - http://www.desrist.org/desrist.

Vaquero, L. M., L. Rodero-Merino, J. Caceres e M. Lindner. 2008. A Break in the Clouds: Towards a Cloud Definition. *ACM SIGCOMM Computer Communication Review* 39 (1):50-55.

Watson, Richard T. 2007. *Information Systems*: Global Text Project.

Webster, J. e R. T. Watson. 2002. Analyzing the Past to Prepare for the Future: Writing a Literature Review.

Wikicfp.com. *A Wiki for Calls For Papers* 2012 (Acedido em 30 de Dezembro de 2012) - http://www.wikicfp.com/cfp/.

Wikipedia contributors. *Uno-actu-Prinzip*. Wikipedia, The Free Encyclopedia. 2012 (Acedido em 12 de Fevereiro de 2012) - http://de.wikipedia.org/w/index.php?title=Uno-actu-Prinzip&oldid=109194015.

Wikipedia contributors. *Multitenancy*. Wikipedia, The Free Encyclopedia. 2013 (Acedido em 12 de Fevereiro de 2013) - http://en.wikipedia.org/w/index.php?title=Multitenancy&oldid=521455557.

Wikipedia contributors. *On-premises software*. Wikipedia, The Free Encyclopedia. 2013 (Acedido em 12 de Fevereiro de 2013) - http://en.wikipedia.org/w/index.php?title=On-premises_software&oldid=528086518.

Wilson, Rosalyn. 2012. *23rd Annual State of Logistics Report: The Long and Winding Recovery*. Council of Supply Chain Management Professionals.

Wu, L., S. K. Garg e R. Buyya. 2011. SLA-based Resource Allocation for Software as a Service Provider (SaaS) in Cloud Computing Environments. *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*.

Youseff, L., M. Butrico e D. Da Silva. 2008. Toward a Unified Ontology of Cloud Computing. *Grid Computing Environments Workshop, 2008. GCE'08*.

ANEXO A – MATRIZ DE CONCEITOS

Conceitos															
				Basilares			Computação em Nuvem				A utilidade de Computação em Nuvem				
Referências	Título	Citações	Ano	Computação	Gestão das Organizações	Gestão de Sistemas de Informação	Origem e a Inovação	Definição	Descrição	Atual Visão Mundial	Serviços de Computação Utilitária	Serviços de Aplicação como um Serviço	Sistemas Logísticos de Transportes	Observações
(March and Smith 1995)	Design and Natural Science Research on Information Technology	1561	1995	(252)										Conceito de tecnologia
(Grossman 2012)	The Structure of Digital Computing: From Mainframes to Big Data	0	2012	x			x							Evolução de computação
(Grossman 2009)	The Case for Cloud Computing	204	2009									x		
(Philipson 2003)	Time Machine: A Brief History of Computing	2	2003	x										História de computação
(Hopcroft, Mchwani, and Ullman 1979)	Introduction to Automata Theory, Languages, and Computation	13424	1979	x										
(Martin 2003)	Introduction to Languages and the Theory of Computation	313	2003	x										
(Shackelford et al. 2006)	Computing Curricula 2005	3	2006	x										Perspetivas de computação
(Stoner, Freeman, and Gilbert 1996)	Management	1982	1996		x									Conceitos,Evolução de organização e gestão
(Soares 1998)	Planeamento de Sistemas de Informação: Estudo das Variáveis que Condicionam a Sua Estratégia de Execução	2	1998		x	x								Conceitos,Evolução de organização, gestão e sistemas de informação
(Anthony 1965)	Planning and Control: A Framework for Analysis	2779	1965		x									Niveis de gestão
(O'Brien 1993)	Management Information Systems: A Managerial End User Perspective	872	1993		x									Niveis de gestão e informação
(Nag, Hambrick, and Chen 2007)	What is strategic management, really? Inductive derivation of a consensus definition of the field	201	2007		(944)									Gestão Estratégica
(Hitt, Ireland, and Hoskisson 2012)	Strategic Management: Competitiveness and Globalization	1660	2012		x									Processo e estado atual de Gestão Estratégica
(Amaral 1994)	PRAXIS: Um Referencial para o Planeamento de Sistemas de Informação	122	1994			(24, 25 e 36)								
(Watson 2007)	Information Systems	0	2007			x								
(Henderson and Venkatraman 1993)	Strategic alignment: Leveraging information technology for transforming organizations	2420	1993			x								
(Luftman 2000)	Assessing Business-IT Alignment Maturity	525	2000			x								
(Carvalho 2000)	Information system? Which one do you mean	24	2000			x								Conceito de Sistema de Informação
(Bahl et al. 2012)	Advancing the State of Mobile Cloud Computing	1	2012				x							
(Böhm et al. 2011)	Cloud Computing--Outsourcing 2.0 or a new Business Model for IT Provisioning?	17	2011				x	(7)						
(Vaquero et al. 2008)	A Break in the Clouds: Towards a Cloud Definition	1179	2008				x	(2)						
(Francis 2009)	Cloud Computing: Implications for Enterprise Software Vendors (ESV)	0	2009				x							
(Youseff, Butrico, and Da Silva 2008)	Toward a Unified Ontology of Cloud Computing	408	2008				x	(1)	x		x	x		
(Stuckenberg, Fell, and Loser 2011)	The Impact Of Software-As-A-Service On Business Models Of Leading Software Vendors	5	2011				x					x		
(Mell and Grance 2011)	The NIST Definition of Cloud Computing	1347	2011					(6)	x					
(Armbrust et al. 2009)	Above the Clouds: A Berkeley View of Cloud Computing	111	2009					(1)			x	x		
(Buyya et al. 2009)	Cloud computing and emerging IT platforms	1330	2009					(3)						
(Buyya, Yeo, and Venugopal 2008)	Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities	823	2008								x			
(Rimal, Choi, and Lumb 2009)	A Taxonomy and Survey of Cloud Computing Systems	237	2009					x			x			
(Katzan Jr 2010)	On An Ontological View of Cloud Computing	18	2010					x						
(Schubert, Jeffery, and Neidecker-Lutz 2010)	The Future of Cloud Computing: Opportunities for European Cloud Computing Beyond 2010	58	2010					x						
(Gartner 2012)	Forecast Overview: Public Cloud Services	0	2012							x				
(Gartner 2012)	Gartner's Hype Cycle Special Report for 2012	0	2012							x			x	
(Horan 2011)	The Essential CIO	0	2011							x				
(Carr 2005)	The end of corporate computing	172	2005								x			
(Plummer et al. 2008)	Cloud Computing: Defining and Describing an Emerging Phenomenon	27	2008									x		
(Chou and Chou 2007)	Software as a Service (SaaS) as an outsourcing model: An economic analysis	5	2007									x		
(SIIA 2000)	Software as a Service: Strategic Backgrounder	17	2000									x	x	
(Saaksjärvi, Lassila, and Nordström 2005)	Evaluating the Software as a Service Business Model: From CPU Time-sharing to Online Innovation Sharing	55	2005									x		
(Pathak 2012)	How IT Departments Benefited by SaaS	0	2012									x		
(Mäkilä et al. 2010)	How to Define Software-as-a-Service--An Empirical Study of Finnish SaaS Providers	13	2010											
(Wu, Garg e Buyya 2011)	SLA-based Resource Allocation for Software as a Service Provider (SaaS) in Cloud Computing Environments	28	2011									x		
(Miller 2008)	Cloud computing: Web-Based Applications That Change the Way You Work and Collaborate Online	230	2008									x		
(Christopher 2011)	Logistics and Supply Chain Management: Creating Value-Adding Networks	4179	2011										(2 e 4)	Conceito de logística e cadeia de abastecimento
(Ghiani Laporte, and Musmanno 2004)	Introduction to Logistics Systems Planning and Control	203	2004										(1, 2 e 3)	Conceito sistema de logística e cadeia de abastecimento
(Sweeney 2005)	Perspectives on Supply Chain Management and Logistics Definitions	0	2005										x	
(Council of Supply Chain Management Professionals 2013)	CSCMP Supply Chain Management	0	2013										x	Conceito Gestão logistica e gestão de cadeia de abastecimento.
(Wilson 2012)	23rd Annual State of Logistics Report: The Long and Winding Recovery	0	2012										x	
(Jeston and Nelis 2006)	Business Process Management: Practical Guidelines to Successful Implementations	230	2006										(11)	Conceito de Business Process Management
(Aalst, Hofstede, and Weske 2003)	Business Process Management: A Survey	740	2003										(1)	Conceito de Business Process Management Systems
(Karpo, Ofowski, and Przybyć 2010)	Logistics in Information Systems	0	2010										x	

Figura A 1 – Matriz de Conceitos.

ANEXO B – RESULTADOS DO MODELO V

This appendix presents the results achieved in the process of deriving the SLT process-level logical architecture.

B.1 Organizational Configurations

Regarding organizational configurations was defined a set of five activity types which represent the kind of participation that actors possess in the SLT scope and a set of scenarios, composed by a description of the information exchange, where some interactions are defined.

B.1.1 Organizational Activities

#A: SLT Solution Services Activities - activities executed within the SLT Solution.

- #A.1: Planning Activities - activities intended to create suitable conditions for the smooth running of the business activities (described in #A.2: Business Activities) by scheduling the operations orders of a node.
- #A.2: Business Activities - activities regarding the operation orders execution of a node, i.e., all the necessary tasks for receive raw materials and/or dispatch final products (this is what the node provides to customers).
- #A.3: Business Analysis Activities - activities related with business performance monitoring. The aim of these activities is to determine solutions to improve the performance of business activities (described in #A.2: Business Activities).
- #A.4: Assistance Activities - activities which provide the necessary support for end-users in order to perform a proper and independent execution of its operations orders;
- #A.5: Incidents Activities - activities concerning the detection and communication of an abnormality or event situation on SLT Solution to users.

#B: Consumer Business Activities - activities regarding the business execution of a business domain entity (e.g. customers, suppliers, carriers of a node and the node);

#C: Supplier Services Management Activities - activities related to the installation, operation and maintenance of SLT services.

#D: Infrastructure Support and Maintenance Activities - typical activities related to the administration, management and monitoring of IT-infrastructures.

B.1.2 Actors

SLT Consumer: directly uses the resources of the SLT solution and has a role as a service consumer of the SLT solution. Normally represents a business domain entity (e.g. customers, suppliers, carriers of a node and the node). An SLT Consumer can be either a Business User (i.e. User, Programmer, Controller, Technician, Analyst and External Entity) or a Remote Business Program (i.e. Other Software Systems). The SLT Consumer naming convention is used as a generalization of the entities whose domain of interactions belong in the scope of consuming, for economic reasons, the functionalities exposed by the SLT solution. When the generalization is not applicable, this role is represented by the proper business actor.

SLT Solution: It is not considered as an entity, but as the execution environment for the functionalities. The SLT Solution represents the service execution, standing between the SLT Consumer's requests and the SLT Supplier's responses. The business actor that execute within the SLT Solution is the system administrator (Admin).

SLT Supplier: provides the SLT services to consumers and has a role as a service supplier of the SLT Solution. The services are made available in the SLT Solution through system administrator (Admin) configurations. This Admin can be a person contractually linked to a software company provider or to an SLT entity node.

B.1.3 Scenario 1: The SLT Consumer Requests an Existing Service Execution in the SLT Solution

This scenario describes when a Business User (SLT Consumer) requests a service execution based on already existing functionalities in the SLT Solution.

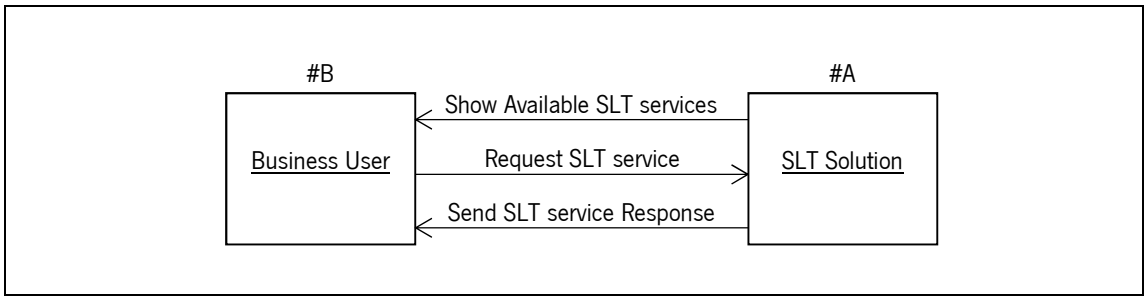


Figura A 2 – Organizational Configuration Scenario 1.

Relevant aspects of the interaction:

- The Business User must have permissions in order to have access to the available functionalities.

B.1.4 Scenario 2: Perform the SLT Solution Services and Infrastructure Maintenance

This scenario describes the SLT Solution's services and infrastructure maintenance process, executed by system administrator (Admin).

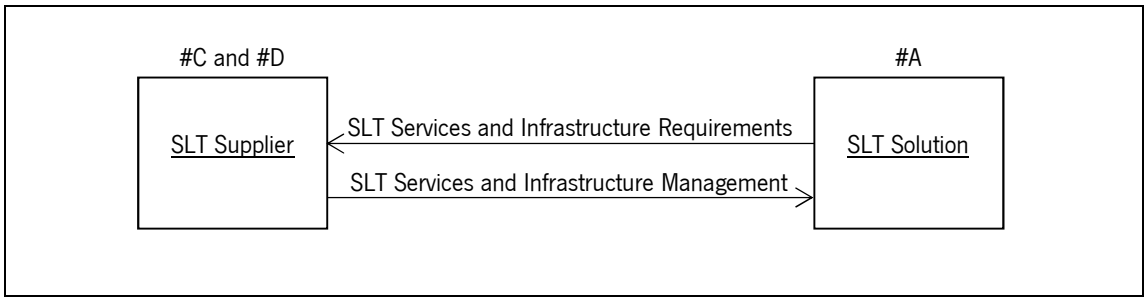


Figura A 3 – Organizational Configuration Scenario 2.

Relevant aspects of the interaction:

- The system administrator manages the services and the infrastructure of SLT Solution, but the cloud management-related decisions, like NIST requirements for business support, provisioning and configuration, portability and interoperability, are made outside the scope of the SLT Solution (maybe in the next iteration of V-Model, the SLT Solution should include the cloud management decisions pointed out by NIST).

B.2 A-Type Sequence Diagrams

In this section is presented a set of A-Type Sequence Diagrams, modeled with the intent to represent the activity flows in the SLT environment. The sequence diagrams are modeled in a stereotyped way, by using actors and use cases (Machado et al. 2007), which allow a behavior clearance of the use case model.

B.2.1 A -Type Sequence Diagram #1: Operation Orders Planning

0a. Programmer configures the schedule of operation orders according to the node's needs and External Entity's needs.
0b. The system sends the schedule of operation orders to the control activities for validation. These control activities are concern on manage new requests of schedule.
0c. The schedule of operation orders is validated.
0d. Programmer receives the validation.
1. External Entity requests a schedule for a specific period.
2. The information of request is sent to the control activities for validation and to process the request.
3. The system filters the information and reply.
4. External Entity receives the schedule information.
5. External Entity requests a given day and a given hour to schedule operation orders;
6. The information of request is sent to the control activities.
7. The information of request can be sent to the Programmer for validation.
8. Programmer reply to request.
9. The system validates the request.
10. External Entity receives the confirmation of request.
11. External Entity can send an alert to the system for User to start an operation order execution.
12. The system contacts with User.
13. User reply to the request.
14. The operation order execution is confirmed to the External Entity.

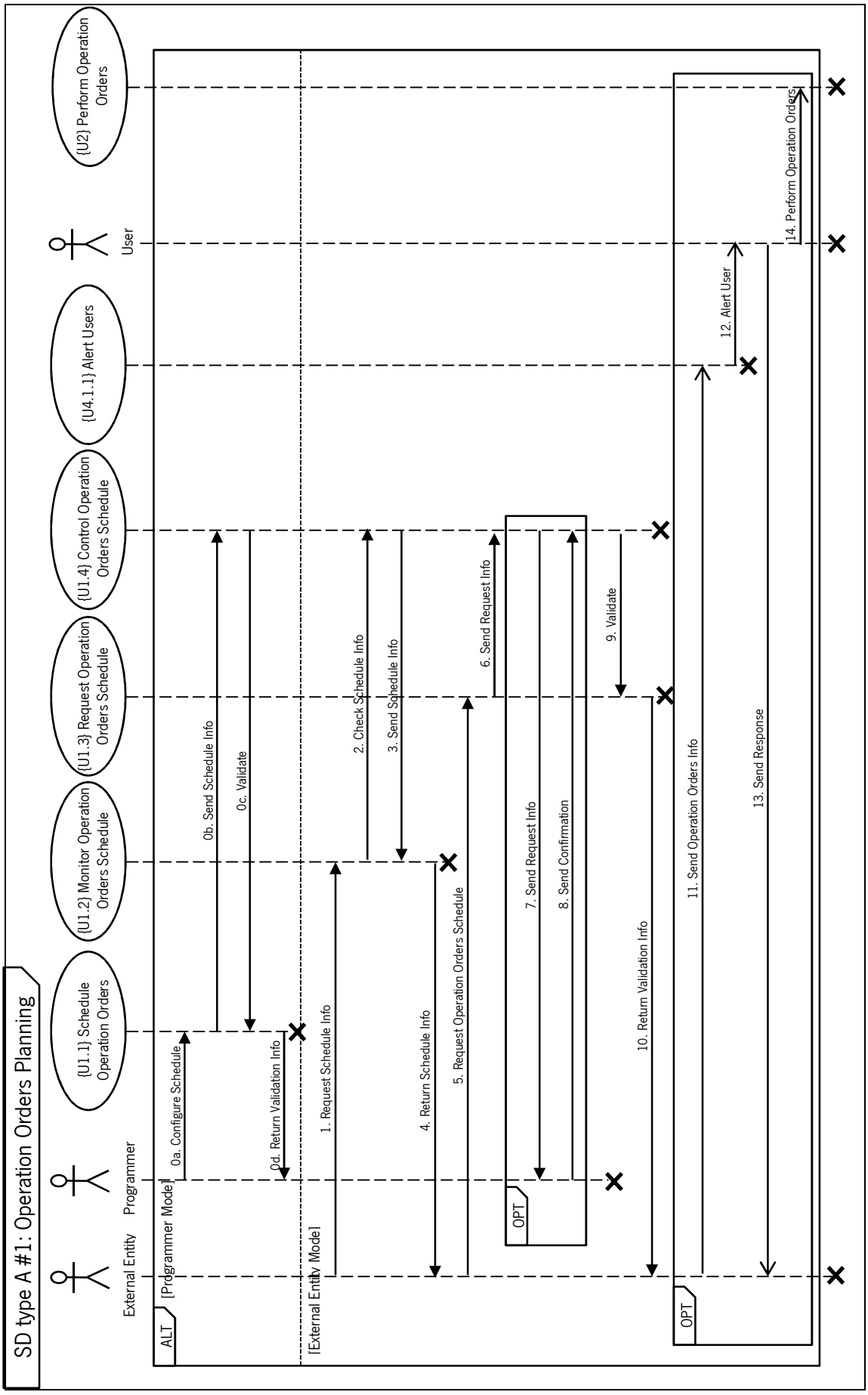


Figura A 4 – A -Type Sequence Diagram #1: Operation Orders Planning.

B.2.2 A -Type Sequence Diagram #2: Parking

1. User presents its identification for check-in.
2. The User identification information can be sent to the ERP for validation.
3. The information is validated.
4. The check-in information is reported to the system.
5. User receives the result of check-in validation.
6. The check-in information is sent to the availability monitoring activities. Normally the availability of a node is determined by the operation order priority, the number of operation orders that a node can process in the moment and the scheduled operation orders.
7. The node is monitored through control devices (e.g. posts, barriers and weighing equipment).
8. The control devices return the information.
9. The received information is checked for availability.
10. If an availability information is detected:
11. An alert is sent to the system.
12. The User receives an availability alert.

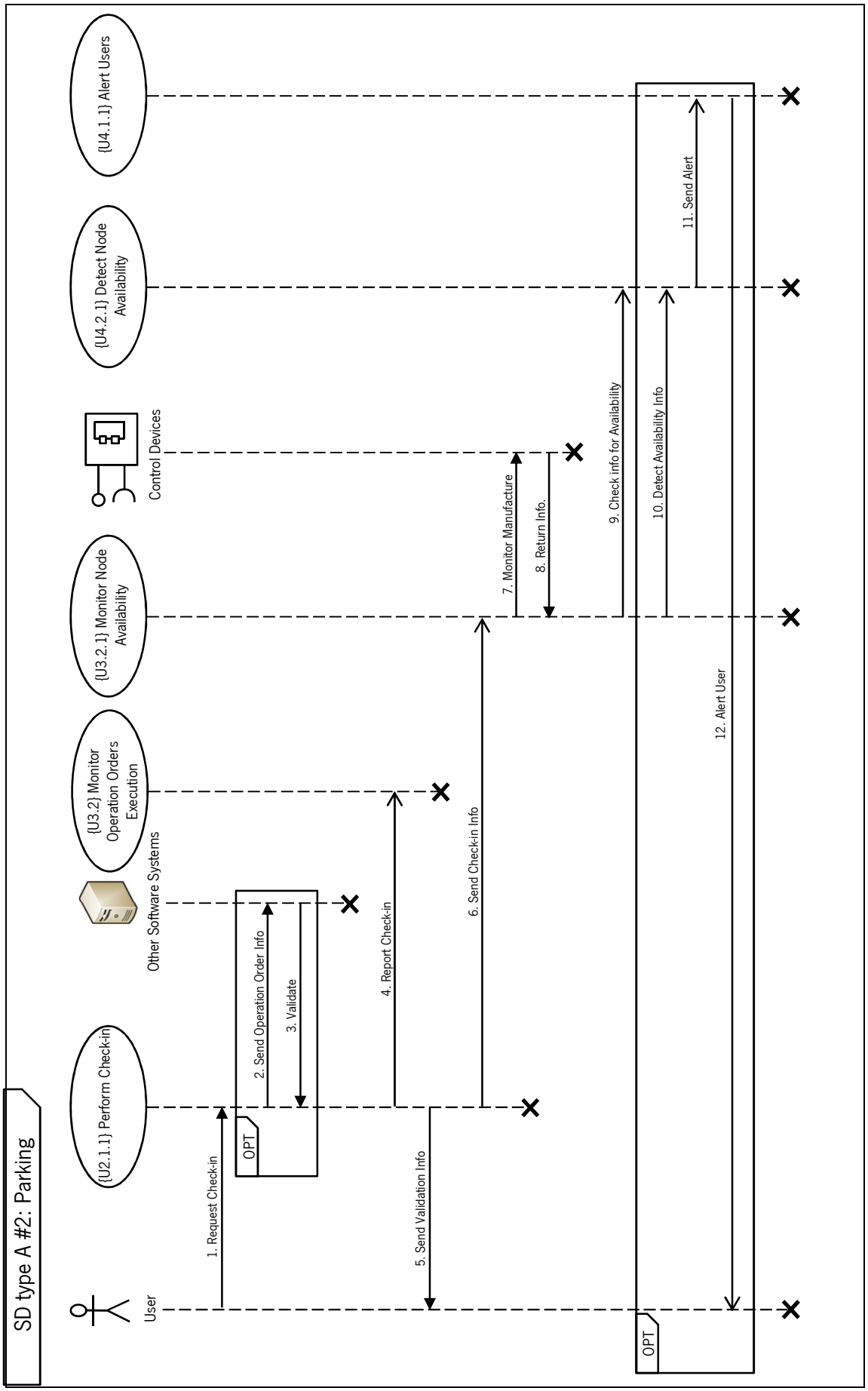


Figura A 5 – A -Type Sequence Diagram #2: Parking.

B.2.3 A -Type Sequence Diagram #3: Entrance/Exit

1. User requests an entrance/exit to realize an operation order.
2. The entrance/exit information is sent to the control activities. This information includes the user identification and also information related with the desired/realized operation orders.
3. The received information is checked for entrance/exit abnormalities. This information compared with the control settings determines the existence of an anomaly.
4. If an anomaly information is detected:
5. An alert is sent to the system.
6. In the case that the check-in is successfully validated hitherto, the entrance/exit information can be sent to the ERP for validation.
7. The information is validated.
8. The entrance/exit information is reported to the system.
9. The response of the validation is formulated.
10. User receives the result of entrance/exit validation.

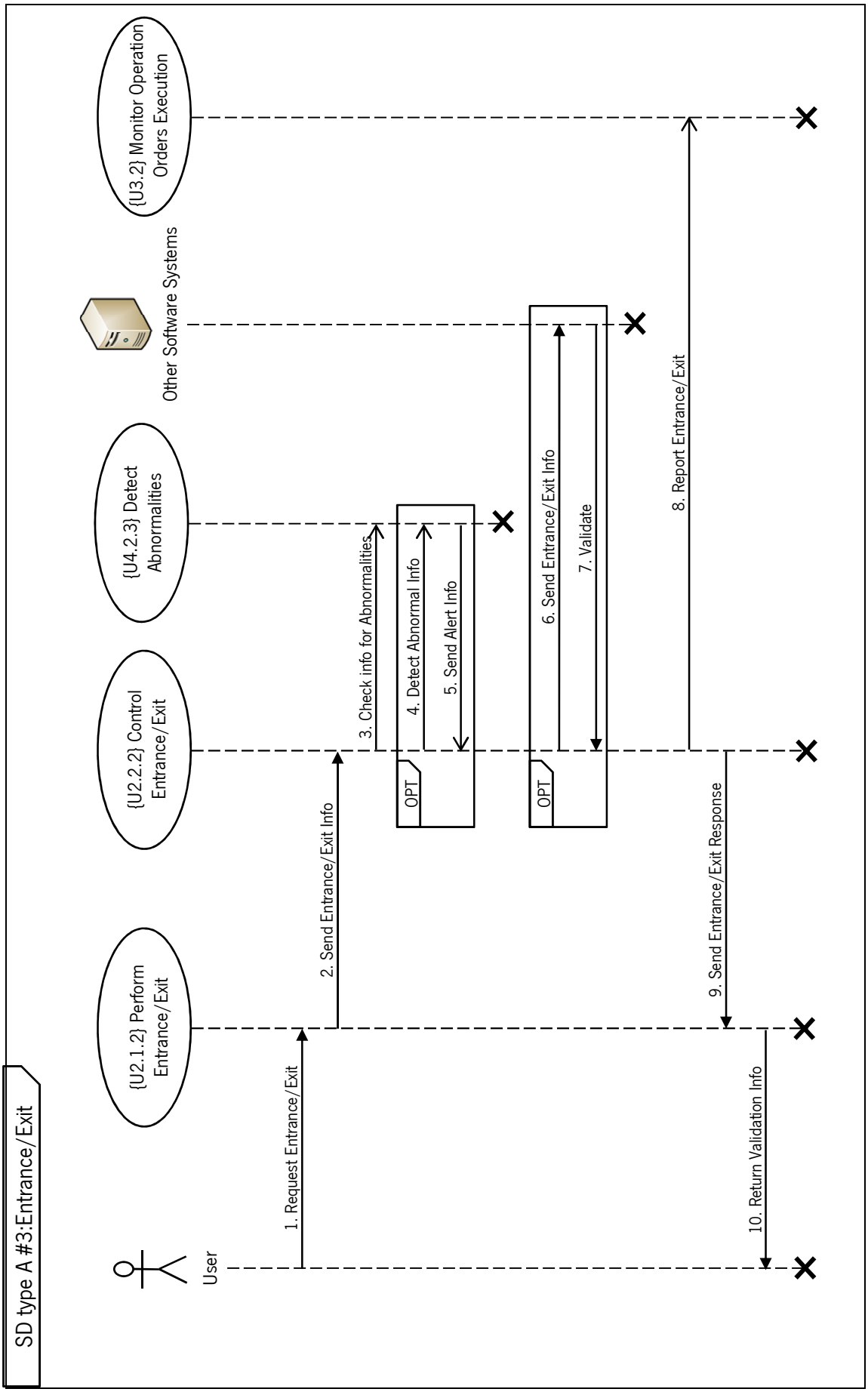


Figura A 6 – A -Type Sequence Diagram #3: Entrance/Exit.

B.2.4 A -Type Sequence Diagram #4: Weighing

1. User requests to weighing before/after an operation;
2. The weighing information is sent to the control activities. This information includes the way how was realized the weighing and its results.
3 and 6. The received information is checked for detect weighing insecurities and abnormalities. One part of this information, about the way how was realized the weighing (e.g. incorrect truck position and lack of scales calibration), confronted with the control settings, related with the devices behavior, determines the existence of an insecurity. The remaining part of information, about the weighing results (e.g. gross, net and tare weight), compared with the control settings, related with the conditions of weighing, determines the existence of an anomaly.
4. If an insecurity information is detected:
5. An alert is sent to the system.
7. If an anomaly information is detected:
8. An alert is sent to the system.
9. In the case of none detected insecurities and anomalies, the weighing information can be sent to the ERP for validation.
10. The information is validated.
11. The weighing information is reported to the system.
12. The response of the validation is formulated.
13. User receives the result of weighing validation and the information about the place where the operation order can be realized.

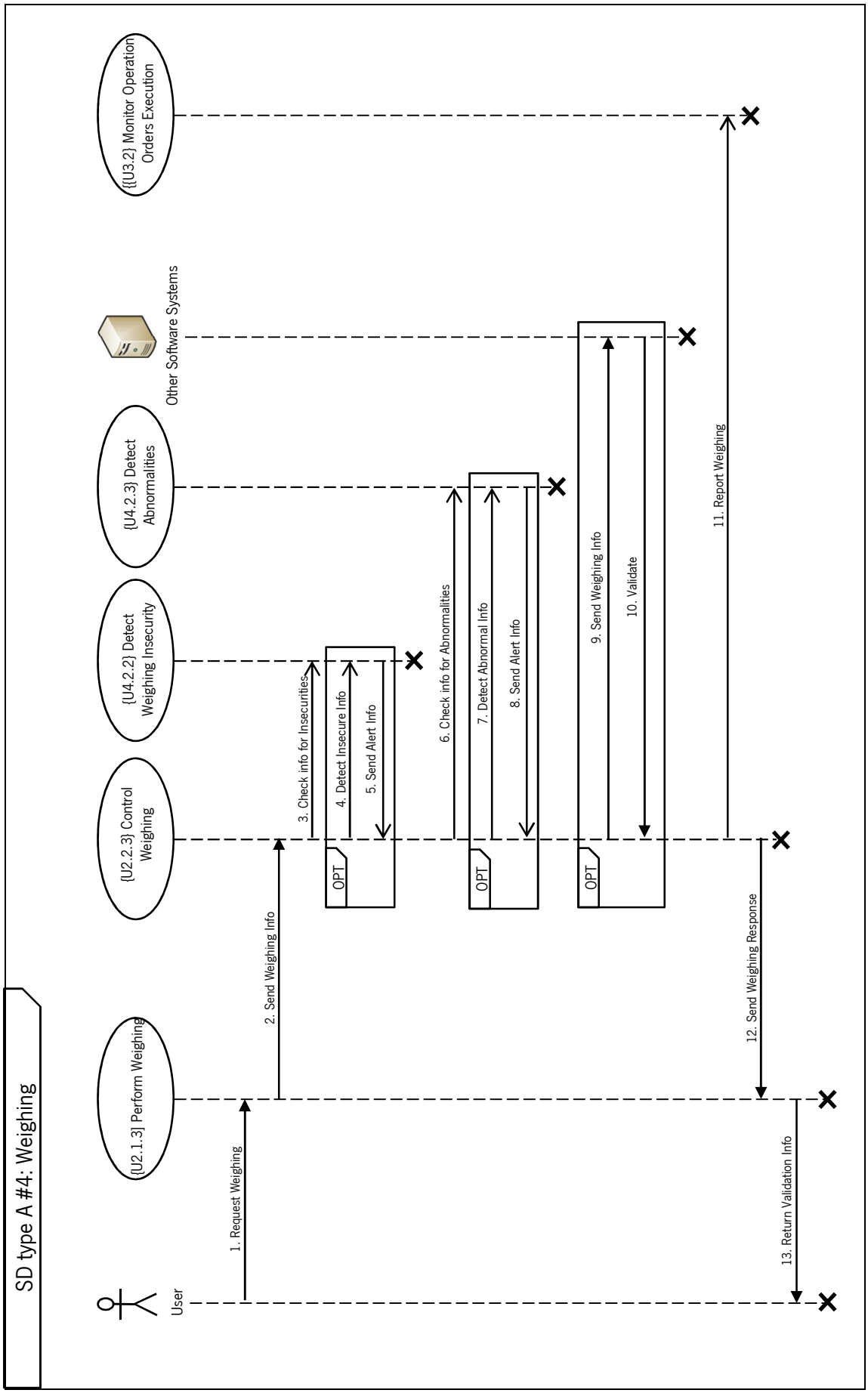


Figura A 7 – A -Type Sequence Diagram #4: Weighing.

B.2.5 A -Type Sequence Diagram #5: Loading

1. User presents their identification to perform a loading task.
2. The User identification information can be sent to the ERP for validation the loading request.
3. The information is validated.
4. User receives the result of the loading request validation.
5. The loading information is sent to the control activities. This information includes the quantity of the products requested. The control activities ensures that the loading is done safely, complying monitoring settings.
6. User requests the beginning of the loading.
7a. One alert to start the loading can be sent to the system, so that can be possible control the loading execution.
7b. The weighing is monitored through monitoring devices (e.g. weighing equipment, sensors and silos).
7c. The monitoring devices (e.g. weighing equipment, sensors and silos) return the information about the current loading values.
7d. The received information is checked to detect abnormalities. One abnormality is determined based on the monitoring settings to control, for example, the pipe position, the products that can't be mixed and the overloading.
7e. If an abnormality is detected:
7f. An alert is sent to the system.
7g. The result of the loading execution is formulated.
8. The loading information is reported to the system.
9. The loading task is finished and the User receives the result of the loading execution.

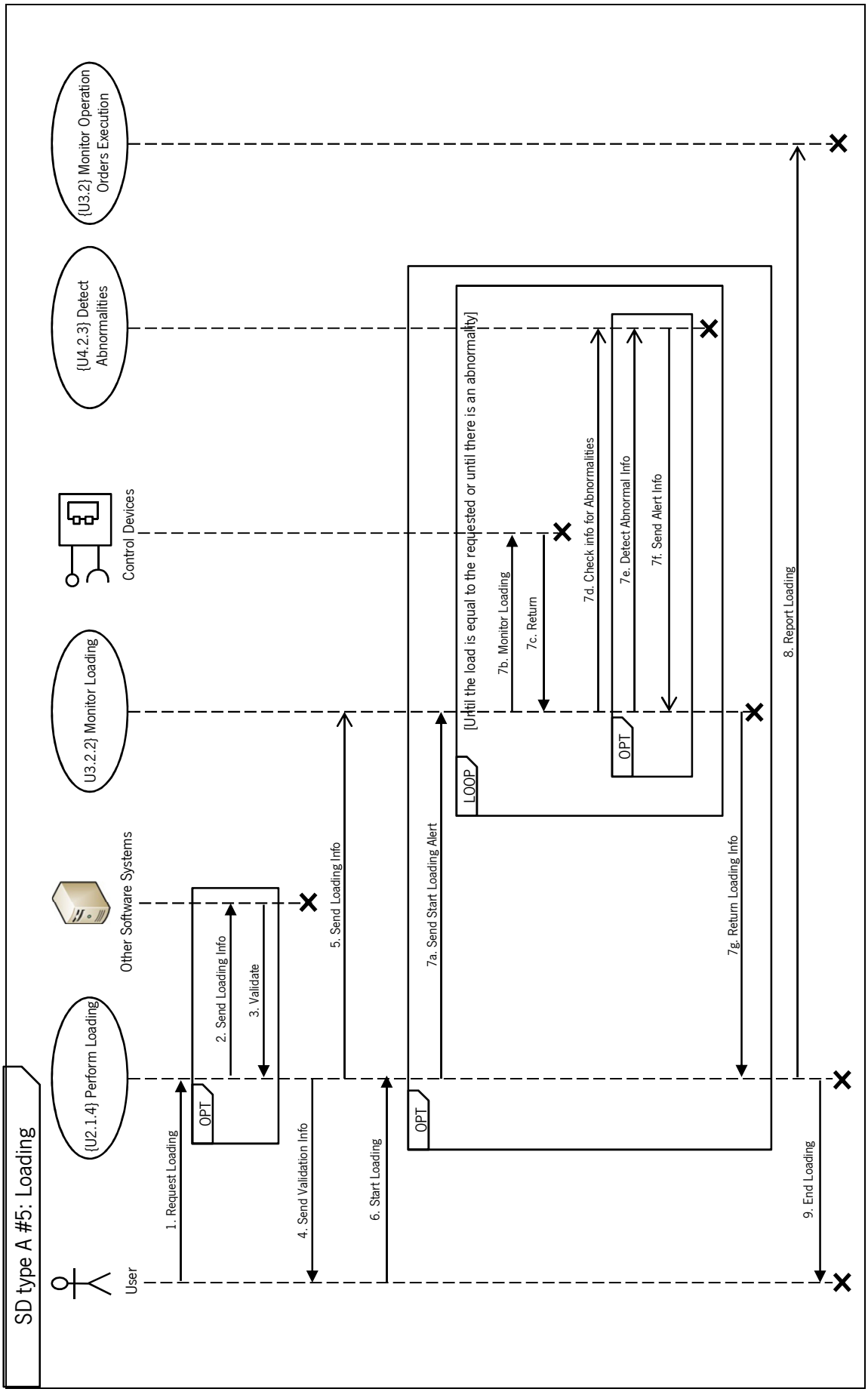


Figura A 8 – A -Type Sequence Diagram #5: Loading.

B.2.6 A -Type Sequence Diagram #6: Unloading

1. User presents their identification to perform an unloading task.
2. The User identification information can be sent to the ERP for validation the unloading request.
3. The information is validated.
4. User receives the result of the unloading request validation.
5. The unloading information is sent to the control activities. This information includes the quantity of the products requested. The control activities ensures that the unloading is done safely, complying monitoring settings.
6. User requests the beginning of the unloading.
7a. One alert to start the unloading can be sent to the system, so that can be possible control the unloading execution.
7b. The weighing is monitored through monitoring devices (e.g. weighing equipment, sensors and silos).
7c. The monitoring devices (e.g. weighing equipment, sensors and silos) return the information about the current unloading values.
7d. The received information is checked to detect abnormalities. One abnormality is determined based on the monitoring settings to control, for example, the pipe position, the products that can't be mixed and the overloading.
7e. If an abnormality is detected:
7f. An alert is sent to the system.
7g. The result of the unloading execution is formulated.
8. The unloading information is reported to the system.
9. The unloading task is finished and the User receives the result of the unloading execution.

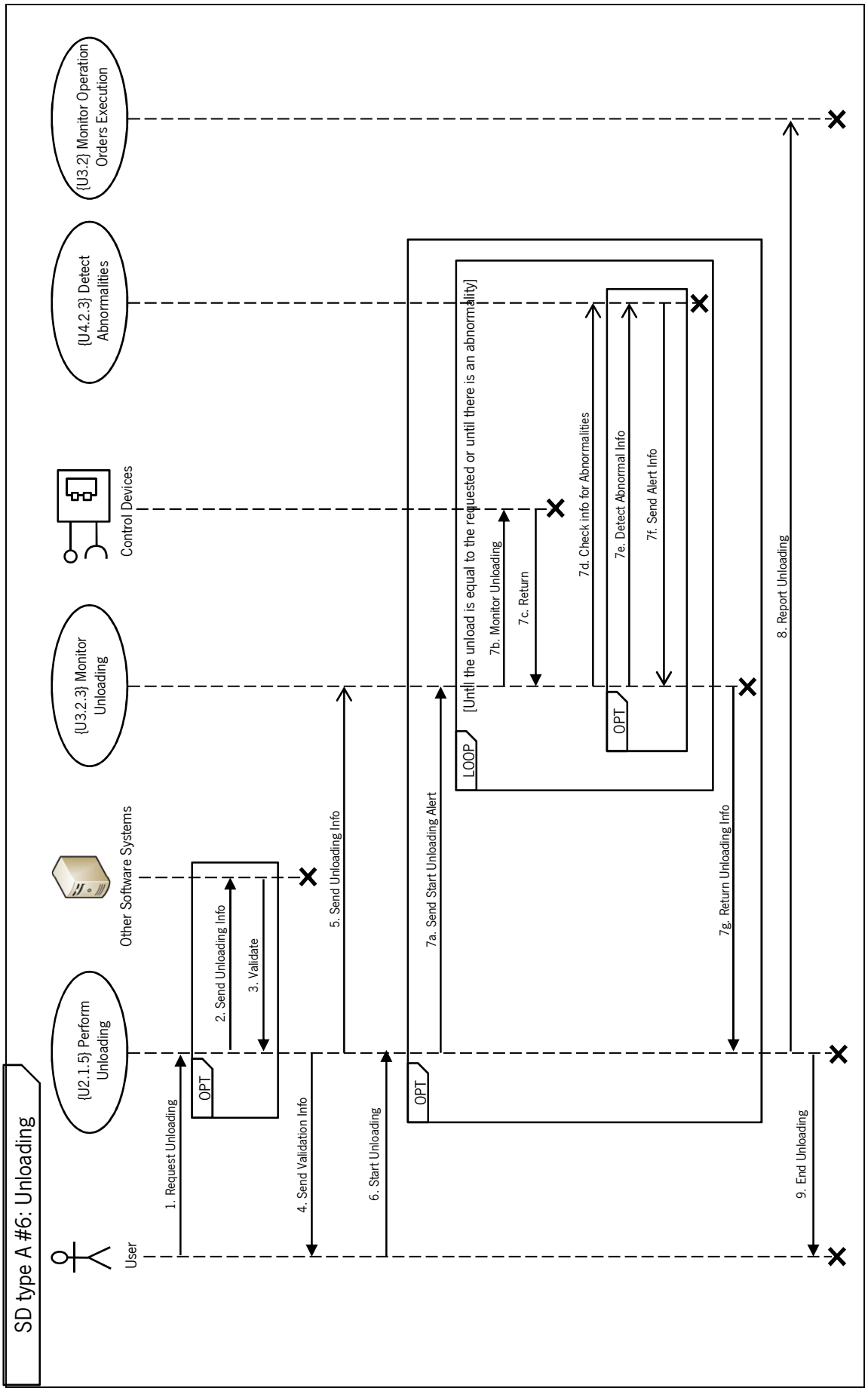


Figura A 9 – A -Type Sequence Diagram #6: Unloading.

B.2.7 A -Type Sequence Diagram #7: Repair/Maintenance Assistance

0a. User requests assistance.
0b. User receives a confirmation that the request is valid.
0c. Controller receives assistance request.
0d. Controller receives an alert.
0e. The request is approved to the User.
1a. Controller receives abnormality information.
1b. Controller receives an alert.
2. Controller gets in touch with the Technician.
3. Technician reply to request.
4. The assistance is confirmed to the Controller.
5. In the case where the estimated time for the assistance execution is high, the Controller can send an alert to all External Entities that may be affected.
6. External Entities receives an alert.

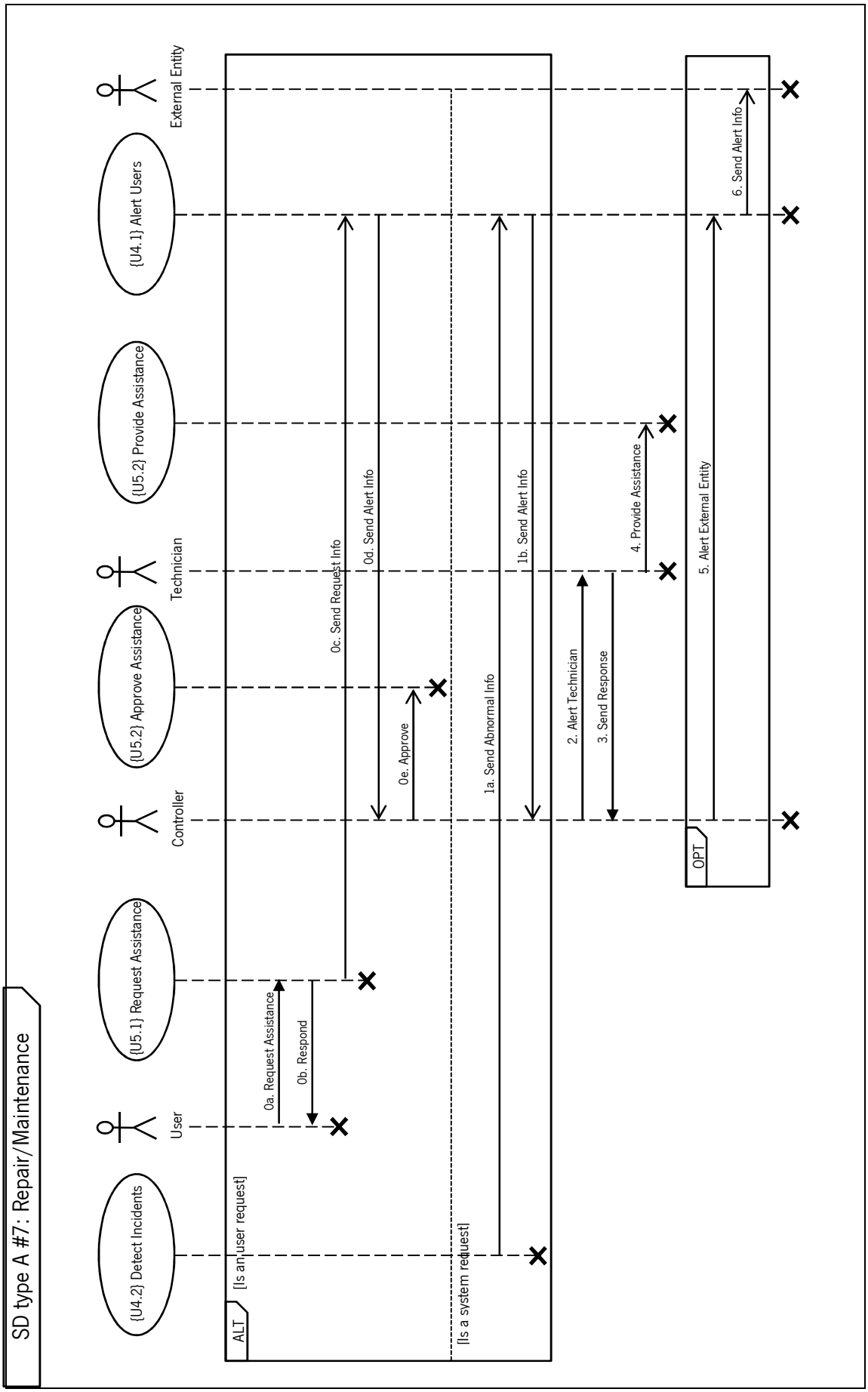


Figura A 10 – A -Type Sequence Diagram #7: Repair/Maintenance Assistance.

B.3 Process - level Use Cases

With respect to the process - level use cases, a set of canonic activities and a set of system actors were defined and detailed. These process-level use cases allow a better understanding of the SLT Solution related activities.

B.3.1 System Actors

An actor represents a user that interacts with the SLT system. The following table presents the system actors descriptions.

Tabela A 1 – System Actors.

Name	Description
User	User represents the actor on which the system under development focuses on. User is presented in this context as an external entity Driver or a node Operator that will directly or indirectly interact with SLT products and services.
Admin	Admin is an actor with privileges to configure the behavior of the Control Devices and to manage SLT infrastructure and services.
Analyst	This actor represents a person able to infer the node service quality and to report improvement suggestions.
Programmer	A Programmer represents a person able to manage, set and program the operation orders schedule of a node.
Controller	A Controller is a person able to examine assistance requests of a node and to get in touch with responsible and suitable Technicians for each emergency situation. In addition, the Controller can also alert all External Entities for delays in a node's operation order processing.
Technician	A Technician is a person able to assist Users and to perform equipment maintenance, both remotely as on onsite.
External Entity	This actor represents external entities of a node, for example, Customers, Suppliers and Transporters companies. The External Entities can plan and manage their node service needs with more accurately.
Other Software Systems	This actor represents already existing software installed and exploited within the context of each node in the supply chain. Through this software (e.g. ERP, CRM or SCM) one person can interact with a SLT and explore its functionalities.
Control Devices	Represents a set of monitoring and control devices (e.g. weighing equipment, sensors, posts and traffic lights) that ensure the safe execution of operation order tasks.

B.3.2 Use Cases Diagrams

In a high-level view of the process were represented five main canonic activities (i.e. use-cases), each directly related to an activity performed by SLT Consumer. In the following figure it is possible to see the SLT main actors and their interactions with the system's use cases, from a process point of view. Each use case has a textual description in a way to achieve representation completeness with the functional modeling, enabling process requirements to be captured and described.

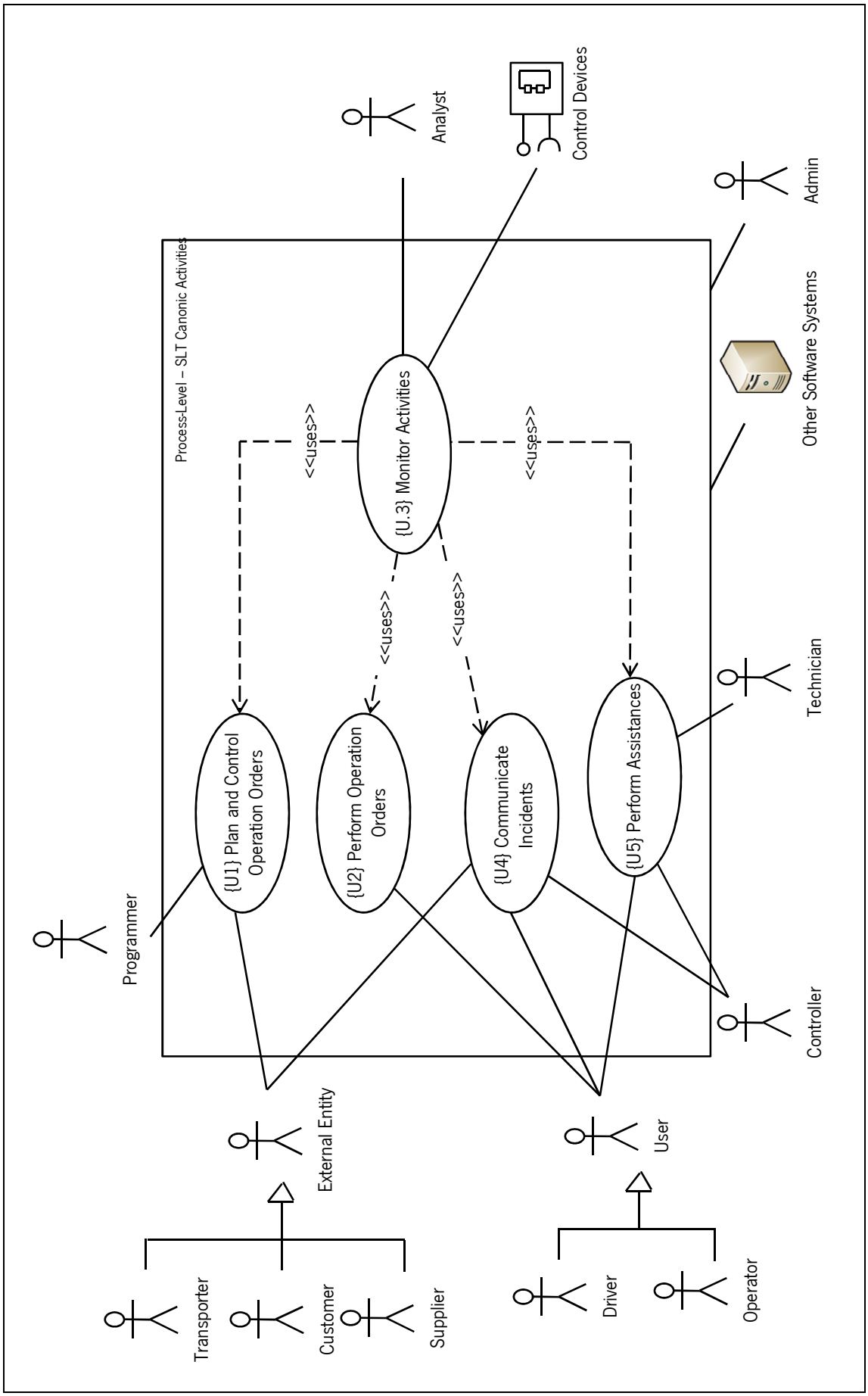


Figura A 11 – SLT Canonic Activities.

In relation to the main canonic activities, the high-level descriptions are:

{U1} Plan and Control Operation Orders: Plan and control a set of operation orders for a node. From a node point of view, an operation order is the reception of raw materials and/or the expedition of the final products. By accessing SLT functionalities, the internal (Programmer) and external (External Entity) entities of a node can collaborate in order to meet the needs of both. This could help to increase efficiency relating to inventory management, freight transportation and orders processing, across the entire supply chain and in particular at the node.

{U2} Perform Operation Orders: Execute a set of required tasks to ensure the automation, control and safety in an operation order processing realized by a User. The number of required tasks depends on the application context of SLT. Therefore, the SLT should allow an Admin to configure the required tasks, as well as their sequence. By default and based on the SLV Cement system, the SLT system will incorporate five tasks ({U2.1.1} Perform Check-in; {U2.1.2} Perform Entrance/Exit, {U2.1.3} Perform Weighing, {U2.1.4} Perform Loading and {U2.1.5} Perform Unloading). With the possibility of augmenting the system with the inclusion of new tasks there is one specific activity for that ({U2.1.6} Perform Custom Tasks).

{U3} Monitor Activities: Allow the system to monitor all SLT activities so the Analyst can identify areas for improvement and monitor tasks of an operation order in order to maintain the safety of an operation order execution. By default, the SLT has three operation order monitor activities ({U3.2.1} Monitor Node Availability, {U3.2.2} Monitor Loading, {U3.2.3} Monitor Unloading). Nevertheless, the system should allow an Admin to configure new operation order monitor activities ({U3.2.4} Monitor Custom Activities).

{U4} Communicate Incidents: Detect incidents related with the use of the SLT, e.g. anomalous parameters or relevant events, and allow the exchange of alerts between users. An anomalous parameter has a value outside a set of predicted values relating to a particular activity (e.g. anomaly values on the quantities charged or an incorrect positioning of the load). A relevant event indicates that the current system state needs to be reported (e.g. availability of node and system failure). When the abnormality or event is detected, the system alerts the specified users associated to that abnormality or event. This specification should be done by the SLT system Admin.

{U5} Perform Assistances: Provides the necessary support to maintain the correct and independent use of SLT. The User or the system (Communicate Incidents) can request assistance, the Controller can manage the assistance requests and the Technician can offer remote and onsite assistance services.

For explain in more detail each activity previously described, process-level use cases were refined by disaggregation. Hereafter the use case refinement models and textual descriptions are presented. The textual descriptions are presented according to the templates proposed by Rational Unified Process (RUP) (Kruchten 2004).

B.3.2.1 Plan and Control Operation Orders Use Case Refinement

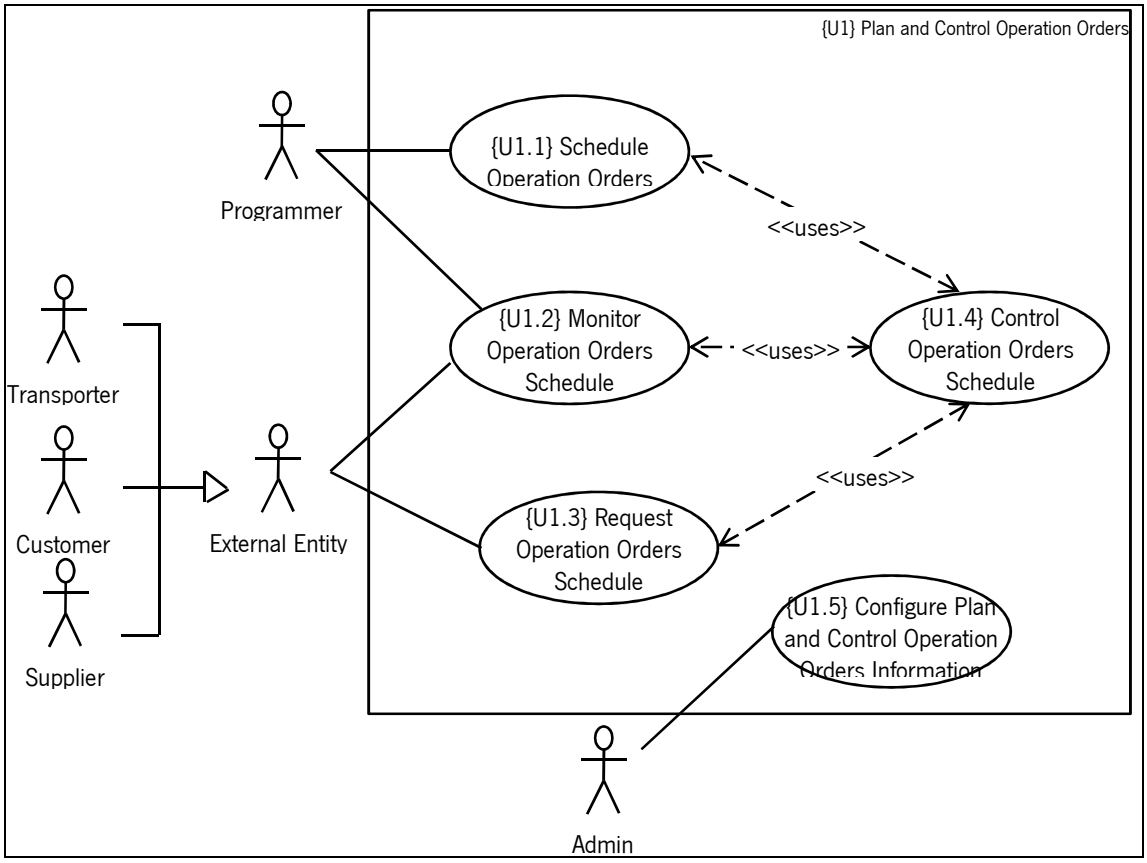


Figura A 12 – Plan and Control Operation Orders use case refinement.

Reference	{U1.1}
Name	Schedule Operation Orders
Short Description	Defines and approves operation orders schedule according to the node and external entities needs. An operation orders schedule contains a list of operation orders. This schedule should include: date and time of operation orders, the estimated duration, the operation orders site, detail of the operation orders, information about the user, customers, suppliers and/or carriers, etc. It is important to schedule the operation orders because internal and external entities of a node are more capable to prepare for future operations. Also, the node can improve its performance in executing orders, since the User can start the operation order in the moment that he arrives to the node, without having to waste time in registering the operation order at the node.
Actors	Programmer
Basic Flow	1. Programmer configures the schedule of operation orders according to the existing scheduled operation orders from programmers and external entities. 2. The system sends the schedule of operation orders for validation. 3. The system receives the result of validation. 4. Programmer receives the validation.
Alternative Flow	
Special Requirements	
Preconditions	
Postconditions	
Extension Points	The execution of this use case depends on validation that results of {U1.4} Control Operation Orders Schedule. This use case outputs an operation orders schedule to be used for the execution of {U1.2} Monitor Operation Orders Schedule and to be used as basis for the execution of {U.2} Perform Operation Orders.

Reference	{U1.2}
Name	Monitor Operation Orders Schedule
Short Description	Analyzes the node's current operation orders schedule. This analysis includes the requested operation orders by external entities and scheduled operation orders by programmer. The current operation orders schedule is constantly monitored by programmer and external entities, to define and approve the schedule and to request operation orders, respectively.
Actors	Programmer; External Entity
Basic Flow	1. Programmer/External Entity requests schedule information for a specific period. 2. The system sends the information request to filter the operation orders. 3. The system receives the filtered information. 4. Programmer/External Entity receives the appropriated schedule information.
Alternative Flow	
Special Requirements	
Preconditions	
Postconditions	
Extension Points	The execution of this use case depends on a filter that results of {U1.4} Control Operation Orders Schedule. This use case may be used by use case {U1.1} Schedule Operation Orders and {U1.3} Request Operation Orders Schedule. The execution of this use case can be triggered by Programmer and External Entity.

Reference	{U1.3}
Name	Request Operation Orders Schedule
Short Description	Proposes an operation orders schedule to be included in the node's current operation orders schedule. This proposal expresses the external entities needs and includes a list of operation orders, the desired date and the site to perform each operation order, etc.
Actors	External Entity
Basic Flow	1. External Entity requests a given day, a given hour and a given site to schedule operation orders. 2. The system sends the information request for validation. 3. The system receives the result of validation. 4. Programmer receives the validation.
Alternative Flow	
Special Requirements	
Preconditions	
Postconditions	
Extension Points	The execution of this use case depends on validation that results of {U1.4} Control Operation Orders Schedule. This use case outputs an operation orders schedule proposal to be used for the execution of {U1.2} Monitor Operation Orders Schedule.

Reference	{U1.4}
Name	Control Operation Orders Schedule
Short Description	Validates the requested operation orders by external entities and the scheduled operation orders by programmer. Additionally, this activity filters schedule information accordingly to the configuration settings defined on {U1.5} Configure Plan and Control Operation Orders Information.
Actors	
Basic Flow	For {U1.1} Schedule Operation Orders: 1. The system receives the scheduled operation orders for validation. 2. The system can send the scheduled operation orders to other programmer for validation. 3. The system receives programmer reply. 4. The system validates the scheduled operation orders. 5. The system sends the result of validation. For {U1.2} Monitor Operation Orders Schedule: 1. The system receives the operation orders information request to filter operation orders

	accordingly to the configuration settings. 2. The system filters the schedule of operation orders. 3. The system sends filtered information about the availability of node's operation orders and the current state of Users' operation orders (Programmer and External Entity). For {U1.3} Request Operation Orders Schedule: 1. The system receives the requested operation orders for validation. 2. The system can send the requested operation orders to the programmer for validation. 3. The system receives programmer reply. 4. The system validates the requested operation orders. 5. The system sends the result of validation.
Alternative Flow	
Special Requirements	
Preconditions	The Programmer or External Entity should have permission to make requests and the data of the requests should be clean, correct and useful.
Postconditions	
Extension Points	The execution of this use case can be triggered within the execution of use cases {U1.1} Schedule Operation Orders, {U1.2} Monitor Operation Orders Schedule and {U1.3} Request Operation Orders Schedule.

Reference	{U1.5}
Name	Configure Plan and Control Operation Orders Information
Short Description	Inserts, edits and instantiates information concerning activities from {U1} Plan and Control Operation Orders (except for this configuration activity). This information is used to recognize patterns in Users' needs (Programmer and External Entity) and to adjust the activities of {U1} Plan and Control Operation Orders (except for this configuration activity) to these patterns. These configurations may change the aspect of the {U1} Plan and Control Operation Orders interfaces and may change the business logic associated to the {U1} Plan and Control Operation Orders activities. For example, an Admin can configure the aspect of the interfaces and the availability (turn on and off) of the functionalities to fit the Users' needs. He can also define business rules to align {U1} Plan and Control Operation Orders activities to the node's business logic.
Actors	Admin
Basic Flow	1. Insert/Edit/Instantiate a set of given parameters, concerning a specific activity and user.
Alternative Flow	
Special Requirements	This use case must precede the first ever execution of use case {U1.1} Schedule Operation Orders, use case {U1.2} Monitor Operation Orders Schedule, use case {U1.3} Request Operation Orders Schedule and use case {U1.4} Control Operation Orders Schedule.
Preconditions	
Postconditions	
Extension Points	Use case {U1.1} Schedule Operation Orders, use case {U1.2} Monitor Operation Orders Schedule, use case {U1.3} Request Operation Orders Schedule and use case {U1.4} Control Operation Orders Schedule are executed based on the configured information provided as input by this use case.

B.3.2.2 Perform Operation Orders Use Case Refinement

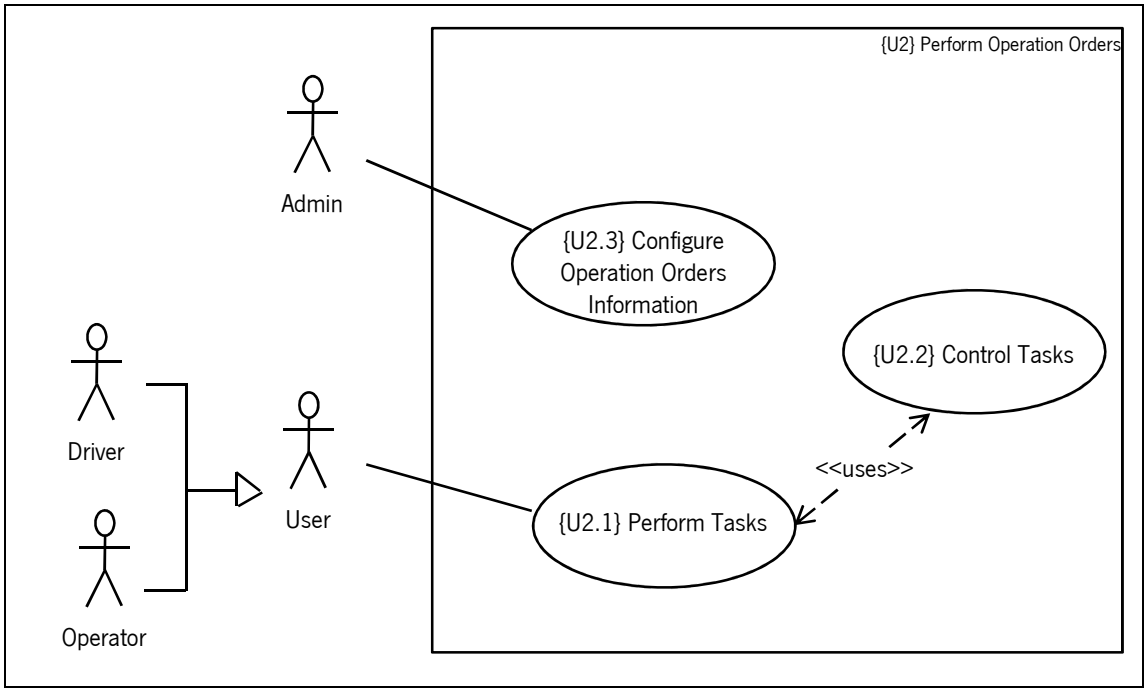


Figura A 13 – Perform Operation Orders use case refinement.

Reference	{U2.1}
Name	Perform Tasks
Short Description	Execute a set of required tasks to accomplish an operation order request. The number of required tasks and the sequence of them are defined accordingly to the system settings. It is noteworthy that, by default and based on the SLV Cement system, the SLT system will incorporate five tasks ({U2.1.1} Perform Check-in; {U2.1.2} Perform Entrance/Exit, {U2.1.3} Perform Weighing, {U2.1.4} Perform Loading and {U2.1.5} Perform Unloading) and can also include new tasks ({U2.1.6} Perform Custom Tasks).
Actors	User
Basic Flow	1. User presents its identification to perform each task. 2. The system sends the user identification to validate each task to perform. 3. The system receives the result of validation. 4. User receives the validation.
Alternative Flow	
Special Requirements	User may have user identification device (e.g. tag RFID) to perform some tasks.
Preconditions	
Postconditions	
Extension Points	The execution of this use case depends on validation that results of {U2.2} Control Tasks.

Reference	{U2.2}
Name	Control Tasks
Short Description	Execute a set of control tasks to ensure the automation, control and safety in an operation order processing. Furthermore, this activity reports the state of each operation order whenever the state is changed to trigger the monitoring activities and the detect incidents activities.
Actors	
Basic Flow	1. The system receives the user identification to validate each task to be performed. 2. The system can send the request to other systems for validation and reporting. 3. The system can receive other systems reply. 4. The system validates the request by sending the request information to the use case {U4.2} Detect Incidents and/or to the use case {U.3} Monitor Activities.

	5. The system updates the state of each operation order. 6. The system can send the state of each operation order to the monitoring activities ({U.3} Monitor Activities) to trigger some monitoring activity. 7. The system sends the result of validation.
Alternative Flow	
Special Requirements	
Preconditions	The User (Driver or Operator) should have permission to perform tasks and the data of the requests should be clean, correct and useful.
Postconditions	
Extension Points	The execution of this use case depends on operation orders schedule that results of {U1.1} Schedule Operation Orders and this use case can be triggered within the execution of use cases {U2.1} Perform Tasks.

Reference	{U2.3}
Name	Configure Operation Orders Information
Short Description	Inserts, edits and instantiates information concerning activities from {U2} Perform Operation Orders (except for this configuration activity). This information is used to recognize patterns in Users' needs (Driver or Operator) and to adjust the activities of {U2} Perform Operation Orders (except for this configuration activity) to these patterns. These configurations may change the aspect of the {U2} Perform Operation Orders interfaces and may change the business logic associated to the {U2} Perform Operation Orders activities. For example, an Admin can configure the aspect of the interfaces and the availability (turn on and off) of the functionalities to fit the Users' needs. He can also define business rules to align {U2} Perform Operation Orders activities to the node's business logic and define what tasks are required as well as what is the sequence of them.
Actors	Admin
Basic Flow	1. Insert/Edit/Instantiate a set of given parameters, concerning a specific activity.
Alternative Flow	
Special Requirements	This use case must precede the first ever execution of use case {U2.1} Perform Tasks and use case {U2.2} Control Tasks.
Preconditions	
Postconditions	
Extension Points	Use case {U2.1} Perform Tasks and use case {U2.2} Control Tasks are executed based on the configured information provided as input by this use case.

B.3.2.2.1 Perform Tasks Use Case Refinement

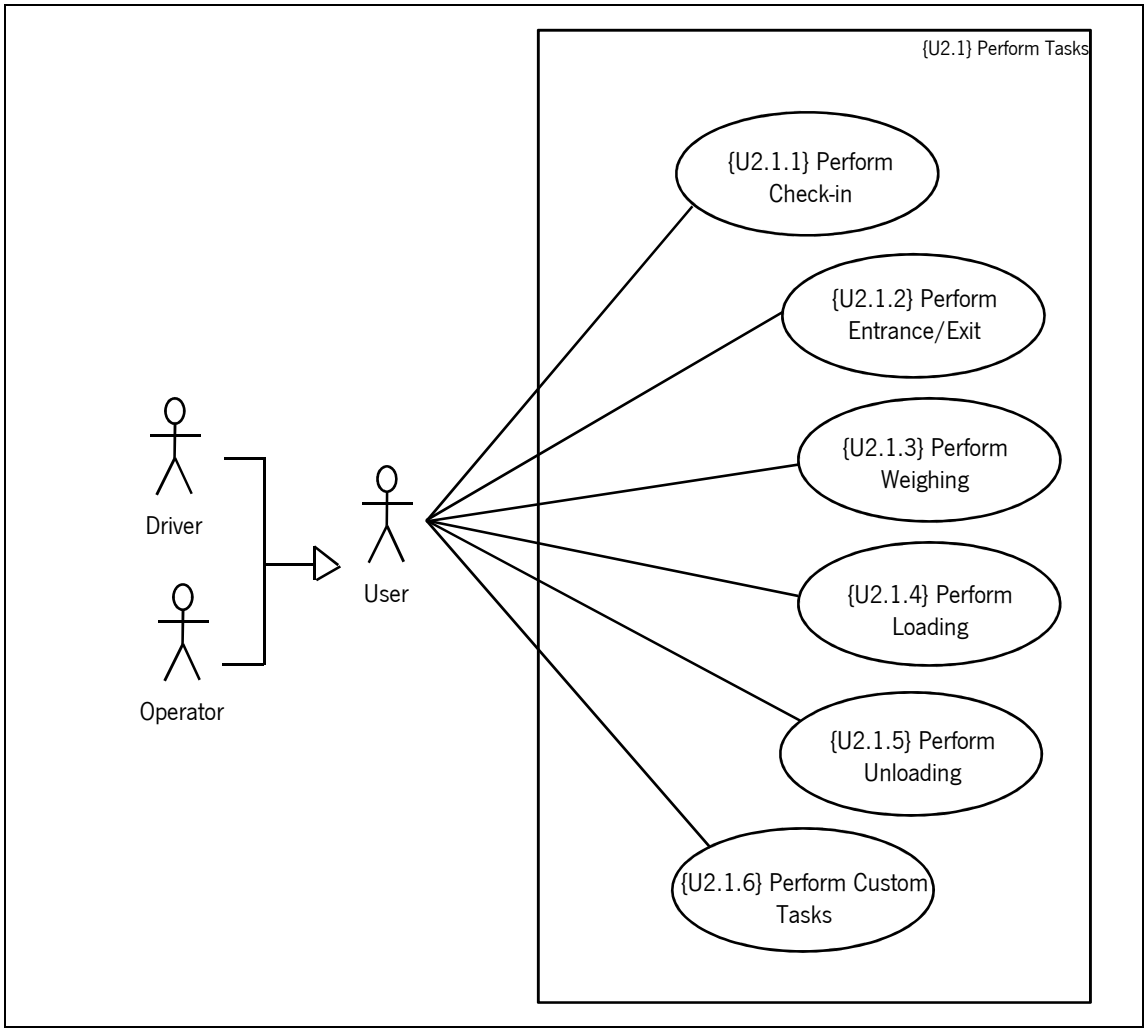


Figura A 14 – Perform Tasks use case refinement.

Reference	{U2.1.1}
Name	Perform Check-in
Short Description	Capture the user's identification and all the requirements needed, accordingly to the configuration settings, for fulfill a check-in task. Usually, this task is the first step to be performed by the user in an operation order processing.
Actors	User
Basic Flow	1. User presents its identification to perform a check-in task. 2. The system sends the user identification to validate a check-in task. 3. The system receives the result of validation. 4. User receives the validation.
Alternative Flow	
Special Requirements	User may have user identification device (e.g. tag RFID) to perform a check-in.
Preconditions	
Postconditions	
Extension Points	The execution of this use case depends on validation that results of {U2.2.1} Control Check-in.

Reference	{U2.1.2}
Name	Perform Entrance/Exit
Short Description	Capture the user's identification and all the requirements needed, accordingly to the configuration settings, for fulfill an entrance/exit task. Typically, this task effectively

	indicates the beginning/ending of an operation order process.
Actors	User
Basic Flow	1. User presents its identification to perform an entrance/exit task. 2. The system sends the user identification to validate an entrance/exit task. 3. The system receives the result of validation. 4. User receives the validation.
Alternative Flow	
Special Requirements	User may have user identification device (e.g. tag RFID) to perform an entrance/exit task.
Preconditions	
Postconditions	
Extension Points	The execution of this use case depends on validation that results of {U2.2.2} Control Entrance/Exit.

Reference	{U2.1.3}
Name	Perform Weighing
Short Description	Capture the user's identification and all the requirements needed, accordingly to the configuration settings, for fulfill a weighing task. Normally, this task is executed two times, after an entrance or before an exit, to control what loads goes in and out.
Actors	User
Basic Flow	1. User presents its identification to perform a weighing task. 2. The system sends the user identification to validate a weighing task. 3. The system receives the result of validation. 4. User receives the validation.
Alternative Flow	
Special Requirements	User may have user identification device (e.g. tag RFID) to perform a weighing task.
Preconditions	
Postconditions	
Extension Points	The execution of this use case depends on validation that results of {U2.2.3} Control Weighing.

Reference	{U2.1.4}
Name	Perform Loading
Short Description	Capture the user's identification and all the requirements needed, accordingly to the configuration settings, for fulfill a loading task. Normally, this task is executed to control what is loaded. Not only in the end of the task but also during the loading task.
Actors	User
Basic Flow	1. User presents its identification to perform a loading task. 2. The system sends the user identification to validate a loading task. 3. The system receives the result of validation. 4. User receives the validation.
Alternative Flow	
Special Requirements	User may have user identification device (e.g. tag RFID) to perform a loading task.
Preconditions	
Postconditions	
Extension Points	The execution of this use case depends on validation that results of {U2.2.4} Control Loading.

Reference	{U2.1.5}
Name	Perform Unloading
Short Description	Capture the user's identification and all the requirements needed, accordingly to the configuration settings, for fulfill an unloading task. Normally, this task is executed to control what is unloaded. Not only in the end of the task but also during the unloading task.
Actors	User
Basic Flow	1. User presents its identification to perform an unloading task. 2. The system sends the user identification to validate an unloading task.

	3. The system receives the result of validation. 4. User receives the validation.
Alternative Flow	
Special Requirements	User may have user identification device (e.g. tag RFID) to perform an unloading task.
Preconditions	
Postconditions	
Extension Points	The execution of this use case depends on validation that results of {U2.2.5} Control Unloading

Reference	{U2.1.6}
Name	Perform Custom Tasks
Short Description	Capture the user's identification and all the requirements needed, accordingly to the configuration settings, for fulfill a custom task. This activity encompasses a range of other customized activities that can be performed in an operation order process.
Actors	User
Basic Flow	1. User presents its identification to perform a custom task. 2. The system sends the user identification to validate a custom task. 3. The system receives the result of validation. 4. User receives the validation.
Alternative Flow	
Special Requirements	User may have user identification device (e.g. tag RFID) to perform a custom task.
Preconditions	
Postconditions	
Extension Points	The execution of this use case depends on validation that results of {U2.2.6} Control Custom Tasks

B.3.2.2.2 Control Tasks Use Case Refinement

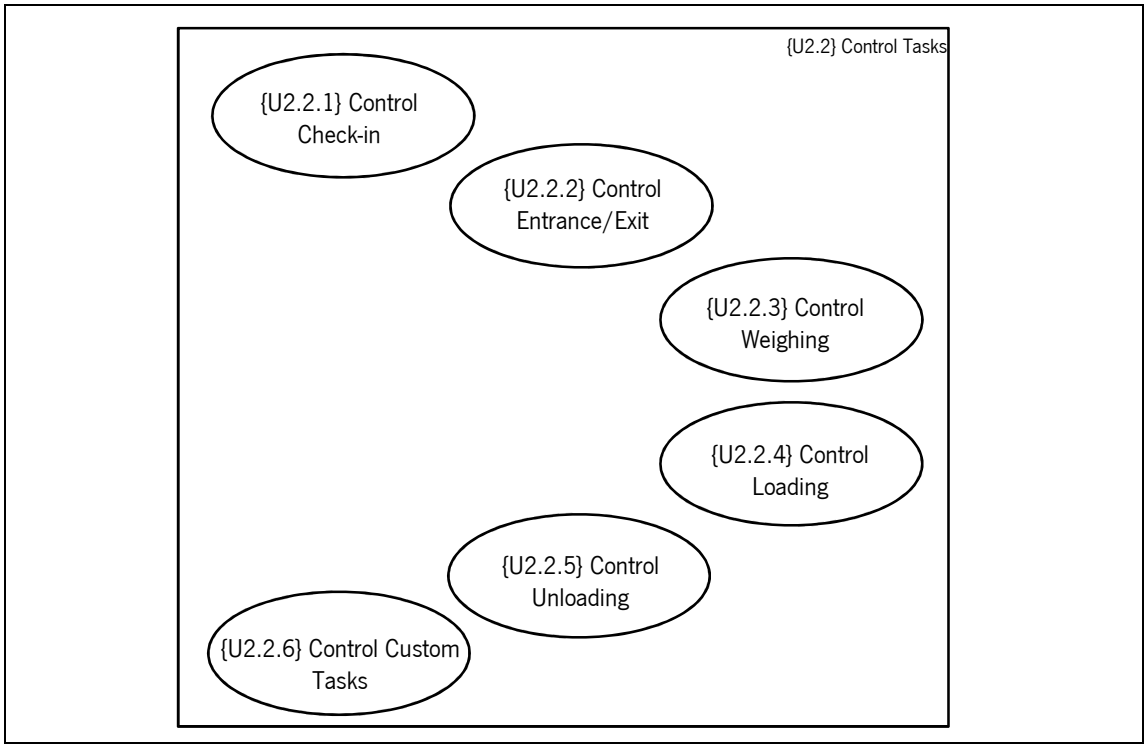


Figura A 15 – Control Tasks use case refinement.

Reference	{U2.2.1}
Name	Control Check-in
Short Description	Validates check-in requested by user and updates the state of an operation order.
Actors	
Basic Flow	1. The system receives the user identification to validate the check-in task to be performed. 2. The system can send the request to other systems for validation and reporting. 3. The system can receive other systems reply. 4. The system validates the request by sending the request information to the use case {U4.2.3} Detect Abnormalities (to detect abnormalities in the request). 5. The system updates the state to check-in in an operation order. 6. The system can send the state of each operation order to the monitoring activities ({U3.2.1} Monitor Node Availability) to detect node availability. 7. The system sends the result of validation.
Alternative Flow	
Special Requirements	
Preconditions	The User (Driver or Operator) should have permission to perform check-in and the data of the request should be clean, correct and useful.
Postconditions	
Extension Points	The execution of this use case depends on the operation orders schedule that results of {U1.1} Schedule Operation Orders and this use case can trigger {U3.2.1} Monitor Node Availability. Lastly, this use case can be triggered within the execution of use case {U2.1.1} Perform Check-in.

Reference	{U2.2.2}
Name	Control Entrance/Exit
Short Description	Validates entrance/exit requested by user and updates the state of an operation order.
Actors	
Basic Flow	1. The system receives the user identification to validate the entrance/exit task to be performed.

	2. The system can send the request to other systems for validation and reporting. 3. The system can receive other systems reply. 4. The system validates the request by sending the request information to the use case {U4.2.3} Detect Abnormalities (to detect abnormalities in the request). 5. The system updates the state to entrance/exit in an operation order. 6. The system sends the result of validation.
Alternative Flow	
Special Requirements	
Preconditions	The User (Driver or Operator) should have permission to perform entrance/exit and the data of the request should be clean, correct and useful.
Postconditions	
Extension Points	The execution of this use case depends on the operation orders schedule that results of {U1.1} Schedule Operation Orders and this use case can trigger {U4.2.3} Detect Abnormalities. Finally, this use case can be triggered within the execution of use case {U2.1.2} Perform Entrance/Exit.

Reference	{U2.2.3}
Name	Control Weighing
Short Description	Validates weighing requested by user and updates the state of an operation order.
Actors	
Basic Flow	1. The system receives the user identification to validate the weighing task to be performed. 2. The system can send the request to other systems for validation and reporting. 3. The system can receive other systems reply. 4. The system validates the request by sending the request information to the use case {U4.2.2} Detect Weighing Insecurity and to the use case {U4.2.3} Detect Abnormalities (to detect problems in the request). 5. The system updates the state to weighing in an operation order. 6. The system sends the result of validation.
Alternative Flow	
Special Requirements	
Preconditions	The User (Driver or Operator) should have permission to perform weighing and the data of the request should be clean, correct and useful.
Postconditions	
Extension Points	The execution of this use case depends on the operation orders schedule that results of {U1.1} Schedule Operation Orders and this use case can trigger {U4.2.2} Detect Weighing Insecurity and {U4.2.3} Detect Abnormalities. Lastly, this use case can be triggered within the execution of use case {U2.1.3} Perform Weighing.

Reference	{U2.2.4}
Name	Control Loading
Short Description	Validates loading requested by user and updates the state of an operation order.
Actors	
Basic Flow	1. The system receives the user identification to validate the loading task to be performed. 2. The system can send the request to other systems for validation and reporting. 3. The system can receive other systems reply. 4. The system validates the request by sending the request information continuously (until the loading task has finished) to the use case {U3.2.2} Monitor Loading (to detect problems in the request). 5. The system updates the state to loading in an operation order. 6. The system sends the result of validation.
Alternative Flow	
Special Requirements	
Preconditions	The User (Driver or Operator) should have permission to perform loading and the data of the request should be clean, correct and useful.
Postconditions	

Extension Points	The execution of this use case depends on the operation orders schedule that results of {U1.1} Schedule Operation Orders and this use case can trigger {U3.2.2} Monitor Loading. Finally, this use case can be triggered within the execution of use case {U2.1.4} Perform Loading.

Reference	{U2.2.5}
Name	Control Unloading
Short Description	Validates unloading requested by user and updates the state of an operation order.
Actors	
Basic Flow	1. The system receives the user identification to validate the unloading task to be performed. 2. The system can send the request to other systems for validation and reporting. 3. The system can receive other systems reply. 4. The system validates the request by sending the request information continuously (until the unloading task has finished) to the use case {U3.2.3} Monitor Unloading (to detect problems in the request). 5. The system updates the state to unloading in an operation order. 6. The system sends the result of validation.
Alternative Flow	
Special Requirements	
Preconditions	The User (Driver or Operator) should have permission to perform unloading and the data of the request should be clean, correct and useful.
Postconditions	
Extension Points	The execution of this use case depends on the operation orders schedule that results of {U1.1} Schedule Operation Orders and this use case can trigger {U3.2.3} Monitor Unloading. Lastly, this use case can be triggered within the execution of use case {U2.1.5} Perform Unloading.

Reference	{U2.2.6}
Name	Control Custom Tasks
Short Description	Validates custom task requested by user and updates the state of an operation order. This activity encompasses a range of other customized activities that can be used to control custom tasks in an operation order process.
Actors	
Basic Flow	1. The system receives the user identification to validate the custom task to be performed. 2. The system can send the request to other systems for validation and reporting. 3. The system can receive other systems reply. 4. The system validates the request by sending the request information to the use case {U3.2.4} Monitor Custom Tasks and/or to the use case {U4.2.4} Detect Custom Incidents (to detect problems in the request). 5. The system updates the state in an operation order. 6. The system sends the result of validation.
Alternative Flow	
Special Requirements	
Preconditions	The User (Driver or Operator) should have permission to custom tasks and the data of the request should be clean, correct and useful.
Postconditions	
Extension Points	The execution of this use case depends on the operation orders schedule that results of {U1.1} Schedule Operation Orders and this use case can trigger {U3.2.4} Monitor Custom Tasks and/or {U4.2.4} Detect Custom Incidents. Finally, this use case can be triggered within the execution of use case {U2.1.6} Perform Custom Tasks.

B.3.2.3 Monitor Activities Use Case Refinement

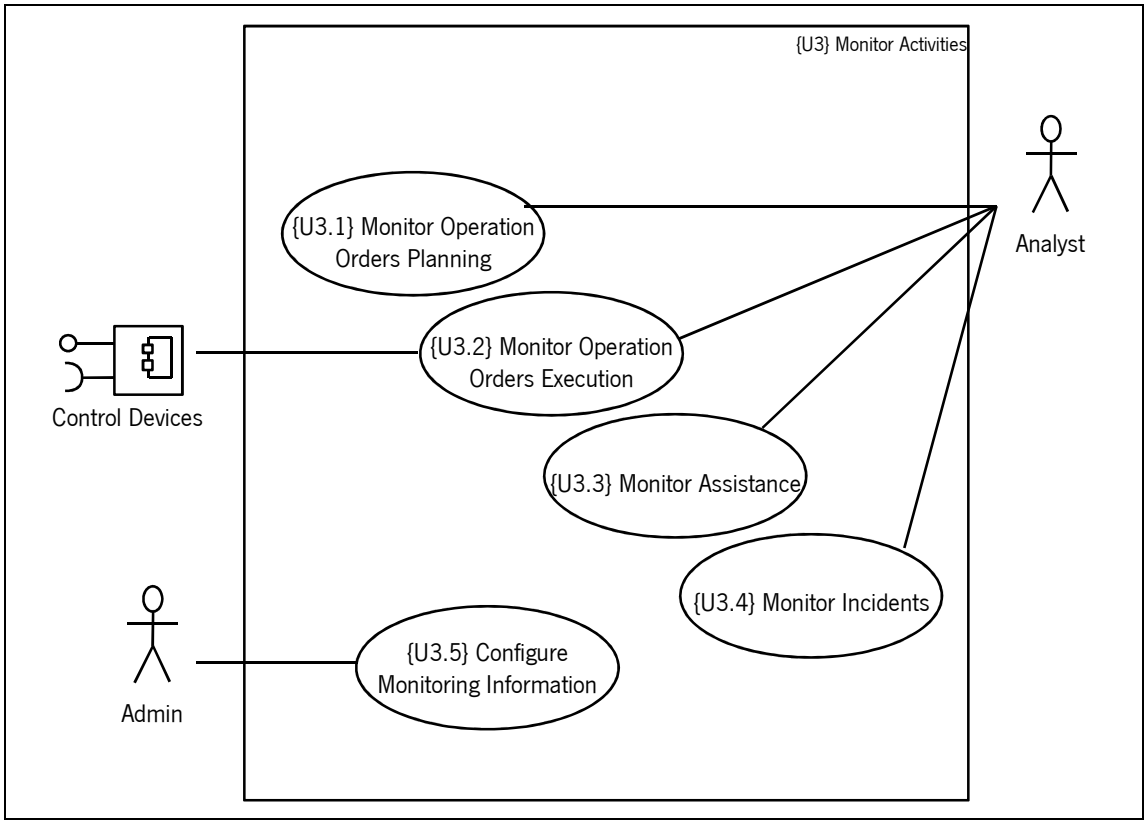


Figura A 16 – Monitor Activities use case refinement.

Reference	{U3.1}
Name	Monitor Operation Orders Planning
Short Description	Performs an analysis based on current and predicted operation order needs (operation orders are defined in use case {U.1} Plan and Control Operation Orders) to evaluate the node's offer.
Actors	Analyst
Basic Flow	1. Analyst requests the analysis of the operation orders plans. 2. Analyst receives the analysis of the operation orders plans.
Alternative Flow	
Special Requirements	
Preconditions	
Postconditions	
Extension Points	This use case outputs the evaluation result.

Reference	{U3.2}
Name	Monitor Operation Orders Execution
Short Description	Performs an analysis based on current and historical operation orders execution and allows the system to monitor tasks in an operation orders execution through control devices (e.g. weighing equipment, sensors and silos). By default, the SLT has three operation order monitor tasks ({U3.2.1} Monitor Node Availability, {U3.2.2} Monitor Loading, {U3.2.3} Monitor Unloading) and can also include new monitor tasks ({U3.2.4} Monitor Custom Tasks).
Actors	Analyst; Control Devices
Basic Flow	For Analyst: 1. Analyst requests the analysis of the operation orders executions. 2. Analyst receives the analysis of the operation orders executions. For Control Devices: 1. The system can receive the request to monitor task execution.

	2. The system/analyst requests the task execution values. 3. Control Devices measure task execution values. 4. Control Devices return measured values. 5. The system can send the measured values to use case {U4.2.3} Detect Abnormalities (to detect abnormalities). 6. The system can send the measured values to use case U4.2} Detect Incidents (to calculate or to detect an incident related to the task). 7. The system/analyst receives the result of the monitoring execution (success or failure) and the state of the task.
Alternative Flow	
Special Requirements	
Preconditions	
Postconditions	
Extension Points	Some use cases of this activity ({U3.2.1} Monitor Node Availability, {U3.2.2} Monitor Loading, {U3.2.3} Monitor Unloading and {U3.2.4} Monitor Custom Tasks) can be triggered within the execution of some use cases in {U2.2} Control Tasks use case ({U2.2.1} Control Check-in, {U2.2.4} Control Loading, {U2.2.5} Control Unloading and {U2.2.6} Control Custom Tasks).This use case outputs the evaluation result and the real time values of a task execution.

Reference	{U3.3}
Name	Monitor Assistance
Short Description	Performs an analysis based on current and historical operation orders assistances (operation orders assistances are defined in use case {U.5} Perform Assistances) to evaluate the performance of the nodes assistances.
Actors	Analyst
Basic Flow	1. Analyst requests the analysis of the operation orders assistances. 2. Analyst receives the analysis of the operation orders assistances.
Alternative Flow	
Special Requirements	
Preconditions	
Postconditions	
Extension Points	This use case outputs the evaluation result.

Reference	{U3.4}
Name	Monitor Incidents
Short Description	Performs an analysis based on current and historical operation orders incidents (operation orders incidents are defined in use case {U.4} Communicate Incidents) to evaluate the occurrence of the nodes incidents.
Actors	Analyst
Basic Flow	1. Analyst requests the analysis of the operation orders incidents. 2. Analyst receives the analysis of the operation orders incidents.
Alternative Flow	
Special Requirements	
Preconditions	
Postconditions	
Extension Points	This use case outputs the evaluation result.

Reference	{U3.5}
Name	Configure Monitoring Information
Short Description	Inserts, edits and instantiates information concerning activities from {U.3} Monitor Activities (except for this configuration activity). This information is used to recognize patterns in Users' needs (Analyst) and to adjust the activities of {U.3} Monitor Activities (except for this configuration activity) to these patterns. These configurations may change the aspect of the {U.3} Monitor Activities interfaces and may change the behavior associated to the activities of {U.3} Monitor Activities controls. For example, an Admin can configure the aspect of the interfaces and the

	availability (turn on and off) of the functionalities to fit the Users' needs. He can also define requirements concerns the behavior of the monitoring devices.
Actors	Admin
Basic Flow	1. Insert/Edit/Instantiate a set of given parameters, concerning a specific activity.
Alternative Flow	
Special Requirements	This use case must precede the first execution ever of use case {U3.1} Monitor Operation Orders Planning, use case {U3.2} Monitor Operation Orders Execution, use case {U3.3} Monitor Assistance and use case {U3.4} Monitor Incidents.
Preconditions	
Postconditions	
Extension Points	Use case {U3.1} Monitor Operation Orders Planning, use case {U3.2} Monitor Operation Orders Execution, use case U3.3} Monitor Assistance and use case {U3.4} Monitor Incidents are executed based on the configured information provided as input by this use case.

B.3.2.3.1 Monitor Operation Orders Execution Use Case Refinement

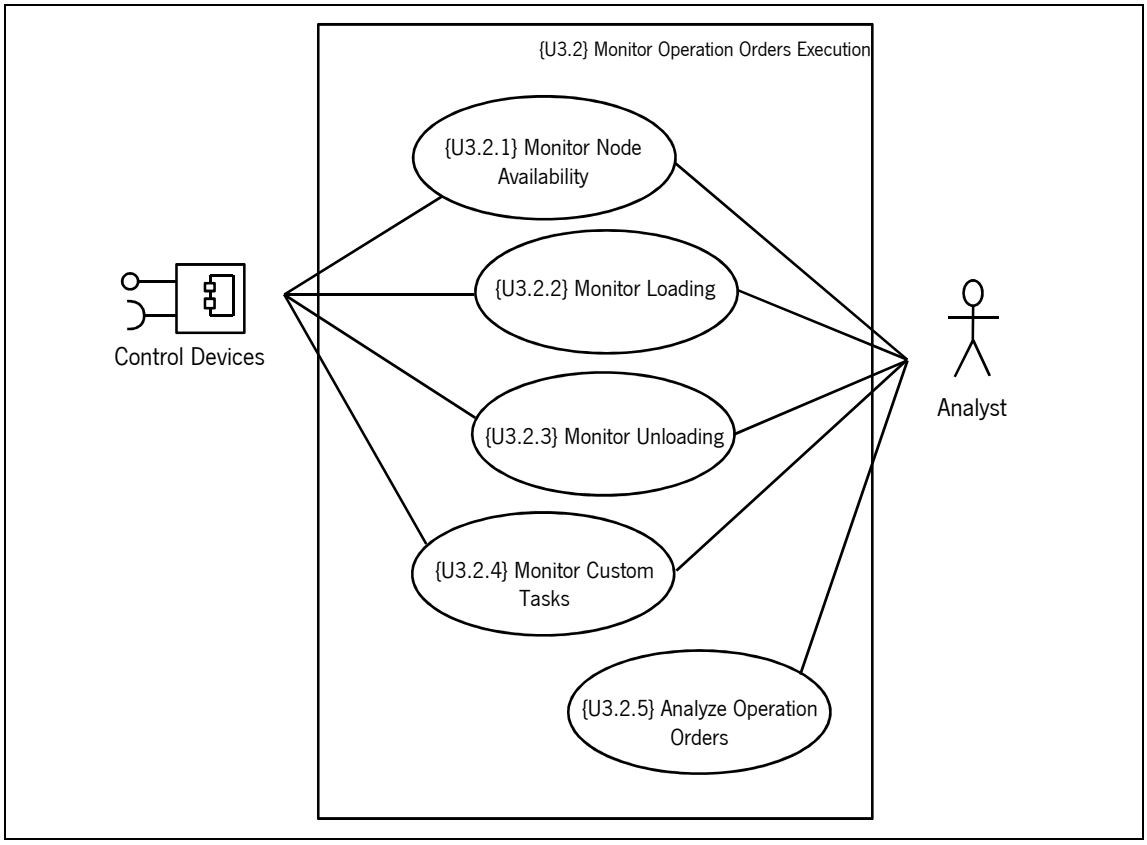


Figura A 17 – Monitor Operation Orders Execution use case refinement.

Reference	{U3.2.1}
Name	Monitor Node Availability
Short Description	Collects and analyzes current node availability. This node availability is constantly monitored and include information concerning the traffic of a node (e.g. How many and where are the trucks?).
Actors	Control Devices; Analyst
Basic Flow	1. The system can receive the request to monitor node availability. 2. The system/analyst requests the node availability. 3. Control Devices measure node availability. 4. Control Devices return measured values. 5. The system sends the measured values to use case {U4.2.3} Detect Abnormalities (to

	detect abnormalities in the node availability values and in the behavior of control devices). 6. The system sends the measured values to use case {U4.2.1} Detect Node Availability (to calculate the node availability). 7. The system/analyst receives the result of the monitoring execution (success or failure) and the state of node availability.
Alternative Flow	
Special Requirements	The node must have implemented monitoring devices through the entire circuit of an operation order execution.
Preconditions	The monitor loading task request can be provided by use case {U2.2.1} Control Check-in.
Postconditions	
Extension Points	The measured values can trigger use case {U4.2.1} Detect Node Availability.

Reference	{U3.2.2}
Name	Monitor Loading
Short Description	Collects and analyzes current loading values. These loading values are constantly monitored and include information concerning the weights of loads (e.g. net, gross and tare) and the position of loads (e.g. if the truck is well positioned).
Actors	Control Devices; Analyst
Basic Flow	1. The system receives the request to monitor loading task. 2. The system/analyst requests the loading values. 3. Control Devices measure loading values. 4. Control Devices return measured values. 5. The system sends the loading values to use case {U4.2.3} Detect Abnormalities (to detect abnormalities in the loading values and control devices). 6. The system/analyst receives the result of the loading execution (success or failure) and the state of the loading task.
Alternative Flow	
Special Requirements	The loading site must have implemented monitoring devices.
Preconditions	The monitor loading task request is provided by use case {U2.2.4} Control Loading.
Postconditions	
Extension Points	The measured values can trigger use case {U4.2.3} Detect Abnormalities.

Reference	{U3.2.3}
Name	Monitor Unloading
Short Description	Collects and analyzes current unloading values. These unloading values are constantly monitored and include information concerning the weights of unloads (e.g. net, gross and tare) and the position of unloads (e.g. if the truck is well positioned).
Actors	Control Devices; Analyst
Basic Flow	1. The system receives the request to monitor unloading task. 2. The system/analyst requests the unloading values. 3. Control Devices measure unloading values. 4. Control Devices return measured values. 5. The system sends the unloading values to use case {U4.2.3} Detect Abnormalities (to detect abnormalities in the unloading values and control devices). 6. The system/analyst receives the result of the unloading execution (success or failure) and the state of the unloading task.
Alternative Flow	
Special Requirements	The unloading site must have implemented monitoring devices.
Preconditions	The monitor unloading task request is provided by use case {U2.2.5} Control Unloading.
Postconditions	
Extension Points	The measured values can trigger use case {U4.2.3} Detect Abnormalities.

Reference	{U3.2.4}
Name	Monitor Custom Tasks

Short Description	Collects and analyzes current custom task values. These custom task values are constantly monitored and include customized information. This activity encompasses a range of other customized monitoring tasks that can be used to obtain custom task values in an operation order process.
Actors	Control Devices; Analyst
Basic Flow	1. The system receives the request to monitor custom task. 2. The system/analyst requests the custom task values. 3. Control Devices measure custom task values. 4. Control Devices return measured values. 5. The system can send the measured values to use case {U4.2.3} Detect Abnormalities (to detect abnormalities). 6. The system can send the measured values to use case {U4.2.4} Detect Custom Incidents (to calculate or to detect an incident related to the custom tasks). 7. The system/analyst receives the result of the monitoring execution (success or failure) and the state of custom tasks.
Alternative Flow	
Special Requirements	The custom task site must have implemented monitoring devices.
Preconditions	The monitor custom task request is provided by use case {U2.2.6} Control Custom Tasks.
Postconditions	
Extension Points	The measured values can trigger use case {U4.2.4} Detect Custom Incidents.

Reference	{U3.2.5}
Name	Analyze Operation Orders
Short Description	Performs an analysis based on current and historical operation order execution (operation order execution is defined in use case {U.2} Perform Operation Orders) to evaluate the node performance. This includes also the analysis of custom tasks that may be part of an operation order.
Actors	Analyst
Basic Flow	1. Analyst requests the analysis of operation order execution. 2. Analyst receives the analysis of operation order execution.
Alternative Flow	
Special Requirements	
Preconditions	
Postconditions	
Extension Points	This use case outputs the evaluation result.

B.3.2.4 Communicate Incidents Use Case Refinement

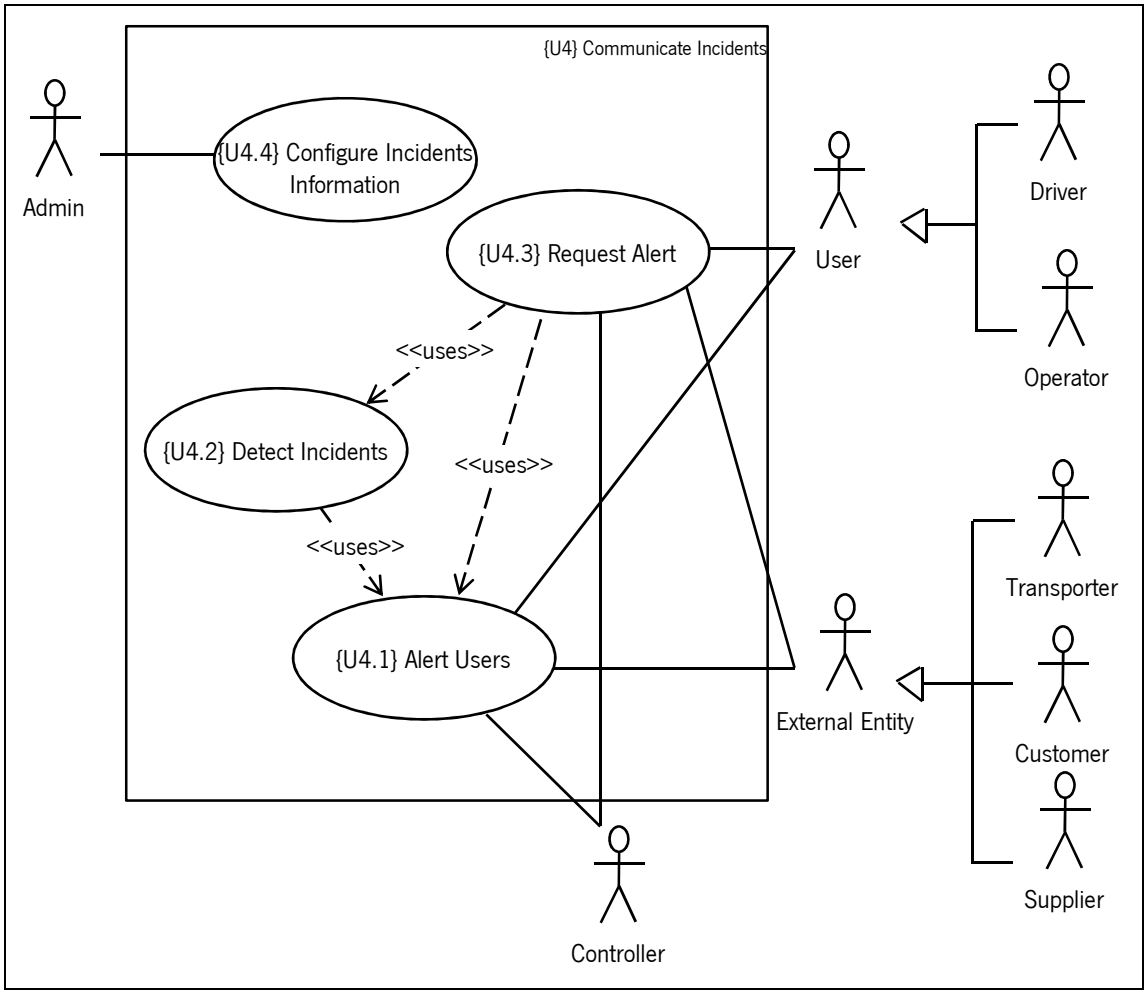


Figura A 18 – Communicate Incidents use case refinement.

Reference	{U4.1}
Name	Alert Users
Short Description	Alert specified users (User; External Entity; Controller) of an incident occurrence accordingly to the settings.
Actors	User; External Entity; Controller
Basic Flow	1. The system receives the incident occurrence information or receives the Users' request information. 2. The system sends alerts to specified users.
Alternative Flow	
Special Requirements	
Preconditions	This use case takes as input the incident detected in use case {U4.2} Detect Incidents, the user's alert request in use case {U4.3} Request Alert or the user's assistance request in use case {U5.1} Request Assistance.
Postconditions	
Extension Points	

Reference	{U4.2}
Name	Detect Incidents
Short Description	Refers to detecting patterns in a given data set that does not conform to an established normal system behavior or that conforms to an established system event that needs to be spread to a number of Users. Normally, it's an irregular situation or a relevant event that the SLT system considers.

Actors	
Basic Flow	1. The system receives a request to detect incidents. 2. The system detects an incident. 3. The system sends incident occurrence information to alert users.
Alternative Flow	
Special Requirements	
Preconditions	The detect incident request can be provided by use case: - {U1.4} Control Operation Orders Schedule; - {U3.2.1} Monitor Node Availability; - {U2.2.2} Control Entrance/Exit; - {U2.2.3} Control Weighing; - {U3.2.2} Monitor Loading; - {U3.2.3} Monitor Unloading; - {U3.2.4} Monitor Custom Tasks; - and {U4.3} Request Alert.
Postconditions	
Extension Points	This use case is able to trigger the execution of {U4.1} Alert Users.

Reference	{U4.3}
Name	Request Alert
Short Description	Defines an alert message to be sent to a set of users, allowing the exchange of alerts between users.
Actors	User; External Entity; Controller
Basic Flow	1. User, External Entity or Controller requests to send an alert to a set of users. 2. The system sends alert information to alert a set of specified Users. 3. User, External Entity or Controller receives the confirmation that the alert was delivered.
Alternative Flow	
Special Requirements	
Preconditions	
Postconditions	
Extension Points	This use case is able to trigger the execution of {U4.1} Alert Users.

Reference	{U4.4}
Name	Configure Incidents Information
Short Description	Inserts, edits and instantiates information concerning activities from {U.4} Communicate Incidents (except for this configuration activity). This information is used to recognize patterns in Users' needs (User; External Entity; Controller) and to adjust the activities of {U.4} Communicate Incidents (except for this configuration activity) to these patterns. These configurations may change the aspect of the {U.4} Communicate Incidents interfaces and may change the behavior associated to the activities of {U.4} Communicate Incidents controls. For example, an Admin can configure the aspect of the interfaces and also define new rules to detect incidents.
Actors	Admin
Basic Flow	1. Insert/Edit/Instantiate a set of given parameters, concerning a specific activity.
Alternative Flow	
Special Requirements	This use case must precede the first ever execution of use case {U4.1} Alert Users, use case {U4.2} Detect Incidents and use case {U4.3} Request Alert.
Preconditions	
Postconditions	
Extension Points	Use case {U4.1} Alert Users, use case {U4.2} Detect Incidents and use case {U4.3} Request Alert are executed based on the configured information provided as input by this use case.

B.3.2.4.1 Detect Incidents Use Case Refinement

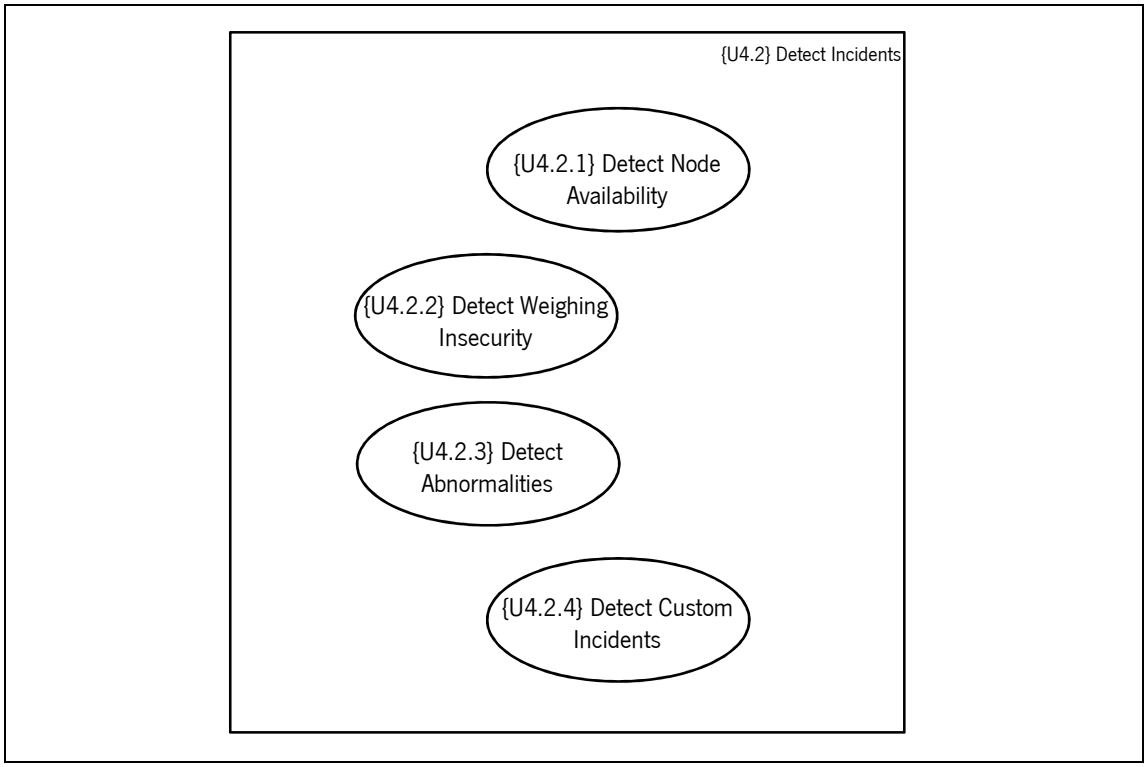


Figura A 19 – Detect Incidents use case refinement.

Reference	{U4.2.1}
Name	Detect Node Availability
Short Description	Refers to detecting patterns in a given data set that indicates the availability of a node in performing an Operation Order. Normally the availability of a node is determined by the operation order priority, the number of operation orders that a node can process at the moment and the scheduled operation orders.
Actors	
Basic Flow	1. The system receives a request to detect node availability. 2. The system detects node availability. 3. The system sends node availability information to alert users.
Alternative Flow	
Special Requirements	
Preconditions	The detect node availability request can be provided by use case {U3.2.1} Monitor Node Availability and use case {U4.3} Request Alert.
Postconditions	
Extension Points	This use case is able to trigger the execution of {U4.1} Alert Users.

Reference	{U4.2.2}
Name	Detect Weighing Insecurity
Short Description	Refers to detecting patterns in a given data set that indicates a weighing insecurity in the weighing task. Typically a weighing insecurity is determined when the loads are incorrectly positioned or the weighing equipment are not calibrated.
Actors	
Basic Flow	1. The system receives a request to detect weighing insecurity. 2. The system detects a weighing insecurity. 3. The system sends weighing insecurity information to alert users.
Alternative Flow	
Special Requirements	
Preconditions	The detect weighing insecurity request can be provided by use case {U2.2.3} Control Weighing and use case {U4.3} Request Alert.

Postconditions	
Extension Points	This use case is able to trigger the execution of {U4.1} Alert Users.

Reference	{U4.2.3}
Name	Detect Abnormalities
Short Description	Refers to detecting patterns in a given data set that does not conform to an established normal system behavior. Normally, it is an irregular situation that the SLT system considers.
Actors	
Basic Flow	1. The system receives a request to detect abnormalities. 2. The system detects an abnormality. 3. The system sends abnormality information to alert users.
Alternative Flow	
Special Requirements	
Preconditions	The detect abnormalities request can be provided by use case: - {U1.4} Control Operation Orders Schedule; - {U2.2.2} Control Entrance/Exit; - {U2.2.3} Control Weighing; - {U3.2.1} Monitor Node Availability; - {U3.2.2} Monitor Loading; - {U3.2.3} Monitor Unloading; - {U3.2.4} Monitor Custom Tasks; - and {U4.3} Request Alert.
Postconditions	
Extension Points	This use case is able to trigger the execution of {U4.1} Alert Users.

Reference	{U4.2.4}
Name	Detect Custom Incidents
Short Description	Refers to detecting patterns in a given data set that indicates the occurrence of a custom incident. This activity encompasses a range of other customized incidents that can be created by Admin.
Actors	
Basic Flow	1. The system receives a request to detect custom incident. 2. The system detects a custom incident. 3. The system sends custom incident information to alert users.
Alternative Flow	
Special Requirements	
Preconditions	The detect abnormalities request can be provided by use case {U3.2.4} Monitor Custom Tasks and use case {U4.3} Request Alert.
Postconditions	
Extension Points	This use case is able to trigger the execution of {U4.1} Alert Users.

B.3.2.5 Perform Assistances Use Case Refinement

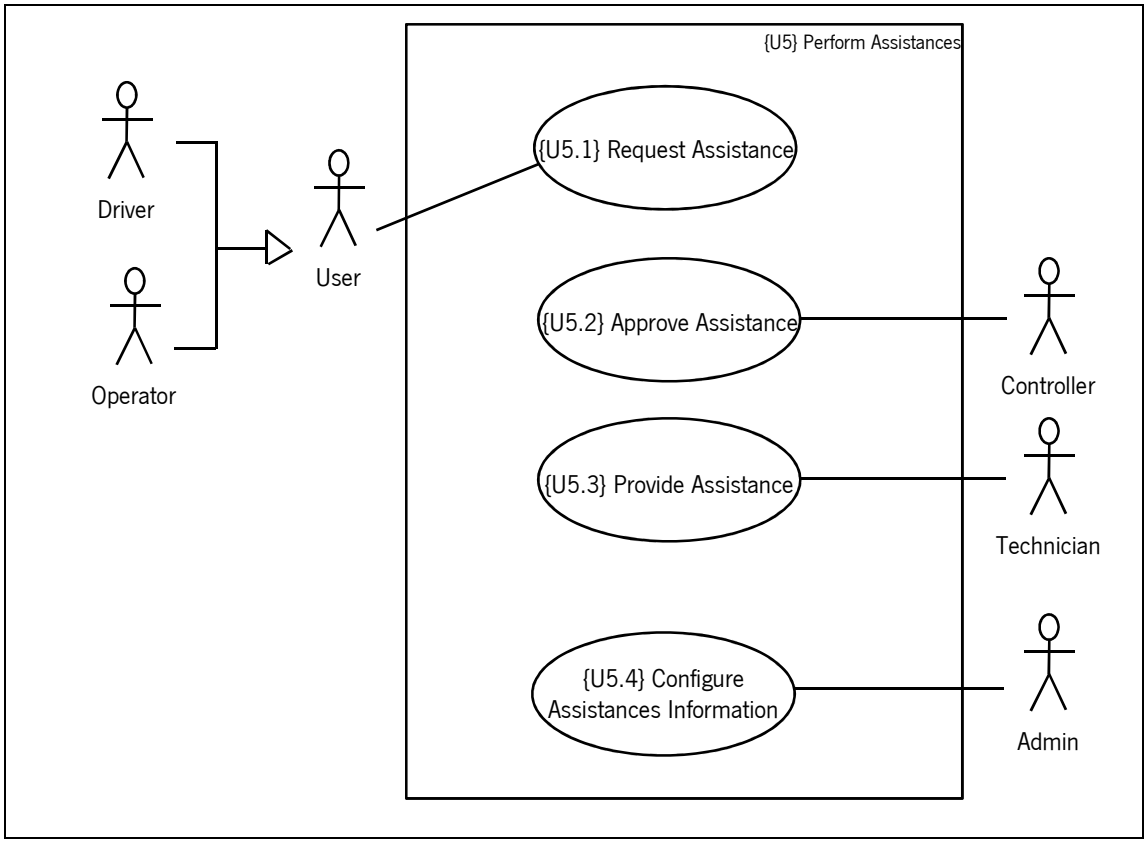


Figura A 20 – Perform Assistances use case refinement.

Reference	{U5.1}
Name	Request Assistance
Short Description	Defines an assistance request in order to trigger an emergency alert.
Actors	User
Basic Flow	1. User requests assistance. 2. The system sends request information to alert users (Controller). 3. User receives the confirmation that the request is being processed.
Alternative Flow	
Special Requirements	
Preconditions	
Postconditions	
Extension Points	This use case is able to trigger the execution of {U4.1} Alert Users.

Reference	{U5.2}
Name	Approve Assistance
Short Description	Accepts an assistance request and notify the assistance technicians.
Actors	Controller
Basic Flow	1. Controller approves assistance. 2. Controller gets in touch with a Technician, providing the necessary information regarding the emergency situation. 3. Technician confirms the calling and provides assistance. 4. Controller can get in touch with External Entity to alert for a delay situation.
Alternative Flow	
Special Requirements	
Preconditions	
Postconditions	
Extension Points	

Reference	{U5.3}
Name	Provide Assistance
Short Description	Offers remote and onsite assistance services to facilitate the execution or to solve a problem of an operation order.
Actors	Technician
Basic Flow	1. Controller provides the necessary information to proceed with assistance. 2. Technician accepts the request.
Alternative Flow	
Special Requirements	
Preconditions	
Postconditions	
Extension Points	

Reference	{U5.4}
Name	Configure Assistances Information
Short Description	Inserts, edits and instantiates information concerning activities from {U.5} Perform Assistances (except for this configuration activity). This information is used to recognize patterns in Users' needs (User, Controller and Technician) and to adjust the activities of {U.5} Perform Assistances (except for this configuration activity) to these patterns. These configurations may change the aspect of the {U.5} Perform Assistances interfaces. For example, an Admin can configure the aspect of the interfaces.
Actors	Admin
Basic Flow	1. Insert/Edit/Instantiate a set of given parameters, concerning a specific activity.
Alternative Flow	
Special Requirements	This use case must precede the first ever execution of use case {U5.1} Requests Assistance, use case {U5.2} Approve Assistance and use case {U5.3} Provide Assistance.
Preconditions	
Postconditions	
Extension Points	Use case {U5.1} Requests Assistance, use case {U5.2} Approve Assistance and use case {U5.3} Provide Assistance are executed based on the configured information provided as input by this use case.

B.3.2.5.1 Provide Assistance Use Case Refinement

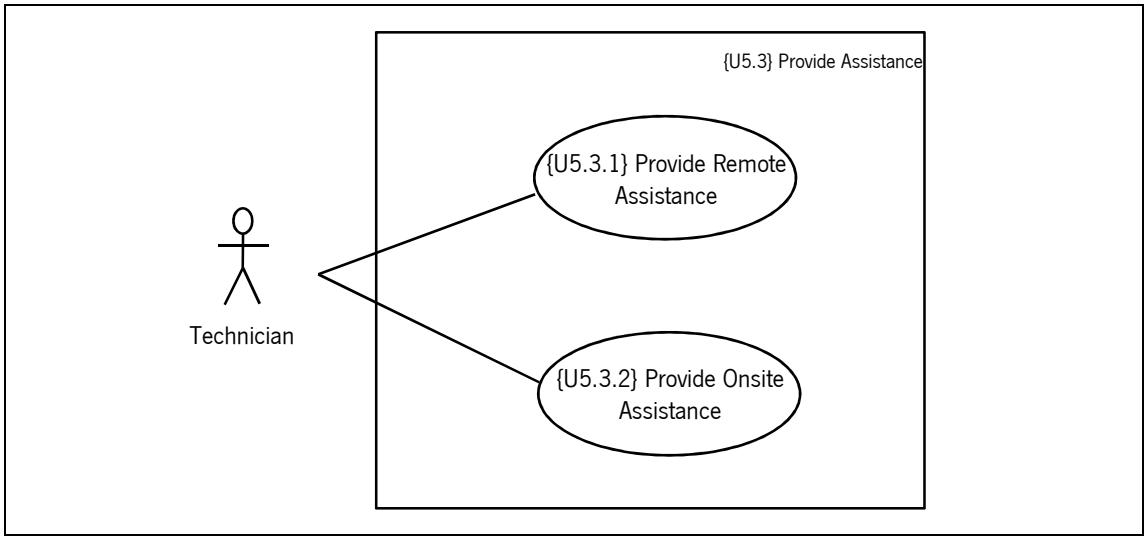


Figura A 21 – Provide Assistance use case refinement.

Reference	{U5.3.1}
Name	Provide Remote Assistance
Short Description	Execute remote assistance to a user or system.
Actors	Technician
Basic Flow	1. Controller provides the necessary information to proceed with a remote assistance. 2. Technician accepts the request.
Alternative Flow	
Special Requirements	
Preconditions	
Postconditions	
Extension Points	

Reference	{U5.3.2}
Name	Provide Onsite Assistance
Short Description	Execute onsite assistance to a user or system.
Actors	Technician
Basic Flow	1. Controller provides the necessary information to proceed with an onsite assistance. 2. Technician accepts the request.
Alternative Flow	
Special Requirements	
Preconditions	
Postconditions	
Extension Points	

B.4 Process Logical Architecture

The 4SRS method execution ends with the construction of a logical diagram which represents the logical architecture of the process-level SLT Solution functionalities. The architecture is composed by the architecture elements that survived after execution of step 2. The packaging executed in step 3 allows the identification of major processes. The associations identified in step 4 are represented in the diagram by the connections between the architecture elements (for readability

purposes, the “direct associations” were represented in dashed lines, and the “use case model associations” in straight lines).

Looking at the derived logical diagram (in the next figure), it is possible to conclude that there is a noticeable interdependency between the macro-packages of the presented logical architecture. Additionally, it is possible to identify that there are major processes which have a temporal dependency ({P1} Schedule Planning, {P2} Operation Order Execution, {P3.2} Task Monitoring, {P4} Incident Communication and {P5} Assistances) and others that are cross-cut to the entire system ({P6} Configurations and {P3.1} Data Monitoring).

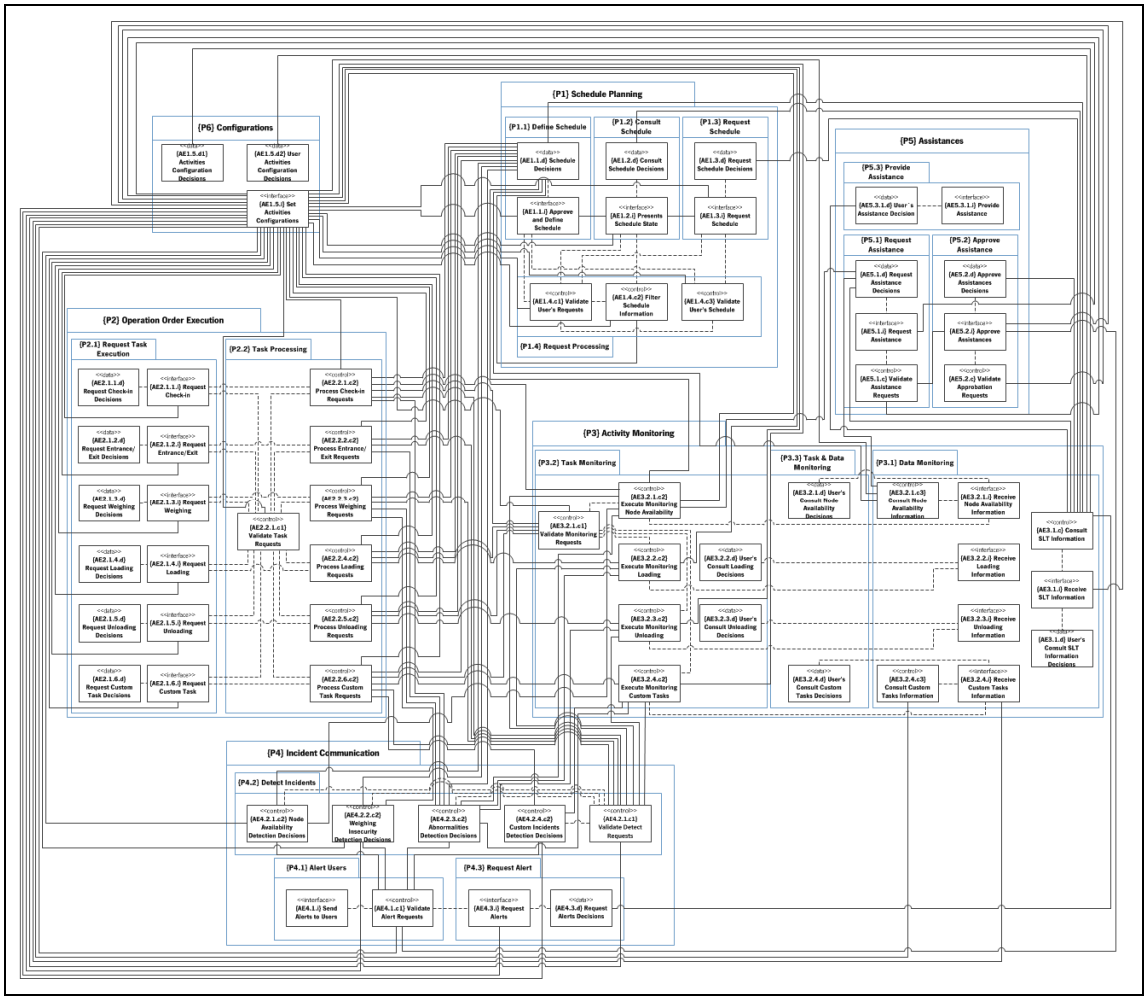


Figura A 22 – SLT Process-level Logical Architecture.

For {P1} Schedule Planning, {P2} Operation Order Execution, {P3} Activity Monitoring, {P4} Incident Communication and {P5} Assistances processes the typical execution starts with an input from human decisions architectural elements, then the information flow is treated by system decisions architectural elements and finalizes with an output in an interface architectural element. Only the execution of {P6} Configurations process starts from human decisions

architectural elements and go directly to the presentation layer because there is no need for any decision within the SLT Solution at a process-level perspective.

B.4.1 Architectural Elements Descriptions

In this subsection is described the architecture elements that exist in the logical diagram. The description of the AEs includes the process detail, inputs and outputs, actors, goals, constraints, etc.

Architectural Element Name:	Schedule Decisions
Reference:	{AE1.1.d}
Description:	Make decisions on what operation orders should be included in the node's schedule. These decisions are made according to the node's needs and External Entity's needs that can be presented in {AE1.2.i}. This AE outputs information that is used in {AE1.1.i}.

Architectural Element Name:	Approve and Define Schedule
Reference:	{AE1.1.i}
Description:	The Programmer's and External Entity's schedule requests (from {AE1.4.c3} or {AE1.4.c4}) can be rejected and/or approved and the operation orders are scheduled (periodically or punctually) by a Programmer with permissions, both according to the decisions made in {AE1.1.d}. The resulting schedule (contains a list of operation orders with the date and time, the estimated duration, the site and the detail of each operation order, the information about the user, customers/suppliers and/or carriers, etc.) expresses the node's or Programmer's needs and it is sent to {AE1.4.c3} for validation. The result of the validation and External Entity' needs is presented in this AE for consultation, so this AE must include the {AE1.2.i}. After the validation, the final schedule is used and is updated in all existing control tasks for each operation order ({AE2.2.1.c}, {AE2.2.2.c}, {AE2.2.3.c}, {AE2.2.4.c}, {AE2.2.5.c} and {AE2.2.6.c})

Architectural Element Name:	Consult Schedule Decisions
Reference:	{AE1.2.d}
Description:	Make decisions on what information (regarding current operation orders schedule state) the user (Programmer or External Entity) needs or has preference to access, inside allowed limits (defined in {AE1.5.i}). For instance, the user (Programmer or External Entity) can customize interfaces, layouts or languages and create your own queries. This AE outputs information that is used in {AE1.2.i}.

Architectural Element Name:	Presents Schedule State
Reference:	{AE1.2.i}
Description:	Request and receives information regarding current operation orders schedule state to and from {AE1.4.c2}, according to the decisions made in {AE1.2.d}. This information is constantly monitored by Programmer and External Entity to aid them to make their own decisions in {AE1.1.d} and {AE1.3.d}. So this AE can be included in {AE1.1.i} and {AE1.3.i}.

Architectural Element Name:	Request Schedule Decisions
Reference:	{AE1.3.d}
Description:	Making decisions on what operation orders should be included in the node's

	schedule. These decisions are made according to the External Entity's needs and node's availability that can be presented in {AE1.2.i}. This AE outputs information that is used in {AE1.3.i}.

Architectural Element Name:	Request Schedule
Reference:	{AE1.3.i}
Description:	The operation orders are scheduled (periodically or punctually) according to the decisions made in {AE1.3.d}. The resulting schedule (includes a list of operation orders, the desired date, the site to perform each operation order, etc.) expresses the External Entity's needs and it is sent to {AE1.4.c4} for validation. The result of the validation and the node's availability can be presented in this AE for consultation, so this AE should include the {AE1.2.i}.

Architectural Element Name:	Validate User's Requests
Reference:	{AE1.4.c1}
Description:	Validate the requests of the Programmer and the External Entity (from {AE1.1.i}, {AE1.2.i} and {AE1.3.i}) accordingly to the configuration settings defined on {AE1.5.i} (data requirements/users). For example, verify if the data of request is clean, correct and useful and if the user has permission to get the request satisfied. This AE outputs a boolean response.

Architectural Element Name:	Filter Schedule Information
Reference:	{AE1.4.c2}
Description:	Filter the schedule information accordingly to the configuration settings defined on {AE1.5.i} (filters/users) and the requests from {AE1.2.i}. Its aim is to gather the appropriate information about the node's current schedule state (stored in {AE1.1.d}) and to present it (also in real-time) to users (Programmer and External Entity trough {AE1.2.i}), so they can make better decisions about their own needs. This AE can also restrict some information for specific users. For example, it is possible exist one business rule that can do some users to have more dates available than others for reservations. In the end, this AE outputs the filtered information about the availability of node's operation orders and the current state of user's operation orders (Programmer and External Entity) and it is used by {AE1.2.i}.

Architectural Element Name:	Validate User's Schedule
Reference:	{AE1.4.c3}
Description:	Validate the requested schedule by Programmer {AE1.1.i} or External Entity {AE1.3.i} accordingly to the configuration settings defined on {AE1.5.i} (node's needs/users). For example, validate if is viable to include the request in the node's schedule (e.g. checks if there is some business rule that not allow the user's reservation). In the case that the request is viable and the Programmer or External Entity has not enough privileges, this AE can send that request to (other) Programmer for approbation ({AE1.1.i}). This AE outputs the state of the request (rejected, approved or waiting for approbation).

Architectural Element Name:	Activities Configuration Decisions
Reference:	{AE1.5.d1}
Description:	Make decisions on configuration requirements for all activities to fit the needs of all users and the needs of the node. These users associated to each of these activities are: - Plan and Control ({AE1.1.i}, {AE1.2.i}, {AE1.3.i} and {AE1.4.c}) => Programmer and External Entity (Transporter, Customer or Supplier); - Perform Operation Orders ({AE2.1.1.i}, {AE2.1.2.i}, ..., {AE2.1.N.i} and {AE2.2.1.c}, {AE2.2.2.c}, ..., {AE2.2.N.c}) => User (Driver or Operator); - Monitor ({AE3.1.i}, ..., {AE3.N.i}, {AE3.1.c}, ..., {AE3.N.c} and {AE3.1.c2}, ...,

	{AE3.N.c2}) => Analyst; - Communicate Incidents ({AE4.1.i}, {AE4.2.1.c2}, {AE4.2.2.c2}, {AE4.2.3.c2}, {AE4.2.4.c2} and {AE4.3.i}) => User (Driver or Operator), External Entity (Transporter, Customer or Supplier) and Controller; - and Perform Assistances ({AE5.1.i}, {AE5.2.i}, {AE5.3.1.i} and {AE5.3.2.i} => User (Driver or Operator), Controller and Technician. For instance, the Admin can customize interfaces, layouts or languages, turn on and off functionalities, apply new filters to queries (for all activities), define business rules to change node's business logic (for Plan and Control and Perform Operation Orders), define what tasks are required as well as what is the sequence of them (for Perform Operation Orders activity), define what incidents are required as well as when and for who the notifications should be sent (for Communicate Incidents) and associate each of these configurations to different users. These decisions will generate a specific configuration.

Architectural Element Name:	User Activities Configuration Decisions
Reference:	{AE1.5.d2}
Description:	Make decisions regarding the information used to recognize patterns in users' needs. The users are: - Programmer; - External Entity (Transporter, Customer or Supplier); - User (Driver or Operator); - Analyst; - Controller; - and Technician.

Architectural Element Name:	Set Activities Configurations
Reference:	{AE1.5.i}
Description:	Insert configuration information. The configuration settings operationalize the decisions made in {AE1.5.d1}, {AE2.3.d1}, {AE3.5.d1}, {AE4.4.d1}, {AE5.4.d1}. The configuration is also concerned with the user's requests to be able to recognize the user's needs (defined in {AE1.5.d2}, {AE2.3.d2}, {AE3.5.d2}, {AE4.4.d2} and {AE5.4.d2}).

Architectural Element Name:	Request Check-in Decisions
Reference:	{AE2.1.1.d}
Description:	Make decisions regarding the information used to perform a check-in task.

Architectural Element Name:	Request Entrance/Exit Decisions
Reference:	{AE2.1.2.d}
Description:	Make decisions regarding the information used to perform an entrance/exit task.

Architectural Element Name:	Request Weighing Decisions
Reference:	{AE2.1.3.d}
Description:	Make decisions regarding the information used to perform a weighing task.

Architectural Element Name:	Request Loading Decisions
Reference:	{AE2.1.4.d}
Description:	Make decisions regarding the information used to perform a loading task.

Architectural Element Name:	Request Unloading Decisions
Reference:	{AE2.1.5.d}
Description:	Make decisions regarding the information used to perform an unloading task.

Architectural Element Name:	Request Custom Task Decisions
Reference:	{AE2.1.6.d}

Description:	Make decisions regarding the information used to perform a custom task.

Architectural Element Name:	Request Check-in
Reference:	{AE2.1.1.i}
Description:	User's identification and all the required information (according to the decisions made in {AE2.1.1.d}) for fulfill a check-in task is presented by User (Driver or Operator). After that, this information is sent to {AE2.2.1.c2} to validate and to proceed the check-in task. In the end, this AE receives and presents the result of the check-in task execution (success or failure).

Architectural Element Name:	Request Entrance/Exit
Reference:	{AE2.1.2.i}
Description:	User's identification and all the required information (according to the decisions made in {AE2.1.2.d}) for fulfill an entrance/exit task is presented by User (Driver or Operator). After that, this information is sent to {AE2.2.2.c2} to validate and to proceed the entrance/exit task. In the end, this AE receives and presents the result of the entrance/exit task execution (success or failure).

Architectural Element Name:	Request Weighing
Reference:	{AE2.1.3.i}
Description:	User's identification and all the required information (according to the decisions made in {AE2.1.3.d}) for fulfill a weighing task is presented by User (Driver or Operator). After that, this information is sent to {AE2.2.3.c2} to validate and to proceed the weighing task. In the end, this AE receives and presents the result of the weighing task execution (success or failure).

Architectural Element Name:	Request Loading
Reference:	{AE2.1.4.i}
Description:	User's identification and all the required information (according to the decisions made in {AE2.1.4.d}) for fulfill a loading task is presented by User (Driver or Operator). After that, this information is sent to {AE2.2.4.c2} to validate and to proceed the loading task. In the end, this AE receives and presents the result of the loading task execution (success or failure).

Architectural Element Name:	Request Unloading
Reference:	{AE2.1.5.i}
Description:	User's identification and all the required information (according to the decisions made in {AE2.1.5.d}) for fulfill an unloading task is presented by User (Driver or Operator). After that, this information is sent to {AE2.2.5.c2} to validate and to proceed the unloading task. In the end, this AE receives and presents the result of the unloading execution (success or failure).

Architectural Element Name:	Request Custom Task
Reference:	{AE2.1.6.i}
Description:	User's identification and all the required information (according to the decisions made in {AE2.1.6.d}) for fulfill a custom task is presented by User (Driver or Operator). After that, this information is sent to {AE2.2.6.c2} to validate and to proceed the custom task. In the end, this AE receives and presents the result of the custom task execution (success or failure). It is possible exist different custom tasks, each one of them can be defined in {AE2.3.i}.

Architectural Element Name:	Validate Task Requests
Reference:	{AE2.2.1.c1}
Description:	Validate the requests of the User (Driver or Operator) from {AE2.1.1.i}, {AE2.1.2.i}, {AE2.1.3.i}, {AE2.1.4.i}, {AE2.1.5.i} and {AE2.1.6.i} accordingly to the configuration settings defined on {AE2.3.i} (data requirements/users). For

	example, verify if the data of request is clean, correct and useful and if the user has permission to get the request satisfied. This AE outputs a boolean response.

Architectural Element Name:	Process Check-in Requests
Reference:	{AE2.2.1.c2}
Description:	Process the requested check-in task by User (from {AE2.1.1.i}) accordingly to the configuration settings defined on {AE2.3.i} (tasks/users). This AE receives the request information, sends the request information to {AE4.2.3.c2} (to detect abnormalities), updates the operation order state in {AE1.1.d} and/or other systems with the check-in information and it can trigger the {AE3.2.1.c2}, to detect the node availability. The AE output is the result of the check-in execution task (success or failure). Normally, to detect a check-in abnormality (in {AE4.2.3.c2}) it is through the validation if there is or not an operation order associated to the requested check-in in the SLT and/other systems. In the case that there isn't match, the Programmer or Controller can receive an alert with an indication that there is a problem to be solved. Typically, the problem is solved by creating a new operation order with the check-in request information.

Architectural Element Name:	Process Entrance/Exit Requests
Reference:	{AE2.2.2.c2}
Description:	Process the requested entrance/exit task by User (from {AE2.1.2.i}) accordingly to the configuration settings defined on {AE2.3.i} (tasks/users). This AE receives the request information, sends the request information to {AE4.2.3.c2} (to detect abnormalities) and updates the operation order state in {AE1.1.d}. The AE output is the result of the entrance/exit execution task (success or failure). Normally, to detect an entrance/exit abnormality (in {AE4.2.3.c2}) it is through the validation if there is or not an operation order associated to the requested entrance/exit in the SLT and/other systems. In the case that there isn't match, the Programmer or Controller can receive an alert with an indication that there is a problem to be solved.

Architectural Element Name:	Process Weighing Requests
Reference:	{AE2.2.3.c2}
Description:	Process the requested weighing task by User (from {AE2.1.3.i}) accordingly to the configuration settings defined on {AE2.3.i} (tasks/users). This AE receives the request information, sends the request information to {AE4.2.2.c2} and {AE4.2.3.c2} (to detect weighing problems) and updates the operation order state in {AE1.1.d}. The AE output is the result of the weighing execution task (success or failure). Typically, to detect a weighing insecurity (in {AE4.2.2.c2}) it is through the validation if the loads are correctly/incorrectly positioned or the weighing equipment is calibrated or not. Normally, to detect a weighing abnormality (in {AE4.2.3.c2}) it is through the validation if there is or not an operation order associated to the requested weighing in the SLT and/other systems. In the case that there isn't match, the Programmer or Controller can receive an alert with an indication that there is a problem to be solved.

Architectural Element Name:	Process Loading Requests
Reference:	{AE2.2.4.c2}
Description:	Process the requested loading task by User (from {AE2.1.4.i}) accordingly to the configuration settings defined on {AE2.3.i} (tasks/users). This AE receives the request information, it continuously (until the loading task has finished) sends the current loading state information to {AE3.2.2.c2} (to detect problems), and updates the operation order state in {AE1.1.d}. The AE output is the result of the loading execution task (success or failure).

	Normally, to detect a loading problem (in {AE3.2.2.c2}) it is through the validation if there is or not an operation order associated to the requested loading in the SLT and/other systems and if the loading equipment is operating correctly in the entire operation. In the case that there isn't match or the loading equipment fail, the Programmer or Controller can receive an alert with an indication that there is a problem to be solved.

Architectural Element Name:	Process Unloading Requests
Reference:	{AE2.2.5.c2}
Description:	Process the requested unloading task by User (from {AE2.1.5.i}) accordingly to the configuration settings defined on {AE2.3.i} (tasks/users). This AE receives the request information, it continuously (until the unloading task has finished) sends the current unloading state information to {AE3.2.3.c2} (to detect problems), and updates the operation order state in {AE1.1.d}. The AE output is the result of the unloading execution task (success or failure). Normally, to detect an unloading problem (in {AE3.2.3.c2}) it is through the validation if there is or not an operation order associated to the requested unloading in the SLT and/other systems and if the unloading equipment is operating correctly in the entire operation. In the case that there isn't match or the unloading equipment fail, the Programmer or Controller can receive an alert with an indication that there is a problem to be solved.

Architectural Element Name:	Process Custom Task Requests
Reference:	{AE2.2.6.c2}
Description:	Process the requested custom task by User (from {AE2.1.6.i}) accordingly to the configuration settings defined on {AE2.3.i} (tasks/users). This AE receives the request information, sends the request information to {AE3.2.4.c2} and/or {AE4.2.4.c2} (to detect problems) and updates the operation order state in {AE1.1.d} and/or other systems with the custom task information. The AE output is the result of the custom task execution (success or failure).

Architectural Element Name:	Consult SLT Information
Reference:	{AE3.1.c}
Description:	Consult and make decisions regarding the SLT activities information in order to build reports.

Architectural Element Name:	User's Consult SLT Information Decisions
Reference:	{AE3.1.d}
Description:	Make decisions on what information the user (Analyst) needs or has preference to access, inside allowed limits (defined in {AE3.5.i}). For instance, the user (Analyst) can customize interfaces, layouts or languages and create your own queries.

Architectural Element Name:	Receive SLT Information
Reference:	{AE3.1.i}
Description:	Receives SLT activities from {AE3.1.c}, {AE3.2.5.c}, {AE3.3.c} and {AE3.4.c} according to the decisions made in {AE3.1.d, {AE3.2.5.d}, {AE3.3.d} and {AE3.4.d}. This information is constantly monitored by Analyst to evaluate the performance of SLT.

Architectural Element Name:	Validate Monitoring Requests
Reference:	{AE3.2.1.c1}
Description:	Validate the monitoring request from {AE2.2.1.c2}, {AE2.2.4.c2}, {AE2.2.5.c2} and {AE2.2.6.c2} accordingly to the configuration settings defined on {AE3.5.i} (data requirements/users). For example, verify if the data of request is clean, correct and useful. This AE outputs a boolean response and it is only executed

	for {AE3.2.1.c2}, {AE3.2.2.c2}, {AE3.2.3.c2} and {AE3.2.4.c2} AEs.

Architectural Element Name:	Execute Monitoring Node Availability
Reference:	{AE3.2.1.c2}
Description:	Process the requested monitor node availability by {AE2.2.1.c2} accordingly to the configuration settings defined on {AE3.5.i} (monitoring tasks/users). This AE receives the request information, collects information from Control Devices, implemented in the entire circuit of an operation order execution, from {AE1.1.d} and {AE5.1.d}, sends the collected information to {AE4.2.3.c2} (to detect abnormalities in the node availability values and in the behavior of control devices) and to {AE4.2.1.c2} (to calculate the node availability). The AE output is the result of the monitoring execution (success or failure) and the state of node availability.

Architectural Element Name:	Consult Node Availability Information
Reference:	{AE3.2.1.c3}
Description:	Consult and make decisions regarding the information of node availability (in {AE3.2.1.c2}, {AE1.1.d} and {AE5.1.d}) in order to build a report.

Architectural Element Name:	User's Consult Node Availability Decisions
Reference:	{AE3.2.1.d}
Description:	Makes decisions on how the measured information from {AE3.2.1.i} is used and on what information (current and historical node availability) the user (Analyst) needs or has preference to access, inside allowed limits (defined in {AE3.5.i}). The information can be used by the SLT/Analyst for preventing traffic congestion in the node or to follow measured values through times. This AE defines for who the values are available and the user (Analyst) can customize interfaces, layouts or languages and create your own queries.

Architectural Element Name:	Receive Node Availability Information
Reference:	{AE3.2.1.i}
Description:	Receives the current values of node availability from {AE3.2.1.c2} and a report from {AE3.2.1.c3}.

Architectural Element Name:	Execute Monitoring Loading
Reference:	{AE3.2.2.c2}
Description:	Process the requested monitor loading by {AE2.2.4.c2} accordingly to the configuration settings defined on {AE3.5.i} (monitoring tasks/users). This AE receives the request information, collects information from Control Devices, implemented in the loading site and sends the collected information to {AE4.2.3.c2} to detect abnormalities in the Control Devices. The AE output is the result of the monitoring execution (success or failure) and the state of loading task.

Architectural Element Name:	User's Consult Loading Decisions
Reference:	{AE3.2.2.d}
Description:	Makes decisions on how the measured information from {AE3.2.2.i} is used. The information can be used by the SLT/Analyst for preventing incorrect loads or to follow measured values through times. This variable information is defined in {AE3.5.i} (users/functionalities). This AE defines for who the values are available.

Architectural Element Name:	Receive Loading Information
Reference:	{AE3.2.2.i}
Description:	Receives the current values of loading execution (information concerning the weights of loads (e.g. net, gross and tare) and the position of loads (e.g. if the truck is well positioned) from {AE3.2.2.c2}.

Architectural Element Name:	Execute Monitoring Unloading
Reference:	{AE3.2.3.c2}
Description:	Process the requested monitor unloading by {AE2.2.5.c2} accordingly to the configuration settings defined on {AE3.5.i} (monitoring tasks/users). This AE receives the request information, collects information from Control Devices, implemented in the unloading site and sends the collected information to {AE4.2.3.c2} to detect abnormalities in the Control Devices. The AE output is the result of the monitoring execution (success or failure) and the state of unloading task.

Architectural Element Name:	User's Consult Unloading Decisions
Reference:	{AE3.2.3.d}
Description:	Makes decisions on how the measured information from {AE3.2.3.i} is used. The information can be used by the SLT/Analyst for preventing incorrect unloads or to follow measured values through times. This variable information is defined in {AE3.5.i} (users/functionalities). This AE defines for who the values are available.

Architectural Element Name:	Receive Unloading Information
Reference:	{AE3.2.3.i}
Description:	Receives the current values of unloading execution (information concerning the weights of unloads (e.g. net, gross and tare) and the position of unloads (e.g. if the truck is well positioned)) from {AE3.2.3.c2}.

Architectural Element Name:	Execute Monitoring Custom Tasks
Reference:	{AE3.2.4.c2}
Description:	Process the requested monitor custom tasks by {AE2.2.6.c2} accordingly to the configuration settings defined on {AE3.5.i} (monitoring tasks/users). This AE receives the request information, collects information from Control Devices and other SLT repositories can send the collected information to {AE4.2.3.c2} (to detect abnormalities) and can send to {AE4.2.4.c2} (to calculate or to detect an incident related to the custom tasks). The AE output is the result of the monitoring execution (success or failure) and the state of custom tasks.

Architectural Element Name:	Consult Custom Tasks Information
Reference:	{AE3.2.4.c3}
Description:	Consult and make decisions regarding the information of custom tasks (in {AE3.2.4.c2} and other SLT repositories) in order to build a report.

Architectural Element Name:	User's Consult Custom Tasks Decisions
Reference:	{AE3.2.4.d}
Description:	Makes decisions on how the measured information from {AE3.2.4.i} is used and on what information (current and historical custom task executions) the user (Analyst) needs or has preference to access, inside allowed limits (defined in {AE3.5.i}). The information can be used by the SLT/Analyst for preventing incorrect task execution or to follow measured values through times. This AE defines for who the values are available and the user (Analyst) can customize interfaces, layouts or languages and create your own queries.

Architectural Element Name:	Receive Custom Tasks Information
Reference:	{AE3.2.4.i}
Description:	Receives the current values of custom task execution (customized information defined in {AE3.5.i}) from {AE3.2.4.c2} and a report from {AE3.2.4.c3}.

Architectural Element Name:	Validate Alert Requests
Reference:	{AE4.1.c1}

Description:	Validate the requested list of alerts by users (from {AE4.3.i}), by Controller (from {AE5.2.i}) or by Detect Incidents activities (from {AE4.2.1.c2}, {AE4.2.2.c2}, {AE4.2.3.c2} and {AE4.2.4.c2}) accordingly to the configuration settings defined on {AE4.4.i} (incidents/users). For example, verify if the data of request is clean, correct and useful and if the user has permission to get the request satisfied. This AE outputs the state of the requested alerts (accepted, rejected, sent or read).

Architectural Element Name:	Send Alerts to Users
Reference:	{AE4.1.i}
Description:	Send Incident Information to User (Driver or Operator), External Entity and/or Controller.

Architectural Element Name:	Validate Detect Requests
Reference:	{AE4.2.1.c1}
Description:	This AE can takes as input the information provided by control activities ({AE2.2.1.c2}, {AE2.2.2.c2} and {AE2.2.3.c2}), monitoring activities (from {AE3.2.1.c2}, ..., {AE3.2.4.c2}), and {AE4.3.i}). This information is validated accordingly to the configuration settings defined on {AE4.4.i}. For example, verify if the data of request is clean, correct and useful and if the user has permission to get the request satisfied.

Architectural Element Name:	Node Availability Detection Decisions
Reference:	{AE4.2.1.c2}
Description:	Makes decisions on how node availability is detected and node availability is notified to user accordingly to the configuration settings defined on {AE4.4.i} (incidents/user roles). For example, an availability of a node can be determined by the operation order priority, the number of operation orders that a node can process at the moment and the scheduled operation orders. In the case of node availability, this AE can consequently call the next User (Driver or Operator) to begin an operation order or to proceed to the next task of an operation order execution. Before sending the detection information to {AE4.1.c1}, this AE can request information to {AE1.1.d}. This AE outputs a boolean response.

Architectural Element Name:	Weighing Insecurity Detection Decisions
Reference:	{AE4.2.2.c2}
Description:	Makes a decision on how weighing insecurity is detected and how weighing insecurity is notified to user accordingly to the configuration settings defined on {AE4.4.i} (incidents/user roles). For example, a weighing insecurity can be determined when the loads are incorrectly positioned or the weighing equipment is not calibrated. In the case of a weighing insecurity, this AE can create an alert to the User (Driver or Operator) of an operation order execution and/or to the Controller. Before sending the detection information to {AE4.1.c1}, this AE can request information to {AE1.1.d}. This AE outputs a boolean response.

Architectural Element Name:	Abnormalities Detection Decisions
Reference:	{AE4.2.3.c2}
Description:	Makes decisions on how abnormality is detected and abnormality is notified to user, both accordingly to the configuration settings defined on {AE4.4.i} (incidents/user roles). For example, an abnormality is an irregular situation that the SLT system considers. In the case of an abnormality, this AE can create an alert to the User (Driver or Operator) of an operation order execution and/or to the Controller. Before sending the detection information to {AE4.1.c1}, this AE can request information to {AE1.1.d}. This AE outputs a boolean response.

Architectural Element Name:	Custom Incidents Detection Decisions

Reference:	{AE4.2.4.c2}
Description:	Makes decisions on how custom Incident is detected and custom Incident is notified to user accordingly to the configuration settings defined on {AE4.4.i} (incidents/user roles). This activity encompasses a range of other customized incidents that can be created by Admin. In the case of a custom Incident, this AE can create an alert to the User (Driver or Operator) of an operation order execution and/or to the Controller and/or External Entity. Before sending the detection information to {AE4.1.c1}, this AE can request information to other SLT AEs. This AE outputs a boolean response.

Architectural Element Name:	Request Alerts Decisions
Reference:	{AE4.3.d}
Description:	Making decisions on what alerts should be created. These decisions are made according to the User's needs (Driver or Operator, Transporter, Customer or Supplier and Controller). This AE outputs information that is used in {AE4.3.i}.

Architectural Element Name:	Request Alerts
Reference:	{AE4.3.i}
Description:	The alerts are created (periodically or punctually) according to the decisions made in {AE4.3.d}. The result is a list of alerts (that includes a message, a list of destination users, the desired date, etc.) expresses the User's needs (Driver or Operator, Transporter, Customer or Supplier and Controller) and it is sent to {AE4.1.c1} for validation and to deliver the message. The result of the validation and the state of alerts and can be presented in this AE.

Architectural Element Name:	Validate Assistance Requests
Reference:	{AE5.1.c}
Description:	Validate the requested assistance by User (Driver or Operator) from {AE5.1.i} accordingly to the configuration settings defined on {AE5.4.i} (kind of assistances/users). For example, verify if the data of request is clean, correct and useful and if the user has permission to get the assistance request satisfied. In the case that the request is viable, this AE can send that request to Controller for approbation {AE5.2.i}. This AE outputs the state of the request (rejected, approved, processing or waiting for approbation).

Architectural Element Name:	Request Assistance Decisions
Reference:	{AE5.1.d}
Description:	Making decisions on what kind of assistance should be requested. These decisions are made according to the User's needs (Driver or Operator). This AE outputs information that is used in {AE5.1.i}.

Architectural Element Name:	Request Assistance
Reference:	{AE5.1.i}
Description:	The assistance is requested according to the decisions made in {AE5.1.d}. This request expresses the User's needs (Driver or Operator) and it is sent to {AE5.1.c} for validation and to proceed the request. The result of validation and the state of the assistance can be presented in this AE.

Architectural Element Name:	Validate Approbation Requests
Reference:	{AE5.2.c}
Description:	Validate the requested approbations by Controller from {AE5.2.i} accordingly to the configuration settings defined on {AE5.4.i} (kind of assistances/users). For example, verify if the data of request is clean, correct and useful and if the user has permission to get the assistance request satisfied. This AE outputs the state of the request (rejected or approved).

Architectural Element Name:	Approve Assistances Decisions
Reference:	{AE5.2.d}
Description:	Make decisions on what assistances should be processed and if is necessary to inform External Entity for a delay situation. These decisions are made according to the node's needs and users' needs {AE5.1.i}. This AE outputs information that is used in {AE5.2.i}.

Architectural Element Name:	Approve Assistances
Reference:	{AE5.2.i}
Description:	The User's assistance requests (from {AE5.1.c}) can be rejected and/or approved and an alert can be sent to External Entity ({AE4.1.c1}) according to the decisions made in {AE5.2.d}. The result is a list of assistances, it expresses the node's/Controller's needs and it is sent to {AE5.2.c} for validation. The result of the validation and Users' needs is presented in this AE.

Architectural Element Name:	User's Assistance Decision
Reference:	{AE5.3.1.d}
Description:	Technician analyzes User´s requirements of an assistance based on information given by Controller from {AE5.3.1.i} and {AE5.2.i}. Technician executes assistances to a user or SLT system.

Architectural Element Name:	Provide Assistance
Reference:	{AE5.3.1.i}
Description:	User (Driver or Operator) or SLT system receives the assistance provided by Technician. This AE outputs the activity of interaction between User or SLT system and Technician. This AE takes as input the information provided by {AE5.3.1.d} and {AE5.3.2.d}.

B.4.2 Collapsed Process Logical Architecture

A collapsed logical diagram was obtained from the logical architecture previously presented by hiding packages details. Therefore, associations are shown at a higher level of abstraction for readability purposes.

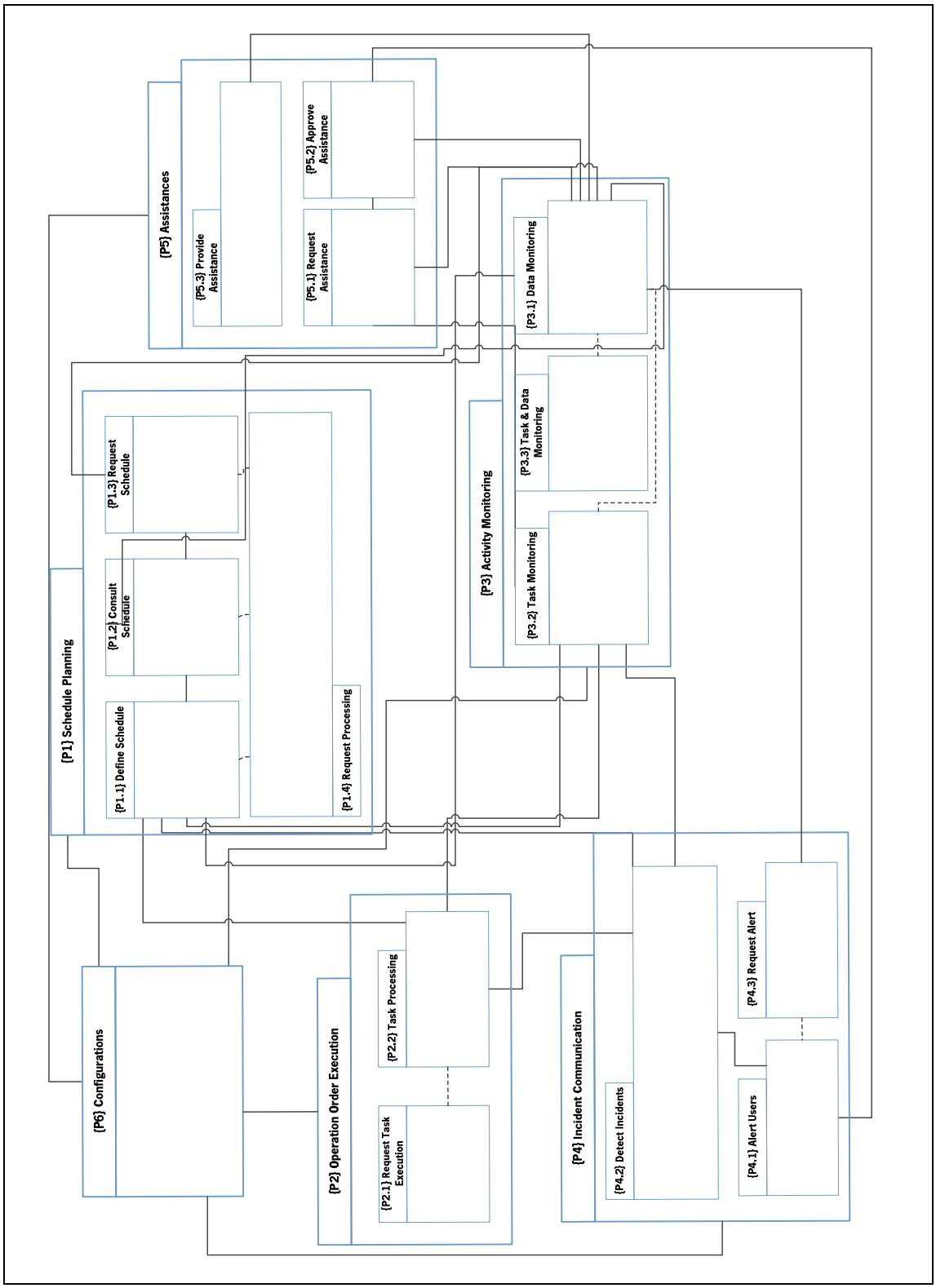


Figura A 23 – Collapsed Logical Architecture.

B.4.3 Collapsed Process Logical Architecture with Actors

Adding actors to the previously collapsed logical architecture provides details on the process participating roles. This information is useful for the process-level sequence diagrams and for the organizational configurations descriptions, presented in detail in following sections.

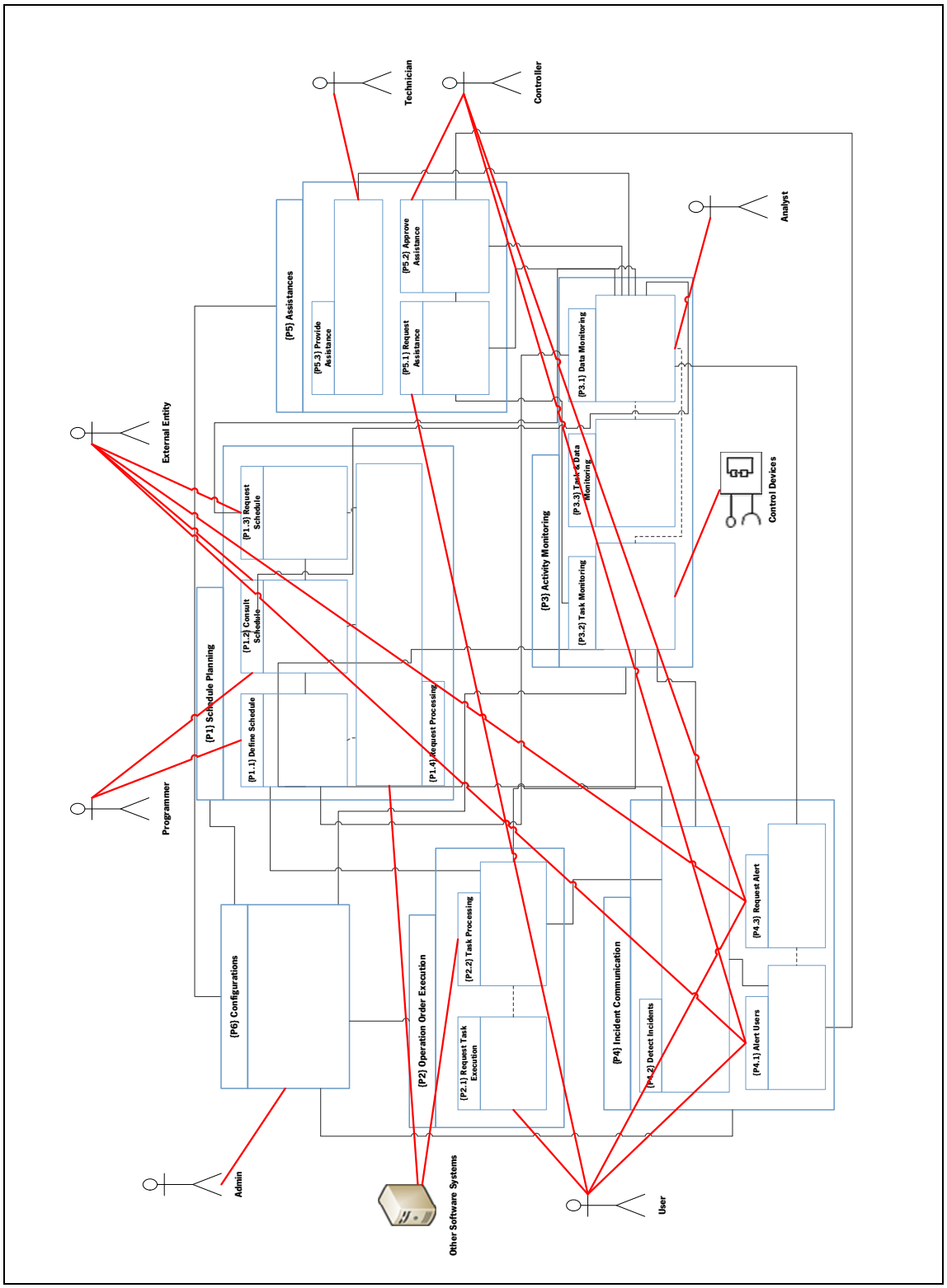


Figura A 24 – Collapsed Logical Architecture with Actors.

B.4 B-Type Sequence Diagrams

In order to assure the right definition of processes concerning the SLT process-level logical architecture, it is advisable the execution of some testing tasks. In this context, this section presents a set of B-Type Sequence Diagrams modeled, with the intent to validate if the behavior of the logical architecture is as expected. The sequence diagrams are modeled in a stereotyped way, by using AEs, actors and packages (if justifiable). The selection of the processes derives from the execution scenarios that were previously defined as critical for the business execution (section B.2).

B.2.1 B -Type Sequence Diagram #1: Operation Orders Planning

0a. Programmer defines what operation orders should be included in the node's schedule.
0b. Programmer configures the schedule of operation orders according to the node's needs and External Entity's needs.
0c and 0d. The system sends the schedule information to the control activities for validation. These control activities are concern about if the data of request is clean, correct and useful, if the user has permission to get the request satisfied and if it is viable to include the request in the node's schedule.
0e. The requested schedule is validated.
0f. Programmer receives the state of the request and the result of the validation.
1. External Entity defines what information he wants or has preference to access in a schedule consult.
2. External Entity requests a schedule for a specific period.
3 and 4. The information of request is sent to the control activities for validation and to process the request. These control activities are concern about if the data of request is clean, correct and useful and if the user has permission to get the request satisfied.
5. The system filters the information and reply.
6. External Entity receives the schedule information.
7. External Entity defines what operation orders should be included in the node's schedule.
8. External Entity requests a given day and a given hour to schedule operation orders according to the node's availability and External Entity's needs;
9 and 10. The information of request is sent to the control activities for validation and to process the request.
11. The information of request can be sent to the Programmer for validation.
12. Programmer reply to request.
13. The system validates the request.
14. External Entity receives the state of the request and the result of the validation.
15. External Entity can send an alert to the User to begin an operation order execution.
16. The system contacts with User.
17. User reply to the request.
18. The operation order execution is confirmed to the External Entity.

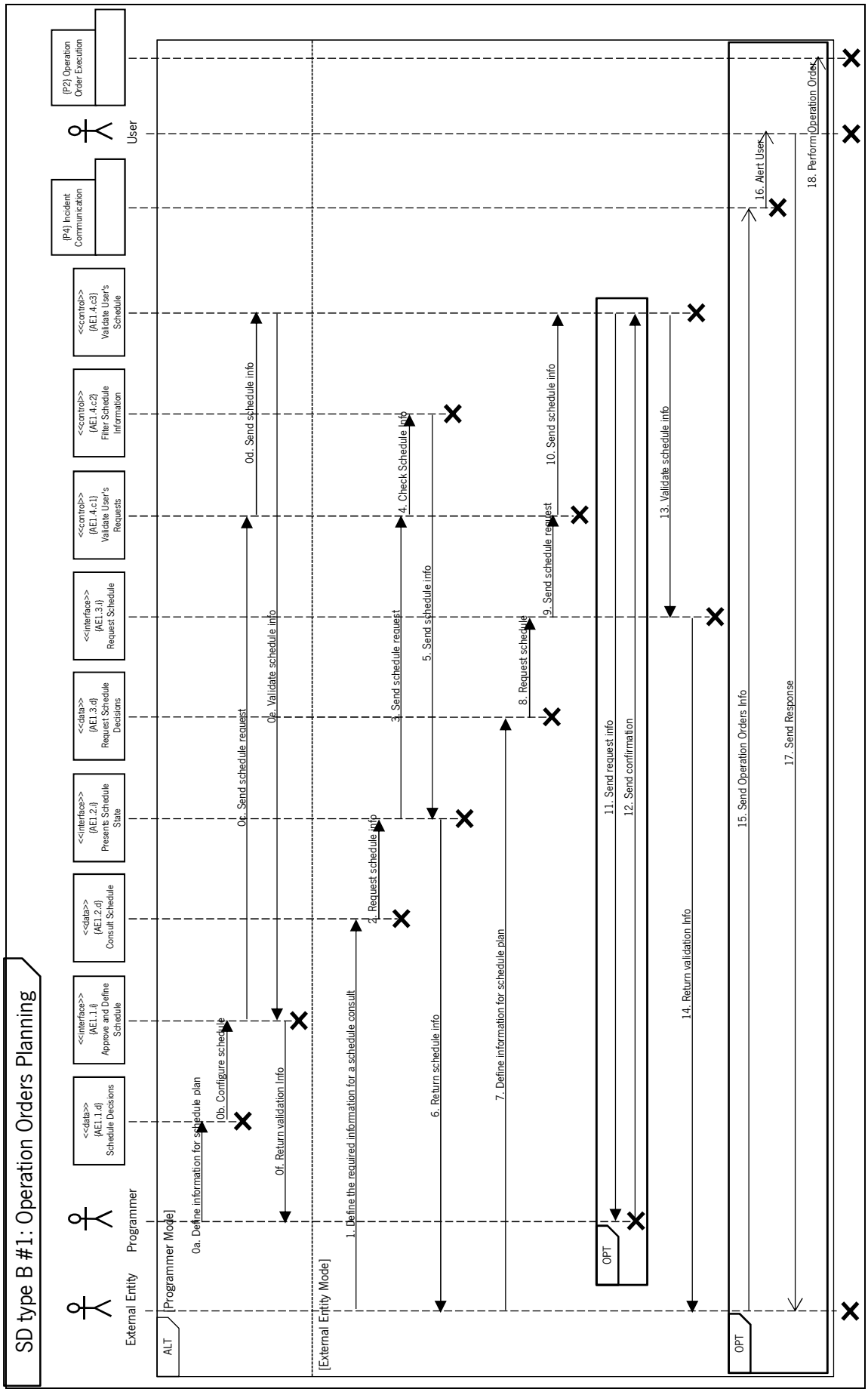


Figura A 25 – B -Type Sequence Diagram #1: Operation Orders Planning.

B.2.2 B -Type Sequence Diagram #2: Parking

1. User presents their identification for check-in.
2 and 3. The information of request is sent to the control activities for validation and to process the request. These control activities are concern about if the data of request is clean, correct and useful and if the user has permission to get the request satisfied.
4. The User identification information can be sent to the ERP for validation.
5. The information is validated.
6. The check-in information is reported to the system.
7. The requested check-in is validated.
8. User receives the result of check-in validation.
9 and 10. The check-in information is sent to the monitoring activities to detect the state of node availability. Normally the availability of a node is determined by the operation order priority, the number of operation orders that a node can process in the moment and the scheduled operation orders.
11. The node is monitored through control devices (e.g. posts, barriers and weighing equipment).
12. The control devices return the information.
13. The received information is checked for availability.
14. The system detects the state of node availability.
15. The monitoring activities receive the state of node availability.
If an availability information is detected:
16. An alert is sent to the system.
17. The User receives an availability alert.

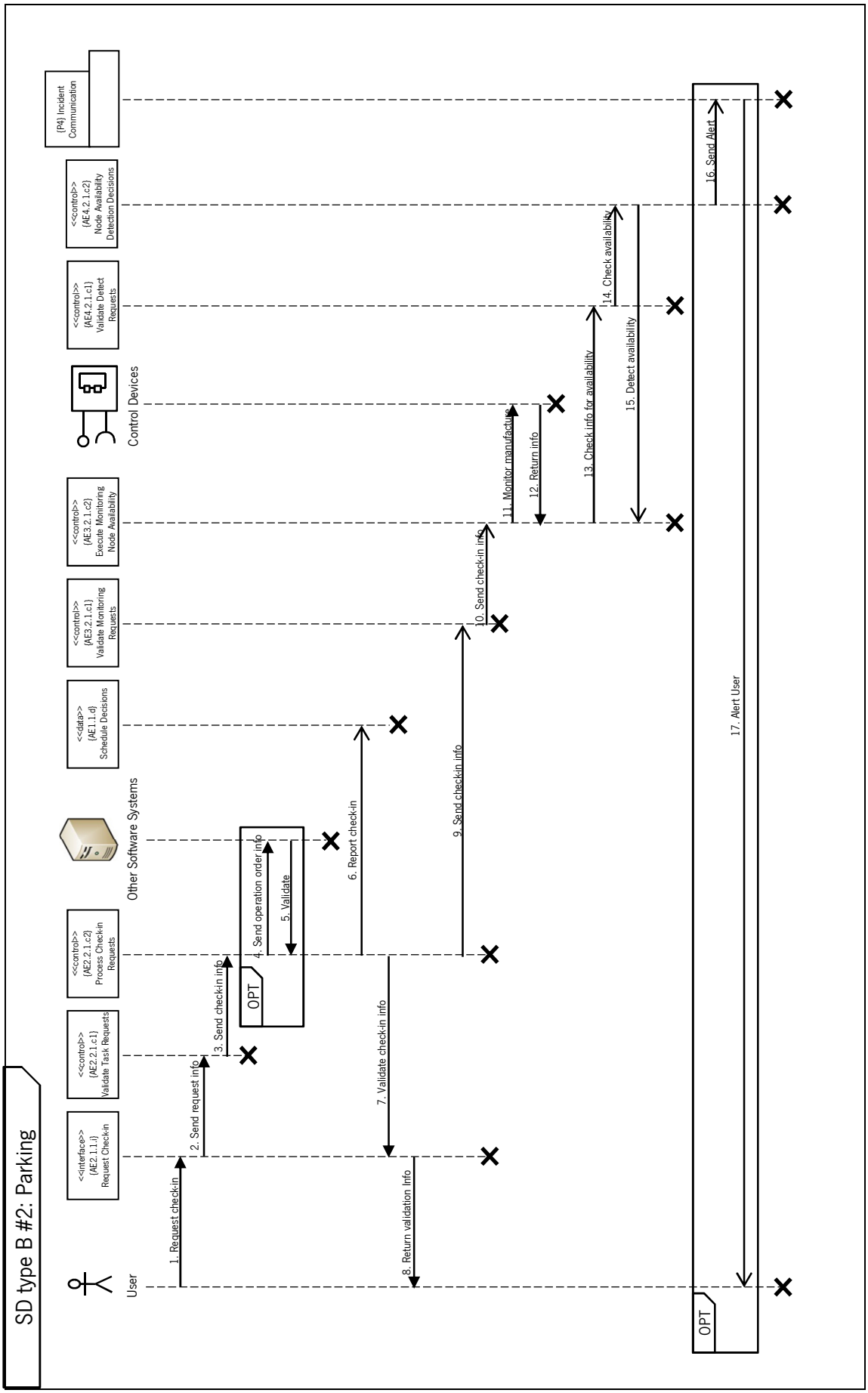


Figura A 26 – B -Type Sequence Diagram #2: Parking.

B.2.3 B -Type Sequence Diagram #3: Entrance/Exit

1. User requests an entrance/exit to realize an operation order.
2 and 3. The entrance/exit information is sent to the control activities. This information includes the user identification and also information related with the desired/realized operation orders.
4. The received information is checked for entrance/exit abnormalities. This information compared with the control settings determines the existence of an anomaly.
5. If an anomaly information is detected:
6. An alert is sent to the system.
7. In the case that the check-in is successfully validated hitherto, the entrance/exit information can be sent to the ERP for validation.
8. The information is validated.
9. The entrance/exit information is reported to the system.
10. The response of the validation is formulated.
11. User receives the result of entrance/exit validation.

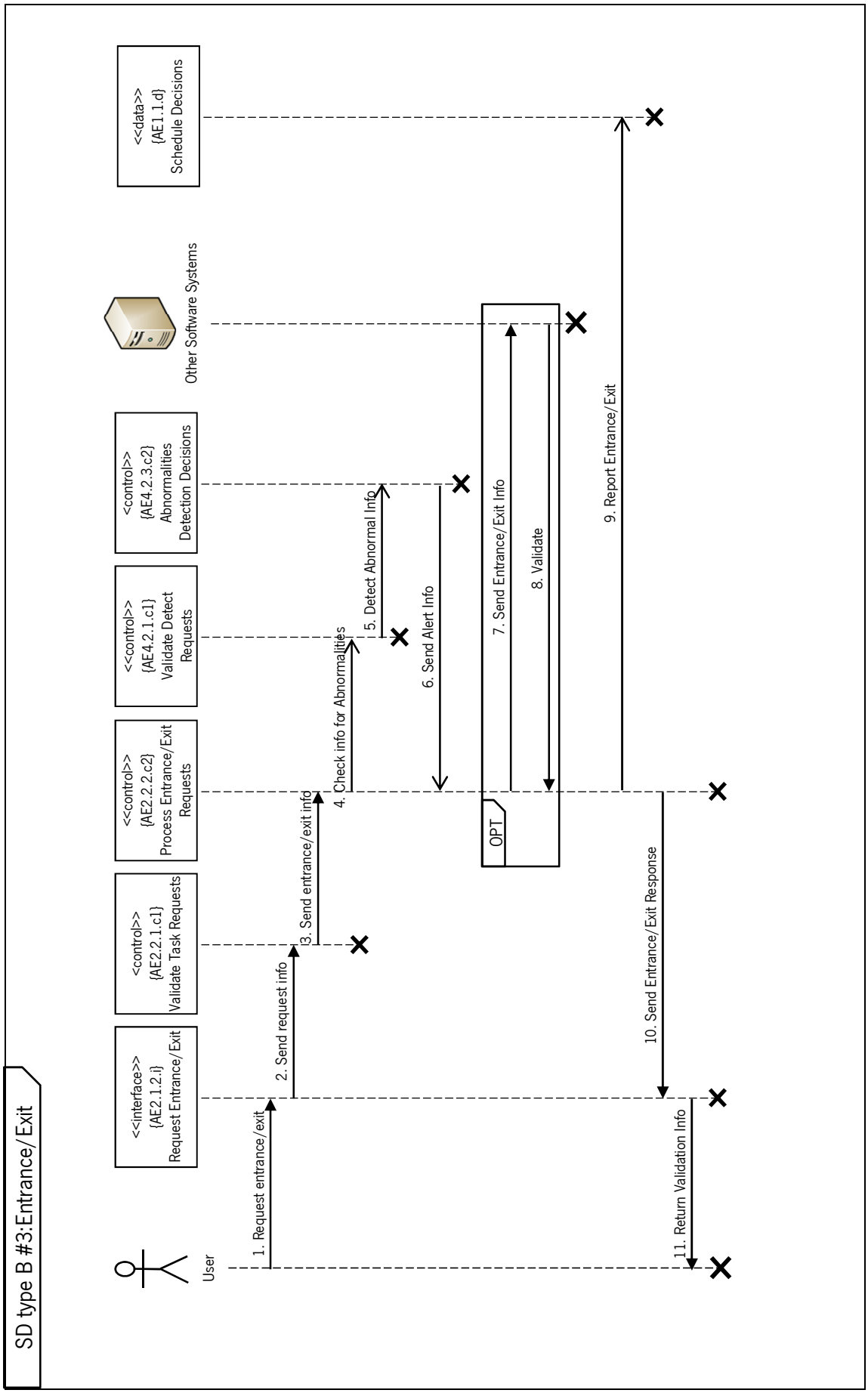


Figura A 27 – B -Type Sequence Diagram #3: Entrance/Exit.

B.2.4 B -Type Sequence Diagram #4: Weighing

1. User requests to weighing before/after an operation;
2 and 3. The weighing information is sent to the control activities. This information includes the way how was realized the weighing and its results.
4 and 7. The received information is checked for detect weighing insecurities and abnormalities. One part of this information, about the way how was realized the weighing (e.g. incorrect truck position and lack of scales calibration), confronted with the control settings, related with the devices behavior, determines the existence of an insecurity. The remaining part of information, about the weighing results (e.g. gross, net and tare weight), compared with the control settings, related with the conditions of weighing, determines the existence of an anomaly.
5. If an insecurity information is detected:
6. An alert is sent to the system.
8. If an anomaly information is detected:
9. An alert is sent to the system.
10. In the case of none detected insecurities and anomalies, the weighing information can be sent to the ERP for validation.
11. The information is validated.
12. The weighing information is reported to the system.
13. The response of the validation is formulated.
14. User receives the result of weighing validation and the information about the place where the operation order can be realized.

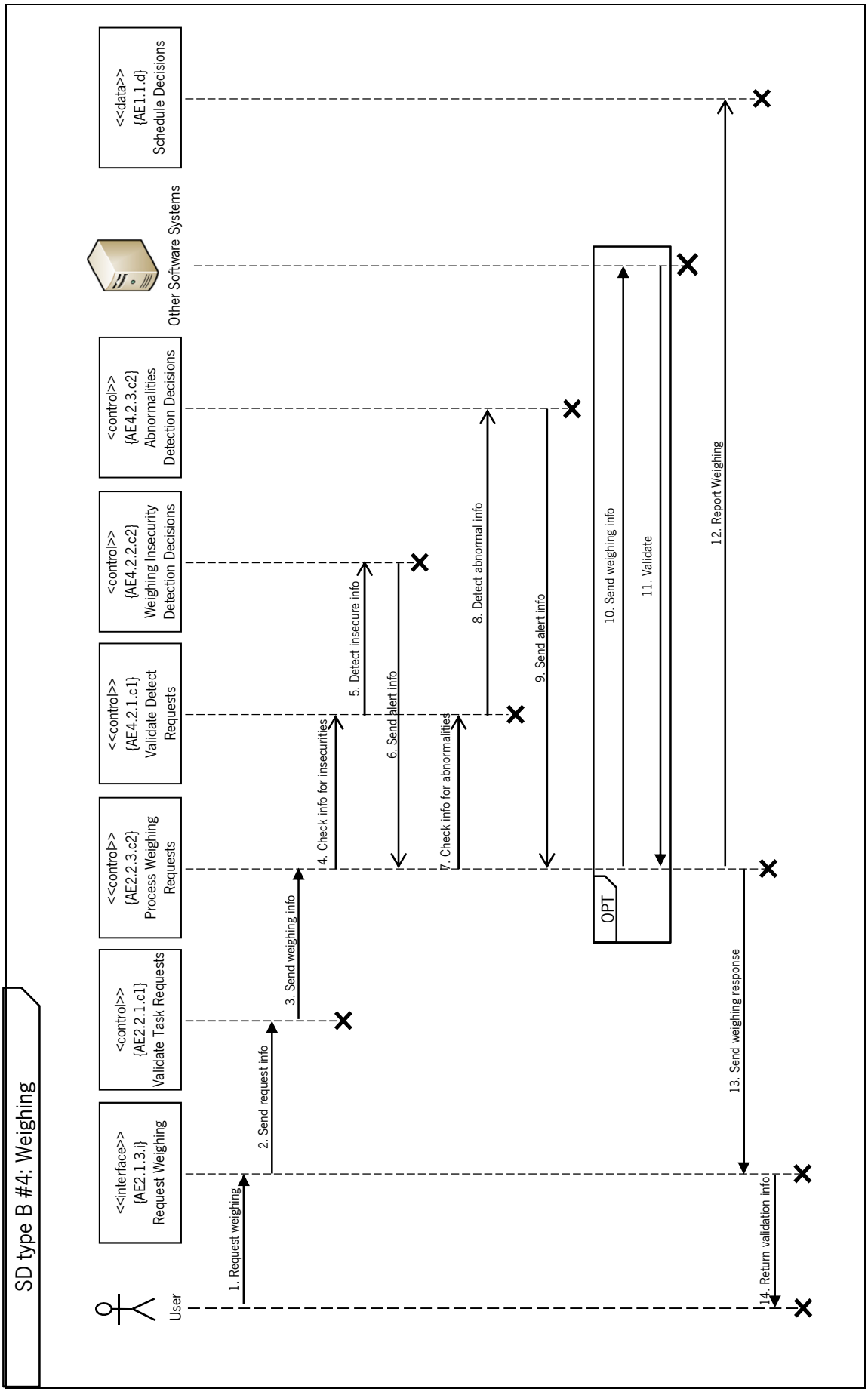


Figura A 28 – B -Type Sequence Diagram #4: Weighing.

B.2.5 B -Type Sequence Diagram #5: Loading

1. User presents their identification to perform a loading task.
2 and 3. The loading information is sent to the control activities. This information includes the user identification and the desired loads.
4. The loading request can be sent to other systems (e.g. ERP) for validation.
5. The request is validated.
6. The loading request is sent to the monitoring activities. This information includes the quantity of the products requested. The control activities ensures that the loading is done safely, complying monitoring settings.
7 and 8. User receives the result of the request validation.
9 and 10. User requests the beginning of the loading.
11 and 12. One alert to start the loading can be sent to the system, so that can be possible control the loading execution.
13. The weighing is monitored through monitoring devices (e.g. weighing equipment, sensors and silos).
14. The monitoring devices (e.g. weighing equipment, sensors and silos) return the information about the current loading values.
15. The received information is checked to detect abnormalities. One abnormality is determined based on the monitoring settings to control, for example, the pipe position, the products that can't be mixed and the overloading.
16. If an abnormality is detected:
17. An alert is sent to the system.
18. The result of the loading execution is formulated.
19. The loading information is reported to the system.
20 and 21. The loading task is finished and the User receives the result of the loading execution.

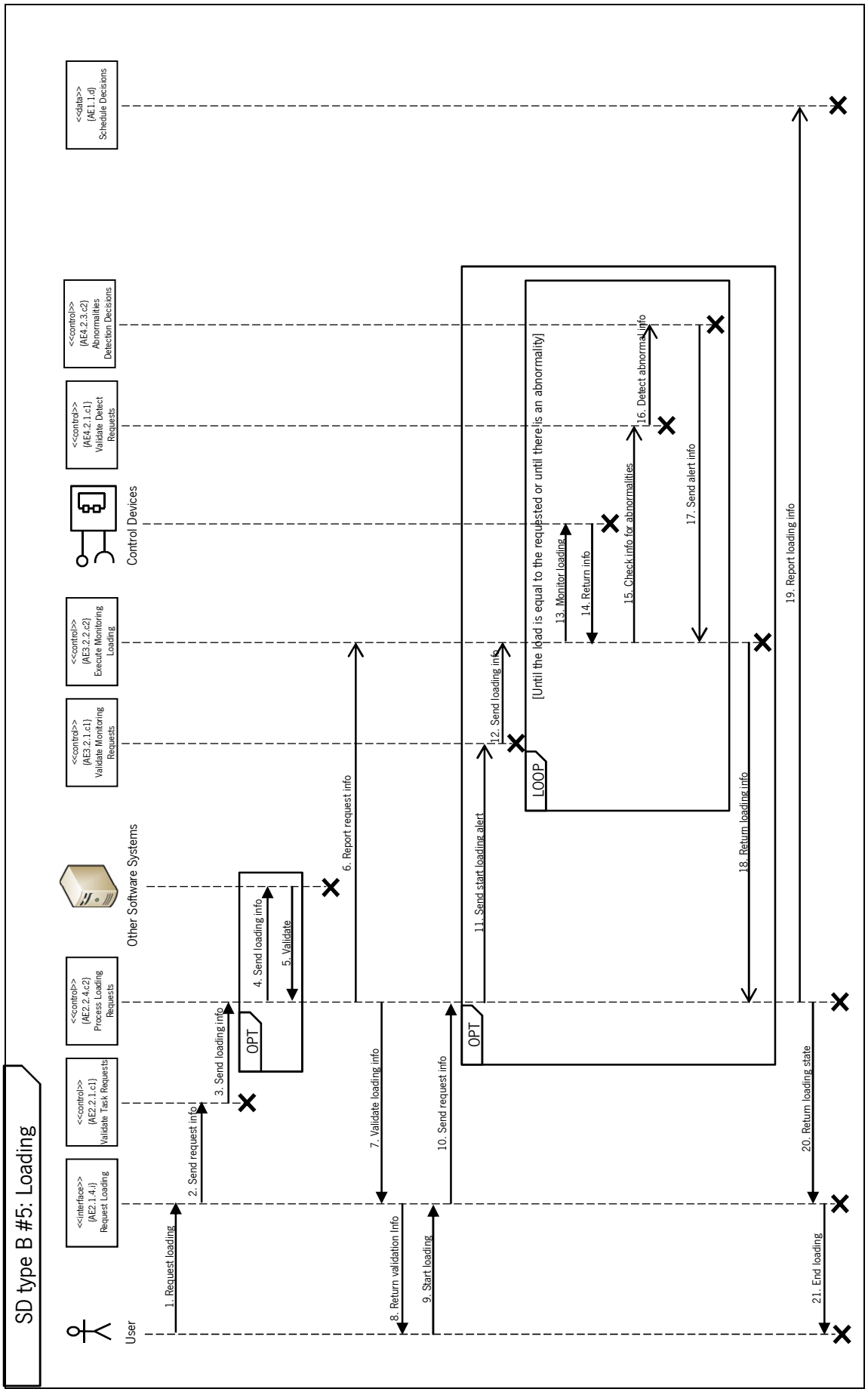


Figura A 29 – B -Type Sequence Diagram #5: Loading.

B.2.6 B -Type Sequence Diagram #6: Unloading

1. User presents their identification to perform an unloading task.
2 and 3. The unloading information is sent to the control activities. This information includes the user identification and the desired unloads.
4. The unloading request can be sent to other systems (e.g. ERP) for validation.
5. The request is validated.
6. The unloading request is sent to the monitoring activities. This information includes the quantity of the products requested. The control activities ensures that the unloading is done safely, complying monitoring settings.
7 and 8. User receives the result of the request validation.
9 and 10. User requests the beginning of the unloading.
11 and 12. One alert to start the unloading can be sent to the system, so that can be possible control the loading execution.
13. The weighing is monitored through monitoring devices (e.g. weighing equipment, sensors and silos).
14. The monitoring devices (e.g. weighing equipment, sensors and silos) return the information about the current unloading values.
15. The received information is checked to detect abnormalities. One abnormality is determined based on the monitoring settings to control, for example, the pipe position, the products that can't be mixed and the overloading.
16. If an abnormality is detected:
17. An alert is sent to the system.
18. The result of the unloading execution is formulated.
19. The unloading information is reported to the system.
20 and 21. The unloading task is finished and the User receives the result of the unloading execution.

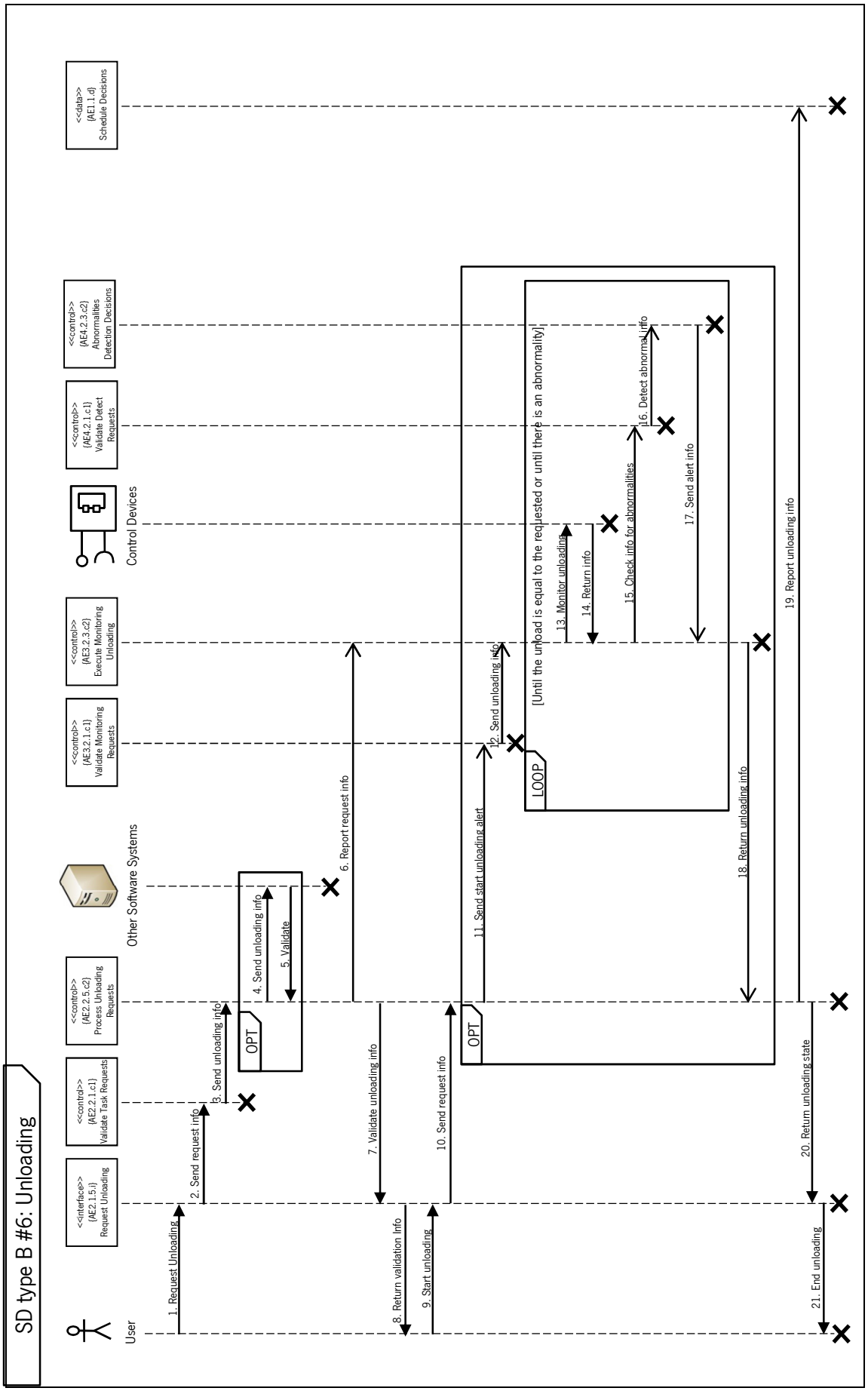


Figura A 30 – B -Type Sequence Diagram #6: Unloading.

B.2.7 B -Type Sequence Diagram #7: Repair/Maintenance Assistance

1. User's defines his assistance requirements.
2. Send User's requirements to provide assistance.
3. Requests assistance.
4 and 5. The system sends the assistance request information to Controller.
6. Controller receives an alert of assistance request.
7. Controller defines what assistances should be processed.
8. The system sends Controller's decisions to approve assistances.
9. The assistances are approved.
10. The approved assistances are validated.
11. Controller contacts with User and confirms assistance.
12 and 13. Controller receives an incident notification for a repair/maintenance assistance.
15. Controller gets in touch with the Technician.
16. Technician reply to request.
17. Technician defines what is needed to solve the assistance problems
18. Technician provides assistance. This AE outputs the activity of interaction between User or SLT system and Technician.
19. In the case where the estimated time for the assistance execution is high, the Controller can send an alert to all External Entities that may be affected.
18 and 20. External Entities receives an alert.

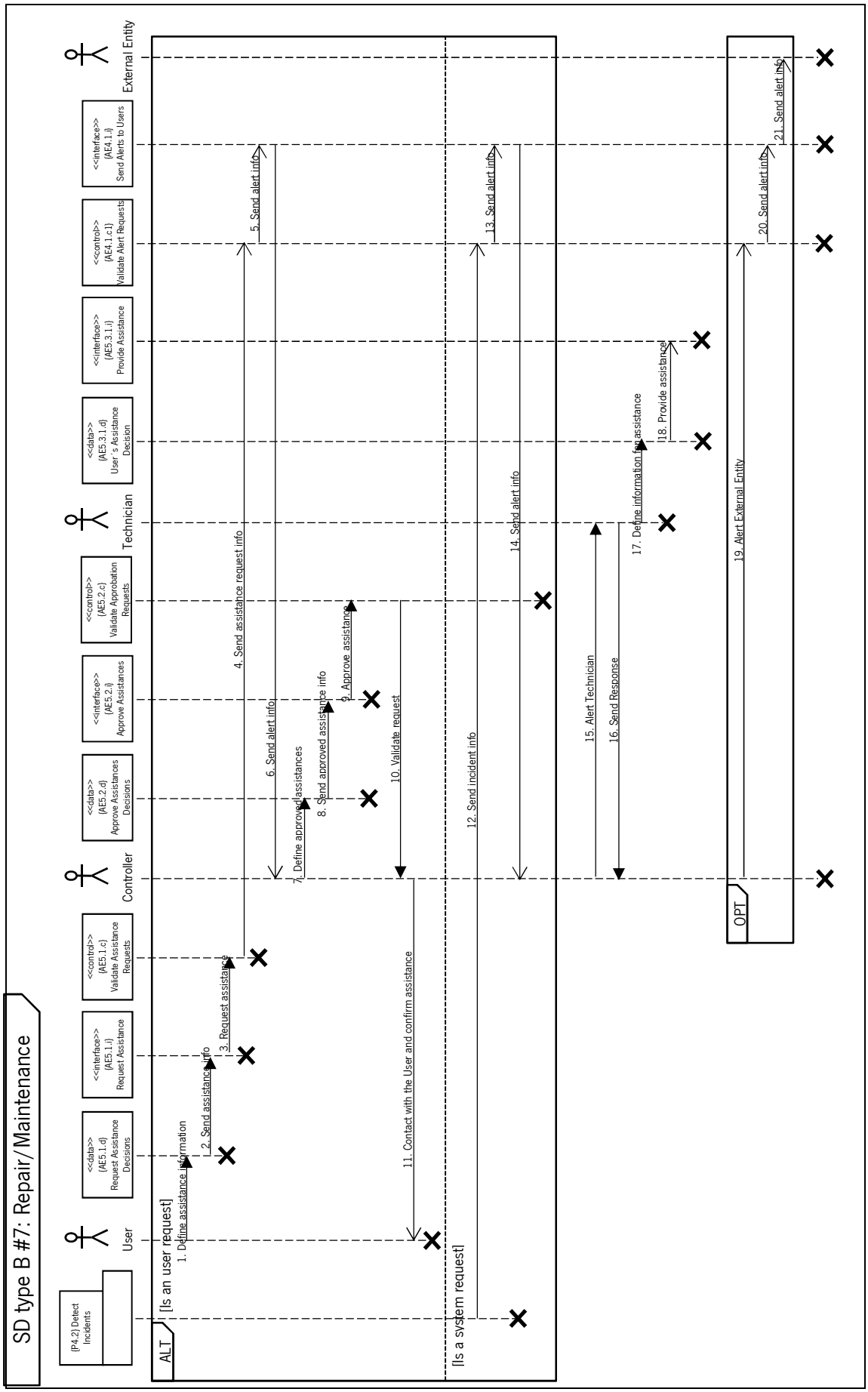


Figura A 31 – B -Type Sequence Diagram #7: Repair/Maintenance Assistance.