TÉCNICO LISBOA

PATETA: PATterns for EnTerprise reference Architectures

Yesika Paula Pereira Reinolds

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Professor André Ferreira Ferrão Couto e Vasconcelos

Examination Committee

Chairperson: Professor Miguel Nuno Dias Alves Pupo Correia
Supervisor: Professor André Ferreira Ferrão Couto e Vasconcelos
Member of the Committee: Professor Pedro Manuel Moreira Vaz Antunes de Sousa

May 2015

”Não existe vento favorável à quem não sabe onde deseja ir”

Arthur Schopenhauer, filósofo alemão do século XIX

Agradecimentos

No final deste percurso académico quero gradecer algumas pessoas que ajudaram de uma forma directa e indirecta a conclusão deste trabalho. Assim sendo gostaria de agradecer,

- Em primeiro lugar quero agradecer a Deus por me ajudar a terminar este percurso da minha vida, por ter colocado pessoas fantásticas nela que me ajudaram a crescer e a lutar.
- Ao Professor André Vasconcelos pela orientação e pela ajuda na solução de problemas que ocorreram na execução do trabalho, assim como a disponibilidade para as reuniões e paciência.
- À Dra. Maria João Santos, da AMA pela disponibilidade ao facultarem dados para a execução do trabalho.
- Quero agradecer ao meu namorado Pedro Castro, pelo seu carinho, por ter estado sempre ao meu lado, por cuidar de mim, por fazer com que a minha persistência não terminasse, pelo companheirismo e pelo apoio incondicional.
- Quero agradecer a minha mãe Alice Ferreira, pelo optimismo nos dias desanimadores, pelas palavras motivadoras, por me fazer ver que nem tudo se consegue a primeira sendo o mais importante nunca desistir e principalmente por confiar em nós mesmos. Ao meu pai João Pereira, quero agradecer pela sua compreensão, por estar ao meu lado quando mais preciso, por me fazer esquecer os momentos menos bons. Ao meu irmão João Reinolds, quero agradecer simplesmente por ser meu irmão, principalmente pelo carinho.
- A minha Tia Graziela Reinolds e a avó Alice Gomes quero agradecer pelo amor que me transmitem a distância, e em especial a minha tia Graciela quero agradecer pelo exemplo universitário e de persistência que sempre me transmitiu.
- Também quero agradecer ao Manuel, Marcelina e Ana Castro por se preocuparem por mim e por me motivarem a terminar a tese.
- A uma pessoa especial Fátima Gonçalves que sempre me acompanhou durante a minha vida e que me dá total suporte.
- Finalmente aos meus amigos e colegas Sandra Elaine,Filipe Azevedo, António Moreira, Filipe Silva, Nídia Spranger, Nuno Nunes, João Abreu e Eduardo Fermé com quem muito aprendi, pela amizade e pelo apoio que me deram.

Resumo

O objectivo desta investigação consiste em descobrir padrões e anti-padrões para as arquitecturas empresariais (AE) com intuito de ajudar no desenvolvimento de projectos. Esta investigação foi endereçada para alguns problemas práticos: a não existência de padrões que cubram todas as arquitecturas pertencentes a AE, a não existência de um catálogo com os padrões e anti-padrões e a não existência de um método para a descoberta de padrões em arquitecturas.

Foi desenvolvido o método RCGD (Revision, Comparison, Generation and Documentation) para ajudar a descobrir os padrões da AE. As fases que constituem esta solução são as seguintes: análise das arquitecturas, comparar arquitecturas, geração e documentação.

A fase de análise das arquitecturas consiste em escolher duas arquitecturas para a descoberta de padrões. A fase de comparar as arquitecturas baseia-se em 3 sub-fases. Na sub-fase comparação *out of the box* é realizada uma análise lexical e detectada as relações entre objectos. Na sub-fase padrões *out of the box* dá-se a classificação do padrão. Na sub-fase comparação *inside the box* dá-se à análise as funcionalidades dos objectos sendo analisado a similaridade funcional. A fase de geração corresponde a criação de um padrão arquitectural. A fase da documentação é uma fase necessária para a transmissão de conhecimento onde os padrões são documentados.

Foi aplicado esta solução a duas arquitecturas de uma universidade. Estas arquitecturas tinham como objectivo actualizar o sistema utilizando e armazenar os dados dos actores destas arquitecturas. As arquitecturas foram desenhadas em XML para realizar à análise de forma automatizada.

Palavras-chave: Aquitectura Empresarial, Padrão, Anti-Padrões, detectar padrões, Design Science Research

Abstract

The goal of this investigation is to find patterns and anti-patterns for Enterprise Architectures (EA) in order to help developing projects. In this paper we address several scientific and practical problems, including: the nonexistence of patterns covering all architectures bellowing to EA, the nonexistence of a catalog with patterns and anti-patterns and the nonexistence of methods to discover patterns in architectures.

We developed a method named RCGD (Revision, Comparison, Generation and Documentation) which can help you finding patterns in many EA. With this method, we can obtain patterns covering all EA, create a catalog with patterns and anti-patterns and also define a method for patterns' discovery.

The RCGD method to discover patterns has four phases: revision, comparison, generation and documentation. In the Revision phase we choose the architectures and perform the analysis. The comparison phase involves a set of steps to detect the candidates' patterns using the similarity's results. To calculate the similarity between two objects we divide it into two groups: out-of-the-box and inside-the-box. In the out-of-the-box we analyze external things like layer type and name. In the inside–the-box we analyze the functionalities. For the Generation phase we generate the architectural pattern. Finally, in the documentation phase we document the pattern using a previously defined structure.

We apply our approach in two university's architectures. The architecture's goal is to update their system and to record all data about their actors. To do the analysis we design these architectures in a XML file, in order to automate the analysis.

Keywords: Enterprise Architectures, Patterns, Anti-patterns, Finding Patterns, Design Science Research

Conteúdo

Agradecimentos v
Resumo vii
Abstract ix
Lista de Figuras xiv

1 Introdução **1**
1.1 Enquadramento 1
1.2 Objectivos 2
1.3 Definição do Problema 3
1.4 Metodologia e Avaliação do Trabalho 3
1.5 Organização do documento 5

2 Estado da Arte **6**
2.1 Arquitectura 6
2.1.1 Principios Arquitecturais 7
2.1.2 Arquitectura Empresarial 7
2.1.3 Modelação Arquitectura Empresarial 8
2.1.4 Arquitectura de Referência 10
2.2 Padrão 12
2.2.1 Estrutura do Padrão de Martin Fowler 13
2.2.2 Estrutura do Padrão de Eric Gamma, Richard Helm, Ralph Johnson e John Vlissides 14
2.3 Resumo 14

3 Contributos **16**

4 Proposta de Solução **19**
4.1 Análise das Arquitecturas - 1ª Fase 19
4.2 Comparar Arquitecturas - 2ª Fase 20
4.2.1 Comparação *out of the box* 20
4.2.2 Padrões *out of the box* 23
4.2.3 Comparação *inside the box* 24
4.3 Geração - 3ª Fase 27

4.4 Documentação - 4ª Fase 29
4.5 Resumo 30

5 Demonstração **32**
5.1 Análise das Arquitecturas - 1ª Fase 33
5.2 Comparar Arquitecturas - 2ª Fase 34
5.2.1 Comparação *out of the box* 34
5.2.2 Padrões *out of the box* 45
5.2.3 Comparação *inside the box* 46
5.3 Geração - 3ª Fase 51
5.4 Documentação - 4ª Fase 52
5.5 Resumo 54

6 Avalição dos Resultados **55**
6.1 Avaliação Quantitativa 55
6.2 Conclusão 56

7 Conclusão **58**

A Arquitectura ArchiMate **64**
A.1 ArchiMate 64

B Arquitecturas de testes **65**
B.1 Arquitectura 1 65
B.2 Arquitectura 2 66

C Estruturas XML **67**
C.1 Estrutura XML guardar dados similares 67

D Dados em XML usados na aplicação **68**
D.1 Preposições e Conjunções 68
D.2 Sinónimos 69
D.3 Antónimos 69

Lista de Figuras

2.1 *Framework* Zachman 9
2.2 Fluxo de conceitos da AR 12

4.1 Fases da RCGD 19
4.2 Etapas da fase de comparação de arquitecturas 20
4.3 Fluxo para a descoberta da relação entre dois objectos 22
4.4 Classificação do padrão/anti-padrão 24
4.5 Matching das funcionalidade entre os objectos dos padrões candidatos 25
4.6 Etapas da fase de comparação de arquitecturas 25
4.7 Actualização da AR 27
4.8 Caracterização do padrão encontrado 28
4.9 Fluxo classificação do padrão 29

5.1 Arquitectura RCGD 32
5.2 Diagrama de classes da estrutura da informação guardada 35
5.3 Fluxo para a descoberta da relação entre dois objectos 37
5.4 Combinações na comparação de dois objectos 37
5.5 Análise realizada durante a comparação de dois objectos 38
5.6 Diagrama de sequência 41
5.7 Fluxo de detecção de objectos similares 41
5.8 Resultado da similaridade dos objectos 43
5.9 Lista de Padrões Candidatos 46
5.10 Classificação dos padrões *out of the box* 46
5.11 Análise as funcionalidades dos objectos similares 47
5.12 Padrão Arquitectural 1 48
5.13 Padrão Arquitectural 2 49
5.14 Padrão Arquitectural 3 49
5.15 Arquitectura de referência 50
5.16 Padrão Candidato 1 53
5.17 Padrão Candidato 2 53
5.18 Padrão Candidato 3 54

6.1 Teste realizado com os utilizadores 56
6.2 Resultado dos testes realizado com os utilizadores 56

Capítulo 1

Introdução

1.1 Enquadramento

A experiência dos peritos faz com que estes encontrem soluções para muitos problemas recorrentes e desta forma são criados os padrões de desenho. Estas soluções permitem que juniores resolvam problemas como se tivessem anos de experiência ao reutilizar soluções para problemas similares [1]. Apesar da importância dos padrões não existe nenhum processo que permita a sua descoberta, pois as especificações dos padrões são realizada por peritos.

A Arquitectura Empresarial (AE) é definida como sendo "um conjunto coerente de princípios, métodos e modelos que são utilizados na concepção e realização de uma estrutura organizacional, processos de negócio, sistemas de informação e infra-estrutura da empresa" [2]. A AE é constituída por sub-arquitecturas: Arquitectura Organizacional, Arquitectura de Negócio e Arquitectura de Sistemas de Informação. Contudo não existem padrões definidos que cubram todas as arquitecturas que constituem uma AE, sendo uma boa parte dos padrões endereçados a arquitecturas de *software* e tecnológica, como é possível verificar em [3].

Existe uma necessidade de definir padrões para qualquer tipo de arquitectura de forma a fazer uso do conhecimento e experiência adquirida por arquitectos ao longo de projectos de forma a ganhar tempo, existir menos esforço no desenvolvimento de arquitecturas e obter resultados mais positivos em projectos futuros [3] [4].

Um padrão apenas estrutura e organiza aspectos de um sistema que são que constantemente são repetidos em várias arquitecturas inerentes a um domínio específico que determinando uma solução específica. Assim sendo um padrão tem que ser abstracto para que o conhecimento possa ser reutilizado em sistemas diferentes. É importante ter em atenção que um padrão é apenas uma espécie de *framework* [5], no sentido que reúne um conjunto de conceitos e relações para resolver problemas num determinado domínio não definindo por inteiro a arquitectura de um sistema, sendo necessário refinar ou ate mesmo utilizar outros padrões até atingir a solução desejada [3].

Um processo para a descoberta de padrões na AE ajuda a ter um método coerente, válido, fiável e metódico para a detecção de padrões e de anti-padrões, sendo que os padrões e anti-padrões desco-

bertos ajudarão a que empresa tenha uma maior eficiência no desenvolvimento de projectos futuros em tempo útil. Devido ao grau de conhecimento adquirido e documentado com detalhes essenciais para a sua utilização. Detalhes como por exemplo, consequências negativas e positivas.

É de referir que existem muitos padrões e anti-padrões por descobrir porque há tipos de arquitecturas que até hoje não foram abrangidas pelos padrões *standards* descobertos por peritos. Apesar de cada sistema ser único é normal que sistemas que sejam de áreas com um domínio semelhante tenham arquitecturas semelhantes, daí existir a possibilidade de determinar padrões e anti-padrões. É necessário reflectir que detectar padrões arquitecturais não é fácil e existem muitas condicionantes, como a aceitação destes na empresa.

1.2 Objectivos

Existem empresas que se inserem na mesma indústria existindo a necessidade de criar padrões, para que as boas práticas sejam transmitidas. O objectivo desta dissertação **é descobrir padrões e anti-padrões para AE de forma a auxiliar o desenvolvimento de projectos para uma específica indústria.** Para tal é necessário **traçar um método para a descoberta de padrões que sirva para a sua detecção e consequentemente a detecção de anti-padrões.** Estes padrões são detectados através da análise de AE que empresa desenvolveu e desenvolve podendo **gerar um catálogo de padrões e anti-padrões** direccionados a uma determinada industria para dar suporte a construção de AE.

Os benefícios de ter padrões são o ganho de tempo no desenho arquitectural, diminuição das consequências negativas não esperadas, ter noção das vantagens e desvantagens na utilização desse padrão e ter informação adquirida pela prática de um outro arquitecto. Apesar destes benefícios não é fácil detectar padrões. Os benefícios de descobrir padrões e anti-padrões arquitecturais para uma indústria específica é que para além de ter padrões definidos para a área de negócio onde a empresa opera, aplicar esses padrões torna-se mais fácil devido à familiaridade dos arquitectos, engenheiros e técnicos com o negócio sendo à sua aplicação muito mais rápida e directa.

Uma outra razão para descobrir padrões para a AE é que os padrões que existem e que são *standards*, como o padrão cliente-servidor [3], não abrangem todas as sub-arquitecturas que constituem uma AE. Desta forma a necessidade de criar padrões que abranjam estas sub-arquitecturas de forma a usufruir das vantagens de usar padrões é relevante, sendo primeiramente necessário traçar um método para a descoberta destes padrões.

Definido quais são os objectivos desta dissertação e quais são as necessidades existentes é necessário realizar algumas questões:

- O que é considerado padrão e anti-padrão numa AE de uma empresa?
- Qual é a barreira que separa o padrão do anti-padrão?
- Quais são os passos para a descoberta de padrões?

- Existem catálogos nas empresas com padrões e anti-padrões?

Estas questões têm relevância pelas seguintes razões: *O que é considerado padrão e anti-padrão numa AE de uma empresa?* É importante definir quais são os aspectos que devem ser analisados numa AE e as métricas que classificam os artefactos encontrados como padrão ou anti-padrão. *Qual é a barreira que separam o padrão do anti-padrão?* Após descobrir os padrões é necessário classifica-los como padrão ou anti-padrão, para tal é necessário definir a barreira que separa um padrão de um anti-padrão. *Quais são os passos para a descoberta de padrões?* O processo para a descoberta de padrões tem que ser metódico para que todos os aspectos na análise sejam processados e cobertos. *Existem catálogos nas empresas que tenham padrões e anti-padrões da empresa?* Esta questão é relevante para indicar a informação que é documentada pela empresa ao longo do seu ciclo de vida de forma a ser reutilizada.

1.3 Definição do Problema

Quando temos uma AE onde se pretende aplicar padrões é importante para o desenho da AE a existência de padrões que cubram todas as arquitecturas. É igualmente necessário verificar se os padrões e anti-padrões encontram-se documentados, para que exista informação suficiente para a sua aplicação.

Para a aplicação de padrões numa AE existe um problema que responde à questão *"Existem padrões que cubram todas as arquitecturas de uma AE?"* esse problema diz respeito à **inexistência de padrões que cubram todas as arquitecturas que constituem uma AE**, isto é, não existem padrões para todas estas arquitecturas: organizacional, de negócio, informacional, aplicacional e tecnológica. Daí existir a necessidade de detectar padrões de forma a ser aplicado as diferentes arquitecturas que constituem a AE.

Na questão *"Existem catálogos nas empresas de padrões e anti-padrões?"* há um problema cultural **da não existência de catálogos com padrões e anti-padrões para uma AE**, direccionados à indústria da empresa para que a informação adquirida pelas empresas numa determinada indústria durante o seu ciclo de vida possa *à posteriori* ser utilizada no desenvolvimento de projectos.

Existe um problema na detecção de padrões e anti-padrões que corresponde à **inexistência de um processo para a sua descoberta**, de forma a facilitar a sua detecção de forma coerente, eficiente e metódica. Este problema resulta da resposta à seguinte questão: *"Que passos são necessários realizar para detectar padrões e anti-padrões?"*.

1.4 Metodologia e Avaliação do Trabalho

O objectivo desde projecto é descobrir padrões e anti-padroes para AE, para tal é necessário desenvolver um método para a sua descoberta. Através da experiência e da investigação pretende-se desenvolver dois tipos de artefactos: um processo para a descoberta de padrões e anti-padrões para uma AE. Três acções essenciais na realização destes artefactos são:

- Desenvolver uma solução para a descoberta de padrões de forma meticulosa.
- Avaliar a metodologia apresentada e posteriormente os padrões descobertos.
- Demostrar de que forma é vantajosa a solução proposta.

Na dissertação a abordagem a ser aplicada é *Design Science Research* (DSR). A metodologia DSR tem por base a produção de artefactos. O termo *Design Science* significa projectar um novo artefacto que não existe na natureza. O termo Research advém do uso de conhecimento para originar um artefacto de forma sistemática e rigorosa [6].

Esta abordagem é aplicada na geração de conhecimento, neste caso com objectivo de criar um processo para a descoberta de padrões a partir de arquitecturas existentes. Esta investigação contribuirá para descoberta de padrões de AE sendo necessário ter conhecimento sobre os objectos e fenómenos naturais envolventes na descoberta de padrões [6].

O DSR é um processo rigoroso que tem como objectivo resolução de problemas, avaliação do artefacto produzido e a comunicação dos resultados conseguidos [7]. Existem vários modelos que definem o conjunto de fases que caracterizam o DSR. Porém vamos abordar o modelo apresentado por Peffers. Segundo Peffers existem 5 fases que compõem o modelo proposto [7, 8, 9]:

Fase 1 - Identificação do Problema e Motivação: nesta fase é definido os problemas da pesquisa e o valor da solução proposta. É necessário justificar a solução com base em duas coisas: a motivação inerente a esta pesquisa e a aceitação da necessidade para ir em busca da solução do problema.

- Esta fase será desenvolvida e encontra-se no capítulo 1 deste projecto de tese.

Fase 2 - Definição dos objectivos para a solução: tendo por base a fase anterior dá-se a definição dos objectivos que se esperam atingir a partir da solução proposta.

- Esta fase será desenvolvida e encontra-se no capítulo 3 deste projecto de tese.

Fase 3 – Desenhar e Desenvolver o artefacto: Nesta fase desenha-se e implementa-se o artefacto proposto. É necessário nesta fase definir as funcionalidades desejadas, a sua arquitectura e desenvolver o artefacto. Nesta fase irá ser apresentado o seguinte:

- Definição da proposta de solução, que se encontra no capítulo 4 deste documento.
- Desenvolver um programa em linguagem C# com interface para o utilizador, que se encontra no capítulo 5.

Fase 4 - Demonstração: Nesta fase é demonstrada a aplicação da proposta de solução. O método para a descoberta de padrões e anti-padrões e aplicado a duas arquitecturas empresariais, desenhadas em ArchiMate de forma a detectar os padrões existentes em ambas. Após a detecção dos padrões é importante a sua documentação para a transmissão e utilização de conhecimento.

Na fase da demonstração é necessário:

- Realizar a simulação com arquitecturas empresariais.

- Comparar Arquitecturas.
- Gerar Padrões e anti-padrões
- Produzir padrões/anti-padrões no formato XML a partir do programa anteriormente desenvolvido.
- Catalogar os padrões e anti-padrões obtidos.

Fase 5 - Avaliação: Nesta fase é avaliado a importância da existência de um método para a descoberta de padrões e anti-padrões, assim como a capacidade deste método cobrir aspectos relevantes e fulcrais na optimização da detecção.

A estratégia de avaliação consiste numa avaliação quantitativa onde será medido o número de padrões descobertos assim como o tempo que demorou a sua descoberta. Nesta análise é distinguido a detecção de padrões simples assim como os complexos, a diferença entre este tipo de padrões é possível observar no ponto **4.2.2 Padrõees *out of the box***.

1.5 Organização do documento

De seguida descreve-se a estrutural deste documento, que é por 7 capítulos.

No capítulo 2 *"Estado da Arte":* é reunido um conjunto de conceitos utilizados no âmbito deste trabalho de investigação. A definição de arquitectura, princípios arquitecturais e de arquitectura empresarial são conceitos que são definidos neste capítulo. De seguida introduz-se a *framework* Zachman e ArchiMate que indicam formas de modelar a AE. Define-se o conceito de padrão assim como, diferentes formas de estruturar um padrão.

No capítulo 3

Capítulo 2

Estado da Arte

2.1 Arquitectura

Uma arquitectura consiste na descrição dos artefactos necessários para satisfazer os *drivers* arquitecturais [1] essenciais de um sistema [10]. O termo arquitectura é utilizado em inúmeras áreas não existindo concordância na sua definição. Martin Fowler diz-nos que existem dois aspectos comuns nas inúmeras facetas que a definição de uma arquitectura pode tomar. O primeiro aspecto consiste na decomposição de um sistema em alto nível e o segundo é a dificuldade de modificação de decisões anteriormente tomadas [3]. Uma arquitectura é desenvolvida e mantida pelo arquitecto de forma satisfazer as necessidades da empresa. Esta função pode realizada por uma pessoa, equipa ou organização ao longo do ciclo de vida da arquitectura sendo responsabilizado pelas suas consequências.

O IEEE define arquitectura como sendo "A estrutura organizacional de um sistema ou componente [11]." A norma IEEE STD 1471-2000 define mais especificamente que arquitectura é um "um conceito fundamental ou propriedade de um sistema incorporado nos seus elementos, relações, e os princípios que guiam o seu desenho e evolução." [2]

Existem decisões que são tomadas numa fase inicial no desenvolvimento de uma arquitectura que durante o ciclo de vida da empresa deixam de satisfazer as necessidades. As razões pela qual uma empresa se modificam são explicadas pelo de Michael Porter no modelo das Cinco Forças de Porter. Este modelo contribui para a definição da estratégia de negócio da empresa no seu ciclo de vida tendo em atenção as forças que influenciam uma empresa. São 5 as forças que influenciam uma empresa, esta influência define a competitividade da empresa no mercado, indicando os ganho da empresa e a capacidade de satisfazer os seus clientes [12]. Estas 5 forças são: competidores tradicionais, novos competidores no Mercado, Substituição de produtos e serviços, cliente e os fornecedores. E para além destas 5 forças tradicionais definidas no modelo de Porter existem mais 3 novas forças que tem que ser tomadas em consideração porque também influenciam a organização: globalização, digitalização e desregulamentação. Este modelo não é relevante para a análise de padrões, mas é importante

[1] Driver arquitectural é a combinação dos requisitos dos funcionais, de negócios e os atributos de qualidade da empresa que moldam a arquitectura ou o elemento em questão. Em http://www.sei.cmu.edu/architecture/start/glossary/ accessed 25-07-2014

[2] "http://www.iso-architecture.org/ieee-1471/defining-architecture.html", accessed 25-07-2014

referenciar a sua existência [12].

Estas cinco forças que influenciam uma empresa também influenciam a arquitectura empresarial que tem que responder as necessidades da empresa. Neste momento que compreendemos o conceito de arquitectura e as razões pelas quais uma empresa tem tendência a realizar mudanças para se adaptar às necessidades, é importante distinguir o conceito de arquitectura e de *design*. O foco da arquitectura reside nos requisitos essenciais a serem cumpridos pela empresa enquanto o design focase na restrição da implementação evitando resultados indesejados [10].

2.1.1 Principios Arquitecturais

Os princípios arquitecturais são fulcrais para garantir a eficiência da arquitectura empresarial garantindo que esta define a direcção futura da empresa alinhando as estratégias de alto nível e as decisões de *design*. Num ambiente de mudança e incertezas os princípios arquitecturais garantem a continuidade e a estabilidade [10]. É de salientar que os princípios arquitecturais sofrem da imaturidade das arquitecturas empresariais, não é claro quais os métodos e técnicas de criar e aplicar os princípios arquitecturais nas AE [10].

Danny Greefhorst no livro "Princípios Arquitecturais" indica que de acordo com o TOGAF, os princípios arquitecturais são directrizes gerais estipuladas para informar e apoiar o desenvolvimento dos objectivos da empresa, sendo raramente alteradas. Os princípios arquitecturais devem ser aprovados pelo gestor sénior de forma a orientar os passos durante a implementação, assim sendo os princípios tem que ser claros, concisos, de alto nível e de um número reduzido (normalmente a volta de 10) [10]. Os princípios arquitecturais preenchem as lacunas entre a estratégia e o *design* da solução assegurando a eficiência e eficácia da arquitectura empresarial como guia para coordenar o desenvolvimento de um projecto, servindo como pilar para a AE [10].

2.1.2 Arquitectura Empresarial

Em [2] uma arquitectura empresarial é definida como sendo "um conjunto coerente de princípios, métodos e modelos que são utilizados na concepção e realização de uma estrutura organizacional, processos de negócio, sistemas de informação e infra-estrutura da empresa".

Uma empresa ao longo do seu ciclo de vida é sujeita a vários desafios o que obriga a existência de uma evolução caso pretenda manter-se no mercado de forma competitiva. O desenvolvimento de uma AE faz com que se ganhe um maior controle na evolução da empresa ao longo da sua vida. Este controlo é alimentado pela definição dos seus produtos ou serviços que são oferecidos aos seus clientes através dos processos de negócios definidos e dos sistemas de informação que suportam esses processos. Os desafios de uma empresa resultam da mudança do ambiente onde esta se insere, por exemplo a globalização [10].

A AE é vista como um instrumento para definir a direcção futura da empresa, permitindo uma coesão da organização e da integração, graças a uma abordagem de multi-perspectivas [10]. Uma arquitectura deve oferecer um papel regulador, instrutivo e informativo. Regulado definindo directrizes de forma

clara e concisa de forma a restringir a liberdade de *design*, instrutivo definindo as instruções de forma a orientar o desenvolvimento de projectos e informativo de forma a apoiar a tomada de decisão quando necessária [10].

Estar alinhado com os diferentes aspectos de uma empresa (tais como, negocio, TI, recursos humanos, infra-estruturas, etc.) é o propósito da AE. Isso é conseguido através da definição das propriedades necessárias e suficientes para cumprir com os objectivos, providenciando desta forma uma restrição normativa da liberdade de *design* [10]. A respeito da AE, é de salientar que não existe um consenso como pode ser utilizado nem ate mesmo o seu conteúdo [10].

Uma AE, como já foi refecido anteriormente, está sujeita a mudanças sendo necessário definir os seus elementos. A AE é constituída por sub-arquitecturas: Arquitectura Organizacional, Arquitectura de Negócio e Arquitectura de Sistemas de Informação, sendo esta última sub-arquitectura constituída por arquitecturas informacional, Arquitectura Aplicacional e Arquitectura Tecnológica [13].

A arquitectura organizacional relaciona aspectos que envolvam recursos humanos para que estes estejam todos orientados no mesmo sentido garantindo a concretização da missão da empresa. A arquitectura organizacional não se relaciona com as especificidades do negócio ou com os mecanismos para a criação de valor mas garante o alinhamento dos recursos humanos com a estratégia de negócio [10].

A arquitectura de negócio específica os processos de negócio que é o ponto fulcral desta arquitectura pois descreve como a organização opera. A arquitectura informacional define as entidades informacionais que a organização necessita para definir os processos de negócio. As entidades informacionais são definidas ao detalhe tendo em consideração as suas propriedades e as suas inter-relações [13] [10].

A arquitectura aplicacional define as aplicações essenciais para dar suporte aos processos de negócio e as entidades informacionais. Este suporte é garantido através da utilização da matriz de CRUD que garante a relação entre as aplicações e os processo e as entidades informacionais. O suporte que é dado por esta arquitectura tem que garantir o acesso aos dados num período, formato e custos aceitáveis [10]. A arquitectura tecnológica centra-se na definição das tecnologias que dão suporte as aplicações que foram definidas na arquitectura aplicacional.

2.1.3 Modelação Arquitectura Empresarial

Framework Zachman

John Zachman em 1987 publicou a *framework* Zachman uma *framework* para AE. Esta framework reúne informação sobre os vários conceitos de uma organização, estruturando as representações que descrevem a organização de forma a classificar e organizar as arquitecturas que constituem a empresa [2].

A AE pode ser descrita ao utilizar esta *framework* em cinco perspectivas: *Planner(Scope), Owner(Enterpreise model), Designer(System Model), Builder (Technology model) e Subcontrator(Detailed representation)* e estas perspectivas podem ser classificadas segundo seis dimensões *data (what),*

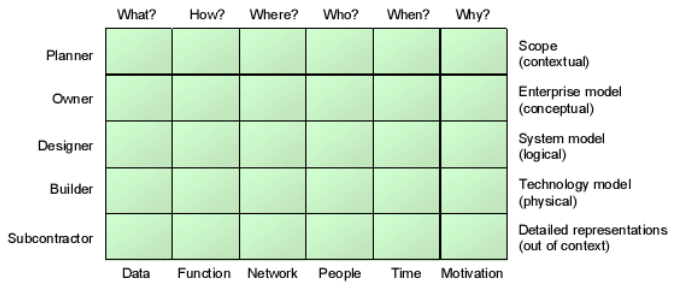


Figura 2.1: *Framework* Zachman

function (how), network (where), people (who), time (when), motivation (why). As dimensões e as perspectivas podem ser cruzadas passando a ter cinco camadas scope, enterprise model, System model, Technology model e Datailed representations [2].

Os conceitos aqui abordados acerca da perspectiva são definidos como se seguem: [2][14] **O Planeador (Scope)**, apresenta a perspectiva de quem planeia, sendo definido para o projecto o seu âmbito e as suas fronteiras. Na perspectiva do **Dono (Modelo Empresarial)**, é criado o primeiro modelo empresarial definindo a perspectiva do produto e a maneira como será utilizado. Nesta perspectiva, o **Projectista (Modelo do sistema)**, é especificado o produto definindo a forma de satisfazer os requisitos do Dono. Na perspectiva do **Construtor (modelo tecnológico)**, é definido o modelo tecnológico identificando o produto, os componentes e a maneira como se interligam. Tendo em consideração a plataforma e outras questões relacionadas com a performance é definido detalhadamente o modelo do sistema. No **SubContrato (Representação detalhada)**, é apresentado o código de implementação dos componentes.

A classificação é definida como se segue:[2] [14] **Dados(O que)**, responde ao "o que" esta envolvido identificando os dados que são importantes para o negócio. A entidade é o foco principal desta coluna, sendo que as entidades estão relacionadas através das relações. **Função(Como)**, indica como os processos são desenvolvido e como os eventos acontecem. **Rede(Onde)**, nesta coluna é indicado onde é necessário na rede fornecer suporte. **Pessoa(Quem)**, indica quem na organização e no negócio tem relevância. **Tempo(Quando)**, indica quando é que as coisas são realizadas. **Motivação(Porque)**, indica qual é a razão pela qual as coisas são realizadas, sedo aqui definidos os objectivos e a estratégia de negócio.

Assim sendo esta framework não indica o processo do desenvolvimento arquitectural, isto é, apenas indica como estruturar e organizar os artefactos não referindo metodologia para a definição da AE. As vantagens desta *framework* é que ela é fácil de entender, é definido de forma independente das ferramentas e metodologias e os prolemas podem ser mapeados de acordo com cada camada de forma a entender onde ela se encaixa [2].

ArchiMate

O ArchiMate é uma linguagem de modelação de arquitecturas empresariais que permite descrever diferentes domínios de uma empresa através da sua descrição e visualização conseguindo um foco nas

relações e dependências entre os componentes. ArquiMate encontra-se dividido em três camadas arquitecturais: negócio, aplicacional e tecnológico [15] [2]. A Camada de Negócio oferece serviços e produtos a clientes externos, que são facultados pelos processos de negócio que são desenvolvidos pela organização. Camada Aplicacional dá suporte a camada de negócio oferecendo serviços aplicacionais realizados por componentes aplicacionais. Camada Tecnológica oferece serviços de infra-estrutura que são necessários para executar aplicações sendo executados em hardware de comunicação e sistemas de software [2].

Em todas as camadas é utilizado o mesmo tipo de relações diferindo na sua granularidade, sendo que os serviços são responsáveis pela ligação entre camadas. Esta linguagem representa os componentes da arquitectura em três dimensões: elementos estruturais passivos, comportamentais e estruturais activos [2]. Os elementos estruturais passivos são os objectos que são manipulados pelo comportamento. Os elementos comportamentais são caracterizados como um objecto que modelam os elementos de comportamento. Os elementos estruturais activos são definidos por objectos que executam o comportamento [15] [2].

Um dos principais objectivos do ArquiMate é oferecer aos *stakeholders* uma forma de visualizar o sistema para que seja fácil interpretar e controlar a arquitectura. O ArquiMate facilita a avaliação da AE sendo possível avaliar o impacto de uma decisão [16].

2.1.4 Arquitectura de Referência

Uma arquitectura de referência (AR) aparece quando existe a necessidade de facilitar a criação de produtos e da necessidade de dar apoio ao ciclo de vida da empresa. A arquitectura de referência fornece [17]:

- Um léxico e taxonomia comum.
- Uma visão comum (Arquitectónica).
- Modularização e um contexto complementar.

O léxico e a taxonomia facilitam a comunicações entre diferentes dimensões. A visão comum arquitectónica tem o foco de alinhar esforços entre várias equipas. A modularização ajuda a dividir o esforço garantindo futuramente uma fácil integração [17].

A arquitectura de referência melhora a eficácia através [17]:

- A gestão da sinergia.
- Facultar orientação (princípios arquitecturais e boas-práticas.
- Fornecer uma base para a arquitectura.
- Capturar e partilhar padrões (arquitectónicos).

Um modelo de referência é um modelo conceptual abstracto que é utilizado para apoiar a especificação de um sistema composto por diferentes componentes, elementos e módulos, em que cada componente é representado por um modelo específico ou sub-modelo [18].

Uma das definições de arquitectura de referência é que especifica um modelo de referência de forma a mapear os elementos de software e o fluxo de dados com as funcionalidades do sistema. Os elementos de software definidos numa AR não necessitam de ser implementado em cada instanciação, uma vez que a alguns dos elementos existentes na arquitectura podem não ser uteis num contexto particular. Proporcionar uma estrutura que possa ser utilizada como uma estrutura base para o desenvolvimento de software um domínio específico é um dos principais objectivos da AR. Uma AR é baseada em padrões que facultam uma estrutura composta por soluções conhecidas e utilizada em sistemas existentes. Uma AR combina soluções de padrões em uma única estrutura sendo constituída por componentes que representam os papéis desempenhados por elementos constituintes dos padrões, [18].

A AR pode ser utilizada como sendo uma base para uma estrutura que é baseada em metadados [3] pois ajuda a estruturar uma solução abordando os principais objectivos num determinado domínio tendo em consideração os requisitos que foram especificados. Alguns componentes que a constituem como já foi referido podem ser removidos ou ser incluídos, depende das necessidades arquitecturais pois uma AR não é uma realização concreta da solução [18].

A AR é uma estrutura que ajuda na orientação durante a concepção e implementação de um sistema. A arquitectura deve guiar o desenvolvimento, na aplicação de um projecto e na modelação de forma sistemática durante o ciclo de vida da empresa [19].

A reflexão da experiência pode ser capturada através de princípios arquitecturais e boas-práticas fornecendo uma linha orientadora para desenvolvimentos posteriores, evitando a recorrência de más experiências do desenvolvimento de uma arquitectura. É melhorado a eficácia pois fornece um ponto de partida para discutir futuras alterações e extensões. A AR serve como uma modelo para arquitecturas futuras impedindo que problemas resolvidos voltem a acontecer [17].

Devido a missão, visão e estratégia da empresa, a AR encontra-se fortemente ligada a empresa. As arquitecturas passadas são transformadas em arquitecturas de referência porém é necessário que esta esteja orientada para o futuro. A missão, visão e estratégia são factores a ter em consideração no desenho das AR utilizando os conhecimentos passados [17].

A AR captura conhecimento anterior sendo necessário que os conceitos utilizados sejam aprovados para que a AR tenha valor. A validação dos conceitos em AR é frequentemente derivados de arquitecturas anteriores. É necessário ter atenção pois falhas em AR propagam-se para diversas arquitecturas e sistemas reais podendo destruir empresas [17].

Na *figura* 2.2 é demonstrado o fluxo de conceitos comprovados, problemas conhecidos e visão existente derivadas das necessidades para a AR. A AR fornece uma forma de orientar a evolução das arquitecturas existentes e influência os clientes e o negócio gerando a necessidade de mudança [17].

[3] Metadados: é uma maneira de descrever a história do design, localização, opções de configuração, as restrições e os requisitos de integreção de um objecto. Em http://dictionary.ieee.org/index/m-6.html accessed 25-07-2014

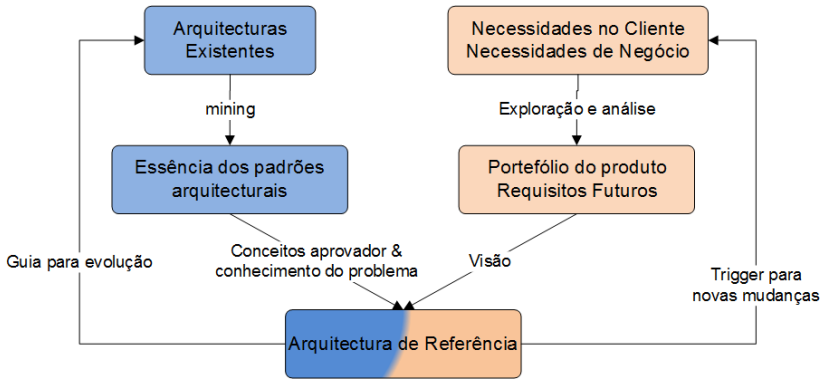


Figura 2.2: Fluxo de conceitos da AR
[17]

A AR deve descrever a essência de uma arquitectura realçando aspectos significativos e relevantes, podendo ser suportados por modelos mais detalhados, API, padrões, etc. O desafio de criar uma AR é que esta seja legível e eficaz para todas as dimensões em que será utilizado não devendo ser ambígua [17].

2.2 Padrão

Segundo Fowler [3], o padrão foca-se na solução para um problema específico que seja de certa forma comum em problemas recorrentes, essas soluções são desenvolvidas pelo conhecimento e experiência dos arquitectos desta forma os padrões estão enraizados na prática. Assim suscita a necessidade de catalogar os padrões para serem reutilizados *a posteriori*. Um padrão resolve problemas para um determinado contexto, porém este pode ser adaptado para outros contextos [20]. Um padrão arquitectural também é conhecido como estilo arquitectural [21].

Os padrões arquitecturais servem como fio condutor para a tomada de decisão por parte dos arquitectos pois estes reutilizam conhecimento já adquirido pela experiência de outrem. Esta reutilização é vantajosa porque o arquitecto consegue uma maior produtividade na busca de uma solução para um problema porque em *design time* ele consegue medir o impacto da sua reutilização pois os padrões são documentados descrevendo o problema, a solução genérica no seu respectivo contexto e a consequência de utilização desse padrão.

Apesar de parecer fácil a tarefa de utilizar informação previamente disponível, a tarefa de documentar essa informação útil por vezes não é simples. Esta dificuldade deve-se pelo facto que documentar factos e acontecimentos, quando colocadas em paralelo para outras tarefas não mais prioritárias, são maior parte das vezes deixadas para segundo plano, e quando chega o momento para passar para papel a informação, muita não é recordada [4].

Os padrões não são soluções originais, são meramente observações que são feitas às arquitecturas. Encontrar padrões é uma tarefa difícil mas é algo muito valioso, a descoberta de padrões dá-se através da observação do funcionamento da arquitectura e pela detecção da essência da solução definida. Contudo é de salientar que quando se pretende utilizar um padrão não basta apenas aplica-lo sem

levantar questões porque apesar de ser a solução para um problema, a solução nem sempre é a mesma [3]. Um anti-padrão não é mais do que um padrão que pode ser utilizado todavia não produz um resultado desejado, um padrão é denominado anti-padrão quando traz mais consequência negativas que positivas.

Os programadores experientes acreditam que quando há um novo problema por resolver existe sempre uma solução que foi anteriormente criada para um problema semelhante. Se os problemas não forem semelhantes então uma solução idêntica raramente irá resolver um novo problema, porém se os problemas forem semelhantes então uma solução semelhante deverá funcionar [22].

Quando existe um novo desenvolvimento de um padrão é difícil a abstracção entre as duas arquitecturas para que os aspectos genéricos da solução original possam ser aplicados a um novo problema [22].

A aplicação dos padrões é uma preocupação associada aos padrões. O problema é identificar a natureza do problema e examinar o melhor padrão que pode ser aplicado existente na biblioteca que pode ser aplicado denominando-se por **padrão de encubação**. Esta denominação indica que não estamos a criar nada de novo mas sim estamos "em desenvolvimento a partir de rudimentos preexistentes". Estes rudimentos preexistentes são os padrões que foram capturados e que podem ser utilizados em busca de uma solução [22].

2.2.1 Estrutura do Padrão de Martin Fowler

Martin Fowler em [3] diz que cada autor escolhe o seu formato para a descrição de padrões. Ele escolheu um formato afirmado que existem muitas questões que forma ignoradas e que são importantes para a especificação do padrão.

A primeira característica que um padrão deve ter é um nome. Ser atribuído um nome a um padrão é importante desta forma estará a ser criado um vocabulário para que os arquitectos, engenheiros, técnico, entre outros comuniquem de forma eficaz. Consequentemente se alguém referir que irá ser utilizado o padrão cliente-servidor [3] e os membros envolventes no projecto conhecerem esse padrão todos estarão a visualizar e a compreender a solução adoptada. Outras duas características que ajudam a descrever um padrão é: a intenção e o esboço. A intenção é apenas um resumo do padrão sendo que o esboço é uma representação gráfica do padrão. O esboço é um auxílio para que o padrão seja lembrado mesmo que o seu nome não seja recordado.

A quarta característica identifica qual é o problema que o padrão tenta resolver, apesar de um padrão poder resolver diversos problemas é necessário identificar especialmente a causa para a sua existência. A quinta característica tem a ver com o "como funciona" de forma a descrever a solução apresentada pelo padrão. Aqui são expostas questões de implementação e variantes. A próxima característica diz respeito ao "quando utilizar" descrevendo quando é que o padrão deve ser utilizado pois os padrões são apenas uma alternativa não sendo soluções exactas e únicas [3].

2.2.2 Estrutura do Padrão de Eric Gamma, Richard Helm, Ralph Johnson e John Vlissides

No livro "Design Patterns: Elements of Reusable Object - Oriented Software"[23] é indicado que o padrão é constituído por quatro elementos essenciais o nome do padrão, o problema, a solução e as consequências. O nome do padrão é um identificador que é utilizado para identificar o problema que foi resolvido, assim como a sua solução e as consequências. Ter um vocabulário de padrões permite que os passamos referenciar transmitindo o conhecimento que o nome desse padrão representa.

O problema indica quando o padrão deve ser usado, explicar o problema e o seu contexto. Por vezes é simplesmente uma lista de condições que devem ser satisfeitas muito antes de aplicar o padrão [23].

A solução passa por descrever os elementos que constituem o *design*, indicando as suas relações e responsabilidades. A solução descreve um desenho genérico para que possa ser aplicado a várias situações diferentes [23].

Finalizando, a descrição dos quatro elementos essenciais identificados em [24], as consequências são o resultado da aplicação de um padrão, sendo importante para avaliar o impacto de uma decisão assim como analisar alternativas para um determinado problema. As consequências permitem compreender o custo e o benefício da aplicação de um padrão, o impacto recaem na flexibilidade, portabilidade e extensibilidade de um sistema.

2.3 Resumo

A arquitectura empresarial é a definição base para esta investigação, uma vez que é a partir dela que são detectados os problemas a resolver e também é a partir dela que os padrões e anti-padrões são gerados.

Foi escolhida a linguagem de modelação ArchiMate para a fase da demonstração. Uma das razões pela qual esta linguagem de modelação foi escolhida é que pode ser representada na arquitectura as motivações e intenções que levaram ao seu desenvolvimento, tais como: os objectivos de negócio, princípios, requisitos, custos e performance, que por sua vez são importantes para a análise de uma arquitectura quando estamos a detectar padrões [15]. Outro factor é porque ela nos fornece uma representação global da AE demonstrando a forma como os diferentes domínios se relacionam, ao contrário da *Framework* Zachman em que as relações entre as células não se encontram explícitas. O ArchiMate, como explica a relação interdomínios permite avaliar o impacto da tomada de uma decisão na arquitectura [16]. Um dos obstáculos da *Framework* Zachman é que contém um elevado número de células e as relações entre as células não se encontram bem especificadas.

Ao analisar as estruturas que são descritas em [3] e em [23]para este projecto verificamos que cada definição apresenta carência de elementos para caracterizar um padrão, por essa razão irá ser utilizada uma caracterização composta por ambas as definições estruturais. Assim sendo, a caracterização do padrão é composta pelo: nome do padrão, problema, contexto, solução, consequência, esboço da arquitectura genérica e forma de funcionamento. Para alinhar a descrição do padrão com a AE, este será

ainda caracterizado pelo tipo de AE presente e pelos princípios arquitecturais da empresa presentes no padrão. Os princípios arquitecturais apesar de se encontrarem fora do âmbito desta investigação, não é desvalorizada a sua importância nas AEs. E é por esse motivo que uma das caracterizações do padrão irá ser os princípios arquitecturais que se encontram presentes nas arquitecturas e que são representados pelos padrões. Na proposta de solução estes elementos serão especificados de forma a esclarecer o que deve ser definido em cada elemento.

Capítulo 3

Contributos

Neste projecto as contribuições gerais consistem em:

- Demonstrar a importância de padrões e anti-padrões para qualquer AE.
- Identificar os pontos que numa arquitectura são relevantes para a detecção de padrões.
- Criar um catalogo com padrões e anti-padrões específicos para uma determinada área de negócio.
- Criar padrões que cubram todas as arquitecturas que constituem uma AE.
- Identificar a estrutura para documentar padrões e anti-padrão de forma a conter informação necessária para a sua utilização.
- Propor um processo para á descoberta de padrões.
- Aplicar um exemplo à proposta de solução definida.

De seguida, descreve-se as contribuições principais do método de descoberta de padrões para uma indústria específica:

- Descoberta de padrões que são específicos numa determinada indústria e que podem ser aplicados na implementação de soluções à problemas comuns, sublinhando as vantagens que a aplicação de um padrão definido e documentado pode trazer. Como por exemplo: ganho de tempo, redução de custos, diminuição dos erros, transparência na aplicação de uma solução devido às consequências que esta poderá trazer e soluções para problemas inerentes a área de domínio onde a empresa se contextualiza.
- Descoberta de anti-padrões de forma a conhecer através da documentação as suas consequências negativas á curto ou a longo prazo. O anti-padrão deverá não ser aplicado numa arquitectura mas caso seja, o arquitecto saberá as consequências que a sua aplicação trará encontrando-se preparado para tal.

- Melhoria da AE devido à análise crítica dos prós e contras da aplicação de um padrão, podendo à *posteriori* melhorar o padrão existente no catálogo complementando com informação.
- Criação de um método para descoberta de padrões que visa a avaliação e documentação dos padrões e anti-padrões descobertos provenientes de arquitecturas geradas pelas empresas de uma determinada área ao longo do seu ciclo de vida.
- Documentar padrões e anti-padrões de forma a reutilizar informação previamente adquirida.

É de salientar que a mudança de cultura numa organização para documentar e detectar padrões para que exista à partilha de conhecimento é essencial numa empresa, mas este aspecto encontra-se fora do âmbito desta tese. Outro aspecto essencial para a concretização dos objectivos e da missão de uma empresa é que as arquitecturas desenvolvidas estejam alinhadas com o negócio para que os padrões detectados tenham esse factor. Todavia este aspecto também encontra-se para além da fronteira de investigação deste projecto. É importante referir que apesar de definir-mos padrões que possam respeitar princípios arquitecturais não é garantido nem obrigatório que todos os padrões detectados respeitem esses princípios pois como já foi referido esse alinhamento encontra-se fora da fronteira delineada para este projecto.

A AMA [1] é a empresa onde realizar-se-á esta investigação e e que pretende usufruir das vantagens do método da descoberta de padrões. A proposta de solução irá ser analisada, avaliada e refinada. Alguma das responsabilidades da AMA com a Administração Pública são as seguites(AP)[25]:

- Gerir e desenvolver redes de lojas para os cidadãos e para as empresas em sistema de balcões multisserviços, integrados e especializados, articulando com os sistemas de atendimento em voz e rede;

Atendendo às responsabilidades da AMA e da sua intervenção na AP, a definição de padrões e anti-padrões ajuda a cumprir os seus objectivos para com esta entidade. É necessário ter em consideração que existe vários Ministérios onde a AMA actua podendo numa indústria específica existir procedimentos semelhante conseguindo detectar padrões e anti-padrões. Este projecto de investigação ajuda AMA a [25]:

- Ter um catálogo de padrões e anti-padrões relacionados com uma determinada indústria.
- Implementar soluções para problemas numa determinada indústria de forma ponderada podendo analisar o historial e a informação necessária para tomar uma decisão de desenho arquitectural.
- Melhorar as AE desenvolvidas devido à utilização de informação já experimentada e documentada.
- Gerar padrões que ao longo do ciclo de vida da AMA serão cada vez mais refinados sofrendo "amadurecimento" devido de informação e experiência à arquitectos provenientes de várias entidades proporcionando uma melhoria na AE.

[1] A Agência para a Modernização Administrativa (AMA) é um instituto público com autonomia financeira e administrativa possuindo o seu próprio património. A AMA é responsável por ≪facilitar o quotidiano dos cidadãos e dos agentes económicos, tornar a interacção com o Estado mais conveniente e transparente e agilizar o funcionamento interno da Administração pública, com o suporte decisivo das tecnologias de informação (TI)≫[25].

- Reduzir custos e tempo de desenho e de implementação arquitectural, possibilitando aplicar uma solução dependendo do problema específico

Capítulo 4

Proposta de Solução

A proposta da solução denomina-se por RCGD (*Revision, Comparasion, Generation and Documentation*, ver *figura* 4.1). A fase *Revision* passa pela escolha de qualquer arquitectura e respectiva análise, a fase *Comparasion* passa por um conjunto de etapas que detectam os padrões candidatos analisando diferentes granularidades granularidade sendo que a posteriori alguns destes padrões poderão ser classificados como padrões *out of the box*. A próxima fase, *Generation* passa por gerar possíveis padrões arquitecturais, que poderão ser aceites ou não por membros responsáveis pela implementação de soluções arquitecturais na organização. E na fase *Documentation* é documentada a informação relevante para a transmissão de conhecimento para uso futuro.

4.1 Análise das Arquitecturas - 1ª Fase

Nesta fase é aceite qualquer arquitectura para a análise sendo necessário reunir a sua documentação. A documentação é importante para a compreensão da arquitectura e para extrair informação pertinente para os passos da RCGD. Inicialmente para cada arquitectura é obrigatório capturar a seguinte informação:

Problema: é necessário indicar qual é o problema que a arquitectura resolve. Sendo possível identificar os requisitos e restrições que a arquitectura implementa.

Contexto: listar situações onde o problema ocorre, oferecendo ao arquitecto uma base para o desenvolvimento de novas arquitecturas. É necessário realizar o levantamento das funcionalidades para

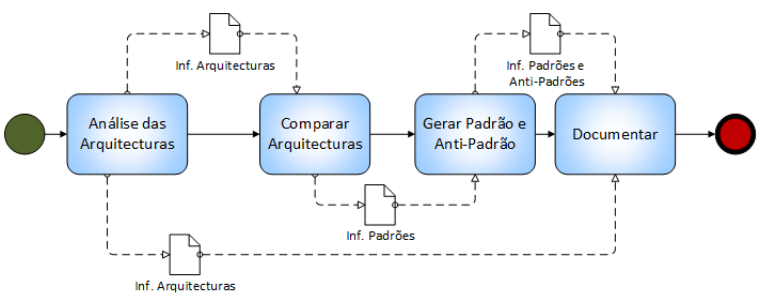


Figura 4.1: Fases da RCGD

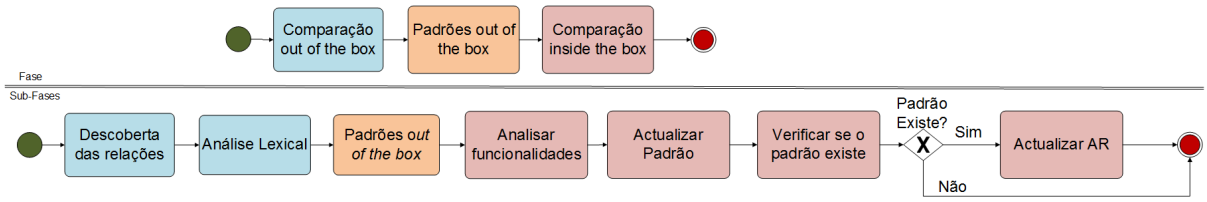


Figura 4.2: Etapas da fase de comparação de arquitecturas

detectar a similaridade entre objectos. O levantamento pode ser realizado em qualquer altura desde que seja garantido que na subfase ”comparação *inside de box*”as funcionalidades dos objectos que constitui o padrão candidato estejam listada. O levantamento das funcionalidades pode ser realizado à todos os objectos que constitui as duas arquitecturas em analise ou apenas aos objectos que constituem o padrão *out of the box*.

Para que seja possível compreender as arquitecturas em análise é necessário colectar informação sobre estas arquitecturas. Nesta fase é necessário ter em consideração que as arquitecturas têm um grande volume de informação sendo necessário um grande esforço [26].

4.2 Comparar Arquitecturas - 2ª Fase

Para que seja efectuado a comparação entre as arquitecturas é necessário que ambas estejam modeladas na mesma linguagem de modelação para que o metamodelo base seja o mesmo. Se as arquitecturas em análise não têm por base o mesmo metamodelo ou problema e contexto similar não podemos prosseguir com a comparação. Por outras palavras, as arquitecturas apenas prosseguem para a comparação se tiverem por base o mesmo metamodelo , o problema e o contexto semelhante.

Nesta fase pretende-se detectar os objectos que são similares realizando comparações de granularidades diferente. São duas as diferentes granularidades que constituem a detecção da similaridade: *out of the box* e também a granularidade *inside the box*. *Out of the box* corresponde a características externas da arquitectura que se salientam a primeira vista como por exemplo o nome do objecto, tipo de camada, tipo de objecto, etc. *Inside the box* corresponde a características internas de cada objecto, mais especificamente as suas funcionalidades. Na detecção de similaridade *out of the box* o principal objectivo é encontrar pares de objectos similares ao nível de *out of the box* gerando um conjunto de objectos com uma maior probabilidade de serem similares para que á análise *inside the box* seja realizada a um menor conjunto de combinações. Na *figura* 4.2 encontram-se os passos para realizar a comparação arquitectural.

4.2.1 Comparação *out of the box*

Primeiramente irá ser realizado uma análise lexical com o intuito de definir que tipo de informação deve ser a *posteriori* analisada, para que seja possível detectar similaridade entre os objectos das arquitecturas. Para que se inicie a detecção da similaridade temos que ter pelo menos duas arquitecturas que tenham passado pela 1ª fase. Na comparação *out of the box* é realizado a primeira detecção de

similaridade sendo por exemplo comparado o tipo de *layer*, o tipo de objecto, nome e as relações entre objectos. A informação *out of the box* que é comparada depende do metamodelo das arquitecturas porque há aspectos *out of the box* que mudam de acordo com o metamodelo utilizado. É *out of the box* porque não é analisado as funcionalidades dos objectos que constitui as arquitecturas. Após ter-mos pares de objectos similares dá-se a detecção das relações similares entre os objectos similares.

Análise Lexical

Atendendo que necessitamos realizar uma análise à todos os objectos de ambas as arquitecturas e compara-los para encontrar objectos similares, denominaremos essa acção por *scanning*. *Scanning* porque é feito um scanner as arquitecturas extraindo e analisando informação.

Para realizar o *scanning* temos que ter em atenção a linguagem de modelação em que a arquitectura foi modelada. É necessário definir que informação é suficiente guardar ao realizar o *scanning* para comparar dois objectos de arquitecturas diferentes. Por exemplo, se uma arquitectura estiver sido modelada em ArchiMate o tipo de camada e o tipo de objecto são informações imprescindíveis de serem armazenadas. Com este tipo de informação diminuí-mos o número de combinações que é necessário realizar para detectar os objectos similares. Como bengala para obter toda a informação imprescindível para a análise temos que responder à seguinte pergunta: O que é necessário que se verifique entre dois objectos para que a similaridade entre ambos seja calculada? É de salientar que as condições que têm de ser respeitadas para que seja realizado a comparação entre dois objectos, apenas referem-se as características da linguagem de modelação da arquitectura. Para que possamos passar para a próxima subfase é necessário definir que informação tem que ser extraída das arquitecturas. Pois necessitamos de definir de que forma irá ser comparada os objectos das arquitecturas diferentes.

O tipo de relação que um objecto tem com outros, assim como os próprios objectos com quem se relaciona são informações que têm que ser armazenadas independentemente da linguagem de modelação em que a arquitectura se encontra modelada.

Descoberta da relação entre os objectos

Nesta fase calcula-se a similaridade entre o nome dos objectos. Esse cálculo apenas acontece após a definição da informação que temos que analisar pois apenas analisamos a similaridade entre dois nomes de objectos se outras condições se verificarem, como por exemplo o tipo de camada e o tipo de objecto.

O nome do objecto pode ser constituído por uma ou mais palavras, para tal é necessário dividir o cálculo da similaridade entre duas fases: primeiramente calcular a similaridade entre cada palavra que constitui os objectos cobrindo todas as combinações e de seguida calcular a similaridade global atendendo ao número de palavras similar entre objectos.

Considera-se que dois objectos são semelhantes se forem do mesmo tipo e tiverem uma similaridade igual ou superior a 50%. Para verificar o tipo de similaridade que existe entre duas palavras é necessário escolher um algoritmo(como por exemplo o Levenshtein Metric, Smith Waterman, JaroMe-

tric, etc [27].).

Porém, antes de calcular qualquer tipo de similaridade é necessário ter em atenção determinados aspectos. Verificar se alguma das palavras é sinónima, antónima, preposição ou conjunção. Se ambas palavras forem sinónimas é escusado realizar o cálculo da similaridade pois estas palavras têm similaridade 100%, de igual forma, se forem antónimas tem similaridade de 0%. Caso algumas das duas palavras em comparação sejam preposições ou conjunções não é calculado a similaridade entre essas duas palavras.

Para a descoberta da relação entre dois objectos seguiu-se o seguinte processo, ver *figura* 4.3. Na *figura* 4.3 é detalhado o fluxo para a descoberta da relação entre dois objectos. Após ter-mos dois objectos seleccionados, o início do fluxo passa pela escolha de uma palavra de cada um dos objectos e de seguida verifica-se se alguma das palavras seleccionadas é uma preposição ou conjunção. Caso alguma das palavras seja conjunção ou preposição é descartada, seleccionando outra combinação. Caso sejam as duas então é novamente escolhido outra combinação de palavras. Se não se conferir que é preposições ou conjunções, verificamos se as palavras são antónimas. Caso sejam então novamente é escolhida outra combinação de palavras. Caso não sejam antónimas dá-se o cálculo de similaridade através de um algoritmo de cálculo da similaridade a escolha, apenas se as palavras não forem sinónimos. Caso existam mais combinações de palavras por analisar o fluxo é inicializado, caso contrário é calculado a similaridade dos objectos tendo por base a similaridade das palavras calculadas. A similaridade entre o nome dos objectos consiste na percentagem de similaridade. Para descobrir as

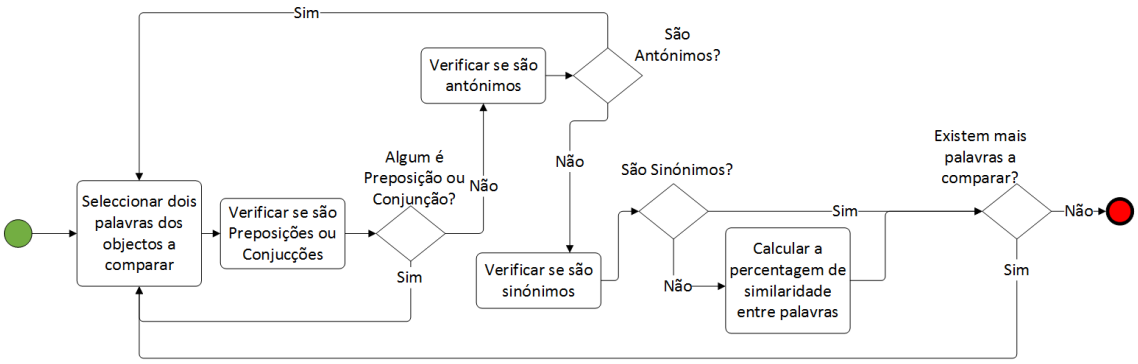


Figura 4.3: Fluxo para a descoberta da relação entre dois objectos

relações entre os objectos temos que verificar em primeiro lugar a sua similaridade.

Nesta fase para além da detecção dos objectos similares é necessário identificar se existe similaridade nas relações entre os objectos que são similares. Isto é, identificar qual dos objectos similares tem relação com outros objectos similares e essa relação tem que existir em ambas as arquitecturas. A análise à similaridade das relações apenas é realizada através da comparação dos objectos que são similares entre si. Esta etapa visa comparar o tipo de relação e o tipo de objectos similares. O processo de comparação entre objectos termina quando todos os objectos tenham sido comparados e as suas relações analisadas.

Os objectos similares são classificados como padrões candidatos, para que sejam classificados

como padrão ou anti-padrão arquitectural tem que passar por algumas fases. Os padrões candidatos podem ser constituídos por um ou mais objectos. Todo o padrão candidato pode ser classificado a posteriori como padrão *out of the box*. Para que o padrão candidato seja classificado como padrão *out of the box* tem que passar por uma selecção. Todo o padrão candidato pode ser classificado como padrão *out of the box*, porém podemos descartar a primeira vista certos padrões candidatos que foram classificados como similares mas que após uma análise não o são. Apenas os padrões *out of the box* podem ser classificados como padrão arquitectural, que poderá ser um anti-padrão.

Em resumo para a descoberta da relação entre os objectos temos que:

- Verificar se os objectos são do mesmo tipo;
- Verificar se alguma das palavra que constitui o nome do objecto é preposição ou conjunção;
- Verificar se as palavras que estão sendo comparadas são sinónimos ou antónimos;
- Calcular a similaridade dos objectos tendo em consideração à similaridade calculada entre as palavras que constituem o nome dos objectos;
- Verificar se os objectos similares tem uma relação com outros objectos similares.

Para detectar se objectos similares têm uma relação com outros objectos similares temos que utilizar a seguinte regra:

Sabendo que A e B correspondem a objectos diferentes e R é a relação entre A e B, então,

`Se A1 tem R1 com B1 e A2 tem R2 com B2 e A1.tipoObjecto = A2.tipoObjecto e A1.TipoCamada = A2.TipoCamada e R1.tipo=R2.tipo e B1.tipoObjecto = B2.tipoObject e B1.TipoCamada = B2.TipoCamadao e B1.nome=B2.nome e A1.nome = A2.nome Então tem uma relação similar`

Esta regra serve para encontrar relações que existam entre objectos similares com outros objectos similares.

4.2.2 Padrões *out of the box*

Nesta fase classificam-se os padrões *out of the box* que chegam a esta fase. Este tipo de padrão não estão completos pois é necessário realizar uma análise as funcionalidades, sendo que este refinamento é realizado na comparação *inside the box*. A classificação é importante para a catalogação dos padrões para que estejam organizados.

Antes de efectuar qualquer classificação dos padrões *out of the box* é necessário ter em consideração como se denominam os padrões encontrados de acordo com a sua constituição. As regras de denominação são as seguintes:

Regra denominação 1: Denomina-se, padrão complexo quando mais que um objecto faz parte do padrão gerado.

Regra denominação 2: Denomina-se, padrão Simples quando apenas um objecto faz parte do padrão gerado.

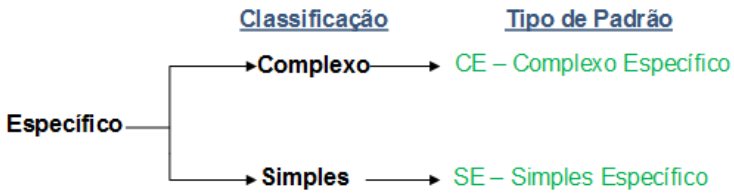


Figura 4.4: Classificação do padrão/anti-padrão

Regra denominação 3: Classifica-se o padrão como específico quando todos os objectos pertencentes ao padrão gerado são objectos semelhantes.

Tendo em consideração as seguintes regras de denominação dos padrões encontrados é necessário classifica-los. A *figura* 4.4 mostra a classificação de um padrão de acordo com as suas características.

4.2.3 Comparação *inside the box*

Nesta fase é analisado os aspectos internos dos objectos que constituem o padrão *out of the box* relativamente as funcionalidades que constituem cada objecto.

Esta fase necessita de intervenção humana para indicar a semelhança das funcionalidades entre dois objectos que se encontram em comparação. Nesta fase irá ser utilizado o levantamento já realizado das funcionalidades dos objectos de cada arquitectura. O objectivo do levantamento das funcionalidades de cada objecto é ajudar a pessoa responsável por verificar se dois objectos são similares entre si.

Matching entre objectos dos padrões *out of the box*

Apenas se pode analisar as funcionalidades entre dois objectos de padrões *out of the box* se estes objectos tiverem sido anteriormente definidos como similares. Assim sendo, para cada par de objectos irá ser comparadas as funcionalidades de forma a detectar a similaridade. Para que se considerem as funcionalidades entre dois objectos semelhantes, não é necessário que sejam 100% similar é apenas necessário que pelo menos uma das funcionalidades seja semelhante entre ambos os objectos. Se os objectos são semelhantes a nível das funcionalidades o padrão out of the box passa a ser um padrão arquitectural, caso contrário é excluído sendo necessário generalizar o padrão encontrado. Mais tarde falare-mos dessa generalização. Resumindo, todos os padrões que passam a fase da análise das funcionalidades de forma positiva são padrões arquitecturais. É de salientar que mesmo que dois objectos de padrões *out of the box* diferentes tenham uma funcionalidade similar, se não forem do mesmo tipo ou da mesma camada não podem ser classificada como similar.

Na *figura* 4.5 é possível verificar de que forma é realizado o *matching* dos objectos das arquitecturas em análise que passaram as fases anteriores. Este *matching* das funcionalidades apenas é realizado se os objectos forem do mesmo tipo e se forem similares. Após todos estes passos conseguimos saber quais os objectos que são similares entre si tendo em atenção as respectivas relações.

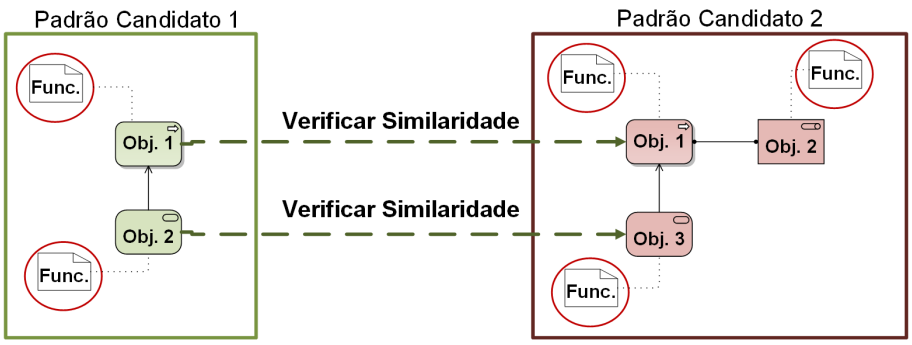


Figura 4.5: Matching das funcionalidade entre os objectos dos padrões candidatos

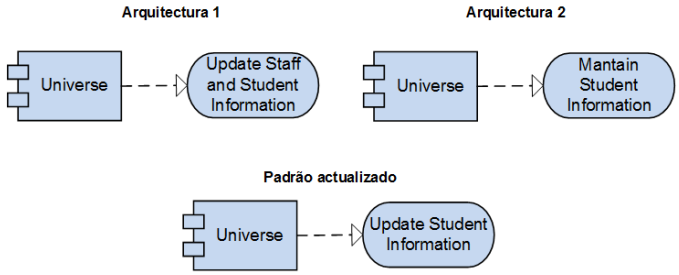


Figura 4.6: Etapas da fase de comparação de arquitecturas

Actualização de padrões

Os objectos são similares porque para além de terem as mesmas características externas também funcionalidades similares. O ponto importante na comparação *inside the box* é que para além de indicar que um padrão *out of the box* é ou não um padrão arquitectural pode-mos completar a especificação do padrão para a sua catalogação, documentando as suas funcionalidades.

Até esta fase não temos nenhum padrão definido, apenas temos objectos que são semelhantes devido à repetição de acontecimentos e que tem informação para gerar um padrão arquitectural. Portanto, tendo por base os pares similares temos que definir um padrão arquitectural. Para tal, é apenas necessário generalizar o nome de cada objecto que compõe o padrão. Na *figura* 4.6 pode-se verificar que temos dois pares de padrões similares que pertencem as arquitecturas 1 e 2 e que após a actualização de padrões é gerado um padrão arquitectural com o nome generalizado.

Arquitectura de Referência(AR)

Após definir se o padrão *out of the box* é um padrão arquitectural é necessário verificar se o padrão descoberto já existe, para tal irá ser utilizado a arquitectura de referência(AR). A AR é actualizanda consoante a descoberta de padrões arquitecturais.

Numa fase inicial não existirá AR uma vez que será construída após a descoberta de padrões arquitecturais. Como é possível verificar na *figura* 2.2 os padrões estão na base da AR. A construção da AR será explicada mais a frente. Primeiro é necessário saber de que forma o padrão descoberto irá ser analisado para verificar se o padrão já foi ou não descoberto. Gostaria de salientar que sempre que não exista relação entre um padrão arquitectural e a AR significa que foi descoberto um novo padrão

arquitectural e que a AR tem que ser actualizada.

- Correspondência dos objectos com a AR

Para a realização deste passo já terá que existir uma AR. Caso não exista uma AR teremos que passar para o próximo passo, a construção da AR utilizando o padrão arquitectural em análise.

Nesta AR os elementos que à constituem denominaremos por *hotspot*. *Hotspot* porque são os pontos de acesso as definições das funcionalidades definidas na AR e é nestes *hotspot* que é realizado a correspondência com os objectos em análise [18].

Para verificar se o padrão arquitectural já foi descoberto temos que comparar os *hotspots* da AR com os elementos do padrão arquitectural realizando uma correspondência entre eles. A correspondência entre o padrão e a AR é realizada analisando as funcionalidades de cada objecto que compõe o padrão arquitectural e as funcionalidades de cada *hotspot* que compõe a AR.

Assim sendo, para cada objecto do padrão arquitectural irá ser comparado as suas funcionalidades com os *hotspots* da AR. Se os objectos que estão a ser analisados coincidirem com algum *hotspot* isso significa que são similares e que esse objecto existe. Para que se considere que existe similaridade entre os objectos do padrão arquitectural e os *hotspots* da AR, não é necessário que sejam 100% similar é apenas necessário que pelo menos uma funcionalidade seja semelhante.

O *matching* das funcionalidades entre o padrão e a AR é realizado da mesma forma como descobrimos a similaridade entre objectos de arquitectura diferentes, analisando primeiramente aspectos externos e depois aspectos internos, ver *figura* 4.5.

Cenários que podem surgir durante a correspondência do padrão com a AR:

  - Todos os objectos que constituem o padrão arquitectural têm correspondência com a AR: quando isto acontece significa que o padrão já foi descoberto portanto a AR não tem que ser actualizada. Sendo necessário verificar se o padrão arquitectural descoberto encontra-se catalogado.
  - Algum dos objectos que constitui o padrão arquitectural tem correspondência com a AR: isto significa que alguns dos objectos que se encontram no padrão arquitectural já se encontram na AR. Portanto a AR deve ser actualizada com os restantes objectos do padrão arquitectural e as suas respectivas relações e o catálogo deve ser actualizado.
  - Nenhum dos objectos que constitui o padrão tem correspondência com a AR: neste caso nenhum objecto do padrão arquitectural existe na AR, assim sendo é necessário actualizar a AR e catalogar o respectivo padrão.

- Construção/Actualização da Arquitectura de Referência

No início da descoberta de padrões poderá não existir uma AR portanto o passo anterior da correspondência entre o padrão arquitectural e a AR não pode ser executado. Portanto o padrão descoberto nesse momento passa a ser a AR.

Ao longo da descoberta de padrões a AR tem que ser actualizada em duas situações, quando:

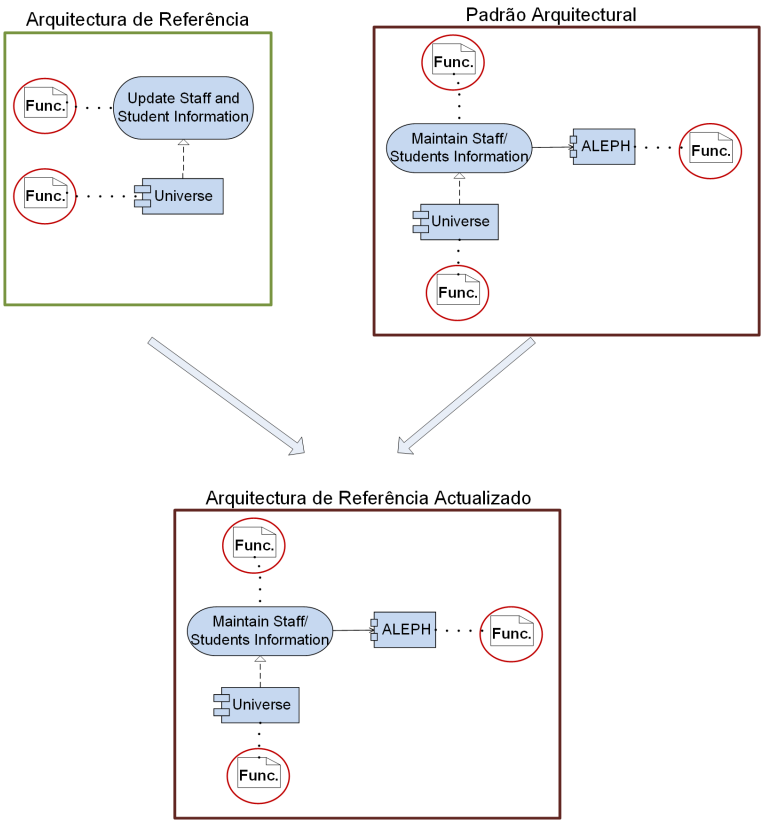


Figura 4.7: Actualização da AR

- Algum dos objectos que constitui o padrão arquitectural tem correspondência com a AR;

- Nenhum dos objectos que constitui o padrão tem correspondência com a AR.

Na *figura* 4.7 é possível visualizar a primeira situação em que mantém-se o *hotspot* com funcionalidade similar e acrescenta-se à AR os restantes objectos do padrão arquitectural. Na segunda situação é acrescentado à AR todo o padrão arquitectural.

Quando dá-se a modelação da AR é necessário que cada *hotspot* tenha uma breve descrição sobre a sua funcionalidade de forma a que correspondências futuras da AR com padrões arquitecturais sejam mais facilitadas.

Tendo em consideração a mudança no ciclo de vida das empresas, a AR pode ficar obsoleta existindo a necessidade de à actualizar de forma a dar resposta as novas necessidade. Desta forma pode ser acrescentado à AR os novos elementos lembrando que essa alteração passa sempre pela sempre pela aprovação.

4.3 Geração - 3ª Fase

Esta 3ª fase pode ser muito subjectiva pois depende da opinião que cada qual tem sobre um sistema e as soluções para os problemas. Nesta etapa é importante reflectir sobre os padrões encontrados, sendo importante pensar se o padrão descoberto traz-nos informação relevante. Para isso é necessário que os padrões arquitecturais sejam aceites por um conjunto de membros da organização. Essa aceitação

Classificação	Consequências		Tipo Solução	
	Negativa	Positiva	Longo prazo	Curto prazo
Positiva		X	X	X
Neutra				X
Negativa	X		X	X

Figura 4.8: Caracterização do padrão encontrado

passa tanto pelos arquitectos como pelos programadores, sendo o arquitecto um dos membros com mais peso na avaliação dos padrões encontrados. De acordo com a aceitação o padrão pode vir a ser descartado.

É necessário verificar aspectos práticos sobre a aplicação do padrão numa arquitectura sendo necessário:

- É necessário realizar um estudo sobre as consequências da aplicação dessa padrão numa arquitectura, tendo por base a experiência da implementação dessa solução nas arquitecturas.
- De acordo com estudo das consequências realizadas é necessário classificar os padrões como: Positivo, Neutro ou Negativo. A classificação como Positivo tem em consideração a utilização do padrão na solução do problema a longo prazo e a curto prazo, não trazendo consequências negativas. O Neutro não gera nem consequências positivas nem consequências negativas, mas resolve o problema a curto prazo. E Negativo resolve o problema a longo ou a curto prazo gerando um maior número de consequências negativas do que consequências positivas.

Atendendo aos aspectos práticos sobre a aplicação do padrão, a documentação das arquitecturas é essencial nesta fase para analisar as consequências positivas e negativas.

Os padrões que não são aceites pelos membros da empresa são colocados no *garbage colector* porque no futuro o valor deste padrão poderão alterar-se passando a ser relevante para a organização. Quando um padrão arquitectural é colocado no *garbage colector* é necessário que a documentação referente ao padrão esteja anexada ao mesmo, para que no futuro o padrão seja utilizado e contenha a sua informação.

Depois de aprovar ou não os padrões gerados é necessário classifica-los como padrões ou anti-padrões tendo em consideração a classificação de positivo, neutro ou negativo.

Os padrões classificados como neutros passam a ser padrões, porque apesar de não terem consequências positivas da sua implementação permitem ajudar a um arquitecto resolver um problema de forma rápida não trazendo consequências negativas. Os positivos passam imediatamente a padrões, enquanto os padrões classificados como negativos à partida são anti-padrões. Um arquitecto tem a liberdade de classificar um padrão com um maior número de consequências negativas do que positivas como padrão se tiver uma justificação plausivél como por exemplo, mesmo que as consequências negativas sejam em maior número que as consequências positivas, esses aspectos negativos podem não afectar o sistema de uma forma muito grave.

As consequências dizem respeito os maus resultados que a aplicação desse padrão pode trazer a

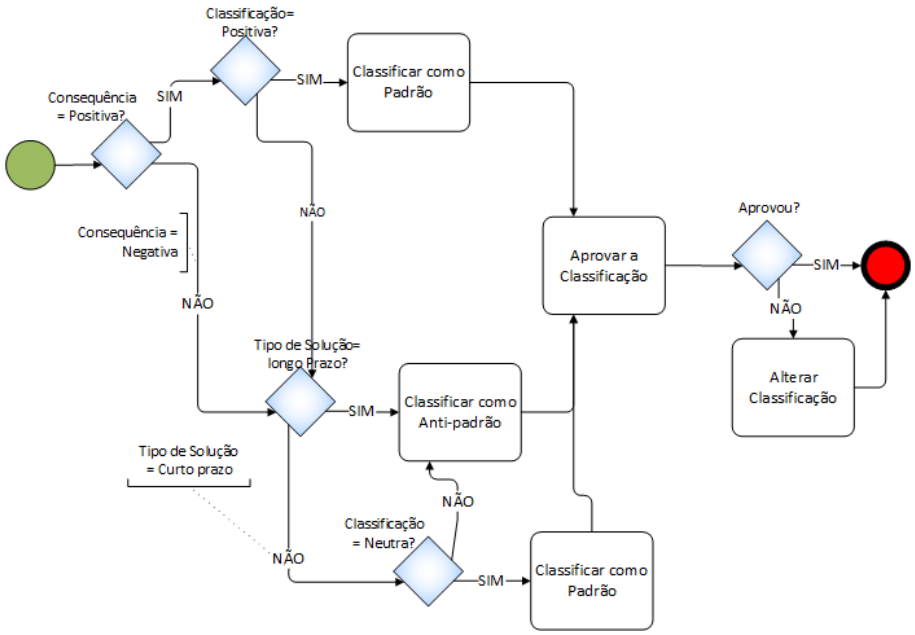


Figura 4.9: Fluxo classificação do padrão

curto e a longo prazo à organização. Esses resultados são obtidos através da documentação de cada arquitectura, caso contrário apenas tendo contacto com as pessoas que implementaram a arquitectura é que temos o feedback sobre as consequências da aplicação desse padrão. Na *figura* 4.9 é possível ver o fluxo da classificação do padrão de acordo com as suas consequências.

4.4 Documentação - 4ª Fase

Documentar padrões e anti-padrões é necessário para transmitir o conhecimento já adquirido e consumado por membros de uma organização. Ao catalogar os padrões e anti-padrões é-nos possível o uso do conhecimento, conseguindo resultados positivos de forma rápida ou conseguindo ter noção dos resultados negativos esperados. A documentação dos padrões deve seguir uma certa estrutura de forma a não perder informação. A catalogação dos padrões terá a seguinte estrutura:

Classificação do padrão(Obrigatório): este campo tem que ser preenchido consoante a classificação atribuída a este padrão na fase de Padrões *out of the box*.

Padrão/Anti-Padrões(Obrigatório): neste campo é indicado se é padrão ou anti-padrão.

Nome do Padrão (Obrigatório): este campo tem que ser preenchido com um nome sugestivo para que seja fácil de memorizar e de identificar o padrão estando relacionado com a funcionalidade disponibilizada.

Problema (Obrigatório): é necessário indicar qual é o problema que a arquitectura resolve. Aqui também pode ser listado os requisitos que a arquitectura implementa e restrições a ter em consideração na aplicação do padrão.

Contexto (Obrigatório): lista situações onde o problema ocorre, oferecendo ao arquitecto exemplos de ambiente onde o padrão poderá ser aplicado.

Consequências (Obrigatório): descrever as consequências positivas e negativas que a aplicação desse padrão pode gerar.

Solução (Obrigatório): Descrever os elementos que constituem o padrão ou anti-padrão, indicando as suas relações e responsabilidades.

Arquitectura genérica (Obrigatório): Criar um esboço arquitectural genérico do padrão ou do anti-padrão tendo em atenção as funcionalidades, componentes e atributos.

Tipo de arquitectura empresarial (Obrigatório): Identificar em que tipo de arquitectura o padrão ou anti-padrão é aplicado. Identificando a arquitectura como organizacional, negócio, informacional, aplicacional e/ou tecnológica.

Principio Arquitectural (Opcional): descrição do princípio arquitectural da empresa que pode existir no padrão ou anti-padrão. Esta informação é proveniente da documentação adquirida pelas arquitecturas.

4.5 Resumo

Para a descoberta de padrões arquitecturais é necessário passar por diversas etapas. Primeiramente é calculada a similaridade entre os objectos das arquitecturas classificando os objectos como similares caso exista similaridade entre ambas arquitecturas. Após a detecção dessa similaridade é verificado se os objectos similares têm relações com outros objectos similares.

Logo que detectar-mos os objectos similares e os classificamos como padrão candidato é necessário escolher entre eles os que a primeira vista parecem similares classificando-os desta forma como padrões *out of the box*. Não é um passo obrigatório escolher quais dos pares dos padrões candidatos são padrões *out of the box*, isto porque nos passos seguintes é analisado a similaridade dos objectos que compões os padrões *out of the box* mas ao nível das funcionalidades.

Depois de ter-mos os padrões *out of the box* é necessário avaliar a similaridade ao nível das funcionalidades, para tal é necessário analisar as funcionalidades dos pares dos objectos que são similares. Quando analisamos as funcionalidades consegue-se detectar os objectos que são similares entre si, essa similaridade não depende apenas da similaridade entre nome, o tipo de objecto e o tipo de camada. Classificando-o desta forma como padrão arquitectural.

Após a detecção dos padrões arquitecturais é necessário verificar se o padrão arquitectural descoberto já existe, caso não exista temos que o documentar. Para verificar se o padrão arquitectural existe é utilizado a AR que contém as funcionalidades dos objectos que à constitui. Sendo necessário fazer *matching* com os objectos do padrão arquitectural, caso o *matching* entre o padrão arquitectural e o *hotspot* da AR coincida significa que o padrão arquitectural já existe.

Como já foi referido, se é um novo padrão arquitectural é necessário prosseguir a sua documentação caso o padrão arquitectural já exista é necessário confirmar se o padrão encontra-se documentado. Para documentar os padrões é necessário preencher uma estrutura já definida cobrindo muitos aspectos da arquitectura. A documentação dos padrões é necessária para a transmissão de conhecimento podendo ser essencial para a precaução de problemas. Antes da documentação dos padrões arquitec-

turais é necessário classifica-los como padrão ou anti-padrão tendo em consideração as consequências que a sua implementação trará a curto ou a longo prazo. Esta classificação depende da documentação existente sobre as arquitecturas em análise.

Capítulo 5

Demonstração

A metodologia utilizada nesta investigação é *Design Science Research* (DSR), como foi indicado na introdução. A demonstração prática corresponde a fase 3 desta metodologia de investigação. Esta fase da metodologia corresponde ao desenho e ao desenvolvimento do artefacto, sendo necessário passar pelas fases anteriores. Na fase 3 do DSR é necessário desenhar e desenvolver um artefacto. Na *figura* 5.1 é possível visualizar a arquitectural funcional da solução.

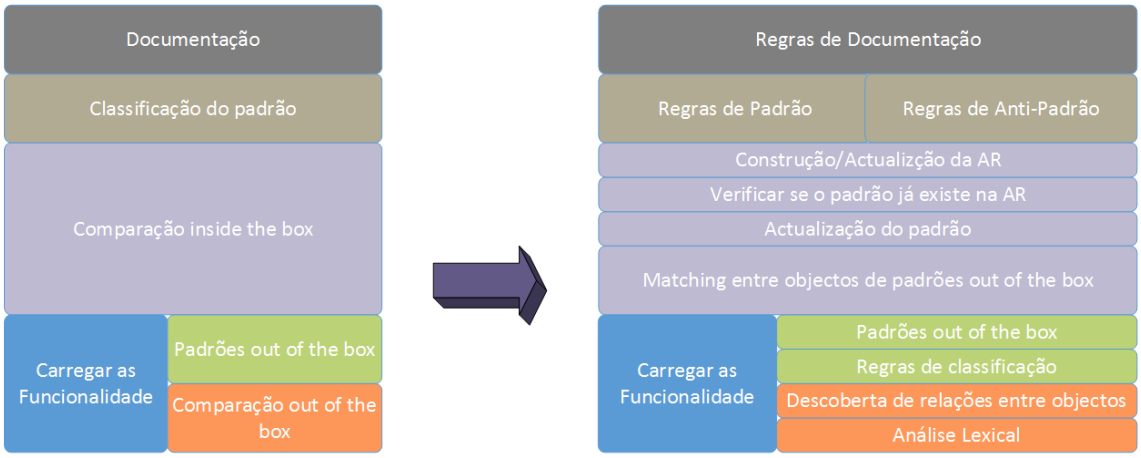


Figura 5.1: Arquitectura RCGD

A *figura* 5.1 mostra a arquitectura RCGD em que é possível visualizar os seis módulos que a compõe.

- O módulo "Carregar as funcionalidades" consiste no levantamento das funcionalidades dos objectos que constituem a arquitectura em análise. Este levantamento é utilizado para a detecção de padrões *inside the box*. Na demonstração prática este módulo encontra-se representado na fase dos "padrões *out of the box*" onde é realizado o levantamento das funcionalidades.
- O módulo "Comparação *out of the box*" consiste na realização da análise lexical às arquitecturas onde definimos que informação é necessário guardar para comparar os objectos. Neste módulo também é calculado a similaridade entre os objectos de forma a detectar quais são os pares de objectos similares.

- O módulo “Padrões *out of the box*” consiste em analisar os padrões candidatos e de acordo com as regras de denominação definidas na subfase de “Padrões *out of the box*” classifica-los como complexo específico ou simples específico. Na demonstração prática este modulo encontra-se representado na subfase “padrões *out of the box*” na fase de comparação das arquitecturas.

- O módulo “Comparação *inside the box*” consiste num módulo onde é realizado uma análise as funcionalidades dos objectos que constituem os padrões *out of the box*, de forma a poder classificar os padrões *out of the box* como padrões arquitecturais. É neste módulo que também a construção/ actualização da Arquitectura de Referência é efectuada. Na actualização da AR é utilizado os padrões arquitecturais encontrados, sendo necessário verificar se os padrões arquitecturais encontrados já se encontram representados na AR. Na demonstração prática este módulo encontra-se representado na subfase “Comparação *inside the box*” na fase de comparação das arquitecturas.

- O módulo “Classificação do Padrão” consiste em analisar as consequências dos objectos que constituem o padrão arquitectural e classificá-lo de acordo com suas consequências. A classificação do padrão arquitectural consiste em identificar se é padrão ou anti-padrão. Na demonstração prática este módulo encontra-se representado na fase da “Geração”.

- O módulo “Documentação” consiste em documentar os padrões arquitecturais de acordo com uma estrutura já definida. Na demonstração prática este módulo encontra-se representado na fase da documentação.

As arquitecturas que serviram como exemplo para a implementação encontram-se em ArchiMate e foram retiradas do [28] , sendo a *posteriori* desenhadas em XML [1] respeitando o metamodelo do ArchiMate [2] desenhado em XML.

Esta é a fase da demonstração onde irá ser realizada uma simulação com arquitecturas empresariais onde temos detectar pares de objectos similares e gerar os padrões e anti-padrões. Após a detecção dos padrões arquitecturais temos como a própria metodologia indica, catalogar os padrões encontrados.

5.1 Análise das Arquitecturas - 1ª Fase

Como foi referido na proposta de solução, nesta fase é necessário realizar o levantamento do problema e do contexto das arquitecturas em análise. As arquitecturas apenas serão seleccionadas se o problema e o contexto forem semelhante.

Para testar a solução ire-mos utilizar duas arquitecturas que tenham um problema e um contexto semelhante. É de salientar que a entidade em questão é uma universidade. O objectivo global das arquitecturas em análise consistem em actualizar o sistema utilizando um sistema de *smartcard* tendo como foco principal actualizar registar informação dos estudantes e funcionários. As arquitecturas

[1] ver as arquitecturas desenhadas em XML hhttps://www.dropbox.com/sh/8mksfccww6qw25f/AABE4DV6otuPgBAUCNtFThsXa?dl=0

[2] ver metamodelo de ArchiMate desenhado em XML https://www.dropbox.com/sh/htdw1rfruvna5ar/AAB34uzxXeiMOB4ikdYXxW79a?dl=0, fonte AMA

utilizadas pertencem a mesma universidade sendo as linhas de negócios diferentes mas o problema e o contexto são semelhante.

Arquitectura 1:

Problema: permitir aos estudantes e funcionários aceder aos edifícios e instalações do centro de desporto da universidade utilizando o cartão de identificação.

Contexto: Este problema insere-se num contexto universitário onde existe actores que pretendem usufruir das instalações do centro de desporto da universidade.

Arquitectura 2:

Problema: permite o uso do cartão de identificação dos actores dentro da biblioteca. O actor tem de ser um estudante matriculado em curso e ter um cartão de identificação do aluno para ser capaz de requisitar livros. E o funcionário tem que ajudar o aluno neste processo e também é capaz de requisitar livros.

Contexto: Este problema insere-se num contexto universitário onde existe actores que pretendem realizar acções na biblioteca.

Resumindo, o problema refere-se a um sistema de controlo de acessos no contexto universitário. Em que o contexto principal ocorre numa universidade quando pretende-se usufruir de um serviço. Por estas razões é que estas arquitecturas foram escolhidas para análise, por terem um problema e contexto semelhante. Ambas as arquitecturas e toda a informação existente foram extraídas da documentação [28].

O levantamento das funcionalidades poderia ser feito a todos os objectos que constituem a arquitectura ou apenas aos objectos que compõe os padrões *out of the box*. Nos optamos por realizar o levantamento das funcionalidades aos objectos que pertencem aos padrões *out of the box*. Sendo que esse levantamento é realizado apenas na 2ª fase da RCGD mais especificamente na subfase padrões *out of the box*. Nesta etapa classificamos os padrões candidatos como *out of the box* tal como foi indicado na proposta de solução. A escolha de realizar o levantamento das funcionalidades nesta etapa deve-se ao facto de ser neste momento que classificamos o padrão *out of the box* parecendo oportuno realizar este levantamento aos objectos que constituem o padrão *out of the box*. Irá ser definida a estrutura em que as funcionalidades de cada objecto irão ser armazenadas. Salientamos novamente que o levantamento das funcionalidade pode ser realizada em qualquer etapa desde que seja realizada antes da fase comparação *inside the box*.

5.2 Comparar Arquitecturas - 2ª Fase

5.2.1 Comparação *out of the box*

Nesta fase é desenvolvido um programa em C# para implementar a 2ª Fase do método RCGD. Este programa apenas irá ter interface para que o utilizador possa carregar as arquitecturas no sistema para uma análise posterior. O carregamento das arquitecturas no sistema será realizado através da selecção de ficheiros XML que contém a arquitectura desenhada de acordo com o metamodelo em

XML do ArchiMate. É gerado um ficheiro XML como output que contém os padrões candidatos que poderão ser classificados como padrões *out of the box*. A estrutura do ficheiro XML é especificada mais a frente. Numa fase intermédia é gerada um ficheiro XML que contém os pares de objectos similares sem as suas relações, ficheiro esse que é utilizado para detectar as relações similares entre objectos similares.

Análise Lexical

Atendendo que existe a necessidade de realizar uma análise as arquitecturas é necessário responder a seguinte questão *"O que é necessário que se verifique entre dois objectos para que a similaridade entre ambos seja calculada?"*. Tendo em consideração que as arquitecturas foram modeladas em ArchiMate, dois objectos podem ser comparados se pertencerem à mesma camada e se forem do mesmo tipo de objecto. Objectos de tipos diferentes não podem ser comparados, o mesmo acontece se os objectos forem de camadas diferentes.

Durante o *scanning* realizado à arquitectura é necessário guardar à seguinte informação: id do objecto, tipo de camada do objecto, tipo de objecto, nome do objecto, o tipo de relações que tem os objectos e o nome dos respectivos objectos com quem se relacionam.

As arquitecturas foram rescritas em XML[3] de forma a poder permitir extrair informação relevante. Foi realizado um *scanning* às arquitecturas escritas em XML de forma a extrair e armazenar informação que foi anteriormente indicada. A informação de cada arquitectura foi armazenada em ficheiros separados seguindo o seguinte diagrama de classes, ver *figura* 5.2:

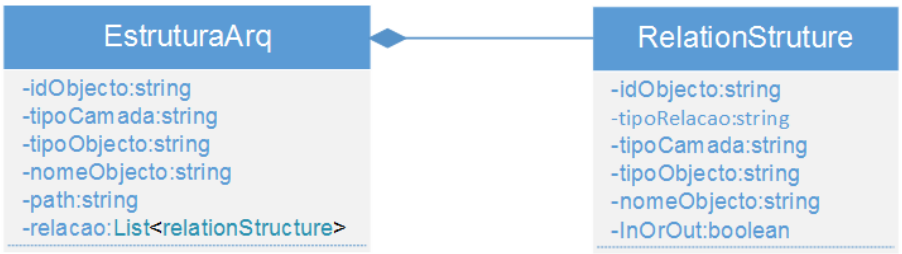
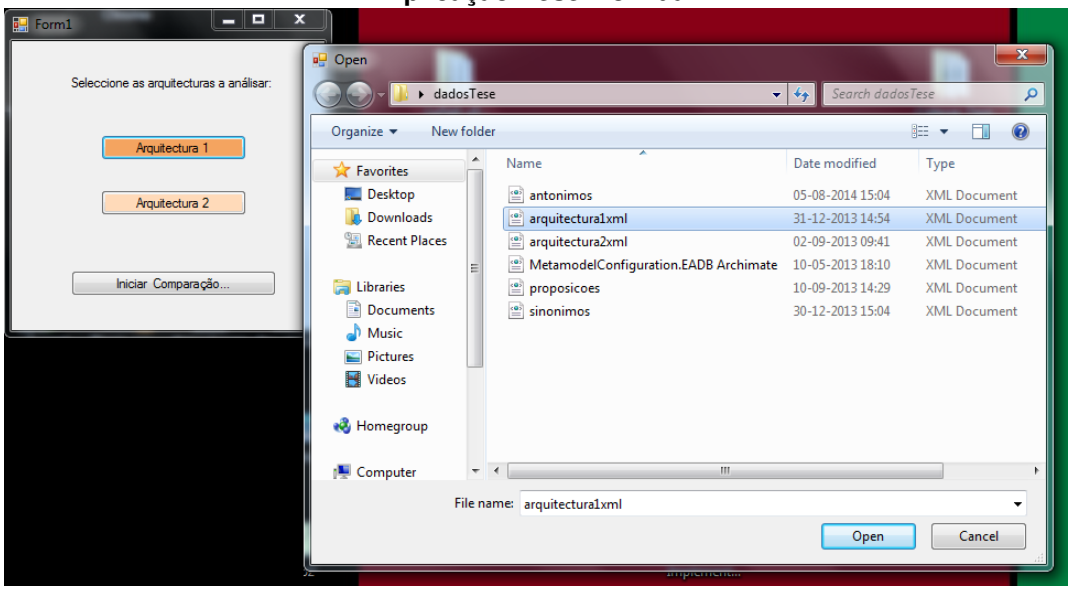


Figura 5.2: Diagrama de classes da estrutura da informação guardada

O problema encontrado foi garantir que os dados guardados correspondiam a toda a informação necessária para que a detecção de padrões pudesse ser realizada.

[3]ver as arquitecturas desenhadas em XML https://www.dropbox.com/sh/8mksfccww6qw25f/AABE4DV6otuPgBAUCNtFThsXa?dl=0

Aplicação Desenvolvida



EX1. Aplicação desenvolvida – Carregamento das arquitecturas

Na *figura* EX1, é possível verificar de que forma são carregadas as arquitecturas no sistema para que a comparação entre ambas possa ser executada. Durante o cálculo da similaridade é necessário verificar se alguma palavra é preposição, conjunção, sinónimo ou antónimo. Estes quatro tipos de categorizações são guardados em ficheiros separados. Para a sua utilização é necessário que esses ficheiros estejam colocados numa pasta no ambiente de trabalho denominada por *dadosTese*. A definição da estrutura de XML de cada ficheiro irá ser indicada mais a frente.

Descoberta da relação entre os objectos

Nesta fase para o cálculo da similaridade entre dois objectos utilizou-se o algoritmo *edit distance* [29].

Para a descoberta da relação entre dois objectos seguiu-se o seguinte fluxo, ver *figura* 5.3. Na *figura* 5.3 é detalhado o fluxo para a descoberta da similaridade entre dois objectos. O programa começa por analisar pares de objectos entre as ambas arquitecturas, isto quer dizer, que é escolhido um objecto de cada arquitectura e é realizada a comparação entre ambos os objectos tendo em consideração que todas as combinações possíveis entre os nomes que as compões têm que ser realizadas. Após ter-mos dois objectos seleccionados, dá-se a comparação aos pares entre as palavras que constituem ambos os objectos. Isto é, cada par é constituído por uma palavra de cada objecto de forma a realizar todas as combinações possíveis entre as palavras dos dois objectos.

Na *figura* 5.4 é-nos mostrado um exemplo de dois objectos de arquitecturas diferentes que estão sujeitos a comparação para detectar a similaridade entre ambos. O objecto 1 é "*Maintain Staff sudent Information*" e o objecto 2 é "*Update staff and student Information*. Neste exemplo pode-se verificar que é realizada todas as combinações possíveis para detectar a similaridade. Para detectar a similaridade é necessário passar por um conjunto de etapas, ver *figura* 5.3. Na *figura* 5.3 o início do fluxo passa pela escolha de uma palavra de cada um dos objectos e de seguida é verificado se alguma das

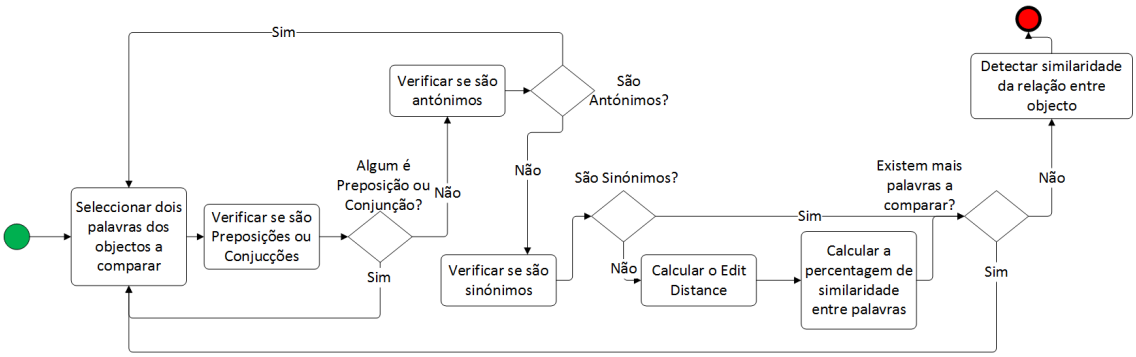


Figura 5.3: Fluxo para a descoberta da relação entre dois objectos

palavras seleccionadas é uma preposição ou conjunção. Caso alguma das palavras seja conjunção ou preposição não é calculado a similaridade, seleccionando outra combinação de palavras. Existe 2 formas de proceder quando é detectado que pelo menos uma das palavras é conjunção ou preposição. Passo 1: As duas palavras em análise são preposições ou conjunções? Se sim, então escolhemos outra combinação de palavras. Passo 2: A palavra da arquitectura 2 é preposição ou conjunção? Se sim, significa que existirá outra combinação entre a palavra do objecto da arquitectura 1 com outra palavra do objecto da arquitectura 2. Se a resposta a pergunta for não, significa que a preposição esta do lado da palavra da arquitectura 1 e tem que ser escolhida outra combinação. Se não se conferir que são preposições ou conjunções, verifica-se se as palavras são antónimos, caso isso se verifique então é novamente escolhida outra combinação de palavras. Caso sejam sinónimas não se dá o cálculo da similaridade pois estas têm similaridade 100%. Caso não sejam antónimos nem sinónimas dá-se o cálculo de similaridade através do algoritmo *edit distance* [29]. O fluxo é inicializado sempre que exista uma nova combinação de palavras que necessitar de ser avaliada quanto a sua similaridade. É necessário relembrar que é necessário realizar o calculo da similaridade a todas as combinações possíveis.

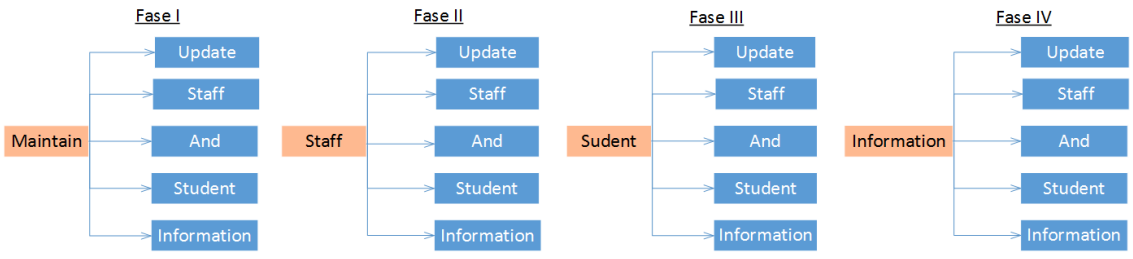


Figura 5.4: Combinações na comparação de dois objectos

Para calcular a similaridade entre dois objectos temos que verificar a similaridade entre as palavras que constituem os objectos. Só depois é que é calculado a percentagem de similaridade entre os objectos. Na *figura* 5.5 é possível verificar que tipo de análise é necessário realizar antes de calcular a similaridade entre os objectos. O exemplo aqui exemplificado pertence as arquitecturas em análise, mais especificamente a dois objectos da camada "Aplication service". Na *figura* 5.5 verifica-se as combinações entre dois objectos, Objecto 1: *Maintain Staff sudent Information* e Objecto 2: *Update*

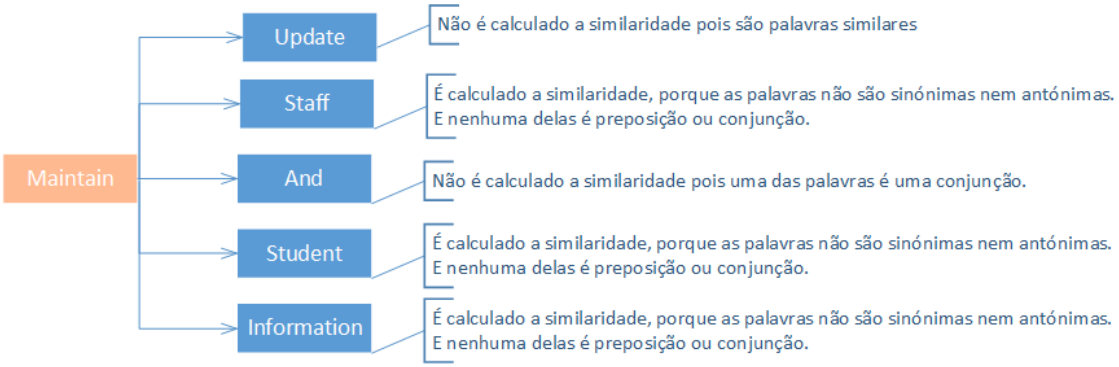


Figura 5.5: Análise realizada durante a comparação de dois objectos

staff and student Information.

1- Verificar se é preposição ou conjunção

Foi listado em XML um conjunto de preposições e conjunções existentes, ver *anexo E.1 Preposições e Conjunções*. A lista de preposições e conjunções encontram-se em inglês pois as arquitecturas encontram-se nessa língua. É necessário salientar que as preposições e as conjunções ficaram listadas no mesmo ficheiro pois não vimos nenhuma vantagem em ter essa informação em ficheiros separados uma vez que a acção executada é a mesma quando descobrimos que uma palavra é ou não proposição ou conjunção. É a partir desse ficheiro que é verificado se alguma das palavras que constitui os objectos é uma preposição ou conjunção. Caso a palavra em análise exista na lista, então não é calculada a similaridade, ver tabela 5.5 gerando desta forma outra combinação para a análise. Na tabela 5.5 tanto no objecto 1 como no objecto 2 temos uma preposição, essa preposição não entra no cálculo da similaridade entre objectos. A lista de preposições e conjunções tem que ser actualizada sempre que existir a necessidade. No *anexo E.1 Preposições e Conjunções* é possível verificar a estrutura em XML onde as preposições e as conjunções são listadas.

2- Verificar se são sinónimos ou antónimos

Foi criado um ficheiro em XML que contém uma lista de sinónimos e outro ficheiro com a lista de antónimos que se verificam nas arquitecturas, ver *anexso E.2 Sinónimos e E.3 Antónimos*. Por exemplo, em alguns casos os funcionário e os estudantes podem ser considerados sinónimos pois podem ter funcionalidades similares. Numa universidade tanto os estudantes como os funcionários podem usufruir dos serviços da livraria tal como dos servidos de desporto. Caso duas palavras sejam sinónimas ou antónimas, não é calculada a similaridade entre elas. Pois se forem sinónimas têm similaridade 100% caso sejam antónimas não existe similaridade (similaridade 0%). A lista de sinónimos e antónimos foram actualizadas a medida que fomos obtendo resultados, esta actualização serviu como forma de refinamento, diminuindo o número de pares similares como resultado. Existia uma probabilidade muito reduzida dos pares que foram eliminados automaticamente serem ao nível das funcionalidades similares. Nos *anexos E.2 Sinónimos e E.3 Antónimos* é possível verificar a estrutura em XML em que os sinónimos e antónimos das arquitecturas em análise são listadas.

3- Cálculo da similaridade Nesta etapa é calculada a percentagem de similaridade entre os objectos em análise. Para calcular a similaridade é utilizado um algoritmo, mais a frente com um exemplo

alusivo esse algoritmo será explicado.

Definição do cálculo da similaridade

Em primeiro lugar o cálculo da similaridade dá-se a partir dos nomes dos objectos. O nome de um objecto pode conter mais que uma palavra, como vimos anteriormente. Uma palavra pode ser uma preposição, conjunção, sinónimo, antónimo ou nenhuma das anteriores. Quando existe uma preposição, conjunção, sinónimos ou antónimos é atribuído similaridade 100%, 0% ou nem é calculado a similaridade tal como foi discutido anteriormente. Vamos agora definir de que forma é calculada a similaridade entre palavras quando nenhum dos casos anteriores é verificado.

Tal como é indicado na proposta de solução para cada combinação de palavras tem que ser calculada a sua similaridade, *figura* EX2.

O cálculo da similaridade dá-se através da utilização do algoritmo edit distance [29] entre duas palavras e a divisão do valor resultante desse algoritmo com o número de caracteres da maior palavra. Na *figura* EX2 é possível verificar um exemplo da aplicação da fórmula

$\frac{editDistance(st1,st2)}{bigCountChar}$, para o cálculo da similaridade. É relevante referir que se o valor calculado for inferior a 0.5 significa que as palavras são similares.

Palavra 1	Palavra 2	Edit Distance	editDistance(st1,st2) / bigCountChar
Hos	hospital	5	0.63
casa	casas	1	0.2
Chucha	espinho	6	0.86
lua	la	1	0.33
server	service	3	0.43
aleph	server	6	1
aleph	aleph	0	0
Server	server	0	0

EX2. Cálculo da similaridade

Após ter-mos calculado a similaridade entre as palavras é necessário calcular a percentagem de similaridade entre os objectos, isto porque o nome do objecto é constituído por um conjunto de palavras. O cálculo para obtermos a percentagem de similaridade entre dois objectos é a seguinte:

$\frac{contarpalavrassimilares(0.5)}{bigWordName}$, para este cálculo as preposições e conjunções não entram.

Após ter-mos calculado a similaridade entre cada palavra que constitui o nome do objecto é necessário fazer um balanço final, calculando a percentagem da similaridade.

Na *figura* EX3 pode-se verificar de que forma é calculado a percentagem de similaridade. Em primeiro lugar conta-se quantas palavras similares existem (similaridade inferior a 0.5) no objecto e depois é realizado divide-se esse valor pelo número total de palavras que o objecto em análise tem (porém apenas é considerado o objecto que contem maior número de palavras).

Artefacto 1	Artefacto 2	Contar Similaridade(<0,5)	Valor Total de similaridade
Aleph server	Aleph server	2	2/2 = 100%
Casa Chucha	Casas Hospital	1	½ = 50%
Lua casa server	La casa	1	1/3= 33%
Server	service	1	1/1 = 100%

EX3. Cálculo da similaridade

Os objectos são considerados similares se a percentagem de similaridade é igual ou superior a 50%. A linha assinalada a vermelho analisando a percentagem resultante é-nos indicado que existe similaridade. Contudo através da observação verifica-se que não o é. Portanto estas palavras podem ser classificadas como antónimas, desta forma numa outra análise realizada com outro par de arquitecturas estas palavras não serão classificadas como similares.

Desta forma constrói-se uma colecção de palavras que são antónimas, de forma que a longo prazo a detecção de similaridade seja mais eficaz. Antónimo neste caso não significa que têm significados contrários mas sim que não são similares.

Um dos problemas encontrado na descoberta de padrões foi chegar à conclusão de que não podíamos aplicar o algoritmo *edit distance* [29] directamente aos nomes dos objectos, tendo que ser aplicado a cada palavra que compõe o mesmo. E foi igualmente difícil deduzir que a percentagem de similaridade entre dois objectos era calculada pela divisão do número de palavras similares pelo número máximo de palavras entre os dois objectos.

Após a explicação de como o algoritmo irá funcionar, de definir a análise lexical e da forma como os dados irão ser armazenamos passamos a implementação do algoritmo em C#. A *figura* 5.6 é um diagrama de sequência do sistema implementado, em que o actor carrega as duas arquitecturas no sistema e após o carregamento das arquitecturas é analisado a similaridade entre o nome dos objectos. Sendo necessário primeiramente verificar se as palavras que constituem os objectos são sinónimos, antónimos, preposições e conjunções. Caso não seja nenhuma das anteriores dá-se o cálculo da similaridade utilizando o *edit distance*. Após a detecção de quais os pares de objectos que são similares é necessário calcular a percentagem de similaridade. Após obtermos os pares similares é necessário detectar quais dos objectos similares que tem uma relação com outros objectos similares. Posteriormente a ter-mos os objectos similares com e sem relações, o responsável por detectar padrões tem que escolher com o auxílio da documentação e com conhecimento da arquitectura quais são os que são classificados como padrão candidato.

De seguida vamos explicar os objectos existentes na *figura* 5.6.

- **No carregamento da Arquitectura**

Implementação

Foi utilizada a mesma estrutura da *figura* 5.2 para carregar as arquitecturas no sistema. O carregamento da arquitectura é inicializado pela leitura do ficheiro XML que contém a arquitectura estruturada e modelada em ArchiMate. As arquitecturas que foram transformadas em XML encontram-se no *anexos B.1 Arquitectura 1 e E.3 B.1 Arqui-*

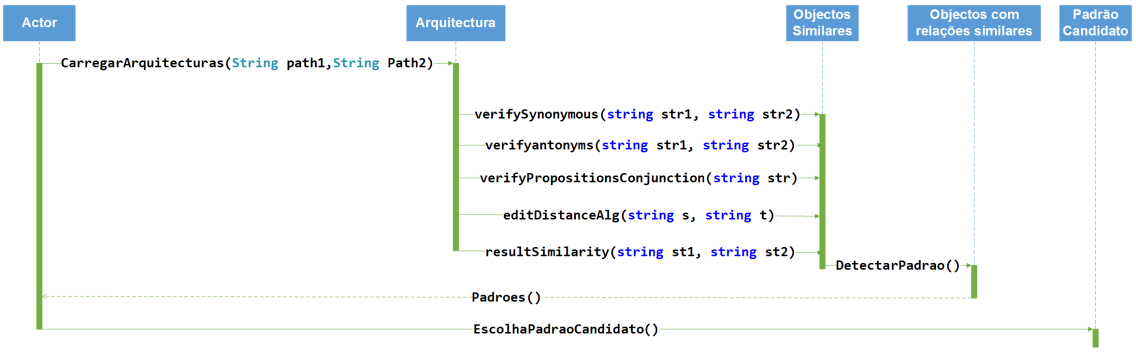


Figura 5.6: Diagrama de sequência

tectura 2. É possível visualizar as arquitecturas traduzidas em XML no seguinte endereço https://www.dropbox.com/sh/8mksfccww6qw25f/AABE4DV6otuPgBAUCNtFThsXa?dl=0. Cada arquitectura é armazenada no sistema na seguinte estrutura *List<EstruturasArq>cotentDataArchitecture*, que utiliza as classes definidas na *figura* 5.2.

Valor de entrada

Para obtermos o resultado num ficheiro XML que contenha informação inerente as arquitecturas em análise numa estrutura específica é necessário que as arquitecturas estejam escritas em XML. Assim sendo as arquitecturas desenhadas em XML corresponde ao valor de entrada no carregamento da arquitectura, como referido anteriormente.

Resultados

O resultado nesta etapa corresponde simplesmente ao carregamento das arquitecturas no sistema e o armazenamento da informação pertinente das arquitecturas num ficheiro XML numa estrutura anteriormente já definida.

- **Objectos Similares**

Implementação

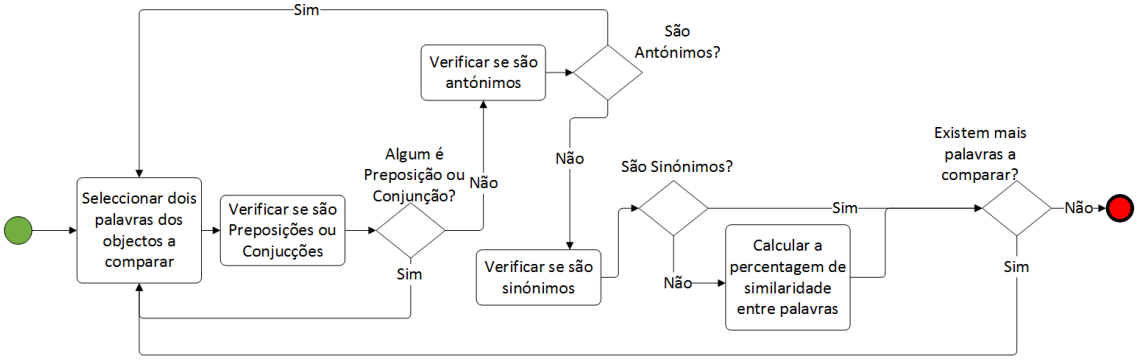


Figura 5.7: Fluxo de detecção de objectos similares

A *figura* 5.7 é muito semelhante a *figura* 5.3 sendo que a diferença é a especificidade da aplicação do algoritmo para o calculo da similaridade. Nesta etapa detectamos quais os objectos similares utilizando técnicas anteriormente descritas no exemplo da "definição do cálculo da similaridade". O artefacto gerado no cálculo da similaridade é um ficheiro XML onde é composto por objectos similares.

A *posteriori* este ficheiro que foi gerado é utilizado para a descoberta das relações entre objectos similares. A estrutura XML onde é guardado os objectos similares é a seguinte ver *anexo C.1 Estrutura XML guardar dados similares*. Durante a implementação foi necessário actualizar a lista de sinónimos e antónimos, desta forma fomos refinando o algoritmo.

Lista de antónimos introduzidos:

- *Server* é antónimo de *service*.
- *Service* é antónimo de *service*.

Justificação: *server* e *sevice* foram denominados como antónimos pois como eram classificados como similares tomou-se a decisão de classifica-los como antónimos para optimizar a detecção de novos objectos similares. Dizemos que *sevice* é antónimo de *sevice* pois notou-se que nas arquitecturas era comum a utilização da palavra *sevice* para caracterizar componentes que facultam serviços. Assim sendo, a maior parte dos componentes caracterizado como semelhante não o era, existindo a necessidade de eliminar a existência dessa similaridade.

Lista de sinónimos introduzidos:

- *Update* é sinónimo de *maintain*.
- *Student* é sinónimo de *staff*.

Justificação: *Update* e *maintain* foram classificados como sinónimos porque durante a análise de ambas as arquitecturas deparamo-nos que actualizar e manter informação tinham o mesmo significado. *Student* e staff foram classificados como similares pois o papel destes actores é similar em muitos casos.

É de salientar que esta lista foi actualizada durante o desenvolvimento da solução e também durante a sua aplicação na descoberta de objectos similares. Esta actualização teve como objectivo optimizar a pesquisa de padrões. A lista de sinónimos e antónimos deve ser sempre actualizada durante a análise das arquitecturas, tal como foi indicado anteriormente.

Valor de entrada

Para a detecção da similaridade entre os objectos é utilizado um ficheiro XML com informação extraída das arquitecturas e é a partir desse ficheiro que é analisado objecto a objecto realizando todas as combinações possíveis entre os objectos. Estas combinações servem para detectar a similaridade.

Resultados

Após a execução do programa é gerado um ficheiro XML com os objectos similares. É possível verificar a lista dos objectos similares na *figura* 5.8 onde é possível ver a percentagem de similaridade entre os objectos. Nesta lista apenas consta os objectos com similaridade igual ou superior a 50%. O valor da similaridade como anteriormente foi referido é calculado utilizando a fórmula

$\frac{editDistance(st1,st2)}{bigCountChar}$.

Após obtermos a lista dos objectos com percentagem de similaridade igual ou superior a 50%, *figura* 5.8 temos que analisar os pares de objectos e seleccionar quais são os que passam a próxima etapa,

Identificação	Objecto Arquitectura 1	Objecto Arquitectura 1	Valor similaridade	
1	ALEPH server	SQL server	50%	
2	Student information	Staff student information	100%	
3	Student information	Student personal information	66%	
4	Student information	Student accommodation information	66%	
5	universe	Universe	100%	
6	Maintain staff student information	Update staff and student information	100%	
7	student	Student	100%	
8	student	Staff	100%	
9	staff	Student	100%	
10	staff	staff	100%	

Figura 5.8: Resultado da similaridade dos objectos

onde serão analisados relações similares entre objectos similares. Na *figura* 5.8 pode-se verificar que o par de objectos que passa à próxima fase são os que estão assinalados a verde. Os que se encontram a vermelho são os que foram excluídos. A combinação número 1 não foi seleccionada pois a palavra Aleph e SQL não tem qualquer similaridade lexical. As combinações 3 e 4 não foram seleccionadas para a próxima fase pois existe uma combinação mais forte com o *student information* que é a combinação 2. Esta combinação é mais forte pois não existem palavras como *accommodation* ou *personal* sendo esta análise é meramente lexical. As combinações 8 e 9 foram excluídas pela mesma causa. O *staff* e o *student* são sinónimos porque esta relação foi colocada na lista de sinónimos, porém como em ambas as arquitecturas existe um estudante e um funcionário não há a necessidade de indicar que estes actores são sinónimos, pois ambos os actores estão já representados em outras combinações.

As combinações que passam a próxima fase deve-se a uma análise lexical e também ao bom senso, por exemplo: temos a combinação 7, 8, 9 e 10 e sabemos que segundo a lista de sinónimos o estudante e o funcionário são similares. Porém temos a indicação que o objecto estudante da arquitectura 1 e 2 são similares assim como os funcionários em ambas as arquitecturas. Não faz sentido neste caso acrescentar a combinação que diga que o funcionário e o estudante são similares. Pois a *posteriori* ire-mos ter que generalizar o nome do padrão e essa generalização não trará informação adicional, pois já temos o estudante e o funcionário como padrão.

- **Objectos com relações similares**

Implementação

Nesta etapa pretende-se que detectar quais são os objectos similares que têm relações com outros objectos similares. Para verificar se existe este tipo de relação é utilizado o ficheiro XML com os objectos similares e o ficheiro XML com as arquitecturas desenhadas em XML.

No ficheiro que contém as arquitecturas é possível verificar quais são as relações que existem com os objectos similares ou não. Com o ficheiro que contém os objectos similares é analisado se os objectos com quem o objecto similar tem relação também é um objecto similar. Após a detecção de

padrões simples e complexos é gerado um ficheiro XML com a identificação desses padrões. Estes padrões descobertos estão sujeitos a classificação como padrão candidatos.

Para detectar objecto similares que tenham relações com outros objectos é necessário aplicar regras para a sua detecção. A regra como já foi referida na proposta de solução é a seguinte: Se A1 tem R1 com B1 e A2 tem R2 com B2 e A1.tipoObjecto = A2.tipoObjecto e A1.TipoCamada = A2.TipoCamada e R1.tipo=R2.tipo e B1.tipoObjecto = B2.tipoObject e B1.TipoCamada = B2.TipoCamadao e B1.nome = B2.nome e A1.nome = A2.nome Então tem uma relação similar

Valor de entrada

Para detectar-mos as relações existente entre os objectos similares é necessário três ficheiros. Dois ficheiros com as arquitecturas em XML e outro com a lista dos objectos similares, estes três ficheiros fazem parte do valor de entrada para este passo.

Resultados

Como resultado desta etapa temos um ficheiro XML que contém tanto objectos com relações como objectos isolados, como é possível verificar no *anexo D.1 Objectos similares*.

Vamos agora exemplificar de que forma foi aplicada a regra utilizada nesta etapa. Com esta regra consegui-mos relacionar quais dos objectos similares entre ambas as arquitecturas tem relações igualmente semelhantes com outros objectos igualmente semelhantes. A *figura* 5.9 é um reflexo da aplicação da regra, onde consegui-mos observar as relações entre ambas as arquitecturas sem ser apenas com um objecto.

Para exemplificar a aplicação da regra ire-mos utilizar o padrão número 1 da *figura* 5.9. Observemos agora a aplicação directa da regra que nos auxilia a identificar quais os objectos similares que tem relações com outros objectos. A aplicação da regra é a seguinte:

- Detectar relação 1 – Relação com o objecto 1

Se **A1** (Universo) tem **R1** (Accesses) com **B1** (Student Information) e**A2** (Universo) tem **R2** (Accesses) com **B2** (Staff Student Information) e **A1.tipoObjecto** (Application Component) = **A2.tipo - Objecto** (Application Component) e **A1.TipoCamada** (Application Architecture) = **A2.TipoCamada** (Application Architecture) e **R1.tipo** (Accesses) = **R2.tipo** (Accesses) e **B1.tipoObjecto** (Data Object) = **B2.tipo - Objecto** (Data Object) e **B1.TipoCamada** (Business Architecture) = **B2.TipoCamada** (Business Architecture) e **B1.nome** (Student Information)= **B2.nome** (Staff Student Information) e **A1.nome** (Universe) = **A2.nome** (Universe) Então tem uma relação similar

- Detectar relação 1 – Relação com o objecto 2

Se **A1** (Universo) tem **R1** (Realizes) com **B1** (Maintain staff student information) e **A2** (Universo) tem **R2** (Realizes) com **B2** (update staff and student information) e **A1.tipoObjecto** (Application Component) = **A2.tipoObjecto** (Application Component) e **A1.TipoCamada** (Application Architecture) = **A2.TipoCamada** (Application Architecture)e **R1.tipo**(Realizes) = **R2.tipo** (Realizes) e **B1.tipoObjecto** (Application Service) = **B2.tipoObjecto** (Application Service) e **B1.TipoCamada** (Infrastructure Architecture) = **B2.TipoCamada** (Infrastructure Architecture) e **B1.nome** (Maintain staff student information)=

B2.nome (update staff and student information) e **A1.nome** (Universe) = **A2.nome** (Universe) Então tem uma relação similar

Esta regra foi aplicada aos objectos similares para detectar se esses objectos têm relações com outros objectos igualmente similares. Mais uma vez referencio que a demonstração desta regra apenas centrou-se o padrão número 1 da *figura* 5.9 para exemplificar a aplicação desta regra.

- **Padrão candidato**

Implementação

Atendendo que podemos classificar como padrão candidato os objectos que tenham ou não relações com outros objectos similares, a pessoa responsável por descobrir padrões tem que escolher quais desses objectos são classificados como tal, passando desta forma a próxima fase. Na próxima fase o padrão candidato pode ser classificado como padrão *out of the box*. Nesta etapa é escolhido quais os pares de objectos similares referente a *figura* 5.8 que são classificados como padrões candidatos. A classificação como padrão candidato dos artefactos e as suas relações está sob à responsabilidade da pessoa a quem foi atribuída essa responsabilidade. Esta classificação apenas avalia a análise lexical resultante à lista de artefactos similares.

Valor de entrada

O ficheiro gerado na etapa "objectos com relações similares" é utilizado nesta fase para a classificação dos padrões candidatos.

Resultado

A *figura* 5.9 é o resumo do resultado dos passos para o calculo da similaridade. Na *figura* 5.9 existe três padrões candidatos resultante dos passos anteriormente executados. O padrão candidato número 1, é classificado como complexo enquanto os outros dois são padrões candidatos simples. Na figura *figura* 5.9 constam os padrões candidatos escolhidos da lista de objectos similares referente a *figura* 5.8.

5.2.2 Padrões *out of the box*

Após a análise lexical dos três padrões existentes na *figura* 5.9 classifica-mos estes três padrões como padrões *out of the box*. Esta classificação deve-se a análise lexical pois aparentam ser objectos similares. Para ter-mos certeza de que o são é necessário realizar uma análise *inside the box*. Após ter-mos os padrões *out of the box* escolhidos é necessário classificar os padrões encontrados utilizando as regras de denominação definidas na proposta de solução, mais especificamente na subfase padrões *out of the box*. Na *figura* 5.10 é classificado os padrões encontrados. O padrão *out of the box* 1 é classificado como Complexo específico pois este padrão é constituído por mais do que um objecto sendo que estes têm uma denominação específica. O padrão *out of the box* 2 e 3 denomina-se por padrão simples específico pois apenas são constituídos por um objecto sendo específico.

Como foi indicado na análise lexical durante a análise das arquitecturas o levantamento das funcionalidades irá ser realizado nesta etapa.

Nº	Objectos		Objectos com quem se relacionam	
	Objecto 1	Objecto 2	Relação com o Objecto 1	Relação com o Objecto 2
1	Application Architecture ApplicationComponent universe	ApplicationArchitecture ApplicationComponent universe	• Accesses • Business Architecture • DataObject • student information	• Accesses • Business Architecture • DataObject • staff student information
			• Realizes • Infrastructure Architecture • ApplicationService • maintain staff student information	• Realizes • Infrastructure Architecture • ApplicationService • update staff and student information
2	Business Architecture BusinessActor Student	Business Architecture BusinessActor Student		
3	Business Architecture BusinessActor Staff	Business Architecture BusinessActor staff		

Figura 5.9: Lista de Padrões Candidatos

Padrão Candidato	Classificação
Padrão candidato 1	Complexo Especifico (CE)
Padrão candidato 2	Simples Especifico (SE)
Padrão candidato 3	Simples Especifico (SE)

Figura 5.10: Classificação dos padrões *out of the box*

- Levantamento das funcionalidades de cada objecto

A *figura* 5.11 é baseada na *figura* 5.9 que contém informação dos padrões *out of the box*. Esta informação é utilizada na próxima fase podendo classificar os padrões *out of the box* como padrões arquitecturais.

Na *figura* 5.11 é possível analisar o papel dos objecto que constituem o padrão *out of the box* de cada arquitectura. Para o levantamento das funcionalidades foi utilizado a documentação das arquitecturas. Este levantamento das funcionalidades irá ser utilizado na classificação de um padrão *out of the box* em padrão arquitectural.

5.2.3 Comparação *inside the box*

Matching entre objectos dos padrões *out of the box*

Vamos passar a fase de comparar as funcionalidades entre os padrões *out of the box* de forma a validar se o padrão gerado pelos padrões *out of the box* é um padrão arquitectural, para tal vamos analisar a *figura* 5.11. A *figura* 5.11 contém os padrões *out of the box* que correspondem a similaridade entre dois ou mais objectos de arquitecturas diferentes e também contém as funcionalidades dos respectivos padrões.

Objectos Arquitectura 1 (Sport)	Objectos Arquitectura 2 (Biblioteca)
Universe: Sistema de registo dos alunos e dos funcionários que fornece tanto informação do estudante como dos funcionários, mediante solicitação de cada um dos sistemas comerciais.	**Universe**: Sistema de registo dos alunos e dos funcionários.
Student Information: São registos de um estudante.	**Staff Student Information:** São registos de um estudante e do staff.
Maintain Staff Student Information: - Preenchimento de formulário para aderir aos centros de desporto universitários. - Actualizar a informação pessoal. - Actualizar o sistema Scuba.	**Update staff and student Information**: - Actualizar informação sobre o staffe os estudantes. - Actualizar a informação pessoal. - Registo da requisição de livros. - Actualizar o valor da multa por atraso na devolução de livros.
Student: É actor do sistema. Tem acesso ao centro de desportos.	**Student**: É actor do sistema. É capaz de requisitar, renovar, reservar ou devolver livros a biblioteca.
Staff: É actor do Sistema. Tem acesso ao centro de desportos e faculta serviços aos estudantes.	**Staff**: É actor do sistema. Ajuda o aluno no empréstimo, renovação, reserva ou devolver livros sem qualquer problema. Também pode requisitar, renovar, reservar ou devolver livros a biblioteca.

Figura 5.11: Análise as funcionalidades dos objectos similares

Vamos agora proceder a avaliação das funcionalidades entre os objectos que constituem os padrões *out of the box*.

- Padrão *out of the box* 1
  - *Universe* (Arquitectura 1) vs *Universe* (Arquitectura 2) : em conclusão ambos os objectos consistem num sistema de registo com informação dos alunos e dos funcionários onde essa informação é facultada. Essa informação pode ser utilizada por outros sistemas.
  - *Student Information* (Arquitectura 1) vs *Staff Student Information* (Arquitectura 2) é uma base de dados com informação sobre os actores.
  - *Maintain Staff Student Information* (Arquitectura 1) vs *Update staff and student Information* consiste basicamente em actualizar informação sobre o staff e os estudantes.
- Padrão *out of the box* 2
  - *Student* (Arquitectura 1) vs *Student* (Arquitectura 2) : é um actor do sistema que tem acessos a serviços disponibilizados pela universidade.
- Padrão *out of the box* 3

– *Staff* (Arquitectura 1) vs *staff* (Arquitectura 2): É um actor que tem acessos a serviços facultados pelo sistema e também faculta serviços a outros actores.

Nos classificamos os três padrões encontrados como padrões arquitecturais porque é similar a nível lexical e funcional.

Actualização de padrões

Tendo em consideração a *figura* 5.11 temos que generalizar os padrões *out of the box* que foram classificados como padrão arquitectural. Os padrões arquitecturais não são mais nem menos do que padrões que foram definidos como similares ao nível de "*inside the box*". De forma a transformar os padrões arquitecturais em apenas um padrão resumo que identifique os principais aspectos similares nos padrões *out of the box* é necessário transformar os dois objectos em um. O objectivo desta etapa é criar um padrão arquitectural generalista que represente os dois que serviram de base para a sua detecção.

Padrão Candidato 1:

Na *figura* 5.12 é possível verificar que o padrão é constituído por mais do que um objecto. Os nomes atribuídos a cada objecto foram generalizados de forma a identificar os objectos que constituem o padrão. O *data object* com a denominação*Student information* advém de dois data objects com a denominação *student information* e *staff student information*. O *application service* denominado por *update studente information* foi gerado partir de dois objectos denomiados por *maintain staff student information* e "*update staff and student Information*". O *application component Universe* denominou-se por esse novo devido similaridade 100 % dos objectos que geraram esse objecto.

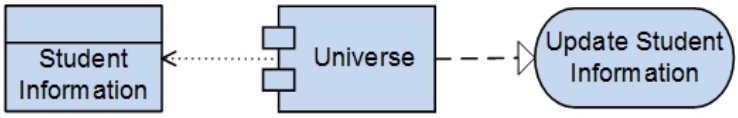


Figura 5.12: Padrão Arquitectural 1

- Funcionalidades do padrão arquitectural:

Objecto - *Student Information*: é uma base de dados com registos sobre os actores.

Objecto - *Universe*: Sistema de registo dos alunos e dos funcionários que fornece tanto informação do estudante como dos funcionários.

Objecto - *Update Student Information*: guarda e actualiza a informação referente aos estudantes e funcionários, como também outro tipo de informação das actividades destes actores.

Nota: Na definição do objecto "*Universe*" a referencia à sistemas comerciais foi removida, pois este objecto fica mais generalizado sem especificar a que sistema é facultado os dados.

Padrão candidato 2:

O padrão na *figura* 5.13 o *business actor* denominou-se "*student*"devido a similaridade 100% entre os objectos que a originaram.

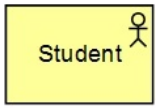


Figura 5.13: Padrão Arquitectural 2

- Funcionalidades do padrão arquitectural:

Objecto - *Student*: É um actor do sistema que tem acessos a serviços disponíveis.

Padrão candidato 3:

Tal como no padrão anterior o padrão da *figura* 5.14 o *business actor* denominado por ”staff”é proveniente de dois objectos com similaridade 100%.

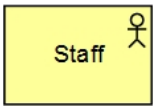


Figura 5.14: Padrão Arquitectural 3

- Funcionalidades do padrão arquitectural:

Objecto - *Staff*: É um actor que tem acessos a serviços facultados pelo sistema e que faculta serviços a outros actores.

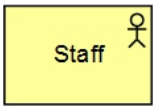
Arquitectura de Referência(AR)

- Correspondência dos objectos com a AR

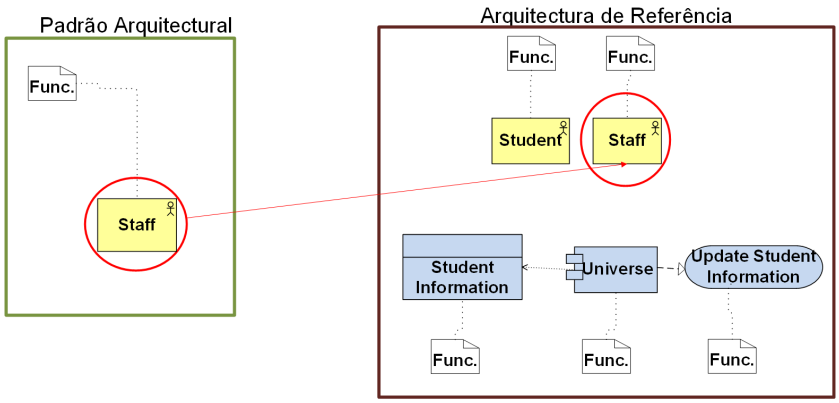
Nesta fase do projecto não temos nenhuma arquitectura de referência de forma a verificar se os padrões arquitecturais encontrados já existem. Porém gostaríamos de realizar um exercício para demonstrar como iria-mos verificar a existência do padrão arquitectural encontrado, este caso é demonstrado na ”demonstração hipotética 1”

Demonstração Hipotética 1

Imaginemos que estamos numa fase não inicial da descoberta de padrões e que já temos uma arquitectura de referência que corresponde a AR definida na *figura* 5.15. Presupondo que o padrão arquitectural descoberto é o seguinte:



Então temos que realizar o *matching* entre a AR e o padrão arquitectural:



Para verificar se este padrão arquitectural já existe temos primeiramente que realizar uma análise *out of the box* e a *posteriori* uma análise *inside the box*.
Caso não consigamos encontrar algum objecto pertencente a AR similar a nível lexical temos que analisar as funcionalidades dos objectos constituintes das AR. Mesmo que a similaridade a nível lexical exista é necessário que sempre se verifique a similaridade da funcionalidade. E apenas se existir algum objecto com funcionalidade similar é que podemos afirmar que o padrão descoberto já existe, caso contrário temos que actualizar a AR adicionando o padrão.

- Construção/Actualização da Arquitectura de Referência

Nesta fase dá-se a actualização da AR caso tenha-mos descoberto novos padrões arquitecturais. Como estamos numa fase inicial em que não existe AR todos os padrões descobertos vão fazer parte da AR como é possível verificar na *figura* 5.15. Assim sendo a *figura* 5.15 é a AR gerada ate ao momento. Caso no futuro seja necessário desenhar uma arquitectura que tenha um problema e um contexto semelhante podemos tomar decisões utilizando esta AR.

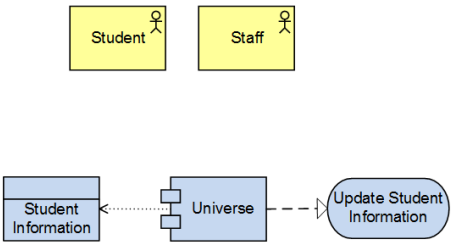


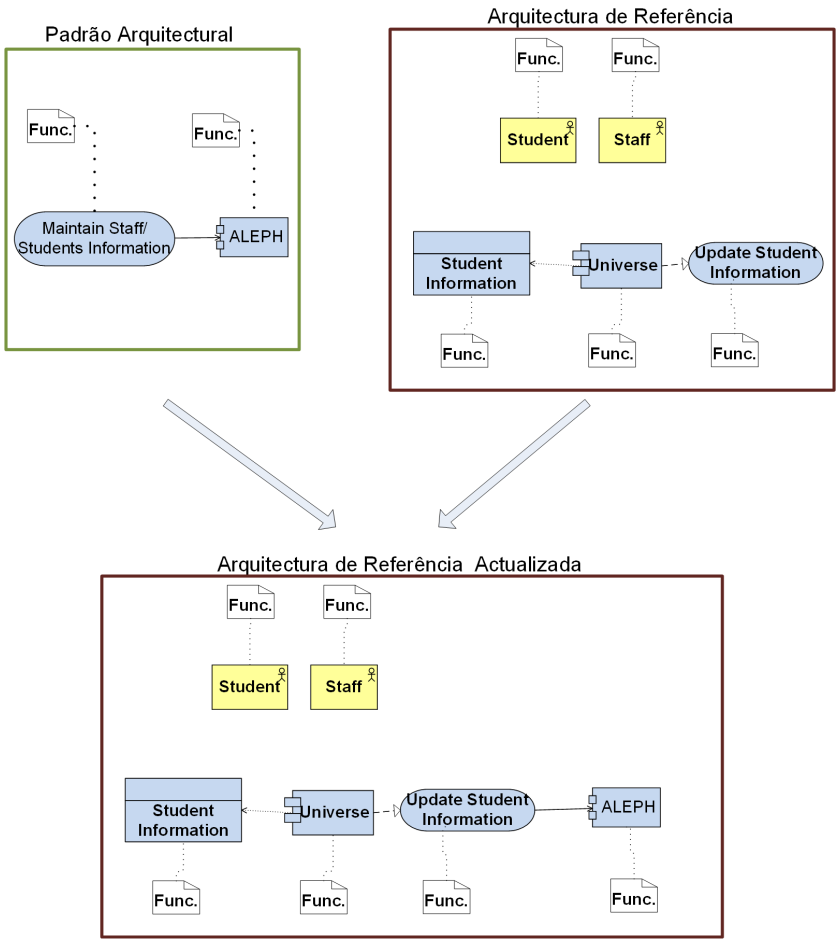
Figura 5.15: Arquitectura de referência

Imaginemos que estamos num estado mais avançado da descoberta de padrões e supondo que já tínhamos uma AR que era necessário actualizar, este caso é demonstrado na ”demonstração hipotética 2”.

Demonstração Hipotética 2

Na *figura* a baixo é-nos mostrado como é realizado a actualização de uma AR. Caso tenha-mos um padrão arquitectural novo, não existindo *matching* entre a AR e o padrão arquitectural é necessário colocar esse padrão na AR, realizando desta forma a actualização da AR.

Assumindo que o objecto *maintain staff/Students Information* do padrão arquitectural e o objecto *update student Information* da AR são similares devido a similaridade existente nas funcionalidades que se encontram colectadas em ficheiro XML, então temos que actualizar a AR. Para a actualizar a AR temos que acrescentar o objecto ”ALEPH”que tem uma relação com o objecto *update student Information* a AR.



Este exemplo é meramente hipotético tendo como objectivo exemplificar a actualização da AR caso estivesse-nos num estado mais avançado da detecção de similaridade.

5.3 Geração - 3ª Fase

Nesta secção vamos avaliar se o padrão arquitectural é um padrão ou anti-padrão de acordo com as consequências encontradas na documentação. Nesta fase ficamos dependentes da documentação ou da experiência adquirida na implementação das arquitecturas em análise, como não temos experiência nas arquitecturas utilizadas então estamos dependentes da documentação existente.

Porém na documentação encontrada não foi possível verificar a existência de consequências positivas ou negativas. O que foi notório durante a leitura é que os componentes faziam o pressuposto, não gerando consequências negativas.

Não conseguimos ter uma grande percepção some o impacto da utilização destes componentes

numa arquitectura. Os componentes faziam o que era pretendido, deduzindo desta forma que isto era um ponto positivo quando queremos desenhar uma arquitectura. Quando estamos a desenvolver uma arquitectura e utilizamos padrões, o objectivo é ter uma base ao nível funcional. Assim sendo, como não conseguimos medir o impacto negativo deduzimos que estes objectos são padrões uma vez que a consequência positiva é a descrição das funcionalidades a ser aplicada num determinado problema e um determinado contexto.

5.4 Documentação - 4ª Fase

Nesta fase temos que proceder a documentação dos padrões arquitecturais encontrados de acordo com o que foi definido na proposta de solução no capítulo 4.

- **Padrão 1**

Classificação do padrão: CE - Complexo Específico

Padrão/Anti-padrão: Padrão

Nome do Padrão: Gestão de Informação

Problema: Ser necessário actualizar sistemas e registar informação sobre actores que executam determinadas acções.

Contexto: O contexto da utilização deste padrão esta direccionados aos actores do sistema, sendo que a lista é a seguinte:

- Guardar informação e documentos;
- Actualizar dados;
- Actualizar sistemas;

Consequência:

- Manter registos sobre os actores do sistema.
- Actualizar novos dados e alterações.

Solução: Os objectos que constituem este padrão é o seguinte:

- *Student Information*: é uma base de dados com registos sobre os actores que utilizam os sistemas.
- *Universe*: Sistema de registo dos alunos e dos funcionários que fornece tanto informação do estudante como dos funcionários.
- *Update Student Information*: guarda e actualiza a informação referente aos estudantes e funcionários, como também outro tipo de informação das actividades destes actores.

"*Universe*" (component) é realizada por *Update tudent Information* (application). O *Universe* cria um novo objecto, ler dados a partir do objecto, escrever ou modifica os dados do objecto do *Student Information*(data object), em que a relação é *Access Notation*.

Arquitectura genérica:

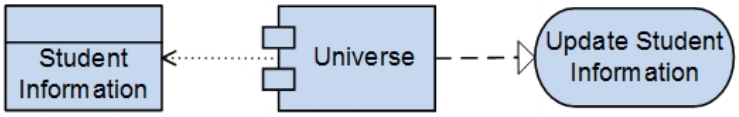


Figura 5.16: Padrão Candidato 1

Tipo de arquitectura empresarial: Arquitectura aplicacional, negocio e de infra-estruturas.

Princípio Arquitectural: none.

- **Padrão 2**

Padrão/Anti-padrão: Padrão

Nome do Padrão: *Student*

Problema: A inserção de um actor que utilize serviços facultados pelo sistema.

Contexto: Quando se pretende que exista um actor que tenha alguma interacção no sistema.

Consequência: A existência de um actor no sistema que utilize serviços.

Solução:

  - Objecto - "Staff": É um actor do sistema que tem acessos a serviços facultados.

Arquitectura genérica:

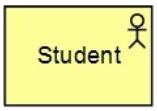


Figura 5.17: Padrão Candidato 2

Tipo de arquitectura empresarial: Arquitectura de negócio.

Princípio Arquitectural: none.

- **Padrão 3**

Classificação do padrão: SE - Simples Específico

Padrão/Anti-padrão: Padrão

Nome do Padrão: Staff

Problema: A inserção de um actor que utilize serviços facultados pelo sistema e que também faculte serviços a outros actores.

Contexto: Quando se pretende que exista um actor que tenha alguma interacção no sistema e que ajude outros actores na utilização de serviços.

Consequência: A existência de um actor no sistema que utilize serviços e que faculte serviços a outros actores.

Solução:

- Objecto - *Student*: É um actor do sistema que tem acessos a serviços facultados.

Arquitectura genérica:

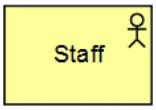


Figura 5.18: Padrão Candidato 3

Tipo de arquitectura empresarial: Arquitectura de negócio
Princípio Arquitectural: none.

5.5 Resumo

Neste capítulo foi aplicado a proposta de solução a duas arquitecturas. Arquitecturas essas que tem um problema e um contexto semelhante tendo como objectivo detectarem os padrões existentes entre ambas.

Durante a aplicação da solução foi necessário realizar pequenos ajustes ao algoritmo de forma a optimizar o processo da descoberta de padrões. A optimização passou por melhorar o algoritmo do calculo da similaridade utilizando o algoritmo *edit distance* com outra informação e também passou pela actualização das listas dos sinónimos, antónimos, preposições e conjunções.

Porém muito antes de optimizar o processo de detecção da similaridade existia um problema que consistia em definir de que forma a informação sobre os sinónimos, antónimos, preposições e conjunções podia ser carregada no sistema, chegando a conclusão que deveria ser através de XML devido a portabilidade. Outro grande problema foi encontrar a forma certa de guardar informação sobre os padrões garantindo todos os dados suficientes de forma a realizar o mínimo de comparações possíveis. Por exemplo, apenas poderiam ser analisados objectos do mesmo tipo e pertencessem a mesma camada.

Um dos maiores problemas que tivemos para aplicar a proposta de solução conforme o definido foi a quase inexistência de informação sobre os padrões descobertos, para conseguir analisar as consequências positivas e negativas da aplicação de um padrão.

Generalizando, foram descobertos três padrões arquitecturais sendo dois padrões simples específico e um complexo específico. Estes padrões foram documentados conforme o definido inicialmente de forma a transmitir o conhecimento adquirindo podendo a informação ser completada a *posteriori*.

Capítulo 6

Avalição dos Resultados

Este capítulo incide na avaliação dos padrões gerados na proposta de solução.

6.1 Avaliação Quantitativa

Esta é uma avaliação quantitativa onde serão medidos os seguintes aspectos:

Número de padrões descobertos (simples e complexos). Tempo de descoberta de padrões.

Foi realizado um pequeno exercício com três pessoas onde tiveram acesso as duas arquitecturas e tinham que analisar e detectar os padrões. O resultado deste exercício apenas pode ser comparado com a fase automatizada desta solução que corresponde a comparação out of the box. É apenas esta fase que se quer avaliar visto que a analise das funcionalidades depende sempre da capacidade de análise das pessoas, sempre sendo realizada de forma manual. A *figura* 6.1 retrata um dos testes realizado, onde é possível verificar as arquitecturas em questão.

Antes de efectuar o teste foi explicado qual era o propósito do exercício. O exercício consistia na descoberta de objectos que a primeira vista fossem similares através no nome, tendo em consideração que o nome dos objectos não tinha que ter 100% de correspondência entre eles. Também foi explicado o metamodelo utilizado na arquitectura para que a pessoa compreende-se o facto de terem que ser do mesmo tipo. Não foi referido o facto de poderem ser sinónimos nem terem que ser do mesmo tipo, pois estes aspectos faziam parte da proposta de solução.

Para que as pessoas não ficassem tempo indeterminado a procura de padrões foi estabelecido um tempo máximo de 30 min para a procura de padrões, podendo terminar a tarefa em qualquer momento.

Resultados obtidos:

- Testes com os utilizadores

Na *figura* 6.2 é possível verificar o resultado dos testes efectuado por três pessoas. Em média os utilizadores detectaram 9 padrões em 27 minutos.

- Proposta de solução

O programa demorou a descobrir 10 padrões apenas 2 segundos na fase comparação out of the box.

Figura 6.1: Teste realizado com os utilizadores

	Padrões descobertos	Tempo de descoberta de padrões	$\frac{Nº\ Padrões\ descobertos}{\text{Tempo de descoberta de padrões}}$
User 1	12	21	0,57 Padrões/min
User 2	6	30	0,2 Padrões/min
User 3	10	30	0.33 Padrões/min
Média	9	27	0,37 Padrões/min

Figura 6.2: Resultado dos testes realizado com os utilizadores

- Factos observados

As pessoas que foram sujeitas a este teste depois de encontrarem os padrões staff e student ficavam um bocado perdidos e começavam a tentar interpretar as funcionalidades dos objectos. Durante os testes foram fazendo perguntas de forma a compreender melhor a arquitectura.

6.2 Conclusão

Durante a realização dos testes viu-se que havia sempre a necessidade de recorrer as funcionalidades de forma a interpretar o que existia para além do nome dos objectos. Muitos dos padrões descobertos não tinham por base a similaridade dos nomes mas sim a suposição de funcionalidades.

A proposta de solução implementada é mais eficaz pela optimização de tempo e por eliminar comparações entre objectos de tipos e de camadas diferentes. Ter uma metodologia permite o foco no propósito, o que não aconteceu durante os testes realizados com estas três pessoas que não tinham uma linha orientadora e tentavam encontrar grandes quantidades de padrões mesmo que não existisse nenhuma similaridade.

Quando os testes terminavam explicava sempre a proposta de solução e recebia sempre o mesmo feedback, que era a necessidade de existir uma linha orientadora para a descoberta de padrões que permitisse o foco evitando a dispersão e a especulação.

Conclui-se que tanto a automação como os passos definidos para a descoberta dos padrões out of the box foram importantes para a primeira detecção de padrões. Também se pode concluir que os passos definidos para a descoberta de padrões basearam-se em aspectos que numa primeira vista eram tomados como irrelevantes mas que eram importantes para o ganho de tempo e aumento de padrões out of the box encontrados, como por exemplo excluindo combinações desnecessárias (diferentes tipos e camadas) assim como a inclusão de padrões que a partida estavam excluidos pela similaridade entre os nomes (utilização de sinónimos).

Capítulo 7

Conclusão

Este trabalho de investigação define a importância dos padrões e anti-padrões na AE, mas em primeiro lugar é necessário detectá-los. O método para a detecção de padrões e anti-padrões encontra-se definida na proposta de solução e denomina-se por RCGD (*Review, Compare, Generate and Documentation*). Os padrões são importantes a utilização de conhecimento já adquirido, permitindo o desenvolvimento de arquitecturas de forma rápida, com menos erros e proporcionando uma redução dos custos devido ao ganho de tempo. A utilização de padrões e a anti-padrões é importante para saber se o desenho arquitectural consegue alcançar o que é pretendido.

A necessidade de definir um método para descobrir padrões advém da inexistência de padrões considerados standards que cubram todas as arquitecturas de uma AE e que sejam específicas indústria onde a empresa se insere.

Assim sendo este trabalho de investigação contribui para indicar a importância da existência de padrões e anti-padrões para qualquer AE, identificar os pontos que numa arquitectura devem ser analisados para a detecção de padrões, criar um catálogo com padrões e anti-padrões para a utilização do conhecimento e propor um método para a descoberta de padrões e anti-padrões.

O método desenvolvido para a descoberta de padrões denomina-se por RCGD(*Revision, Comparation, Generation and Documentation*). Na fase de *Revisioné* escolhido as arquitecturas para análise. É também realizado um conjunto de comparações na fase *Comparation* que detecta quais dos objectos constituintes das arquitecturas em análise são similares entre si. A fase *Generation* passa por gerar padrões arquitecturais que mais tarde irão ser classificados como padrões *out of the box* após a sua aceitação. Na fase da *documentation* dá-se a documentação dos padrões e anti-padrões encontrados de forma existir a transmissão de conhecimento.

Na 1º fase da análise das arquitecturas é escolhido qualquer arquitectura para análise, mas apenas é realizada a comparação se tiverem problemas ou contexto similares, e forem modeladas no mesmo metamodelo. É de salientar que o levantamento das funcionalidades é uma necessidade para realizar uma análise *inside the box*, sendo fulcral para a classificação dos objectos*out of the box* como padrão arquitectural.

A 2º fase que corresponde a comparação refere-se a comparação dos objectos que constituintes

da arquitectura assim como das funcionalidades de cada objecto. Esta fase é constituída por 3 sub-fases comparação *out of the box*, padrões *out of the box* e comparação inside the box. Na sub-fase comparação *out of the box* é realizada a descoberta dos padrões similares a nível do *out of the box*, isto é, tipo de layer, tipo de objecto, nome e as suas relações. Quando é calculado a similaridade entre o nome dos objectos é necessário ter em consideração se as palavras são sinónimas, antónimas, preposições ou conjunções. Na sub-fase padrões *out of the box* é realizada a classificação dos padrões anteriormente encontrados de acordo com a sua constituição. Na sub-fase comparação inside the box é analisada as funcionalidades dos padrões *out of the box* podendo ser classificado como padrões arquitecturais após a análise das funcionalidades. Nesta sub-fase para além do *matching* entre os padrões *out of the box* também é realizado a actualização do padrão unificando os dois padrões similares. Nesta fase também é utilizado uma arquitectura de referência de forma a detectar se o padrão descoberto existe, e caso não exista é necessário actualizar a arquitectura de referência.

A fase 3 da geração do padrão é uma fase subjectiva pois depende da opinião das pessoas responsáveis pela descoberta dos padrões. Nesta fase reflecte-se sobre os padrões encontrados analisando as consequências como positivas, neutra ou negativas para que a *posteriori* sejam classificadas como padrão ou anti-padrão. Na 4ª fase é necessário catalogar os padrões e anti-padrões seguindo uma estrutura previamente especificada.

Gostaríamos de salientar que durante a demonstração prática foram encontrados alguns desafio, sendo necessário tomar determinadas situações. Foi muito difícil conseguir arquitectura já modeladas em ArchiMate que tivessem um contexto e um problema semelhante, pois a maior parte das empresas não partilham essa informação. A escassez de arquitecturas modeladas em ArchiMate foi o nosso maior problema não existindo experiência suficiente para as desenhar nem distanciamento para não gerar tendências resultados.

Durante o desenvolvimento da proposta de solução foram determinados vários limites a detecção de padrões e anti-padrões após a análise de uma arquitectura poderia trazer. Conseguindo encontrar as respostas em conceito como preposições, conjunções, sinónimos e antónimos e o cálculo da similaridade. Os problemas aqui encontrados incidiam em como detectar os objectos que eram similares sem ter-mos que analisar as suas funcionalidades. Visto que analisar as funcionalidades aos pares era impossível um trabalho muito desafiante, demoroso e no nosso ponto de vista irreal.

A escolha de ficheiros XML para guardar arquitecturas assim como dados, deveu-se a sua portabilidade. A escolha da linguagem C# para implementar a 2ª fase da RCGD teve peso a facilidade de criar *front-ends*. Na decisão do levantamento das funcionalidades na 2ª fase da RCGD mais especificamente na sub-fase padrões *out of the box* teve peso a já existência dos padrões *out of the box* para que o número de objectos que iriam requerer a análise inside the box fosse limitado, desta forma os objecto que iriam necessitar desse levantamento seriam ser em menor número.

Para o cálculo da similaridade entre os nomes dos objectos foi utilizado o algoritmo edit distance. Primeiramente tínhamos pensado em utilizar apenas o algoritmo mas após vários testes concluiu-se que simplesmente o algoritmo não trazia resultados satisfatórios. Já na demonstração prática ao longo da aplicação da proposta de solução foi necessário ir aumentado a lista de sinónimos e antónimos de

forma a refinar a solução. Nesta solução a pessoa responsável por descobrir os padrões tem um papel importante, pois é este que analisa compara as funcionalidades, que classifica as consequências assim como o padrão. O levantamento das funcionalidades foi dificultado pela pouca informação existente sobre a arquitectura assim como dos objectos que a constitui.

A limitação deste trabalho de investigação para a classificação de um padrão encontrado como padrão e anti-padrão a documentação existente não satisfazia as necessidades sendo que a classificação teve por base pouca informação e a métrica utilizada baseada apenas nas consequências é pobre. Outra limitação é a análise das similaridade entre as funcionalidades dos objectos ser feita de forma manual.

Como trabalho futuro propomos: traçar um plano para uma mudança cultural de forma a implementar a cultura de detectar e catalogar padrões e anti-padrões numa empresa, garantir que os padrões descobertos encontram-se alinhados com o negócio assim como os princípios arquitecturais da empresa e por fim definir outras métricas para classificar uma padrão encontrado como padrão ou anti-padrão de forma a que esta classificação seja mais minuciosa. Em conclusão, os objectivos desta dissertação foram alcançados pois conseguiu-se descobrir padrões para uma AE de forma a auxiliar no futuro o desenvolvimento de projectos, foi traçado um método para a descoberta de padrões assim como a geração de um catálogo de padrões ou anti-padrões.

Ao implementar esta metodologia numa empresa é necessário trabalhar a mudança das mentalidades das organizações, de forma a conseguir adoptar esta ou qualquer outra estratégia de descoberta e catalogação de padrões e anti-padrões para usufruir das suas vantagens. O maior obstáculo é gerar conança na utilização de novos padrões gerados pela empresa para que sejam utilizados com a mesma conança que padrões com mais maturidade como por exemplo os padrões presentes no livro Patterns of Enterprise Application Architecture de Martin Fowler.

Referências

[1] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. *Pattern-oriented Software Architecture: A System of Patterns*, volume Volume 1. WESLEY, 2001.

[2] Marc Lankhorst. *Enterprise Architecture at Work:Modelling, Communication and Analysis*. Springer, 2005.

[3] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley, Boston, MA, USA, 2002.

[4] N. Harrison, P. Avgeriou, and U. Zdun. Using patterns to capture architectural decisions. *Software, IEEE*, 24(4):38–45, 2007.

[5] E.Johnson Ralph. *How frameworks compare to other object - oriented reuse techniques: Frameworks = Components + Patterns*, volume 40. 1997.

[6] Design science aplicada ás pesquisas em administração: reflexos a partir do recente histórico de publicações internacionais. *RAI - Revista de Administração e Inovação*, 8:10–36, 2011.

[7] Ken Peffers, T Tuunanen, M Rothenberger, and S Chatterjee. A design science research methodology for information systems research. 24:45–78, 2008.

[8] S. Hain and A. Back. Towards a maturity model for e-collaboration - a design science research approach. In *System Sciences (HICSS), 2011 44th Hawaii International Conference on*, pages 1–10, 2011.

[9] Anne Cleven, Philipp Gubler, and Kai M. Huner. Design alternatives for the evaluation of design science research artifacts. In *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology*, DESRIST '09, New York, NY, USA, 2009. ACM.

[10] Danny Greefhorst and Danny Greefhorst. *Architecture Principles*. Springer, 2005.

[11] Standards Coordinating Committee of the Computer Society of the IEEE. Ieee standard glossary of software engineering terminology, 1990.

[12] Kenneth C. Laudon and Jane P. Laudon. *Management Information Systems: Managing the Digital Firm*. Prentice Hall, Upper Saddle River, NJ, USA, 7th edition, 2011.

[13] André Vasconcelos. *Arquitecturas dos Sistemas de Informação: Representação e Avaliação*. Doutoramento, Instituto Superior Técnico, Lisboa, 2007.

[14] André Vasconcelos. Arquitectura de sistemas de informação no contexto do negócio. diploma thesis, Instituto Superior Técnico, 07 2001.

[15] Ana Reis. Representação de indicadores chave de performance na framework archimate. diploma thesis, Instituto Superior Técnico, 05 2012.

[16] Victor Silva. Comparação de cenários arquiteturais. diploma thesis, Instituto Superior Técnico, 06 2011.

[17] Gerrit Muller and Eirik Hole. Reference architectures; why, what and how. Technical report, Embedded Systems Institute and Stevens Institute of Technology, 06 2007. White Paper Resulting from Architecture Forum Meeting.

[18] Eduardo Guerra, Felipe Alves, Uirá Kulesza, and Clovis Fernandes. A reference architecture for organizing the internal structure of metadata-based frameworks. *The Journal of Systems and Software*, 86:1239–1256, May 2013.

[19] Ricardo Chalmeta, Christina Campos, and Reyes Grangel. Reference architectures for enterprise integration. *The Journal of Systems and Software*, 57:175–191, 5 July 2001.

[20] E.B. Fernandez and N. Delessy. Using patterns to understand and compare web services security products and standards. In *Telecommunications, 2006. AICT-ICIW '06. International Conference on Internet and Web Applications and Services/Advanced International Conference on*, pages 157–157, 2006.

[21] Paul Clements Len Bass and Rick Kazman. *Software Architecture in Practice*. Addison-Wesley, Boston, MA, USA, 2003.

[22] Bruce Powel ouglass. *Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems*. The Addison-Wesley object technology series. Addison-Wesley Professional, Boston, San Francisco, Paris, 2003. Le cédérom d'accompagnement contient : Related papers ; Object Management Group (OMG) specifications ; Rhapsody, a UML compliant design automation tool that captures the analysis and design of systems and generates full behavioral code with intrinsic model-level debug capabilities ; RapidRMA, a tool that integrates with Rhapsody to perform schedulability and timeliness analysis of UML models.

[23] Erich Gamma, Richard Helm, Ralph Johnsin, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. ADDISON-WESLEY, 1995.

[24] Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. Generic schema matching with cupid. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, pages 49–58, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[25] Grupo de Projeto para as Tecnologias de Informação e Comunicação. Plano global estratégico de racionalização e redução de custos nas tic, na administração pública, 2011.

[26] Leonardo Silva and Fernando Silva. Reconhecimento depadrões arquitecturais em sistemas legados java. Technical report, Universidade da Tecnologia e do Trabalho, 2008.

[27] Flavio Gondima. Algoritmo de comparação de strings para integração de esquemas de dados. diploma thesis, Universidade Federal de Pernambuco, Fevereiro 2006.

[28] N Czechowski, S Padam, I Anderson, and C Woodcock. Enterprise architecture evaluation report. Technical report, Coventry University, 07 2011.

[29] E.S. Ristad and P.N. Yianilos. Learning string-edit distance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(5):522–532, 1998.

Anexos A

Arquitectura ArchiMate

A.1 ArchiMate

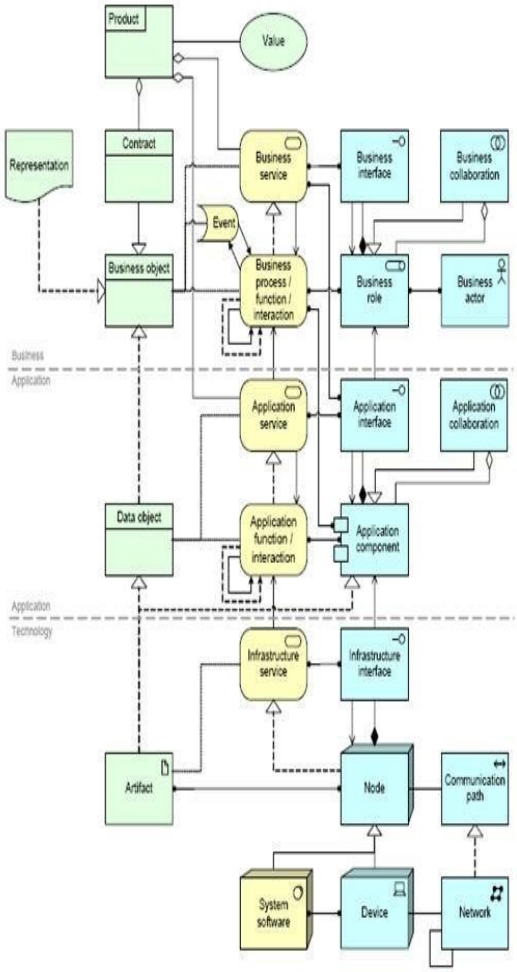


Figura A.1: Metamodelo de ArchiMate

Anexos B

Arquitecturas de testes

B.1 Arquitectura 1

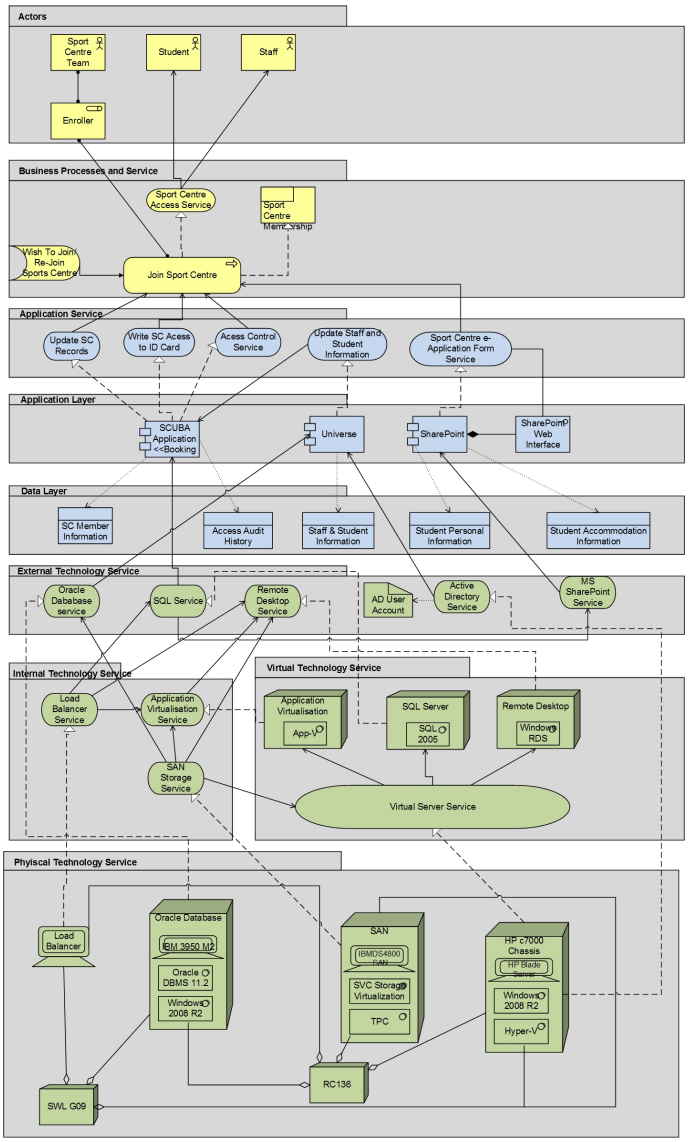


Figura B.1: Arquitectura1

B.2 Arquitectura 2

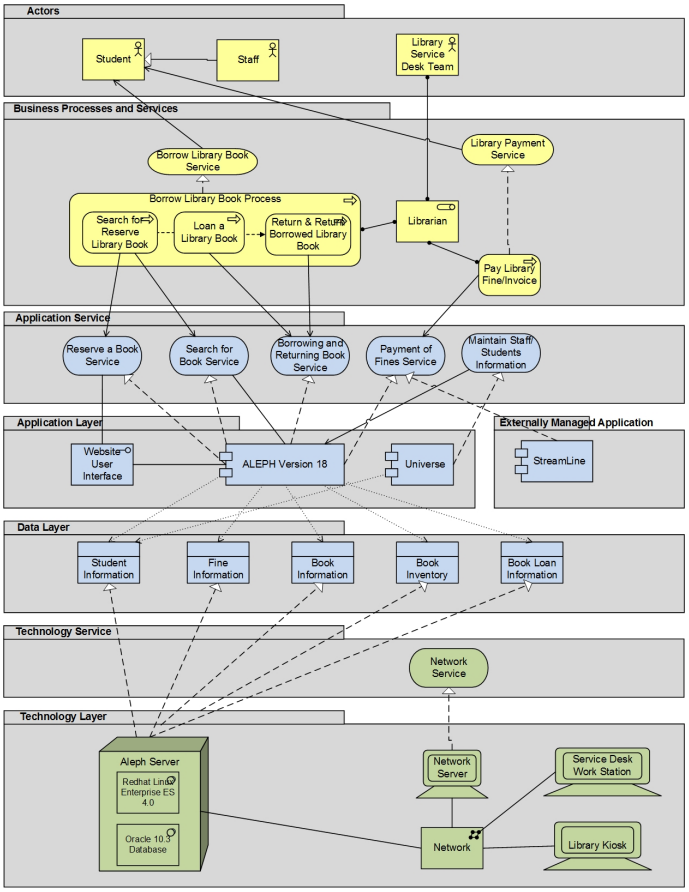


Figura B.2: Arquitectura 2

Anexos C

Estruturas XML

C.1 Estrutura XML guardar dados similares

Figura C.1: Exemplo objecto similares

Anexos D

Dados em XML usados na aplicação

D.1 Preposições e Conjunções

```xml
<?xml version="1.0" encoding="utf-8"?>
<PropositionsConjunction>
    <Data>
        <Name>for</Name>
        <Name>and</Name>
        <Name>nor</Name>
        <Name>but</Name>
        <Name>or</Name>
        <Name>yet</Name>
        <Name>so</Name>
        <Name>as</Name>
        <Name>if</Name>
        <Name>than</Name>
        <Name>till</Name>
        <Name>about</Name>
        <Name>after</Name>
        <Name>at</Name>
        <Name>by</Name>
        <Name>from</Name>
        <Name>of</Name>
        <Name>on</Name>
        <Name>to</Name>
        <Name>with</Name>
        <Name>within</Name>
        <Name>without</Name>
    </Data>
</PropositionsConjunction>
```

Figura E.1: Preposições e Conjunções

D.2 Sinónimos

```
<?xml version="1.0" encoding="utf-8"?>
<Synonymous>
    <Data>
        <List>
            <Name>update</Name>
            <Name>maintain</Name>
        </List>
        <List>
            <Name>student</Name>
            <Name>staff</Name>
        </List>
    </Data>
</Synonymous>
```

Figura E.2: Sinónimos

D.3 Antónimos

```
<?xml version="1.0" encoding="utf-8"?>
<Synonymous>
    <Data>
        <List>
            <Name>server</Name>
            <Name>service</Name>
        </List>
        <List>
            <Name>service</Name>
            <Name>service</Name>
        </List>
    </Data>
</Synonymous>
```

Figura E.3: Antónimos