FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Modelação de Fluxos Documentais – Estudo e Implementação de uma Ferramenta de Modelação para a Plataforma dCore

Henrique Manuel Pereira Mesquita e Mota

VERSÃO FINAL DE DISSERTAÇÃO
MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA E COMPUTAÇÃO

Orientador: José António Faria (PhD)

Abril de 2010

Modelação de Fluxos Documentais – Estudo e Implementação de uma Ferramenta de Modelação para a Plataforma dCore

Relatório de Dissertação
Mestrado Integrado em Engenharia Informática e Computação

provado em provas públicas pelo Júri:

Presidente: Luís Paulo Reis (PhD)

Arguente: Paulo Novais(PhD)

6 de Abril de 2010

Resumo

Uma das temáticas de interesse em torno das tecnologias normalizadas de modelação de Processos de negócio é a flexibilidade que possuem para se adaptar a casos específicos que se desviam do padrão.

Nesta dissertação é proporcionada uma visão acerca do planeamento e implementação de uma ferramenta de modelação gráfica de fluxos documentais para servir uma plataforma concebida para a sua gestão. Estes fluxos documentais podem ser traduzidos como processos orientados aos documentos, em contraponto com os processos de negócio BPM(*Business Process Management*), que são orientados às rotinas de trabalho empresariais.

A primeira análise efectuada ao problema, foi compreender os componentes que definiam a lógica de um fluxo documental nesta plataforma. Por um lado seriam previstos um conjunto de elementos, que deveriam ser utilizados para o desenho do fluxo. Deste conjunto de elementos destacariam-se as operações, às quais poderiam ser associados serviços/*plugins* para que fossem aplicados automatismos. Em segunda análise foi verificado que o conjunto de elementos é bastante simples, o que permite concluir que a plataforma em questão, aposta na flexibilidade através destes serviços.

Os objectivos deste projecto foram conciliar a simplicidade esperada no desenho de fluxos, com a possibilidade da lógica de desenho aumentar, tendo em vista o perfil dos utilizadores e as medidas necessárias para a sua manutenção e evolução. Embora as notações escolhidas BPMN(*Business Process Modeling*) e XPDL(Xml Process Definition Language), apresentassem as condições necessárias para que por um lado se conseguisse definir os fluxos e por outro lado possuir uma reserva para cobrir o surgimento de novos elementos, foi necessário prever também o caso de esta notação deixar de ser útil para o caso em questão. Como consequência esta solução foi implementada sobre uma arquitectura de camadas MVVM(Model View View-Model), que abstrai o interface com o utilizador e o modelo de dados.

Os objectivos deste trabalho foram cumpridos com sucesso, já que foi construída uma solução que através de formatos normalizados assegura a base necessária para a construção de um fluxo mediante as condições da plataforma. A definição do fluxo foi dividida em módulos que separam a lógica de desenho e a lógica dos serviços, facilitando a manutenção e definição de cada um destes componentes.

Palavras-Chave: Fluxo Documental, BPMN, XPDL, Processos de Negócio, Gestão Documental.

Abstract

One of the interesting questions around the business processes modeling standards, is if they have enough flexibility to be adapted to specific cases that deviate from the normal BPMN(Business Process Management) approach.
This dissertation provide an overview on the planning and implementation of a graphical modeling tool of document flows to serve an existing platform to manage them. The document flow can be described as a document oriented process in opposition to business processes, BPM, which are oriented to business work routines.
The first analysis of the problem was to understand the components that in which all the flow logic is divided. There are a set of elements, that define the flow design. In that set of elements there is a particular one, the operation, that can be associated to services/plug-ins, responsible to provide a certain automatism. In the second analysis was concluded that the set of elements is quite simple. The core of this platform is the the logic flexibility provided by the services associated to the automatic operations.

The main goal of this project was to find a way to balance between the simplicity of the flow architecture logic with the perspective of this logic suffers a evolution, taking into account the users profiles and the maintainability. Although the standards chosen, BPMN (Business Process Modeling) and XPDL (XML Process Definition Language), provide the necessary means so that the flow can be defined and at the same time save some elements to cover the emergence of new components in the flow design, it was also necessary to take into account the case of this notation became unuseful for this platform. As a consequence, this solution was implemented under an layer architecture named MVVM (Model View View-Model), which abstracts the user interface and data model.
The goals of this work were completed successfully, since it was built a solution using standard formats that ensures the necessary basis to define a flow under the rules that this platform demands. The definition of the flow was divided into modules that separate the draw logic and services logic, facilitating the maintenance and definition of each one of these components.

Keywords: Document Flow, BPMN, XPDL, Business Process, Document Management.

Agradecimentos

Gostaria de agradecer em primeiro lugar, ao meu pai pelo esforço financeiro e todo o apoio incondicional que me deu para que este dia chegasse. Agradeço também ao meu orientador José António Faria pela disponibilidade. Gostaria também de agradecer a todos os elementos da DOCKS, pelo tratamento que me deram desde o primeiro dia, especialmente ao meu orientador na empresa Nuno Carvalho.

Um abraço aos meus amigos, que muitas vezes me aturaram durante estes anos, especialmente Tiago Cavaleiro, Daniel Deira e José Mota.

Finalmente a todos os professores que me incentivaram e me transmitiram conhecimento.

Henrique Mota

Índice

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

API	*Application Programming Interface*
BPD	*Business Process Diagram*
BPEL	*Business Process Execution Language*
BPI	*Business Process Improvement*
BPM1	*Business Process Management*
BPM2	*Business Process Modeling*
BPMN	*Business Process Modeling Notation*
DZP	*dZign Parametrization extension*
GUI	*Graphic User Interface*
OMG	*Object Management Group*
PDF	*Portable Document Format*
UML	*Unified Modeling Language*
XML	*eXtensible Markup Language*
XPDL	*XML Process Definition Language*
XSD	*XML Schema Definition*

1 Introdução

O projecto apresentado nesta dissertação teve por objectivo especificar e desenvolver uma ferramenta de modelação gráfica de processos vocacionada à representação de fluxos de operações efectuadas sobre documentos, que foi baptizada de dZign.

Apesar de actualmente existir um mecanismo para a definição de fluxos para esta plataforma, não foram levados em conta os factores importantes para a sua utilização, já que exige dos utilizadores conhecimentos técnicos de baixo nível que não se justificam dado os conceitos e tecnologias que hoje existem para resolver problemas deste tipo.

De seguida são apresentados a motivação, o contexto em que se desenvolveu e os objectivos que se pretendiam alcançar através deste projecto, bem como o paradigma de gestão de processos subjacente.

1.1 Contexto e Motivação

A modelação sempre foi parte integrante no processo de interacção do ser humano com o mundo que o rodeia, embora muitas vezes este não tenha percepção deste facto, o ser humano age com base nos dados que extrai do mundo e as conclusões a que estes lhe proporcionam. No entanto este modelo é subjectivo, já que cada indivíduo interpreta o mundo de uma forma particular o que consequentemente dificulta um eventual processo de partilha e interacção entre entidades.[1]

Com o surgimento e evolução da computação e dos sistemas informáticos, foram sendo criadas condições para suportar diversos problemas existentes.

Uma das áreas que tem tem sido beneficiada com esta evolução é sem dúvida o mundo empresarial, que tem visto os seus processos serem modelados, optimizados e automatizados. Para que tenha sido possível suportar esta evolução, foi necessário existir comunicação entre os dois tipos de entidades envolvidas, informáticos, que arquitectam e desenvolvem os sistemas, e os administradores de empresas, conhecedores dos processos. Esta necessidade foi suportada com o surgimento de modelos gráficos de estados, que representam os processos de negócio das empresas, facilitando assim a interacção entre entidades com diferentes áreas e conhecimentos técnicos.

A evolução da computação e a necessidade de automatização de processos, proporcionou também o aparecimento de sistemas assistidos por humanos[1] em contraponto com os sistemas assistentes dos humanos[2]. Como normalmente os sistemas assistidos por humanos necessitam de dados submetidos pelo ser humano à priori para que se possa proceder à sua execução, passa a existir a necessidade que as ferramentas de modelação desempenhem também o papel de interface de submissão de dados, já que a percepção natural que o ser humano tem de um dado cenário, não se adequa aos dados requeridos pelos sistemas.[2]

Pretendendo-se então implementar uma solução de modelação para um sistema assistido por humanos, onde a modelação é vista como uma ferramenta de submissão de dados, procede-se à explicação da origem desta necessidade, bem como a sua motivação.

1.1.1 A Origem

A DOCKS com a experiência adquirida e desafios lançados pelos clientes, foi construindo a sua própria visão acerca da relação entre a gestão documental e os processos de negócio. Segundo esta visão, os fluxos documentais não são alvo da atenção adequada na abordagem actual BPM(1) (*Business Process Management*):

"Normalmente, por restrições orçamentais e tempo disponível, a abordagem BPM[1] é reduzida a um universo restrito de processos, muitas vezes mesmo a apenas um único processo. No entanto, pouca ou nenhuma atenção é dada aos Fluxos Documentais."[4]

Segundo esta perspectiva é necessário automatizar, monitorizar e analisar os processos orientados aos documentos, para que todo o processamento destes seja agilizado. Ao contrário da abordagem tradicional BPM[1], em que se focam os processos orientados ao trabalho, dá-se ênfase às operações executadas sobre os documentos.

No entanto é preciso referir que com esta abordagem não se pretende ignorar os processos de negócio, na verdade o que se pretende é continuar a agilizar estes processos considerando também esta perspectiva. Assim para além de se considerar as rotinas de trabalho de uma empresa, leva-se em consideração os documentos que as sustentam.

De seguida é apresentada a Figura 1.1, que pretende ilustrar a interpretação que a DOCKS desenvolveu sobre este problema:

1 Sistema envolvido em todo o tipo de actividades. Só quando não é possível realizar algum tipo de actividade é que requer a interacção do ser humano. Para mais referências consultar [2]

2 Sistema que apenas auxilia o ser humano a desenvolver as actividades. O propósito de todo o processo e sequência das actividades está fora do conhecimento do sistema. Para mais referências consultar [2]

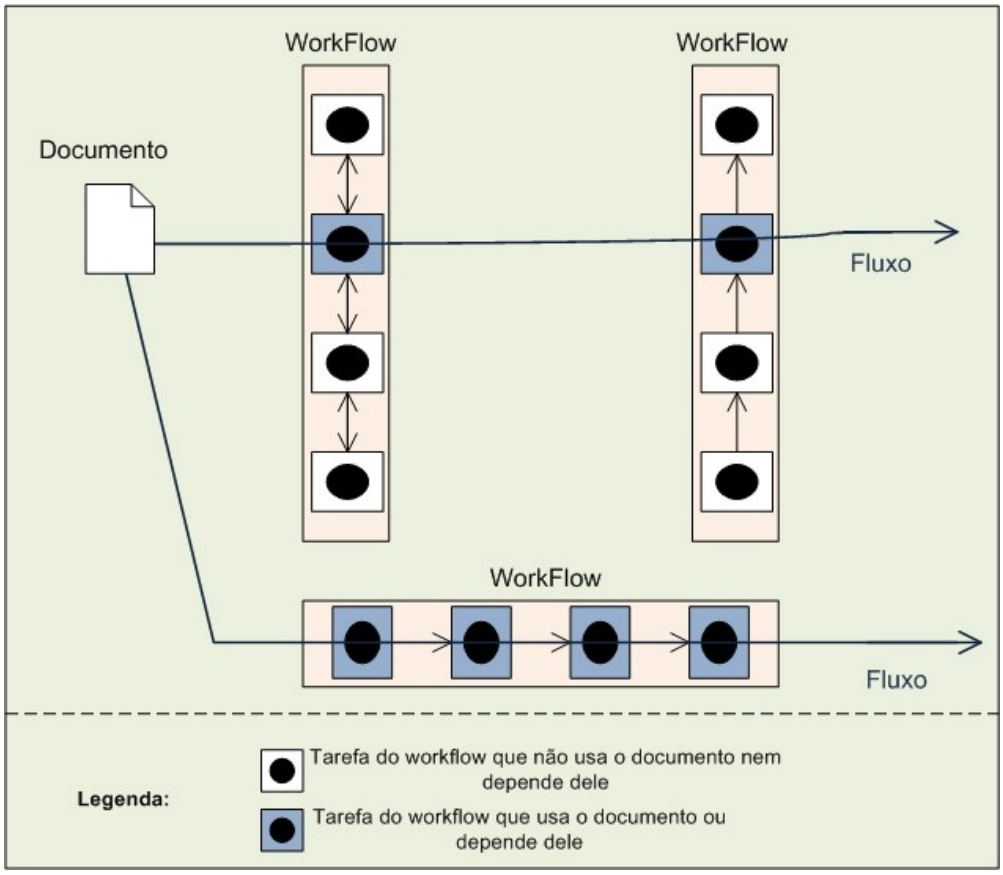

Figura 1.1: Fluxo Documental e a sua relação com WorkFlow.

Como se pode verificar na Figura 1.1, o fluxo documental é transversal aos *workflows* de uma empresa. Um fluxo documental pode ser traduzido como um processo orientado ao documento, e apresenta a vantagem de proporcionar uma visão de todos os pontos de afectação de um dado documento, mesmo que estes pertençam a *workflows* diferentes.

Para que se compreenda melhor a perspectiva enunciada anteriormente, de seguida é ilustrado um hipotético cenário.

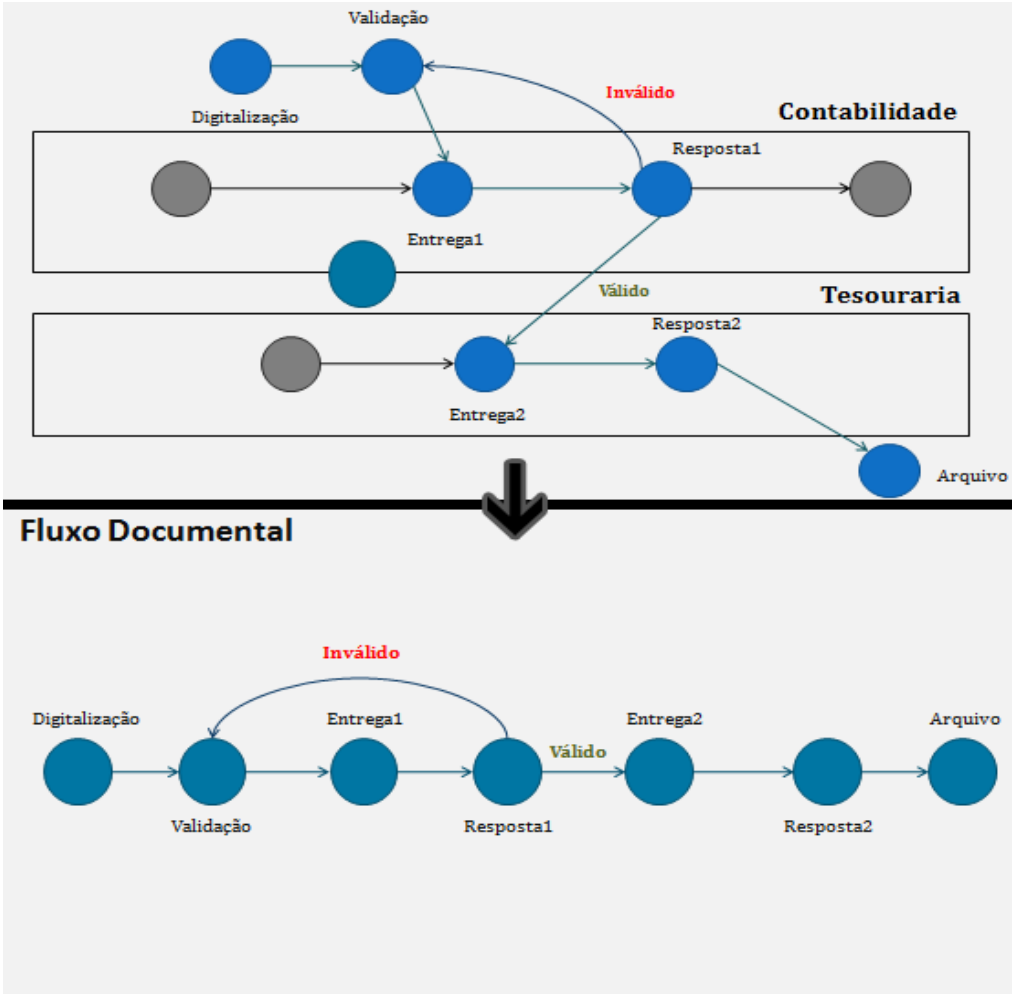

Figura 1.2: Exemplo de um fluxo documental.

Como é ilustrado na Figura 1.2, a factura é primeiro capturada e validada (no que diz respeito à coerência dos dados capturados), posteriormente é entregue ao departamento de contabilidade(embora não seja visível e descriminado nesta perspectiva). É esperada então uma resposta quanto à sua validade no que diz respeito ao contexto de negócio (exemplo: o fornecedor descriminado na factura existe na base de dados), caso este seja válido o documento segue para a tesouraria onde é emitido um comprovativo de pagamento (outro documento) e arquivado. Caso contrário é necessário afinar a captura e proceder de novo à entrega e resposta.

A grande vantagem desta abordagem, será sem dúvida a perspectiva que proporciona par posteriormente reduzir o tempo de espera entre as operações executadas sobre o documento. Pela análise a um determinado tipo de documentos e através de estatísticas, poderão ser determinados o *bottleneck* do fluxo e proceder a alterações nos *workflow* para agilizar todo o processamento.

Todo este paradigma originou a construção de uma ferramenta, baptizada de dCore, que pretende fornecer uma infra-estrutura de apoio ao processamento dos documentos. Esta

ferramenta, apresenta uma lógica a nível de fluxos muito simples, com poucos componentes, sendo que a sua força está na definição dos serviços que pretendem automatizar operações. Neste momento também está a ser desenvolvida uma consola, com o nome dConsole, em que é pretendido monitorizar estes fluxos graficamente. Uma explicação mais detalhada sobre a empresa está presente no anexo A e o escopo de ferramentas que se pretende desenvolver sobre o dCore está descrito no anexo B.

1.1.2 Motivação

Actualmente existe um mecanismo de definição de fluxos, que apresenta algumas limitações, conforme irá ser detalhado no capitulo 2. Este mecanismo consiste em definir dois ficheiros XML, num esquema não normalizado, o primeiro contendo toda a lógica de desenho e o segundo a especificação dos serviços/plugins que contêm a lógica programável. O primeiro inconveniente que se pode concluir desde já, consiste no facto de ser requerido ao utilizador um conhecimento do XML e o esquema deste, que não sendo normalizado torna nula a possibilidade de o conhecer antes de utilizar este mecanismo. O segundo inconveniente, prende-se com o facto de não existir uma distinção dos perfis necessários para efectuar o desenho e a parametrização do fluxo. Esta distinção é essencial para que possa existir um planeamento sequencial do fluxo e divisão de tarefas, já que um arquitecto de fluxos não possui obrigatoriamente um conhecimento técnico, suficiente para implementar os serviços das operações automáticas sobre o documento. Em último lugar, o facto de não existir uma referência gráfica para o fluxo no seu processo de definição, impossibilita o acompanhamento por parte do cliente, o que seria um factor positivo a considerar. Com base nesta motivação e na é apresentado de seguida na Figura 1.3, um esquema que ilustra o processo desejado:

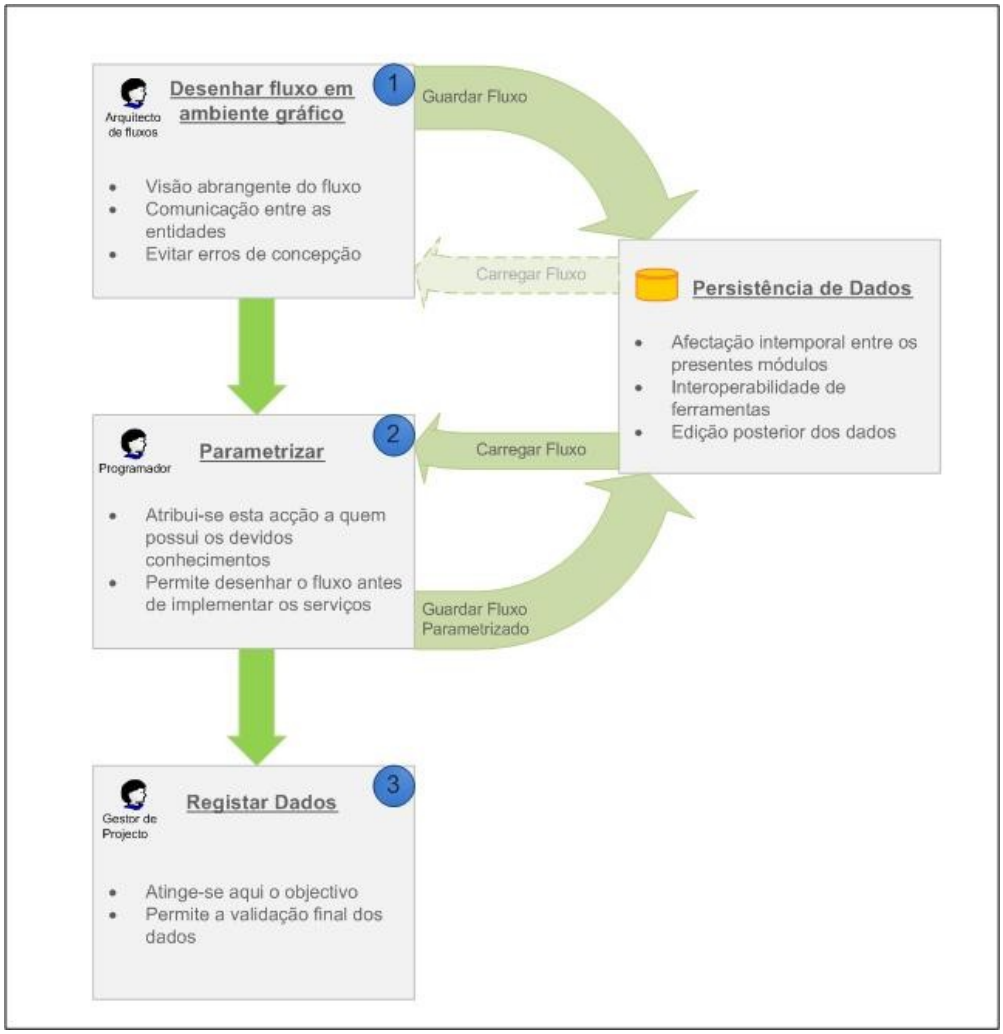

Figura 1.3: Processo desejado de definição de um fluxo

A segunda grande motivação deste projecto, prende-se com o facto de a plataforma dCore ser bastante recente e uma análise aos seus conceitos e modelo de dados ser sempre útil para a sua evolução. Assim ao encontrar uma solução para a definição de fluxos, haverá sempre um contacto directo com a plataforma, serão elaboradas novas perspectivas e poderão ser encontradas algumas inconsistências.

1.2 Objectivos do Projecto

Finalizada toda a contextualização do problema e tendo presente as motivações que levaram à sua realização, interessa traçar em linhas gerais, os objectivos que se esperam atingir ao arquitectar e implementar o dZign:

- Encontrar uma notação gráfica, de preferência standard, para a definição dos fluxos.

- Encontrar um mecanismo de persistência de dados, de preferência com o auxilio de um esquema aberto e standard, para que o fluxo possa ser editado e sempre que necessário afectado por diversos módulos/sistemas.
- Encontrar uma arquitectura que permita a execução de cada requisito presente na definição de um fluxo dCore, tendo em conta os diversos perfis necessários (Arquitecto de fluxos/Programador/Cliente).
- Efectuar o registo do fluxo na plataforma dCore através de uma API disponibilizada para o efeito.
- Elaborar um mecanismo de validação, tendo sempre em conta que o fluxo terá que respeitar as regras ditadas pelo dCore.
- Potenciar a usabilidade e simplicidade.
- Não limitar a manutenção da solução, dado que o dCore poderá estar sujeito a constantes evoluções.

1.3 Estrutura da Dissertação

No Capítulo 1 - Introdução, foi feita uma descrição do problema, explicando os conceitos no qual se baseia. De seguida este foi contextualizado e descrita a sua motivação, para que no final pudessem ser traçados os objectivos.

Para além da introdução a dissertação será composta por mais cinco capítulos.

No Capítulo 2 - Análise de Necessidades , é feita uma análise das limitações do mecanismo actual de definição de fluxos para a plataforma dCore, para que de seguida sejam especificados as características a levar em conta para encontrar os requisitos .

No Capítulo 3 - Tecnologias, é feita uma análise às tecnologias disponíveis no mercado que poderão ser utilizadas para implementar a solução, procedendo-se então às escolhas que mais se adequam ao que é pretendido.

No Capítulo 4 - Análise e Especificação de Requisitos, são listados os requisitos funcionais e não funcionais do dZign, para que desta forma se tenha uma base para definir a arquitectura.

No Capítulo 5 - Concepção, é descrita a arquitectura utilizada para implementar o dZign, com base nos requisitos anteriormente definidos, as tecnologias escolhidas para implementar esta solução. São especificados alguns pormenores e decisões mais baixo nível da implementação e demonstrados alguns resultados finais que se atingiram.

No Capítulo 6 - Conclusões e Trabalho Futuro, é feito um balanço de todo o projecto e traçadas directivas para que no futuro o dZign possa ser melhorado.

2 Análise de Necessidades

Neste capítulo é efectuada uma visão do problema actual, passando-se depois a uma revisão cientifica para que possam ser traçadas linhas de procedimento para o presente projecto.

2.1 Descrição da plataforma dCore

No sentido de apoiar o seu paradigma a DOCKS abriu caminho à implementação de uma API/plataforma, baptizada de dCore, que permite gerir e agilizar as operações que são executadas sobre um documento durante o seu ciclo de vida. Com esta ferramenta é esperado redesenhar os processos de negócio de uma empresa, segundo a perspectiva dos documentos, considerando apenas todos os processos e tarefas que os afectam ou são afectados por estes, como foi ilustrado na Figura 1.2. Com base nesta abordagem e num conjunto de métricas, pretende-se por via de monitorização, agilizar o fluxo documental, reduzindo o tempo de espera de um dado documento, entre o final de um processo e o começo de outro. Como consequência desta abordagem é esperado atingir uma maior disponibilidade por parte dos documentos e uma inevitável melhoria dos próprios processos de negócio. Uma das grandes funcionalidades presentes nesta plataforma será também a actualização dos fluxos em tempo real, ou seja, mudar o desenho do fluxo e o código executado sem ser necessário voltar a proceder ao registo dos dados, permitindo suspender o fluxo, alterar e analisar o efeito destas alterações. O dCore enquanto sistema de gestão de fluxos documentais, apresenta conceitos próprios, pois a sua abordagem apresenta um desvio sobre a abordagem tradicional de BPM[1]. É pretendido tratar todos os pontos de afectação do documento, ao mesmo nível, seja um processo de negócio ou uma simples tarefa. De seguida serão descritos os conceitos da plataforma, apresentados na Tabela 2.1:

Tabela 2.1: Elementos de um fluxo aceite pela plataforma dCore

Nome:	Descrição:
Flow	Conjunto de dados que representam todas as operações e a ordem/condições de sua execução.
Object Type	Identificador de um *Mapping* no contexto do dCore.
Operation	Unidade atómica do fluxo.

Step	Conjunto operação/destinos possíveis de acordo com condições.
Step Path	Condição/Ligação entre passos.
Mapping	Descrição de um conteúdo. Para além de ter as propriedades, são artefactos que implementam a sua persistência e contêm regras de validação.
Phase	Sequência de operações. Pode ser interpretado como um sub-fluxo reutilizável.
Service	Código executável, responsável pela automatização de um operação.

Neste momento, está a ser desenvolvida também uma consola web-based com o objectivo de reunir as funcionalidades do dCore e de outros produtos. Esta consola apresentará um interface gráfico para o utilizador, para que este possa ter um visão mais abrangente de cada problema. Espera-se assim uma organização das ferramentas e portabilidade, para que se possa facilmente implementá-las num rede intranet ou mesmo na internet.

2.2 Diagnóstico

Apesar de o conjunto de elementos que compõe um fluxo na plataforma dCore, se apresentar bastante simples, o mecanismo actual de submissão de fluxos, já referido anteriormente, apresenta algumas limitações, não no que diz respeito à sua funcionalidade e cumprimento de objectivos, mas sim em questões de usabilidade, validação, manutenção e conivência com os perfis dos utilizadores desejáveis para este processo.

A primeira limitação, prende-se com o facto de não existir um suporte gráfico à definição de um fluxo, o que resulta na seguinte lista de inconvenientes:

- O arquitecto, não tendo uma referência gráfica do fluxo, não tem condições de facilmente detectar e corrigir erros a nível de desenho.
- A comunicação entre os arquitectos de fluxos e os clientes não é suportada em tempo de construção do fluxo, o que impossibilita eventuais reparos que o cliente poderia desejar efectuar.
- Sendo o fluxo definido num formato XML, não padronizado quanto à estruturação da relação entre os seus elementos, implica não só conhecimentos técnicos acerca desta tecnologia, como também uma formação especifica nessa estrutura.
- Mesmo possuindo os conhecimentos necessários para construir o fluxo, estará sempre associada uma probabilidade elevada de erro de definição do XML, que vai agravando conforme a complexidade dos fluxos aumenta, induzindo um factor de stress no utilizador, o que é um aspecto negativo em qualquer sistema informático.

A segunda limitação prende-se com com o facto de a persistência dos dados, estar a ser realizada num formato não padronizado. Embora apresente a grande vantagem de encaixar na perfeição com os conceitos da plataforma dCore, já que foi especificado exclusivamente para este efeito, mesmo que este formato fosse transparente para o utilizador, através de um interface gráfico, apresentariam-se os seguintes inconvenientes:

- Se o dCore fosse expandido em termos conceptuais ao nível do fluxo, todo este formato teria que ser repensado, tendo em conta a relação dos elementos já existentes anteriormente e os mais recentes.

- Não é viável a interoperabilidade com ferramentas externas, o que é sempre uma desvantagem, dado a existência de várias ferramentas de modelação no mercado.
- Para existir evolução deste mecanismo é necessário conhecer o formato dos dados, o que implica uma formação neste formato, caso uma nova equipe de trabalho tome conta do projecto. Se os dados fossem representados num formato normalizado, existiria uma probabilidade de conhecimento por parte desta equipe e mesmo que esse conhecimento tivesse que ser adquirido no momento, poderia ser reutilizável para outros projectos.

2.3 Levantamento de Necessidades

Apesar de esta ser uma situação extraordinária, já que estamos presentes de um cenário que envolve um paradigma e abordagem nova ao modo como se deverá agilizar um fluxo documental, deverão ser definidos os caminhos a seguir na implementação do dZign com base em algumas conclusões que podem ser inferidas de conceitos existentes e da lógica. Os pontos que de seguida serão referidos, poderão ter consequências directas na escolha das tecnologias, na especificação de requisitos e até mesmo no planeamento arquitectónico da aplicação.

2.3.1 Notação Gráfica

A escolha da notação gráfica, assume um carácter relevante neste projecto, já que esta se apresentará como a principal ferramenta que o arquitecto de fluxos terá ao seu dispor para executar a sua tarefa. Apesar de a abordagem defendida no paradigma apresentado diferir da abordagem tradicional BPI, seria interessante proceder à escolha de uma notação gráfica BPM^2(Business Process Modeling), para que se potenciasse a interoperabilidade entre ferramentas e entre abordagens.

O primeiro aspecto essencial na escolha da notação gráfica, será a cobertura fornecida aos elementos dos fluxos dCore. Embora não se espere que nenhuma notação gráfica BPM^2 traduza integralmente os conceitos abordados no presente paradigma e consequentemente na ferramenta dCore, é essencial que possa ser traçado um paralelismo entre os elementos desta notação e os elementos dCore, para que no final possam ser realizadas as devidas conversões e chegar ao resultado desejado. Torna-se evidente então que a notação gráfica não poderá sobrepor-se aos interesses e restrições patentes nos fluxos dCore.

O segundo aspecto a considerar será o universo de elementos da notação gráfica. Como o dCore é uma plataforma que se prevê passível de sofrer constantes evoluções, o dZign deverá estar preparado para ser actualizado e acompanhar o ritmo. Caso tal não aconteça, o dZign verá revogada a sua utilidade, pelo que se torna importante considerar o universo de elementos que se deverá estender muito para além da simplicidade que hoje o conjunto de elementos dos fluxos dCore apresenta.

O terceiro aspecto prende-se com a comunidade de utilizadores desta notação. Este aspecto é traduzido em dois factores. Ao escolher uma notação com um grande número de utilizadores, assegura-se que existirão um grande numero de ferramentas que poderão interagir no futuro com o dZign/dCore, potenciando a interoperabilidade. Na mesma linha de raciocínio, escolher uma notação com um grande número de seguidores, traduz-se na possibilidade de integrar um maior número de recursos humanos com o seu conhecimento e num maior suporte e troca de experiências sobre a tecnologia.

2.3.2 Persistência de Dados

O formato da persistência de dados, à semelhança da notação gráfica apresenta-se com um grau de importância elevada para o dZign. Se por um lado é na escolha de um formato padrão para a notação gráfica que começa a ser traçado o objectivo do dZign e de onde se pode começar a procurar características como a compatibilidade à abordagem tradicional de BPI e a outras ferramentas, por outro lado é na persistência de dados que encontramos o elo de comunicação, já que é este formato que a informação relativa ao desenho do fluxo chegará aos módulos a jusante do desenho de fluxos e aos sistemas externos ao dZign.

A primeira característica que deverá ser levada em conta, prende-se com a notação gráfica. O formato de persistência de dados deverá traduzir todos os cenários representados através desta. Caso seja verificada esta condição, garante-se assim que a escolha do standard da notação gráfica já garante por si só a compatibilidade com os fluxos dCore.

A segunda característica que se pretende privilegiar, será o suporte à definição das características gráficas dos elementos. Como neste momento está a ser implementado uma consola gráfica que pretende servir de interface entre o utilizador e o dCore, guardar este tipo de informação seria assegurar desde já uma base para posteriormente as representar nesta consola.

Em terceiro e último lugar, seria importante que este formato fosse assente sobre a tecnologia XML(eX*tensible Markup Language).* Pensando nas tecnologias que poderão ser utilizadas para implementar o dZign, a escolha de um formato deste tipo aumentaria o universo de possibilidades, já que hoje em dia a maior parte apresenta um bom suporte ao XML.

2.3.3 Validação dos Dados

A validação é um processo que deverá estar presente no dZign. A primeira justificação e a mais óbvia prende-se com o facto de a modelação gráfica estar a ser utilizada para submeter dados a uma aplicação *human assisted*, por este motivo tem que ser levado em conta o formato de dados que é suportado por esta aplicação. A segunda justificação, é consequência directa da escolha de uma notação gráfica e formato de persistência de dados standard. Ao realizar uma escolha deste tipo, terão que ser sempre respeitadas as regras, caso contrario deixa-se de estar na presença de padrões.

O modo como esta validação tem que ser estruturada é importante para o sucesso do dZign. Existindo primeiro uma tarefa de desenho de fluxo e de seguida uma tarefa de parametrização do mesmo fluxo com os serviços necessários, terá que ser assegurado primeiro que nenhum erro é cometido no desenho, de seguida terá que se assegurar que as parametrizações estão de acordo com o desenho especificado. Preferencialmente esta validação deverá ser gradual e deverá existir até um controlo para ir contextualizando as acções do utilizador de maneira a evitar o retorno de erros na validação. Antes de submeter os dados para que se proteja a integridade do próprio dCore, deverá ser efectuada uma validação que cobre os pontos anteriores.

2.3.4 Usabilidade e Simplicidade

Como em qualquer sistema informático moderno, a usabilidade deverá ser uma característica presente no dZign. Apesar de este ser um dado adquirido existem questões que não poderão ser ignoradas, primeiro temos que ter em conta um principio básico da usabilidade:

"User interfaces should be designed to match the skills, experience and expectations of its anticipated users" [3]

Como foi referido anteriormente é necessário proceder primeiro ao desenho do fluxo e posteriormente à parametrização das tarefas automáticas que o mesmo contém. O desenho do fluxo está associado a um utilizador que apresenta uma visão geral deste e que não tem que necessariamente possuir conhecimentos técnicos de programação. A parametrização está associada a um utilizador que tem conhecimentos de programação, que está habituado a implementar rotinas, não tendo necessariamente que apresentar uma visão do fluxo.

É sabido que as aplicações que não apresentam uma organização e uma racionalização das suas funcionalidades, pode desviar o utilizador do objectivo que realmente é pretendido. Uma abordagem em torno da simplicidade, permitirá que o utilizador se foque nos seus objectivos e não perca tempo a tentar compreender para que servem todos os elementos/funcionalidades ao seus dispor.

Porém a simplicidade não se deverá esgotar apenas no que diz respeito ao utilizador. A simplicidade na manutenção, deverá ser verificada, já que existe sempre o perigo da notação gráfica ou o formato de persistência de dados se tornar inútil para o objectivo desta ferramenta. Caso tal se verifique e de forma a evitar a implementação de uma nova ferramenta de raiz, deverão ser facilitadas as trocas destas tecnologias.

3 Tecnologias

Tendo sido realizada a revisão cientifica do trabalho, os moldes sobre os quais este vai operar, será apresentada aqui a revisão tecnológica que terá como base a análise feita até este ponto. Primeiro é necessário determinar qual a melhor notação para ser utilizada no desenvolvimento do dZign, procedendo-se depois à escolha do formato desejável para a persistência de dados e finalmente são analisadas as tecnologias de desenvolvimento para que se possa optar pela(s) que se apresenta(m) mais convenientes(s).

3.1 Notações Para Representar os Processos

3.1.1 BPMN

O BPMN (*Business Process Modeling Notation*) [5], é uma notação gráfica mantida pela OMG (*Object Management Group*), especificada para cobrir as exigências no que diz respeito à representação de processos de negócio.

Apresenta 4 categorias básicas de elementos, como é ilustrado na Figura 3.2:

Figura 3.1: Elementos da notação BPMN

Em termos de conjunto de utilizadores, o BPMN é das notações mais utilizadas para os processos de negócio e apresenta uma lista extensa de ferramentas que a utilizam como notação gráfica.

Podemos concluir que a tecnologia BPMN pode servir a plataforma dCore, sendo necessário aplicar para o efeito algumas regras contextuais.

3.1.2 UML

A tecnologia UML [6] é mantida pela unidade OMG e foi desenvolvida para apresentar um conjunto de elementos que auxiliem no planeamento e modelação de software.

È possível representar fluxos com esta tecnologia, e não existirão problemas no caso da sua aplicação ao dZign. No entanto como não foi especificada para seguir a modelação de processos de negócio e apresenta muitos elementos que a nível de fluxo nunca poderiam ser utilizados, mesmo que se restringisse esses elementos de forma a prevalecer a simplicidade para o utilizador, poderia existir um comprometimento da expansibilidade e extensibilidade, tal é a grandeza do universo de elementos que a equipa de desenvolvimento teria que analisar.

Apesar de tudo existe um aspecto que pode ser vantajoso no futuro, como o UML é mantido pela mesmo entidade que o BPMN, OMG, poderá ser especificada no futuro uma interoperabilidade entre estas duas tecnologias. Dado que existem operações que necessitarão de serviços para executar, será interessante explorar a possibilidade de modelação dos próprios ou indo mais longe, explorar a possibilidade de modelar estruturas de dados consoante os fluxos e/ou *object types*, que ajudem a encontrar métricas úteis para cada caso.

3.1.3 XPDL

A tecnologia XPDL(XML Process Definition Language) [7] é um formato aberto para definir diagramas de processos de negócio e actualmente é muito utilizado para fazer persistir diagramas do BPMN.

A maior vantagem e utilidade deste formato é preocupar-se com todos os aspectos de um BPD(*Business Process Diagram*), incluindo as coordenadas gráficas dos elementos constituintes.

Caso seja tomada como opção de formato de persistência de dados para o dZign, apresenta-se a vantagem de se encontrar disponível um esquema de validação, restando apenas a preocupação de validar no contexto dCore, já que o formato XPDL também tem que cumprir os requisitos BPMN necessários para o dZign.

3.1.4 BPEL

O BPEL (*Business Process Execution Language*) [8] é um standard especificado para a execução de processos de negócio, promovendo a interacção entre estes e os serviços.

Embora seja possível a sua utilização como formato de persistência de dados para o BPMN, após uma analise das suas especificações, foi fácil verificar que é difícil manter o diagrama original definido com BPMN, quando convertido em BPEL. O facto de este não ter sido especificado para suportar a definição dos elementos BPMN, não guardando por exemplo as coordenadas gráficas, torna-o desvantajoso em alguns cenários.

A sua aplicabilidade no dZign era de facto interessante, se apenas fosse tido em conta a coerência funcional do fluxo e não existisse o intuito de mais tarde apresentar este graficamente. Como o XPDL este formato apresenta um XSD para a sua validação.

3.1.5 Conclusões

Após a análise destas tecnologias de suporte à apresentação gráfica e persistência de dados, pelas características de cada uma, a melhor solução será a utilização do conjunto BPMN e XPDL. O BPMN porque se adequa mais nos conceitos dos fluxos dCore, sendo mais fácil traçar o paralelismo entre elementos de um e de outro e o XPDL porque foi pensado para definir BPD ao contrário do BPEL que é um suporte à execução de processos de negócio.

3.2 Ferramentas para Representação de Processos

3.2.1 *BizAgi BPM Modeler*

O *BizAgi BPM Modeler [9]* é uma ferramenta de modelação, integrada num pacote denominado BPMN Suite, que foi concebido para melhorar os processos de negócio sem a necessidade de programar. Esta ferramenta apresenta o que é necessário através da notação BPMN para modelar processos de negócio e apresenta mecanismos de validação.

Apresenta a opção de exportar para o formato XPDL e formato *Visio*, bem como para elementos de suporte à documentação, tem-se a opção de exportar para imagem, PDF(*Portable Document Format*) e até um relatório acerca dos elementos no formato *Microsoft Word*.

Em termos de funcionalidades e usabilidade, dá a hipótese de ir construindo o diagrama através de um menu que de contexto em cada elemento, o que potencia o cumprimento das regras BPMN, já que este menu apenas apresenta os elementos BPMN válidos de serem destino de uma ligação a partir do elemento portador desse mesmo menu. Na Figura 3.1 é ilustrado um exemplo deste menu, que aparece após a selecção de um elemento:

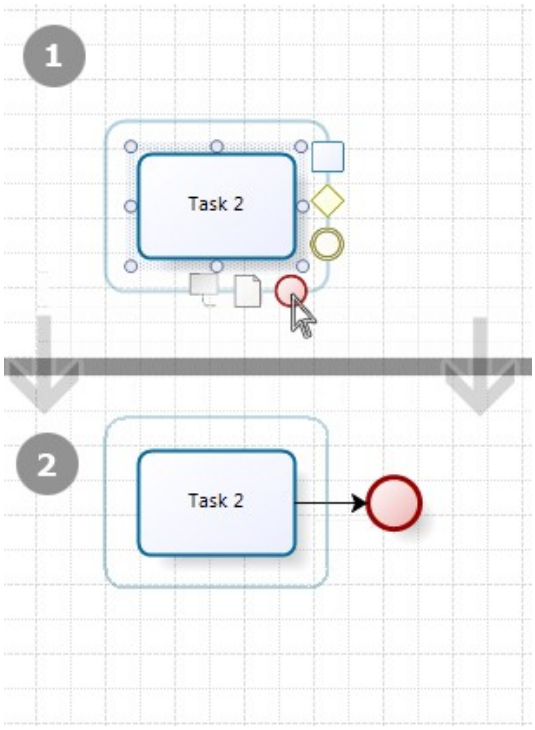

Figura 3.2: *BizAgi BPM Modeler*: Menu de contexto de uma *task*

Como se vê pela figura na Figura 3.1, apenas é permitido que a Task 2 seja ligada a um conjunto restrito de elementos, de acordo com a lógica de desenho BPMN.

3.2.2 Oryx Editor

O *Oryx Editor* é um editor *web-based* para modelos de processo de negócios desenvolvida pelo grupo de pesquisa *Business Process Technology* do Instituto *Plattner* na Universidade de *Potsdam.* [10]

Apresenta diversas funcionalidades, mas no entanto a usabilidade não é a melhor, devido muitas vezes ao excesso de atributos e funcionalidades apresentadas. No entanto esta ferramenta apresenta uma grelha com a listagem dos atributos de cada elemento BPMN a quando da sua selecção, existindo uma opção para os editar, o que de facto é eficiente, porque o utilizador vai tendo sempre presente o valor dos atributos podendo-os alterar, caso estes não estejam em conformidade com o que pretende. Na Figura 3.3, é apresentada a grelha de listagem e edição de atributos de uma *task* na ferramenta *Oryx Editor*:

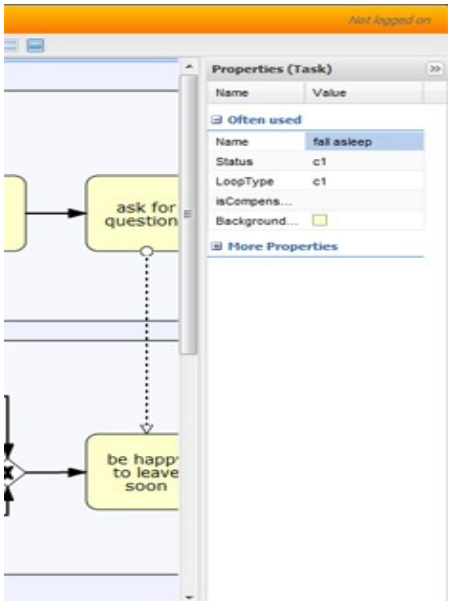

Figura 3.3: Oryx Editor: Grelha de listagem e edição de atributos de um elemento.

3.3 Tecnologias para Desenvolvimento da Aplicação

Tendo em conta a revisão científica apresentada acerca do problema, teremos que encontrar as tecnologias tendo em conta essencialmente 3 factores, os objectivos de cada módulo, a arquitectura de desenvolvimento (MVVM) e a tecnologia da plataforma dCore.

Em primeiro lugar é necessário implementar uma estrutura comum que servirá para validar os requisitos do fluxo como um todo e realizar a persistência de dados do mesmo, que seja reutilizável por todos os módulos da aplicação. A tecnologia que se mostra mais acertada para este tipo de implementação é o Microsoft .NET, onde através do WCF (*Windows Communication Foundation*), se pode desenvolver com relativa facilidade *webservices*, permitindo assim uma reutilização de funcionalidades e um controlo do lado do servidor sobre a lógica destas. Será utilizada através do pacote de ferramentas *Microsoft Visual Studio 2008*.

Para desenvolver o módulo gráfico, as hipóteses mais sonantes seriam a tecnologia AJAX, *Microsoft Silverlight* e a tecnologia da *Macromedia Flash*. Para este módulo era esperado uma fluidez das acções *drag'n'drop*, compatibilidade com a maioria dos navegadores web e um suporte à tecnologia XML para poder operar-se sobre o formato XPDL. Embora a tecnologia *Macromedia Flash* não apresente um suporte directo à arquitectura MVVM, é possível estruturar em camadas toda a lógica da aplicação, e visto que esta é a tecnologia que se mostra mais compatível com o conjunto de navegadores e Sistemas Operativos no mercado, foi a escolhida para implementar este módulo.

Tendo o módulo de parametrização uma componente gráfica mais leve, onde apenas é necessário especificar alguns parâmetros, foi escolhida a utilização do Microsoft .Net com recurso a formulários e a tecnologia AJAX para auxiliar a validação e controlo dos formulários.

Finalmente o módulo de registo de fluxos na plataforma dCore, usará a tecnologia *WPF(Windows Presetation Fundation)* com recurso a c#. Será *windows based* ao contrário dos módulos anteriores pelo facto de se pretender o registo local dos fluxo por parte do gestor dCore. Foi escolhida a tecnologia *microsoft C#* para efeitos de utilização da API dCore de registo dos dados do fluxo.

3.4 Conclusões

Após a análise às tecnologias possíveis para atingir os objectivos a que esta ferramenta se propõe, foi concluído que deveriam ser utilizadas para notação gráfica e persistência de dados, o BPMN e o XPDL respectivamente, já que este conjunto é o que se enquadra melhor no universo de elementos possíveis de existir num fluxo dCore e é o que dará maior suporte à implementação futura de uma consola gráfica para a monitorização dos fluxos, onde será importante a definição dos atributos gráficos de cada elemento.

Foram analisadas duas ferramentas de modelação, o *BizAgi* e *Oryx Editor*, das quais puderam ser extraídos dois mecanismos que poderão potenciar o controlo e usabilidade do dZign, o menu contexto dos elementos BPMN e a grelha de listagem e edição de atributos.

Finalmente foi necessário tomar a decisão quanto às tecnologias a utilizar para o desenvolvimento, para a estrutura base e para o módulo de parametrização foi escolhida a tecnologia Microsoft .NET, para o módulo gráfico a tecnologia *Macromedia Flash* e finalmente para o módulo de registo a tecnologia *Microsoft WPF* com recurso a C#.

4 Análise e Especificação de Requisitos

Neste capítulo são especificados os requisitos necessários para o dZign. A primeira abordagem a fazer, será analisar a estrutura de dado de um fluxo dCore, para que se possa ter uma ideia de como estes se relacionam. Feita esta análise, ficará conhecido o resultado final que se pretende produzir com a utilização do dZign. Como abordagem complementar à analise da estrutura de dados de um fluxo dCore, teremos os casos de uso necessários considerar no dZign, para que possa ficar explicito de que forma é que se pode chegar ao resultado final.

4.1 Requisitos de Negócio

De modo a proceder à definição dos casos de uso e requisitos necessários são apresentados os requisitos de negócio:

- Ferramenta gráfica para desenhar fluxos.
- Geração automática da persistência de dados (gráficos e de parametrização).
- Parametrização das tarefas automáticas e criação de *object types.*
- Fornecer a opção de apenas criar *object types.*
- Validação e registo.
- Divisão em passos de acordo com os perfis de utilizadores necessários para concluir todo o processo.
- Enquadramento do dZign com a plataforma dCore.

4.2 Requisitos Funcionais

Na especificação dos requisitos funcionais, interessa primeiro abordar aqueles que são indispensáveis. De seguida é apresentada a visão sobre os casos de uso, primeiro da definição de fluxos e em segundo do registo na plataforma dCore.

4.2.1 Definição de Fluxos

Na Figura 5.1, é apresentado o diagrama de casos de utilização geral:

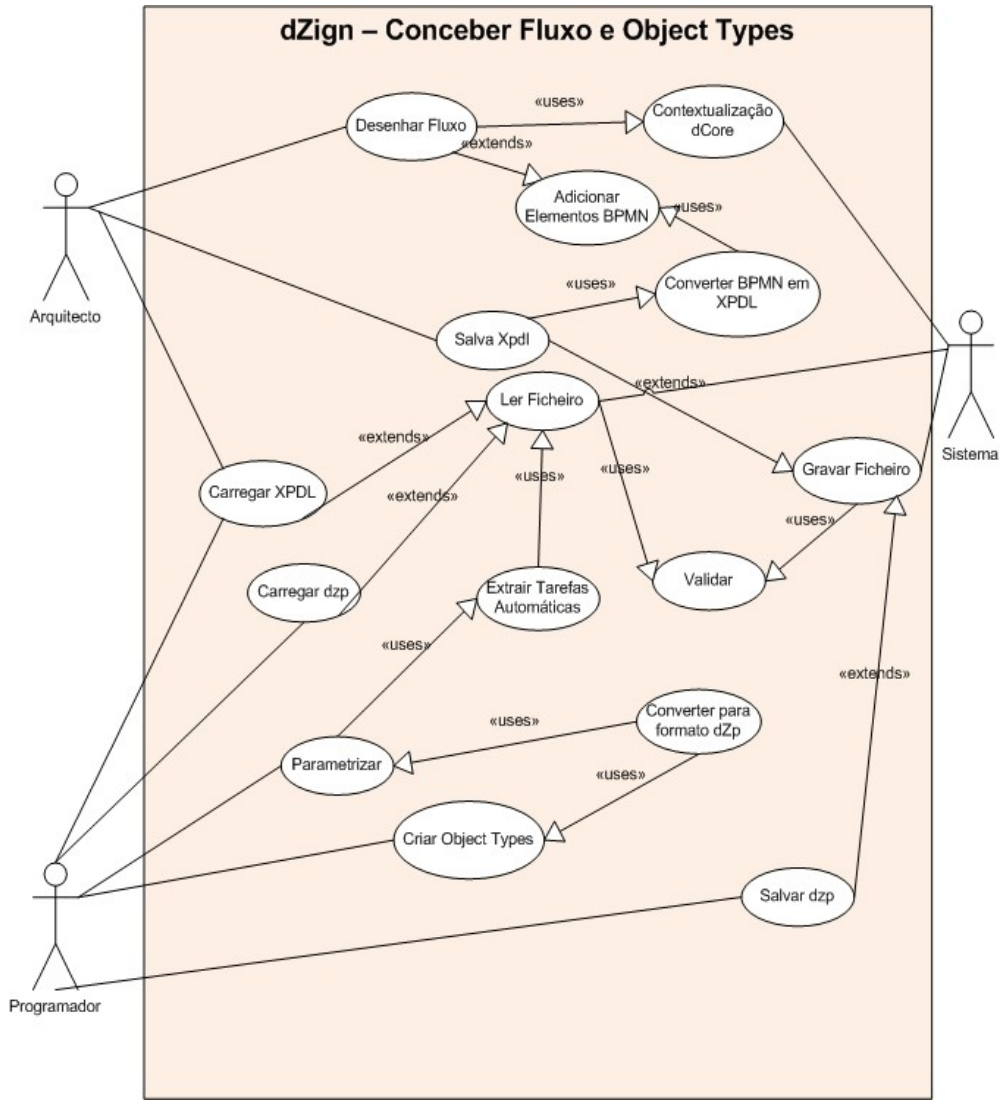

Figura 4.1: Diagrama de casos de uso de definição de fluxos

4.2.1.1 Actores

Como tinha sido referido anteriormente, os actores humanos que interagem neste processo , serão os arquitectos de fluxo, que possuem uma ideia e visão geral do que se pretende construir, e os programadores, que apresentam conhecimentos de programação e que indicarão os serviços a executar nas *tasks* automáticas e os *object types*.

Por outro lado como actor não humanos, temos o sistema que é intermediário das acções do utilizador com a persistência dos dados.

4.2.1.2 Casos de Utilização

Tabela 4.1: Caso de Utilização UC01-Desenhar Fluxo

Identificador	UC01
Nome	Desenhar Fluxo
Descrição	Através de uma menu com os elementos necessários o utilizador combina elementos BPMN sobre um *canvas*, para que possa representar o fluxo desejado.
Cenário	Após a adição de cada elemento até definir todo o fluxo, o sistema efectua um controlo para assegurar que as regras são aplicadas (caso de utilização UC2). No fim do desenho do fluxo e quando este é submetido para gravar (UC03) o sistema valida o fluxo e devolve uma mensagem *feedback*.
Actores	Arquitecto de Fluxos
Prioridade	Elevada
Pré-Condições	Execução do UC02 e a adição de elementos BPMN
Pós-Condições	Terão que ser respeitadas as regras BPMN e da plataforma dCore, para que o fluxo possa ser gravado e tenha utilidade.
Pressupostos	N/A

Tabela 4.2: Caso de Utilização UC02-Contextualização dCore

Identificador	UC02
Nome	Contextualização dCore
Descrição	Através de regras existente, o sistema contextualiza os elementos BPMN presentes na área de desenho do fluxo, para evitar erros de contextualização dos mesmos.
Cenário	O elemento é submetido a uma validação de controlo em tempo de execução e caso este não respeite o contexto pretendido para o fluxo, é accionada uma acção de contextualização (ex: *task* entre duas *lanes* é “arrastada” para uma delas)
Actores	Sistema
Prioridade	Média
Pré-Condições	N/A
Pós-Condições	O utilizador tem que adicionar ou editar um elemento BPMN.
Pressupostos	Pressupõe-se que seja realizado no lado do cliente para que tenha de facto utilidade, não sendo assim confundido com o processo de validação.

Tabela 4.3: Caso de Utilização UC03-Salvar XPDL

Identificador	UC03
Nome	Salvar XPDL
Descrição	É accionada através de um menu uma acção pelo utilizador com o

	intuito de guardar o fluxo no servidor em formato XPDL.
Cenário	O servidor recebe ordens por parte do actor para gravar o fluxo no disco, tendo que este ser validado antes de ser persistido. Caso seja bem sucedido retorna uma mensagem de sucesso, caso contrário retorna uma mensagem de erro.
Actores	Arquitecto de Fluxo
Prioridade	Elevada
Pré-Condições	Existir a estrutura XPDL
Pós-Condições	Que haja uma estrutura XPDL do fluxo produzido e seja executado o requisito UC09.
Pressupostos	N/A

Tabela 4.4: Caso de Utilização UC04-Carregar XPDL

Identificador	UC04
Nome	Carrega XPDL
Descrição	É enviado um pedido para carregar um determinado ficheiro XPDL para o sistema, para que o sistema execute o carregamento de um ficheiro UC10.
Cenário	Caso o ficheiro seja carregado com sucesso é accionado um mecanismo de conversão dos dados de persistência do fluxo em elementos BPMN e o fluxo é apresentado ao utilizador, sendo que a partir deste ponto pode ser editado. Caso contrário é devolvida uma mensagem de erro, descrevendo o problema que levou à impossibilidade de carregar o fluxo.
Actores	Arquitecto de fluxo e Programador
Prioridade	Média
Pré-Condições	Existência de uma lista de ficheiros XPDL disponíveis no servidor.
Pós-Condições	É necessário o XPDL ser válido
Pressupostos	N/A.

Tabela 4.5: Caso de Utilização UC05-Parametrizar

Identificador	UC05
Nome	Parametrizar
Descrição	O actor preenche as caixas de texto necessárias para indicar os serviços para as tarefas automáticas, especificando assim os atributos necessários.
Cenário	Ao submeter a parametrização, caso os serviços não infrinjam nenhuma regra é gerado um ficheiro dzp, contendo estas especificações caso contrário são geradas mensagens de feedback indicando os erros presentes nesta especificação.
Actores	Programador
Prioridade	Elevada

Pré-Condições	É necessário que tenha sido carregado um fluxo previamente desenhado, para que se consiga extrair as tarefas automáticas e gerar os campos de parametrização.
Pós-Condições	É requerido que todas as *tasks* parametrizadas correspondam às presentes no fluxo.
Pressupostos	Todas as tarefas automáticas deverão ser parametrizadas e caso não existam tarefas automáticas, a acção de criação de um dzp é ainda assim obrigatória, apesar de na prática não existir parametrização.

Tabela 4.6: Caso de Utilização UC06-Criar Object Types

Identificador	UC06
Nome	Criar *Object Types*
Descrição	O actor através de caixas de texto vai adicionando object types à memória do módulo de parametrização. Estes objecct types têm que cumprir requisitos em termos de caracteres permitidos, e não podem ter campos vazios (Name, Assembly Name e Class Name).
Cenário	Após a submissão de um novo *object type* e caso este seja válido é apresentado numa tabela com opção de edição e remoção.
Actores	Programador
Prioridade	Elevada
Pré-Condições	Não existe na memória do módulo de parametrização, outro *object type* com o mesmo nome.
Pós-Condições	N/A
Pressupostos	N/A

Tabela 4.7: Caso de Utilização UC07-Salvar dzp

Identificador	UC07
Nome	Salvar dzp
Descrição	É accionada através de um menu uma acção pelo utilizador com o intuito de gerar o ficheiro DZP.
Cenário	O servidor recebe ordens por parte do actor para gravar a parametrização do fluxo e/ou *object types* no disco, tendo que este ser validado antes de ser persistido. Caso seja bem sucedido retorna uma mensagem de sucesso, caso contrário retorna uma mensagem de erro.
Actores	Programador
Prioridade	Elevada
Pré-Condições	Existir a estrutura DZP
Pós-Condições	Que haja uma estrutura XPDL do fluxo produzido e seja executado o requisito UC
Pressupostos	N/A

Tabela 4.8: Caso de Utilização UC08-Desenhar Fluxo

Identificador	UC08
Nome	Carregar dzp
Descrição	É enviado um pedido para carregar um determinado ficheiro DZP para o sistema, para que o sistema execute o carregamento de um ficheiro UC10.
Cenário	Caso o ficheiro seja carregado com sucesso existem dois cenários possíveis, se este DZP estiver associado a um XPDL é pedido ao servidor que carregue também o XPDL,caso contrário é sinal que se está perante uma definição apenas de *object types*. A aplicação trata de traduzir e guarda em memória todos os dados carregados.
Actores	Arquitecto de fluxo e Programador
Prioridade	Média
Pré-Condições	Existência de uma lista de ficheiros DZP disponíveis no servidor.
Pós-Condições	É necessário o DZP ser válido e coerente com um XPDL no caso de estar associado com algum.
Pressupostos	N/A.

Tabela 4.9: Caso de Utilização UC09-Salvar Ficheiro

Identificador	UC09
Nome	Salvar Ficheiro
Descrição	É executada uma operação de gravação de dados no disco provenientes da função que encovou este serviço.
Cenário	Em caso de sucesso retorna uma mensagem de operação concluída com êxito, caso contrário é retornado um erro de gravação
Actores	Sistema
Prioridade	Elevada
Pré-Condições	Apenas se aceitam pedidos de gravação nos formatos XPDL ou DZP.
Pós-Condições	
Pressupostos	Um dos actores humanos accionou um pedido para salvar um ficheiro num dos formatos possíveis(XPDL ou DZP)

Tabela 4.10: Caso de Utilização UC10-Carregar Ficheiro

Identificador	UC10
Nome	Carregar Ficheiro
Descrição	É executada uma operação de leitura de dados no disco provenientes da função que encovou este serviço.
Cenário	Em caso de sucesso retorna uma mensagem de operação concluída com êxito, caso contrário é retornado um erro de gravação
Actores	Sistema
Prioridade	Elevada
Pré-Condições	Apenas se aceitam pedidos de carregamento de ficheiros nos formatos

	XPDL ou DZP e o processo de validação tem que ter um resultado positivo.
Pós-Condições	N/A
Pressupostos	Um dos actores humanos accionou um pedido para carregar um ficheiro num dos formatos possíveis(XPDL ou DZP)

4.2.2 Registo dos Fluxos na plataforma dCore

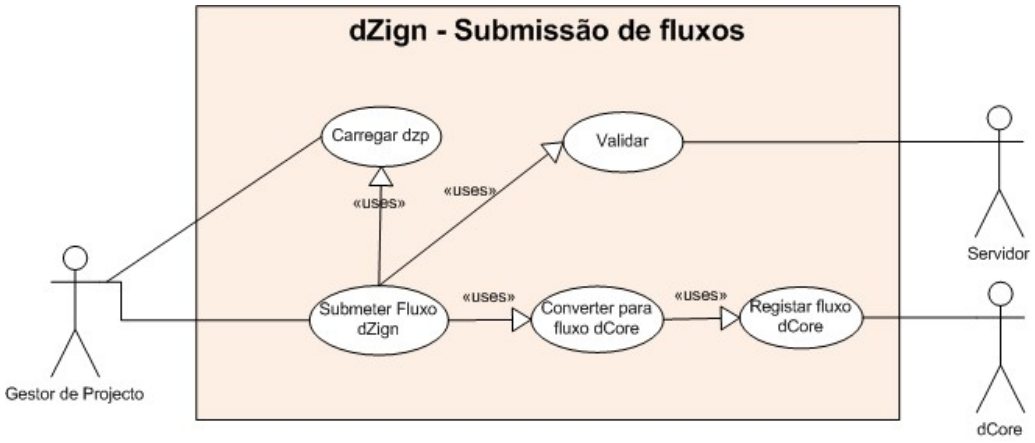

Figura 4.2: Diagrama de casos de uso de submissão de fluxos

4.2.2.1 Actores

No presente caso de uso, apenas é necessária a presença de um actor, o gestor de projecto, que não tem que necessariamente possuir nenhum conhecimento técnico em concreto, mas que tipicamente deverá ser o administrador do servidor.

Para por em prática este caso de uso é necessário a presenta de dois actores não humanos, o sistema dZign, e o dCore.

4.2.2.2 Casos de Utilização

Tabela 4.11: Caso de Utilização UC11-Desenhar Fluxo

Identificador	UC11
Nome	Carregar dzp
Descrição	É enviado um pedido para carregar um determinado ficheiro DZP para o sistema, para que o sistema execute o carregamento de um ficheiro UC10.
Cenário	Caso o ficheiro seja carregado com sucesso existem dois cenários possíveis, se este DZP estiver associado a um XPDL é pedido ao

	servidor que carregue também o XPDL,caso contrário é sinal que se está perante uma definição apenas de *object types*. A aplicação trata de traduzir e guarda em memória todos os dados carregados.
Actores	Arquitecto de fluxo e Programador
Prioridade	Média
Pré-Condições	Existência de uma lista de ficheiros DZP disponíveis no servidor.
Pós-Condições	É necessário o DZP ser válido e coerente com um XPDL no caso de estar associado com algum.
Pressupostos	N/A.

Tabela 4.12: Caso de Utilização UC12-Contextualização dCore

Identificador	UC12
Nome	Submeter Fluxo dZign
Descrição	O utilizador invoca a acção de submeter um fluxo, sendo esperado que que haja uma conversão dos dados em memória para o formato dCore se proceda à sua submissão para o repositório dCore.
Cenário	Após a submissão é retornada a mensagem de *feedback*, indicando se a operação foi executada com sucesso ou não.
Actores	Gestor de Projecto
Prioridade	Elevado
Pré-Condições	Foi carregado um ficheiro dZP e o fluxo bem como os object types encontram-se validados.
Pós-Condições	N/A
Pressupostos	Foi indicado o endereço para o repositório de dados.

Tabela 4.13: Caso de Utilização UC13-Validar dzp

Identificador	UC13
Nome	Validar dzp
Descrição	O sistema valida o fluxo de acordo com as regras que têm que ser cumpridas. Estas regras são afectadas pela notação BPMN, pelo formato XPDL e pelos requisitos do fluxos dCore.
Cenário	Caso a estrutura do fluxo esteja válida, o sistema retorna um sinal de que se reuniram as condições para proceder ao registo do fluxo, caso contrário é retornado um *feedback* para o utilizador contendo os eventuais erros.
Actores	Sistema
Prioridade	Elevada
Pré-Condições	Foi submetido um fluxo para o dCore e o sistema dZign é intermediário.
Pós-Condições	N/A
Pressupostos	N/A

Tabela 4.14: Caso de Utilização UC14-Registar fluxo dCore

Identificador	UC14
Nome	Registar fluxo dCore
Descrição	É uma acção executada pela API do dCore, que visa registar os fluxos e *object types* no seu repositório de dados. Não interessa qual é o repositório já que esta API tem mesmo o intuito de o abstrair.
Cenário	Caso a o fluxo seja correctamente submetido, é retornada a mensagem do seu sucesso, caso contrário é retornada uma mensagem com as causas do seu insucesso.
Actores	dCore
Prioridade	Elevada
Pré-Condições	O fluxo ou *object types* reúnem os requisitos necessários para serem submetidos.
Pós-Condições	N/A
Pressupostos	N/A

4.3 Requisitos Não Funcionais

4.3.1 Modularidade

Devido à necessidade da existência de dois perfis de utilizadores, no processo de definição de um fluxo, como foi referido anteriormente e à natureza do registo dos dados na plataforma dCore, o dZign deverá ser dividido em 3 diferentes módulos, sendo que é necessário obviamente a existência de uma base de contextualização no caso dos módulos "*web-based*", para que não haja discrepância acentuada entre o visual destes e para que possa existir navegação entre eles. Este módulos são, por ordem de execução no que diz respeito à definição do fluxo, os seguintes:

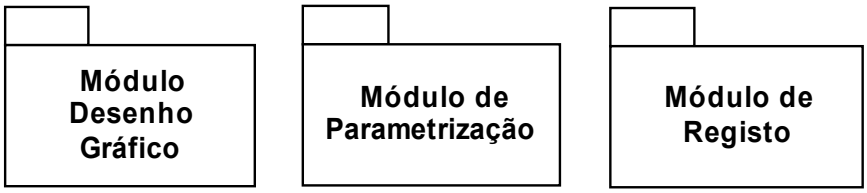

Figura 4.3: Módulos da ferramenta dZign

4.3.2 Língua

Devido ao facto de existir a forte possibilidade de esta ferramenta ser disponibilizada de forma gratuita no mercado e às perspectivas existentes de a DOCKS se expandir

internacionalmente, é requisito que as ferramentas produzidas tenham os seus interfaces na língua inglesa, assim como as variáveis utilizadas na implementação, para que a utilização e manutenção desta, não se limite apenas aos colaboradores de língua Portuguesa.

4.3.3 Simplicidade

Tendo em vista a fase de operação do dZign, é esperado que o sistema seja simples quanto ao número de elementos e funcionalidades que apresenta, com o intuito de evitar distracções e confusões mantendo assim o utilizador concentrado no desenvolvimento do fluxo.

A simplicidade também é um requisito em termos de substituição de tecnologias de suporte à modelação, sendo necessário arquitectar a solução de forma a prever uma eventual troca da dessas tecnologias, evitando primeiro que o dZign arrisque o comprometimento na totalidade da sua utilidade, no caso de uma das tecnologias não cobrir o universo de elementos de um fluxo dCore caso seja expandido, e prevendo o caso de uma destas tecnologias ser descontinuada.

4.3.4 Manutenção

O dZign deverá ser arquitectado tendo em conta a manutenção e tendo em vista também a extensibilidade/expansibilidade. Sendo imprevisível o rumo e ritmo de evolução do dCore, interessa assegurar que o dZign seja o mais independente possível, com excepção por questões óbvias da sua estrutura de dados.

O primeiro aspecto importante da manutenção, tem a ver com o facto de não interessar que o dZign se encontre “limitado” ao repositório de dados actual do dCore, que se preveja a existência de uma mudança de formato. Para este efeito deverá existir uma abstracção deste repositório.

O segundo aspecto importante, será a troca de tecnologias de suporte à definição de fluxos, neste caso BPMN e XPDL, caso uma destas tecnologias seja ultrapassada ou deixe de ser útil no contexto dCore, permitir a sua troca sem evolver a restante estrutura.

Por fim, também deverá ser levada em conta a extensão do dZign a novos elementos. O BPMN de facto já dispõe de uma panóplia de elementos que poderão ser úteis no futuro, mas é necessário sustentar a possível adição destes elementos na arquitectura dos dados.

4.3.5 Usabilidade

A usabilidade, facilidade com que os utilizadores usarão o GUI do sistema, assume um papel de extrema importante no dZign, especialmente no que diz respeito ao módulo de desenho gráfico, devido ao facto de ser uma ferramenta de desenho de fluxos, onde têm que ser cumpridas algumas regras para que estes estejam válidos. De seguida é apresentada a lista de requisitos para o módulo de desenho gráfico:

- Fornecer os elementos BPMN no contexto oportuno de aplicação
- Controlar de uma forma lógica e não “agressiva”, as acções do utilizador para que se minimize o número de erros no momento de submissão e consequente validação do fluxo.
- Fornecer mensagens de *feedback* em resposta às acções do utilizador, sempre que se justifique, para que este tenha noção do sucesso ou não das mesmas.

- Potenciar a visão geral sobre o fluxo a ser desenhado, ou seja, garantir no mínimo a opção do utilizador “navegar” por todo o fluxo.

O módulo de parametrização, não necessita de um controlo tão especifico como o módulo de desenho gráfico, visto que os dados que se espera que o utilizador introduza são meramente textuais, um caso típico de utilização de formulários para a submissão de dados. De seguida é apresentada a lista de requisitos para o módulo de desenho gráfico:

- Dado que aceita ambos os formatos dos ficheiros de definição de um fluxo (XPDL e DZP) , e que para um mesmo fluxo poderão corresponder dois ficheiros, utilizar o conceito de projecto, apresentando apenas o nome do fluxo.
- Fornecer mensagens de *feedback* às acções do utilizador, pós-submissão e pré-submissão (sempre que possível), no que diz respeito ao carregamento de projectos, preenchimento dos parâmetros quer de um *object type* quer dos *services* e submissão dos dados, para que o utilizador tenha noção do sucesso ou não das suas acções e para que possam ser reparados eventuais erros.

4.3.6 Segurança

A fim de evitar utilizações mal intencionadas, garantir confidencialidade e evitar mesmo erros do utilizador que poderão comprometer a coerência dos dados submetidos para o dCore, para além de existir o processo de validação, deverá-se assegurar que apenas no contexto do dZign serão lidos e editados os ficheiros de persistência de dados relativos aos fluxos.

4.3.7 Módulos *Web-Based*

Devido ao facto de o interface da plataforma dCore estar previsto ser desenvolvido em tecnologia web e afim de integrar o dZign nesse contexto, foi tido em conta o requisito de sustentar os módulos de definição de um fluxo no mesmo tipo de tecnologia.

4.4 Resumo e Conclusões

Neste capítulo foram especificados os requisitos funcionais e não funcionais do dZign, partindo da estrutura dos fluxos dCore, para que fossem determinados os requisitos dos dados a quando submetidos no repositório dCore. A partir desta análise e da revisão científica, foram especificados os requisitos funcionais, ilustrando-os com os casos de uso e os requisitos não funcionais.

5 Concepção

Este capítulo tem o objectivo de especificar a arquitectura utilizada para implementar a solução pretendida bem como os detalhes da implementação. Durante a fase de desenvolvimento, o dZign contou com a colaboração de um designer para tratar da aparência da aplicação. Embora por motivos profissionais e de custos associados, essa colaboração não possa ter sido a tempo inteiro, foi bastante útil para discutir alguns pormenores de usabilidade e ideias para implementar no futuro, que certamente poderão ser úteis para facilitar o trabalho dos arquitectos de fluxo.

5.1 Arquitectura de Dados dCore

Devido ao facto de o dZign se apresentar para o dCore como uma ferramenta de definição de fluxos, é importante verificar em primeira instância a estrutura dos elementos que os compõe e aferir a natureza da relação entre esses elementos. Com esta especificação presente, atingem-se as condições para encontrar os elementos da notação gráfica, que servirão para a sua representação.

O fluxo é constituído por um conjunto de elementos que apresentam uma relação hierárquica e que têm uma relação entre si que muitas vezes implica alguns condicionamentos.

É apresentado na Figura 5.2 o diagrama de classes dos fluxos aceites pelo dCore.

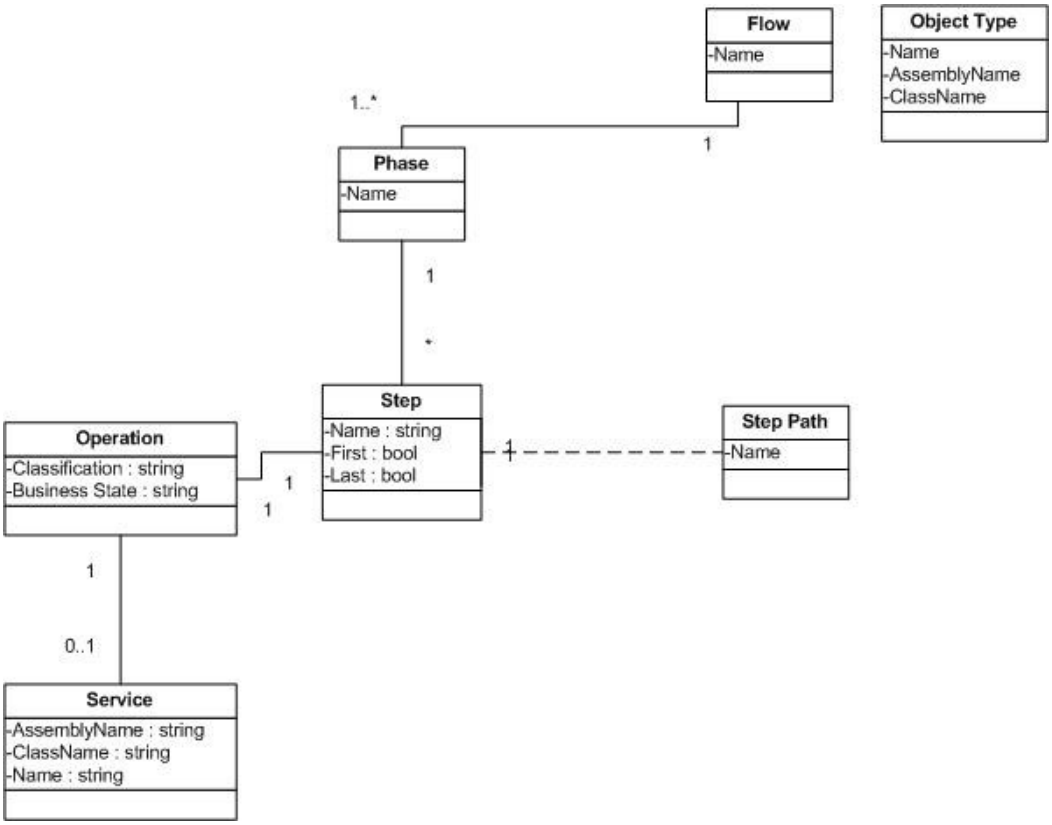

Figura 5.1: Diagrama de Classes do dCore

A observação do diagrama de classes permite deduzir alguns requisitos que são necessários levar em conta no planeamento e desenvolvimento do dZign, no entanto é preciso ter presente também a natureza de algumas relações entre as classes, que são particulares quanto ás suas exigências e a natureza dos atributos presentes em cada uma delas.

5.1.1 Especificação das Classes e seus Atributos

É apresentado na Tabela 5.1 a descrição dos atributos de cada classe e são traçadas da mesma forma considerações acerca da relação com outros elementos.

Tabela 5.1: Descrição das classes dCore

Nome:	Atributos	Descrição
Flow	Id (obrigatório) Name (obrigatório)	A classe *flow* como o próprio nome indica é a representação do fluxo, para além dos elementos esta estrutura deverá conter obrigatoriamente, os atributos apresentados na coluna à esquerda. Possui relação directa com os *object types* que poderá aceitar e as *phases* que contém.
Phase	Id (obrigatório)	A classe *phase* representa um “aglomerado” de

	Name (obrigatório)	*steps* e *steps paths*, e foi concebida para que através de serviços automáticos, se possa saltar para outro phase.
Step	Id (obrigatório) Name (obrigatório)	A classe *step* representa um conjunto operação/destinos possíveis. Tem relação directa com os *step paths* e *operations.* Pode ser classificada como inicial, sendo que esta classificação apenas pode ocorrer num *step* por *phase*. Pode também ser classificada como final, sendo que neste caso não existe um mínimo ou máximo de *steps* com esta classificação . Os *steps* podem ser executados em múltipla instâncias.
Step Path	Id (obrigatório) Name	A classe *step path* , representa uma transição, definida por um conjunto *step* de origem/ *step* de destino. Tem relação directa com os *steps* que pretende unir.
Operation	Business State Category (obrigatório/atributo lógico)	A classe *operation*, representa a estrutura responsável pela execução de uma acção sobre o(s) objecto(s) que circulam no fluxo. Apresenta um atributo categoria que é obrigatório e que caso seja automático terá que ser sujeito à especificação de um serviço para a sua execução no passo de parametrização.
Service	Name (obrigatório) Assembly Name(obrigatório) Class Name (obrigatório)	A classe *service*, representa a estrutura necessária implementar para que uma dada operação seja automática. Com o dZign não se pretende implementar essa estrutura, mas sim especificar os dados dessa implementação para que o dCore tenha as informações necessárias para a encontrar.
Object Type	Name (obrigatório) Assembly Name (obrigatório) Class Name (obrigatório)	A classe *object type*, representa a estrutura necessária para estar definido um tipo de objecto implementado com tecnologias de programação, aceite por um dado fluxo na plataforma.

5.2 Representação dos Dados

Quais os elementos BPMN que servirão para representar um fluxo do dCore?

A resposta a esta pergunta foi um passo importante no projecto, a representação da classe *flow* já estava garantida com o conceito de BPD que consiste no conjunto elementos BPMN, tendo dois atributos, nome e versão que serão obrigatórios preencher segundo os requisitos dCore.

A *phase* tem características que se assemelham a um processo de negócio, e um processo de negócio na notação BPMN é representado por uma *Pool.*

Tabela 5.2: Paralelismo *Phase → Pool*

Phase → Pool

Atributo Phase no dCore	**Atributo da Pool no BPMN**
Id	Id
Name	Name

Um *step* é o conjunto formado por uma *operation* e *steps* possíveis de alcançar, segundo a verificação de determinadas condições (*step path).* No BPMN não existe um elemento que represente rigorosamente um *step*, no entanto, existe um paralelismo entre *operation*, elemento atómico de um fluxo dCore, e *step path* um caminho entre steps. Estes elemento são respectivamente a *task*, elemento atómico de um diagrama BPMN com características paralelas à operation e *step*, e o *sequence flow*, conector responsável pelo sequenciamento fluxos.

Tabela 5.3: Paralelismo *Step*/*Operation* → Task

***Step*/*Operation* → Task**	
Atributo Step/Operation no dCore	**Atributo da Task no BPMN**
Id (Step/Operation)	Id
Name (Step)	Name
Category (Atributo lógico da *Operation*)	Lanes
Business State (Operation)	Extended Attribute com nome Operation BusinessState.
Multiple Instance (Atributo lógico da *Operation*)	Multiple Instance {Sequential or Parallel}
Start Step (Step)	Proveniente de um Start Event
End Step (Step)	Convergente para um End Event

Tabela 5.4: Paralelismo *Step Path*→ Flow Connector

***Step Path* → Flow Connector**	
Atributo *Step Path* no dCore	**Atributo do Flow Connector no BPMN**
Id	Id
Name,Condition	Name,Condition
From	Campo Id do elemento de origem
To	Campo Id do elemento de destino

Na Figura 5.1, é representado o formato gráfico dos elementos escolhidos:

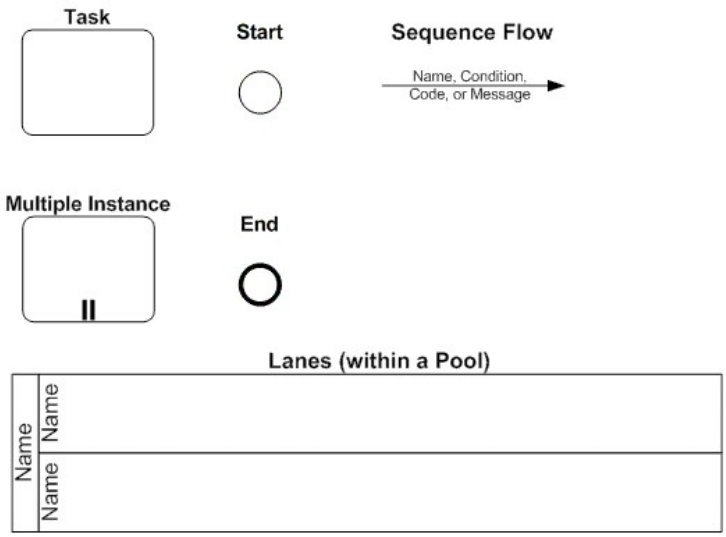

Figura 5.2: Elementos BPMN escolhidos

Após a escolha dos elementos BPMN, para representar os elementos de um fluxo dCore, é importante primeiro existir uma diferenciação em termos de utilidade. Por um lado temos a *task, sequence flow connector e a pool*, dos quais vão resultar directamente elementos de um fluxo dCore. Por outro lados temos o s*tart event* e o e*nd event,* que apenas servirão para indicar se um *step* é categorizado de *start step e/ou end step*. Finalmente teremos talvez a decisão mais controversa em termos de utilização BPMN, mas consensual no seio da equipa de desenvolvimento da DOCKS, a categorização das tarefas (automática, semi-automática ou manual) através de *lanes*, quando o BPMN possui mecanismos directos de categorização das tarefas (automatic tasks, user tasks e manual tasks). Esta escolha é justificada pelo facto de estas categorizações serem obrigatórias, existindo uma necessidade desta acção se tornar intuitiva, e sendo as *lanes* contentores passíveis de utilização no que toca a atribuição de papeis, não se infringe nenhuma regra relativa à notação BPMN.

5.2.1 Arquitectura MVVM

A arquitectura MVVM (*Model-View-ViewModel*)[4] é um standard especificado pela *Microsoft*, que surgiu da necessidade de abstrair o interface gráfico do modelo de persistência de dados. Este modelo defende a criação de três camadas:

- View: Camada de interface para o utilizador;
- ViewModel: Camada de conversão de dados entre o View e o Model;
- Model.: Camada de acesso de dados e lógica de negócio;

Na figura Figura 5.3 é ilustrado como funciona este padrão de arquitectura:

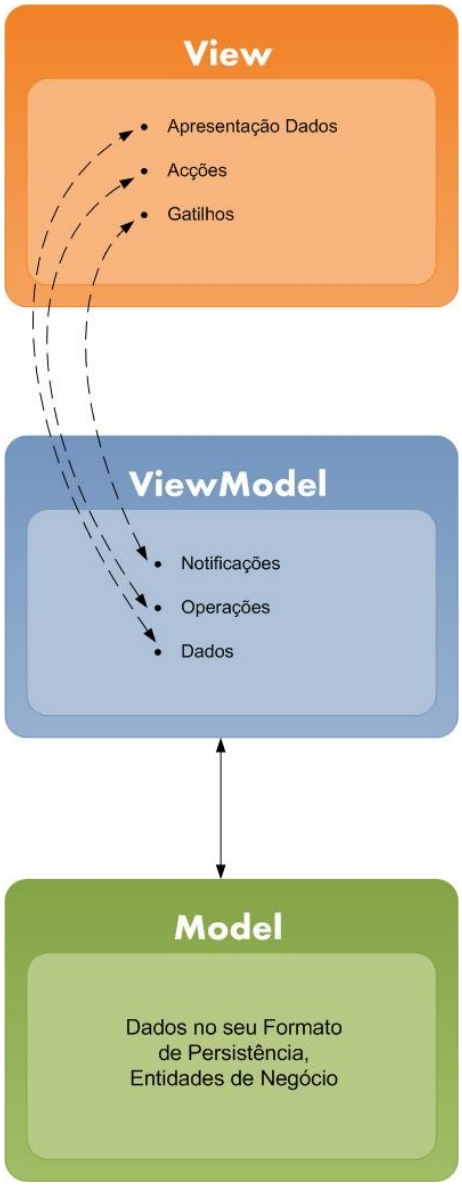

Figura 5.3: Diagrama das camadas *Model View ViewModel*

No que diz respeito ao dZign, este padrão de arquitectura vai trazer flexibilidade, pois permite mudar a notação gráfica sem afectar o formato em que é realizada a persistência de dados e vice-versa. Na prática possibilitando a modificação do ecrã de edição sem grandes impactos para a camada de negócio.

5.3 Arquitectura Proposta

Para que o dZign alcance o estatuto de uma ferramenta de modelação gráfica simples e extensível/expansível, será necessário utilizar uma arquitectura que por um lado tenha camadas bem definidas quanto às suas funções, para que no futuro não seja necessário alterar toda a estrutura da aplicação e por outro proceder à implementação de uma camada lógica que

assegure a interoperabilidade entre estas. Na Figura 5.2, é apresentado um diagrama da arquitectura geral do dZign, tendo em conta a existência dos referidos interfaces:

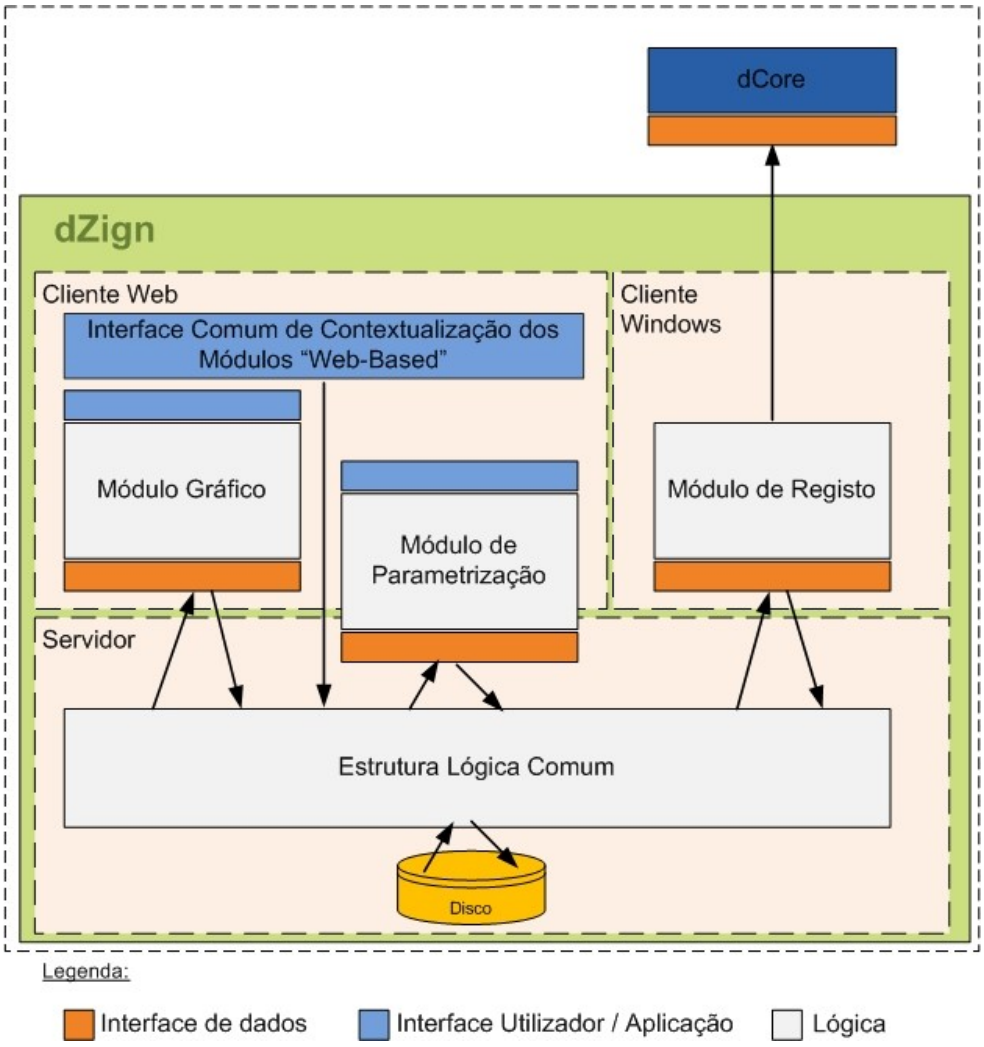

Figura 5.4: Arquitectura Geral

Sendo que o sistema foi arquitectado tendo em conta a existência dos interfaces gráficos e de dados, é necessário existir uma estrutura lógica que interaja entre estas duas camadas., para que se implemente a arquitectura MVVM.

No caso do módulo de desenho gráfico o interface gráfico converte as acções do utilizador em elementos BPMN apresentado-as em tempo real ao mesmo, enquanto que o interface de dados converte o formato XPDL em BPMN. Para que todas estas acções sejam executadas, segundo os três contextos existentes, BPMN, XPDL e dCore, é necessário que exista um controlo das acções do utilizador e uma validação por parte do cliente para que não se sobrecarregue os pedidos ao servidor.

O módulo de parametrização, apresenta igualmente um interface gráfico, no entanto mais simples que o anterior, que consiste em caixas de texto para introduzir informações necessárias dos *object types* que se pretende adicionar e/ou os parâmetros necessários de especificar para cada tarefa automática. Existe também por parte desta camada, a acção de apresentar os dados

que vão surgindo ao utilizador, dando a possibilidade ao mesmo de os editar ou eliminar, este último no caso dos *object types*.

No interface de dados deste módulo, é esperado converter os dados armazenados na memória da aplicação no formato “dzp”, para que possam ser editados posteriormente ou submetidos a registo, ou na operação inversa carregar os dados de um “dzp” e proceder à sua transformação no formato que foi concebido para os armazenar na memória.

Na figura 5.1 o módulo de registo não apresenta um interface gráfico devido ao facto de este não se encontrar explicito em termos de arquitectura, pois este módulo é simples apenas consiste na validação dos projectos criados e respectiva submissão dos dados à plataforma dCore.

5.4 Base Comum

Para que exista uma integração dos diferentes módulos é necessário construir um interface que os contextualize. Dado que se atribui algum interesse na possibilidade de realizar o *upload* de desenho de fluxos, para promover a interoperabilidade das ferramentas, já que o BPMN e o XPDL são padrões, e que esta acção não depende nem pode ser contextualizada em qualquer um dos dois módulos *web-based* (o utilizador pode querer editar o desenho do fluxo ou passar directamente à sua parametrização), optou-se por incluir esta funcionalidade neste interface comum.

Apesar do processo de definição de fluxos estar dividido em módulos, o fluxo no final deverá ser visto como um todo e como tal no final a validação deverá ser feita sob essa perspectiva. No entanto para existir coerência entre os seus passos de definição, este fluxo deverá ser sujeito a validações intermédias. De modo às funções serem aproveitadas entre estes módulos foi optado pela utilização de *webservices* WCF *rest* como tecnologia para a implementação da persistência e validação dos dados.

5.4.1 Menu

Para que os módulos *web-based* sejam utilizados de forma correcta, tem que existir um menu que os leve a executar de forma sequencial. Outra das funcionalidades que este módulo deve prever será o upload de desenhos de fluxos, devendo aparecer um botão que quando pressionado, acciona um *popup* com um formulário de submissão de ficheiros no formato XPDL.

5.4.2 Definição de Dados do Lado do Servidor

Para sustentar os requisitos de validação e segurança no que toca à persistência de dados e o aproveitamento das funcionalidades comuns será implementada uma estrutura comum acedida através de *webservices.* Esta estrutura foi pensada como uma hierarquia, no topo temos a classe projecto que abarca a classe *flow* e um conjunto de classes *object types*, a classe *flow* por sua vez irá conter o conjunto de elementos que a constituem, sendo que a task irá conter o serviço que a implementa caso seja automática. Na Figura 5.2, é apresentado o diagrama de classes, correspondente à estrutura comum:

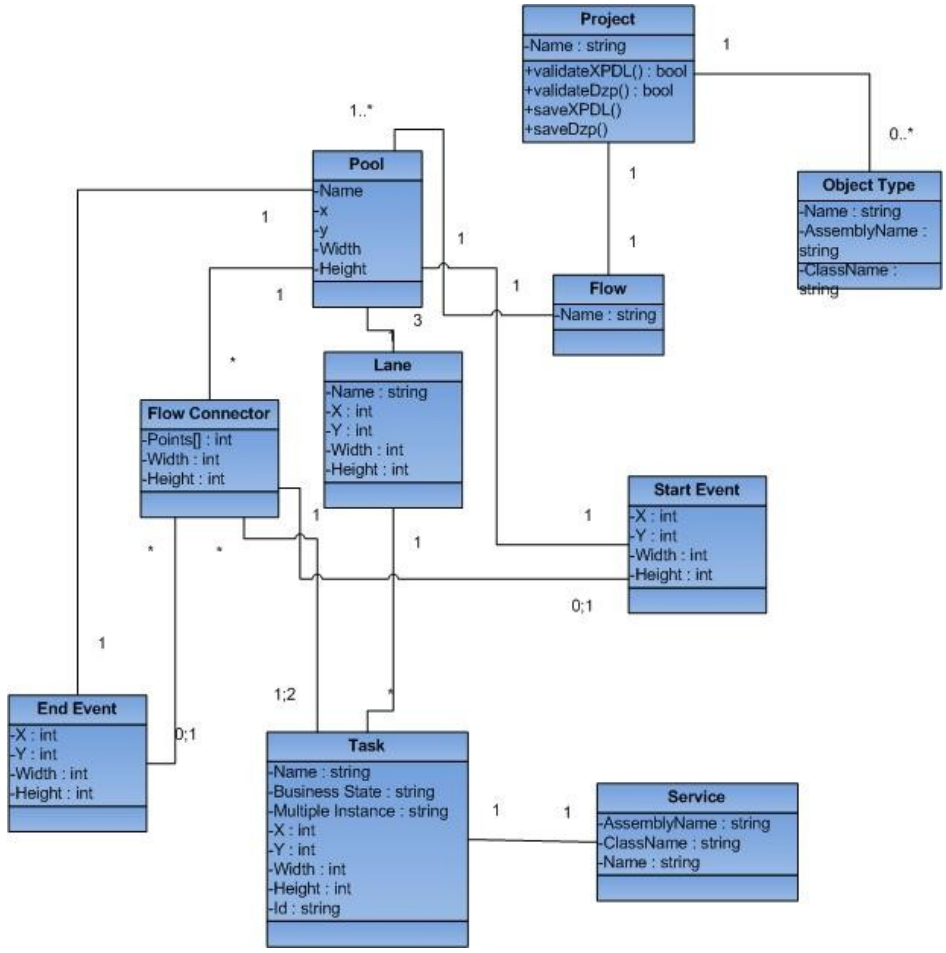

Figura 5.5: Diagrama de classes da estrutura comum

5.5 Arquitectura dos Módulos

5.5.1 Módulo de Desenho Gráfico

De seguida são apresentados o protótipo de interface gráfico deste módulo e o diagrama de classes usados para representar os dados necessários para o implementar.

5.5.1.1 Protótipo do interface

O GUI do módulo de desenho gráfico requer cuidados especiais, pela sua natureza e tecnologia de implementação. Tratando-se de uma ferramenta de desenho de fluxos, e obedecendo a um conjunto de regras inerentes à notação gráfica BPMN em conjunto com as regras dos fluxos dCore, é necessário existirem mecanismos de facilitação e controlo. Se tivermos também em conta a tecnologia que é utilizada para implementar este módulo, sabe-se

que apresenta um universo vasto de possibilidades quanto à construção de componentes e aparência visual dos mesmos, o que apesar de ser uma vantagem em termos de interactividade pode-se tornar um factor de risco, caso não apresentem um desenho que potencie a funcionalidade pretendida.

Na Figura 5.2, é apresentado o protótipo GUI do modulo visual:

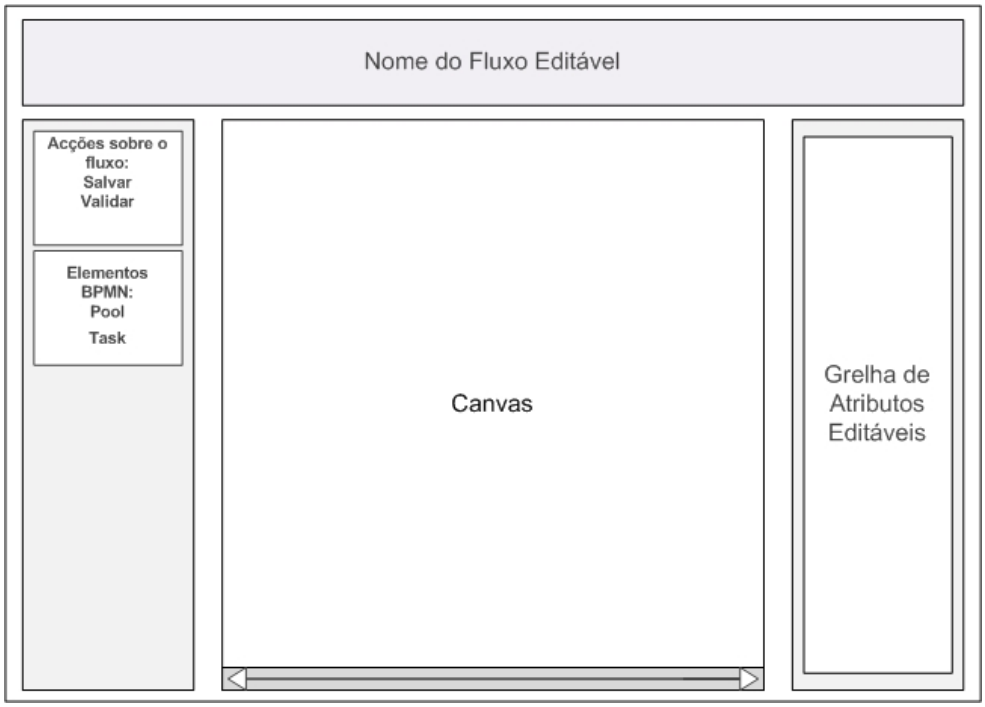

Figura 5.6: Protótipo do Interface do Módulo de Desenho Gráfico

Para o módulo de desenho gráfico temos como a base a área de desenho, onde se poderá adicionar os elementos BPMN para a definição do fluxo. Um aspecto importante será a presença de uma *scrollbar*, que permitirá ao utilizador navegar pelo fluxo, independentemente das proporções que este atinja.

Do lado esquerdo teremos o menu principal de acções, dividido em dois módulos, o primeiro onde poderá ser salvo, validado e carregado o fluxo, o segundo onde pode ser adicionado uma *pool* e uma *task*. A razão pela qual os outros elementos escolhidos para a definição do fluxo não constarem neste menu, prende-se com o facto de se pretender controlar a utilização desses elementos, para que se verifiquem as regras pretendidas. Para que se atinja este feito, foi implementado um menu de contexto similar ao utilizado no *BizAgi*, como é demonstrado na Figura 3.2 e desta maneira controlam-se as acções do utilizador para que os elementos estejam de acordo com as regras BPMN e dCore.

No Topo teremos uma caixa de texto que poderá ser editada, com o nome do fluxo, que quando criado de raiz estará por defeito preenchida com o nome "*untitled* ", já que este campo é obrigatório.

Finalmente do lado direito termos uma grelha com os atributos e respectivos valores, para permitir consultá-los e editá-los a quando da selecção de um elemento, similar ao utilizado no *Oryx Editor*, como demonstrado na Figura 3.3.

5.5.1.2 Diagrama de classes

No módulo de desenho não são necessárias todas as classes representadas na estrutura comum, pois apenas se está a utilizar os elementos de desenho. Na Figura 5.6, é apresentado o diagrama de classes:

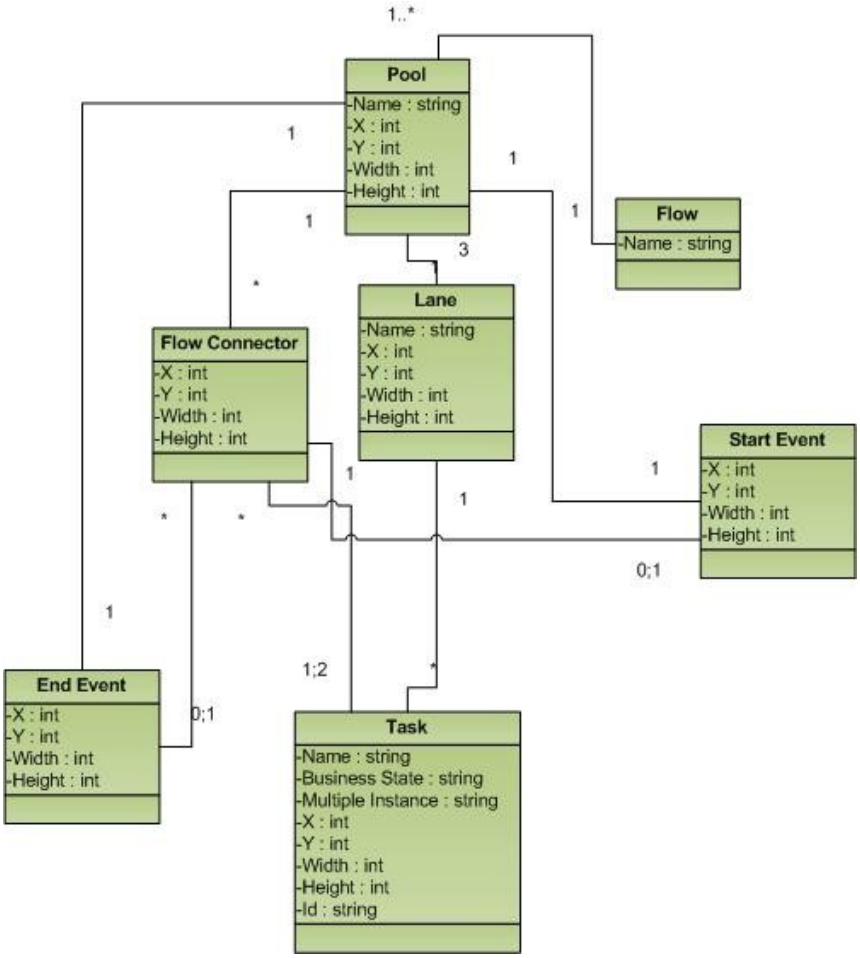

Figura 5.7: Diagrama de classes do módulo desenho gráfico

5.5.2 Módulo Parametrização

5.5.2.1 Protótipo do interface

O protótipo do interface de parametrização foi construindo tendo em conta que é um interface web de submissão de dados clássico. O maior desafio ao construir este protótipo foi mesmo organizar as caixas de submissão dos dados de forma a não ser confuso. Também foi tido em conta a separação lógica entre a parametrização dos object types e a parametrização das tarefas automáticas. Na Figura 5.7, é apresentado o protótipo GUI do modulo visual:

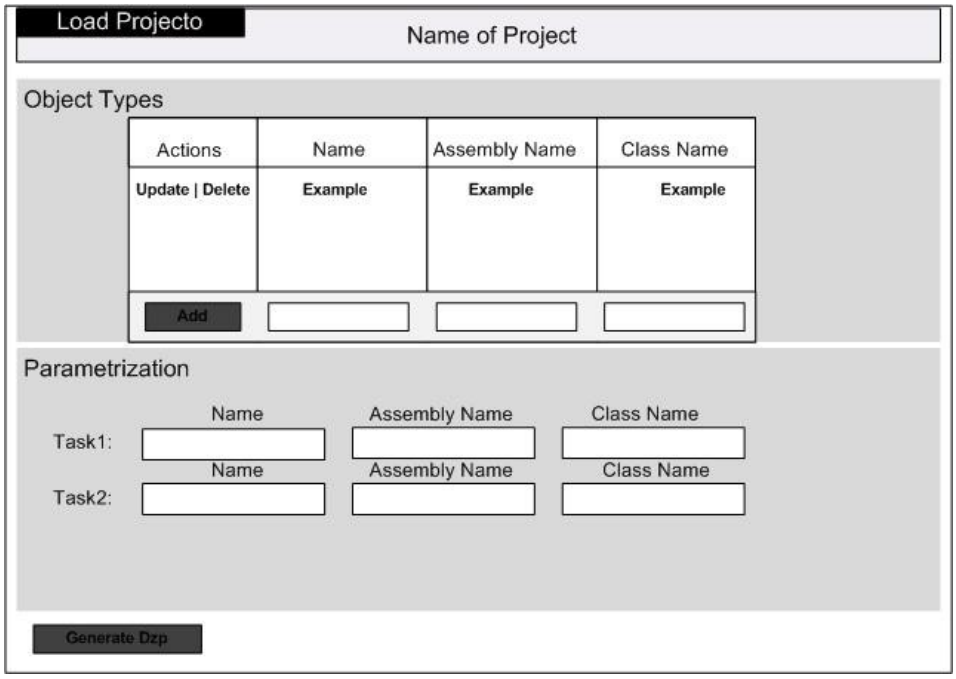

Figura 5.8: Protótipo do Interface do Módulo de Parametrização

5.5.2.2 Diagrama de classes

Neste módulo são utilizadas todas as classes de definição de um fluxo, desta forma foi pensado para este módulo a mesma estrutura que é apresentada na Error: Reference source not found.

5.5.3 Módulo Registo

5.5.3.1 Protótipo do interface

O protótipo de interface deste módulo é o mais simples apenas consiste numa caixa de texto para introduzir o endereço do repositório e um menu para seleccionar o ficheiro dzp. È facilitado também uma caixa de texto para a devolução das mensagens *feedback* do registo.

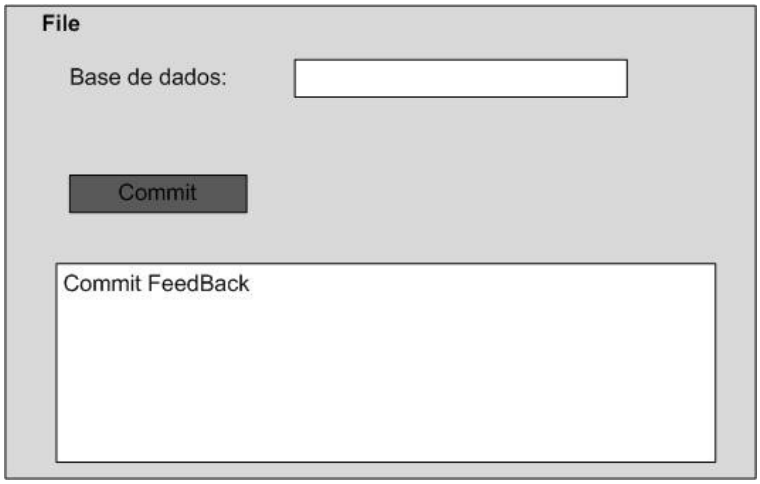

Figura 5.9: Protótipo do Interface do Módulo de Registo de Dados

5.5.3.2 Diagrama de Classes

Neste módulo são utilizadas todas as classes de definição de um fluxo, desta forma foi pensado para este módulo a mesma estrutura que é apresentada na Figura 5.5.

5.6 Implementação

5.6.1 Módulo de Desenho Gráfico

5.6.1.1 Decisões

Após ter sido arquitectado e aquando da sua implementação, existiram algumas decisões que tiveram que ser tomadas.

A primeira decisão a ser tomada, foi a forma como deveria ser feita a conversão da notação BPMN no formato XPDL. Esta poderia ser realizada de duas formas, ou se convertiam todos os dados em XPDL no momento anterior a acção de persistir os dados, ou por outro lado em tempo real todas as acções de adicionar, editar ou remover os elementos eram traduzidas no formato XPDL. Foi escolhida a segunda hipótese, dado que permite realizar backups do fluxo, salvaguardando os casos em que a aplicação possa ter um final inesperado e para o caso de o fluxo ser demasiado extenso, permite dosear a carga de conversão dos elementos.

A segunda decisão relacionou-se com a arquitectura MVVM, dado que não existe um suporte nativo a este padrão de desenvolvimento por parte da tecnologia flash. A

implementação desta arquitectura foi suportada através da instanciação de eventos, que quando disparados podem ser detectados nas camadas adjacentes. Desta forma torna-se eficiente a comunicação entre as camadas, podendo estas operarem sem ter acesso aos métodos das restantes.

5.6.1.2 Implementação baseada em eventos

De forma a existir independência principalmente entre a camada GUI e as restantes, foram criadas duas estruturas diferentes de dados guardados na memória *run-time* deste módulo. Assim os dados relativos aos elementos que são disponibilizados para o utilizador, ficam da inteira responsabilidade do designer, existindo maior flexibilidade na forma como estes poderão ser apresentados.

A camada GUI e a camada de controlo e decisão foram implementadas na tecnologia escolhida para este módulo (*Macromedia Flash + ActionScript*), ao passo que a camada de persistência de dados foi implementada com recurso também aos *webservices*. De seguida é ilustrado na Figura 5.9, o esquema representativo de como funciona a comunicação entre as camadas e na Tabela 5.5, a descrição dos eventos de cada camada.

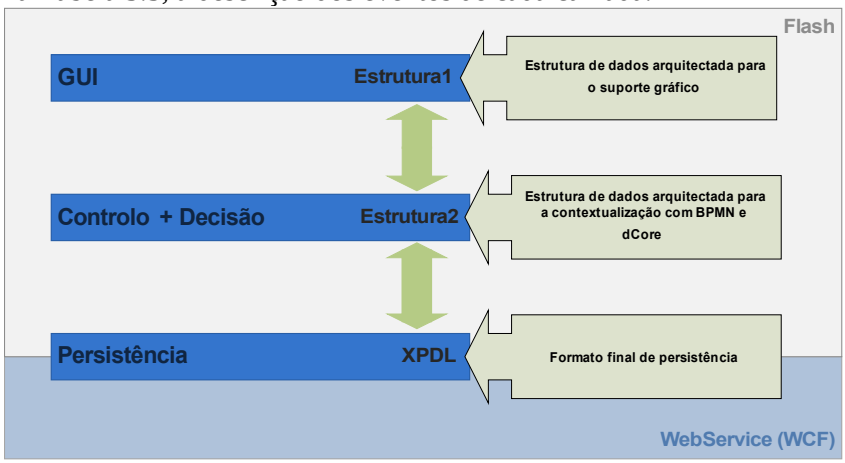

Figura 5.10: Esquema das camadas do módulo gráfico

Tabela 5.5: Descrição dos eventos das camadas do módulo gráfico

Camada:	Eventos:
GUI	Instala: os eventos que correspondem às acções do utilizador. Escuta: os eventos que ao controlo e decisão bem como os de feedback. No processo de carregamento de um fluxo escuta os eventos correspondentes a elementos provenientes da camada de controlo e decisão.
Controlo + Decisão	Instala: os eventos que correspondem às decisões tomadas com o intuito de fazer persistir os dados. No processo de carregamento de um fluxo e

	após escutar os eventos provenientes da camada de persistência de dados instala os eventos para que a camada GUI os possa escutar. Escuta: os eventos relativos às acções do utilizador e feedback da persistência de dados. No processo de carregamento de um fluxo escuta os elementos provenientes da camada de persistência de dados.
Persistência	Instala: os eventos de *feedback*, em relação as acções de persistência de dados, incluindo eventuais erros relacionados com a indisponibilidade da rede. Escuta: os eventos provenientes da camada de controlo e decisão, para que estes em tempo real possam ser traduzidos em XPDL.

5.6.1.3 ScreenShot

Na Figura 5.10 é apresentado o *screenshot* do módulo gráfico, como se pode ver foram utilizadas *lanes* para categorizar as tarefas, foi implementado o menu contextual bem como a grelha para ser promovida usabilidade e controlo à nossa aplicação.

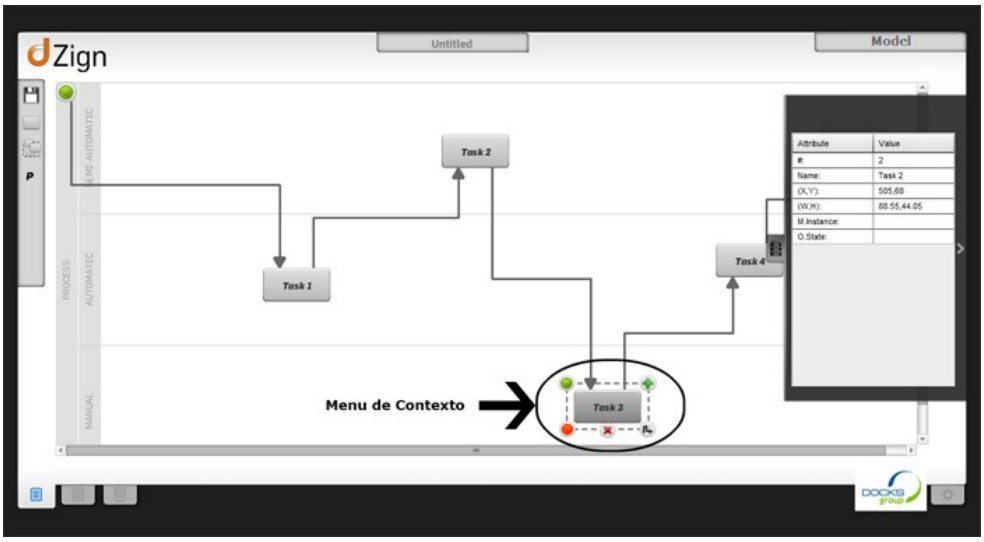

Figura 5.11: Screenshot do Módulo de Desenho gráfico

5.6.2 Módulo de Parametrização

5.6.2.1 Decisões

Para este módulo as únicas decisões a serem tomadas, seriam os componentes a utilizar, para permitir a apresentação,edição e remoção dos conteúdos na memória *run-time* da

aplicação. De seguida são listados os blocos de informação a ser tratada bem como o que se pretende com cada um deles:

- Projectos: Uma lista dos projectos para que o utilizador possa seleccionar qual pretende carregar para este módulo.
- *Object Types*: Uma estrutura que permita ao utilizador adicionar um *object type* à memória *run-time*. Uma lista que apresente os *object types* já introduzidos pelo utilizador, permitindo também a sua edição e remoção da memória *run-time*. Os *object types*, apresentam três campos: Name, Assembly Name e Class Name e ambos apresentam uma restrição quanto aos caracteres possíveis de serem utilizados na sua especificação. É de referir também que o nome de um *object type* não poderá aparecer em duplicado.
- Parametros: Caso um projecto seja carregado, criar uma estrutura que assegure que o utilizador associa a cada tarefa automática um *service*. Os *services*, apresentam três campos: Name, Assembly Name e Class Name e ambos apresentam uma restrição quanto aos caracteres possíveis de serem utilizados na sua especificação. É de referir também que o nome de um *service* não poderá aparecer em duplicado.

Para o primeiro e segundo caso foi escolhido um componente do ASP.NET, *Grid View*, que permite que exista selecção de objectos, útil no primeiro caso. Esta estrutura também prevê a remoção e edição dos componentes, o que se torna útil para o segundo caso. É de referir que para os *object types* também foram facilitadas 3 caixas de texto para que o utilizador possa introduzir os elementos deste tipo que pretende. No último caso para especificar os parâmetros foram escolhidas caixas de texto que são adicionadas dinamicamente após o carregamento de um projecto. Para que seja criado o dzp é necessário que estas caixas se encontrem todas preenchidas.

5.6.2.2 Controlo

Para auxiliar o controlo dos campos e reduzir a sobrecarga no servidor, foi usada a biblioteca de ajax, jQuery[]. Através desta biblioteca, tornou-se fácil facilitar os avisos necessários ao utilizador para que fossem cumpridos os requisitos que os campos a preencher exigiam. De seguida são listados os requisitos e respectivas soluções encontradas com ou sem a utilização do jQuery:

- Caracteres não permitidos nos campos de *object types* e *services*: Sempre que o utilizador insere um caractere não permitido, é visualizada uma caixa informativa a apontar para a caixa de texto onde o utilizador esta com o foco, a informar que tal caractere é inválido.
- Nomes dos *services* ou *object types* repetidos: Sempre que o utilizador introduzir um nome do *object type* já existente é seleccionado na tabela de *object types*, o seu percursor com o mesmo nome. No caso dos *services*, é adicionado uma borda vermelha à caixa de texto que contém o mesmo nome.
- Parametrização não preenchida na totalidade: Caso se verifique o caso de a parametrização das tarefas automáticas não estar preenchida na totalidade, visualizam-se asteriscos vermelho junto das caixas de texto não preenchidas, solicitando então o seu preenchimento por parte do utilizador.

5.6.3 ScreenShot

Na Figura 5.12 é apresentado o *screenshot* do módulo gráfico, onde como se pode ver foram utilizadas *lanes* para categorizar as tarefas, foi implementado o menu contextual bem como a grelha para ser promovida usabilidade e controlo à nossa aplicação.

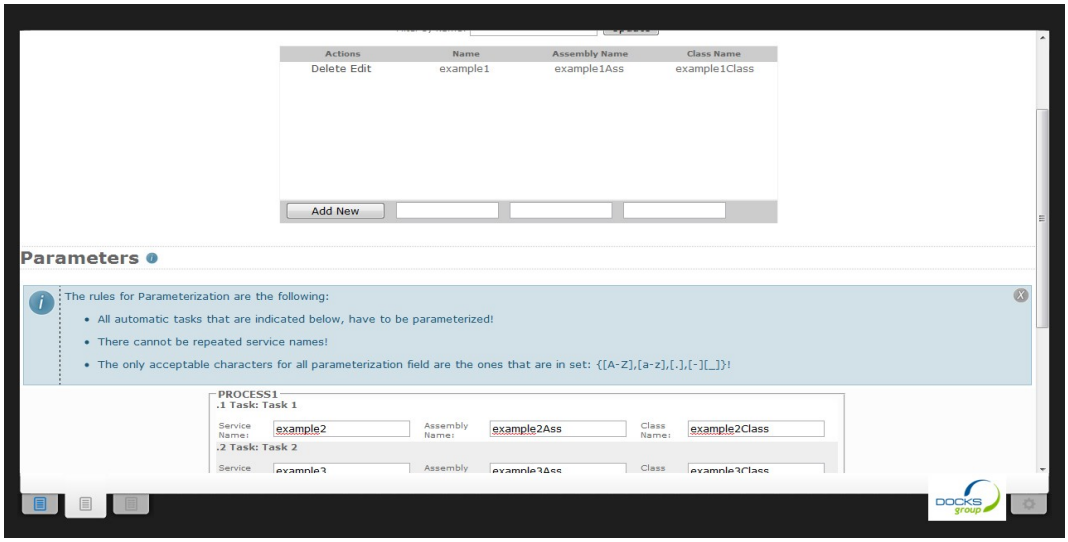

Figura 5.12: Screenshot do Módulo de Parametrização

6 Conclusões e Trabalho Futuro

6.1 Satisfação dos Objectivos

A satisfação dos objectivos foi bem sucedida, já que foi estudado o estado em que se encontrava a submissão de fluxos para a ferramenta dCore e a partir deste ponto traçadas as directivas para se encontrar uma solução mais rica, que aproveitasse as tecnologias existentes no mercado para definir os fluxos. Foi concluído que é possível conciliar a abordagem DOCKS com a abordagem tradicional BPM[1] ao nível da modelação gráfica, o que trará uma maior valia no futuro caso se opte pela implementação da cooperação entre ambos. De seguida são listados os objectivos que foram propostos no inicio deste projecto, a sublinhado e especificado o seu respectivo grau de satisfação:

- Encontrar uma notação gráfica, de preferência standard, para a definição dos fluxos. Foi encontrada a notação BPMN, que sendo uma notação que foi pensada para representar os processos de negócio, ainda assim cobre as exigências dos fluxos dCore e garante um conjunto de elementos que poderão cumprir exigências futuras que advirão da evolução da plataforma dCore.
- Encontrar um mecanismo de persistência de dados, de preferência standard, para que o fluxo possa ser editado e sempre que necessário afectado por diversos módulos/sistemas. No seguimento da escolha da notação BPMN, foi escolhida a notação XPDL, que suporta os elementos desta notação, tendo em atenção as coordenadas gráficas dos elementos. Com esta escolha, foi criada uma base para que no futuro possa existir um aproveitamento desta informação relativa as coordenadas dos elementos por parte da consola.
- Implementar uma arquitectura que permita a execução de cada requisito presente na definição de um fluxo dCore, tendo em conta os diversos perfis necessários (Arquitecto de fluxos/Programador/Cliente). A forma encontrada para que pudessem ser executadas as tarefas por quem realmente teria a competência necessária para as executar, foi dividir o processo de desenho do fluxo, de parametrização e de registo em módulos. Desta forma, primeiro as tarefas são executadas

- Efectuar o registo na plataforma dCore através de uma API disponibilizada para o efeito. O registo do fluxo é efectuado sobre uma API que foi construída para este efeito. Assim existe uma abstracção do repositório dos dados e uma mudança futura deste repositório não afectará o dZign.
- Elaborar um mecanismo de validação, tendo sempre em conta que o fluxo terá que respeitar as regras ditadas pelo dCore. O mecanismo de validação foi elaborado tendo em conta as regras ditadas pelo dCore e a notação gráfica. A validação da notação gráfica BPMN a par da do XPDL é efectuada através de um esquema XSD, especificado pela WMFC o que facilita a actualização das regras por parte do dCore sem afectar as das tecnologias de suporte à modelação gráfica.
- Potenciar a usabilidade e simplicidade. A forma como o dZign foi pensado facilita o cumprimento deste objectivo. Apenas os elementos úteis para representar o fluxo dCore, são disponibilizados, para que desta forma não existam eventuais confusões que poderiam surgir do excesso de funcionalidades.
- Não limitar a manutenção da solução, dado que o dCore poderá estar sujeito a constantes evoluções. A notação gráfica BPMN possui um conjunto de elementos que poderão ser utilizados de futuro tendo em vista eventuais evoluções da plataforma dCore. A arquitectura MVVM foi escolhida para que se possam separar em camadas o interface com o utilizador (Notação Gráfica), regras e persistência de dados(XPDL) e desta forma poderá-se com maior facilidade trocar uma destas camadas sem afectar as restantes.

6.2 Futuro do dZign

6.2.1 Usabilidade e Funcionalidades

Apesar dos objectivos terem sido cumpridos, existe um longo caminho a percorrer dado que o paradigma DOCKS é bastante recente e que o seu amadurecimento irá implicar melhorias no dZign. Sendo que esta ferramenta servirá os consultores numa fase de planeamento dos seus projectos, seria interessante levar em conta as seguintes melhorias:

- Módulo de Desenho Gráfico – *tasks fit to screen*: Proporcionar uma visão privilegiada das *tasks* já adicionadas a um fluxo, através da sua transposição ao tamanho do ecrã. Assim os arquitectos de fluxos poderão ter a qualquer momento noção do trabalho já desenvolvido e do que ainda falta desenvolver.
- Módulo de Desenho Gráfico – comunicação e documentação: Potenciar a comunicação entre os intervenientes neste processo e facilitar a construção de relatórios para efeitos de documentação dos projectos. Uma boa forma de o conseguir é gerar de forma automática relatórios dos fluxos produzidos, tirando proveito do formato aberto de persistência de dados XPDL.
- Estrutura Base – *workspaces* e atribuição de papeis: Gestão de *workspaces* e atribuição de papéis, para que se privilegie confidencialidade e a definição dos intervenientes individuais em todo o processo.
- Módulo de Desenho Gráfico – Previsão de execução de fluxos: Implementar uma funcionalidade de previsão da execução do fluxo, para o arquitecto tenha uma imagem do seu funcionamento.

- Módulo de Desenho Gráfico – Reutilização de *tasks*: Permitir a reutilização de tarefas de fluxos já definidos anteriormente. Desta forma poupa-se trabalho ao arquitecto e ao próprio responsável por implementar os serviços das tarefas automáticas.
- Módulo de Desenho Gráfico – Fluxos Modelo: Conceber um mecanismo para fornecer modelos base, já que muitos fluxos poderão apresentar uma estrutura semelhante

6.2.2 Manutenção

Embora a interacção com os utilizadores seja o mais importante, já que são estes que irão dar utilidade à ferramenta dZign, a manutenção não pode ser descurada, dado que os conceitos evoluem rapidamente e é preciso estabelecer as condições adequadas para encurtar o tempo despendido na actualização desta ferramenta. Assim são sugeridas algumas melhorias que e relacionam directamente com este ponto:

- Serialização dos Elementos: Serializar os elementos na comunicação dos módulos com os *webservices*, controlando assim os formatos de persistência de dados (actualmente dzp e XPDL) inteiramente do lado do servidor.
- Implementar um motor de regras partilhado com o dCore: Um motor de regras partilhado com o dCore, seria um passo importante na manutenção da ferramenta, já que uma alteração nestes requisitos pelo lado do dCore, implicaria uma actualização imediata no dZign.

Referências

[1] A Skeptic's Guide to Computer Models J. D. Sterman, 1991. http://jsterman.scripts.mit.edu/docs/Sterman-1991-ASkepticsGuide.pdf (Mental and Computer Models Chapter),(ultima vez acedido a 1 de Março de 2010)

[2] Ilia Bider, State-Oriented Business Process Modeling: Principles, Theory and Practice, (ultima vez acedido a 1 de Março de 2010)

[3] Ian Sommerville, Software Engineering 8th Edition, Addison-Wesley,2006.

[4] Martins Pedro, Soares Miguel. O Conceito de Fluxo Documental e a Sua Gestão como Meio de Optimização do Negócio, DOCKSGroup , Março 2010.

[5] The Model-View-ViewModel (MVVM) Design Pattern,2009. http://msdn.microsoft.com/en-us/magazine/dd419663.aspx, (ultima vez acedido a 1 de Março de 2010)

[6] Business Process Modeling Notation Specification,2006. http://www.bpmn.org/Documents/BPMN_1-1_Specification.pdf, (ultima vez acedido a 12 de Dezembro de 2009)

[7] Unified Modeling Language. http://www.uml.org/ (ultima vez acedido a 1 de Março de 2010)

[8] XML Process Definition Language Complete Specification,2008. http://www.wfmc.org/index.php?option=com_docman&task=doc_download&Itemid=72&gid=132, (ultima vez acedido a 11 de Novembro de 2009)

[9] Business Process Execution Language Specification,2003. http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel/ws-bpel.pdf , (ultima vez acedido a 1 de Fevereiro)

[10] BizAgi. http://www.bizagi.com/ , (ultima vez acedido a 1 de Março de 2010)

[11] OryxEditor. http://code.google.com/p/oryx-editor/ , (ultima vez acedido a 1 de Março de 2010)

[12] DOCKS Group. http://www.docksgroup.com,(ultima vez acedido a 1 de Março de 2010)

Anexo A: Empresa DOCKS

A DOCKS é uma PME (Pequena ou Média Empresa), cuja missão é: "*Desenhar, implementar e fornecer soluções que melhorem de forma drástica o tratamento de documentos e informação, reduzindo custos operacionais e aumentando a produtividade no local de trabalho, restituindo ao cliente o foco no seu próprio negócio.*"[12]

Olhando para a tecnologia dos sistemas informáticos e da computação, no seu estado actual, como um adereço que permite automatizar um conjunto de tarefas, a DOCKS decidiu apostar no desenvolvimento de software, tendo em vista o auxilio à consultoria na gestão documental. Assim para além dos sistemas informáticos disponíveis no mercado usados até então, existirão vantagens como o suporte aos paradigmas e ideias que vão surgindo internamente e o desenvolvimento de soluções especificas à medida de cada cliente.

Figura 6.1: Logótipo da DOCKS Group

Anexo B: Escopo de ferramentas pretendidas

O dCore é apenas o início da aplicação do paradigma que a DOCKS desenvolveu. De futuro é pretendido desenvolver um conjunto de módulos, que irão possibilitar várias aplicações para esta plataforma. De seguida é apresentada uma figura com a arquitectura prevista para toda esta infra-estrutura:

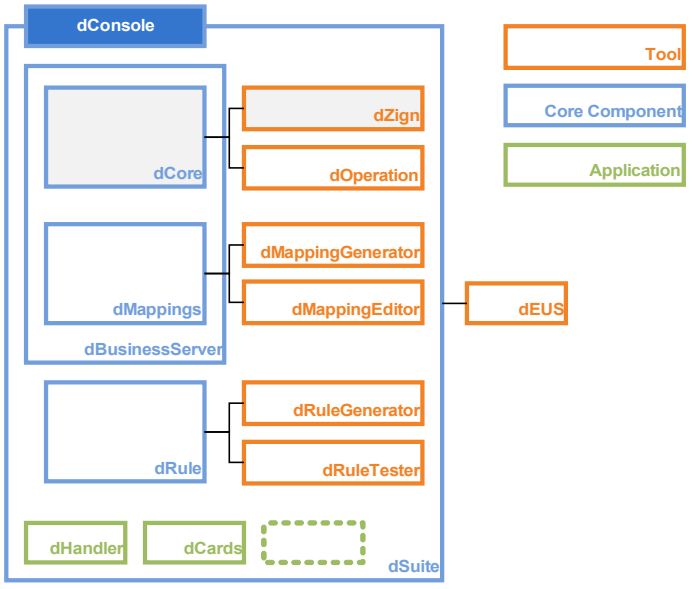

Figura 6.2: Arquitectura da infra-estrutura que se pretende implementar

A.1 dCore como elemento do núcleo

O dCore é uma plataforma que pretende automatizar o processamento dos documentos sobre os fluxos documentais. Esta plataforma apresenta-se como um dos módulos integrantes do núcleo que vai dar vida a uma a infra-estrutura que se pretende implementar. Esta infra-estrutura servirá para dar vida a aplicações cuja a execução é beneficiada pela perspectiva DOCKS. No entanto também é necessário promover a implementação de ferramentas que auxiliem toda a lógica do núcleo que se pretende que sustente esta infra-estrutura, como é o caso do dZign.

A.2 Ferramentas

As ferramentas, representadas a laranja suportarão a definição dos fluxos, *mappings*(documentos e as suas variáveis de estado) e as regras que se pretende aplicar.

A.3 Aplicações

As aplicações, serão então as aplicações que correrão sobre o núcleo desta infra-estrutura, temos como o exemplo o dCards, que irá ser uma aplicação de gestão de cartões empresariais e que será desenvolvida sob os fluxos geridos pelo dCore.