FRANCISCO JOSÉ MONTEIRO DUARTE

Engenharia de Software Orientada aos Processos

Dissertação de Mestrado

Mestrado em Informática

UNIVERSIDADE DO MINHO
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE INFORMÁTICA

Braga – Julho 2002

FRANCISCO JOSÉ MONTEIRO DUARTE

Engenharia de Software Orientada aos Processos

Dissertação de Mestrado

Mestrado em Informática

Dissertação realizada sob a orientação do
Prof. Doutor João Miguel Lobo Fernandes
Professor Auxiliar do Departamento de Informática da
Escola de Engenharia da Universidade do Minho

UNIVERSIDADE DO MINHO
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE INFORMÁTICA

Braga – Julho 2002

Resumo

O processo de desenvolvimento de sistemas informáticos deve ser orientado ao cliente. Sistemas que utilizam tecnologia de ponta, desenvolvidos e concluídos em tempo útil e dentro do orçamento disponível, podem não ter uma utilização plena, ou mesmo não serem usados, se, entre outros, os requisitos do cliente não forem devidamente captados e transformados em aplicações.

Modelos semi-formais e visuais de informação, como os contidos na Unified Modeling Language (UML), que permitem visualizar, especificar, construir, e documentar os componentes de um projecto, e processos que utilizem esses modelos com benefício para o cliente final, como o Rational Unified Process (RUP), conduzem a um maior entendimento entre os engenheiros de software e os clientes.

Os clientes finais para soluções de software podem ser agrupados em duas classes distintas, classificados segundo as expectativas relativas às soluções que adquirem: os utilizadores pessoais, como os que usam sistemas operativos ou outras aplicações previamente desenvolvidas; e as organizações, que, além de conterem utilizadores de software semelhantes aos utilizadores pessoais, são elas próprias a base de execução de processos de negócio muitas das vezes sustentados por soluções de software. É neste último tipo de utilizadores que, nesta dissertação, é centrada a atenção sobre o processo de desenvolvimento de produtos de software.

Actualmente, as organizações tendem a afastar-se de modelos piramidais hierárquicos subdivididos em departamentos, nos quais o trabalho é executado por colaboradores com uma visão parcial da organização, e tendem a organizar-se em equipas multidisciplinares que têm sob a sua responsabilidade processos horizontais que podem atravessar toda a organização. Deste modo, a abordagem no desenvolvimento de sistemas de software deve ser bifocal: o processo de desenvolvimento do software deve ser adequado e controlado; e a plataforma organizacional onde os processos vão ser executados deve ser modelada e considerada, tanto na organização que desenvolve o software como na organização onde o produto vai ser utilizado.

Nesta dissertação é proposto um modelo genérico para organizações orientadas aos processos, que serve de base à modelação de organizações alvo. Com base neste modelo, é também proposto um modelo de processos para organizações que desenvolvem software, a sua respectiva forma de gestão, e a instanciação dos seus processos com os fluxos de trabalho do RUP sempre que disponíveis ou com outro tipo de processos. São também apresentados dois casos de estudo relacionados com o uso de UML e do RUP para desenvolver software para organizações alvo orientadas a processos.

Abstract

The development process for computer systems must be oriented to the client needs. Systems using recent technologies, developed and concluded in time and in budget, could have the risk of not having a full usage or even not being used at all, if, among other points, the client requirements weren't properly understood and transformed in applications.

Visual and semi-formal information models, like the ones kept inside the Unified Modeling Language (UML), which allow to visualise, specify, construct, and document the components of a project, and processes that use this language with added value to the final client, like the Rational Unified Process (RUP), lead to a better understanding between the software engineers and the clients.

The final clients for software applications can be grouped into two classes, distinguished by the expectation with respect to the solution: personal users, like the ones who use operating systems or other applications previously developed; and organisations, which beyond having users similar to personal users, are themselves the base for running business processes often supported by software applications. This dissertation focuses on the development of software products for users of the latter class.

Currently, organisations are moving from a hierarchical pyramidal model sub-divided into departments where the activities are run by collaborators with a limited view of the organisation, to a model where multi-skilled teams run horizontal business processes that can cross all the organisation. Thus, the approach to develop software systems must address two aspects: the software development process must be adequate and controlled; the organisational platform where the processes will run must be modelled and considered, both in the software development organisation and in the target organisation of the product.

This dissertation presents a proposal for a general model for process oriented organisations, which is the basis for modelling target organisations. With this model, it is also proposed a process model for software development organisations, the corresponding way of managing it, and the instantiation of their processes with RUP workflows, whenever they are available, or with other kind of processes. Two case studies, related with the usage of UML and RUP to develop software for a process oriented organisation, are also presented.

Para o João Nuno,

"There ain't no rules around here! We're trying to accomplish something!"

Thomas Alva Edison

"Thought this be madness, yet there is method in it"

William Shakespeare

Agradecimentos

Agradeço ao Professor Doutor João Miguel Fernandes pela total disponibilidade de tempo demonstrada, pelas adequadas sugestões de leitura, pelos comentários e conselhos, bem como pelo extraordinário empenho pessoal evidenciado ao longo de todo o processo de elaboração desta dissertação.

Agradeço à empresa Blaupunkt Auto-Rádio Portugal, Lda., e ao grupo Bosch da qual faz parte, a prontidão e apoio demonstrados que conduzem à valorização dos seus colaboradores, nos quais eu me incluo.

Tabela de Conteúdo

1. Introdução

Neste capítulo são apresentadas, em termos genéricos, as características de um processo adequado para o desenvolvimento de software, bem como o tipo de organizações consideradas como clientes finais de soluções informáticas. Em seguida, são explicadas as razões que levaram à percepção da necessidade da existência de processos controlados de desenvolvimento de software e os objectivos do trabalho. Para terminar, mostra-se a forma como esta dissertação foi estruturada.

1.1 O Processo de Desenvolvimento de Software

Um factor fundamental para a obtenção de um produto de qualidade, no caso presente um produto de software, é a qualidade do seu processo do desenvolvimento. É mais fácil obter um produto com qualidade a partir de um processo de desenvolvimento controlado do que a partir de um processo descontrolado. Em termos formais, a qualidade de um produto [ISO, 2000] é vista como o cumprimento das especificações iniciais para esse mesmo produto e não a noção do senso comum de ser elegante, ou que incorpora a última tecnologia. Assim, o processo de desenvolvimento de um produto de software tem que garantir que o produto cumpre os requisitos levantados junto do cliente final, no início do processo ou durante o seu decurso de uma forma interactiva, ou determinados a partir das características pressupostas de um potencial conjunto de utilizadores finais.

Na captura dos requisitos do sistema torna-se necessária a utilização de uma forma de registo e intercâmbio de informação que, por um lado, permita uma interacção entre o engenheiro de software e o cliente final, de maneira a que este consiga expor àquele de uma forma inequívoca, e não especializada, as características para o produto final, e que, por outro lado, permita ao engenheiro de software validar as suas decisões junto do cliente final, sem existirem ambiguidades.

Além da interactividade, a utilização dum formato de registo de informação deste tipo, como a Unified Modeling Language (UML) [Rumbaugh et al., 1999], permite que ao longo de todo o processo de desenvolvimento de software exista uma relação entre os componentes das várias fases do processo, conseguindo-se assim uma navegação tanto para montante como para jusante. Esta característica permite que o processo não seja estanque nas suas fases. Depois de se ultrapassar uma determinada fase, é sempre possível voltar atrás para fazer alterações e continuar novamente a partir desse ponto.

A informação contida nos requisitos do cliente final propaga-se através das várias fase do processo de desenvolvimento e sofre transformações essencialmente na sua forma e não no conteúdo, o que conduz a um processo que gera um produto de software que está de acordo com as especificações, sendo deste modo um produto com qualidade.

A complexidade dos sistemas pedidos a um engenheiro de software deve-se, não a factores técnicos como as tecnologias usadas na sua implementação, mas principalmente à correcta integração nos processos modelados e desenhados com os processos reais existentes numa organização.

Actualmente, as organizações industriais ou comerciais, nomeadamente as que possuem uma dimensão nacional ou multinacional, são compostas de várias outras suborganizações que estão dedicadas a uma área mais específica das actividades da organização. É comum que cada área de actividade mais restrita se estenda por mais que uma região ou um país, contando com a existência de várias instalações físicas distribuídas geograficamente. Deste modo, por questões de economia e de normalização, os macro processos de negócio, dentro da organização ou suborganização, têm que ser modelados de forma a que abarquem todas as especificidades locais mas mantenham coerência ao nível da organização como um todo.

Toda esta complexidade empresarial traduz-se na prática pela necessidade de reorganizar o trabalho de uma forma dirigida para os processos [Hammer, 1997]. As tradicionais pirâmides hierárquicas apresentam uma estrutura demasiado rígida e pouco adequada para a implementação prática de trabalho em equipas dirigido aos processos. Com uma organização deste tipo, um processo global da organização é fatiado e cada uma das partes é atribuída a um subconjunto da organização responsável por essa área. Deste modo, a noção de processo perde significado e o que se torna importante é cumprir objectivos limitados pelas fronteiras desse subconjunto da organização. Como resultado, pode esperar-se que os subconjuntos da organização percam a noção da importância do processo como um todo e se limitem a cumprir as tarefas, mesmo quando têm um excedente de capacidades financeiras ou humanas numa mesma ocasião em que o subconjunto organizacional imediatamente "ao lado" está com dificuldades em cumprir os seus objectivos. Como consequência imediata, o processo como um todo corre o risco de não ser executado ou mesmo de falhar.

Por contraponto, numa organização orientada aos processos não existe a noção de uma cadeia hierárquica rígida. Os colaboradores são agrupados em pólos de competências. Estas competências podem abranger actividades empresariais tão distintas como a informática, a logística, ou a gestão de processos. Quando se torna necessária a instanciação e execução dum processo, é criada uma equipa composta por elementos dos diversos pólos, e é nomeado um gestor de processo que coordena a equipa. É esta equipa que vai ser responsável pelo processo como um todo, e o sucesso da equipa é o sucesso do processo.

É portanto, neste tipo de ambiente de organizações destinatárias para os produtos desenvolvidos por engenheiros de software que se torna necessário avaliar alternativas para a implementação de processos de desenvolvimento de software, enfatizando os conceitos mais relevantes e extraindo os que não acrescentam valor. Deste modo, o processo de desenvolvimento de software e os produtos que dele resultam terão uma configuração mais adequada a organizações orientadas aos processos.

1.2 Motivação

É impensável actualmente a existência de uma sociedade industrializada sem o recurso a sistemas informáticos. Há um cada vez maior número de utilizadores, e o âmbito destes sistemas é cada vez mais alargado e com tarefas mais complexas. Deste modo, no desenvolvimento de sistemas, nomeadamente de software, tem que se garantir que o produto final tenha qualidade e que o seu processo de desenvolvimento seja controlado.

Torna-se pois necessário que tanto os produtos de software como os processos que lhes deram origem sejam adequados ao tipo de organizações encontradas presentemente. Neste sentido, a adopção de processos estandardizados genéricos e bem documentados, apresenta-se como vantajosa.

1.3 Objectivos do Trabalho

Este dissertação de mestrado tem como propósito genérico avaliar, do ponto de vista do desenvolvimento de software, a diferença que possa existir entre o desenvolvimento de software usando apenas uma perspectiva puramente tecnológica dando relevo às características técnicas do produto ou das ferramentas com que foi desenvolvido, e o desenvolvimento de software com uma perspectiva puramente orientada para os resultados finais, ou seja, se o cliente final estiver satisfeito com o produto, este e o processo que lhe deu origem são adequados.

Como principais objectivos deste trabalho podem ser apontados:

- Aferir as capacidades de modelação e de entendimento que uma linguagem gráfica de modelação proporciona;
- Verificar a possibilidade e ajuizar o valor acrescentado para a organização que desenvolve o software, da adopção de um processo controlado, standard, e adequado de desenvolvimento de software quando o cliente final é uma organização orientada aos processos;
- Avaliar a capacidade de um processo deste tipo tratar com sistemas já em execução, que tenham sido desenvolvidos sem uma perspectiva orientada aos processos, e como fazer a sua adequação;
- Sugerir um modelo genérico para organizações orientadas a processos, que modele organizações alvo, e propor a sua forma de gestão;
- Propor um modelo genérico para organizações que desenvolvem software incorporando os fluxos de trabalho do RUP;
- Aplicar os conceitos em casos de estudo.

1.4 Conteúdo e Organização do Trabalho

Nos capítulos 1 a 5, são expostos os conceitos e áreas de estudo nucleares que formam a base de conhecimento para o presente trabalho.

Assim, no presente capítulo são introduzidos alguns conceitos que são pressupostos de base para o projecto a desenvolver. São também explicados os objectivos do projecto que engloba esta dissertação.

Nos capítulos 2 a 4 são abordadas tecnologias e metodologias na área da Engenharia de Software.

No capítulo 2 é feita uma panorâmica geral dos conceitos actuais em engenharia de software como a gestão dos projectos de software e de fases do processo de desenvolvimento de software, como sejam a análise de requisitos e especificação, o desenho da solução, a validação e verificação, o desenho para sistemas com elevados requisitos de disponibilidade. São também abordados o desenvolvimento por métodos formais de programação, e a manutenção de software. No capítulo 3 são apresentados a Unified Modeling Language e os processos de desenvolvimento Unified Software Development Process [Jacobson et al., 1999] e MIDAS [Fernandes, 2000]. O capítulo 4

aborda as questões relacionadas com a qualidade de um produto de software e da capacidade de uma organização para a geração de produtos de software, aqui através duma perspectiva do Capability Maturity Model [Curtis et al., 1995].

Nos capítulos 5 a 7 são apresentados os contributos e as propostas mais relevantes que esta dissertação de mestrado se propôs atingir.

O capítulo 5 aborda em detalhe as características de organizações orientadas aos processos, sendo propostos modelos genéricos de organizações orientadas a processos modelando organizações alvo, e a sua instanciação num modelo para organizações que desenvolvem software.

No capítulo 6 estão contidos os casos de estudo, sendo estes dedicados à implementação prática dos conceitos expostos nos capítulos anteriores. Este capítulo inclui a descrição da organização alvo, a descrição dos casos de estudo, os artefactos referentes ao fluxo a que foi dado mais relevo no RUP (Modelação de Negócio) de um dos casos de estudo, os aspectos positivos e negativos encontrados, e as lições assimiladas.

A dissertação termina com o capítulo 7, onde são apresentadas as conclusões finais e é apontado o possível trabalho futuro.

2. Engenharia de Software

Este capítulo começa por apresentar modelos e meta-modelos de processos de desenvolvimento de software. Em seguida são apresentados vários fluxos de trabalho com relevo na engenharia de software: Análise de Requisitos e Especificação, Desenho, Gestão de Projectos. São ainda mencionados os métodos formais, pois são considerados pelo autor como uma contribuição positiva para processos de desenvolvimento de software, e a manutenção de software, que é também fundamental para o sucesso de um produto de software e que deve ser planeada aquando do processo de desenvolvimento.

2.1 Introdução

A engenharia de software é uma área de conhecimento composta por teorias, métodos e conjuntos de ferramentas necessários à geração de um produto de software [Sommerville, 1997]. A necessidade da existência de uma área de estudo relacionada com este tema deve-se à exigência trazida por factores económicos, conduzindo a uma optimização de processos e assim à manutenção da competitividade no mercado, e também à crescente complexidade dos sistemas tratados, o que proporciona uma evolução da disciplina, que começou a tomar forma no final dos anos de 1960. Actualmente, os produtos de software estão presentes não só no mais tradicional computador pessoal ou no mais sofisticado computador governamental, mas também em dispositivos nómadas, como telefones celulares, ou em dispositivos responsáveis pelo encaminhamento em redes de computadores.

O desenho de soluções de software, para dar resposta à crescente complexidade e diversidade dos sistemas, bem como às alterações nos requisitos [Armour, 2001] durante o seu processo de desenvolvimento, necessita do auxílio de uma área de conhecimento, a engenharia do software, que melhore as garantias oferecidas quanto à qualidade do produto de software. Não se trata de garantir que o produto irá ser sempre óptimo, mas que normalmente irá ser pelo menos bom, produzido de uma forma controlada, e dentro do orçamento disponível.

Historicamente, a engenharia de software começou por ser utilizada em projectos de desenvolvimento de grandes sistemas de software para organismos estatais ou industriais. Actualmente, a maioria das organizações que desenvolvem software são pequenas companhias [Laitinem et al., 2000] e existem mesmo processos específicos de desenvolvimento de software que têm como ambiente de execução esse tipo de organizações [Nunes et al., 2000].

O tipo de desenvolvimento foi mudando radicalmente ao longo dos anos. Inicialmente, as organizações tinham ferramentas, processos, e componentes próprios e actualmente têm um desenvolvimento baseado em processos geridos e medidos, com recurso a componentes já desenvolvidos por outras organizações, e onde tipicamente se desenvolve apenas 30% de novos componentes em cada projecto [Royce, 2001].

A importância do software é progressivamente mais óbvia. Cada vez mais vidas humanas dependem do bom desempenho do software (e.g. em automóveis, aviões, ou sistemas de medicina), e também, actualmente os produtos de software são a forma de transmissão do bem mais precioso das sociedades: a informação [Pressman, 1997]. A

importância do processo de desenvolvimento é tal que em organizações com projectos de tecnologia de ponta, logo com orçamento elevados (e.g. Agência Espacial Europeia, NASA), existem normas standard (e.g. PSS-05-0 da AEE) para a escolha do modelo de processo de desenvolvimento de software bem como do conjunto de artefactos produzidos [Mazza et al., 1994].

O processo de desenvolvimento de software tem que garantir que as características pretendidas ou pressupostas pelo cliente final do produto de software sejam atendidas e incorporadas no produto de software resultante desse processo. A engenharia de software ajuda a que esse propósito seja atingido.

Sendo a engenharia de software uma disciplina que ainda não atingiu a maturidade de outros ramos da engenharia (e.g. a engenharia civil), é importante a existência de processos que garantam qualidade e controlo a quem desempenha actividades nessa área. Sendo assim, o papel dos recursos humanos intervenientes nesse processo é um factor decisivo para o sucesso dos projectos. Existem características desejáveis na sua gestão [Bott et al., 1995]:

- Planeamento estratégico a longo prazo, assegurando-se assim uma plataforma de orientação para todos os intervenientes;
- Empenho na organização, passando-se do mero cumprimento de regras e prazos para uma atitude de empenho e auto-motivação;
- Auto-gestão pelos intervenientes, cabendo à gestão a delegação de tarefas e a anulação de um controlo sufocante;
- Perspectiva unitária, garantindo-se que os esforços individuais se conjugam para um desempenho em equipa e premiando-se o mérito individual;
- Promoção da existência de intervenientes flexíveis.

A perspectiva económica do processo de desenvolvimento de software é um aspecto que não pode ser negligenciado. Assim, a estimativa do custo de um processo de desenvolvimento de software e os factores que o influenciam são aspectos que devem ser considerados em engenharia de software. O tipo de abordagem para a racionalização dos custos de desenvolvimento deve ser balanceado e abranger os seguintes factores [Royce, 2001]:

- Reduzir o tamanho e a complexidade do que é necessário desenvolver no projecto, através:
  - Gestão do âmbito: pesar entre os custos de desenvolvimento e o valor ganho pela incorporação de cada requisito no produto;
  - Redução da quantidade de código gerado por pessoas: uso componentes;
  - Gestão da complexidade: uso de modelação visual (e.g. UML);
- Melhorar o processo de desenvolvimento, através:
  - Passagem de processos clássicos (e.g. modelo em cascata) para processos iterativos (e.g. processo unificado);
  - Ataque aos riscos mais significativos o mais cedo possível através do uso de processos de desenvolvimento com enfoque na arquitectura;
  - Uso de boas práticas no desenvolvimento de software: desenvolver iterativamente, gerir os requisitos, uso de componentes, modelar visualmente, verificar continuamente a qualidade, e gerir as mudanças;
- Aumentar o desempenho das equipas, através:
  - Utilização de equipas com o menor número possível de pessoas;

    - Melhoria das performances individuais dos elementos das equipas: formação, cobertura e experiência nos métodos, ferramentas, e tecnologias usadas no projecto;
    - Melhoria do trabalho em equipa: definição clara das tarefas atribuídas;
    - Melhoria da capacidade organizacional (melhor aferida pela análise das tendências dos indicadores da performance dos projectos, como sejam custo, qualidade, e cumprimento de prazos, do que por listas de verificação ou auditorias ao processo);
- Usar ferramentas integradas e de maior automação, através de:
    - Aumento da produtividade das pessoas através do uso de ferramentas de teste, de modelação visual, etc. que permitem uma maior automação;
    - Eliminação das fontes de erro pelo uso de ambientes que permitem aumentar a coerência entre os produtos do processo;
    - Melhoria do processo: a existência de indicadores objectivos e das suas tendências são a base para a tomada de decisões que permitirão em caso de desvio que o processo tenha sucesso.

A padronização e o grau de cumprimento dos métodos e regras de um processo de desenvolvimento de software não se devem sobrepor nem ser uma barreira aos resultados obtidos pelo processo.

2.2 Modelos de Processos de Desenvolvimento de Software

O processo de desenvolvimento de software é o conjunto de actividades e resultados associados dos quais resulta um produto de software. Existem quatro actividades fundamentais nos processos de desenvolvimento de software [Sommerville, 1997]:

- A especificação, onde as funcionalidades e restrições devem ser identificadas;
- O desenvolvimento, no fim do qual deve ser produzido um produto que incorpore as características detalhadas na especificação;
- A validação, onde o software é testado para garantir que corresponde ao que o cliente pediu;
- A evolução, para que o software seja adaptável às mudanças de necessidades do cliente.

A descrição das actividades constituintes de um processo tem que obedecer a um balanço entre uma especificação demasiado abstracta, o que não ajudaria tecnicamente no desempenho das actividades, e uma especificação demasiado detalhada, que poderia levar à falta de adaptabilidade a situações pontuais no decurso de um processo [Humphrey, 1990].

Não existem processos de desenvolvimento de software mais ou menos correctos, mas sim processos de desenvolvimento mais ou menos adequados à complexidade do projecto pedido, ao tipo de equipa que o pratica, e ao tipo de utilização pretendido para o produto de software resultante. Quando se selecciona um processo de desenvolvimento inadequado, os resultados obtidos, ou seja, o produto final de software, tenderão a possuir uma qualidade inferior do que se se usasse um modelo de processo adequado.

Para uma selecção adequada do modelo de processo a usar no desenvolvimento de software é necessário conhecer com rigor as características do modelo e a partir daí as diferenças entre os modelos. Uma forma de se conseguir este propósito é através da formalização dos modelos [Heimbigner, 1989] ou de outro ambiente não formal mas que proporcione uma base comum para a descrição dos modelos dos processos [Cheatham, 1990]. Apesar da escolha do uso de um modelo de processo em detrimento de outro ser um factor crítico para o sucesso de um projecto, a sua execução é ainda mais relevante. Neste âmbito, a qualificação e consequente desempenho dos intervenientes no processo é um dos factores mais críticos de sucesso.

Aquando da escolha do modelo de processo a usar num projecto concreto, em paralelo deve ainda levar-se em consideração as duas seguintes abordagens genéricas [Boehm et al., 1990]:

- Desenvolvimento incremental: envolve a organização do desenvolvimento numa série de incrementos de funcionalidade e usado preferencialmente em casos em que se verifique a necessidade de funcionalidades disponíveis em fases iniciais, restrições de pessoas e de orçamento, sistema de alto risco, requisitos mal especificados ou entendidos, aplicações de grande porte;
- Desenvolvimento orientado ao custo ou ao plano de prazos: envolve a criação de prioridades nas capacidades requeridas para o sistema e o desenvolvimento das capacidades por ordem decrescente de importância até se atingir o limite de custo ou de tempo disponível.

2.2.1 O Modelo em Cascata

Este foi um dos primeiros modelos usado para o desenvolvimento de software. É composto por fases estanques (fig. 2.1), em que cada uma delas no seu final é validada, permitindo assim que o processo avance para a próxima fase.

Este modelo de processo também é chamado por vezes de Ciclo de Vida do Software ou Modelo Sequencial Linear [Pressman, 1997]. Este modelo é composto pelas seguintes fases:

- Definição dos requisitos, onde as características, as restrições, e os objectivos para o produto de software são estabelecidos de forma entendível tanto pelos clientes finais como pelo engenheiro, ou equipa de engenheiros, de software;
- Desenho do sistema, na qual são criadas eventuais partições entre hardware e software (ao contrário do que sucede com projectos de co-projecto [Fernandes, 2000]). Nesta fase é estabelecida uma arquitectura global para o sistema e é feito o desenho inicial do produto de software com a divisão por funcionalidades que podem conduzir a uma possível implementação em vários módulos ou programas executáveis;
- Implementação e teste de unidades, onde de acordo com as decisões tomadas na fase anterior, são realizados os programas executáveis ou módulos. Cada um dos módulos é testado para verificação do cumprimento das especificações;
- Integração e teste do sistema, na qual os módulos individuais ou programas são integrados e são testados como um todo, para certificação do cumprimento de todos os requisitos iniciais. Então, o sistema completo é entregue ao cliente final;

- Operação, onde o sistema é instalado e posto em uso. A manutenção envolve a correcção de possíveis erros não detectados em fases anteriores, o melhoramento do sistema pelo aperfeiçoamento dos módulos e programas que o compõe, e a integração de novas funcionalidades entretanto apontadas e requeridas pelo cliente.

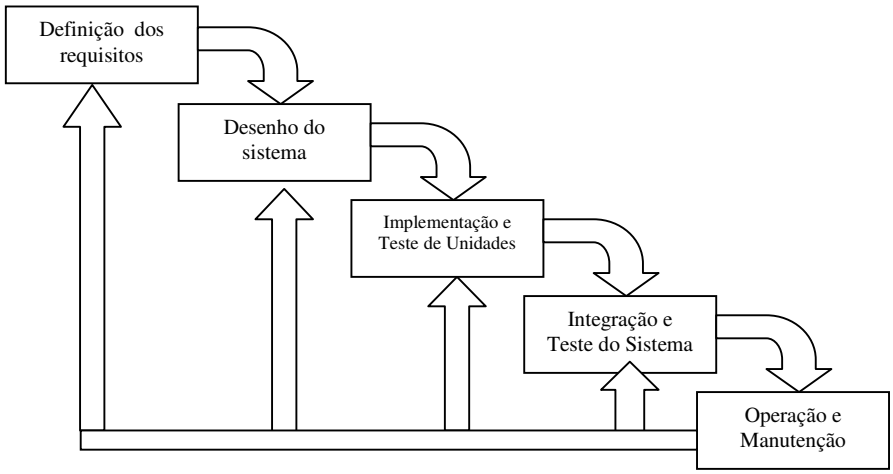

Fig. 2.1 **– O Modelo em Cascata.**

Este modelo de desenvolvimento de software exibe características de compartimentação entre as suas fases, e também a existência de iterações. Estas duas particularidades podem fazer com que os requisitos do cliente não sejam cumpridos [Sommerville, 1997] e que, devido a um excessivo número de iterações, seja difícil conseguir que os gestores do projecto estabeleçam e verifiquem o estado do projecto com pontos de verificação.

Este modelo exige também que o cliente final exprima de uma forma completa os requisitos iniciais, o que é difícil de acontecer, e só após a quarta fase do processo receba a primeira versão do produto [Pressman, 1997]. Pode acontecer também que se as diferentes fases do processo estiverem atribuídas a recursos humanos distintos existam tempos de espera pela conclusão da fase anterior. Normalmente são atribuídos ao modelo em cascata problemas na resolução tardia de riscos, no tratamento das expectativas conflituosas dos interessados, e numa focagem excessiva na documentação e nas reuniões em vez do produto [Royce, 2001].

No entanto, este modelo de processo é ainda muito usado e pode produzir também bons resultados, além de constituir uma abordagem tradicional de engenharia.

2.2.2 O Modelo de Protótipos

É frequente um cliente especificar apenas um subconjunto dos requisitos e objectivos, e não detalhar questões como, por exemplo, a forma da apresentação visual dos sistema. Pode também acontecer que a equipa de desenvolvimento não tenha a certeza absoluta sobre se o algoritmo a usar é o mais adequado ao problema a tratar [Pressman, 1997]. É assim recomendável que o processo de desenvolvimento do software se faça de uma forma evolucionária (fig. 2.2). É gerada uma implementação

inicial, ou protótipo, que não contempla todos os requisitos, e que é exposta ao cliente, sendo depois incorporadas as suas sugestões. Assim, cria-se um esquema de refinamento até se obter a solução final.

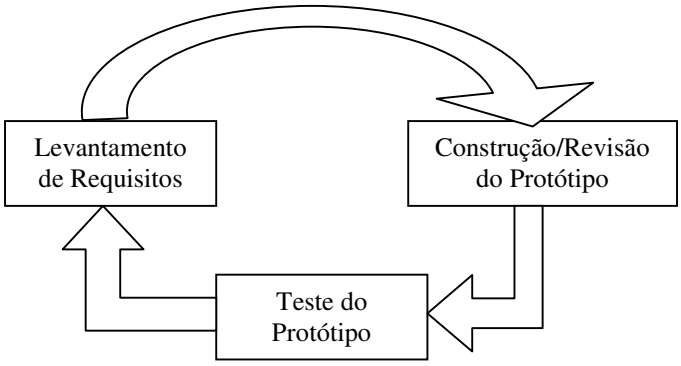

Fig. 2.2 **– O Modelo de Desenvolvimento por Protótipos.**

No início do processo, o cliente especifica os requisitos conhecidos e define os objectivos gerais do produto. É então feito um primeiro desenho dos aspectos visíveis aos utilizadores, nomeadamente as entradas e saídas do sistema [Pressman, 1997]. Este desenho conduz a um protótipo que é depois testado pelo cliente e as suas sugestões e os seus refinamentos aos requisitos são incorporados na geração do próximo protótipo. Este ciclo repete-se até se obter um protótipo final que satisfaça os requisitos do cliente. Posteriormente, é necessário gerar o produto a partir da informação recolhida nos protótipos.

Neste processo é necessário inicialmente informar o cliente que o protótipo final não é a solução final, embora esta possa conter partes do protótipo, para deste modo não se criarem falsas expectativas quanto à conclusão do projecto no cliente aquando da geração do protótipo final.

2.2.3 O Modelo em Espiral

O modelo em espiral de Boehm é um modelo genérico de processo de desenvolvimento de software que incorpora a noção de risco [Boehm, 1988] (fig. 2.3). Este modelo mistura características do modelo de protótipos (como a geração de soluções não finais ao longo de várias iterações) e características do modelo em cascata (como o controlo e a sistematização). Este modelo é um gerador de modelos de processos [Boehm et al., 1990], daí poder ser classificado como meta-modelo.

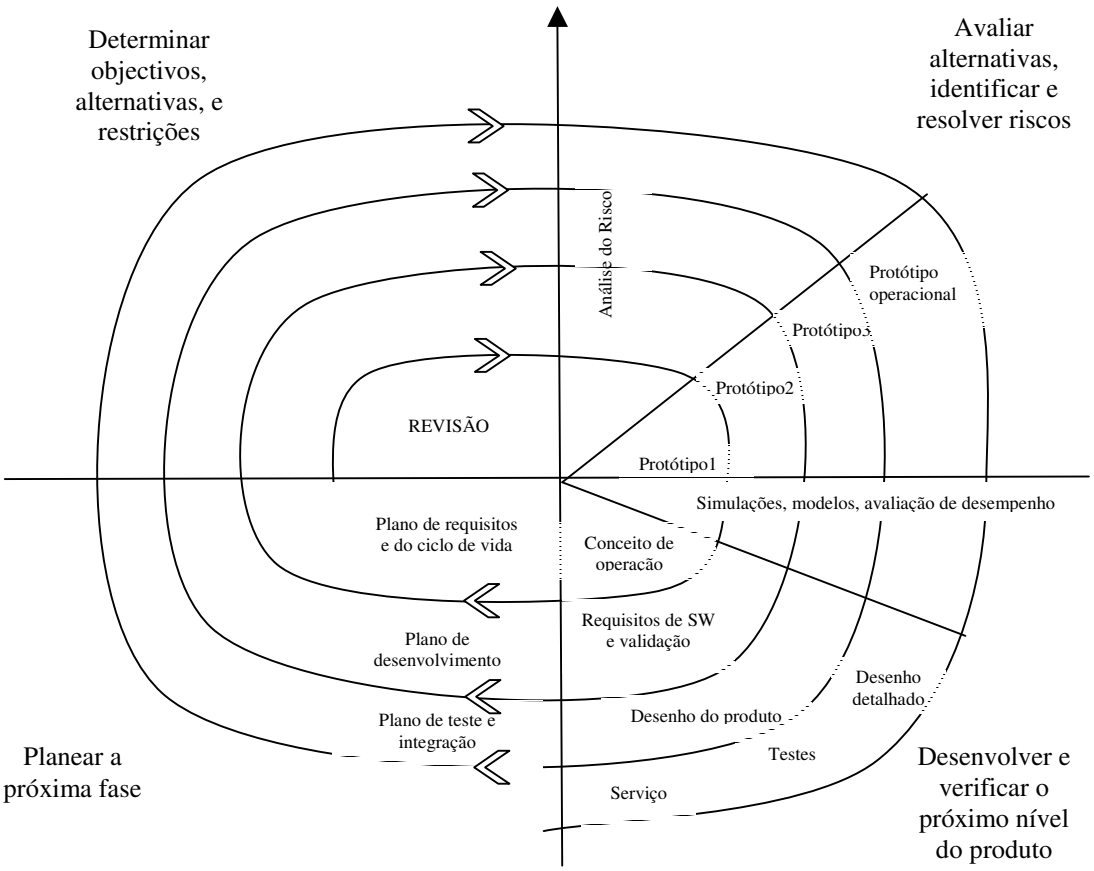

Fig. 2.3 **– O Modelo em Espiral de Boehm [adaptado de Sommerville, 1997].**

Este modelo toma a forma de uma espiral em que cada volta representa uma fase. Neste modelo não existem fases fixas, o que implica que aquelas que são mostradas na fig. 2.3 constituem apenas um exemplo demonstrativo. A gestão do processo é que deve decidir quais as suas fases. O processo começa no interior da espiral e vai caminhando no sentido dos ponteiros do relógio para o exterior. Cada uma das voltas da espiral, ou seja cada uma das fases, está dividida em quatro sectores:

- Estabelecimento de objectivos. Neste sector são definidos os objectivos para cada fase do processo, conjuntamente com as restrições ao produto e ao processo. É feito um plano detalhado de gestão e delineados planos alternativos no caso de se justificarem;
- Identificação e valiação dos riscos e sua redução, onde é feita uma análise detalhada para cada um dos riscos identificados com o propósito de gerar acções que os minimizem;
- Desenvolvimento e validação, é aqui escolhido o tipo de modelo de desenvolvimento de software consoante as análises de risco. Por exemplo, se dominarem os riscos com a interface do utilizador, em princípio, escolhe-se o modelo de protótipos, ao passo que, se dominarem riscos com vidas humanas, a escolha pode recair nos modelos formais;

- Planeamento, onde se fazem as revisões do projecto e se verifica se é necessária um novo ciclo. Se este for o caso, é feito um plano para o próximo ciclo.

Este modelo permite ainda que em cada uma das fases se adopte um modelo distinto de desenvolvimento. Em cada fase (volta da espiral) deve fazer-se uma apreciação dos riscos contendo os seguintes pontos de análise:

- Objectivos: os propósitos da análise;
- Restrições: os factores que limitam as possibilidades;
- Alternativas: os modos alternativos para atingir os objectivos;
- Riscos: as incertezas possíveis para as alternativas identificadas;
- Resolução de riscos: as estratégias usadas para redução dos riscos identificados;
- Resultados: as consequências das estratégias de resolução de riscos;
- Planos: as estratégias relativamente à forma de abordar a próxima fase da análise;
- Compromissos: as decisões da gerência sobre como continuar.

Para o sucesso de processos de desenvolvimento de software que utilizem este modelo, é necessário um conhecimento aprofundado de como realizar as análises de risco e prever-se o tempo para as realizar.

É proposta a seguinte metodologia para determinação do modelo mais apropriado a adoptar num projecto, usando o modelo em espiral como gerador do modelo de processo para um projecto específico [Boehm et al., 1990]:

- Determinar os objectivos e as restrições do processo;
- Identificar as alternativas de modelos de processos que possam ser usados (incluindo modelos que resultem da mistura de características de distintos modelos de processos);
- Avaliar as vantagens e as desvantagens que cada alternativa de modelo de processo fornece de encontro aos objectivos e restrições, e identificar os riscos de cada alternativa;
- Avaliar os riscos;
- Usar a avaliação dos riscos para decidir o modelo de processo a usar.

2.3 Análise de Requisitos e Especificação

No desenvolvimento de um produto de software é essencial uma orientação ao cliente final. Por muito bom que seja o desenho do produto e por muito bem que o produto esteja codificado, se não existir uma boa aproximação ao cliente final, na forma de recolha e análise dos seus requisitos, e posteriormente na sua especificação, o produto final de software será muito provavelmente inadequado.

Um requisito pode ser entendido como uma condição ou funcionalidade à qual o sistema desenvolvido deve corresponder. A análise de requisitos é o processo de descobrir, refinar, modelar, e especificar os propósitos do cliente [Pressman, 1997]. Nesta fase os técnicos operacionais de software trabalham conjuntamente com o cliente final na descoberta, entre outros, de quais os serviços que o produto deve oferecer, qual o seu campo de aplicação, e quais as restrições de hardware existentes. Esta fase é composta por várias subtarefas (fig. 2.4), começando pelo entendimento do domínio

pretendido para o produto e terminando com a validação dos requisitos, e desenvolvendo-se em ciclos que melhoram gradualmente o entendimento do analista acerca dos requisitos do cliente.

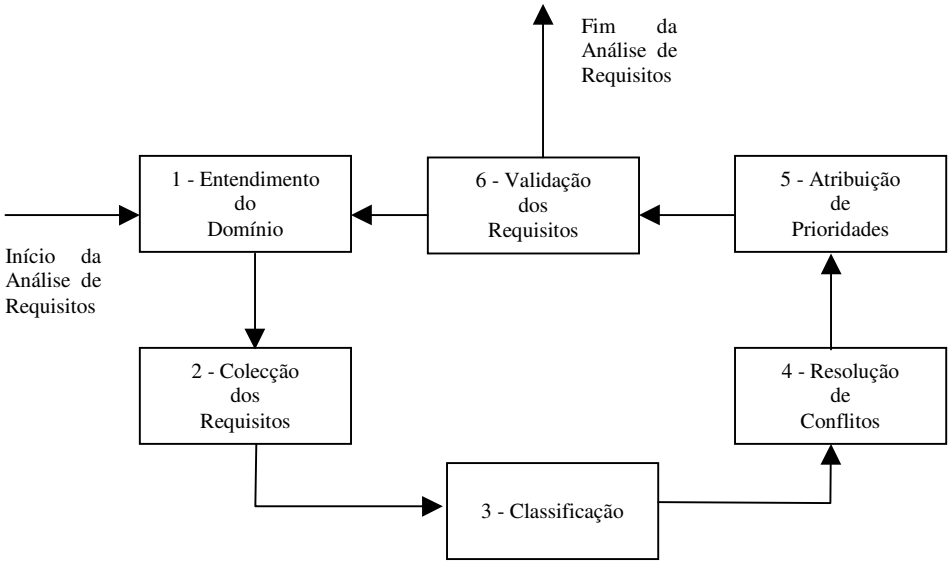

Fig. 2.4 – **O Processo Genérico de Análise de Requisitos.**

O processo de análise de requisitos inicia-se pelo entendimento, por parte do analista, do domínio do produto. Se, por exemplo, o produto de software a desenvolver se destinar à gestão hospitalar, o analista deve aprofundar os seus conhecimentos sobre esse tema. Na fase seguinte, a colecção dos requisitos, são recolhidos junto do cliente final os requisitos do sistema. Posteriormente, na fase de classificação, à colecção desorganizada de requisitos é dada ordem através do agrupamento em blocos relacionados. Torna-se então necessária uma fase, a resolução de conflitos, onde se faça a descoberta e a resolução de conflitos entre os requisitos. Neste estado do processo, onde os requisitos já foram recolhidos, agrupados, e expurgados de conflitos, é necessária uma fase, a atribuição de prioridades, onde se escalonem em conjunto com o cliente final quais os requisitos mais relevantes e prioritários.

Os requisitos do sistema podem ser classificados pela sigla FURPS+[1] [Grady, 1992]:

- Funcionais, por exemplo, a forma de resposta a algumas entradas ou o que o sistema não pode fazer, e que normalmente representam as características principais do produto [Eeles, 2001];
- Não funcionais, podendo neste caso serem requisitos relativos:
    - Utilização: as características do interface com o utilizador;
    - Fiabilidade: disponibilidade, precisão dos cálculos efectuados, recuperação após falha, taxa de falhas aceitável;

[1] FURPS+ significa (F)unctionality, (U)sability, (R)eliability, (P)erformance, (S)upportability, e o + relembra que existem ainda outro tipo de classificação para os requisitos como os de desenho, implementação, interface, ou físicos.

    - Desempenho: tempo de respostas a determinado evento, tempo de recuperação, a velocidade de processamento de um conjunto de dados, tempo para arranque;
    - Suportabilidade: relacionado com características de manutenção, teste, compatibilidade, escalabilidade;
- Outras classes de requisitos:
    - Desenho: por exemplo, um determinado formato para a base de dados;
    - Implementação: como a exigência para o desenvolvimento ser efectuado numa determinada linguagem de programação;
    - Interface: quais os sistemas externos com os quais o sistema deve interagir, qual o formato para essa interacção;
    - Físico: tamanho ou capacidade do hardware usado para suportar o produto de software.

Os requisitos funcionais devem ser completos, no sentido em que todos os serviços pretendidos pelo cliente devem ser definidos, e devem ser consistentes, para não existirem contradições entre os requisitos definidos.

Na especificação dos requisitos deve ser usada uma forma rigorosa de linguagem de modo a reduzir a ambiguidade. Podem ser usadas, entre outras, a linguagem natural estruturada, notações gráficas [Rumbaugh et al., 1999], ou especificações matemáticas como as redes de Petri [Fernandes, 2000].

Normalmente do conjunto de requisitos recolhidos e especificados, a grande maioria (funcionais ou não) é significativa para a arquitectura final do produto. Na sua realização pode assim ser usado um mecanismo de arquitectura, que não é mais do que uma solução para um problema frequentemente encontrado. Estes mecanismos podem ser agrupados em [Eeles, 2001]:

- Mecanismos de análise que representam uma solução independente da implementação (e.g. a persistência ou tipo de comunicação do sistema);
- Mecanismos de desenho que são um refinamento ou resposta aos mecanismos de análise (e.g. usar uma base de dados relacional para a persistência, ou uma fila de mensagens para tratar da comunicação);
- Mecanismos de implementação que são um refinamento dos mecanismos de desenho e que indicam finalmente uma implementação exacta para um determinado requisito (e.g. usar Oracle ® como base de dados relacional, ou usar Microsoft Message Queue ® como fila de mensagens).

De notar que os requisitos FURPS[1] influenciam os mecanismos de análise, que influenciam os mecanismos de desenho, que por sua vez influenciam os mecanismos de implementação. Paralelamente, os requisitos de desenho influenciam os mecanismos de desenho, e os requisitos de implementação influenciam os mecanismos de implementação.

[1] Aqui referem-se os requisitos apenas FURPS e não FURPS+.

2.4 Desenho

O objectivo desta fase é a produção de modelos ou representações do produto e das suas parte que mais tarde serão construídos. Sendo esta uma fase de criação, o seu sucesso depende em grande parte da experiência dos desenhadores obtida com produtos semelhantes. Contudo, podem existir também métodos e critérios para a fase de desenho [Rational, 2000a]. Posteriormente à fase do processo de desenvolvimento em que o desenho tem mais peso, passa-se a fases em que a geração de código e em seguida os testes ao produto são mais preponderantes nos recursos consumidos e nos artefactos produzidos.

O processo de desenho começa com uma representação de alto nível, facilmente relacionável nos requisitos, e à medida que as iterações ocorrem entra-se cada vez mais num nível de minúcia na representação do desenho do produto. No final desta fase vão existir vários desenhos com diferentes vistas sobre o produto, como sejam:

- Desenho de dados: onde são definidas as estruturas de dados e as suas relações, que posteriormente vão ser usadas na implementação;
- Desenho da arquitectura: onde o produto é dividido e as relações entre os seus elementos estruturais são representadas;
- Desenho das interfaces: onde são descritas as comunicações entre os elementos estruturais do produto com outros produtos de software e com os utilizadores;
- Desenho de componentes: onde é feita a distribuição das funcionalidades pelos vários componentes do produto;
- Desenho de algoritmos: onde os algoritmos são desenhados em pormenor e especificados.

Nesta fase são tomadas decisões que podem afectar muito a qualidade final do produto, seja pelo cumprimento dos requisitos ou por questões de manutenção do software. Não só todos os requisitos explicitados pelo cliente, mas também os requisitos implícitos ao produto, devem ser incorporados no desenho.

Até à relativamente pouco tempo a estratégia de desenho de um produto era funcional, com a funcionalidade a ser decomposta pelos vários componentes e a informação de estado mantida numa área central de dados. Actualmente, os produtos e também o seu desenho usam uma estratégia orientada aos objectos. O sistema é visto como uma colecção de objectos e não de funções. Cada objecto contém um conjunto de atributos que definem o seu estado e um conjunto de operações que agem sobre esses seus dados. Assim, o estado do sistema está descentralizado. No entanto, a perspectiva funcional do desenho ou a perspectiva orientada a objectos são complementares dado poderem ser utilizadas em níveis distintos de desenho [Sommerville, 1997].

Genericamente, a qualidade do desenho pode ser vista como a correcção e extensão da implementação da especificação. Existem características que tornam o desenho mais eficiente como a coesão, porque cada componente deve implementar apenas uma entidade ou funcionalidade lógica, ou a sua adaptabilidade, porque pode ser necessário alterar o desenho.

A fase de desenho deve fornecer às equipas que geram código e às que fazem a manutenção de software uma perspectiva completa do software, explicitando os seus aspectos funcionais, comportamentais, e relativos aos dados, mas sempre com uma perspectiva de implementação.

2.5 Manutenção de Software

Tradicionalmente, a área da manutenção de software diz respeito à forma como os engenheiros de software lidam com a implementação de alterações a produtos de software após estes estarem operacionais. No entanto, a abordagem à manutenção de software deve ser revista e o início das actividades de manutenção de software deve ser o próprio início do processo de desenvolvimento de um novo produto [Pigoski, 1997]. Esta abordagem permite estender os interesses existentes aquando do desenvolvimento de um novo produto: os clientes e os responsáveis pelo desenvolvimento querem um produto com qualidade, dentro dos custos e dos prazos; os responsáveis pela manutenção querem um produto ao qual seja fácil dar suporte.

As alterações a produtos de software podem ser de três tipos:

- Melhorias;
- Correcções de erros;
- Adaptações.

Em organizações industriais, a manutenção de software exige dos recursos humanos uma ocupação muito maior do que a produção de novos sistemas (podendo chegar a 50%) sendo que o seu custo pode atingir valores de 70% do custo total do projecto.

A manutenção de software pode ser efectuada através de modelos distintos [Takang et al., 1996]:

- O modelo quick-fix é uma abordagem ad hoc utilizada em situações onde o tempo é o factor mais relevante. Não é feita uma análise detalhada dos efeitos a longo prazo das alterações introduzidas. Este modelo não é desejável dado que funciona num ciclo composto pela correcção de um problema e eventualmente pela criação de um outro, sendo depois necessário repetir de novo o ciclo. No entanto, em casos em que a manutenção é feita por uma equipa muito restrita de pessoas que conhecem muito bem os sistemas e funcionem bem sem o recurso a documentação, esta abordagem pode conduzir a resultados rápidos e de baixo custo;
- O modelo de Boehm representa a manutenção como um ciclo fechado de actividades onde as decisões são baseadas em critérios económicos. A gestão toma decisões sobre a manutenção de software baseada na análise de custos para as soluções alternativas. Em seguida, as alterações são implementadas, o software é posto em uso, e depois avaliado dando origem a propostas de alterações que, fechando o ciclo, vão ser entradas para a tomada de decisões para a gestão;
- O modelo de Osborne faz uma abordagem mais realista das situações, onde é feita a manutenção de software dado não existir sempre documentação disponível acerca de cada produto de software. Neste modelo são recomendadas acções preventivas como a inclusão de requisitos de manutenção na especificação das mudanças, ou a existência de programas de garantia da qualidade do software produzido;
- O modelo iterativo usa três fases: a análise do sistema existente; a caracterização das propostas de alteração; o redesenho da versão actual e a sua implementação. Este modelo suporta a reutilização de código e pode acomodar outros modelos como o quick-fix. Como neste modelo a documentação é revista sempre que existam alterações e nem sempre nas situações práticas esta existe, podem surgir problemas tanto com a

documentação como com a capacidade das pessoas analisarem o sistema em toda a sua extensão;

- O modelo orientado à reutilização é baseado no princípio que a manutenção pode ser vista como uma actividade que envolve a reutilização de componentes já existentes de programas. Neste modelo existem quatro passos principais: a identificação das partes do software que são candidatas para a reutilização; análise e compreensão dessas partes; modificação para acomodação dos novos requisitos; integração das partes modificadas no sistema.

A migração de sistemas com manutenção problemática pode ser feita de um modo integrado entre três tecnologias [Takang et al. 1996], perfazendo assim um processo de reengenharia: a engenharia reversa para análise e compreensão do sistema implementado, seguida dum processo de geração de software adequado à obtenção de um produto com qualidade, recorrendo a um paradigma orientado a objectos, permitindo assim a reutilização de software.

A organização e o tamanho das equipas são factores determinantes para a eficiência da manutenção de software. Se ocorrem muito pedidos de melhoria, é importante que se atribua uma equipa de manutenção ao seu tratamento, e outra equipa separada, funcionando como um serviço de atendimento, para a resolução dos problemas da correcção de erros e da adaptação. O dimensionamento das equipas de manutenção de software pode ser calculado através duma função da taxa de chegada dos pedidos e do tempo para servir esses pedidos, acordado com o cliente. O seguinte método em cinco passos, para estimar o tamanho adequado a uma equipa de manutenção de software pode ser usado [Ramaswamy, 2000]:

- Passo 1: Estimar o volume de pedidos (VP) de chegada. O VP é um produto da média diária estimada do número de pedidos pelo esforço médio estimado para servir os pedidos, em pessoas por hora ou por dia. Esta estimativa pode recorrer a dados de manutenção disponíveis para aplicações semelhantes, ou recorrer ao benchmarking;
- Passo 2: Escolher um nível de preparação (NP) adequado. O nível de preparação é a probabilidade de um pedido de manutenção ser atendido prontamente sem ter que ir previamente para uma fila de espera. Se o cliente desejar um nível de preparação muito alto torna-se necessário consciencializá-lo que isso exige uma quantidade de recursos também muito elevada (ver tabela 2.1);
- Passo 3: Determinar a quantidade base de pessoas pertencentes à equipa usando a tabela 2.1 resultante da aplicação do modelo de serviço a uma fila de espera (neste caso de pedidos de manutenção) por servidores (neste caso pessoas da equipa de manutenção);

Tabela 2.1 – **Tabela para determinação da quantidade base de pessoas (adaptado de [Ramaswamy, 2000]).**

Volume pedidos (VP)	Nível de preparação (NP) desejado					
	80%	**85%**	**90%**	**95%**	**99%**	**99,9%**
1	2	3	3	3	4	5
1,5	3	3	4	4	5	7
2	4	4	4	5	6	8
2,5	4	5	5	6	7	9
3	5	5	6	6	8	10
3,5	6	6	7	7	9	11
4	6	7	7	8	10	12
5	8	8	9	9	11	13
6	9	9	10	11	13	15
7	10	11	11	12	14	17
8	11	12	12	13	16	18
9	12	13	14	15	17	20
10	14	14	15	16	18	21
12	16	17	17	19	21	24
14	18	19	20	21	24	27
16	21	21	22	23	26	30
18	23	23	24	26	29	33
20	25	26	27	28	31	35

- Passo 4: Adicionar capacidade extra. Para projectos pequenos (inferiores a 12 pessoas) deve-se adicionar uma capacidade adicional que pode oscilar de 10% até aos 30%. Deve-se levar em conta a capacidade da equipa tratar com novas tecnologias que seja obrigada a usar. Nos casos em que o nível de preparação é muito elevado, e como os pedidos de manutenção não chegam a uma taxa regular, é essencial manter ocupada a equipa de manutenção mesmo sem ter pedidos de manutenção. Esta ocupação pode ser conseguida pelo estudo da documentação relativa ao sistema, pela aquisição de conhecimentos na área (e.g. logística) em que a aplicação se insere, ou pelo estudo de novas tecnologias;
- Passo 5: Estabelecer uma estratégia de diminuição dos pedidos em fila. Podem existir várias filas de espera para cada tipo de pedidos, dependendo da sua importância, e mesmo dentro de cada fila de espera deve-se verificar a sua prioridade relativa (não apenas servi-los pela ordem de chegada). Nos casos em que pedidos importantes estejam em fila de espera há mais tempo que o aceitável devem-se atribuir as pessoas mais capazes para esse pedido e atribuir outras para sua substituição (pertencendo ou não estas pessoas à equipa de manutenção de software).

Devido aos custos, tanto da ocupação dos recursos humanos como os económicos, os requisitos de manutenção de software podem ser incorporados nos requisitos iniciais, mesmo sem a iniciativa do cliente final.

2.6 Gestão de Projectos de Software

A engenharia de software profissional está sujeita a restrições orçamentais e de tempo como qualquer outro tipo de processo de desenvolvimento. Os gestores de

software devem planear as datas e verificar o estado do projecto de maneira a que os resultados tanto a nível do produto como orçamentais sejam os esperados. A boa gestão do processo de desenvolvimento de software não garante que o produto seja bom, mas uma má gestão tem muito mais possibilidades de conduzir a um produto que seja entregue fora das datas planeadas, que lhe faltem características especificadas nos requisitos, ou que contenha erros, conduzindo assim a elevadas cargas de manutenção.

A gestão de projectos de desenvolvimento de software exibe algumas características que a tornam distinta das gestão de outros projectos de engenharia:

- O produto é intangível, logo para os gestores é difícil aperceberem-se do estado real do projecto. É necessária informação produzida por outros para relatar o estado;
- Não existe um processo standard, porque normalmente não é possível escolher com certeza absoluta qual o processo mais adequado ao produto que se quer desenvolver;
- Devido à constante evolução do software e do hardware disponíveis para a construção de um novo produto, normalmente os processos de desenvolvimento não são reutilizáveis. É necessário adaptar um modelo de processo para cada caso específico.

A função de um gestor de software envolve actividades relacionadas com os seguintes aspectos [Sommerville, 1997]:

- A escrita duma proposta, envolvendo uma descrição inicial dos objectivos e de como atingi-los. Pode incluir uma estimativa de custos e prazos;
- A orçamentação, onde se indicam o orçamento disponível e os custos estimados para as actividades do projecto;
- O planeamento e a calendarização do projecto, onde se especificam as actividades, pontos de controlo, e resultados conjuntamente com a respectiva calendarização;
- A monitorização, que é uma actividade contínua ao longo de todo o processo e que permite ao gestor a comparação entre o que está feito na prática e o que estava planeado que estivesse disponível nessa mesma data;
- A selecção de recursos humanos, em que o gestor escolhe (normalmente dentro de algumas limitações) quais vão ser as pessoas atribuídas ao projecto;
- A produção de relatórios e apresentações aos clientes.

A gestão de projectos de software deve levar em conta três eixos fundamentais que limitam as decisões tomadas pelo gestor do projecto [Pressman, 1997]. O primeiro deles, as pessoas, que são a base de qualquer organização, obriga que o gestor pratique acções e que seja possuidor de uma forma de liderança que motive e estimule o grupo de operacionais a desempenharem o seu papel de uma forma tecnicamente correcta, mas também com interacções com os outros elementos da equipa. Devido à importância deste factor, existe mesmo uma recomendação do SEI, PM-CMM [Curtis et al., 1995], acerca de como cativar, melhorar, e manter os talentos humanos necessários para o sucesso de projectos de software. O segundo eixo, o problema, conduz à existência de um entendimento prévio entre o cliente e o engenheiro de software acerca dos objectivos do projecto e do seu âmbito, onde os dados, funcionalidades, e comportamentos são discriminados preferencialmente de uma forma quantitativa. O terceiro eixo é o processo de desenvolvimento de software. Além da escolha do modelo

de processo, o que por si só conduz a um conjunto característico de actividades, existem outras actividades de mais alto nível que devem ser atendidas, como são a garantia da qualidade do software, ou as métricas exigidas para o desempenho do produto.

Tradicionalmente, quando se usam processos de desenvolvimento de software baseados no modelo em cascata, a gestão do projecto baseia-se nos seguintes princípios [Royce, 2000]:

- Congelar os requisitos antes da fase de desenho;
- Proibir a geração de código antes de existir um desenho detalhado do sistema;
- Usar uma linguagem de programação de alto nível;
- Completar os testes das unidades constituintes do sistema antes de se iniciar a sua integração;
- Manter uma rastreabilidade detalhada entre todos os artefactos do processo;
- Documentar extensivamente cada fase do desenho;
- Verificar a qualidade através de uma equipa independente;
- Inspeccionar tudo;
- Planear tudo previamente e com grande fidelidade;
- Controlar o código fonte.

Estes princípios não se aplicam às necessidades actuais e futuras do mercado de desenvolvimento de software. Com esta abordagem, entre outras limitações, o redesenho está posto de parte pois implicaria grandes atrasos, os problemas entre componentes do sistema só são detectados demasiado tarde no projecto. O mercado de software exige que o processo esteja preparado para incorporar a mudança nos requisitos do cliente sem que isso represente graves atrasos, e principalmente exige que o tempo necessário para desenvolver um produto seja curto devido à concorrência existente, o que normalmente implica o recurso à reutilização de componentes já desenvolvidos e testados de software. De notar que para uma efectiva utilização de componentes reutilizáveis é necessária a existência de componentes já desenvolvidos para um domínio específico de área que o produto de software vai tratar [Arango, 1989]. Assim, para o desenvolvimento de produtos sugerem-se os seguintes princípios [Royce, 2000]:

- Basear o processo numa abordagem que dê relevo à arquitectura;
- Estabelecer um ciclo de vida iterativo e que exponha os riscos o mais cedo possível;
- Usar métodos de desenho que dêem relevo à utilização de componentes;
- Estabelecer um ambiente para a gestão da mudança;
- Utilizar ferramentas que suportem o maior número de fases do processo;
- Capturar os requisitos de desenho de uma forma rigorosa;
- Orientar o processo para um controlo de qualidade objectivo e para a verificação do progresso;
- Usar demonstrações para verificação de artefactos intermédios;
- Planear protótipos intermédios com aumento do nível de detalhes;
- Estabelecer um processo configurável e económico.

No entanto, a transição de um processo baseado no modelo em cascata para um processo iterativo de desenvolvimento, como por exemplo, o RUP é difícil. O uso de processos iterativos não garante necessariamente menos trabalho ou um tempo de desenvolvimento menor [Krutchen, 2000], mas sim garante maior previsibilidade nos

resultados e no plano de prazos, assim como produtos com maior qualidade satisfazendo os requisitos do cliente. A gestão de processos baseados no modelo em cascata é mais fácil mas o trabalho da equipa de desenvolvimento é maior, ao contrário do desenvolvimento iterativo que facilita o trabalho da equipa de desenvolvimento mas acrescenta complexidade à gestão do projecto.

A consolidação de boas práticas na gestão de projectos de desenvolvimento de software convida a que se usem métricas para avaliar quer o estado do processo de desenvolvimento de software que está a ser gerido (e.g. dias de atraso ou avanço nas tarefas, controlo orçamental), quer o produto em si (e.g. o número de erros, o desempenho em tempo real). No entanto, é conveniente ter sempre presente que a existência de demasiadas métricas é caro, visto não ser uma actividade que directamente acrescente valor ao produto final de software, e pode causar perturbações no seu processo de desenvolvimento [Melton, 1995]. Assim, o gestor do projecto tem que decidir:

- O que medir;
- Qual a métrica usar;
- Porquê medir esse parâmetro.

Compõe-se deste modo uma plataforma para comunicação aos intervenientes responsáveis pelo desenvolvimento ou pela manutenção que dá resposta às seguintes questões:

- Quem é o responsável pela colecta de dados?
- Quando vão ver recolhidos e com que frequência?
- Em que parte do processo?
- Como se recolhem esses dados?

Existem conjuntos de boas práticas para a gestão de projectos de software que ao serem seguidas garantem que o gestor do projecto não cai em erros comuns [ICE, 1999], dos quais se destacam:

- Adoptar um programa contínuo para a gestão dos riscos ao longo de todo o ciclo de vida do projecto;
- Estimar o custo e o plano de prazos empiricamente e posteriormente refinar assim que se obtenha informação mais detalhada;
- Usar indicadores quantificáveis para assim permitir uma gestão eficaz;
- Contabilizar o valor ganho. O valor ganho é binário: 100% quando uma tarefa está completa, 0% em todos os outros casos;
- Verificar os defeitos contra os objectivos pré-negociados para a qualidade do produto;
- Tratar as pessoas como o recurso mais importante;
- Verificar os custos e os riscos da reutilização;
- Gerir os testes como uma actividade contínua;
- Compilar e fazer testes de carga frequentemente.

Mesmo com as melhores práticas e os melhores processos de desenvolvimento de software é sempre necessário em todos os projectos ter um plano de prazos realista, ter um orçamento adequado, e ter uma equipa capaz e em número suficiente. A falta de um destes factores base pode fazer com que o resultado do processo de desenvolvimento não tenha a qualidade requerida pelo cliente.

2.7 Métodos Formais

Os métodos formais de programação são um meio de garantir que os produtos de software têm qualidade, ou seja que cumprem os requisitos especificados. Para obter este resultado os métodos formais são suportados por linguagens para especificação formal baseadas em áreas da matemática (por exemplo na lógica e na teoria de conjuntos) como VDM-SL [Fitzgerald et al., 1998], Camila [Almeida et al., 1997], ou VDM++ [Larsen, 1999].

O desenvolvimento de um produto de software pode ser visto como uma sequência de tarefas de modelação. Os modelos, como abstracções da realidade, são criados para tarefas de análise de requisitos, desenho da arquitectura, ou especificação do domínio de aplicação do produto de software. Actualmente estes modelos são construídos com recurso a descrições textuais e a modelos descritivos gráficos. Para se obter uma qualidade constante nos processos de desenvolvimento é necessário o uso de métodos formais garantindo-se uma base teórica fiável para a modelação e descrição de métodos, notações, e conceitos [Broy, 2001]. Apesar da modelação com recurso a descrições textuais e da modelação gráfica permitirem a construção de modelos abstractos, a estes não lhes é possível atribuir uma semântica formal, sendo assim impossíveis tarefas com a análise automática rigorosa através da verificação do modelo ou a execução de modelos através da simulação [McUmber et al., 2001].

Existem abordagens genéricas para se conseguir a formalização de linguagens semi-formais (como UML) em linguagens formais (e.g. Z):

- O mapeamento directo duma linguagem na outra;
- O mapeamento de UML num formato intermédio e posteriormente na linguagem formal [McUmber et al., 2001];
- O recurso à anotação dos diagramas UML com OCL (Object Constraint Language) [Rumbaugh et al., 1999] apesar desta última não proporcionar uma formalização completa.

Um modelo expresso na linguagem formal VDM-SL é composto por dados descritos por tipos, constantes, e invariantes, e pela funcionalidade expressa através da definição de funções e das operações que podem ser utilizadas no domínio dos dados considerado e de pré-condições para limitar os valores das entradas das funções. Este modelo vai posteriormente ser validado acerca da sua consistência, testando-se se a sintaxe da especificação (descrita na linguagem formal) está correcta e se os resultados das funções podem ser calculados com os tipos de dados de entrada descritos, e da sua precisão na descrição dos requisitos do cliente. Assim, quando é necessária uma alteração ao modelo, ela é feita nestas fases iniciais antes de entrar na fase de desenho do sistema final. Neste tipo de métodos é mais fácil existirem ferramentas para a geração automática de código dado que a especificação não é ambígua. Apesar das fases de análise e especificação serem mais longas que num método tradicional, o tempo total do projecto tende a ser menor, porque a propagação de erros é diminuída no atravessamento das fases do processo e também porque o código pode ser gerado, total ou parcialmente, de forma automática.

Existem ainda algumas questões a ponderar aquando da decisão de um engenheiro de software em usar métodos formais para o desenvolvimento de um novo produto: não são de uso comum e ainda são desconhecidos dos gestores; existem ainda

poucas ferramentas para os suportar (ver, no entanto, IFAD VDMTools em [Larsen, 1999]); e o uso de ramos da matemática para modelar e especificar sistemas pode provocar erroneamente, dado que em alguns casos são necessários apenas conhecimentos médios de matemática, a não adopção de métodos formais.

No entanto, quando se usam métodos formais obtêm-se benefícios, como a prova da correcção do programa, a redução do tempo necessário para chegar à solução final, a redução substancial dos erros na solução final, e a capacidade de adequação à utilização na especificação, análise, e verificação de sistemas críticos. Este tipo de métodos, porque se baseiam na matemática que é uma forma sucinta e exacta de representação de informação, reduz os problemas de contradições, ambiguidades, ou representações incompletas que podem existir em métodos menos formais, além de permitirem um suporte muito mais forte para a automatização de passos do processo como a geração automática de código, a simulação, e a validação da consistência dos requisitos implementados no sistema.

O uso de métodos formais pode ainda ser estendido para sistemas de especificação reversa [Neves et al., 1999] permitindo que sistemas legados (normalmente mal documentados) possam ser re-especificados e assim se obter um suporte formal.

3. Modelação Visual e Processos Orientados ao Objecto

Neste capítulo são descritos brevemente uma linguagem de modelação gráfica (UML) e processos de desenvolvimento de software orientados a objectos (RUP e MIDAS).

3.1 Introdução

Os processos de engenharia de software têm como objectivo a geração de produtos de software com qualidade, que vão de encontro às necessidades dos clientes finais, dentro de prazos e com orçamentos estabelecidos e controlados. A abordagem das metodologias orientadas ao objecto melhora, ao longo de todo o processo, a capacidade de introdução de alterações, e dota-o de capacidades como a abstracção de dados e o encapsulamento, facilita a reutilização de componentes, e mostra mais adequação para lidar com a concorrência e com o dinamismo do sistema do que as metodologias funcionais.

A natureza intangível de um produto de software conduz a que as pessoas responsáveis, segundo uma perspectiva meramente técnica, pelo seu processo de desenvolvimento, como os clientes finais ou os gestores do projecto, tenham uma maior dificuldade na percepção do estado de desenvolvimento do que no caso de produtos mais palpáveis. Assim, as abordagens visuais para a representação da informação, além de melhorarem a visibilidade do processo para a gestão, aumentam o entendimento entre o engenheiro de software e os clientes e são um suporte adequado que reduz as ambiguidades ao longo do processo de desenvolvimento de software.

3.2 Unified Modeling Language

A Unified Modeling Language (UML) é uma linguagem visual genérica de modelação que pode ser usada para especificar, visualizar, construir, e documentar os componentes de um sistema de software [Rumbaugh et al., 1999]. Pode ser usada com qualquer dos modelos de processo da engenharia de software, e pode cobrir todo o ciclo de desenvolvimento. As vistas sobre o sistema podem ser agrupadas em três grandes áreas (tabela 3.1).

A área estrutural descreve os elementos do sistema e as suas relações e é a base para as vistas da área dinâmica. Estas vistas descrevem o comportamento do sistema ao longo do tempo. A área de gestão do modelo é onde está descrita a organização dos próprios modelos em pacotes. Pode ainda existir outra área relacionada com a extensibilidade, como seja a descrição de restrições, e que é aplicável aos elementos de todas as vistas.

Tabela 3.1 – **Diagramas e Vistas em UML (adaptado de [Rumbaugh et al., 1999]).**

Áreas	**Vistas**	**Diagramas**
Estrutural	Estática	Classes
	Casos de Uso	Casos de Uso
	Física	Componentes
		Entrega
Dinâmica	Máquina de Estados	Estados
	Actividades	Actividades
	Interacção	Sequência
		Colaboração
Gestão do Modelo	Gestão do Modelo	Classes

A UML é a linguagem de modelação usada nos dois processos de engenharia de software apresentados mais à frente e escolhidos para suporte aos casos de estudo: o Rational Unified Process [Jacobson et al., 2000], e o MIDAS [Fernandes, 2000].

3.2.1 Vista Estática

Nesta vista não são descritos comportamentos dependentes do tempo mas sim conceitos relativos ao produto e à sua implementação. Os constituintes principais desta vista são as classes e as suas relações, como sejam a associação, a generalização, ou vários tipos de dependência. Esta vista é concretizada através dos diagramas de classes, de que se mostra um exemplo na fig. 3.1.

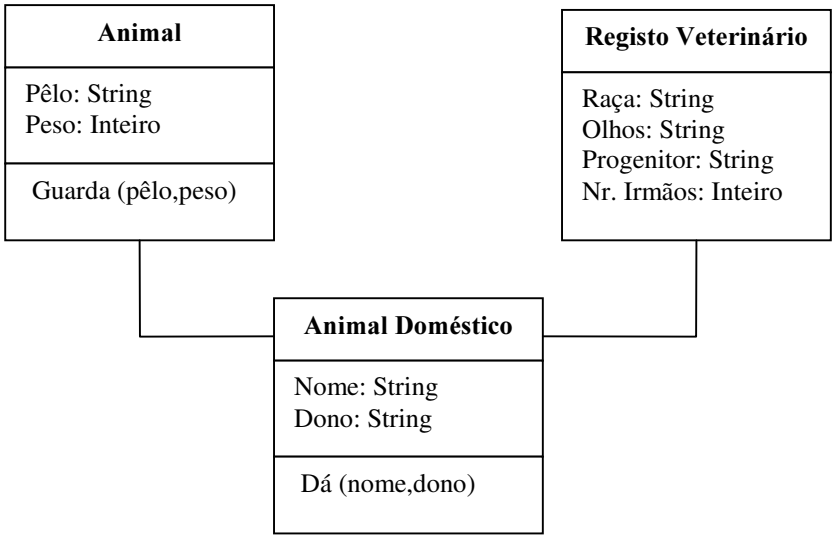

Fig. 3.1 – **Diagrama de Classes.**

De notar que a descrição das classes, à medida que o desenvolvimento ocorre, aumenta de precisão. Assim, nas fases mais avançadas do desenvolvimento a descrição das classes pode comportar decisões de desenho e detalhes de implementação.

3.2.2 Vista dos Casos de Uso

Nesta vista é modelada a funcionalidade do sistema da forma que os utilizadores externos, os actores, a vêem (fig. 3.2). São assim descriminados os actores e os casos de uso, e a participação de cada actor nos casos de usos.

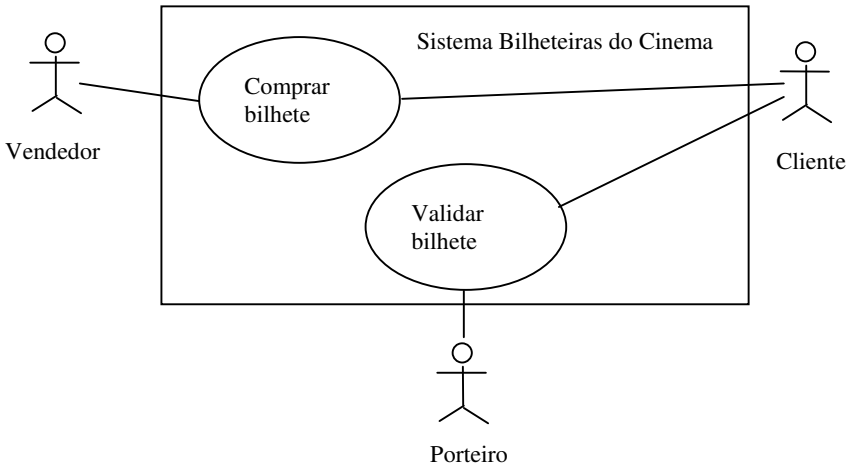

Fig. 3.2 **– Diagrama de Casos de Uso.**

No exemplo de diagrama de casos de uso da fig. 3.2, o cliente é um actor do sistema porque interage com ele. O caso de uso "Comprar bilhete", que representa uma funcionalidades ou serviços disponibilizados pelo sistema aos seus utilizadores, pode ser realizado pelo cliente através, por exemplo, da Internet. Neste tipo de diagramas estão apenas representados os actores que interagem directamente com o sistema.

3.2.3 Vistas Físicas

As vistas físicas do sistema fornecem, através do diagrama de componentes (fig. 3.3) na vista de implementação e através do diagrama de entrega na vista de entrega, uma visão dos componentes do sistema e das suas dependências e do seu arranjo em tempo de execução. Ao passo que nas vistas anteriores, estática e casos de uso, se modelava a aplicação de um ponto de vista lógico, estas vistas fornecem uma visão do mapeamento das classes em componentes da aplicação e eventuais nodos físicos distribuídos onde a aplicação irá ser executada.

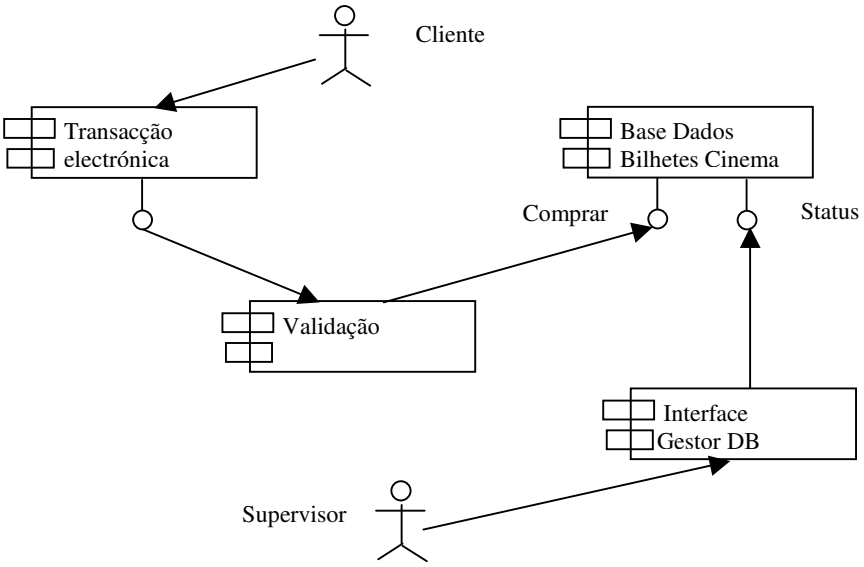

Fig. 3.3 **– Diagrama de Componentes.**

A vista física de implementação modela, no diagrama de componentes, os componentes do sistema, ou seja, quais as partes em que a aplicação está dividida, e as relações entre elas. Na vista física de entrega e através do diagrama de entrega, são representados os arranjos em tempo de execução dos componentes pelos nodos físicos do sistema tais como computadores, dispositivos, ou memória. No diagrama de entrega podem ser também mostrados os diversos tipos de nodos e os tipos de comunicações entre eles.

3.2.4 Vista da Máquina de Estados

A máquina de estados modela as hipóteses possíveis de transições de estado para um objecto de uma dada classe. Cada estado modela um período de tempo no ciclo de vida de um objecto. Quando ocorre um evento pode acontecer uma transição no estado dos objectos. A visualização da máquina de estados é feita com o recurso aos diagramas de estados (fig. 3.4).

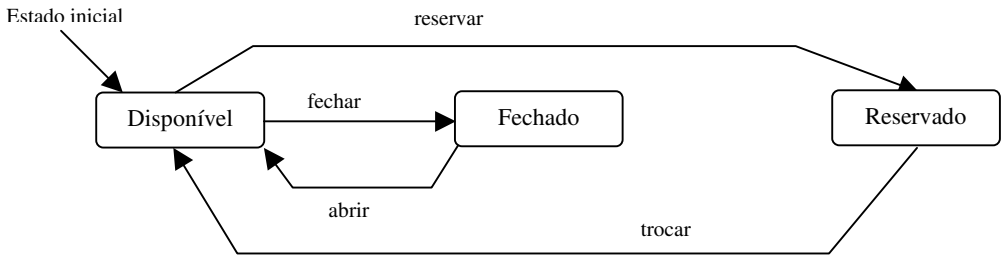

Fig. 3.4 **– Diagrama de Estados.**

As máquinas de estado podem ser usadas para descrever interfaces, subsistemas reactivos mas também objectivos passivos.

3.2.5 Vista das Actividades

A vistas das actividades é visualizada através dos diagramas de actividades (fig. 3.5) que é onde se mostram exactamente as acções necessárias para executar uma tarefa.

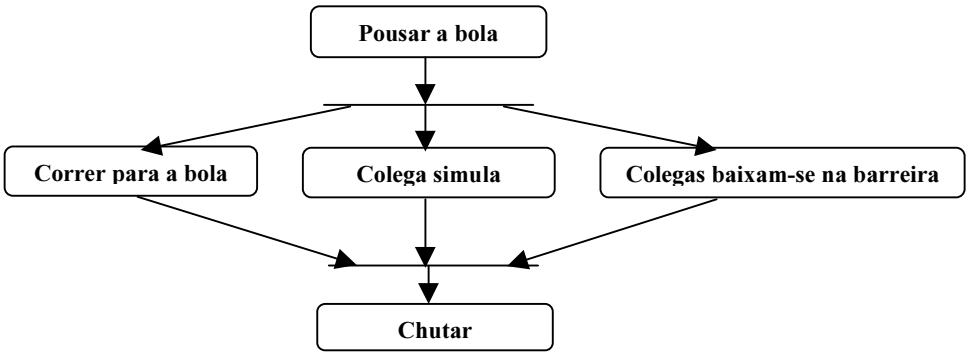

Fig. 3.5 – **Diagrama de Actividades.**

Neste tipo de diagramas, que são variantes da máquina de estados, podem existir actividades, ou grupos de actividades, sequenciais ou concorrentes.

3.2.6 Vista de Interacção

Esta vista descreve as sequências de mensagens trocadas entre os papéis que implementam o comportamento do sistema e dá uma visão do fluxo do controlo ao percorrer vários objectos. A visualização de informação é feita através dos diagramas de sequência (fig. 3.6) e dos diagramas de colaboração (fig 3.7).

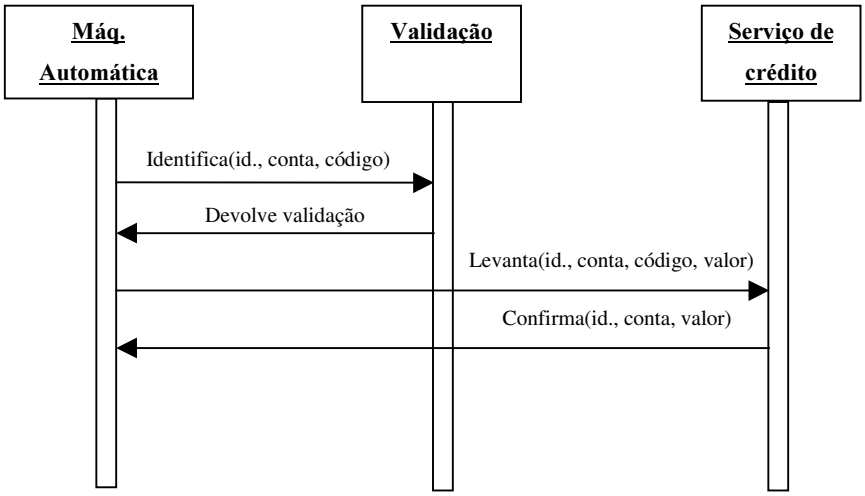

Fig. 3.6 **– Diagrama de Sequência.**

Os diagramas de sequência mostram cenários de trocas de mensagens entre objectos com uma ordenação temporal. Uma utilização para os diagramas de sequência é a modelação do comportamento de um caso de uso. Cada mensagem neste diagrama corresponderia assim a uma operação numa classe ou a um evento numa da máquina de estados. Os diagramas de colaboração, que são equivalentes aos diagramas de sequência, modelam os objectos e as ligações que têm significado numa interacção.

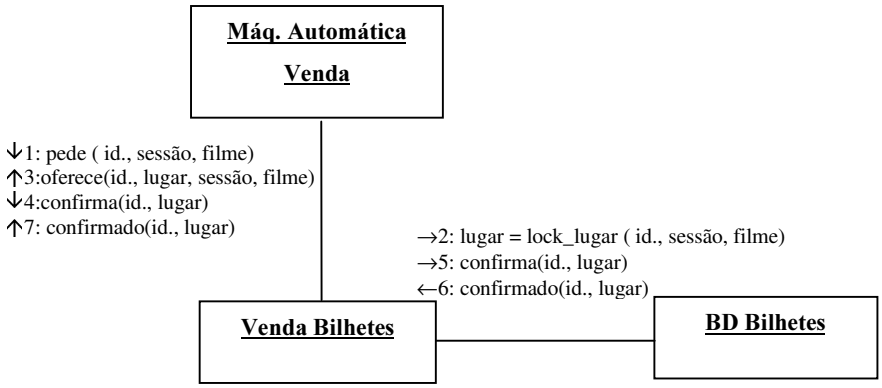

Fig. 3.7 **– Diagrama de Colaboração.**

Os objectos e as ligações expressos nos diagramas de colaboração só têm significado no contexto da interacção analisada, podendo, estes diagramas, ser usados para mostrar a implementação de uma operação conjuntamente com os seus parâmetros e variáveis locais.

3.2.7 Vista de Gestão do Modelo

A vista de gestão do modelo dá uma visão da organização do próprio modelo. Um modelo é composto de pacotes, sendo que estes podem guardar elementos do modelo como classes, máquinas de estado, e casos de uso. É normal existir uma representação do sistema, recorrendo normalmente a diagramas de classes, com uma decomposição em subsistemas cada um dos quais contendo vários pacotes, e as conexões existentes entre eles.

3.2.8 Mecanismos de Extensibilidade

UML inclui três construtores que permitem a sua própria extensibilidade sem pôr em causa o seu meta-modelo [Rumbaugh et al., 1999]:

- Restrições: são constrições semânticas em linguagem natural ou formal;
- Atribuições: são pares etiqueta/valor que podem ser aplicados a qualquer elemento individual do modelo;
- Estereótipos: são elementos de modelação construídos à custa de outros onde apesar do conteúdo e da forma da informação que contêm ser a mesma do elemento base, o seu significado e uso são diferentes.

3.3 O Processo Unificado de Desenvolvimento de Software

Actualmente as organizações que desenvolvem software estão perante dois tipos de forças por vezes contraditórias. Por um lado, a necessidade de produzir cada vez mais rapidamente produtos de software devido à concorrência e a novas exigências dos clientes. Por outro a necessidade de gerar produtos com qualidade mas respeitando orçamentos reduzidos. Assim, torna-se necessário que este tipo de organizações sejam ágeis e eficientes. Para que estas propriedades se manifestem, é necessário o suporte de um processo que cubra todas as fases do desenvolvimento dos produto de software e que seja suficientemente abstracto e abrangente para lidar com a heterogeneidade dos sistemas actuais, mas com uma perspectiva de racionalização de meios utilizados.

Um exemplo de processo de engenharia de software que exibe potencialidades para cumprir os objectivos de uma organização actual que desenvolva produtos de software é o Processo Unificado[1] (RUP) [RUP, 2001]. Este processo potencia a produtividade das equipas envolvidas com os requisitos, desenho, teste, ou gestão do projecto visto, fornecer linguagem, vistas, e processo comuns. A produção de documentação num projecto, tarefa tradicionalmente consumidora de recursos, é obviada pela utilização de modelos, em vez de grandes volumes de documentação, que proporcionam representações semanticamente ricas do produto em desenvolvimento [Jacobson et al., 1999]. O RUP pode também ser visto como um guia para o uso eficaz de UML [Booch, 2000], que pode ser auxiliado por um conjunto de ferramentas [Rational, 2000a] capazes de automatizar partes significativas do processo. Este processo é escalável, podendo adequar-se ao desenvolvimento feito por pequenas ou grandes equipas, utilizando para isso um *Development Kit* que auxilia na configuração do processo e na adaptação às necessidades de uma organização de desenvolvimento de software.

O RUP permite que estejam incluídas no processo de desenvolvimento de software características que normalmente as organizações com sucesso exibem:

- **Desenvolvimento Iterativo**: Devido à complexidade actual dos sistemas é muitas vezes difícil definir sequencialmente o problema, desenhar a sua solução, construir o produto, e testá-lo. Assim, uma abordagem iterativa permite um entendimento crescente do problema e porque em cada iteração há a geração de uma versão, a geração de soluções cada vez mais adequadas;
- **Gestão de Requisitos**: O RUP descreve como extrair, organizar, e documentar as funcionalidades e restrições, como capturar e comunicar os requisitos de negócio, e como registar e documentar os compromissos e decisões tomadas, através das noções de casos de uso e de cenários;
- **Arquitecturas Baseadas em Componentes**: Sendo os componentes reutilizáveis uma força de mercado, e que permitem que subsistemas testados e operacionais já existam para o cumprimento de uma funcionalidade específica, o RUP suporta plataformas para que estes sejam montados numa arquitectura bem definida de um modo ad hoc ou em arquitecturas como CORBA ou COM;
- **Modelação Visual do Software**: Dado o RUP utilizar UML, as abstracções visuais do software são um meio para comunicar diferentes aspectos do

[1] O Processo Unificado (Rational Unified Process) é propriedade intelectual da Rational Software Corporation.

software, através dos vários diagramas de UML, e um modo preciso e sem ambiguidades;

- **Gestão da Qualidade do Software**: Esta gestão está integrada no RUP com métricas e critérios objectivos para garantir a confiabilidade, funcionalidade, e desempenho tanto da aplicação como do sistema completo;
- **Controlo de Alterações ao Software**: É pacífico que os produtos de software estão actualmente sujeitos à mudança. Assim, o RUP descreve como controlar, registar, e monitorar as mudanças ao longo das várias iterações no desenvolvimento ou na manutenção do software.

O RUP é baseado em três ideias fundamentais [Jacobson et al., 1999]:
- É dirigido aos casos de uso, ou seja, tem a orientação que o cliente, humano ou outro sistema informático, pretende dele;
- É iterativo e incremental, permitindo um aperfeiçoamento do produto;
- É orientado à arquitectura, isto é à forma do produto, porque uma orientação puramente funcional como a dos casos de uso não é suficiente para garantir um produto com qualidade.

O ciclo de vida do RUP é uma repetição de ciclos (fig. 3.8), cada um deles concluído com o lançamento de uma nova versão, até que a versão produzida satisfaça os requisitos do cliente. Cada ciclo é composto por quatro fases: Início, Elaboração, Construção, e Transição, sendo que cada uma termina com um marco no projecto onde se tomam decisões críticas e se verifica o cumprimento de objectivos intermédios como a produção de certos documentos e modelos. Em cada uma destas fases existem subdivisões, as iterações, que de uma forma controlada e sempre orientadas aos objectivos refinam a fase onde se inserem.

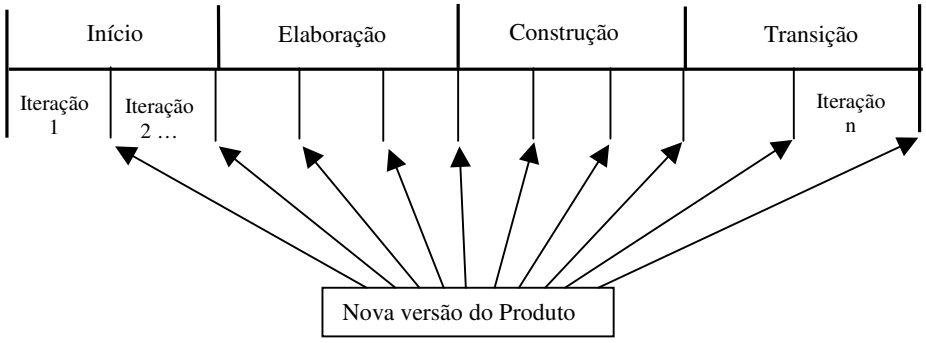

Fig. 3.8 **– Um Ciclo no Processo Unificado.**

O produto final do RUP é composto não só pelas fontes e executáveis do produto de software mas também pelos requisitos, casos de uso, especificações não funcionais, e plataformas de testes, ou seja por tudo o que é necessário para um eventual novo ciclo no desenvolvimento do produto.

O RUP pode ser descrito através de duas dimensões: a dimensão temporal que acomoda os aspectos dinâmicos do processo e é expressa em termos de ciclos, fases, iterações, e marcos do processo; e a dimensão estática, que acomoda as actividades, artefactos (pedaços de informação produzida ou usada no processo, como o modelo de casos de uso), trabalhadores, e fluxos de trabalho.

3.3.1 A Dimensão Temporal do Processo

A organização dinâmica do processo ao longo do tempo está dividida em ciclos cada um terminando numa nova geração do produto. Em cada ciclo existem quatro fases consecutivas, cada qual com um propósito específico e terminada com um marco do processo.

A **Fase de Início** é onde se delimita o âmbito do projecto e se enquadra o produto nos outros sistemas com que vai interagir. É necessário identificar todos os actores e descrever os mais significativos. Estabelecem-se também os critérios de sucesso, riscos envolvidos, estimativa dos recursos necessários, e um plano com as datas dos marcos mais importantes do projecto [Rational, 2000a].

No final desta fase deverão existir:

- Um documento com uma visão dos requisitos fundamentais do projecto, restrições existentes, e características principais;
- Um modelo de casos de uso (tipicamente 10% a 20% dos que irão aparecer no modelo final);
- Um glossário do projecto ou alternativamente um modelo do domínio do projecto;
- Uma análise das perspectivas de negócio, como previsões financeiras ou o reconhecimento no mercado;
- Uma estimativa inicial dos riscos;
- Um plano do projecto com as fases e iterações;
- Um ou vários protótipos.

No final desta fase ocorre o marco "Objectivos do Ciclo de Vida" e cujos critérios de avaliação, entre outros, são:

- A análise dos gastos planeados e dos já efectuados;
- Credibilidade das estimativas de custos, prioridades, riscos;
- Profundidade dos protótipos desenvolvidos.

Se este marco não tiver uma apreciação positiva o projecto pode ser cancelado ou ter de alterado significativamente.

Na **Fase de Elaboração** faz-se a análise do domínio do problema, estabelece-se a arquitectura do sistema, desenvolve-se o plano do projecto, e eliminam-se os riscos evidentes do projecto. No final desta fase deverão existir:

- Um modelo de casos de uso (80% completo) com todos os actores identificados e a maioria dos casos de uso desenvolvidos;
- A captura de requisitos não funcionais e de outros não relacionados com os casos de uso;
- Uma descrição da arquitectura do software;
- Um protótipo executável já com os aspectos arquitecturais incorporados;

- Uma lista de riscos e uma análise do negócio já revistas;
- Um plano de desenvolvimento para o projecto com um plano mostrando as iterações e os critérios de avaliação para cada iteração;
- Um manual de utilizador preliminar.

No final da Fase de Elaboração ocorre o segundo marco do projecto. No marco "Arquitectura do Ciclo de Vida", os critérios de avaliação envolvem as respostas às questões:

- A visão do produto está estável?
- A arquitectura escolhida é estável?
- O protótipo executável mostrou que os maiores riscos já foram eliminados?
- O plano para a Fase de Construção existe e é detalhado e preciso? É baseado num conjunto de estimativas credível?
- Todos os engenheiros envolvidos no desenvolvimento do produto, assim como clientes e os gestores do projecto, concordam que os objectivos podem ser alcançados se ocorrer o desenvolvimento do produto como está planeado e na arquitectura escolhida?
- A razão entre os custos planeados e os reais é aceitável?

Assim como no marco da fase anterior também o projecto pode ser cancelado ou refeito.

Na **Fase de Construção** todos os componentes e características da aplicação que ainda não tenham sido introduzidos são desenvolvidos e integrados no produto e são feitos testes exaustivos ao produto. Esta é a primeira fase de produção, dado as anteriores serem essencialmente fases de desenvolvimento de conceitos. De modo a acelerar o processo podem existir equipas paralelas para a fase de construção, aumentando no entanto a complexidade da gestão de recursos e da sincronização. No final desta fase existe já um produto pronto para ser entregue aos clientes finais, e que incluirá:

- O produto de software integrado nas plataformas adequadas;
- Os manuais dos utilizadores;
- Uma descrição da versão actual.

No final desta fase ocorre o terceiro marco "Capacidade das Operações Iniciais", em que se decide se o software está pronto para passar à utilização. A versão do produto no final desta fase é também chamada por vezes de versão beta [Rational, 2000a]. Os critérios de avaliação para a fase de construção envolvem a resposta, entre outras, às questões:

- O produto está suficientemente estável para ser entregue aos utilizadores?
- A razão entre os custos planeados e os reais continua aceitável?

No caso de este marco não ter uma apreciação positiva a fase de transição terá que ser adiada até pelo menos o aparecimento de uma nova versão.

A **Fase de Transição** tem como objectivo a transferência do produto para a comunidade de utilizadores. Senão o sistema todo, pelo menos uma parte significativa e que faça o sistema usável, já deve estar concluída e com um nível aceitável de

qualidade, bem como a documentação para os utilizadores. Devem existir as seguintes actividades:

- Testes beta para validar o novo sistema quando confrontado com as expectativas dos utilizadores;
- Operação paralela com os sistemas legados que vai substituir;
- Conversão de bases de dados operacionais;
- Formação aos utilizadores e aos responsáveis pela manutenção do software;
- E se for caso, a entrega do produto às equipas de marketing, distribuição, e vendas.

Nesta fase podem ocorrer várias iterações, com o lançamento de versões beta, versões para correcção de erros, versões que melhoram o desempenho do sistema. Como esta é uma fase que tem como clientes os utilizadores finais deve ser dada atenção principalmente às suas sugestões para a instalação, a configuração, e a utilização do produto. Os objectivos principais desta fase é que os utilizadores sejam auto-suficientes na utilização do produto, que os planos e datas para entrega da versão final estejam de acordo com a realidade, e que o produto final seja alcançado o mais rapidamente possível mas com os custos controlados.

No final desta fase ocorre o quarto marco "Liberação do Produto" onde se responde às questões:

- Os utilizadores estão satisfeitos?
- A razão entre os custos planeados e os reais continua aceitável?

3.3.2 A Estrutura Estática do Processo

O RUP usa quatro elementos básicos para modelar as características estáticas [Jacobson et al., 1999] do produto:

- Os *trabalhadores*, ou seja a resposta à questão "Quem?", que são o papel da pessoa ou pessoas que desempenham uma tarefa e não as pessoas em si;
- As *actividades*, a resposta a "Como?", são bocados de trabalho sempre atribuídos a algum trabalhador;
- Os *artefactos*, a resposta a "O Quê?", que são pedaços de informação produzida, modificada, ou usada pelo processo e que são os elementos tangíveis do processo;
- Os *fluxos de trabalho*, a resposta a "Quando?", que são a descrição das sequências de trabalho que produzem algum resultado significativo ou mostram a interacção entre os trabalhadores e que em UML podem ser expressas por diagramas de sequência, de colaboração, ou actividades.

3.3.3 Os Fluxos de Trabalho Fundamentais

No RUP existem nove fluxos de trabalho fundamentais [Rational, 2000] divididos em fluxos de trabalho nucleares e fluxos de trabalho auxiliares.

Os seis **Fluxos de Trabalho Nucleares** são:

- A *Modelação do Negócio*, onde os processos de negócio são modelados recorrendo a casos de uso de negócio. Estes casos de uso de negócio são analisados para compreensão de como o negócio suporta os seus processos;

- Os *Requisitos*, onde se descreve o que o sistema deve fazer. É criado um documento de visão do projecto onde constam a identificação dos actores, a identificação dos casos de uso, e a descrição dos casos de uso. O modelo de casos de uso vai ser usado ao longo de todo o processo;
- A *Análise e o Desenho*, onde se mostra como o sistema vai ser concretizado na fase de implementação. Deste fluxo resulta um modelo de desenho que serve como uma abstracção da estrutura do código fonte com o desenho das classes estruturado por pacotes e subsistemas. As actividades de desenho estão focadas na noção de arquitectura;
- A *Implementação*, onde se implementam as classes e os objectos, se define a organização do código por eventuais níveis, se testam os componentes desenvolvidos, e onde se integram os vários subsistemas para produzir um produto executável;
- O *Teste*, onde se verifica as interacções entre os objectos, a integração dos componentes no sistema final, a correcta implementação de todos os requisitos, e onde também se identificam e se assegura que não passam defeitos após a entrega do produto;
- A *Entrega*, cujo objectivo é produzir versões de produtos e entregá-las aos utilizadores finais e que inclui actividades como a distribuição do software, a assistência aos utilizadores, ou a migração de dados ou sistemas existentes.

Paralelamente a estes seis fluxos de trabalho nucleares existem outros três **Fluxos de Trabalho Auxiliares**:

- A *Gestão do Projecto*, que inclui a eleição entre objectivos opostos, a gestão dos riscos, a superação das dificuldades, a monitorização do projecto, a escolha de pessoal, tudo para se conseguir produzir um produto de qualidade dentro das limitações de tempo e de orçamento apontadas inicialmente;
- A *Gestão da Mudança e Configurações*, onde se tenta evitar a confusão e os conflitos entre os artefactos produzidos em paralelo pelos diversos trabalhadores;
- O *Ambiente*, que tem como propósitos dotar a organização que está a desenvolver o software de processos e ferramentas que ajudem a equipa de desenvolvimento. Existem actividades para, por exemplo, configurar o processo no contexto do desenvolvimento de um produto específico, embora no RUP não estejam descritas actividades para seleccionar, adquirir, ou construir ferramentas de suporte.

3.4 Metodologia MIDAS

A metodologia MIDAS [Fernandes, 2000] é uma metodologia orientada ao objecto para o desenvolvimento de sistemas embebidos que usa uma abordagem operacional e faz uso de especificações visuais neutras e multi-vista e que dá uma ênfase especial à fase de análise. A utilização de tecnologia orientada a objectos permite essencialmente a uma mais fácil reutilização de código já desenvolvido e testado [Fowler, 1997], encurtando-se o tempo de execução do processo de desenvolvimento, diminuindo os custos e aumentando a qualidade do produto final de software.

Os desafios actuais para os engenheiros de software são o desenvolvimento de sistemas complexos e heterogéneos e que abarcam muitas vezes sistemas embebidos.

Para atacar com sucesso o desenvolvimento de um produto com estas características e para manter a coerência com as restantes partes de um processo de desenvolvimento unificado é necessário o suporte de uma metodologia, por exemplo a MIDAS, que use a mesma notação, embora nela apenas se use um conjunto restrito dos tipos de diagramas UML, que os restantes elementos do projecto.

A metodologia MIDAS apresenta um *macro-processo* sequencial de processo (com escala temporal de meses ou anos) composto pelas fases de análise de requisitos, análise, concepção, implementação, e de testes com ligações às três últimas. Ao nível do *micro-processo*, com uma escala temporal ao nível das semanas ou dias, é seguido o modelo em espiral de Boehm.

3.4.1 A Fase de Análise

Em MIDAS a fase de análise é composta por diversos passos que transformam os modelos obtidos na fase de estudos de viabilidade em modelos a usar nas fases de concepção e implementação [Fernandes, 2000]. A fase de análise inicia-se com a captura do ambiente através de um diagrama de contexto que mostra todos os actores que interagem com o sistema. O passo seguinte é a modelação funcional, com o auxílio do diagrama de casos de uso que é um refinamento do diagrama de contexto e onde são explicitados, para além dos actores, os casos de uso. Segue-se a descrição, onde se faz uma descrição textual que caracteriza os aspectos funcionais dos casos de uso. Surge então a fase de transformação onde os casos de uso são transformados, através de um conjunto de regras próprio, em objectos. Em seguida, e visto a MIDAS ser desenhada para sistemas embebidos, existe uma fase de modelação do sistema controlado que permite determinar e explicitar por exemplo as características mecânicas ou eléctricas do sistema controlado e depois um fase de modelação comportamental do sistema controlado. Surge então a fase de selecção de quais os objectos que irão seguir o desenvolvimento através dum critério definido, e os objectos criados passam à fase de classificação onde se atribuem estes à classe a que pertencem e se indicam as relações entre as classes. Posteriormente, e usando diagramas de sequência, entra-se na fase de modelação de protocolos onde se indicam as interacções entre os objectos do sistema. Em seguida, na fase de formalização, com base nos diagramas de sequência e na especificação da utilização do sistema controlado são criados diagramas de estados para os objectos cujo comportamento dinâmico o justifique. Passa-se então à fase de especificação onde com base nos diagramas de objectos, de classes, e de estados se cria uma especificação executável em linguagem Oblog (ou outra que a substitua, como Java). Posteriormente, na fase de simulação, usando a especificação Oblog como entrada, o sistema é parcial ou completamente simulado, obtendo-se como saídas diagramas de sequência que descrevem o funcionamento das partes simuladas. Por último surge a fase de comparação onde se verifica se o comportamento esperado, descrito nos diagramas de sequência da fase de formalização, corresponde ao comportamento especificado na fase de simulação.

3.4.2 As Fases de Concepção e de Implementação

As fases de concepção e de análise não são tratadas em detalhe na metodologia MIDAS, embora se sugiram [Fernandes, 2000] alguns passos para transformar os diagramas de objectos em aspectos do modelo de implementação:

- Identificação do ambiente de implementação;

- Desenvolvimento duma primeira versão do ambiente de concepção;
- Descrição da interacção entre os objectos.

Devido ao uso de especificações em linguagem Oblog (ou alternativamente outra linguagem como Java) e de a partir destas se poder gerar código, as especificações, neste modelo de desenvolvimento de sistemas, podem ser consideradas executáveis.

4. Qualidade nos Processos de Desenvolvimento

Neste capítulo são abordados alguns aspectos relacionados com a qualidade nos processos de desenvolvimento e nas organizações que desenvolvem software, e uma forma de avaliação: o modelo Capability Maturity Model.

4.1 Introdução

A qualidade formal de um produto é o grau de cumprimento, ou seja de implementação no produto dos requisitos postos inicialmente pelo cliente. Quanto mais requisitos incorporar e quantas mais repetições do produto forem iguais entre si, mais qualidade tem o produto. A qualidade de um produto de software está directamente relacionada com a do seu processo de desenvolvimento, bem como com a organização onde este está a decorrer. É certo que um bom processo de desenvolvimento a decorrer numa organização que proporcione boas condições de trabalho não garante um bom produto de software, mas por outro lado, um processo descontrolado ou inexistente a decorrer numa organização caótica tem muitas mais probabilidades de produzir software com má qualidade. Assim, a qualidade de um processo de desenvolvimento de software pode ser avaliada pela qualidade média dos produtos desenvolvidos recorrendo a esse processo.

4.2 Os Níveis do Capability Maturity Model

As organizações clientes de produtos de software quando fazem estudos de mercado, ou quando lançam pedidos de propostas para o fornecimento de produtos de software, são confrontadas com um problema de escolha. Podem existir propostas de valor baixo e com características técnicas adequadas, mas vindas de organizações acerca das quais não existe um conhecimento profundo das garantias que oferecem para cumprir com o prometido.

Deste modo, é necessária a existência de um padrão de avaliação para assegurar a maturidade do processo de desenvolvimento de software de uma organização, e que pode ser obtida pelo uso do Capability Maturity Model (CMM) [Paulk et al., 1996]. O CMM descreve como uma organização que desenvolve software pode passar de um processo de desenvolvimento ad hoc para um processo maduro e disciplinado. O CMM cobre as áreas de planeamento, engenharia, gestão do desenvolvimento do software e manutenção.

É, no entanto, necessário realçar que as organizações contratantes podem ser "enganadas" durante entrevistas ou no fornecimento de documentação [O'Connell et al., 2000], fazendo com que as organizações contratadas atinjam valores mais altos no CMM. Mesmo que esta situação não ocorra, uma organização com um nível alto no CMM não é o garante que o software que vai produzir seja de alta qualidade [Voas, 1999].

Um dos processos de desenvolvimento de software que permite alcançar os níveis 2 (repetível) ou 3 (definido) do CMM é o RUP [Rational, 2000b]. O CMM tem cinco níveis de maturidade (Nível 1 até Nível 5), cada um dos quais composto por Áreas

Chave do Processo (KPA[1]). A passagem de um nível para outro faz-se através de um procedimento de melhoria contínua, privilegiando uma abordagem evolucionária em vez do uso de inovações radicais. Cada KPA identifica um grupo de actividades relacionadas que quando executadas em conjunto, permitem alcançar um conjunto de objectivos considerados importantes para estabelecer a capacidade do processo num nível de maturidade.

4.2.1 O Nível Inicial

Neste nível, o processo de desenvolvimento de software é um processo ad hoc e por vezes caótico. Poucos procedimentos estão descritos e o sucesso depende de esforços individuais. A organização não fornece um ambiente estável para o desenvolvimento e a manutenção de software. Mesmo que exista um bom processo de engenharia de software, a organização não está preparada para o suportar, e tipicamente em épocas de crise o processo é abandonado e passa-se a ciclos de codificação e teste. A capacidade das organizações neste nível é imprevisível porque os orçamentos, prazos, funcionalidade, e qualidade final do produto são aleatórios.

No entanto, é possível a estas organizações produzirem produtos de qualidade mas sendo estes o resultado do esforço e competência de algumas pessoas.

4.2.2 O Nível Repetível

Neste nível existem processos básicos para a gestão de projectos e procedimentos implementados que permitem o seguimento dos custos, prazos, e funcionalidade. Existe também a necessária disciplina para se conseguir repetir sucessos prévios no desenvolvimento de produtos semelhantes. O planeamento e a gestão de novos projectos são feitos recorrendo à experiência. Existem standards para os projectos de software.

A gestão dos projectos deve-se focar no seu próprio processo para assim alcançar um processo de desenvolvimento de software controlado.

4.2.3 O Nível Definido

No nível definido, o processo standard para o desenvolvimento e manutenção de software está documentado e inclui tanto os aspectos técnicos como os aspectos de gestão. Existe também um programa de formação na organização, que garante que os gestores e o grupo de trabalho tenham os conhecimentos e as competências necessárias para a realização das tarefas que lhes estão atribuídas.

Tanto o processo técnico de desenvolvimento de software como o processo de gestão são estáveis e repetíveis. São controlados os custos, prazos, e funcionalidades além da qualidade do software ser monitorada.

4.2.4 O Nível Gerido

No nível gerido, a organização define objectivos quantificados para a qualidade do produto e para a qualidade do processo. Nos projectos alcança-se o controlo dos produtos e do processo pela definição de intervalos para o desempenho do processo. A

[1] Key Process Areas

capacidade de desenvolvimento de software de organizações deste nível é previsível dado o processo ser medido e operar dentro de limites mesuráveis.

Os produtos são previsivelmente de alta qualidade.

4.2.5 O Nível Optimizado

Neste nível toda a organização está focada no processo de melhoria contínua. A organização tem a capacidade de identificar e corrigir fraquezas no processo com o objectivo de prevenir a ocorrência de defeitos.

As equipas de desenvolvimento de processos deste nível analisam os defeitos encontrados para determinar a sua causa. Os processos de software são avaliados para prevenir tipos conhecidos de erros, e o conhecimento obtido num processo é disseminado para os outros.

4.3 A Aplicação do CMM

A atribuição de um dos cinco níveis do CMM é feita pelo resultado obtido num questionário de avaliação que indica o grau de maturidade da organização. Para cada nível existe um conjunto de KPAs que descreve as funções que devem existir para satisfazer uma boa prática num nível particular [Pressman, 1997]. Assim, em cada um dos níveis devem existir as seguintes KPAs:

Nível 2

- Gestão da Configuração do Software;
- Garantia da Qualidade do Software;
- Gestão de Subcontratos de Software;
- Visão Global e Monitorização do Projecto de Software;
- Planeamento do Projecto de Software.

Nível 3

- Revisões;
- Coordenação Entre Grupos;
- Engenharia do Produto de Software;
- Gestão Integrada do Software;
- Programa de Formação;
- Definição do Processo de Organização;
- Foco no Processo de Organização.

Nível 4

- Gestão da Qualidade do Software;
- Gestão Quantitativa do Processo;

Nível 5

- Gestão de Mudanças no Processo;
- Gestão de Mudanças de Tecnologias;
- Prevenção de Defeitos.

Cada KPA é definida à custa de um conjunto de práticas essenciais que contribuem para alcançar os seus objectivos. Essas práticas são políticas, procedimentos, e actividades que devem ocorrer antes de uma KPA estar totalmente instituída.

5. Organizações Orientadas para Processos

Neste capítulo faz-se uma proposta dum modelo para uma organização genérica orientada a processos com o intuito de esta se adaptar mais à frente ao caso específico de uma organização que desenvolve software. Adicionalmente, esta proposta serve também como modelo para qualquer organização orientada a processos e seja o alvo do desenvolvimento de um produto de software. Este modelo segue a filosofia de criar um número relativamente pequeno de processos de alto nível, de forma a se controlar a complexidade, sem no entanto pôr em causa a pormenorização desses processos de alto nível em sub-processos.

5.1 Introdução

O conceito de organizações orientadas para processos [Hammer, 1997] é uma forma de focalizar as actividades de uma organização de um modo orientado para os seus clientes. As actividades de uma organização são, assim, orientadas e avaliadas pelo cliente interno ou externo. A perspectiva normal de um cliente é querer que os seus pedidos de produtos ou serviços sejam satisfeitos duma maneira eficiente e com qualidade. A reengenharia, e a sua orientação por processos, deve ser aplicada para antecipar a mudança e não como medida correctiva quando uma organização apresenta maus resultados [Rodrigues et al., 1995].

Em organizações mais tradicionais (fig. 5.1 a), os interesses do cliente colidem frequentemente com os interesses internos na cadeia de produção. As disposições tradicionais hierárquicas em forma de pirâmide nas organizações são um obstáculo real à satisfação dos clientes pois neste tipo de organizações, as funções estão distribuídas por departamentos. Assim, é normal existir, por exemplo, um departamento responsável pelas compras de materiais, outro pela produção, outro pela expedição de produtos. Cada um destes departamentos tem objectivos próprios, sejam eles impostos pela gestão de topo ou decididos internamente no departamento, podendo criar conflitos interdepartamentais pela sua divergência.

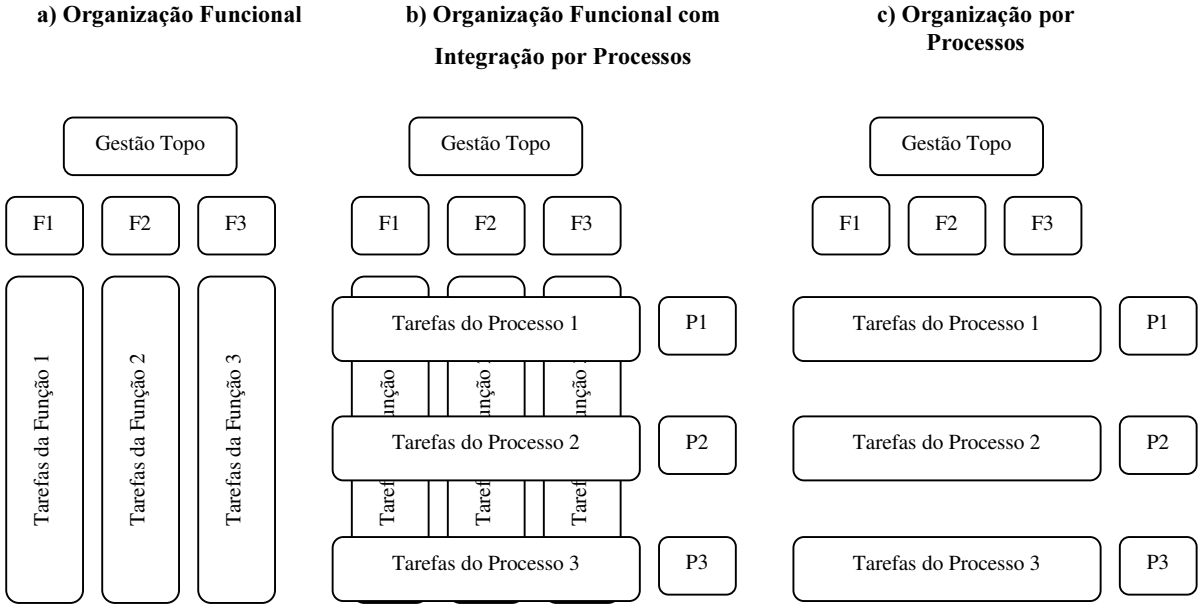

Fig. 5.1 **– Organização Funcional e Organizações por Processos.**

A abordagem duma organização orientada aos processos para lidar com estes problemas de conflitos é a criação de estruturas que tomem por seus os objectivos dos clientes. Deste modo, são criados processos coordenados que englobam funções (fig. 5.1 c), por exemplo a compra de material, a produção, ou a expedição de produtos, e cujos objectivos são os mesmos do cliente, ou seja, entregar produtos na data, com a qualidade e quantidade pretendidas, e com os custos acordados. Do mesmo modo, nas organizações funcionais com integração de processos, os objectivos dos processos deverão sobrepor-se aos objectivos dos departamentos.

A existência de uma organização apenas faz sentido se existir utilidade no seu desempenho para os clientes, visto esta ser a forma mais sustentada de se garantir o alcance dos seus objectivos.

Existem quatro princípios que questionam o papel e a forma das organizações [Hammer, 1997]:

1. A missão de uma empresa é criar valor para os seus clientes;
2. São os processos da empresa que criam valor para os seus clientes;
3. O sucesso empresarial vem da excelência no desempenho;
4. O desempenho superior de um processo obtém-se pelo projecto superior do processo, pelas pessoas certas para executá-lo, e pelo ambiente de trabalho certo.

Nas organizações orientadas aos clientes, e consequentemente aos processos, terá que existir uma fácil adaptação à mudança para continuamente satisfazer os pedidos dos clientes. Este clima é propício, portanto, à disponibilidade para a melhoria contínua. Assim, a tudo o que existe na empresa, seja produto, processo, forma de organização, ou software utilizado, podem ser acrescentados pequenos melhoramentos resultantes das actividades de melhoria contínua.

As tecnologias de informação estão no centro do potencial para reorganizar por processos uma organização [Spurr et al., 1994]. O desenvolvimento de produtos de

software para organizações deste tipo deve levar em conta a forma e as características da organização onde o produto final de software vai ser utilizado. Os processos de engenharia de software devem incorporar no conjunto de requisitos levantados junto do cliente uma modelação da organização. Obtém-se, deste modo, uma melhor adequação do produto à organização e permite-se que o processo de manutenção de software seja mais eficiente porque não necessita de resolver problemas de adaptação e também porque o produto já incorpora características que torna o problema da introdução de melhorias mais fácil de lidar.

O software é muitas vezes a realização das regras de negócio [Pressman, 1997]. À medida que estas regras mudam também o software tem obrigatoriamente que acompanhar essa mudança. No entanto, a simples implementação de processos de negócio já existentes numa organização em sistemas informáticos não demonstra todas as vantagens que essa implementação poderia fornecer [Fowler, 1997]. É desejável que paralelamente ao processo de implementação informática exista um processo de redefinição e optimização das formas de trabalho, concretizando-se deste modo a oportunidade fornecida por essa implementação. O desempenho de uma organização que não se limite apenas à automatização de velhas formas de trabalho, mas inclua novas tecnologias informáticas e a preocupação constante na melhoria dos seus processos de negócio é superior a outra que não tenha essa preocupação.

A validação da melhoria dos processos das organizações deve ter sempre em conta a perspectiva de negócio, e avaliado segundo a criação de valor, para o cliente do processo [Spurr et al., 1994].

O uso de técnicas orientadas a objectos para a modelação conceptual de uma organização e dos seus processos permite que a análise de sistemas informáticos e a reengenharia de processos de negócio sejam praticamente uma só actividade [Fowler, 1997] com um conjunto comum de artefactos produzidos: diagramas de casos de uso, diagramas de transição de estados, etc..

Se complementarmos a abordagem orientada a objectos com metodologias que produzam artefactos facilmente perceptíveis por todos os intervenientes quer no desenvolvimento de software quer na definição de processos de negócio, como é o caso do Processo Unificado, obtém-se uma plataforma que permite ganhos evidentes para o cliente final, como seja o caso da criação de novas oportunidades de negócio pelo simples uso de uma tecnologia (e.g. comércio electrónico).

5.2 Os Processos nas Organizações

Os processos de negócio podem ser vistos como conjuntos de tarefas relacionadas que são executadas para atingir um determinado objectivo do negócio. Dentro do processo de negócio, as pessoas, os recursos materiais, ou os procedimentos internos são combinados e coordenados para um objectivo comum. A contratação de um novo colaborador, o desenho dum novo produto, a criação de meios auxiliares de produção, são exemplos de processos de negócio.

Uma empresa pode ser vista como um conjunto disperso de centros de competências que, quando existe uma instanciação dum processo, são agrupados e passam a partilhar um objectivo comum (fig. 5.2).

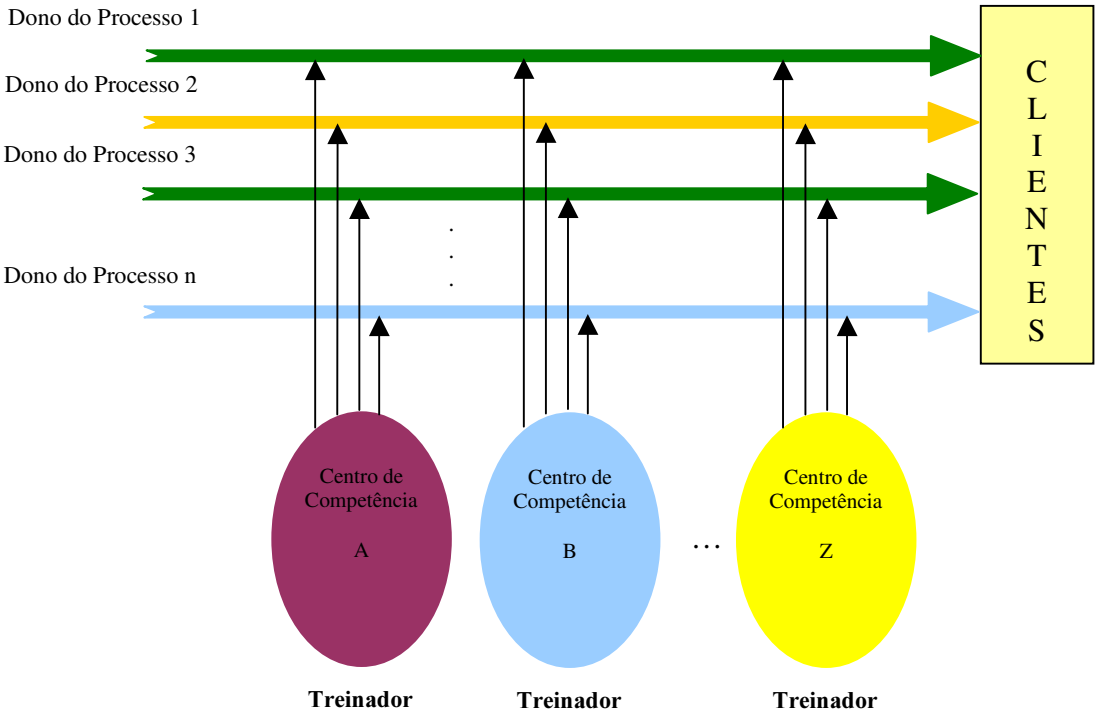

Fig. 5.2 – **Estrutura Genérica duma Organização Orientada aos Processos.**

Cada centro de competência, que deverá ser também um centro de excelência, tem atribuído um treinador cuja missão é a gestão dos membros do seu centro. O treinador é responsável pela formação contínua dos membros do seu centro. Quando solicitado, pela atribuição de membros do seu centro a um ou vários processos. Exemplos de centros de competência são a informática, as compras, ou a garantia da qualidade, e exemplos de processos são o desenvolvimento de produtos, ou definição da estratégia da organização.

Os donos dos processos são responsáveis por conseguir que o seu processo funcione de acordo com os objectivos dos seus clientes, potenciando o trabalho em equipa.

Neste tipo de organizações, os seus elementos deixam de estar sujeitos a cadeias hierárquicas de comando. Cada elemento é competente em, pelo menos, uma área e é dentro dessa área que exerce as suas funções, contando com a ajuda de um treinador para o desenvolvimento das suas competências, e com um proprietário de processo, para auxiliar na coordenação das suas actividades com as dos outros membros da equipa do processo.

A carreira dos elementos da organização não passa por promoções para posições de comando, evitando-se muitas vezes perder um bom técnico para ganhar um mau chefe, mas passa pela progressão dentro da sua área de competência, existindo em paralelo outros elementos na organização com competências para a gestão das equipas e dos processos.

5.2.1 Modelo Genérico para uma Organização Orientada aos Processos

Genericamente, uma qualquer organização existe porque fornece um conjunto de produtos ou serviços aos seus clientes. Para poderem fornecer esse conjunto de serviços ou produtos, as organizações necessitam de executar um conjunto de actividades internas. Na realidade as organizações não existem isoladas, mas fazem antes parte de mercados, onde têm organizações concorrentes, organizações suas fornecedoras, e também organizações ou indivíduos seus clientes. A geração de riqueza, no caso de organizações privadas, ou a prestação de produtos ou serviços com carácter social, no caso de organizações sem fins lucrativos, são duas faces do mesmo pressuposto básico para a existência duma organização: o preenchimento das necessidades dos clientes. Assim, as necessidades e expectativas dum cliente duma organização devem ser aspectos fulcrais para a estrutura interna duma organização.

Um processo dentro duma organização pode ser visto como um agrupamento de actividades que com uma colecção de entrada de serviços e/ou de bens materiais fornece um conjunto de saída de serviços e/ou bens materiais, usando para isso uma congregação de recursos (e.g. humanos, materiais, financeiros, tecnológicos), mas sempre na óptica da orientação para as necessidades do cliente do processo e da geração de valor acrescentado. Isto implica que os requisitos dos clientes são sempre uma das entradas que devem ser consideradas tanto no desenho como no desempenho de um processo.

Os clientes dos processos tanto podem ser clientes internos à organização como podem ser os clientes finais para os produtos ou serviços da organização.

Numa organização não existem apenas processos que trazem valor para os clientes dessa organização, materializados nos processos de valor acrescentado na fig. 5.3. É necessária a existência de mais classes de processos que assegurem por exemplo o planeamento estratégico para a organização, o recrutamento de recursos humanos ou os imperativos fiscais, instanciados nos outros tipos de processos da fig. 5.3 (processos de Gestão e processos de Suporte).

Fig. 5.3 – **Enquadramento dos Processos numa Organização Genérica.**

A introdução e utilização de processos numa dada organização pressupõe que esta tenha uma estrutura diferente da hierárquica funcional. É necessário que os processos estejam alinhados e cumpram os objectivos estratégicos da organização.

Assim, para uma organização com processos é proposta uma estrutura com os seguintes componentes:

- **Equipa de Topo para Gestão dos Processos**: Esta equipa é composta pelos membros da gestão de topo (e.g. administradores, gerentes, donos da empresa) e por todos os donos de processos. Tem como missão a revisão de todos os processos de acordo com os objectivos estratégicos da organização, rever a efectividade da gestão por processos (e.g. alterando donos dos processos), decidir sobre problemas não resolvidos de interface entre os processos[1]. Esta equipa deve reunir numa base anual ou semestral;

- **Padrinho do Processo**: Para cada um dos processos deve existir um padrinho que deve ser um membro da gestão de topo. A sua missão é ajudar e instruir o dono do processo, decidir em caso de problemas de interface entre os processos, determinar a orientação estratégica do processo (e.g. determinar que o processo de desenvolvimento de um produto tem mais prioridade sobre o desenvolvimento de outro), assegurar que o processo é uniforme em toda a organização, ou seja, que por exemplo o processo de desenvolvimento de um novo produto seja igual ao processo de desenvolvimento de outro produto;

- **Dono do Processo**: Para cada processo é necessário um dono, que para além da capacidade de gestão de processos e de equipas, deve ser escolhido pela competência nas áreas que o processo atravessa. Tem como missão liderar a Equipa Multidisciplinar do respectivo processo, sendo também responsável pelo seu desenho, medição, melhoria, e eficiência. O dono do processo, em conjunto com o padrinho, deve designar os membros da equipa multidisciplinar do seu processo, e esta deve ser composta por elementos da organização com conhecimentos efectivos dos diversos sub-processos e actividades sempre com um número de elementos que permita uma gestão efectiva (elementos a menos podem provocar falta de conhecimento e de capacidade de intervenção na realidade, mas, pelo contrário, elementos a mais podem provocar atrasos nas decisões, dificuldades em reunir, ou sobreposição de áreas de conhecimento);

- **Equipa Multidisciplinar**: As equipas multidisciplinares devem existir para cada processo de valor acrescentado, dado serem os mais críticos para os clientes da organização. Para os processos de gestão e para os processos de suporte, dependendo do tamanho do organização e dos seus objectivos estratégicos, podem também existir equipas multidisciplinares. Esta decisão deve ser tomada pelo dono e pelo padrinho do processo. A missão da equipa multidisciplinar é a monitorização do seu processo, a definição dos indicadores chave e dos objectivos do processo, assegurar que a documentação do processo está actualizada, analisar os indicadores chave,

[1] De notar que os processos duma organização não são auto-contidos. Existem muitas actividades que pertencendo a um processo têm um impacto importante, ou são mesmo partilhadas, com outro processo.

decidir a utilização e coordenar equipas de melhoria do processo, gerir as equipas de execução do processo (e.g. definir a quantidade de equipas, os critérios de segmentação, validar membros). Esta equipa deve reunir numa base mensal;

- **Líderes de Equipas e Equipas de Execução**: As equipas e os seus líderes são as instanciações de um processo. Assim, na execução de um processo na organização vão existir equipas que utilizam o processo com um enfoque específico. Por exemplo, num dado processo pode existir uma equipa para o executar em clientes industriais e outra equipa para executar o mesmo processo em clientes individuais. Esta segmentação deve ser gerida pela equipa multidisciplinar do processo, mas deve ser uma tarefa do líder da equipa a sugestão da sua composição. Cada líder de equipa deve obedecer às definições e utilizar o quadro de indicadores chave do seu processo. Podem existir processos nos quais exista apenas uma equipa e que esta coincida com os membros da equipa multidisciplinar. As equipas de execução como executam actividades e tarefas do processo diariamente devem reunir numa base semanal ou inferior.

Para que a organização por processos seja alinhada com os objectivos estratégicos da organização, é necessário que estes sejam baseados na declaração da missão e da visão, e nos princípios e valores da organização. Tendo como base os objectivos estratégicos, deve ser definido um plano que contenha o plano de negócio e os objectivos gerais que vão decidir a prioridade aquando da escolha dos processos chave da organização.

Depois de se terem completado estas fases prévias mais ligadas com a gestão de topo, para a introdução prática de processos na organização, propõe-se o seguinte procedimento:

1. Definir os processos: identificar o quadro de processos existentes numa organização e decidir da importância relativa de cada um;
2. Definir os sub-processos e tarefas: cada processo é decomposto em sub-processos (podendo estes também serem decompostos) até se alcançar um nível de detalhe que permita identificar uma actividade composta por várias tarefas;
3. Identificar os interfaces entre processos;
4. Estabelecer os donos e padrinhos dos processos bem como os membros das equipas multidisciplinares;
5. Definir os objectivos dos processos;
6. Definir os indicadores chave;
7. Definir os líderes e as equipas de execução dos processos;
8. Medir e monitorizar os processos;
9. Executar acções correctivas;
10. Rever e melhorar continuamente os processos.

A identificação dos processos chave de uma organização pode ser efectuada com recursos a critérios que ajudem a decidir se se está em presença de um processo chave ou de outro processo menos importante da organização. Alguns desses critérios são:

- Avaliação se o processo atravessa várias funções. Se tal não suceder está-se em presença de uma actividade ou tarefa;
- Verificação se o processo é mensurável e está integrado no plano de negócio. Em caso negativo, deve ser terminado, visto estar desalinhado com a estratégia da organização;
- Verificação da existência dum enfoque nos clientes e nos seus requisitos. Se tal não acontecer pode estar-se em presença de um processo não eficiente;
- Averiguação da importância para a qualidade final do produto;
- Averiguação da importância para o cumprimento da missão da organização;
- Verificação se é pluridisciplinar. Se tal não ocorrer pode estar-se em presença de uma actividade ou tarefa;
- Averiguação da importância para o êxito da organização.

Com base nesta triagem, de entre todos os processos de uma organização são escolhidos os que vão ser alvo de uma gestão por processos e de quais as actividades e tarefas que os compõem.

Existirão certamente processos, actividades, e tarefas que vão ser alvo de finalização visto não acrescentarem directamente valor para os clientes (principalmente no caso dos processos de valor acrescentado), nem para a própria organização, nem indirectamente para os clientes (principalmente no caso dos processos de gestão e dos processos de suporte). Esses mesmos processos, actividades, e tarefas terminados ou redefinidos e as respectivas consequências em termos de reorganização e de impacto nos recursos humanos da empresa, são a essência da reengenharia [Hammer, 1997].

Para além deste enquadramento geral (fig. 5.3), dentro de cada categoria de processos (gestão, suporte, valor acrescentado) existem vários processos que definem com mais detalhe os tipos de actividades existentes nas organizações (fig. 5.4).

Os processos de negócio são desenhados com a intenção de trazer valor para a organização onde ocorrem. Esse valor tem que ser mensurável para servir de base ao rastreio do estado do processo e a eventuais melhorias. Como indicadores para cada processo propõem-se [Marshall, 2000]:

- Produtividade: relação entre o valor das saídas e o custo das entradas;
- Valor Acrescentado: valor das saídas menos o custo das entradas;
- Tempo de Ciclo: tempo decorrido entre a data de início e a data de fim;
- Tamanho da Fila: tamanho médio da fila de pedidos de execução do processo;
- Índice de Qualidade: número de defeitos como uma percentagem das instâncias que já ocorreram do processo.

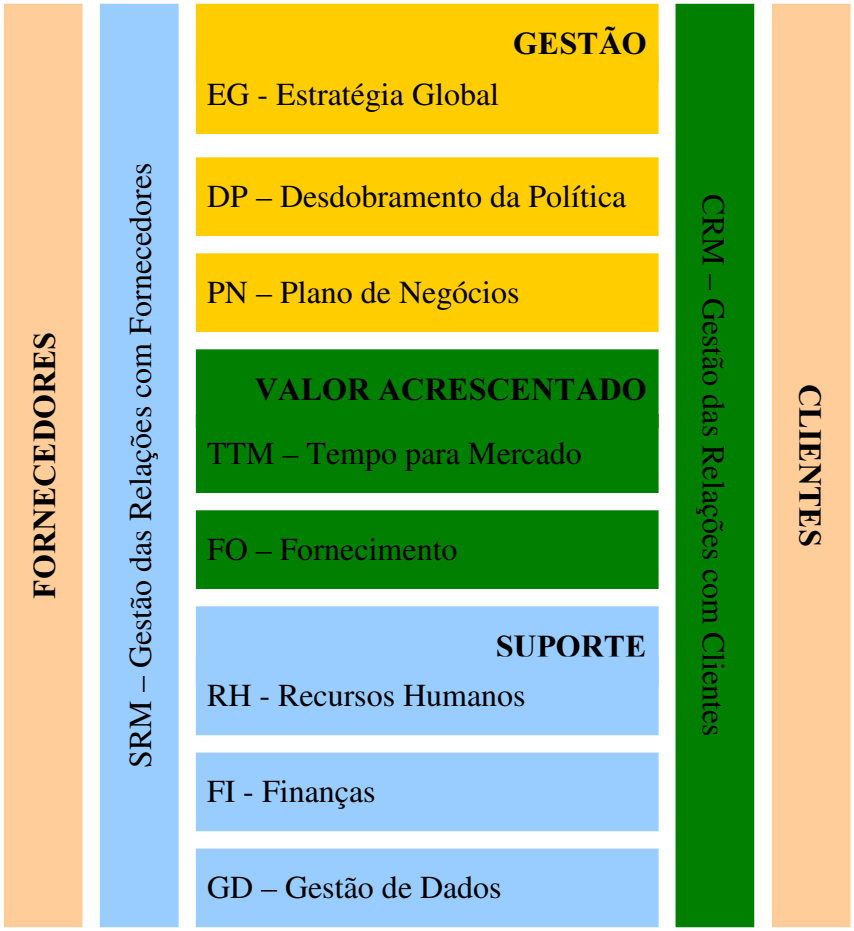

Fig. 5.4 **– Quadro Geral dos Processos duma Organização Genérica.**

Nos processos de gestão (parte superior da fig. 5.4), propõem-se três diferentes processos genéricos:

- EG - Estratégia Global que é o processo de mais alto nível de gestão e onde são avaliados os potenciais clientes, onde é feito planeamento das inovações a introduzir na organização, e que define a política geral da empresa, como sejam a missão, a visão, os princípios, e os valores da organização bem como os objectivos a longo prazo;
- DP - Desdobramento da Política onde a política e os objectivos a curto prazo da organização, normalmente anuais, são desdobrados e entregues os sectores e processos;
- PN - Plano de Negócios onde são feitos, normalmente numa base anual, o planeamento e a orçamentação das actividades e a sua distribuição pelos processos e centros de competência que a organização executará no ano seguinte, bem como as actividades de controlo da execução do plano.

Na categoria dos processos de valor acrescentado (parte central da fig. 5.4), existem os seguintes três processos:

- TTM[1] - Tempo para Mercado que é todo o processo de desenvolvimento de um novo produto. Inclui actividades como a cotação e estabelecimento de novas peças (no caso de produtos), o desenvolvimento, a criação de protótipos, a gestão da qualidade para o novo produto ou serviço. Normalmente este processo ocorre entre a descoberta de uma oportunidade de negócio no mercado, ou então pela criação de um produto inovador cumprindo os objectivos estratégicos, e entre o seu completo desenvolvimento, ou seja com o produto ou serviço apto para passar para o processo de fornecimento;
- FO - Fornecimento que inclui entre outras as actividades de aprovisionamento, gestão de stocks, produção, entregas aos clientes. Este processo normalmente ocorre após o TTM para um mesmo produto ou serviço, e normalmente começa com uma encomenda do cliente, com uma decisão de produção ou de fornecimento de um serviço, e termina com a entrega ao cliente desse produto ou serviço com a qualidade e prazos requeridos;
- CRM[2] - Gestão das Relações com os Clientes inclui actividades como marketing, serviço pós-venda, equipas conjuntas com membros da organização e do cliente (normalmente no caso de os clientes serem outras organizações), gestão das contas dos clientes, e de um modo geral todas as actividades que permitam que os clientes estejam satisfeitos com o serviço ou produto que a organização lhes fornece.

Na categoria de processos de suporte (parte inferior da fig. 5.4), são propostos quatro processos:

- SRM[3] - Gestão das Relações com os Fornecedores que inclui actividades com a escolha de fornecedores para cada material necessário aos produtos ou serviços da organização, a execução de auditorias para verificação da qualidade, o estabelecimento de parcerias para fornecimentos just-in-time;
- RH - Recursos Humanos onde se incluem as actividades de contratação de pessoas, gestão dos salários, e genericamente todas as actividades relativas ao bem-estar dos colaboradores da empresa bem como à sua qualificação;
- FI - Finanças contendo todas as actividades relativas ao ambiente fiscal onde a organização se insere;
- GD - Gestão de Dados onde se incluem as actividades relativas à garantia da qualidade dos dados e indicadores da organização, a segurança dos dados, a definição de prioridades dos projectos informáticos, o controlo do orçamento informático, ou as actividades de datawarehousing e datamining.

A ênfase e a prioridade, bem como as actividades concretas de cada um destes processos, dependem em primeiro lugar do tipo de organização onde vão ocorrer, e em seguida dos objectivos estratégicos da gestão de topo.

As actividades de melhoria contínua, de gestão da qualidade e ambiente (incluindo processos de certificação) são exemplos de actividades que muitas vezes tendem a ser confundidas com processos. Estas actividades fazem parte integrante dos

[1] Time To Market.
[2] Customer Relationship Management .
[3] Supplier Relationship Management.

processos, como os da fig. 5.4, e não devem ter uma gestão separada dos processos de gestão, valor acrescentado, ou de suporte.

5.2.2 Modelo para uma Organização que Desenvolve Software

O modelo de processos apresentado (fig. 5.4) podendo modelar uma organização genérica, com o intuito de modelar as organizações alvo onde correm as aplicações de software, serve também como base para uma proposta de modelação de uma organização que desenvolve software.

Como é conhecido o tipo de organização a modelar, neste caso as organizações que desenvolvem software, é possível detalhar melhor e propor tipos de processo já existentes para os processos genérico apresentados (fig. 5.5).

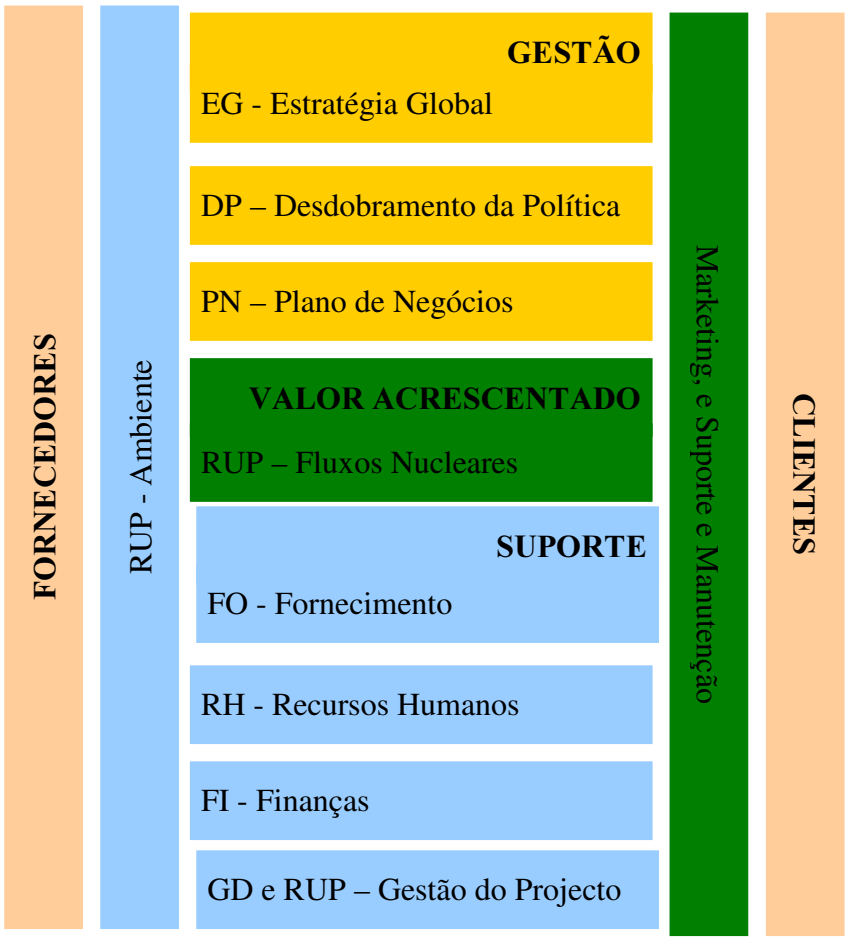

Fig. 5.5 – **Quadro Geral dos Processos duma Organização que Desenvolve Software.**

Faz-se de seguida uma análise das especificidades da fig. 5.5, relacionando-a com a fig. 5.4 que está na sua origem, evidenciando os aspectos que as diferenciam. Assim, nas organizações que desenvolvem software, propõe-se que os processos de negócio tenham o seguinte significado:

- Qualquer tipo de organização necessita de processos de gestão. Desta forma, os processos de Estratégia Global, Desdobramento da Política, e Plano de Negócios são equivalentes aos de outras organizações, embora se deva levar em conta as especificidades do mercado de desenvolvimento de software, como sejam a rápida mudança nas tecnologias, ou a competição em mercados alargados;

- Como o software é um bem intangível, obviamente que não são necessárias matérias primas tangíveis para lhe dar origem. Como contra exemplo, numa organização que produza carros, o processo SRM encarregar-se-ia da compra de pneus, estofos, etc. O processo de Gestão das Relações com Fornecedores instancia-se aqui na forma do fluxo de trabalho Ambiente do RUP, visto este fornecer o ambiente de trabalho (e.g. ferramentas de desenvolvimento) para as equipas e o conjunto de linhas de orientação para desenvolvimento a cumprir pelas equipas dentro da organização;

- O processo de Fornecimento resume-se a cópias de uma aplicação. Devido também ao carácter do software (ao contrário dum produto duma indústria tradicional, como por exemplo a têxtil, onde é talvez o processo mais importante) este operação é de fácil execução. Normalmente o tipo de materiais de suporte (e.g. CD-ROM ou DVD com cópias da aplicação, manuais impressos) são realizados com recurso a empresas externas especialistas. A realização externa deste processo deve-se a ele não acrescentar valor a uma organização que desenvolve software. Assim, numa organização que desenvolve software o processo de Fornecimento não é um processo de valor acrescentado mas sim um processo de suporte;

- O processo Tempo para Mercado toma a forma do conjuntos de fluxos de trabalho nucleares do RUP: Modelação do Negócio, Requisitos, Análise e Desenho, Implementação, Teste, e Entrega. Este conjunto de actividades, ou sub-processos, como facilmente se verifica no RUP, ocorrem em paralelo para um mesmo projecto de desenvolvimento [RUP, 2001]. Este é o processo mais importante numa organização que desenvolve software;

- O processo de Recursos Humanos neste tipo de organizações é o mesmo de outro tipo de organizações, com a ressalva que o mercado de software é um mercado de mão de obra muito especializada, sendo o recrutamento um processo crítico para o sucesso de futuros projectos na organização;

- Também o processo Finanças é o processo comum de cumprimento de obrigações fiscais;

- O processo Gestão das Relações com Clientes fica composto por dois sub-processos: Marketing, e Manutenção e Suporte. Fica deste modo assegurado que o ciclo de vida de uma aplicação de software não termina com a sua entrega aos clientes finais mas sim continua com este processo, incorporando mudanças e correcções, até essa aplicação deixar de ser usada pelo cliente final. O Marketing assume, neste particular, uma forma semelhante à observada noutro tipo de organizações;

- O processo de Gestão de Dados, nos dados relativos aos fluxos de trabalho nucleares do RUP, assume a forma do fluxo de trabalho Gestão do Projecto do RUP. Este processo mantém todas as actividades relativas a todos os outros processos. Neste fluxo existem actividades que conduzem à produção de indicadores do estado do projecto. A sua existência é a base para a tomada de decisões baseada em factos, relativas tanto ao andamento do projecto como para ajuste e melhoria do processo de desenvolvimento de software.

A melhoria contínua dos processos de desenvolvimento, necessária à obtenção de níveis elevados no modelo CMM, deve fazer parte de cada um dos processos e não existir como um processo autónomo.

Os fluxos de trabalho nucleares do RUP, que na fig. 5.5 implementam o processo de valor acrescentado, estão subdivididos em actividades, podendo estas ser encaradas com sub-processos. A descrição destes é feita com recurso a diagramas de actividades, de que se apresenta um exemplo na fig. 5.6, podendo ser complementada com outro tipo de diagramas, como os de interacção (sequência e colaboração), ou de objectos de negócio (incluindo trabalhadores e entidades de negócio) dependendo do sub-processo a modelar. Esta representação é também válida para todos os outros processos duma organização genérica, como a apontada na fig. 5.4.

Quando uma organização que desenvolve software, como a mostrada na fig. 5.5, executa um processo de desenvolvimento de software, o processo Tempo para Mercado, entre outros, vai ser executado. Como é proposto que este processo seja implementado pelos seis fluxos de trabalho nucleares do RUP, vai ocorrer também uma execução do fluxo de trabalho Modelação do Negócio. Como este fluxo de trabalho sugere que a modelação seja feita recorrendo a diagramas de actividades (entre outros tipos de diagramas), surge como resultado que uma organização alvo, como a proposta na fig. 5.4, seja modelada com um conjunto desses diagramas.

Paralelamente, na organização que desenvolve software, o fluxo de trabalho Modelação do Negócio do RUP é um sub-processo do processo TTM, pelo que a modelação desse fluxo é também ela descrita por um diagrama de actividades, que é apresentado na fig. 5.6.

5.3 Modelação dos Processos de Negócio

Dos seis fluxos de trabalho nucleares do RUP (Modelação do Negócio, Requisitos, Análise e Desenho, Implementação, Teste, e Entrega) faz-se aqui uma descrição apenas do fluxo Modelação do Negócio, por ser aquele que está mais directamente relacionado com o entendimento das necessidades do cliente, e por influenciar directamente os fluxos que se lhe seguem. Durante o processo de desenvolvimento de produtos de software surge frequentemente a necessidade de uma percepção comum (às equipas de desenvolvimento, aos clientes e a todos os outros interessados) dos processos de negócio que ocorrem na organização alvo.

Esta realidade não se limita aos óbvios sistemas de informação organizacionais, pois mesmo em produtos que incluem sistemas embebidos de tempo real pode surgir a

necessidade da sua modelação devido à integração de funcionalidades desse tipo de sistemas com processos de negócio.

Se o desenvolvimento de aplicações, seja num modelo feito à medida da organização alvo, seja num modelo de aplicações comerciais previamente disponíveis, não tiver em conta os processos de negócio existentes (ou que se pretendam implementar) numa organização, pode significar o risco da aplicação final desenvolvida não ter o sucesso pretendido. Este insucesso pode, por exemplo, ser devido aos utilizadores finais não utilizarem correctamente a aplicação por esta não modelar e suportar correctamente as actividades sob sua responsabilidade.

No RUP, a modelação de processos de negócios é feita no fluxo de trabalho "Modelação do Negócio" (fig. 5.6).

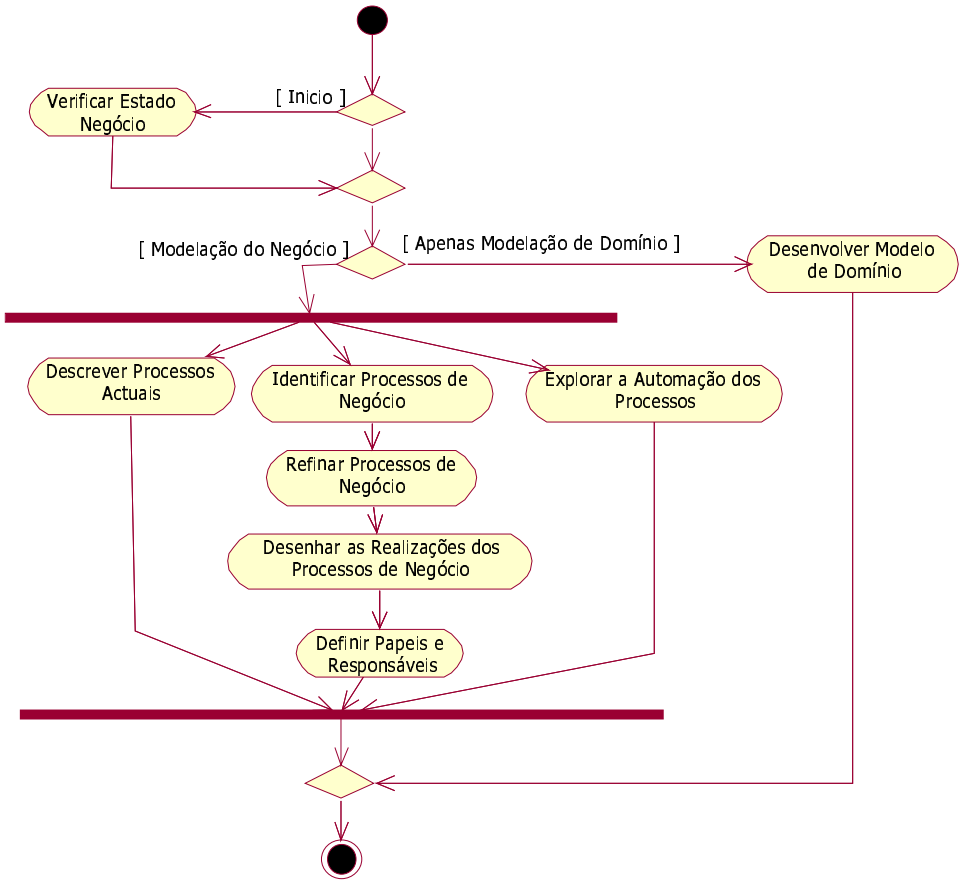

Fig. 5.6 – **Diagrama de Actividades na Modelação de Negócio no RUP.**

As actividades principais constantes neste fluxo de trabalho estão centradas na identificação, refinamento, e realizações dos processos de negócio, e na definição de papéis e responsáveis, nomeadamente trabalhadores de negócio e unidades organizacionais, intervenientes no negócio.

Cada papel do RUP (e.g. o agente que executa alguma actividade na organização que desenvolve software) existente neste fluxo de trabalho tem sob sua responsabilidade a execução de várias actividades (tabela 5.1) que vão ter como resultado artefactos (tabela 5.2).

Tabela 5.1 **– Papéis e Actividades na Modelação do Negócio do RUP.**

Papel	Actividade
Analista de Processos de Negócio	- Verificar Organização Alvo; - Estabelecer e Ajustar Objectivos; - Capturar um Vocabulário de Negócio Comum; - Descobrir Actores de Negócio e Casos de Uso de Negócio; - Manter as Regras de Negócio; - Estruturar o Modelo de Casos de Uso de Negócio; - Definir a Arquitectura do Negócio.
Revisor do Modelo de Negócio	- Rever o Modelo de Casos de Uso de Negócio; - Rever o Modelo de Objectos de Negócio.
Desenhador do Negócio	- Detalhar Casos de Uso de Negócio; - Descobrir Trabalhadores de Negócio e Entidades de Negócio; - Definir os Requisitos de Automação; - Detalhar Entidades de Negócio; - Detalhar Trabalhadores de Negócio.

De entre todas as actividades e os seus consequentes artefactos, existem alguns destes com carácter mais obrigatório e outros com carácter mais opcional. Esta flexibilidade permite a configuração do RUP de modo a adequar-se a uma organização específica e ainda a um projecto específico.

Tabela 5.2 **– Papéis e Artefactos na Modelação do Negócio do RUP.**

Papel	Artefacto
Analista de Processos de Negócio	- Regras de Negócio; - Modelo de Casos de Uso de Negócio; - Glossário de Negócio; - Modelo de Objectos de Negócio; - Visão do Negócio; - Especificação Suplementar do Negócio; - Verificação da Organização Alvo; - Documento da Arquitectura do Negócio.
Desenhador do Negócio	- Unidades Organizacionais.

Deste modo, e salvaguardando-se as opções feitas nas escolhas dos artefactos, este conjunto de artefactos permite fazer a modelação dos processos de negócio, por possuírem as seguintes características [RUP, 2001]:

Visão do Negócio

Este artefacto captura os objectivos que o esforço de modelação do novo sistema pretende alcançar. Deve aqui estar explícito "o porquê" e "o que", relacionados com o projecto. A visão do negócio serve de barómetro para todas as decisões de alto nível futuras no projecto.

Glossário de Negócio

O objectivo principal deste artefacto é a captura de termos de negócio necessários ao bom entendimento entre os interessados no negócio. Paralelamente, nos casos em que a equipa de desenvolvimento não conhece os termos de negócio usados estes artefacto serve de base para o entendimento do negócio.

Regras de Negócio

As regras de negócio correspondem a declarações de políticas e de condições que devem ser satisfeitas do ponto de vista do negócio. São semelhantes a requisitos do sistema mas têm o âmbito do negócio, podendo expressar regras e regulamentos subjacentes ao negócio como também a sua arquitectura e estilo. A sua modelação deverá ter um carácter o mais rigoroso possível, podendo ser expressa através do uso de Object Constraint Language (OCL) como especificada no UML [OMG, 2001a], ou através de uma forma de linguagem natural estruturada. Apesar das vantagens de se usar uma abordagem mais formal como o OCL, tem que ser pesado se esta forma de descrição se adequa ao bom entendimento por todos os interessados no sistema.

As regras de negócio no caso actual são modeladas com o recurso à linguagem natural estruturada, com os seguintes construtores [Odell, 1998]:

- SE;
- APENAS SE;
- QUANDO;
- ENTÃO;
- SENÃO;
- DEVE SEMPRE VERIFICAR-SE QUE;
- FOI COMPLETADO CORRECTAMENTE.

Uma possível catalogação para as regras de negócio, pode ser a sua divisão em regras de restrição e regras de derivação:

- **Regras de Restrição**: especificam políticas e condições que restringem a estrutura e comportamento dos objectos de negócio. Estas regras podem ainda ser subdivididas em:
  - Regras de estímulo e resposta: restringem o comportamento especificando “se” e “quando” as condições devem ser verdadeiras de modo ao comportamento ser iniciado;
  - Regras de restrição das operações: especificam as condições que devem ser verdadeiras antes e depois duma operação para assegurar que uma operação é feita correctamente;
  - Regras de restrição da estrutura: especificam políticas ou condições acerca de classes, objectos e as suas relações que não devem ser violadas.
- **Regras de Derivação**: especificam políticas para inferência de factos computacionais a partir de outros factos. Estas regras podem ser subdivididas em:
  - Regras de inferência: especificam que quando algum facto é verdadeiro, uma conclusão pode ser inferida;
  - Regras de Computação: derivam os seus resultados pelo processamento de algoritmos.

Modelo de Casos de Uso de Negócio

O propósito principal deste artefacto é mostrar como o negócio é usado pelos seus clientes e parceiros. Aqui são modelados e descritos os processos de negócio e as suas interacções com parceiros externos.

A modelação é feita, entre outros objectos UML usados em diagramas de casos de uso, recorrendo a estereótipos para os casos de uso de negócio e para os actores de negócio.

Normalmente os processos de negócio têm subjacentes um fluxo de trabalho que necessita de ser modelado e que mostra o que o negócio deve fornecer em valor acrescentado para os actores. Esta modelação pode ser feita com recurso a diagramas de actividades, que podem ser estendidos pela representação das unidades organizacionais intervenientes no processo de negócio e pela distribuição das actividades pelas respectivas unidades organizacionais. Deste modo consegue-se perceber mais facilmente "quem faz o quê".

Modelo de Objectos de Negócio

Este artefacto é um modelo de objectos que descreve a realização de casos de uso de negócio. Serve de base ao entendimento de como os trabalhadores de negócio e as entidades de negócio se relacionam e com quem necessitam de colaborar para executar os processos de negócio. Os elementos essenciais deste modelo são:

- Trabalhadores de negócio, que demonstram as responsabilidades que as pessoas podem ter atribuídas e que representam os papéis dos colaboradores da organização;
- Entidades de negócio, que representam recursos, eventos, bens tangíveis que são usados ou produzidos, ou seja, as "coisas" com que os colaboradores interagem;
- Realizações de casos de uso de negócio, que devem demonstrar como os trabalhadores de negócio e as entidades de negócio colaboram para cumprir um fluxo de trabalho da organização. Estas realizações podem ser documentadas com o recurso a:
    - Diagramas de classes, onde identificam estaticamente os trabalhadores de negócio e entidades de negócio;
    - Diagramas de actividades, onde se descreve o fluxo de tarefas para cumprir pelas entidades de negócio e as responsabilidades de as executar (recorrendo a partições verticais por trabalhadores de negócio);
    - Diagramas de interacção (sequência ou colaboração), para verificação dos detalhes da relação entre os trabalhadores de negócio, actores de negócio, e como as entidades são acedidas durante a execução dum caso de uso de negócio. Os diagramas de sequências deverão ser usados para modelar situações complexas e para explicitar sequências de eventos numa ordem cronológica. Os diagramas de colaboração deverão ser usados para mostrar as mensagens e comunicações entre objectos.

Neste modelo, a primeira escolha para documentar a realização de um caso de uso de negócio deve ser um ou mais diagramas de actividades [RUP, 2001] visto serem adequados para representar fluxos de trabalho. A realização dos casos de uso pode ser complementada com diagramas de sequência e de colaboração

Especificação Suplementar do Negócio

Este artefacto apresenta as definições de negócio que não estejam contidas no modelo de casos de uso de negócio ou no modelo de objectos de negócio.

Verificação da Organização Alvo

Aqui é descrito o estado actual da organização alvo. Esta descrição inclui processos actuais, ferramentas existentes, competências e atitude das pessoas, clientes, concorrentes, desafios técnicos, problemas e áreas de melhoria. Este artefacto pode ser usado para explicar aos interessados a necessidade de alterações aos processos de negócio.

Documento da Arquitectura do Negócio

Este artefacto fornece uma visão detalhada em termos de estrutura e propósito do negócio, recorrendo a várias vistas arquitecturais. Normalmente assume uma forma gráfica informal ou recorre a *storyboards*.

Unidades Organizacionais

O recurso à representação de unidades organizacionais (recorrendo a um estereótipo UML) permite estruturar o modelo de objectos de negócio dividindo-o em partes mais pequenas. Uma unidade organizacional é uma colecção de trabalhadores de negócio, entidades de negócio, relações, realizações de casos de uso de negócio, diagramas, e outras unidades organizacionais.

No RUP, o conjunto dos modelos de negócio apresentados vai servir como entrada para outros modelos gerados noutros fluxos de trabalho, como sejam a vista de casos de uso do sistema e a vista lógica apresentada no modelo de análise.

Para além da abordagem proposta pelo RUP para a modelação de processos de negócio e a respectiva transformação em sistemas, nota-se que UML é cada vez mais visto como a linguagem para suportar essa modelação. Para além das ferramentas que suportam o RUP, como o Rational Rose®, um exemplo dessa aceitação crescente é a ferramenta ARIS® [IDS, 2002] que suporta UML, rastreando os seus modelos e diagramas até ao nível da implementação (i.e. transacções em SAP R/3), ou convertendo esses diagramas UML em código middleware[1] recorrendo à arquitectura MDA[2] [OMG, 2001b].

A modelação de processos de negócio e a sua implementação têm muitas vezes como alvo o uso de software standard, como os ERPs. Este tipo de software tem vantagens relativamente ao uso de software desenvolvido à medida [Kirchmer, 1998] como sejam a fiabilidade da aplicação, a sua flexibilidade de adaptação a mudanças nos processos de negócio, ou a qualidade do suporte oferecido pela organização que fornece o ERP.

A tentação do desenvolvimento de aplicações à medida em sistemas relacionados com o negócio pode ter consequências perigosas para o sucesso da organização. Para além do tempo e esforço necessários para as desenvolver, a qualidade do produto final pode ser sempre posta em causa: a aplicação vai modelar os requisitos de uma organização específica. Deste modo, o modelo de referência de processos de

[1] Por exemplo: CORBA IDL, XMI, ou Microsoft .NET®.
[2] MDA = Model Driven Architecture.

negócio constante em aplicações de software standard não vai ser aproveitado pela organização para a criação de vantagens competitivas de um modo rápido. Para além disso, os processos modelados e implementados vão ser os processos desejados pela organização alvo, o que não se traduz em processos refinados e de qualidade elevada.

Aquando da implementação de processos de negócio, ao conjunto de processos disponibilizado num ERP sugere-se a aplicação das seguintes operações:

- Modelação com recurso às técnicas constantes no fluxo de trabalho "Modelação de Negócio" do RUP do estado actual da organização e do estado futuro pretendido;
- Comparação com os processos oferecidos pela aplicação de software standard;
- Renegociação com os clientes dos requisitos não disponíveis. A alternativa seria a extensão das funcionalidades oferecidas pela aplicação de software standard, mas esta opção pode conduzir a problemas graves de migração nas previsíveis actualizações do sistema;
- Parametrização da aplicação de modo a satisfazer as necessidades mais específicas dos clientes. Esta actividade significa normalmente a "escolha" de entre um conjunto de opções disponibilizado pela aplicação de uma alternativa adequada à organização, ou pela decisão da não utilização de funcionalidades[1], e não pela inclusão de novas funcionalidades.

Actualmente o conhecimento dos processos de qualidade internacional que implementam o modelo apresentado na fig. 5.4 está disponível e implementado em soluções de software standard, ou em arquitecturas de middleware. Mesmo a sua distinção por mercados verticais (e.g. indústria química de processos, ou a banca) está prevista e parametrizada nesses mesmos ERPs e middleware.

Quanto maior for a dimensão de uma organização e quanto maior for o nível de detalhe com que se modelam os processos, maior é a necessidade de se recorrer a abordagens estruturadas e de fácil entendimento, por todos os interessados no negócio e nas aplicações que o suportam.

[1] Isto deve-se ao facto de que o conjunto das funcionalidades oferecidas por uma solução de software standard é normalmente mais abrangente que as necessidades de uma organização específica.

6. Casos de Estudo

Neste capítulo são apresentados dois casos de estudo. O primeiro, "Gestão de Viagens", permite validar a relevância da utilização de metodologias de captura de requisitos e de modelação facilmente perceptíveis pelos clientes, bem como do desafio da arquitectura alvo ser constituída por software standard. No segundo, "Salário Prémio", o processo de desenvolvimento escolhido é usado de uma forma mais intensa. É avaliada a capacidade deste processo lidar com realidades organizacionais complexas: a aplicação final pedida pressupõe a reengenharia completa de alguns processos de negócio (bem como da aplicação já existente que os suporta), a extensão de capacidades em mais dois processos de negócio, e finalmente o desenho e implementação de raiz de um novo processo de negócio e da sua aplicação de suporte.

6.1 Introdução

A existência dum ambiente organizacional onde as ideias possam ser postas em prática, permite uma melhor aprendizagem dos processos propostos, quer na percepção da validade dos conceitos propostos quer na percepção das dificuldades reais na execução de um projecto. Este conhecimento permite não só validar as ideias, mas também descobrir os pontos críticos do processo de desenvolvimento de software, assimilando lições, e apontando áreas de melhoria.

A organização alvo em ambos casos de estudo é a mesma: Blaupunkt Auto-Rádio Portugal, Lda, sediada em Braga. Esta organização emprega cerca de 2300 funcionários e dedica-se à produção de auto-rádios fazendo parte da multinacional Blaupunkt sediada em Hildesheim na Alemanha. A empresa de Braga controla a MotoMeter Portuguesa, Lda, sediada em Vila Real, que é uma organização com cerca de 170 colaboradores e que funciona como uma extensão das capacidades de produção de Braga, embora o tipo de produtos que manufactura são distintos: antenas e CD Changers. A Blaupunkt é a divisão dedicada às funcionalidades multimedia nos automóveis da multinacional Robert Bosch.

A Blaupunkt tem uma organização mista, com departamentos hierárquicos mas também com orientação a processos de negócio, tanto internos como externos (fig. 5.1.b). Os processos de negócio modelados no primeiro caso de estudo abrangem apenas os departamentos de contabilidade, recursos humanos, e o departamento relativo ao requerente de uma viagem. No segundo caso de estudo são abrangidos os departamentos de produção, contabilidade, engenharia industrial, recursos humanos e garantia da qualidade.

A organização que desenvolve as aplicações é a mesma nos dois casos de estudo: o departamento de informática local da empresa de Braga. Os departamentos locais de informática nas organizações Bosch apesar de estarem inseridos na empresa local e da maior parte das suas responsabilidades serem de âmbito local, estão também inseridos hierarquicamente numa organização de informática global. Esta situação particular faz com que as organizações que desenvolvem software neste contexto conheçam perfeitamente a realidade da organização local mas exibam também características de organizações externas à organização alvo.

6.2 O Caso de Estudo "Gestão de Viagens"

6.2.1 Abordagem do Problema

A necessidade e motivação para a implementação de um sistema informático de suporte às actividades de Gestão de Viagens, principalmente o Planeamento das Viagens e Liquidação das Despesas de Viagem, deve-se ao facto de estes serem processos sem nenhum suporte informático[1] e que ocorrem muito frequentemente dado a organização alvo ser uma multinacional integrada numa outra. Estes processos de negócio, apesar ocorrerem frequentemente, não são processos críticos, como facilmente se verifica pela análise da fig. 5.4.

Este projecto tinha como principal característica a imposição, por parte da organização alvo, de uma arquitectura final perfeitamente definida: a ferramenta ERP SAP R/3. Esta imposição deriva de decisões estratégicas da organização alvo:

- Não proliferação de plataformas computacionais distintas dentro da mesma organização:
  - Redução da diversidade de conhecimentos técnicos necessários ao suporte e à manutenção;
  - Redução da diversidade de ferramentas que os utilizadores finais necessitam de conhecer;
  - Aumento do grau de especialização, e consequentemente o desempenho, dos colaboradores informáticos e dos utilizadores finais.
- Vantagens económicas obtidas pela negociação de um grande volume de licenças;
- Vantagens no suporte. A organização em vez de ser um pequeno cliente de várias organizações que fornecem software, é um grande cliente de apenas uma organização que fornece software. É claro que esta vantagem tem que ser ponderada considerando o risco de dependência.

Sendo a arquitectura escolhida um ERP, e sendo este implementado e mantido exteriormente à organização alvo e à organização que desenvolveu o software, os aspectos relativos à arquitectura de hardware, suporte à operação, disponibilidade do sistema, entre outros, não são relevantes para o desenvolvimento da aplicação final, visto dependerem apenas de acordos económicos entre a organização onde vai correr a aplicação e a organização que disponibiliza o ERP.

Este projecto foi escolhido, pela organização que desenvolveu o software (departamento de informática da organização alvo) para a introdução de metodologias de modelação gráfica de informação, nomeadamente UML, com as seguintes motivações:

- Eliminar a má qualidade da documentação de suporte comprovada nos projectos antecedentes (descrições textuais incompletas no âmbito e no detalhe);
- Aumentar a facilidade de percepção entre todos os interessados;
- Aumentar o rigor da informação documentada;

[1] Excluindo alguns *templates* em Microsoft Excel…

- Conseguir aumentar a qualidade do produto final, sendo esta entendida como o grau de cumprimento dos requisitos dos clientes;
- Facilitar o suporte à aplicação final.

Como processo de desenvolvimento de software foi escolhido o RUP e não o tradicional processo de desenvolvimento de aplicações para SAP, o ASAP [SAP, 2000]. Apesar da evidente adequação do ASAP ao desenvolvimento de projectos com a arquitectura alvo SAP, este processo enferma de alguns aspectos não desejáveis nos processos actuais: a descrição da situação actual (As Is) e a da situação futura (To Be) são baseadas em texto livre, e apresenta dificuldades na gestão da mudança de requisitos. Para além destes factores, na escolha do RUP teve peso a avaliação da sua introdução na organização que desenvolve software.

Este projecto integrou também um estágio de final de licenciatura [Silva, 2001] onde estão detalhados todos os artefactos produzidos pelo processo de desenvolvimento de software, razão pela qual são omitidos aqui.

6.2.2 Análise dos Resultados

Neste secção analisam-se os pontos fracos e pontos fortes da parametrização do processo de desenvolvimento de software específico para o desenvolvimento deste produto de software.

Assim, tendo em conta a forma como o RUP e UML foram utilizados, os principais pontos positivos que foram identificados na execução deste projecto, são os seguintes:

- Obrigatoriedade da avaliação dos riscos e respectivas medidas de correcção e contenção. Esta actividade permitiu que existissem menos "surpresas" durante a execução do projecto e que quando elas ocorreram estivessem já planeadas acções para as corrigir e conter;

- Limitação do esforço de desenho e implementação da arquitectura da solução final, visto esta já estar disponível e documentada;

- A introdução do RUP numa organização com um nível de maturidade do seu processo de desenvolvimento de software comparável no máximo ao CMM Nível 3, obriga a uma sistematização e consciencialização que o desenvolvimento de software não é apenas escrever código, mas que existem muitas outras actividades com uma importância muito grande na qualidade do produto final;

- A qualidade da documentação melhorou em quantidade, detalhe, e rigor, formando assim o exemplo para o resto da organização e permitindo também uma melhor manutenção e suporte;

- As reuniões mantidas com todos os interessados melhoraram na relação entre o tempo gasto e os resultados obtidos (normalmente definição de requisitos) devido ao tipo de interface utilizado: uma linguagem de modelação gráfica (UML);

- Os requisitos dos clientes foram implementados de forma satisfatória.

Como principais pontos negativos foram identificados:

- Dificuldade no mapeamento dos requisitos nas funcionalidades já existentes numa solução de software standard;

- Necessidade de um conhecimento profundo do software standard;

- À introdução do RUP, como em qualquer novo processo numa organização, resistem alguns elementos da equipa de desenvolvimento.

- A falta de análise do impacto económico da aplicação, ou seja do seu retorno do investimento, fazem com que os clientes quando a aplicação está pronta, tenham tendência para avaliar apenas os custos. Este facto é mais relevante quando a arquitectura alvo é o do tipo software standard;

- As Verificações de Qualidade[1] feitas de modo informal e sem a participação de todos os interessados podem conduzir a conclusões imprecisas sobre o estado do projecto;

- A não utilização conveniente do fluxo "Modelação do Negócio". A descrição da situação actual foi descrita em texto (pegando num regulamento interno da organização alvo), e a situação futura foi descrita apenas com casos de uso e suas realizações relativas ao sistema final e não do negócio.

E como principais lições assimiladas após a execução deste projecto, constatou-se que:

- A existência de soluções standard de software (como os ERPs) obriga a uma fase extra de negociação de conflitos entre os requisitos dos clientes e as funcionalidades oferecidas pelo ERP. Apesar de ser normal a possibilidade de extensão ou de modificação do ERP esta não é desejável, pois na próxima versão do ERP estas funcionalidades não serão consideradas correndo-se o risco de um novo desenho e implementação da solução. Esta negociação só pode ser mediada pelos elementos da equipa de desenvolvimento de software, pois os clientes normalmente não têm conhecimento das funcionalidades já existentes no ERP;

- É necessário que a introdução do RUP seja feita num projecto em que os artefactos de mais alto nível[2] (e.g. modelo de casos de uso de negócio) tenham uma transcrição fácil e perceptível para artefactos de mais baixo nível (e.g. código). Sem esta característica corre-se o risco do RUP cair em descrédito tanto nos clientes como na equipa de desenvolvimento, e ser visto como um gerador de artefactos em papel e não um gerador de aplicações;

[1] *Milestones* entre as fases do RUP.
[2] Por mais alto nível entende-se a maior proximidade com os clientes.

- A introdução do RUP não deve ser feita por imposição hierárquica (de cima para baixo), mas antes através de apresentações com explicação das vantagens previstas. Deste modo a equipa na organização que desenvolve software toma muito mais facilmente partido por um comportamento pró-activo e de resolução de problemas;

- Não é viável a introdução do RUP sem um esforço, não desprezável, de formação em UML e no próprio RUP;

- A organização alvo tem uma influência decisiva no desempenho da organização que desenvolve o software. Problemas como a falta de redundância em elementos chaves (impedidos temporariamente de trabalhar) e a falta de decisão na definição dos requisitos por parte de alguns interessados, que são da exclusiva responsabilidade dos clientes, podem ter um impacto negativo importante na imagem da organização que desenvolve software;

- As Verificações de Qualidade devem ser feitas formalmente, documentadas, e contar com a presença de todos os interessados no sistema;

- O fluxo de trabalho "Modelação do Negócio" é de extrema importância e deve ser efectuado cuidadosamente em todos os projectos que envolvam aplicações de negócio;

- A falta de controlo sobre a versão do ERP utilizada, visto este estar sediado numa organização externa, fez com que houvesse uma mudança de características na arquitectura alvo devido à alteração (durante a execução do processo de desenvolvimento) da versão SAP R/3 4.6B para SAP R/3 4.6C, o que obrigou a novo ciclo de aprendizagem das funcionalidades oferecidas pelo software standard.

6.3 O Caso de Estudo "Salário Prémio"

6.3.1 Abordagem do Problema

O processo de pagamento de um prémio em dinheiro no salário mensal de cada colaborador directo[1] é um processo organizacional crítico devido à possibilidade de conflitos sociais por mau cálculo ou impossibilidade de explicação de valores pagos. A intenção de uma organização ao recorrer a este tipo de incentivo é o aumento dos seus valores de produtividade.

Para além da criticidade do processo de negócio, este é também complexo devido à dependência de outros processos. No caso presente, o processo de pagamento de um prémio depende de três factores base: o absentismo individual, a qualidade do tipo de produtos das linhas onde o colaborador trabalhou, e o desempenho individual de produtividade. Aos dois primeiros sub-processos foi necessário proceder à sua extensão para suportarem as funcionalidades requeridas pelo sistema de prémio. Para o terceiro, foi necessário proceder à sua reengenharia completa, visto a qualidade de dados e o desenho do processo existente na organização não serem suficientes para os requisitos do sistema de prémio. Finalmente, para o cálculo e pagamento dos valores dos prémio foi necessário desenhar, modelar, e implementar um novo processo de negócio. Todos estes processos de negócios são suportados por aplicações informáticas.

A escolha do RUP como base do processo de desenvolvimento do software neste projecto deve-se, para além das suas características intrínsecas, ao objectivo da organização que desenvolve software de padronizar o seu processo de desenvolvimento, e de verificar o desempenho do RUP num projecto organizacional complexo.

Sendo o RUP um processo parametrizável e susceptível de se adequar tanto a pequenos como a grandes projectos, serão apresentados neste capítulo os artefactos produzidos. Para essa parametrização ocorrer, é necessária a escolha de um subconjunto de artefactos, de entre o universo completo de artefactos, bem assim como do seu grau de detalhe. Estas escolhas são validadas pelas Verificações de Qualidade. Deste modo, tanto o subconjunto de artefactos usado como o seu grau de detalhe não podem ser antecipadamente previstos com rigor, mas sim escolhidos com base na experiência da equipa de desenvolvimento em relacionar as características de cada projecto específico com as funcionalidades oferecidas pelos artefactos.

Os critérios para concretização dessa escolha estão relacionados com:

- Características do próprio projecto (e.g. criticidade dos processos de negócio modelados, tipo de organização alvo);
- Características da organização que desenvolve o projecto (e.g. tamanho da equipa, grau de conhecimento de regulamentos internos);
- Restrições temporais. Como os recursos em qualquer projecto de engenharia não são ilimitados, é sempre necessário um compromisso entre a quantidade e detalhe dos artefactos produzidos e os prazos para implementação do projecto.

Os artefactos produzidos resultam de um conjunto de actividades que ocorrem dentro desses fluxos e que têm um responsável pelos mesmos (papel na terminologia do RUP). Neste caso de estudo foi ainda identificada a necessidade dos artefactos

[1] Trabalhador *Blue collar*.

representarem duas situação distintas em termos de negócio: parte do projecto representa actividades de reengenharia de alguns processos de negócio (e.g. cálculo da produtividade por linha) , e outra parte representa a introdução de um novo processo de negócio (e.g. pagamento de um prémio no salário de cada colaborador directo).

6.3.2 Descrição

Neste sub-capítulo faz-se uma descrição do projecto baseada no RUP e em UML. Em vários diagramas (e.g. Modelo de Casos de Uso de Negócio) o UML standard é aumentado com o recurso a estereótipos descritos no RUP permitindo assim a criação de artefactos próprios do RUP.

São também usadas técnicas não contidas no RUP, como a classificação de requisitos FURPS+ (secção 2.3) ou o planeamento da equipa de manutenção para o sistema de prémio (recorrendo ao sugerido no secção 2.5 com uma extensão para este projecto específico).

São apresentados os artefactos usados relativos aos fluxos de trabalho Modelação do Negócio e Gestão do Projecto.

O sistema de pagamento de um valor de prémio no salário dos colaboradores, será implementado em primeiro lugar na MotoMeter, em Vila Real, e após essa fase será transposto para a Blaupunkt em Braga.

Modelação do Negócio

Este fluxo de trabalho tem como objectivos principais:

- Compreender a estrutura e a dinâmica da organização onde o sistema vai correr;
- Compreender os problemas actuais da organização alvo e identificar potenciais de melhoria;
- Assegurar que clientes, utilizadores finais, equipa de desenvolvimento têm um entendimento comum da organização alvo;
- Deduzir requisitos do sistema necessários para suportar a organização alvo.

São de seguida apresentadas diversos modelos e tabelas que documentam a forma como este fluxo foi seguido e parametrizado no caso de estudo. Este procedimento foi igualmente adoptado nas descrição dos restantes fluxos de trabalho que o RUP contém.

Tabela 6.1 – **Artefactos na Modelação do Negócio.**

Artefacto	Utilizado	Não Utilizado	Motivo	Papel Responsável
Regras de Negócio	X		Rigor na descrição de um novo processo de negócio	Analista de Processos de Negócio
Modelo de Casos de Uso de Negócio	X		Primeira descrição das funcionalidades e dos actores de negócio da organização	Analista de Processos de Negócio
Glossário de Negócio		X	Os termos de negócio são comuns à organização alvo e a organização que desenvolve software (são duas suborganizações da mesma organização)	Analista de Processos de Negócio
Modelo de Objectos de Negócio	X		Realização dos casos de uso de negócio	Analista de Processos de Negócio
Visão do Negócio		X	A visão de negócio é comum e faz parte dos objectivos da organização alvo e da organização que desenvolve software	Analista de Processos de Negócio
Especificação Suplementar do Negócio		X	Modelo de Casos de Uso de Negócio e Modelo de Objectos de Negócio são suficientes	Analista de Processos de Negócio
Verificação da Organização Alvo		X	A organização alvo é perfeitamente conhecida da equipa de desenvolvimento de software. A modelação dos processos actuais é feita no Modelo de Casos de Uso de Negócio – Situação Actual	Analista de Processos de Negócio
Documento da Arquitectura do Negócio		X	Detalhe contido no Modelo de Casos de Uso de Negócio e no Modelo de Objectos de Negócio é suficiente	Analista de Processos de Negócio
Unidades Organizacionais	X		Mapeamento das funcionalidades do processo de negócio na estrutura da organização alvo	Desenhador do Negócio

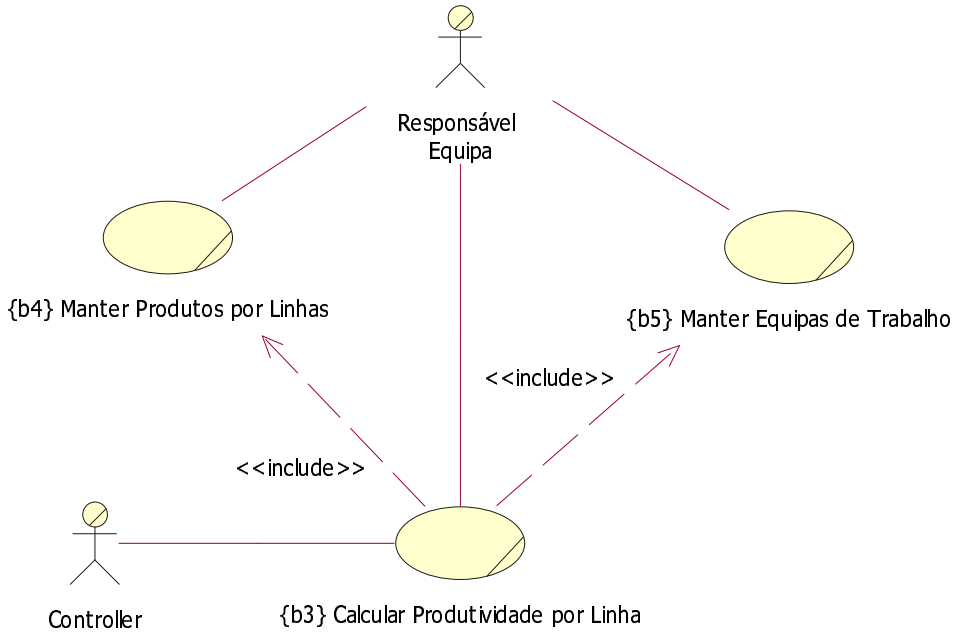

Fig. 6.1 – **Modelo de Casos de Uso Negócio – Situação Actual.**

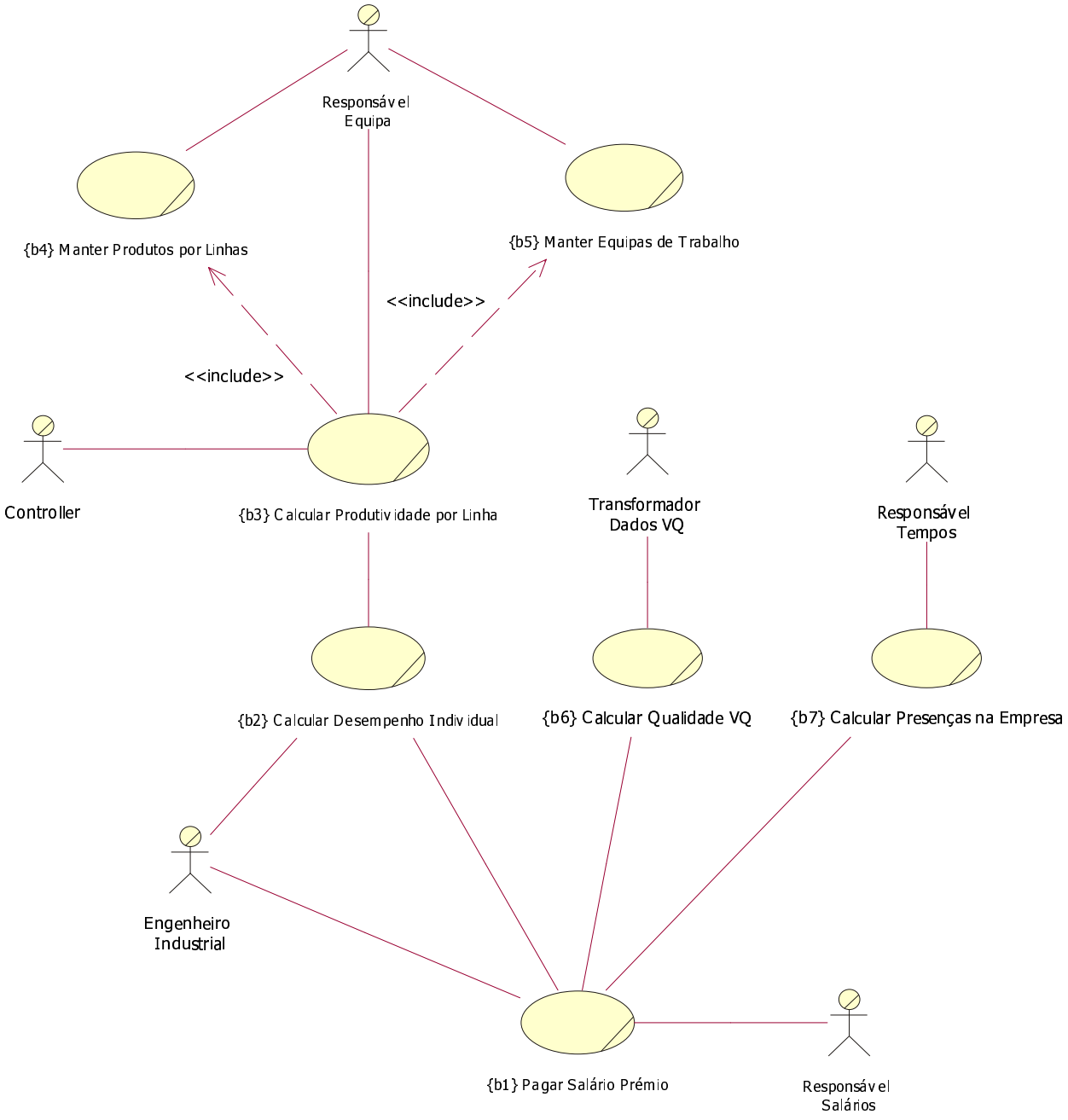

Fig. 6.2 – **Modelo de Casos de Uso Negócio – Situação Futura.**

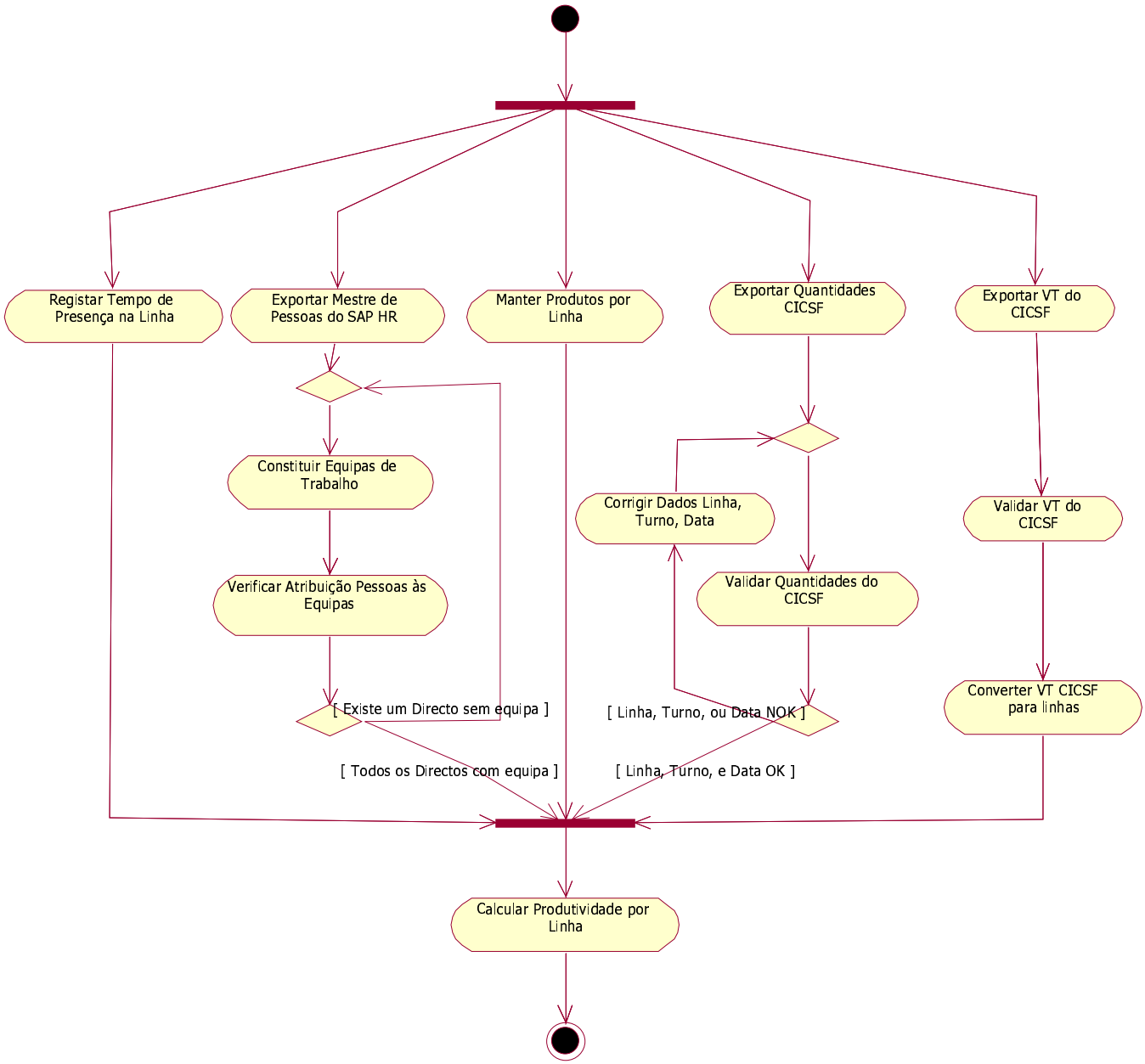

Fig. 6.3 – **Modelo de Casos de Uso de Negócio – Diagrama de Actividades de Produtividade por Linha.**

Fig. 6.4 – **Unidades Organizacionais.**

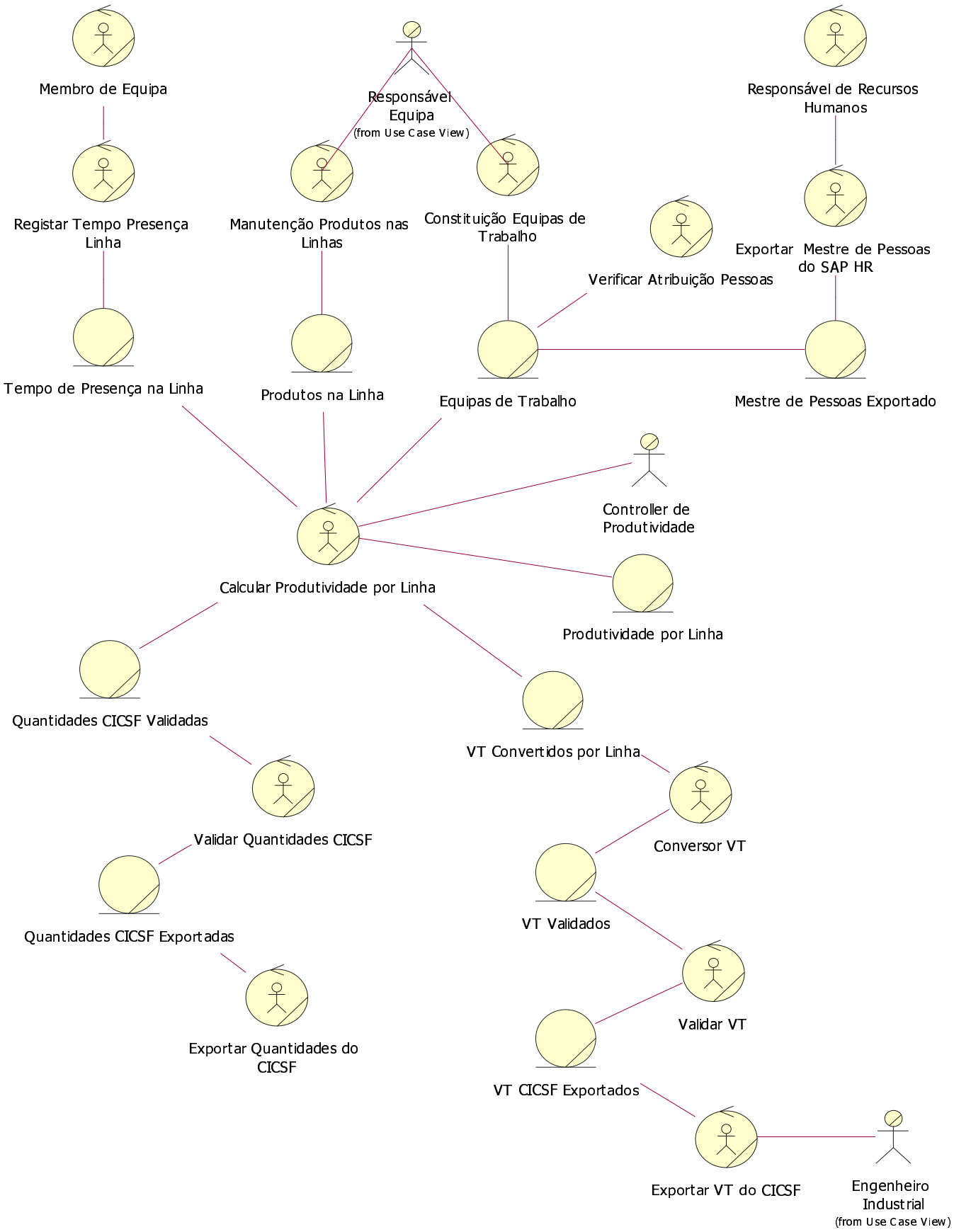

Fig. 6.5 – **Modelo de Objectos de Negócio – Produtividade por Linha.**

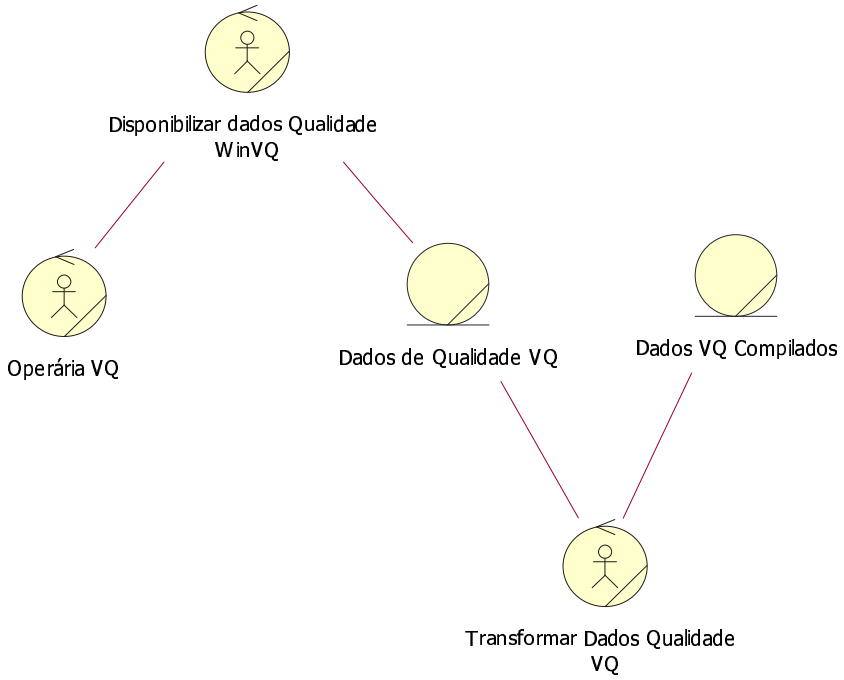

Fig. 6.6 – **Modelo de Objectos de Negócio – Qualidade VQ.**

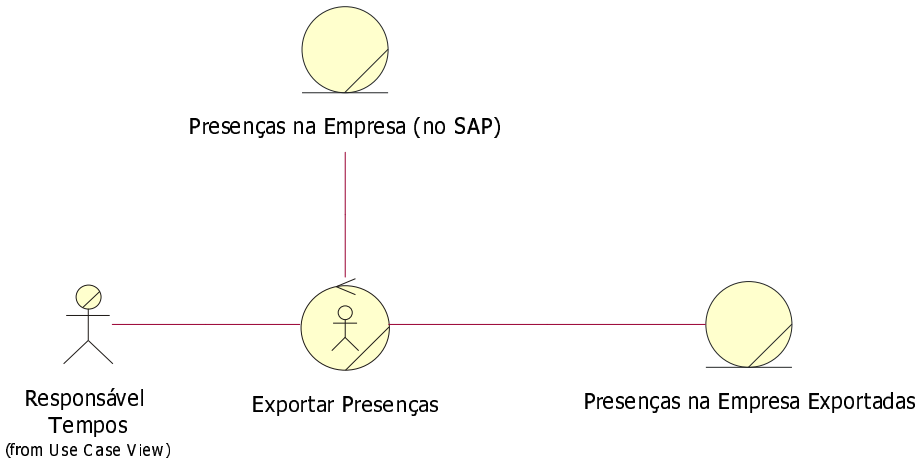

Fig. 6.7 – **Modelo de Objectos de Negócio – Presenças na Empresa.**

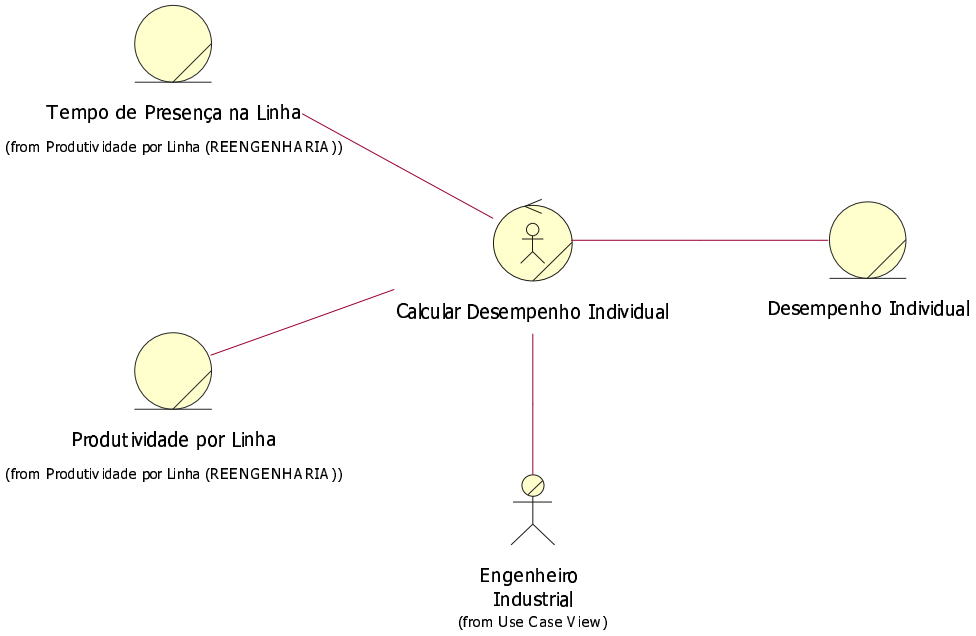

Fig. 6.8 – **Modelo de Objectos de Negócio – Desempenho Individual.**

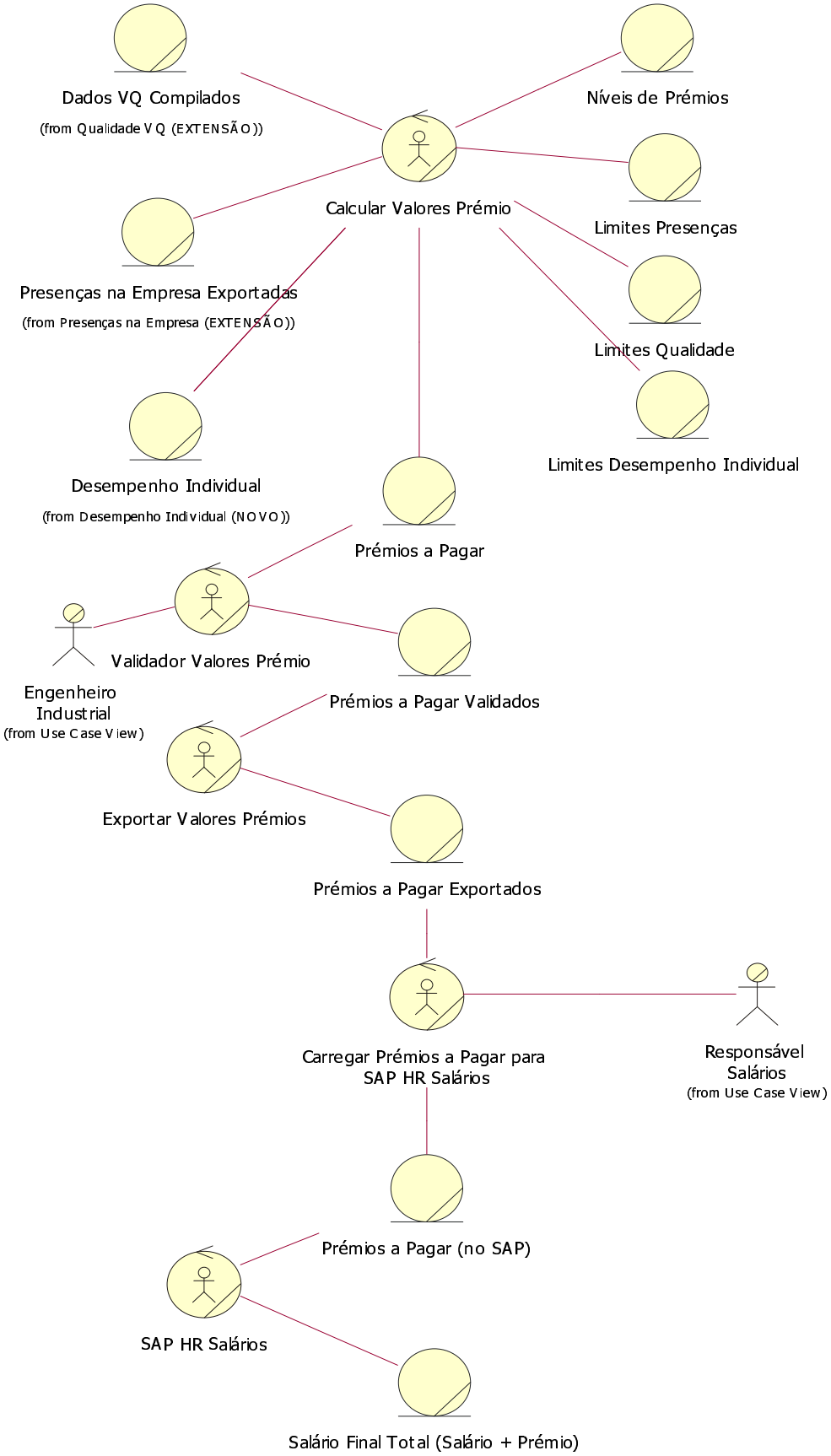

Fig. 6.9 – **Modelo de Objectos de Negócio – Salário Prémio.**

Requisitos

Tabela 6.2 **– Artefactos nos Requisitos.**

Artefacto	Utilizado	Não Utilizado	Motivo	Papel Responsável
Especificação dos Requisitos de Software		X	A colecção de requisitos de software não necessita toda estar agrupada	Especificador de Requisitos
Especificação Suplementar	X		Existência de requisitos não capturados pelo modelo de casos de uso. Aqui vai ser usada a abordagem FURPS+	Analista de Sistemas
Modelo de Casos de Uso	X		Essencial na captura de requisitos, além de servir como contrato entre o cliente e a organização que desenvolve software	Analista de Sistemas
Visão		X	A visão dos interessados acerca das necessidades e características estava previamente definida (imposta pela administração)	Analista de Sistemas
Plano de Gestão dos Requisitos		X	A gestão dos requisitos foi feita por via oral.	Analista de Sistemas
Pedidos dos Interessados	X		Visão global agrupada pelos interessados dos pedidos	Analista de Sistemas
Pacote Caso de Uso		X	Agrupamento por pacotes dos casos de uso não é relevante	Especificador de Requisitos
Atributos dos Requisitos	X		Repositório comum dos requisitos, atributos e dependências	Analista de Sistemas
Glossário		X	Dimensão (4 pessoas) e conhecimento prévio da equipa de desenvolvimento não necessita de um glossário	Analista de Sistemas
Storyboard dos Casos de Uso		X	A interacção dos utilizadores com os interfaces do sistemas vão ser desenvolvidos pela equipa e testados nos protótipos executáveis. Os interfaces de 4 dos 5 módulos do sistema global já são conhecidos dos utilizadores	Desenhador dos Interfaces com o Utilizador
Protótipo de Interface com o Utilizador		X	Os interfaces de 4 dos 5 módulos do sistema global já são conhecidos dos utilizadores. O restante módulo segue as normas vigentes na organização	Desenhador dos Interfaces com o Utilizador

Análise e Desenho

Tabela 6.3 – **Artefactos na Análise e Desenho.**

Artefacto	Utilizado	Não Utilizado	Motivo	Papel Responsável
Modelo de Entrega	X		Modelo dos nós de processamento, comunicação, e objectos contidos em cada nó	Arquitecto de Software
Modelo de Análise		X	Apenas usado o modelo de desenho	Arquitecto de Software
Modelo de Desenho	X		Essencial para a implementação e teste	Arquitecto de Software
Modelo de Dados	X		Dados persistentes são relevantes neste sistema	Desenhador da Base de Dados
Documento da Arquitectura do Software	X		Visão detalhada da arquitectura do sistema	Arquitecto de Software
Prova do Conceito de Arquitectura		X	Os requisitos dos interessados não apresentam riscos significativos de arquitectura	Arquitecto de Software
Desenho Subsistemas		X	Sistema simples apenas com classes e pacotes	Desenhador
Pacote de Desenho	X		Estruturação do modelo de desenho em subpacotes	Desenhador
Classe de Desenho	X		Classes de desenho do sistema. Aqui foram usados também os estereótipos de interface, controlo, dados para as classes de desenho	Desenhador
Classe de Análise	X		Classes de análise do sistema. Usadas para capturar em alto nível os agrupamentos de responsabilidades do sistema	Desenhador
Cápsula		X	Apenas para sistemas de tempo real	Desenhador de cápsulas
Realização de Casos de Uso	X		Representação da realização dos casos de uso no modelo de análise	Desenhador
Referências de Arquitectura	X		Artefactos já disponíveis na organização e que permitem a reutilização	Arquitecto de Software

Implementação

Tabela 6.4 – **Artefactos na Implementação.**

Artefacto	Utilizado	Não Utilizado	Motivo	Papel Responsável
Modelo de Implementação	X		Descrição dos subsistemas de implementação e respectivos componentes (código executável ou fonte)	Arquitecto de Software
Subsistemas de Implementação	X		Usado para estruturação do modelo de implementação	Implementador
Componente	X		Código fonte ou ficheiros de inicialização ou de ajuda	Implementador
Plano de Integração		X	O plano detalhado de integração dentro de uma iteração não é necessário devido às características da equipa de desenvolvimento	Integrador
Construção	X		Compilação e linking do código fonte	Integrador

Teste

Tabela 6.5 – **Artefactos no Teste.**

Artefacto	Utilizado	Não Utilizado	Motivo	Papel Responsável
Plano de Teste	X		Objectivos, estratégias, e recursos necessários aos testes no projecto	Desenhador de Testes
Documento de Análise da Carga	X		Base para previsão da carga dos sistema	Desenhador de Testes
Procedimento de Teste		X	Já conhecidos e usados pela equipa	Desenhador de Testes
Script de Testes		X	Testes manuais	Desenhador de Testes
Casos de Teste	X		Identificação das condições a testar	Desenhador de Testes
Modelo de Teste	X		Estrutura e conteúdo do modelo de testes	Desenhador de Testes
Sumário de Avaliação de Testes	X		Resultados dos testes e medidas	Desenhador de Testes
Resultados dos Testes		X	Repositório de dados capturados durante os testes que servem de base para indicadores.	Testador
Classe de Teste		X	Não foram implementadas funcionalidades específicas de teste	Desenhador
Pacote de Teste		X	Não foram implementadas funcionalidades específicas de teste	Desenhador
Componente de Teste		X	Não foram implementadas funcionalidades específicas de teste	Implementador
Subsistema de Teste		X	Não foram implementadas funcionalidades específicas de teste	Implementador

Entrega

Tabela 6.6 – **Artefactos na Entrega.**

Artefacto	Utilizado	Não Utilizado	Motivo	Papel Responsável
Plano de Entrega	X		Descrição das tarefas para instalar o produto final	Gestor de Entrega
Lista de Materiais		X	O produto foi desenvolvido à medida e apenas vai ter uma instalação	Gestor de Entrega
Notas de Entrega		X	Documento com identificação de alterações e falhas conhecidas não é relevante pois a organização de suporte está muito relacionada com a organização alvo	Gestor de Entrega
Produto	X		Produto final de software	Gestor de Entrega
Artefactos de Instalação	X		A instalação é feita usando ferramentas standard na organização (Windows Terminal Server® e Bosch PeaCy)	Implementador
Materiais de Formação		X	A formação dos utilizadores não vai ser feita formalmente em sala, logo os materiais usados nas formações vão ser basicamente os manuais de utilizadores finais	Responsável de Cursos
Unidade de Entrega	X		Consiste numa construção do programa, documentação, e artefactos de instalação	Gestor de Configuração
Arte no Produto		X	Não vão existir marca ou logotipos para o produto	Artista Gráfico
Materiais de Apoio ao Utilizador Final	X		Ensinar utilizador como usar o produto. Diminui capacidade necessária para suporte	Escritor Técnico

Configuração e Gestão da Mudança

Tabela 6.7 **– Artefactos na Configuração e Gestão da Mudança.**

Artefacto	Utilizado	Não Utilizado	Motivo	Papel Responsável
Resultados de Auditoria de Configuração		X	As actividades e artefactos iriam ser auditadas pela equipa de execução do projecto. Logo, não faz sentido executar a auditoria	Gestor de Configuração
Plano de Gestão da Configuração	X		Plano de prazos com actividades, responsabilidades, recursos durante ciclo de vida do projecto	Gestor de Configuração
Repositório do Projecto	X		Directorias e ficheiros com versões dos produtos	Gestor de Configuração
Pedido de Alteração		X	Dado a comunicação informal dentro da organização, os pedidos de alterações foram feitos de uma forma oral	Gestor de Controlo das Mudanças
Espaço de Trabalho (Integração)	X		Área comum onde as construções do produto final estão presentes	Integrador
Espaço de Trabalho (Desenvolvimento)	X		Área de desenvolvimento onde os artefactos quando são alterados são imediatamente visíveis por toda a equipa	qualquer papel

Gestão do Projecto

Tabela 6.8 – **Artefactos na Gestão do Projecto.**

Artefacto	Utilizado	Não Utilizado	Motivo	Papel Responsável
Plano de Desenvolvimento do Software	X		Coleccionar toda a informação necessária para a gestão do projecto	Gestor do Projecto
Caso de Negócio		X	A decisão para implementação do sistema obedeceu apenas a critérios políticos internos e não a retorno de investimento	Gestor do Projecto
Plano de Iterações	X		Os interessados ficam a saber quais as funcionalidades esperadas nos protótipos e quais as datas de publicação	Gestor do Projecto
Verificação das Iterações		X	A captura das lições aprendidas durante as iterações foi feita sem recurso a este artefacto	Gestor do Projecto
Verificação do Estado	X		A verificação periódica do estado do projecto foi feita semanalmente pela equipa de coordenação do projecto	Gestor do Projecto
Plano de Resolução de Problemas		X	Dada a dimensão da equipa e da relação com a organização alvo a resolução de problemas segui uma via informal	Gestor do Projecto
Plano de Gestão dos Riscos		X	Integrado no Plano de Desenvolvimento de Software	Gestor do Projecto
Lista de Riscos	X		Fornece a base para minorar ou eliminar os possíveis problemas futuros	Gestor do Projecto
Ordem de Trabalho		X	A dimensão da equipa não obriga a usar este artefacto	Gestor do Projecto
Plano de Aceitação do Produto	X		Critérios que definem se os artefactos resultantes aos clientes são aceitáveis por este	Gestor do Projecto
Plano de Medição	X		Definição dos objectivos e indicadores para monitorar o progresso do projecto	Gestor do Projecto
Plano de Garantia da Qualidade		X	A avaliação dos resultados de qualidade atingidos pelo processo foi feita à posteriori por ser uma das primeiras vezes usado	Gestor do Projecto
Medidas do Projecto	X		Repositório de dados métricos relativos ao projecto	Gestor do Projecto
Registos de Verificação		X	Não foram utilizados verificações aos artefactos por ser um processo ainda recente na organização	Verificador do Projecto

Ambiente

Tabela 6.9 – **Artefactos no Ambiente.**

Artefacto	Utilizado	Não Utilizado	Motivo	Papel Responsável
Casos de Desenvolvimento	X		Descrição do desenvolvimento escolhido para este projecto	Engenheiro do Processo
Verificação da Organização de Desenvolvimento	X		Estado actual dos processos, ferramentas, competências, atitude, clientes, problemas, desafios tecnicos, e áreas de desenvolvimento	Engenheiro do Processo
Formatos Próprios do Projecto		X	Os formatos dos artefactos e relatórios é o formato standard usado na organização alvo	Engenheiro do Processo
Linhas de Orientação para o Desenho		X	Experiência, dimensão, e conhecimento mútuo não exigem estas orientações	Arquitecto de Software
Linhas de Orientação para a Programação		X	Experiência, dimensão, e conhecimento mútuo não exigem estas orientações	Arquitecto de Software
Linhas de Orientação da Modelação do Negócio		X	Modelação de Negócio executados apenas por uma pessoa	Analista do Processo de Negócio
Linhas de Orientação da Modelação dos Casos de Uso		X	Casos de uso executados apenas por uma pessoa	Analista de Sistemas
Linhas de Orientação do Interface dos Utilizadores		X	Experiência, dimensão, e conhecimento mútuo não exigem estas orientações	Desenhador do Interface com Utilizador
Linhas de Estilo dos Manuais		X	Não relevante neste projecto	Escritor Técnico
Linhas de Orientação de Testes		X	Experiência, dimensão, e conhecimento mútuo não exigem estas orientações	Desenhador de Testes
Linhas de Orientação de Ferramentas		X	Experiência, dimensão, e conhecimento mútuo não exigem estas orientações	Especialistas de Ferramentas
Ferramentas	X		Ferramentas que suportam o processo de desenvolvimento	Especialistas de Ferramentas
Infra-estrutura de Desenvolvimento	X		Infra-estrutura já existente e usada na organização	Administrador do Sistema

Outros Artefactos Relevantes

Além dos artefactos propostos pelo RUP, considera-se relevante que existam documentados em cada projecto de desenvolvimento duma aplicação de software:

- Lista de verificação para avaliação do estado do projecto nas Verificações de Qualidade que avaliam a transição entre as fases do processo (início, elaboração, construção, e transição);
- Planeamento e uma análise da capacidade requerida (como proposto no capítulo 2.5) para suporte e manutenção do sistema quando este estiver na fase produtiva.

Para o presente caso de estudo são estimados os seguintes valores:
PASSO 1: Estimar o Volume de Pedidos de Chegada (VP):

Tabela 6.10 – **Estimativa do Volume de Pedidos.**

Unidade Organizacional	N.º Pedidos / dia	Tempo Resolução (dias)	
Produtividade por Linha	1	1/8	
Qualidade VQ	1/3	1/16	
Presenças na Empresa	1/22	1/8	
Desempenho Individual	1/22	1/8	**Volume de Pedidos (p/d)**
Cálculo Salário Prémio	1/22	1/8	
TOTAL	1,46	0,56	0,82

Estima-se que cada pedido é servido apenas por uma pessoa.

PASSO 2: Estimar o nível de preparação adequado:
Na aplicação actual foi acordado que um nível de preparação de 80% é adequado.

PASSO 3: Determinar Quantidade Base de Pessoas:
Usando a tabela 2.1, com VP = 0,82 e NP = 80%, obtém-se um valor inferior ou igual a 2 pessoas.

PASSO 4: Adicionar Capacidade Extra:
No caso actual a capacidade de 2 pessoas parece suficiente devido à experiência da organização no suporte a sistemas semelhantes.

PASSO 5: Estabelecer Estratégia de Diminuição Pedidos na Fila:
Na aplicação actual, são considerados os seguintes pontos:

- Os pedidos vindos das unidades organizacionais "Produtividade por Linha" e "Cálculo Salário Prémio" são de alta prioridade;
- Os pedidos de todas as outras unidades organizacionais são de média prioridade;
- Os pedidos de alta prioridade são sempre servidos em primeiro lugar.

Existe também a possibilidade de se criarem filas com prioridades separadas para cada subsistema, mas esta não foi a opção pois pretendeu dar-se uma estimativa global para o suporte e manutenção da aplicação que arca com o processo de negócio completo.

6.3.3 Análise dos Resultados

Nesta secção são analisados os pontos positivos e negativos da parametrização do RUP para a implementação da aplicação que modela o processo de negócio do Salário Prémio, e são descritas as lições assimiladas.

Assim, como principais pontos positivos, nesta parametrização particular do RUP, foram identificados:

- O RUP, e principalmente os artefactos do seu fluxo de trabalho Modelação do Negócio, tiveram um bom desempenho na modelação do processo de negócio pois serviram de base para os artefactos de fluxos de mais baixo nível;

- Os artefactos UML, com estereótipos sugeridos no RUP, foram um meio muito eficaz de entendimento com todos os interessados;

- Os artefactos gerados no RUP, nomeadamente o código fonte, é de fácil uso para futuros sistemas que o necessitem, permitindo assim a reutilização de código;

- A padronização do processo de desenvolvimento na organização que desenvolve software foi iniciada, como o objectivo final, de adoptar o uso do RUP;

- O RUP provou que pode adaptar-se às necessidades de organizações que desenvolvem software e projectos específicos;

- O conjunto de fluxos de trabalho do RUP permite gerir a complexidade e assegura à organização que desenvolve software que nenhum aspecto importante do desenvolvimento de software vai ser descurado, diminuindo assim o risco de o projecto falhar;

Como principais pontos negativos, neste projecto específico, foram identificados:

- Quanto maior o grau de utilização do RUP e de UML, obviamente que maior é o grau de conhecimentos exigido à equipa de desenvolvimento. É necessário um esforço maior para formação;

- A necessidade de criação de novos papéis para decorrer o processo de desenvolvimento de software, como o desenhador de processos de negócio ou o especialista de ferramentas, obriga a reorganizações internas;

E como lições assimiladas:

- A importância de todos os intervenientes dum processo de negócio (e não apenas o cliente) se aperceberem das tarefas envolvidas no uso dum novo produto de software, ou seja, o RUP recomenda que não apenas o cliente que pede o novo software, mas todos os interessados tenham conhecimento e validem os requisitos do sistema;

- Foi executada a transformação de casos de uso em diagramas de classes usando a seguinte metodologia [Fernandes et al., 2001]:
  (1) Transformar os Casos de Uso do sistema em objectos de três tipos: Interface, Controlo, Dados;
  (2) Em seguida, do conjunto inteiro são apenas mantidos os mais significativos (dependendo da descrição e realização dos casos de uso);
  (3) Agregar os objectos comuns restantes (e.g. os objectos relacionados com bases de dados);
  (4) Criar ligações entre os agrupamentos do passo 3.

  Em seguida, dos objectos restantes foi inferido um mapeamento numa arquitectura de três níveis. Assim, os agrupamentos de objectos resultantes da aplicação do método anterior, foram distribuídos pelas três camadas do modelo de níveis (Apresentação, Negócio, e Dados) do seguinte modo: os agrupamentos de Interface foram assignados à camada de Apresentação, os agrupamentos de Controlo foram atribuídos à camada de Negócio, e os de Dados à camada de Dados. Este método de transformação de casos de uso numa arquitectura, permite começar a ter uma vista mais aproximada de qual irá ser a arquitectura final do sistema;

- Nas organizações mistas, como as da fig. 5.1 b), surgem frequentemente problemas externos aos clientes e à equipa de desenvolvimento, apenas porque ainda existem departamentos, e sendo eles interessados tentam bloquear o desenvolvimento de processos organizacionais pois alguns dos interesses dos seus departamentos são postos em causa;

- A utilização do RUP, mesmo durante os primeiros projectos, não constitui um factor para atraso no projecto.

7. Conclusões

7.1 Trabalho Realizado

Para além das conclusões obtidas nos casos de estudo, nas secções 6.2.2 e 6.3.3, e relativamente aos objectivos propostos no início desta dissertação de mestrado, podem-se coligir os seguintes comentários:

- As capacidades de modelação e o entendimento que uma linguagem gráfica de modelação proporciona a todos os interessados, é na realidade uma mais valia no evitar de erros de comunicação que podem-se traduzir em funcionalidades mal implementadas;

- A existência de um processo que garanta à organização que desenvolve o software um controlo sobre as actividades e uma garantia da qualidade dos produtos é uma mais valia.

- Quando a organização alvo pode ser modelada numa organização orientada aos processos, como a da fig. 5.4, o RUP provou que consegue tratar da sua modelação e da transformação de processos de negócio em aplicações informáticas que os suportam;

- O RUP demonstrou (caso de estudo "Salário Prémio", na unidade organizacional "Produtividade por Linha") que consegue tratar com sistemas já em execução, desenvolvidos sem uma perspectiva orientada aos processos, fazendo a sua integração num novo sistema, e com a reengenharia do processo de negócio, através de diagramas de casos de uso de negócio da situação actual e da situação futura (bem como das respectivas realizações de casos de uso);

- O modelo genérico para organizações orientadas a processos, que inclui entre outros a forma de gestão da organização, capaz de modelar adequadamente as organizações alvo;

- Foi proposto um modelo genérico para organizações que desenvolvem software (fig. 5.5), incorporando os fluxos de trabalho do RUP e outros processos de negócio importantes para uma organização deste tipo;

- Os conceitos expostos nesta dissertação foram postos em prática através de casos de estudo.

Relativamente a aspectos mais específicos foram extraídas as seguintes conclusões:

Linguagens de Modelação – UML

- UML é um bom interface entre todos os interessados. Esta situação é relevante devido ao facto de o cliente final não ser normalmente um especialista informático e ter dificuldades em validar e entender os artefactos de um projecto;

- Embora não possua uma formalidade completa, considera-se que é um passo para se conseguir uma maior automação dentro dos fluxos de trabalho num processo de desenvolvimento de software, e logo para uma geração de código mais automatizada e com mais qualidade.

Processos de Desenvolvimento de Software – RUP

- Um bom entendimento dos processos de negócio é essencial para a construção de sistemas correctos. O RUP recomenda e fornece o suporte para este fluxo de trabalho poder ser executado correctamente;

- Os protótipos executáveis são fundamentais para o sucesso do produto final de software, visto transformarem algo intangível, como o software durante o seu processo de desenvolvimento, em algo que pode ser visto e comentado. Trata-se de uma característica fundamental para a validação intermédia do produto;

- O peso burocrático e a configuração do processo deve ser adequado a cada organização e principalmente a cada projecto;

- O macro-modelo contido no RUP adequa-se a outras definições de processos internos já existentes em organizações que desenvolvem software (e.g. versão do modelo em cascata na Bosch), visto permitir usar-se Verificações de Qualidade para passagem de fase não proprietárias;

- As ferramentas de suporte ao processo de desenvolvimento de software são um meio poderoso para se evitar o desperdício de capacidade da equipa de desenvolvimento;

- A qualidade das pessoas envolvidas no processo de desenvolvimento de software é o peso mais relevante para a qualidade final do produto.

Organização Alvo

- A reengenharia de processos de negócio pode ser feita com o recurso à modelação disponibilizada no RUP no fluxo de trabalho "Modelação de Negócio". Quando se substituem trabalhadores de negócio, que representam pessoas ou sistemas legados, num diagrama de objectos de negócio da situação actual, por trabalhadores de negócio numa aplicação informática actualizada recorrendo-se a um diagrama de casos de uso de negócio da situação futura, prova-se que pode existir um processo de análise duma situação e a sua transformação noutra com mais valor para a empresa. Esta é a base da reengenharia;

- A estrutura da organização onde vai ser implementado o projecto é um factor condicionante para o processo de desenvolvimento de software e para o sucesso da sua utilização.

Organização que Desenvolve Software

- A maior valia de uma organização que desenvolve software não é a programação. Ao entendimento dos requisitos dos clientes deve ser dado cada vez mais relevo, tendo as tarefas de mais baixo nível tendência para serem automatizadas;

- O planeamento do suporte e da manutenção duma aplicação deve ser feito durante o seu processo de desenvolvimento;

- A organização que desenvolve software deve dar garantias para a qualidade do produto ao cliente. A estrutura da sua própria organização, o processo de desenvolvimento que utiliza, e a qualidade dos seus colaboradores são factores decisivos.

A engenharia de software, como várias outras actividades de controlo de projectos, embora exiba características distintas devido ao tipo de produto que gera, está também limitada por factores orçamentais, de calendário, ou de outro tipo, que conduzem a que por melhor que seja o processo usado, o produto gerado não seja sempre óptimo, mas deva pelo menos ter uma qualidade aceitável pelo cliente.

7.2 Trabalho Futuro

Seguidamente são apresentadas algumas propostas para o desenvolvimento de actividades futuras relacionadas com a presente dissertação:

- Alcançar um maior detalhe na descrição dos processos e sub-processos duma organização que desenvolve software orientada aos processos, como a mostrada na fig. 5.5. Este modelo detalhado permitiria a comparação do modelo com mais organizações existentes, proporcionando-lhes uma base para fazer a sua própria reengenharia de processos;

- Prever e incluir distintos modelos de organizações para cumprimento de níveis de CMM maiores que 2. Deste modo, a situação futura de uma organização que desenvolve software estaria perfeitamente descrita concedendo-lhes a possibilidade de executarem um plano de transição com resultados mais garantidos;

- Definir um processo auxiliar de parametrização do RUP, pesando aspectos relativos à organização alvo, à organização que desenvolve software, e ao projecto específico. Esta escolha seria uma base de trabalho adequada para organizações que não têm conhecimento para executar a correcta parametrização do RUP.

- Criar um ambiente que baseado na modelação dos processos duma organização permitisse a sua execução, teste, validação, e previsão dos impactos da reengenharia ou de implementação de novos processos.

Bibliografia

[Adams et al., 2001] Adams, E., Guckenheimer, S. (2001). *Achieving Quality by Design. Part I: Best Practices and Industry Challenges. Part II: Using UML*, Rational Edge.

[Agarwal et al., 1999] Agarwal, R., De, P., Sinha, A. (1999). *Comprehending Object and Process Models: An Empirical Study*, IEEE Transactions on Software Engineering, Vol. 25, N.º 4, Julho/Agosto.

[Almeida et al., 1997] Almeida, J., Barbosa, L., Neves, F., Oliveira, J. (1997). *Camila: Prototyping and refinement of constructive specifications*. Em M. Johnson, editor, *Algebraic Methodology and Software Technology*, Springer LNCS, Dezembro. Proceedings da 6.ª Conferência Internacional AMAST'97, Sydney, Australia.

[Anderson et al., 1998] Anderson, A., C3 Team (1998). *At Chrysler, Objects Pay – Extreme Programming*, White Paper, Chrysler Corporation, Outubro.

[Arango, 1989] Arango, G. (1989). *Domain Analysis – From Art Form to Engineering Discipline*, Proc. International Workshop on Software Specification and Design, (Pittsburgh PA, 1989), IEEE Computer Society Press, 1989, pp. 152-159.

[Armour, 2001] Armour, P. (2001). *Zeppelins and Jet Planes: A Metaphor for Modern Software Projects*, Communications of the ACM, Vol. 44, N.º 10, Outubro.

[Augustine, 2001] Augustine, L. (2001). *Using the Rational Unified Process (RUP) Successfully for Small Development Projects*, Rational Edge.

[Behrends et al., 2000] Behrends, R., Stirewalt, R. (2000). *The Universe Model: An Approach for Improving Modularity and Reliability of Concurrent Programs*, SIGSOFT FSE 2000: pp. 20-29, ACM.

[Boehm, 1988] Boehm, B. (1988). *A Spiral Model of Software Development and Enhancement.* IEEE Computer, Vol. 21, N. 5, pp. 61-72.

[Boehm et al., 1990] Boehm, B., Belz, F. (1990). *Experiences With the Spiral Model as a Process Model Generator*, Proceedings of the 5th international software process workshop on Experience with software process models, pp. 43 – 45, IEEE.

[Booch et al., 1998] Booch, G., Martin, R., Newkirk, J. (1998). *Object Oriented Analysis and Design With Applications*, Capítulo preliminar 2.ª Edição, Addison Wesley Longman, Inc.

[Booch, 1998] Booch, G. (1998). *The Visual Modelling of Software Architecture for the Enterprise*, White Paper, Microsoft Corporation.

[Booch, 2000] Booch, G. (2000), *Unifying Enterprise Development Teams with the UML*, White Paper, Rational Software.

[Bosch, 2000] Robert Bosch (2000). *Procedure Model for R/3 Projects*, White Paper interno, Robert Bosch GmbH.

[Bott et al., 1995] Bott, F., Coleman, A., Eaton, J., Rowland, D. (1995). *Professional Issues in Software Engineering*, 2.ª Edição, University College London Press, ISBN 1-85728-450-X PB.

[Broy, 2001] Broy, M. (2001). *Toward a Mathematical Foundation of Software Engineering Methods*, IEEE Transactions on Software Engineering, Vol. 27, N.º 1, Janeiro.

[Caliò et al., 2000] Caliò, A., Autiero, M., Bux, G. (2000). *Software Process Improvement by Object Technology*, ESSI PIE 27785 SPOT, ACM, 2000.

[Cheatham, 1990] Cheatham, T. (1990). *Process Programming and Process Models*, Harvard University, Estados Unidos, IEEE.

[Curtis et al., 1995] Curtis, B., Hefley, W., Miller, S. (1995). *Overview of the People Capability Maturity Model*, Software Engineering Institute, Carnegie Mellon University, Estados Unidos.

[Eeles, 2001] Eeles, P. (2001). *Capturing Architectural Requirements,* Rational Edge, Novembro.

[Engels et al., 2000] Engels, G., Groenewegen, L. (2000). *Object-Oriented Modelling: A Roadmap*. ACM 2000 1-58113-253-0/00/6.

[Fernandes et al., 2001] Fernandes, J., Machado, R. (2001). *From Use Cases to Objects: An Industrial Systems Case Study Analysis*. 7th International Conference on Object-Oriented Information Systems (OOIS'01), Ed. Y. Wang, S. Patel and R. H. Johnston, Calgary, Canadá, pp. 319-28, Springer-Verlag, Agosto, (ISBN 1-85233-546-7)..

[Fernandes, 2000] Fernandes, J. (2000). *MIDAS: Metodologia Orientada ao Objecto para Desenvolvimento de Sistemas Embebidos*, Tese de Doutoramento, Departamento de Informática, Universidade do Minho, Portugal.

[Fitzgerald et al., 1998] Fitzgerald, J., Gorm Larsen, P. (1998). *Modelling Systems:*

Pratical Tools and Techniques in Software Development, Cambridge University Press, ISBN 0-521-623480.

[Fowler, 1997] Fowler, M. (1997). *Analysis Patterns: Reusable Objects Models*, Addison Wesley, ISBN 0-201-89542-0.

[Glinz, 2000] Glinz, M. (2000). *Problems and Deficiencies of UML as a Requirements Specification Language*, Proceedings of the 10th International Workshop on Software Specification and Design, Novembro.

[Grady, 1992] Grady, R. (1992). *Practical Software Metrics for Project Management and Process Improvement*. Prentice-Hall.

[Hammer, 1997] Hammer, M. (1997), *Além da Reengenharia*, Campus, ISBN 85-352-0107-6.

[Heimbigner, 1989] Heimbigner, D. (1989). *P4: A Logic Language For Process Programming*, Proceedings of the Fifth International Software Process Workshop, Kennebunkport, Me, 10-13 September.

[Heller, 1999] Heller, R. (1999). *Na Senda da Excelência Europeia*, Executive Digest n.º 59, Setembro.

[Henderson-Sellers et al., 2001] Henderson-Sellers, B., Collins, G., Graham, I. (2001). *UML-Compatible Processes*, Proceedings of the 34th Hawaii International Conference on System Sciences, 2001.

[Hruby, 1998] Hruby, P. (1998). *Structuring Specification of Business Systems with UML (with an Emphasis on Workflow Management Systems)*, OOPSLA'98 Business Object Workshop IV.

[Humphrey, 1990] Humphrey, W. (1990). *Modelling Implications of the Personal Software Process*, Software Engineering Institute, Carnegie Mellon University, Estados Unidos.

[ICE, 1999] Integrated Computer Engineering (1999). *16 Critical Software Practices for Performance-Based Management, versão 5.1*. ICE, 1999.

[IDS, 2002] IDS (2002). *Business Driven Software Engineering – ARIS UML Designer,* White paper, Junho 2002, IDS Scheer AG.

[ISO, 2000] Instituto Português da Qualidade (2000). *NP ISO DIS 9001-2000, Norma Portuguesa – Sistemas de Gestão da Qualidade*, IPQ.

[Jacobson et al., 1999] Jacobson, I., Booch, G., Rumbaugh, J. (1999). *The Unified*

Development Process, Addison-Wesley, ISBN 0-201-57169-2.

[Jacobson, 2000] Jacobson, I.(2000), *Applying UML in The Unified Process*, White Paper, Rational Software.

[Jeffries, 1999] Jeffries, R. (1999). *Extreme Testing*, Software Testing & Quality Engineering, Março/Abril.

[Kirchmer, 1998] Kirchmer, M. (1998). *Business Process Oriented Implementation of Standard Software: How to Achieve Competitive Advantage Quickly and Efficiently,* Springer-Verlag, ISBN 3-540-63472-X.

[Krutchen, 1995] Krutchen, P. (1995). *Architectural Blueprints – The "4+1" View Model of Software Architecture*, IEEE Software, Vol. 12, n. 6, pp. 42-50, Novembro.

[Krutchen, 2000] Krutchen, P. (2000). *From Waterfall to Iterative Development – A Challenging Transition for Project Managers*, Rational Edge, Dezembro.

[Krutchen, 2001] Krutchen, P. (2001). *Going Over the Waterfall With the RUP*, Rational Edge.

[Laitinen et al., 2000] Laitinen, M., Fayad, M., Ward, R. (2000). *Software Engineering in the Small*. IEEE Software, vol 17, n. 5, Setembro/Outubro 2000.

[Larsen, 1999] Larsen, P. (1999). *VDM++ Crash Course*, Especificação e Desenvolvimento Formal de Software, Mestrado em Informática 2000/2001, IFAD - Departamento de Informática, Universidade do Minho.

[Lilly, 2000] Lilly, S. (2000) *Use Case Pitfalls: Top 10 Problems from Real Projects Using Use Cases*, Software Development Magazine, Janeiro.

[Mäkinen et al., 2001] Mäkinen, E., Systä, T. (2001). *MAS – An Interactive Synthesiser to Support Behavioural Modelling in UML*, Dep. Of Computer and Information Sciences, University of Tampere, Finlândia.

[Marshall, 2000] Marshall, C. (2000). *Enterprise Modeling with UML: Designing successful software through business analysis*, Addison Wesley, ISBN 0-201-43313-3.

[Mazza et al., 1994] Mazza, C., Fairclough, J., Melton, B., De Pablo, D., Scheffer, A., Stevens, R. (1994). *Software Engineering Standards*,

Prentice Hall, ISBN 0-13-106568-8.

[McUmber et al., 2001] McUmber, W., Cheng, B. (2001). *A General Framework for Formalising UML with Formal Languages*, ICSE 2001, pp. 433-442, IEEE.

[Melton, 1995] Melton, A. (ed.) (1995). *Software Measurement*, Cambridge University Press. ISBN 1-85032-7178-7.

[Microsoft, 2001] Microsoft (2001). *Microsoft Content Management Server 2001*, Microsoft Corporation.

[Mitchell, 2000] Mitchell, R. (2001). *Adding Value to the Unified Process*, Proceedings of the Technology of Object-Oriented Languages and Systems (TOOLS-34'00), IEEE.

[Nakagawa et al., 1990] Nakagawa, A., Futatsugi, K. (1990). *Product-Based Process Models*, SRA Inc., Electrotechnical Laboratory, Japão, IEEE.

[Neves et al., 1999] Neves, F., Silva, J., Oliveira, J. (1999). *Converting Informal Meta-data to VDM-SL: A Reverse Calculation Approach*. VDM in Practice, FM'99: The World Congress on Formal Methods, Setembro.

[Nunes et al., 2000] Nunes, N., Cunha, J. (2000). *Wisdom: A Software Engineering Method for Small Software Development Companies*, IEEE Software, Setembro/Outubro.

[O'Connell et al., 2000] O'Connel, E., Saiedian, H. (2000). *Can You Trust Software Capability Evaluations?*, IEEE Computer, Fevereiro 2000.

[Odell, 1998] Odell, J. (1998). *Advanced Object-Oriented Analysis & Design Using UML.* Cambridge University Press.

[OMG, 2001] OMG (2001). *OMG Unified Modeling Language Specification versão 1.4*, Object Management Group, Inc.

[OMG, 2001b] OMG (2001). *Model Driven Architecture Specification*, Object Management Group, Inc.

[Parunak et al., 2001] Parunak, H., Odell, J. (2001). *Representing Social Structures in UML*, AGENTS'01, ACM.

[Paulk et al., 1996] Paulk, M., Curtis, B., Chrissis, M., Weber, C. (1996). *Capability Maturity Model for Software, Version1.1*, Software Engineering Institute, Carnegie Mellon University.

[Pigosky, 1996] Pigosky, T. (1996). *Practical Software Maintenance: Best Practices for Managing Your Investment*, John Wiley & Sons,

ISBN 0-471-17001-1.

[Pressman, 1997] Pressman, R. (1997). *Software Engineering*, McGraw-Hill, 4.ª Edição, ISBN 0-07-052182-4.

[Probasco, 2000] Probasco, L. (2000). *The Ten Essentials of RUP – The Essence of an Effective Development Process*, White Paper, Rational Software.

[Ramaswamy, 2000] Ramaswamy, R. (2000). *How to Staff Business-Critical Maintenance Projects*, IEEE Software, Maio/Junho 2000.

[Rational, 2000a] Rational (2000), *Rational Unified Process*, White Paper, Rational Software.

[Rational, 2000b] Rational (2000), *Reaching CMM Levels 2 and 3 with the Rational Unified Process*, White Paper, Rational Software.

[Rodrigues et al., 1995] Rodrigues, J., Cardoso, J. (1995). *O Frente-a-Frente Esperado: Entrevista a Michael Hammer e James Champy.* Exame Executive Digest, Dezembro 1995, Editora Exame, Portugal.

[Rombach, 1989] Rombach, H. (1989). *Specification of Software Process Measurement*, Proc. of ISPW5, Maine, USA, Oct. Pp. 127-179, IEEE.

[Rosenberg et al., 2001] Rosenberg, D., Scott, K. (2001). *Top Ten Use Case Mistakes*, Software Development Magazine, Fevereiro.

[Royce, 2000] Royce, W. (2000). *Software Management Renaissance*, IEEE Software, Julho / Agosto.

[Royce, 2001] Royce, W. (2001). *Improving Software Development Economics. Part I: Current Trends. Part II: Reducing Software Product Complexity and Improving Software Processes. Part III: Improving Team Proficiency and Improving Automation. Part IV: Accelerating Culture Change Through Common Sense*, Rational Edge, Abril / Maio / Junho / Julho 2001.

[Rumbaugh et al., 1999] Rumbaugh, J., Jacobson, I., Booch, G. (1999). *The Unified Modeling Language*, Addison-Wesley, ISBN 0-201-30998-X.

[RUP, 2001] Rational Software Corporation (2001). *Rational Unified Process, versão 2001.03.00*. Rational Software Corporation.

[SAP, 2000] SAP (2000). *Manual de Formação ABAP/4.* SAP Portugal, S.A.

[Silva, 2001] Silva, M. (2001). *Projecto final de estágio da Licenciatura em Informática de Gestão: Sistema de Administração de Viagens de Negócio*. Departamento de Sistemas de Informação, Universidade do Minho, Portugal.

[Soley et al., 2000] Soley, R., OMG Staff Strategy Group (2000). *Model Driven Architecture*, White Paper Draft 3.2, Object Management Group, Novembro.

[Sommerville, 1997] Sommerville, I. (1997). *Software Engineering*, Addison-Wesley, 5.ª Edição, ISBN 0-201-42765-6.

[Spurr et al., 1994] Spurr, K., Layzell, P., Jennison, L., Richards, N., editores, (1994). *Software Assistance for Business Re-Engineering*, John Wiley & Sons, ISBN 0-471-94240-5.

[Takang et al., 1996] Takang, A., Grubb, P. (1996). *Software Maintenance*, International Thomson Computer Press, ISBN 1-85032-192-2.

[Voas, 1999] Voas, J. (1999). *Software Quality's Eight Greatest Myths*, IEEE Software, Setembro/Outubro.

[Wang et al., 1999] Wang, Y., King, G., Dorling, A., Wickberg, H. (1999). *A Unified Framework for the Software Engineering – Process System Standards and Models*, Centre for Software Engineering, IVF, Suécia, Research Centre for Systems Engineering, Southampton, Reino Unido.

[Welty et al., 1999] Welty, C., Ferrucci, D. (1999). *Instances and Classes in Software Engineering*, ACM Intelligence, Verão 1999, ACM 1523-8822 99/0600.

[Yang et al., 1997] Yang, K., Pooley, R. (1997). *Process Modelling to Support the Unified Modelling Language*, Proceedings of the COMPSAC'97 – 21st International Computer Software and Applications Conference.

[Zündorf, 2001] Zündorf, A. (2001). *Tutorial: From Use Cases to Code – Rigorous Software Development With UML and Java*, European Software Engineering Conference, Vienna, Austria, September, IEEE.