Agradecimentos

O bom término da presente dissertação de mestrado reflecte uma longa caminhada nem sempre fácil de percorrer, no entanto para ultrapassar os momentos mais tempestuosos em muito valeu o apoio, a compreensão, o sábio conhecimento, a amabilidade, a generosidade e o eterno carinho daqueles que me são mais próximos.

Uma injustiça seria cometida se aqui não fizesse um especial e sentido agradecimento, desde logo:

Ao Professor Doutor Pedro Ribeiro, por ter aceitado orientar este trabalho e pela dedicação e amparo que me prestou, elucidando-me sobre os erros cometidos e apontando caminhos a seguir, que muito contribuíram para a conclusão desta dissertação; quero agradecer-lhe, também, pela permanente disponibilidade para me receber e esclarecer dúvidas, incluindo mesmo o fim-de-semana, revelando um elevado espírito altruísta de serviço à educação dos seus alunos.

Ao Dr. José Pedro, Director do Centro de Tecnologias e Sistemas de Informação da Inspecção-Geral de Finanças, pelo sempre bem-disposto apoio e incentivo à prossecução deste trabalho.

Agradeço, também, às Funcionárias dos Serviços de Pós-Graduação da Universidade do Minho – Pólo de Azurém, sempre extremamente prestáveis na resolução dos problemas burocráticos relacionados com o Mestrado, bem como no esclarecimento de qualquer dúvida.

Um agradecimento especial à minha esposa Cristina, pelo tempo que não foi possível dedicar-lhe, em prol da condução deste trabalho e, sobretudo, por me fazer acreditar que era capaz.

Finalmente, gostaria de agradecer de forma muito especial aos meus Pais, pelo seu carinho, apoio sempre incondicional, e grande incentivo, na prossecução dos meus estudos, e da minha actividade profissional, ao longo da vida.

Bem Hajam meus Pais!

Bem Hajam todos os que comigo colaboraram!

Resumo

A situação actual de elevada concorrência comercial coloca múltiplos problemas às organizações que fazem do software o seu Core Business, designadamente, o controlo dos custos, a qualidade do software, a comunicação entre as organizações e os clientes, o controlo da produtividade e a gestão do risco.

Ao longo do tempo foram desenvolvidas diversas abordagens à gestão de projectos e, mais recentemente, à gestão de projectos de software, o que por sua vez, despoletou muitas propostas metodológicas para o desenvolvimento do software, concretamente, para gestão do ciclo de vida do software (Cascata, Incremental, Evolutivo, etc.), sendo as mais actuais denominadas metodologias ágeis (XP, Scrum, Crystal, etc.).

Acontece que as metodologias tradicionais, e mesmo as mais modernas, não conseguiram, ainda, dar resposta aos principais problemas da gestão de projectos de software. Estes problemas não são exclusivos desta área, pois em todas as áreas da indústria, com maior ou menor intensidade, tem-se verificado a existência destes mesmos problemas, para os quais foram sendo desenvolvidas soluções, com bons resultados, designadamente o desdobramento da qualidade (QFD), o custeio por actividades (ABC) e a sua mais recente evolução, o custeio por actividades orientado ao tempo (Time-Driven ABC), a orçamentação por actividades (ABB) e a correspondente versão temporal (Time-Driven ABB). Estas são opções que, ainda, estão a dar os primeiros passos na indústria do software, existindo muito poucos trabalhos sobre a sua aplicação, de forma integrada, à gestão de projectos de software.

O principal objectivo desta dissertação é propor um Framework de Gestão de Projectos de Software, que procure dar resposta aos problemas que inquietam as organizações dedicadas à produção de software, mediante a aplicação do QFD, o ABC/Time-Driven ABC e ABB/Time-Driven ABB, integrados por meio da Engenharia do Valor (EV) ou de modelos de Programação Linear (LP) – que funcionam como elementos agregadores do framework; para o controlo e monitorização da qualidade e dos custos é proposta a utilização da gestão do valor ganho (EVM).

Abstract

The current situation of high commercial competition, poses many problems for organizations that make software its Core Business, in particular, control costs, software quality, communication between organizations and customers, productivity control and risk management.

Over time, were developed several approaches to the management of projects and, more recently, to software project management, which in turn, triggered many methodological proposals for software development, specifically for managing the software life cycle (Waterfall, Incremental, Evolutionary, etc.), the most current called agile methodologies (XP, Scrum, Crystal, etc.).

However, the traditional methods, and even the most modern, failed responding to the main problems of software project management. These problems are not exclusive from this area, because all areas of industry, with greater or lesser intensity, record the existence of these same problems, for which solutions were developed, with good results, including quality function deployment (QFD), activity-based costing (ABC) and its latest developments, time-driven activity-based costing (Time-Driven ABC), activity-based budgeting (ABB) and the corresponding time-driven version (Time-Driven ABB). These are options that are still taking the first steps in software industry, where is very little work on its application, in an integrated way, to software project management.

The main objective of this dissertation is to propose a Software Project Management Framework, which seeks to respond to the problems that concern organizations dedicated to software production, by application of QFD, ABC/Time-Driven ABC and ABB/Time-Driven ABB, integrated through Value Engineering (EV) or models of Linear Programming (LP) - functioning as framework aggregation elements; for the control and monitoring of quality and costs is proposed the usage of earned value management (EVM).

Índice Geral

Relação de Siglas

ABB	Activity-Based Budgeting
ABC	Activity-Based Costing
ABM	Activity-Based Management
ACWP	Actual Cost of Work Planned
AHP	Analytical Hierarchy Process
BAC	Budget at Completion
BCWP	Budget Cost of Work Planned
BCWS	Budget Cost of Work Scheduled
C/SCSC	Cost/Schedule Control Systems Criteria
CPI	Cost Performance Index
CV	Cost Variance
DSDM	Dynamic Systems Development Method
EAC	Estimated at Completion
ETC	Estimated to Complete
EVM	Earned Value Management
ISO	International Standards Organization
PL	Programação Linear
PMI	Project Management Institute
QD	Quality Deployment
QFD	Quality Function Deployment
QFDr	Quality Function Deployment restricted
RUP	Rational Unified Process
SPI	Schedule Performance Index
SQC	Statistical Quality Control
SV	Schedule Variance
TDABC	Time-Driven Activity-Based Costing
TQC	Total Quality Control
UML	Unified Modeling Language
UP	Unified Process
VAC	Variation at Completion
VE	Value Engineering

WBS	Work Breakdown Structure
XP	Extreme Programming

Índice de Figuras

Índice de Quadros

CAPÍTULO 1 – Introdução

1.1. Enquadramento

A queda das barreiras comerciais e a abertura de novos mercados impelem as organizações de tecnologias de informação a procurarem novos processos, que lhes permitam ultrapassar os actuais limites de actuação. Cada vez mais, as organizações procuram o crescimento através da inovação [Porter, 1998].

No ambiente económico e tecnológico actual, as transformações no mercado são constantes, pelo que as organizações que procuram o sucesso têm que estar preparadas para enfrentar grandes desafios técnicos e para aproveitar as oportunidades de negócio, adoptando processos que lhes permitam transformarem-se e melhorarem continuamente os seus produtos [Porter, 2001-a].

As organizações de todo o mundo estão envolvidas nesta corrida de procura de vantagens competitivas, que lhes permita sobreviver neste ambiente económico difícil, impondo-lhes uma constante de adaptação ou mesmo reinvenção dos seus modelos de negócio. Assim, as organizações vivem hoje uma época de conhecimento e inovação [Porter, 2007].

Neste contexto, as organizações devem adoptar estratégias que lhes permitam encarar as exigências actuais e futuras do mercado, servindo-se das experiências vividas e das informações recolhidas [Porter, 2001-b].

As organizações produtoras de software, também, são desafiadas pela globalização dos mercados e dos meios de produção; também precisam de encontrar o caminho para melhorarem os seus produtos.

As organizações que não se preparam para enfrentar a concorrência, reduzindo os custos de produção, fomentando a qualidade e a inovação, arriscam seriamente a sua posição no mercado [Porter, 1998].

Assim, as organizações que fazem do software o seu negócio e que desejem destacar-se, têm que dirigir as suas acções para alteração da gestão do desenvolvimento dos seus produtos, para torná-los capazes de produzirem software de elevada qualidade, com custos competitivos.

O processo de desenvolvimento do produto é o principal factor de sucesso da organização [Akao, 1990]. A orientação das organizações na procura da efectividade da produção de produtos passa, necessariamente, pela implementação de práticas ou procedimentos que maximizem a performance do processo de produção [Akao, 1990].

O problema da elevada frequência de insucessos no desenvolvimento de software, existe desde os primórdios da revolução computacional e provavelmente persistirá no futuro [Brooks, 1987].

Os maiores erros dos projectos são cometidos no planeamento, durante a análise dos requisitos e na estimativa dos custos [Ewusi-Mensah, 2003].

É expectável que, com o decorrer do tempo, os gestores de projectos de software adquiram mais conhecimentos, que lhes permita compreenderem melhor as complexidades da tecnologia, e consequentemente, reduzir substancialmente a incidência de erros no software [Ewusi-Mensah, 2003].

Porém, até que chegue esse momento, é preciso fazer algo mais para compreender o problema da gestão de projectos de software e procurar ultrapassá-lo.

A gestão de projectos de software depara-se fundamentalmente com cinco grandes problemas, designadamente, os custos, a qualidade, a comunicação, a produtividade e o risco.

1.2. O Problema do Controlo dos Custos

Os diversos estudos efectuados sobre a gestão de projectos revelaram taxas de insucesso, que chegam aos 80%, em que raramente é cumprido o custo e o calendário previstos [Ewusi-Mensah, 2003].

Esta elevada percentagem de insucesso está directamente relacionada com as lacunas existentes no processo de verificação do trabalho executado, bem como no deficiente controlo do custo incorrido, verificando-se que, na maioria dos casos, não existe mesmo qualquer sistema de monitorização dos custos, que de forma periódica, compare o custo estimado (ou planeado) com o custo efectivo [Stepanek, 2005].

Tradicionalmente, o controlo de custos em projectos de software inicia-se com a identificação dos custos directos dos subprodutos do projecto, normalmente com

base numa estrutura hierárquica, que descreve o trabalho a realizar – Work Breakdown Structure (WBS) – para, depois, proceder ao controlo da variação do custo e do calendário, em relação ao que foi planeado [Forsberg et al., 2005]. No entanto, os custos indirectos são distribuídos arbitrariamente por todas as unidades de trabalho realizadas, sem qualquer preocupação discricionária [Cokins, 2001]. Este é um problema sério na actualidade, pois cada vez mais, a indústria de software recorre à utilização de software, produzido pela própria organização ou por terceiros [Fichman e Kemerer, 2001].

Pelo contrário, o processo de custeio por actividades – ABC, procura afectar directamente aos elementos da WBS produzidos, não só os custos directos, mas também os indirectos, tendo em conta o grau em que os elementos do produto consomem actividades do projecto [Kaplan e Cooper, 1998]. O ABC diferencia-se, assim, de outros processos tradicionais, que são suportados por um único factor de contabilização dos custos, normalmente directos, como por exemplo, o número de horas/homem [Kaplan e Cooper, 1998].

1.3. O Problema da Qualidade

Para aumentar a satisfação geral dos clientes e cumprir com os diferentes atributos de qualidade do produto, é necessário que estes sejam considerados no planeamento e concepção do software, não esquecendo, porém, as diversas sensibilidades dos clientes [Evans, 2004]. É importante que cada característica de qualidade relevante seja especificada e avaliada, recorrendo, sempre que possível, a métricas validadas e completamente aceites pelos clientes [Evans, 2004]. A definição de qualidade não é consensual, porém, poderá definir-se, resumidamente, como as características do software que satisfazem a necessidades e expectativas dos clientes, ou conseguem mesmo ultrapassá-las. As expectativas dos clientes, porém, estão sempre a mudar e, tipicamente, algo que hoje é excitante para o cliente, será amanhã um item de considerado básico, isto é, uma vez introduzida uma função excitante, rapidamente, será imitada pela concorrência, e os clientes estarão à espera de encontrá-la em qualquer produto [Xie et al., 2003].

Os clientes começaram a ser, mais e mais, exigentes sobre a qualidade dos produtos e os serviços que consomem, sendo imperativo para qualquer

organização procurar conhecer os requisitos dos seus clientes e tentar dar-lhes resposta. Mas a meta de satisfazer todos os requisitos dos clientes, não é fácil de concretizar.

O desdobramento da função qualidade – QFD, foi desenvolvido e usado em várias organizações no Mundo desde 1960. No início o QFD tinha como principal objectivo assegurar a qualidade de novos produtos, porém, mais tarde foi expandido para poder considerar outros factores [Akao, 1997].

Na verdade, actualmente o QFD abrange múltiplas áreas, como a engenharia, os custos, a fiabilidade, etc., mantendo, no entanto, uma forte relação com o processo de definição da qualidade do produto [Xie et al., 2003].

1.4. O Problema da Comunicação

Os métodos clássicos usados na análise de requisitos têm como principal objectivo a transformação dos requisitos dos clientes, expressos em linguagem natural, em especificações formais, que servirão de base à concepção técnica e à codificação. Porém, a maioria destas abordagens aos requisitos está orientada para a construção de modelos de dados, funções e processos, focalizando sobretudo os aspectos técnicos do software [Liu et al., 2007].

Acontece que os clientes não estão familiarizados com estes modelos, criados para especialistas, mas também não estão interessados em investir em conhecimento, num esforço necessariamente elevado, o que, obviamente, origina múltiplos problemas de comunicação [Herzwurm et al., 2002].

Neste sentido, existe uma dificuldade na comunicação dos clientes com os engenheiros de software (ou gestores de projectos), o que coloca grandes obstáculos à correcta definição do âmbito dos projectos.

O QFD, com a sua linguagem simples e natural, abrangendo múltiplas áreas possui também um grande potencial como agente facilitador da comunicação em projectos de software.

1.5. O Problema do Controlo da Produtividade

O custo planeado e o custo actual do projecto, para um dado período, são insuficientes para traduzir o verdadeiro estado de um projecto, verificando-se a necessidade de introduzir uma terceira variável – o custo planeado para o trabalho efectivamente realizado [Fleming e Koppelman, 1999-a].

À medida que o projecto progride, o custo planeado, do trabalho realizado, deve ser comparado com o custo real, para verificar se existem variações no custo. No entanto, também, o custo planeado para o trabalho realizado deve ser comparado com o custo do trabalho planeado, para o período em causa, servindo de indicador para eventuais desvios ao calendário [Fleming e Koppelman, 1999-b].

Em suma, é necessária uma forma de comparar a informação sob o custo, prestada pelo ABC, ou outro sistema de custeio, com o custo (ou valor) do trabalho efectivamente realizado – produtividade. Para solucionar este problema é preciso introduzir o conceito de gestão por valor ganho – EVM como mecanismo de monitorização e controlo em projectos de software [Budd e Budd, 2005].

1.6. O Problema do Risco

O risco é, comummente, percepcionado como sendo a exposição à possibilidade de condições adversas, o que em projectos de software se traduz, invariavelmente, em perdas económicas [Bohem, 1991].

A gestão do risco preocupa-se com a redução do impacto negativo dos potenciais problemas, assumindo uma atitude pró-activa, por oposição a uma reacção passiva aos eventos. Contudo, uma estratégia eficaz de redução do risco obriga a que, primeiro, sejam identificados os possíveis problemas e, posteriormente, definidas prioridades de acção [Heldman, 2005].

A importância do risco pode ser determinada usando uma abordagem qualitativa ou quantitativa [Kendrick, 2003]; a primeira, recorre ao conhecimento subjectivo dos gestores e engenheiros de software para analisar o risco e, embora exija um menor conhecimento, é pouco precisa; a segunda usa técnicas estatísticas que permitem modelar com precisão um risco em particular [Heldman, 2005]. Acontece que, na maioria dos casos, não há dados suficientes que possibilitem a

utilização imediata das abordagens estatísticas. É, assim, necessário um mecanismo de tradução da análise subjectiva para algo mais concreto, embora seja evidente que a quantificação de uma opinião é bastante difícil [Heldman, 2005].

1.7. Objectivos Gerais

A procura de uma resposta para os problemas descritos motivou o desenvolvimento de uma abordagem à gestão de projectos de software, que discrimine claramente os custos, a qualidade, a produtividade e o risco envolvido, mantendo uma estrutura ágil e facilitadora da comunicação.

A gestão de um projecto do software é, simultaneamente, uma arte e uma ciência, exigindo um conhecimento aprofundado do ciclo de vida do desenvolvimento do software, a saber: definir a visão, recolher os requisitos, planear as tarefas, estimar o esforço exigido, conseguir as pessoas adequadas para o trabalho, criar um calendário, conceber e codificar o software, monitorizar o trabalho, e finalmente, testar o produto.

Mas, durante todo o ciclo de vida, os membros das equipas de projecto, responsáveis por uma ou mais destas tarefas, vêem-se confrontados com múltiplos problemas. É função do gestor de projecto, conhecer suficientemente o trabalho realizado, para poder assegurar que o projecto continua na direcção certa. O gestor de projecto, para ser eficaz, precisa de uma visão de conjunto dos problemas que ocorrem, mas sobretudo, perceber como poderá, na medida do possível, evitá-los e corrigi-los.

Esta investigação aborda as áreas identificadas como sendo problemáticas, de modo a que seja possível ao gestor de projecto gerir de forma integrada a equipa na execução das tarefas decorrentes do projecto, explorando algumas das principais ferramentas, técnicas e práticas, específicas das várias áreas de gestão de projectos, que poderão ser colocadas na prática em projectos de software, designadamente QFD, AHP, ABC e EVM, contribuindo desta forma para melhorar o processo de desenvolvimento de software e, por sequência, aumentar a satisfação dos clientes com os produtos resultantes.

O trabalho a desenvolver tem como meta investigar as possíveis respostas aos problemas que se colocam actualmente à gestão de projectos de software, seguindo uma estrutura de problema-solução, em que para cada um dos problemas é apresentada uma possível solução, procurando, porém, que esta se integre com as restantes, de modo a que, globalmente, se consiga obter um framework integrado da gestão de projectos de software, que possa servir de

orientação aos gestores de projectos. A meta definida é conseguida pelo sucesso dos seguintes objectivos específicos:

- Avaliar as possibilidades do QFD como ferramenta de descrição das várias fases de desenvolvimento do software (identificação de necessidades dos clientes, definição dos requisitos do software, concepção da arquitectura, codificação e testes). Pretende-se, sobretudo, verificar em medida é possível usar o QFD para sistematizar e comunicar, de forma informal, as correlações entre os inputs e os outputs, das diversas fases por que passa o desenvolvimento do software;

- Serão abordadas as possibilidades da utilização de escalas numéricas para os diversos critérios de avaliação do risco, usando o processo de análise hierárquica – AHP para graduar as opiniões emitidas por pessoas, para, assim, determinar em que medida é possível eliminar as arbitrariedades na fixação das escalas de avaliação do risco;

- Considerar, ainda, a possibilidade de construir uma matriz QFD para seleccionar as tarefas que, considerando os vários critérios de risco, contribuem para reduzir as probabilidades de insucesso do projecto.

- Introduzir o conceito de controlo de custos por actividades – ABC na afectação de custos em projectos de software, procurando evidenciar as vantagens sobre outros métodos, designadamente o detalhe da informação proporcionada acerca do "como" e "porquê" das actividades desenvolvidas;

- A investigação a desenvolver, procurará, também, explorar as oportunidades que resultam da combinação da análise de Engenharia do Valor e modelos de Programação Linear com o ABC e o QFD, para definir e avaliar os requisitos do software e as actividades subjacentes;

- Explorar a aplicação da gestão por valor ganho – EVM, como mecanismo de controlo de custos em projectos de software, mas numa perspectiva de gestão por actividades;

No fim desta investigação, espera-se, conseguir integrar o QFD, o ABC e o EVM, de forma a estabelecer uma framework de gestão de projectos de software, que considere ao longo do projecto as reais necessidades dos clientes, e simultaneamente, considere o valor acrescentado pelas actividades desenvolvidas.

1.8. Organização da Investigação

O Capítulo 1 faz a introdução ao trabalho desenvolvido, delimitando as ideias e metas a atingir e a discutir com a pesquisa conduzida;

O Capítulo 2 apresenta o essencial da gestão de projectos, bem como as características processuais e as fases da maioria dos projectos, independentemente da sua natureza, descrevendo, também, os principais processos de software;

O Capítulo 3 estuda a aplicação do processo do desdobramento da função qualidade (QFD), para identificar os requisitos dos clientes e determinar a melhor forma de satisfazê-los, incorporando a sua vontade em todas as fases do desenvolvimento do software. São apresentados os elementos base do QFD e a forma como se relacionam, para depois mostrar como é possível adaptá-los aos projectos de software;

O Capítulo 4 contempla uma análise do sistema de custeio por actividades (ABC); discute-se, também, a gestão por actividades (ABM), e principalmente a orçamentação por actividades (ABB), que, com base nas informações prestadas pelo sistema de custeio por actividades, permite efectuar estimativas e análises dos custos das actividades relacionadas com projectos de software. É ainda abordada a mais recente evolução do ABC – o Time-Driven ABC (TDABC) – e a sua utilização no desenvolvimento de software;

No Capítulo 5 é analisada a integração do processo QFD com o processo TDABC/TDABB, mediado pela Engenharia do Valor, ou os modelos de custos com recurso à Programação Linear, que considerem os custos na decisão sobre os requisitos do software a implementar. Este capítulo aborda, ainda, o EVM como forma de medir a produtividade das actividades e a performance de execução dos requisitos;

No Capítulo 6 são apresentadas as conclusões e algumas recomendações consideradas pertinentes para trabalhos futuros.

CAPÍTULO 2 – Gestão de Projectos

Este capítulo pretende apresentar os princípios básicos da gestão de projectos na actualidade. Este ponto de situação é indispensável para contextualizar as propostas apresentadas nos capítulos seguintes. É colocada ênfase na gestão de processos de gestão e execução de projectos, mas sobretudo nos processos relacionados com a produção de software e nos principais modelos de desenvolvimento usados na actualidade. Estes modelos formam a infra-estrutura sobre a qual assentarão os processos QFD, ABC/ABB e TDABC/TDABB. A gestão de projectos abarca uma vasta gama de conhecimentos, o que obrigou a estabelecer alguns limites às áreas de conhecimento abordadas. No entanto, é possível e desejável, seleccionar as áreas mais importantes, isto é, aquelas que têm um maior impacto directo ou indirecto na satisfação das necessidades dos clientes e onde se situam os problemas mais críticos. Assim, neste capítulo merecem especial atenção a gestão do âmbito, do custo, do tempo, da qualidade e do risco.

2.1. Definição de Projecto

Em geral, pode dizer-se que qualquer organização executa actividades que compreendem operações e/ou projectos. Ambas possuem características comuns, ou seja, são executadas por pessoas, trabalham com recursos limitados e são planeadas, executadas e sujeitas a controlo. Contudo diferem nalguns aspectos: as operações são repetitivas; os projectos são temporários e únicos [Wysocki e McGary, 2003].

Assim, os projectos podem definir-se como empreendimentos únicos, com objectivos claramente definidos, em função de um problema, oportunidade ou interesse, de uma pessoa ou organização [PMI, 2004].

O resultado do projecto é o desenvolvimento de uma solução que corresponde aos interesses do cliente, respeitando restrições temporais e de recursos. O grau de sucesso do projecto depende largamente da medida em que são cumpridos esses critérios [Wysocki e McGary, 2003].

2.2. Aplicação da Gestão de Projectos

Normalmente, a gestão de projectos pode ser aplicada a qualquer empreendimento ou actividade, que tenha um início e fim definidos, e que possua metas bem claras [PMI, 2004].

O Project Management Institute [2004] define a gestão de projectos como a aplicação de conhecimentos, capacidades, ferramentas e técnicas que procuram atingir os requisitos do projecto e satisfazer ou exceder as necessidades e expectativas dos indivíduos e organizações interessados no projecto, também conhecidos como Stakeholders [PMI, 2004].

Contudo, em resultado das experiências de muitas organizações na implantação de técnicas de gestão chegou-se à conclusão de que a introdução de conceitos e técnicas sobre a gestão de projectos nas organizações é um processo difícil [Bechtold, 1999].

As mudanças culturais representam as maiores dificuldades, pois as alterações dos processos, não são facilmente aceites por alguns membros da organização, que não acreditam nos benefícios que as mudanças trazem para a organização [Evans 2004].

A principal vantagem das práticas da gestão de projectos está na aplicabilidade a projectos de qualquer tipo, independente do tamanho, complexidade, necessidades e restrições de recursos [Bechtold, 1999].

Entre os benefícios da implementação da gestão de projectos, destacam-se as seguintes:

- Facilita o processo de decisão, uma vez que as informações são disponibilizadas de forma estruturada [Cook, 2005];

- Aumenta o controlo de todas as fases de desenvolvimento a percorrer, mediante a descrição detalhada de todas as actividades a executar [Cook, 2005];

- Antecipa eventuais situações desfavoráveis, para que as eventuais acções correctivas e/ou preventivas, possam ser tomadas antes de se transformarem em problemas reais [Cook, 2005];

- Facilita e orienta as revisões da estrutura do projecto, motivadas por alterações no mercado ou na organização, permitindo a adaptação do projecto ao novo contexto [Hallows, 2005];

- Optimiza a atribuição de recursos [PMI, 2004].

2.3. Projectos de Software

A preocupação com a gestão de projectos de software tem vindo a aumentar, representando hoje um ramo perfeitamente definido da engenharia de software [McConnell, 1996].
Mas, apesar da grande evolução registada na área de desenvolvimento de software, com efeito, ainda se verifica actualmente uma reduzida taxa de aceitação do produto pelos clientes, o que é um facto marcante [Ewusi-Mensah, 2003].
Um dos motivos para a falta de aceitação do produto prende-se com a inexistência de um processo de software, que garanta a entrega do produto no tempo previsto e com a qualidade desejada [Ewusi-Mensah, 2003].
O aumento do interesse pela gestão de projectos resulta também da preocupação crescente com a rentabilização da produção de software [Denne e Cleland-Huang, 2003].

2.4. Variáveis da Gestão de Projectos

O gestor de projecto deve preocupar-se fundamentalmente (1) com os aspectos económicos, influenciados pelos interesses comerciais da organização, e (2) com os aspectos técnicos, relacionados com as actividades de desenvolvimento do projecto (análise, concepção, implementação, teste, etc.) [Cook, 2005].
O tempo, a qualidade, o custo e o âmbito são as quatro principais variáveis a considerar na gestão do projecto [Wysocki e McGary, 2003; Kendrick, 2003; Bainey, 2004; Newell e Grashina, 2004]. As restantes variáveis podem ser deduzidas a partir destas quatro. Inevitavelmente, existe uma tensão entre estas variáveis: o tempo pode ser reduzido, mas somente aumentando os gastos (prejudicando o custo) ou diminuindo o potencial do software (reduzindo o âmbito e/ou a qualidade), ou mesmo prejudicando os três. Da mesma maneira, a

qualidade pode ser aumentada somente se o custo, o tempo ou o âmbito forem prejudicados, ou prejudicando os três [Wysocki e McGary, 2003].

Na Figura 1, é ilustrada a relação entre âmbito, custo, tempo e qualidade de um projecto. O que se observa é que, quando um elemento é favorecido (representado por uma área maior), pelo menos uma das três restantes áreas é prejudicada (representados através de uma área mais reduzida). Por exemplo, quando o projecto está atrasado, somente será possível terminar no prazo estipulado se diminuir o âmbito e/ou a qualidade e/ou, aumentar os custos, com vista a proceder à aquisição de mais recursos.

Em ambientes onde as alterações são frequentes, é comum haver uma reavaliação contínua da relação âmbito, custo, tempo e qualidade durante o desenvolvimento.

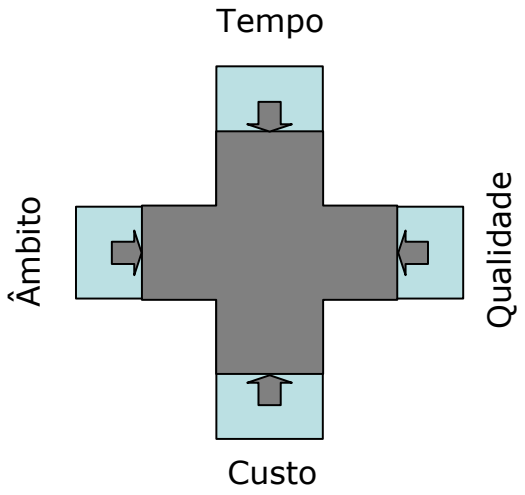

Figura 1: Relações de Pressão entre Âmbito, Custo, Tempo e Qualidade (Adaptado de [Wysocki e McGary, 2003; Newell e Grashina, 2004])

O gestor deve garantir um balanceamento adequado entre estas quatro variáveis, de forma a alcançar um equilíbrio que garanta o sucesso do projecto.

Mas os projectos não conseguem o sucesso sem um planeamento adequado. É necessário um planeamento, que permita ao gestor de projecto controlar a performance das diferentes variáveis durante a transição entre as diversas fases do projecto (ciclo de vida do projecto) [Bainey, 2004].

2.5. Ciclo de Vida do Projecto

O ciclo de vida do projecto define o trabalho a realizar em cada fase do projecto e quais os recursos a usar na sua execução. As fases do ciclo de vida são específicas de cada tipo de projecto e variam entre as organizações, porém, na maioria dos casos, o ciclo de vida dos projectos divide-se em quatro fases principais – iniciação, planeamento, execução e encerramento – às quais estão associados vários processos, mas que muitas das vezes se sobrepõem [PMI, 2004] (ver Figura 3).

Normalmente, antes que uma fase termine, a fase seguinte é iniciada. Este processo de sobreposição das fases do projecto é designado por Fast Tracking (compressão do calendário do projecto, pela sobreposição de actividades que normalmente estariam em sequência) [Newell e Grashina, 2004].

2.6. Organização do Projecto por Processos

Para o PMI, a gestão de projectos é composta por vários processos, descritos em termos do seu inter-relacionamento e dos objectivos a alcançar por cada um deles. Os processos estão reunidos em cinco grupos, a saber: Iniciação, Planeamento, Execução, Monitorização e Controlo e Encerramento [PMI, 2004] (ver Figura 2).

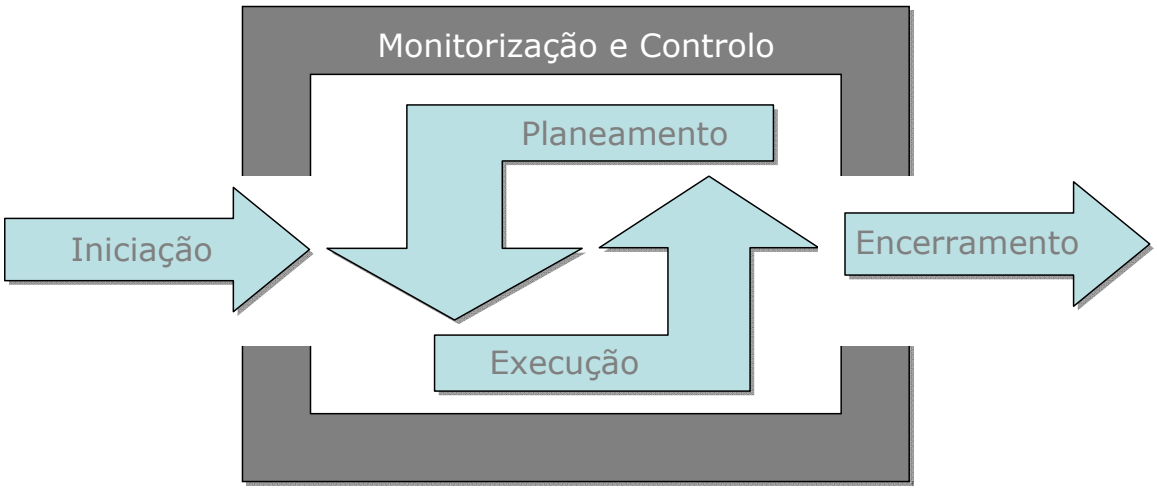

Figura 2: Grupos de Processos que Integram a Gestão de Projectos
(Adaptado de [PMI, 2004])

2.6.1. Processos de Iniciação

Este grupo é composto por processos que procuram definir e aprovar o projecto ou uma das respectivas fases. Como inputs desta fase, contam-se as informações externas ao projecto (por exemplo, o contrato que formaliza o projecto), junto a outras provenientes de fases anteriores, por exemplo, quando esta fase estiver incluída num sub projecto. É mediante estes processos que é seleccionado o gestor do projecto e documentados os pressupostos e restrições iniciais. É também este o momento de decidir se o projecto avança, se é adiado ou abortado [PMI, 2004];

2.6.2. Processos de Planeamento

Os processos deste grupo visam o planeamento e a gestão do projecto. Neste ponto, são elaborados diversos planos e estimativas, como o plano de projecto, a estrutura analítica de trabalho (WBS), as actividades necessárias, milestones, o calendário, a estimativa de custos, entre várias outras informações consideradas necessárias [PMI, 2004];

2.6.3. Processos de Execução

Este grupo preocupa-se com a execução do plano de gestão, a fim de poder cumprir com os requisitos do projecto, e abrange a garantia da qualidade, a coordenação dos recursos (físicos ou humanos) e a partilha de informações. A maior parte dos recursos atribuídos ao projecto serão gastos pelos processos deste grupo [PMI, 2004];

2.6.4. Processos de Monitorização e Controlo

Os processos deste grupo fazem o acompanhamento da execução do projecto, procurando identificar quaisquer variações entre a execução e o planeamento, com intuito de definir acções correctivas que permitam devolver o projecto ao percurso traçado. Adicionalmente, poderão ser recomendadas acções preventivas, para antecipar outros problemas potenciais. Em projectos com múltiplas fases de execução, os processos de monitorização e controlo recolhem as informações respeitantes a uma determinada fase, num dado momento, e transmitem-nas às fases seguintes [PMI, 2004];

2.6.5. Processos de Encerramento

Os processos deste grupo finalizam formalmente as actividades de um projecto ou de uma das suas fases, e verificam se todos os processos dos outros grupos de gestão foram concluídos [PMI, 2004];

2.7. Grupos de Processos de Gestão de Projectos

Os grupos de iniciação e encerramento dão início e fim ao ciclo de vida do projecto, uma vez que, por definição, um projecto terá que ter um princípio e um fim claramente definidos. Os grupos são independentes da área do projecto em que são aplicados, porém, são sempre executados na mesma sequência [PMI, 2004].

Os grupos não se referem a fases do projecto, mas sim a conjuntos de processos que são repetidos em todas as fases ou sub projectos do projecto. No entanto, todos os processos estão interligados, uma vez que, os outputs de uns são os inputs de outros. Além disso, geralmente os grupos não actuam isoladamente, mas são compostos por actividades que, pelo menos num dado momento, são executadas em paralelo ou em conjunto [PMI, 2004] (ver Figura 3).

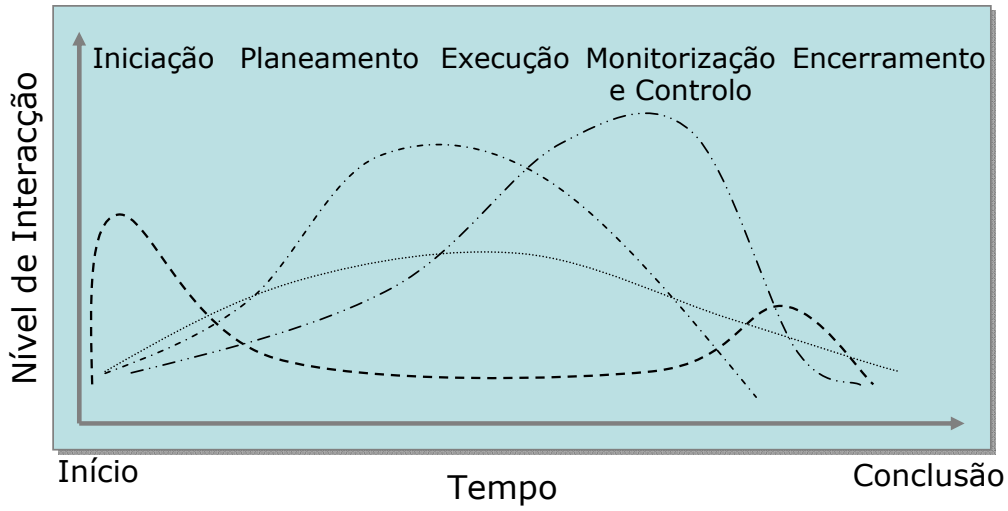

Figura 3: Interacção dos Grupos de Processos
(Adaptado de [PMI, 2004])

2.8. Áreas de Conhecimento Críticas

Além de agregados em grupos, os processos estão organizados por nove áreas distintas: integração, âmbito, tempo, custos, qualidade, recursos humanos, comunicações, riscos e aquisições [PMI, 2004].

Cada uma destas áreas tem as suas próprias particularidades e níveis de complexidade sem, no entanto, deixarem de estar integradas. Certamente, dependendo do projecto ou da organização em questão, existem áreas com maior ou menor relevância, porém, é aconselhável que o gestor de projecto exerça controlo sobre todas elas [PMI, 2004].

Cada uma destas áreas descreve os conhecimentos e as práticas da gestão de projectos em termos dos processos que as compõem [PMI, 2004].

O presente estudo focará a integração da gestão de cinco das nove áreas do conhecimento (ver Figura 4), designadamente, âmbito, custo, tempo, qualidade e risco. A gestão adequada destas áreas tem um papel fundamental na implementação de uma estrutura eficaz de gestão de projectos, que responda aos problemas actuais de que padecem os projectos de software.

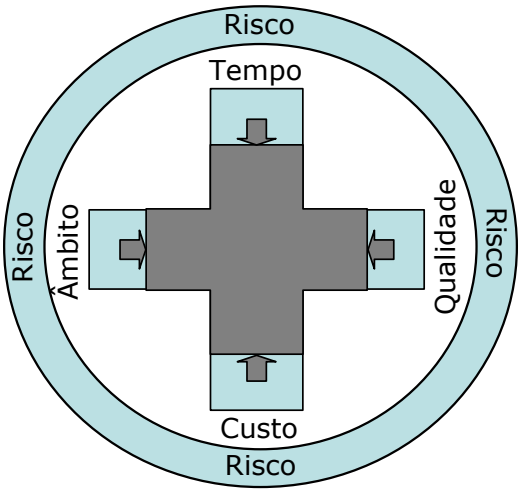

Figura 4: Gestão Integrada do Âmbito, Custo, Tempo, Qualidade e Risco

2.8.1. Gestão do Âmbito

O âmbito do projecto consiste na especificação do produto principal e dos respectivos subprodutos, isto é, quais os resultados do projecto, frequentemente designados por deliverables. O âmbito do projecto define-se assim como o

trabalho a desenvolver, para garantir a entrega de um determinado produto, com todas as especificações e funções pretendidas, numa data pré-definida [Wysocki e McGary, 2003]. A gestão do âmbito reparte-se por cinco processos, a saber:

- Iniciação: Este processo é executado quando se reconhece a necessidade de iniciar um novo projecto. O resultado da iniciação é um Termo de Iniciação, também conhecido como Project Charter, subscrito pela gestão de topo, atribuindo ao gestor do projecto a autoridade para aplicar os recursos da organização em actividades relacionadas com o projecto [Bainey, 2004];

- Planeamento do âmbito: Com este processo é planeado e documentado o âmbito, delimitando também as fronteiras de actuação do projecto, o que servirá de base para as decisões futuras [Marchewka, 2003];

- Definição do âmbito: Este processo consiste na subdivisão dos principais resultados do projecto, em subprodutos menores, de gestão mais fácil. O principal produto deste processo é a estrutura analítica de trabalho do projecto, ou WBS, que divide o resultado final desejado em componentes, actividades e tarefas, estabelecendo uma relação de causa-efeito entre os objectivos do projecto e o planeamento das actividades indispensáveis à sua concretização [Marchewka, 2003];

- Verificação do âmbito: É um processo de aprovação formal do âmbito, após verificação dos resultados do trabalho, de modo a garantir que foram concluídos todos os trabalhos de forma satisfatória [Bainey, 2004];

- Controlo de Alterações ao Âmbito: Este processo avalia as alterações ao âmbito, de modo a garantir que são cumpridos os objectivos pré-estabelecidos [Bainey, 2004].

A gestão do âmbito é o primeiro processo da gestão do projecto. O planeamento do âmbito tem como finalidade produzir a especificação do âmbito, ou seja, documentar as metas do projecto, os resultados práticos esperados e os requisitos do cliente, e o respectivo plano de gestão; a especificação do âmbito constitui a linha de base do projecto, o que significa que, se surgirem dúvidas ou se forem sugeridas alterações, será possível compará-las com o que foi documentado durante o planeamento do âmbito [Wysocki e McGary, 2003].
Nesta fase são tomadas decisões cruciais que influenciarão bastante o custo e a duração do projecto e, consequentemente, a sua viabilidade [Marchewka, 2003].
Uma vez definido o âmbito e a estrutura de trabalho do projecto (WBS), a sua gestão, ao longo da vida do projecto, será a principal função do gestor de projecto.

2.8.2. Gestão dos Custos

Os processos de gestão dos custos do projecto são responsáveis por criar estimativas dos custos e dos recursos, bem como controlar a sua evolução. A estimativa dos custos do projecto, também designada por orçamento, deve levar em consideração o custo dos recursos a usar, mas também a duração das actividades a executar. A gestão do custo divide-se em quatro processos:

- Planeamento dos recursos: Procura determinar quais os recursos (pessoas, equipamentos e materiais), e em que quantidade, serão usados para executar as actividades do projecto [PMI, 2004].

- Estimativa dos custos: São desenvolvidas estimativas dos custos, tendo em conta os recursos necessários para executar as actividades do projecto [PMI, 2004].

- Orçamentação dos custos: As estimativas de custos globais são repartidas pelos pacotes individuais de trabalho a realizar [PMI, 2004].

- Controlo dos custos: Este processo tem como principal função controlar as alterações ao orçamento atribuído ao projecto. Os desvios detectados nos

custos de um projecto são, normalmente, imputados a elementos externos aos processos de custos, ficando normalmente a dever-se a lacunas na definição do âmbito, interpretações erróneas do trabalho a ser realizado, um calendário pouco rigoroso ou muito optimista, ou ainda devido à deficiente avaliação e quantificação dos riscos [Stepanek, 2005].

2.8.3. Gestão do Tempo

A gestão do tempo do projecto inclui todos os processos que contribuem para que o projecto seja implementado no calendário pré-estabelecido. Os processos envolvidos na gestão do tempo são:

- Definição das actividades: Identifica as actividades que devem ser realizadas para produzir os resultados esperados do projecto [PMI, 2004];

- Sequenciamento das actividades: Identifica e documenta as relações de dependência entre as actividades [PMI, 2004];

- Estimativa da duração das actividades: Estima os períodos de trabalho necessários para executar cada uma das actividades [PMI, 2004];

- Desenvolvimento do calendário: Analisa a sequência de actividades, a sua duração, e os recursos necessários, com vista a criar um calendário de execução do projecto [PMI, 2004];

- Controlo do calendário: Controla as alterações efectuadas ao calendário do projecto [PMI, 2004].

A gestão do tempo também é considerada como uma das razões mais importantes para o nascimento de conflitos entre os interessados no projecto (os denominados Stakeholders), com uma tendência para aumentar à medida que progride o projecto [Stepanek, 2005].

2.8.4. Gestão da Qualidade

Na gestão da qualidade do projecto incluem-se os processos dedicados a garantir que o projecto satisfaz cabalmente as necessidades para as quais foi empreendido. Os principais processos de gestão da qualidade do projecto são:

- Planeamento da qualidade: Identifica os padrões de qualidade relevantes para o projecto e encontra a forma de satisfazê-los [PMI, 2004];

- Garantia da qualidade: Avalia de forma periódica o desempenho geral do projecto para assegurar que são satisfeitos os padrões de qualidade [PMI, 2004];

- Controlo da qualidade: Monitoriza os resultados específicos do projecto para determinar se estão de acordo com os padrões de qualidade considerados relevantes e identifica as formas de eliminar as causas dos desempenhos insatisfatórios identificados [PMI, 2004].

Na gestão da qualidade, o principal interesse consiste em decidir que processos usar e adaptar ao projecto. Durante a execução do projecto, são levadas a cabo diversas actividades que têm como objectivo a garantia da qualidade, certificando-se de que o processo adoptado é seguido e, é o mais adequado.
Os processos do controlo da qualidade têm como missão verificar se os produtos possuem defeitos [Evans, 2004].
A qualidade de um produto é muito influenciada a pela qualidade dos processos organizacionais usados para implementá-la e mantê-la. Esta premissa implica que haja uma preocupação com os processos de desenvolvimento e, bem assim, com a qualidade do produto [Forsberg et al., 2005].

2.8.5. Gestão dos Riscos

A gestão dos riscos fornece mecanismos que permite evitá-los, reduzir a probabilidade da sua ocorrência e, por conseguinte, minimizar as suas consequências. A gestão de riscos é uma forma organizada de identificar e medir

os riscos e de desenvolver, aplicar e gerir as possíveis opções para controlá-los [Heldman, 2005]. A gestão do risco é composta por cinco processos:

- Identificação dos riscos: São determinados os riscos mais prováveis para o projecto e documentadas as características de cada um [PMI, 2004];

- Análise qualitativa dos riscos: Os riscos são analisados qualitativamente e estabelecidas as condições para a definição de prioridades [PMI, 2004];

- Análise quantitativa dos riscos: É medida a probabilidade e o impacto dos riscos, bem como as implicações que têm nos objectivos do projecto [PMI, 2004];

- Planeamento da resposta aos riscos: São desenvolvidos procedimentos e técnicas que favoreçam as oportunidades e reduzam as ameaças dos riscos [PMI, 2004];

- Controlo e monitorização dos riscos: São monitorizados os riscos residuais, identificados novos riscos, executados os planos de redução do risco e avaliada a sua eficácia, isto durante todo o ciclo de vida do projecto [PMI, 2004].

A gestão de riscos oferece a oportunidade de compreender melhor a natureza do projecto, envolvendo todos os membros da equipa de projecto na identificação e resposta aos potenciais riscos [Heldman, 2005].

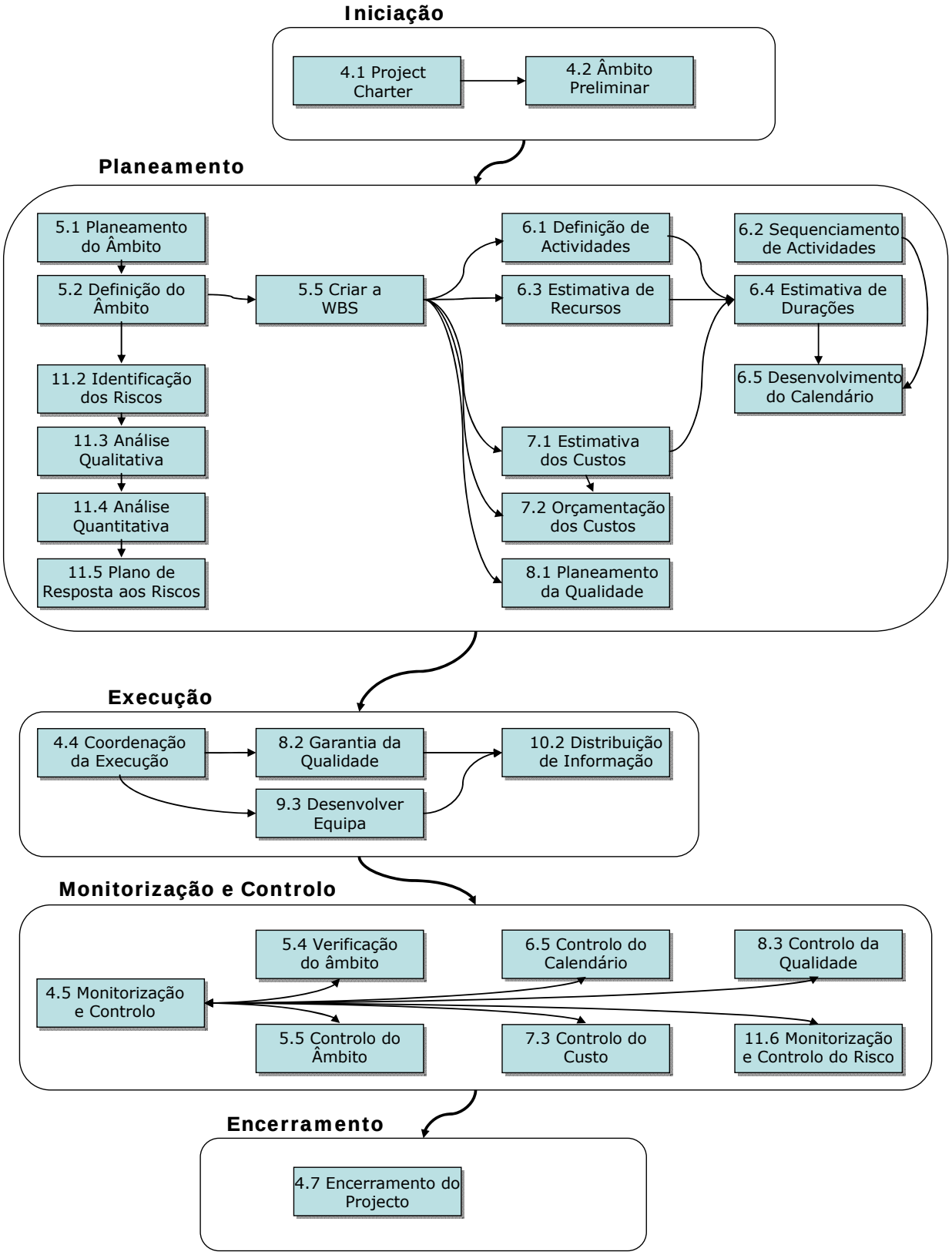

Figura 5: Grupos de Processos e o Âmbito, Custo, Tempo, Qualidade e Risco (Adaptado de [PMI, 2004])

2.8.6. Interacção do Âmbito, Custo, Tempo, Qualidade e Risco

Os processos relativos ao âmbito, custo, tempo, qualidade e risco interagem mutuamente, bem como, com os processos das restantes áreas de conhecimento. Cada processo pode envolver o esforço de um ou mais indivíduos ou grupos de indivíduos, dependendo das necessidades do projecto. Cada processo ocorre, geralmente, pelo menos uma vez em cada fase do projecto. A Figura 5 descreve os relacionamentos entre os processos envolvidos na gestão e execução de projectos e as áreas de conhecimento alvo de estudo.

2.9. Problemas com os Projectos de Software

Os projectos de software acumularam, ao longo do tempo, a reputação de se atrasarem sempre, extrapolarem o uso dos recursos e excederem o orçamento definido, entre outros contratempos [Ewusi-Mensah, 2003].

A motivação para a criação de numerosas abordagens para a gestão de projectos de software deve-se à diversidade da natureza dos projectos de software; muita desta diversidade é directamente atribuível à complexidade do ambiente em que as equipas de projecto desenvolvem a sua actividade [Armour, 2004].

A complexidade é uma característica permanente de projectos de software, ao contrário da complexidade dos sistemas naturais – nenhum aspecto da sua estrutura pode ser inferido de qualquer outra [Ewusi-Mensah, 2003]. O sucesso de um projecto de software obriga, por conseguinte, que seja considerada a complexidade do software e as suas principais características [Armour, 2004].

2.10. Processos de Software

O conceito de processo está no centro da engenharia de software. O termo processo aplica-se a um método particular de fazer alguma coisa, geralmente envolvendo um certo número de actividades [Armour, 2004].

Há múltiplas actividades que têm de ser executadas num projecto de software, se bem que, qualquer projecto inclui obrigatoriamente actividades de desenvolvimento e actividades de gestão; a soma destes dois tipos de actividades constitui o processo de desenvolvimento do software [PMI, 2004].

Um processo de software especifica um método para desenvolver software [Armour, 2004].

Cada projecto de software inicia-se com as necessidades do cliente e conclui com um software que pretende satisfaze-las. O processo de desenvolvimento do software, deverá especificar um conjunto de actividades a executar, para que a partir das necessidades do cliente seja possível obter o produto desejado; a cada projecto corresponde forçosamente um processo, podendo, no entanto, existir múltiplos projectos executados com base num mesmo processo (ver Figura 6).

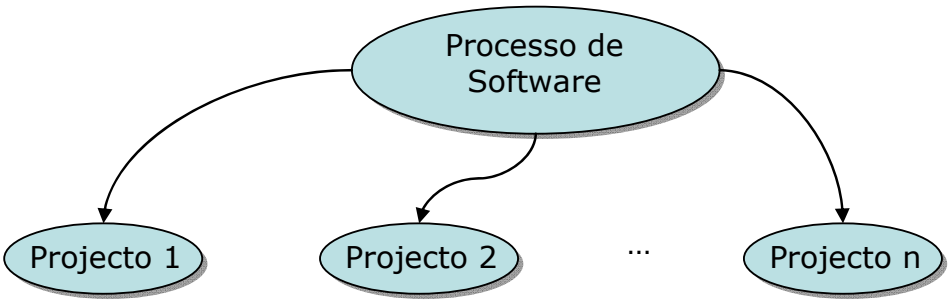

Figura 6: Processos e Projectos

O desenvolvimento de software com qualidade requer uma definição do processo, que indique as actividades a executar e os resultados esperados para cada uma delas. O processo também deverá especificar a estruturação das actividades (sequenciadas, concorrentes ou paralelas) [Armour, 2004].

Além de ser uma das fontes de conhecimento mais importantes das organizações, os processos de software também facilitam a comunicação e o melhoramento do próprio processo [McConnell, 1996].

A escolha do processo afecta a estruturação da equipa, quando e como interagem as pessoas, bem como a atribuição de responsabilidades sobre as diversas vertentes do projecto. Na escolha do processo é considerada a natureza do projecto, os métodos e as ferramentas a usar, os controlos e os produtos a desenvolver. Esta escolha é importante porque proporciona uma base para dar resposta às questões mais críticas sobre o planeamento, delegar responsabilidades e, bem assim, estimar e rastrear os resultados obtidos [Bechtold, 1999].

Assim, podemos concluir que somente com um processo definido pode haver planeamento e controlo [Stepanek, 2005].

Os processos de desenvolvimento de software podem ser definidos com diferentes níveis de abstracção [SWEBOK, 2004]. O desenvolvimento de software pode ser representado através de modelos de ciclo de vida (representações abstractas do processo de software), que contém informações gerais sobre a sua execução [SWEBOK, 2004].

Em geral, as actividades de desenvolvimento de software são organizadas sob a forma de fases, às quais são associados métodos, ferramentas e procedimentos, indispensáveis à sua execução [Armour , 2004].

Os modelos de ciclo de vida estabelecem uma representação de alto-nível das fases de desenvolvimento, apresentando somente as actividades de alto-nível e as suas inter-relações. Os modelos mais comuns são o desenvolvimento em cascata, do qual existem várias variantes, os modelos incremental e evolutivo, em espiral e ainda o RUP [McConnell, 1996].

Além de seleccionar um modelo de ciclo de vida, é preciso identificar uma, ou duas, das melhores metodologias a usar. A metodologia é seleccionada em simultâneo ou depois de escolhido o ciclo de vida [Bechtold, 1999]. A metodologia de software, tipicamente, identifica as principais actividades a serem executadas (análise, concepção, codificação, testes) e indica quais as pessoas (utilizadores, gestores, programadores, etc.) a envolver em cada actividade e os papéis a desempenhar [Evans, 2004].

As metodologias descrevem as condições a satisfazer para que o projecto transite para a fase seguinte e em que momentos deverão ser verificadas as condições [McConnell, 1996].

2.11. Ciclo de Vida do Software

Os grandes objectivos do ciclo de vida do software são a definição das actividades a executar no âmbito do projecto, introduzir consistência entre os vários projectos de software da organização e estabelecer pontos de controlo que facilitem a gestão e a tomada de decisões [Marchewka, 2003]; isto não diminui a dificuldade da tomada de decisões, a ponderação de alternativas, as guerras de interesses, as negociações com os utilizadores, o apoio à equipa, ou qualquer outra dificuldade que um gestor de projecto normalmente tem de enfrentar [Taylor, 2004].

O ciclo de vida do software inicia-se quando o produto é concebido e termina quando é colocado em produção, incluindo todas as actividades de desenvolvimento [Marchewka, 2003].

Os projectos de software possuem em regra um ciclo de vida com cinco fases principais (1) definição dos requisitos do cliente, (2) definição de requisitos do software, (3) concepção da arquitectura do software, (4) codificação e testes, (5) colocação em produtivo [Taylor, 2004; Forsberg et al., 2005].

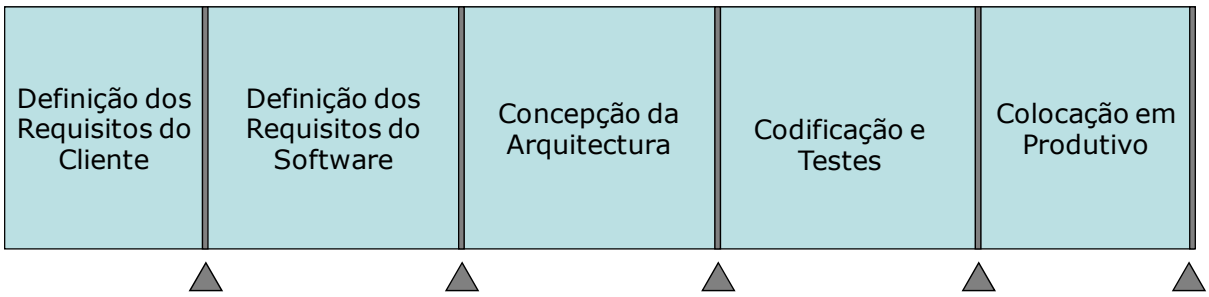

Figura 7: Modelo de Ciclo de Vida do Software
(Adaptado de [Forsberg et al., 2005])

Cada fase termina com uma revisão do trabalho realizado [Forsberg et al., 2005].

Na Figura 7, os triângulos assinalam as fronteiras entre as diversas fases do ciclo de vida do software. A primeira fronteira é ultrapassada com a criação de um documento com os requisitos do cliente. Após a aprovação desse documento, serão necessárias mais quatro fases para que o produto seja colocado em operação [Forsberg et al., 2005].

Os subprodutos entregues em cada fase são sujeitos a uma revisão, e somente depois de aprovados, podem prosseguir para a próxima fase. Assim que o software é colocado em operação, dá-se início às actividades de encerramento do projecto [Forsberg et al., 2005]. Este evento marca o fim do ciclo de vida do projecto de software.

Os pontos de fronteira constituem o mínimo necessário para assegurar uma relação contratual com o cliente, porém, ao longo do projecto devem ser adicionados novos pontos, para medir com mais exactidão o progresso do projecto.

Em seguida são descritas as principais actividades e resultados esperados em cada uma das fases referidas.

2.11.1. Requisitos do Cliente

É o momento em que é definida a visão do projecto. Os requisitos do cliente são identificados e documentados, mediante entrevistas ou questionários, gerando assim um documento com a especificação dos requisitos e o caderno de encargos [Hamilton, 1999].

2.11.2. Requisitos do Software

Esta é a fase de análise do projecto de software. Uma parte vital desta fase é a construção de um modelo que descreva a acção do software, do qual resulta um documento com os requisitos do software [Taylor, 2004].

Ainda durante esta fase, deve ser elaborado um plano preliminar de gestão do projecto e estimado o seu custo [Wysocki e McGary, 2003]. São levadas a cabo várias iterações para que todas as dificuldades técnicas ou os elementos mais críticos possam ser identificados e o modelo do software possa ser apurado.

2.11.3. Concepção da Arquitectura

O modelo construído na fase anterior é o ponto de partida para a definição da estrutura do software – a concepção da arquitectura – que dispõe as funções e os componentes do sistema (ou subsistemas) e define o fluxo e o controlo de dados [Taylor, 2004].

O produto desta fase é um documento com a arquitectura do software [Wysocki e McGary, 2003].

2.11.4. Codificação e Testes

O objectivo desta fase é detalhar a concepção, codificar, documentar e testar o software [Taylor, 2004]. Contudo à medida que os testes decorrem, também é necessário verificar a qualidade do produto, mediante a condução de testes apropriados [Evans, 2004].

São três os produtos desta fase: o código do software e a documentação com o detalhe do projecto e os resultados dos testes efectuados [PMI, 2004].

2.11.5. Colocação em Produtivo

A finalidade desta fase é esclarecer se o software cumpre com as exigências do documento de requisitos do cliente, o que se consegue com a instalação do software no ambiente produtivo e subsequente aplicação de testes de aceitação [Taylor, 2004]. Só quando o software tiver passado por todos os testes de aceitação, poderá então finalmente ser objecto de aceitação formal pelo cliente [Taylor, 2004].

2.12. Modelos do Ciclo de Vida do Software

Os distintos problemas, pessoas ou organizações, precisam de diferentes modelos de processo de produção de software, pois a cada modelo correspondem diferentes estratégias de ordenação das actividades e mecanismos próprios de gestão e controlo dos processos. A abordagem adoptada deve ser definida para cada projecto, com os devidos reflexos no plano de gestão do projecto. O modelo de ciclo de vida do software apresentado na Figura 7 resume as fases e actividades que devem existir em qualquer projecto de software.

A secção seguinte expõe alguns modelos de ciclo de vida do software, com vista a conhecerem-se as principais variantes dos processos de desenvolvimento de software.

2.12.1. Modelo em Cascata

O modelo em cascata puro, muitas das vezes designado por ciclo de vida clássico, segue uma abordagem sequencial no desenvolvimento do software, começando pelos requisitos do cliente e avançando para a análise dos requisitos do software, concepção da arquitectura, codificação e teste [McConnel, 1996] (ver Figura 8).

Neste modelo, as fases são executadas sequencialmente, e apesar de ser permitido, o retorno às fases anteriores para a eventual correcção de erros, nem sempre é fácil ou desejável [Marchewka, 2003].

A entrega do software final ocorre num único ponto, no final da fase de codificação e testes.

O modelo sequencial linear é o modelo de ciclo de vida mais antigo e mais amplamente usado na indústria de software. Porém a sua aplicação levanta alguns problemas, designadamente:

- Na prática, muitos dos projectos de software não se enquadram no fluxo sequencial de actividades proposto pelo modelo [Larman, 2003];

- Apesar de poder ser difícil para o cliente expor os requisitos explicitamente de uma única vez, para o modelo sequencial linear isso é um requisito, o que dificulta a gestão da incerteza inicial, existente na maioria dos projectos de software [Koch, 2005];

- O cliente somente terá uma versão operacional do software no final do projecto [McConnel, 1996];

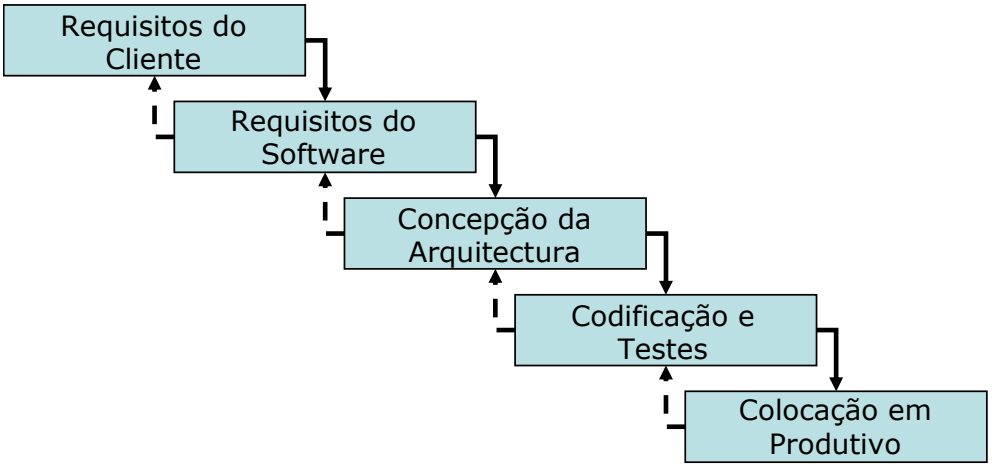

Figura 8: Modelo do Ciclo de Vida em Cascata
(Adaptado de McConnel [1996])

Apesar destes problemas, o modelo de ciclo de vida clássico tem um papel indispensável na engenharia de software, pois, pese embora as suas fraquezas, é claramente melhor do que uma abordagem casual [McConnel, 1996].
Na prática, para problemas pequenos e bem definidos, onde os analistas/programadores conhecem e dominam o problema e quando os requisitos podem ser especificados de forma clara no inicio do projecto, a

adopção deste modelo de ciclo de vida deve ter prioridade sobre os restantes, uma vez que é o mais fácil de gerir [Marchewka, 2003]. Mas, mesmo nos casos em que as vantagens do modelo cascata são superiores às desvantagens, a utilização de algumas variantes do modelo cascata, poderá ser mais vantajosa [McConnel, 1996].

As actividades identificadas num modelo cascata puro (descritas anteriormente) são intrínsecas ao software e não podem ser evitadas. Muitos dos problemas que surgem com este modelo derivam do facto de as actividades serem tratadas como fases independentes entre si. É possível corrigir ou reduzir as fraquezas deste modelo, mediante a sobreposição parcial das actividades de desenvolvimento [McConnel, 1996].

As principais variantes ao modelo cascata são o modelo Sashimi, desenvolvido pelo Fuji-Xerox, o modelo cascata com sub projectos e o modelo cascata com redução do risco [McConnel, 1996].

2.12.2. Modelo Incremental

O modelo incremental deriva do modelo de ciclo de vida sequencial linear, no qual as fases de especificação de requisitos, análise e concepção da arquitectura são realizadas de forma sequencial. Porém, uma vez definida a arquitectura do software, as restantes fases (desenho detalhado, codificação e testes) são divididas em unidades mais pequenas, para que assim o software possa ser desenvolvido por versões, com funcionalidades e capacidades acrescidas, como mostra a Figura 9 [McConnel, 1996].

Este modelo de processo aplica-se quando o projecto de software está perfeitamente definido, mas não há recursos para o seu desenvolvimento de uma única vez, ou então, quando se pretende apresentar rapidamente resultados ao cliente [Larman, 2003; Koch, 2005].

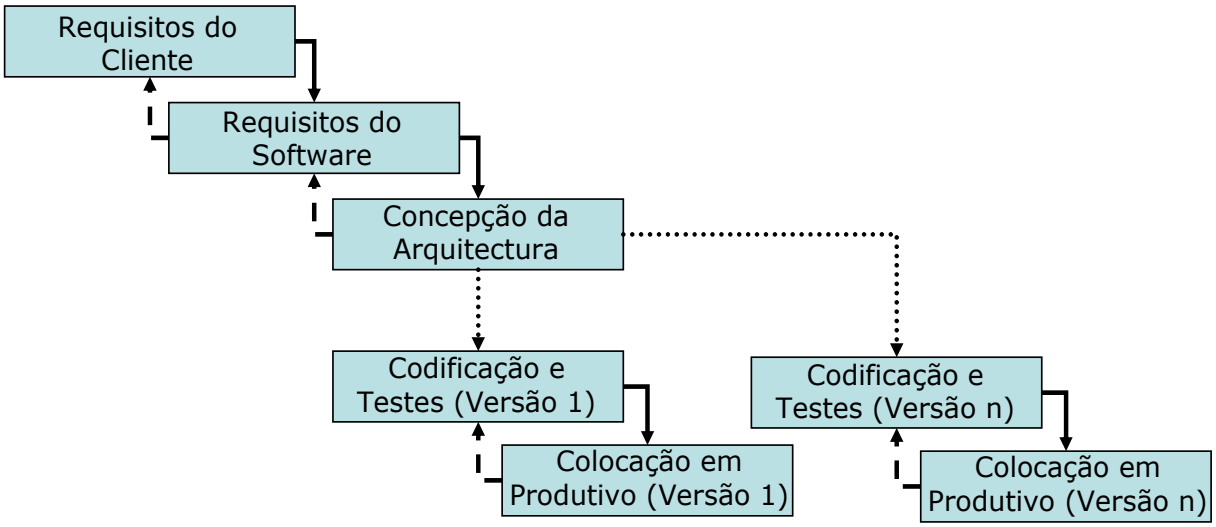

Figura 9: Modelo do Ciclo de Vida Incremental
(Adaptado de McConnel [1996])

2.12.3. Modelo Evolutivo

Os requisitos do negócio e do software mudam, cada vez mais, à medida que o desenvolvimento avança, correndo-se o risco de o produto final não corresponder às expectativas do cliente [Larman, 2003]. Os prazos, cada vez mais apertados, dificultam muito a conclusão de um produto de software com todas as funcionalidades pretendidas, sendo, no entanto, possível produzir uma versão limitada, que responda às pressões do negócio ou à concorrência [Koch, 2005].

Nestas situações, é preciso um modelo de ciclo de vida que evolua ao longo do tempo. Assim, quando o problema a solucionar não está bem definido ou não pode ser totalmente especificado no início do desenvolvimento, deve-se optar por um modelo evolutivo [McConnel, 1996; Forsberg et al., 2005].

Os modelos evolutivos aplicam sequências lineares, à medida que o calendário avança (ilustrado na Figura 10), onde de cada sequência resulta um subproduto passível de ser colocado em operação. A utilização em produtivo de uma versão limitada do software permite levantar novos requisitos, que servirão de base para o início de um novo ciclo de desenvolvimento [Bainey, 2004].

Este modelo permite que os engenheiros de software desenvolvam iterativamente versões do software sucessivamente mais completas [McConnel, 1996]. A primeira versão implementa um subproduto com os requisitos mais básicos. Este subproduto é colocado em produtivo e em função do resultado do

uso e/ou da avaliação conduzida pelo cliente, é desenvolvido um plano para a próxima iteração. A seguir à entrega dos produtos resultantes de cada iteração, o processo repete-se, até que seja obtido um produto completo [McConnel, 1996].

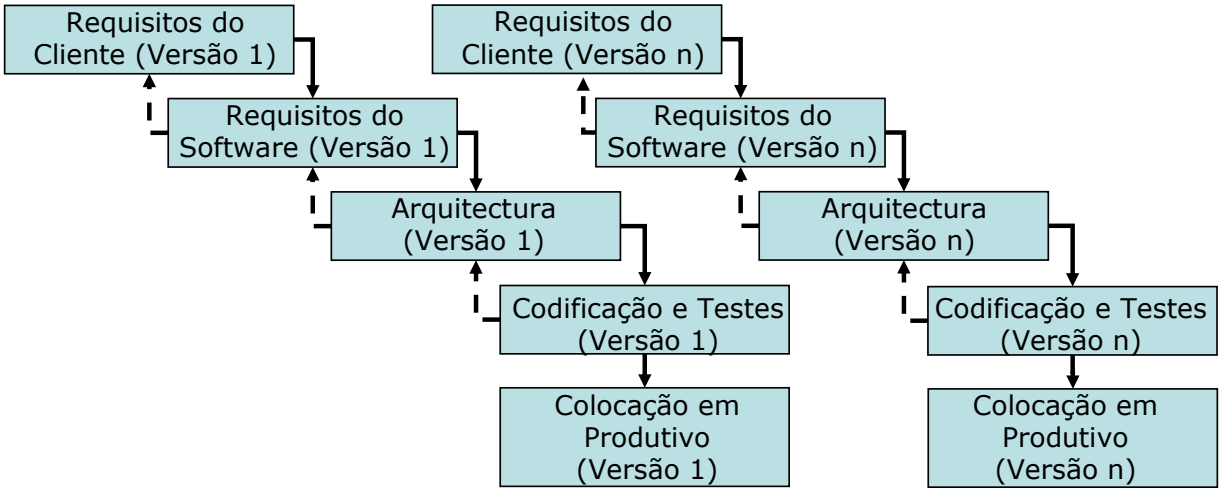

Figura 10: Modelo do Ciclo de Vida Evolutivo
(Adaptado de McConnel [1996])

2.12.4. Modelo Espiral

O modelo espiral tem como objectivo uma constante análise de riscos e uma forte interacção com os clientes. Este modelo foi proposto por Barry Bohem, em 1988 [Marchewka, 2003].

O modelo apresenta uma aproximação orientada ao risco, em que o software é subdividido em pequenos projectos, que abordam um ou mais dos principias riscos, para que os grandes riscos sejam abordados [McConnel, 1996]. Os principais problemas ou desafios colocados pelos riscos surgirão nos primeiros ciclos de desenvolvimento, pelo que o modelo em espiral possibilita a redução do custo do projecto [Marchewka, 2003].

É possível iniciar o projecto com uma série de iterações, para reduzir o risco para um nível aceitável, adoptando em seguida um modelo cascata ou outro modelo, que não esteja vocacionado para lidar com o risco.

A principal vantagem do modelo espiral é o facto de os riscos se reduzirem à medida que os custos aumentam [Bechtold, 1999]. A ideia base consiste em iniciar as actividades de desenvolvimento, em pequena escala, para facilitar a

identificação dos riscos, e depois, estabelecer um plano para eliminar ou reduzir os riscos e avaliar as possíveis alternativas [Bechtold, 1999].
A espiral inicia-se com a identificação dos requisitos dos subprodutos da primeira iteração, que são depois implementados e verificados, para então se proceder ao planeamento da próxima iteração. Assim, cada iteração é composta por cinco passos [McConnel, 1996]:

- Determinar quais os objectivos, as alternativas e as restrições do software;
- Identificar alternativas aos riscos e possíveis formas de os eliminar;
- Proceder ao desenvolvimento dos subprodutos e verificar que cumprem os requisitos;
- Planear a próxima iteração;
- Proceder a uma avaliação dos resultados com o cliente, para decidir se é necessária uma próxima iteração.

Em suma, inicialmente é conduzida uma recolha dos requisitos e efectuado o planeamento; é aplicada a análise de riscos, com base nos requisitos iniciais e avaliada a viabilidade da prossecução ou não do projecto. Se a decisão for a de prosseguir, então é desenvolvido um protótipo do software para uma primeira avaliação do cliente. A partir daqui serão desenvolvidos vários ciclos, tal como é mostrado na Figura 11, até que surja um produto que satisfaça o cliente [Bechtold, 1999].

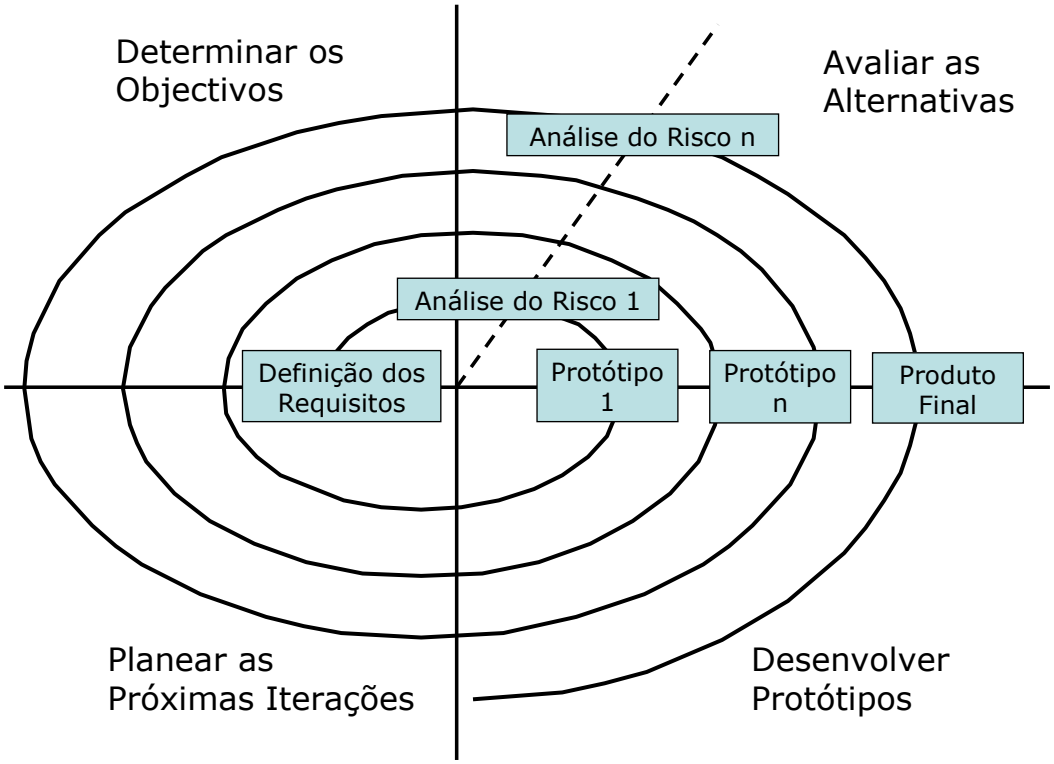

Figura 11: Modelo de Ciclo de Vida em Espiral
(Adaptado de McConnel [1996])

2.12.5. Outros Modelos

Além de todos os métodos entretanto já mencionados existem outros (Scrum, XP - Extreme Programming, Crystal, RUP, DSDM, etc.), denominados por processos ágeis, dos quais o RUP – Rational Unified Process é provavelmente o mais extensamente usado [Stepanek, 2005]. Encontra-se em milhares de organizações de desenvolvimento espalhadas pelo mundo, o que não significa que seja bem aplicado ou compreendido [Larman, 2003].

Os modelos, Crystal, XP e RUP foram, todos, criados no princípio da década de 90. O RUP, no entanto, ao contrário dos modelos Crystal e XP, tem um pedigree [Stepanek, 2005]. É uma fusão do trabalho de três dos mais proeminentes engenheiros de software dessa época: Grady Booch, Ivar Jacobson e Jim Rumbaugh. Em meados da década de 90, cada um destes especialistas tinha a sua própria metodologia de desenvolvimento de software, a sua própria notação para modelação e a sua própria empresa de software, porém, face à lenta ascensão destas metodologias, devido à falta da padronização, decidiram, em

1995, fundir as suas metodologias, notações e empresas para criar o UP – Unified Process (ou Processo Unificado), o UML – Unified Modeling Language (ou Linguagem de Modelação Unificada), e a empresa Rational Software Corporation; o RUP é versão da empresa Rational do modelo Unified Process original [Stepanek, 2005].

Um dos pontos-chave do RUP é o desenvolvimento das primeiras iterações orientado ao risco, focando-se na criação do núcleo da arquitectura e reduzindo os riscos mais elevados. O RUP inclui também a definição de produtos de trabalho, tais como a visão, a arquitectura do software e uma lista de riscos [Larman, 2003].

Além destes, existem outros modelos, menos conhecidos, que incluem: o modelo Microsoft Solutions Framework, que descreve as melhores práticas usadas pela Microsoft; o modelo OPEN de Henderson-Sellers, Firesmith e Graham; e os modelos WinWin Spiral e MBASE Spiral de Barry Bohem [Larman, 2003].

2.13. Conclusões do Capítulo

Este capítulo apresentou os princípios básicos da gestão de projectos, e em particular dos projectos de software. Seguindo as orientações das boas-práticas recomendadas por algumas organizações internacionais, mais concretamente, o Project Management Institute, foram apresentados, de forma breve, os processos envolvidos na condução de um projecto e, bem assim, algumas das áreas de conhecimento mais críticas para o sucesso dos projectos de software, responsáveis pelos principais problemas de que sofrem os projectos de software na actualidade.

Os processos descritos aplicam-se à gestão dos projectos, independentemente da sua natureza, existindo, no entanto, especificidades muito próprias dos projectos de software, que envolvem processos próprios, que representam os fluxos de informação ao longo do ciclo de vida do software. Nesse sentido, foram descritos vários modelos de abstracção do processo de desenvolvimento de software, tendo-se procurado apresentar uma amostra representativa dos modelos usados pela maioria das organizações produtoras de software, apesar de que, muitas destas organizações adaptam estes modelos à sua realidade.

No próximo capítulo é analisada a forma como o desdobramento da função qualidade – QFD, pode contribuir para melhorar os processos de gestão de projectos de software, mas sobretudo o ciclo de desenvolvimento do software, ao melhorar a gestão das áreas de conhecimento mais críticas, permitindo, deste modo, ultrapassar parte dos problemas do software identificados no primeiro capítulo.

CAPÍTULO 3 – Desdobramento da Função Qualidade

Este capítulo inclui uma breve introdução ao QFD, começando por efectuar uma descrição das suas principais características e apresentar um pouco da sua história – é efectuada uma retrospectiva do QFD, que inclui as suas diversas aplicações, e algumas das evoluções mais recentes desta metodologia. Em seguida procura-se dissecar os elementos constituintes do QFD, com o objectivo de conhecer o seu funcionamento, mas principalmente, como é que estes elementos poderão ser introduzidos no ciclo de vida de um projecto de software, para melhorarem a definição do âmbito e a qualidade do produto. Mas não só, pois esta é apenas uma das perspectivas do QFD. O QFD é usado, também, para identificar os processos, e respectivas actividades, que suportam o modelo de ciclo de vida de desenvolvimento do software, bem como as actividades de suporte ao ciclo de vida do projecto.

3.1. Evolução e Estrutura do QFD

O desdobramento da função qualidade foi concebido no Japão, na década de 1960, numa era em que as industrias Japonesas abandonaram o desenvolvimento de produtos, por meio da imitação e cópia, e procuraram o desenvolvimento de produtos com base na originalidade [Akao, 1997].

Após a Segunda Guerra Mundial, o controlo estatístico da qualidade (SQC), foi aplicado de forma intensiva na indústria química do Japão, porém, os processos de garantia da qualidade preocupavam-se apenas com as operações de produção, porque, uma vez estabelecidas as operações de produção, a qualidade dos produtos resultantes era assegurada pelo SQC [Yamamoto et al., 2005].

Em 1961, o movimento para o controlo da qualidade foi reforçado pela publicação do controlo total da qualidade (TQC). Como resultado, no período que decorreu entre 1960 e 1965, o SQC transformou-se no TQC [Akao, 1997].

A qualidade passou a envolver, a concepção da própria qualidade, a identificação do mercado alvo, a concepção do processo de produção, os recursos, a produção propriamente dita, a inspecção e, finalmente, as vendas esperadas [Akao, 1997].

As primeiras tentativas do desdobramento da qualidade (QD) foram iniciadas por Akao, a partir de 1966, motivado pelas dificuldades na determinação da qualidade dos produtos, acrescidas pelo facto de as linhas de produção não receberem informação sobre os pontos mais críticos para a qualidade do produto [Akao, 1997].

Entre 1966 e 1972, Akao realizou várias pesquisas em conjunto com diversas organizações Japonesas, das quais resultou uma publicação, com os conceitos básicos do desdobramento da qualidade (QD), mas cujas fases ainda não eram suficientemente claras sobre a forma de garantir a qualidade dos produtos [Akao, 1997].

Em 1972, os estaleiros da Mitsubishi Heavy Industries elaboram a Matriz da Qualidade, que unia o desdobramento da função qualidade restrito (QFDr), definido por Shigeru Mizuno, com o desdobramento da qualidade (QD), proposto por Akao [Akao, 1997].

Em 1978, Akao e Shigeru publicaram o primeiro livro sobre o QFD, estabelecendo um framework teórico [Jiang et al., 2007]. Apesar de existirem actualmente muitos livros e artigos sobre como usar o QFD, a publicação de matrizes, com exemplos práticos, é escassa. Algumas das organizações mais notáveis dos Estados Unidos que usam o QFD, incluem a Ford Motor Company, Procter & Gamble, 3M Corporation, AT&T, Hewlett Packard e a Digital Equipment Corporation [Cohen, 1995].

Nos EUA, o QFD começou a ser divulgado em 1983, por Furukawa, Kogure e Akao, durante um seminário. Neste mesmo ano, foi publicado o artigo "Quality Function Deployment and CWQC in Japan", por Kogure e Akao [Xie et al., 2003].

Em 1984, na consequência da troca de informações entre Clausing e Makabe, empregados da Xerox, realizaram-se vários seminários e iniciativas de divulgação do QFD, conduzidas pelo próprio Clausing e Bob King, entre outros [Cohen, 1995].

A partir daqui o QFD seguiu dois caminhos diferentes [Cohen, 1995]:

- No Japão, seguiu a concepção dos professores Akao e Mizuno;

- Nos EUA, nasceram duas novas versões do QFD: na primeira, de Don Clausing, o QFD é composto por quatro matrizes aplicáveis a qualquer produto; na segunda, de Bob King, o QFD é composto por uma multiplicidade de matrizes, aplicáveis em função da natureza do produto.

As primeiras aplicações do QFD procuravam que os processos de produção gerassem produtos com a qualidade desejada [Akao, 1990]. Com o aumento da proficiência com o QFD, a atenção virou-se para a melhoria dos processos mais a montante, a fim de melhorar a qualidade da concepção do produto. Nos anos mais recentes, o QFD continuou este movimento para montante, procurando melhorar a qualidade da forma como são identificados os requisitos que condicionam a concepção do produto [Akao, 1990].

O QFD tem sido aplicado a produtos tangíveis ou intangíveis, tais como serviços, software, melhoramento de processos e planeamento estratégico [Cohen, 1995; Stylianou et al., 1997]. O QFD poderá ser aplicado apenas a fases específicas dos processos de produção, como a identificação de oportunidades, até à definição dos requisitos da produção [Akao, 1990; Cohen, 1995].

Nos EUA, predomina o uso da matriz da qualidade, em detrimento das matrizes que lidam com os requisitos do processo de fabricação; o QFD é usado na fase inicial do desenvolvimento, na transformação da voz do cliente em atributos do produto [Cohen, 1995]. No Japão, o QFD é usado em toda a sua extensão, desde a análise dos requisitos dos clientes e do produto, até aos requisitos do processo [Akao, 1990].

O QFD é um sistema capaz de traduzir as necessidades dos clientes em requisitos técnicos apropriados, permitindo incorporar as suas necessidades e expectativas no produto – coloca os clientes em posição de destaque.

Este sistema é aplicado a cada fase do ciclo de vida do produto, desde a pesquisa inicial e concepção do produto, passando pela produção, marketing e vendas. Assegura, de forma sistemática que as especificações do produto, bem como a

selecção de tecnologias e processos sejam orientados para as necessidades do cliente [Bossert, 1991]; converte os requisitos dos clientes em características de qualidade do produto, desdobrando de forma sistemática as relações entre os requisitos e as características de qualidade do produto [Akao, 1990]; o próprio QFD constitui um processo catalisador, pois fomenta a cooperação das equipas, transformando-se num mecanismo de comunicação [Tran e Sherif, 1995].

O QFD encurta o ciclo de vida de desenvolvimento do produto, com menos alterações e o consequente aumento da produtividade e redução dos custos [Fehlmann, 1998].

Esta evolução do controlo da qualidade (qualidade reactiva) para a garantia da qualidade (qualidade activa), procura incorporar no produto, desde o primeiro momento até à conclusão do projecto, a qualidade que os clientes desejam [Ouyang et al., 1997].

Os principais conceitos que constituem a base do QFD são [Cohen, 1995]:

- Perguntar aos clientes o que querem, para que lhes seja dada a possibilidade de se expressarem, isto é, entender como é que os clientes definem e percebem os produtos que pretendem;

- Utilizar a experiência e o conhecimento da equipa de projecto para identificar as características, mensuráveis, que dão resposta às necessidades e desejos dos clientes;

- Estabelecer prioridades, de forma a concentrar os esforços nas características mais importantes;

- Promover constantemente a inovação, considerando as expectativas dos clientes e as acções dos concorrentes, de tal forma que o produto possa ser lucrativo durante todo o seu ciclo de vida.

3.2. Unidades Elementares do QFD

O QFD utiliza tabelas, matrizes e modelos, como unidades elementares [Albuquerque et al., 2003]. As tabelas contêm dados correlacionados mediante matrizes; as interacções entre as matrizes são demonstradas por um modelo [Akao, 1990; Herzwurm et al., 2002].

O processo QFD inicia-se com o preenchimento de diversas tabelas, com base nos dados fornecidos pelos clientes, e bem assim, elementos recolhidos em pesquisas de mercado, ou resultantes da própria experiência da organização [Stylianou et al., 1997].

Os dados são agrupados em estruturas hierárquicas, para que, as características, funções ou requisitos, inicialmente pouco explícitos, se tornem progressivamente mais evidentes [Cohen, 1995].

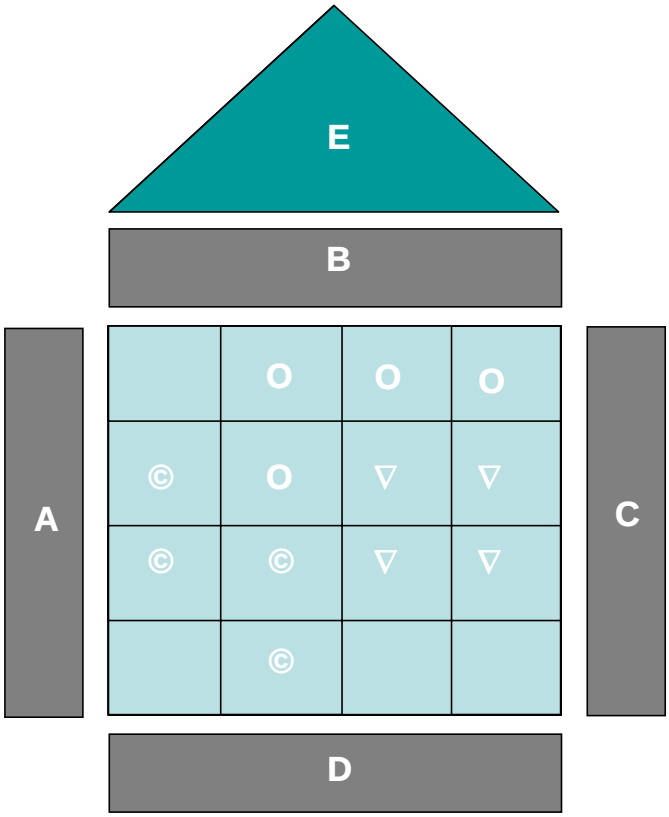

Figura 12: Matriz Base do QFD (Casa da Qualidade)

As matrizes pretendem visualizar de forma rápida, a correlação entre os dados das tabelas, e ao mesmo tempo, servem de repositório de dados.

Na Figura 12, a matriz central contém as células onde é representada a correlação entre cada um dos elementos das tabelas A e B. Com recurso a números ou símbolos, é indicado o grau de correlação entre os elementos das tabelas. A tabela C contém a importância relativa de cada elemento da tabela A. Na tabela D, avalia-se o grau de importância dos elementos da tabela B, derivado da importância atribuída aos elementos da tabela A (registados na tabela C) e das correlações inscritas nas células da matriz central. Os elementos da tabela B poderão, ainda, ser correlacionados, entre si, na matriz E [Cohen, 1995].

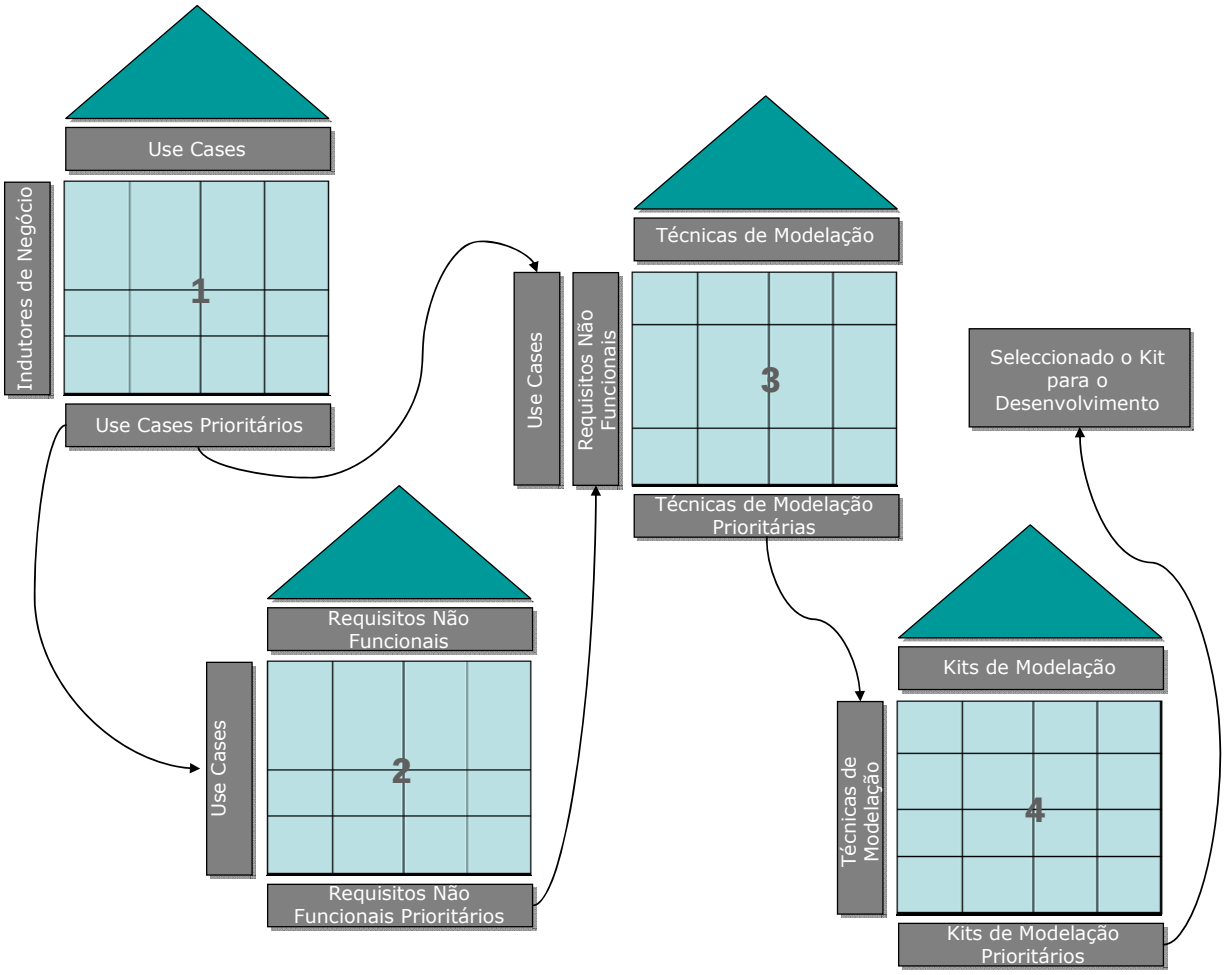

Figura 13: Exemplo de um Modelo QFD
(Adaptado de [Denney, 2005])

O modelo é uma estrutura de matrizes, que permite visualizar o percurso seguido, para desdobrar sucessivamente as características do produto [Jagdev et

al., 1997]. A sequência de matrizes permite visualizar a existência de uma relação causal entre as características do produto final, os seus componentes, as suas funções, os custos, recursos e os processos [Albuquerque et al., 2003].
É, deste modo, registado de forma visível e detalhada, todo o percurso do produto e dos processos (a Figura 13 apresenta um exemplo de um modelo).
O modelo pode ser abordado na perspectiva da qualidade dos processos e recursos envolvidos na elaboração do produto, mas analisando, também, os aspectos tecnológicos e dos custos [Krogstie, 1999]. Os aspectos a serem abordados no modelo dependem no entanto dos projectos e da natureza do produto a ser desenvolvido.

3.3. QD e QFD (restrito)

O desenvolvimento da metodologia do QFD resulta da integração de vários conceitos, tais como a qualidade, a função qualidade, o desdobramento da qualidade e o desdobramento da função qualidade [Carpinetti e Peixoto, 1998]. Estes conceitos evoluíram ao longo do tempo e, juntos, formaram o conceito actual do QFD, a saber:

- Qualidade: A ISO 9001:2000 define a qualidade como a "… medida em que um conjunto de características inerentes possui a capacidade de satisfazer as necessidades ou expectativas, expressas de forma explícita ou implícita …" [Hoyle, 2005];

- Função Qualidade: Refere-se a qualquer área da organização que influencie a qualidade do produto [Akao, 1990; Akao, 1997];

- Desdobramento da Qualidade – QD: Engloba todas as actividades a executar pelas diversas áreas para assegurar a qualidade do produto [Akao, 1997; Carpinetti e Peixoto, 1998; Yacuzzi e Martín, 2003];

- Desdobramento da Função Qualidade (restrito) – QFDr: Consiste na descrição detalhada dos recursos e das actividades a usar em cada fase do

ciclo do produto [Akao, 1997; Carpinetti e Peixoto, 1998; Yacuzzi e Martín, 2003];

- Desdobramento da Função Qualidade (amplo) – QFD: Resulta da composição do Desdobramento da Qualidade e do Desdobramento da Função Qualidade (sentido restrito) [Akao, 1997; Carpinetti e Peixoto, 1998; Yacuzzi e Martín, 2003] (ver Figura 14).

O QFD refere-se à combinação do desdobramento da qualidade e do QFD em sentido restrito. Essa definição é representada na Figura 14.

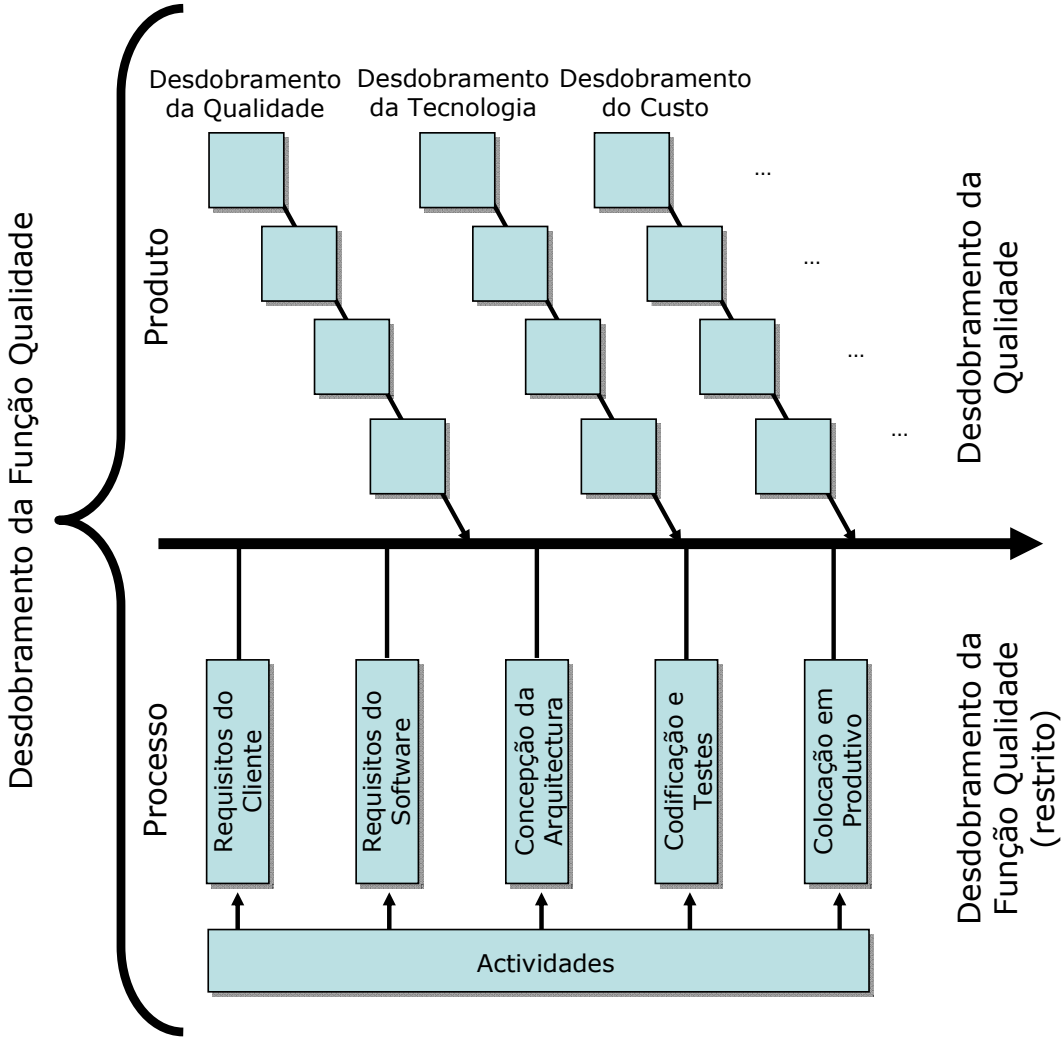

Figura 14: Componentes Conceptuais do QFD para Software
(Adaptado de [Dimsey e Mazur, 2002])

O QD procura recolher informações que permitam o desenvolvimento do produto, começando pelo levantamento das necessidades dos clientes – a voz do cliente [Akao, 1990; Akao 1997]. A partir da determinação da voz do cliente, são estabelecidas as funções do produto, os processos de desenvolvimento e, definidos os valores que permitirão controlar e avaliar a performance do produto e dos processos.

O QD abarca a definição de metas e os principais pontos de garantia da qualidade, isto é, a definição de metas a atingir, e em que pontos, por cada elemento que compõe as matrizes, sendo usado em todos os pontos do ciclo de vida do produto. O propósito do desdobramento da qualidade é, portanto, estabelecer uma rede de qualidade que possa assegurar a qualidade do produto.

O QFD restrito é derivado da engenharia do valor, originalmente aplicada na definição de funções do produto, para efectuar o desdobramento das funções dos processos de negócio, porquanto, a palavra "função" refere-se às funções a usar na execução do produto, e não às funções do produto em si [Jiang et al., 2007].

Akao e Mizuno adicionaram o (F) à sigla QD, para referir o uso de técnicas de engenharia do valor e a análise de funções, que visam a sistematização e a melhoraria do produto, bem como as funções operacionais da organização [Akao e Mazur, 1998].

O propósito do QFD restrito é estabelecer uma rede de procedimentos, formada pelas várias actividades conduzidas para alcançar um produto com qualidade [Jiang et al., 2007]. Esta rede inclui as actividades executadas em todas as fases do processo de desenvolvimento, desde o levantamento dos requisitos do software, concepção da arquitectura, codificação e testes, colocação em produtivo.

O QFDr refere-se assim ao desdobramento do trabalho a realizar, isto é, às operações a executar que assegurem a qualidade dos produtos, incluindo o controlo de cada função operativa; o QFDr concentra-se nos processos da organização que asseguram que as operações e tarefas realizadas contribuem para a qualidade [Yacuzzi e Martín, 2003]; não tem por objectivo o sequenciamento das actividades de desenvolvimento, mas a especificação de quem fará o trabalho e como será realizado; são identificadas as actividades

intervenientes no processo de desenvolvimento do produto, a que nível e em que momento.

Apesar de o QFDr ser uma parte essencial do QFD, para uma implementação adequada, é muito frequente que o termo QFD seja usado nas referências ao QD. O QFD restrito foi completamente ignorado pela maioria das organizações fora do Japão [Jiang et al., 2007].

No entusiasmo da adopção e implementação do QFD, os Americanos, na indústria automóvel e eventualmente em quase todas as indústrias, não Japonesas, foram incapazes de diferenciar o QD do QFDr [Akao, 1997].

O modelo QFD comummente aplicado caracteriza-se pela existência de quatro fases: (1) planeamento do produto, (2) planeamento dos componentes, (3) planeamento do processo e (4) planeamento do desenvolvimento [Cohen, 1995; Carpinetti e Peixoto, 1998; Krogstie, 1999].

Na primeira fase aplica-se a chamada matriz da casa da qualidade, para transformar os requisitos dos clientes em requisitos técnicos. Os requisitos técnicos considerados mais importantes são relacionados, numa segunda matriz, com as características dos componentes do produto. É seguido um processo análogo com as restantes matrizes, isto é, os elementos mais importantes de cada matriz, são relacionados com os elementos da matriz subsequente (ver Figura 15).

O QFD promove, deste modo, o desdobramento dos requisitos/soluções em múltiplas matrizes, cujas tabelas derivam, em parte, umas das outras, num processo semelhante ao refinamento usado nas abordagens mais tradicionais (ver Capítulo 2).

Como se pode verificar, um dos aspectos mais importantes do QFD é a definição da qualidade pelos clientes, pelas suas próprias palavras, o que faz descolar este método das abordagens tradicionais, que definem a qualidade do produto e dos sistemas em geral em termos técnicos, mas que podem significar pouco ou nada para os clientes [Stylianou et al., 1997]. O QFD centra-se nas necessidades dos clientes, assegura a qualidade do projecto (um dos pontos mais importantes para o sucesso do produto) e coordena o fluxo de informações e, bem assim, organiza as actividades de cada uma das áreas funcionais, promovendo deste modo a

integração inter-funcional e a rápida resolução de problemas [Akao, 1990; Akao e Mazur, 1998].

Apesar da interligação das casas da qualidade ser o primeiro modelo QFD criado no ocidente, e ainda o mais usado, esta ferramenta foi no entanto demasiado simplificada no ocidente, deixando de fora muita informação importante do modelo QFD original [Jiang et al., 2007].

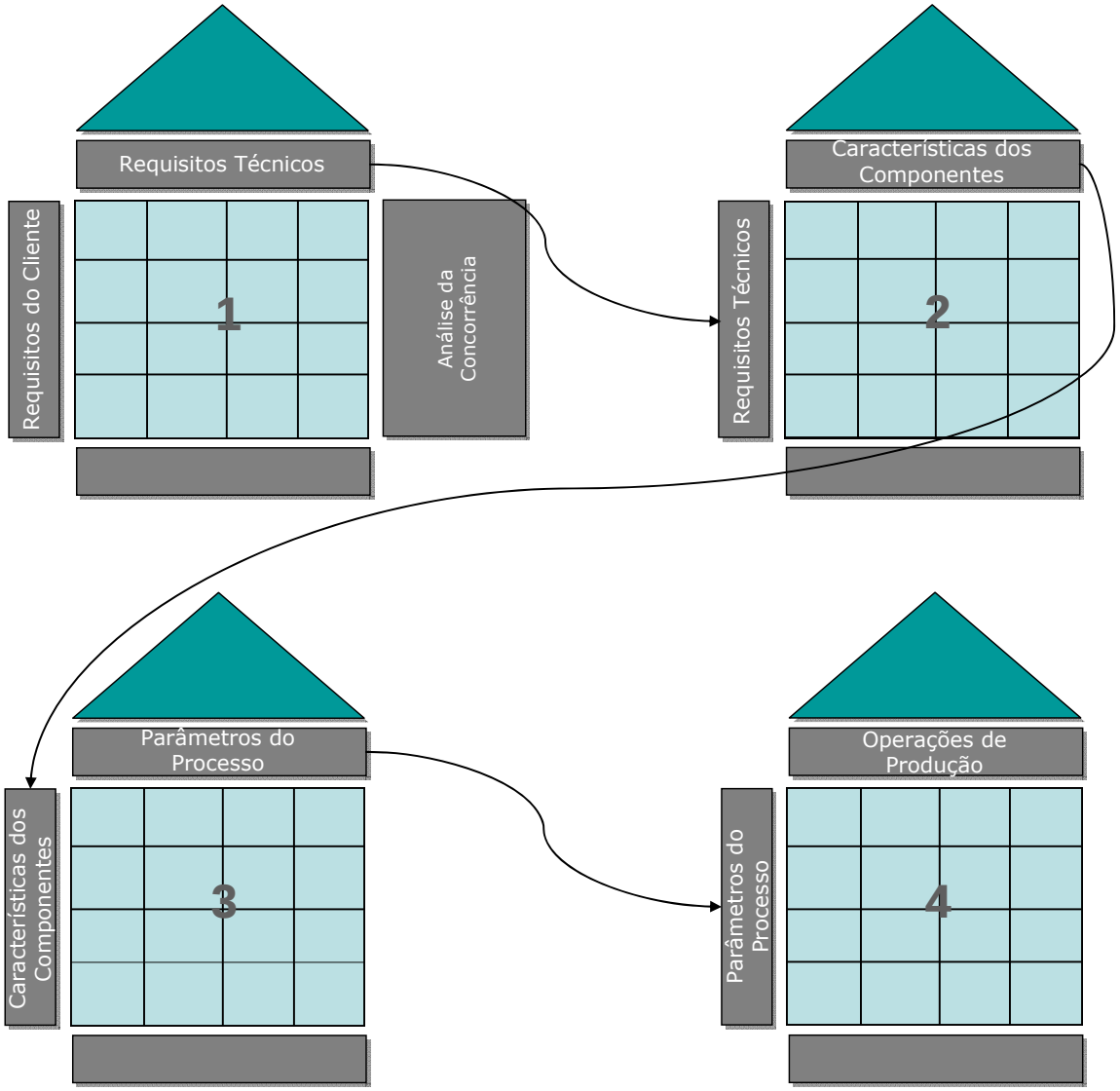

Figura 15: QFD em Quatro Fases
(Adaptado de [Cohen, 1995])

3.4. Casa da Qualidade

A casa da qualidade é a primeira matriz do QFD, e a mais amplamente usada, assim designada pela forma triangular que apresenta na parte superior do diagrama de representação, parecida com o típico telhado de uma casa [Akao, 1990].

A casa da qualidade tem como principal finalidade a sistematização das necessidades dos clientes, reveladas mediante expressões linguísticas, convertendo-as então em soluções técnicas, mostrando a correlação entre as necessidades e soluções propostas [Cohen, 1995].

A casa da qualidade pode ser vista como um processo composto por três actividades: (1) a sistematização das necessidades dos clientes; (2) a sua transformação em soluções técnicas; e (3) a identificação das correlações entre ambas [Bossert, 1991].

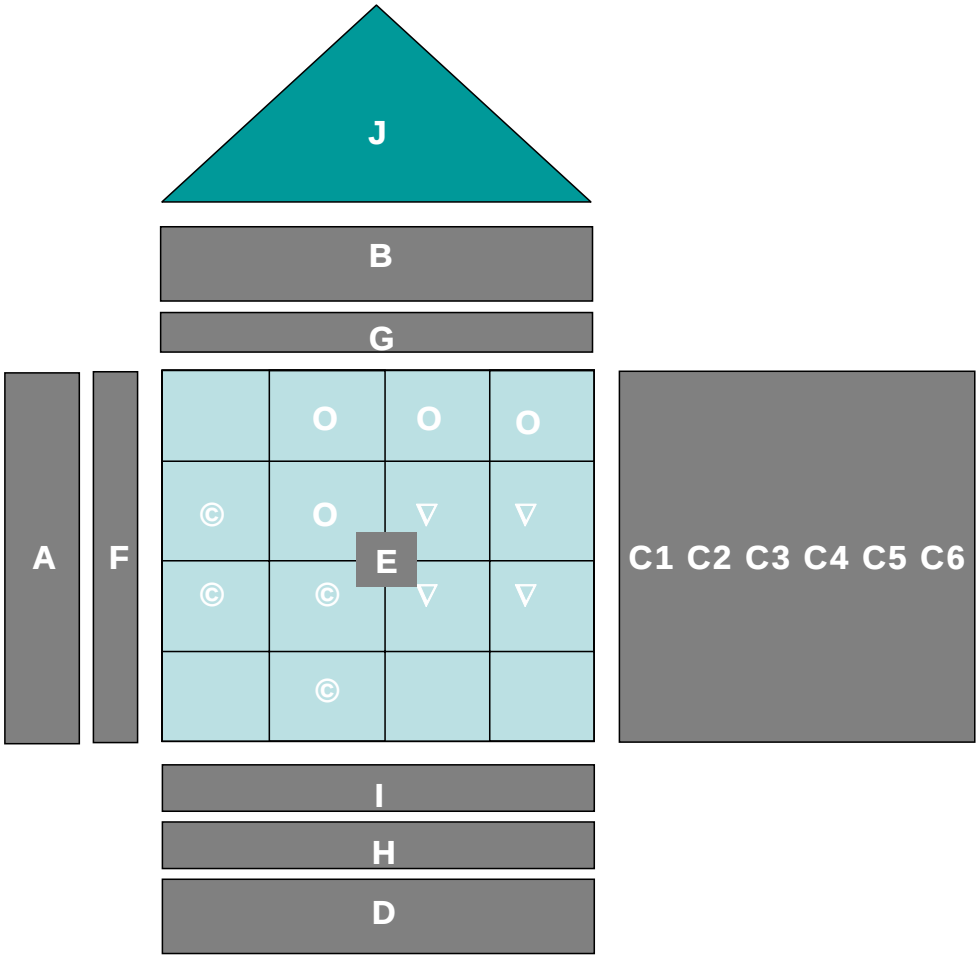

Figura 16: As Principais Secções da Casa da Qualidade

A casa da qualidade funciona assim como um sistema, onde o input é a voz dos clientes, na forma de expressões linguísticas, e o output consiste em especificações do produto do ponto de vista dos clientes, na forma de soluções técnicas. A Figura 16 apresenta as principais secções que formam a casa da qualidade, seguindo-se uma breve descrição sobre cada uma.

3.4.1. Secção A – Requisitos do Cliente

Esta secção é ocupada por uma tabela que contém as necessidades do cliente, sob a forma de expressões linguísticas, ou os requisitos do cliente que resultaram da conversão da expressão natural do cliente, em algo mais substancial, os designados requisitos do cliente. Os requisitos são obtidos, de preferência, a partir do cliente, podendo, no entanto, originar directamente da experiência da equipa de projecto [Bossert, 1991; Krogstie, 1999].

Os requisitos devem ser organizados preferencialmente por níveis hierárquicos, mediante a aplicação de diagramas de afinidades, numa estrutura em árvore [Cohen, 1995].

3.4.2. Secção F – Importância dos Requisitos

Esta secção pretende identificar o grau de importância que o cliente dá a cada um dos seus requisitos. Normalmente esse grau é obtido directamente a partir do cliente, que atribui uma classificação a cada requisito, de acordo com uma escala pré-determinada, relativa ou absoluta [Cohen, 1995; Jagdev et al., 1997].

A equipa de QFD, também, pode usar como método o Analytical Hierarchy Process - AHP para ajudar o cliente a determinar a importância dos requisitos [Akao, 1990; Cohen, 1995; Mazur, 1996; Xie et al., 2003; Hierholzer et al., 1998; Dimsey e Mazur, 2002].

Segue-se uma pequena descrição do AHP, com vista a explicar os rudimentos do seu funcionamento, uma vez que este método, também será usado para quantificar os riscos, e poderá ser usado sempre que os gestores de projecto achem necessário quantificar avaliações subjectivas, ou quando não existem dados quantitativos, como por exemplo, na estimativa dos custos de determinados requisitos do cliente ou do software.

3.4.3. Análise Hierárquica Estruturada (AHP)

Esta é uma técnica de análise de decisão e planeamento com múltiplos critérios desenvolvida por Thomas Saaty, em 1971. A teoria do AHP reflecte o que parece ser um método natural de funcionamento da mente humana, a qual, ao ser confrontada com um grande número de elementos, procura agregá-los em grupos com características comuns [Saaty, 1991].

O AHP proporciona uma estrutura para ordenar as opiniões de uma forma intuitiva e consistente, que traduz de forma clara a preferência de quem decide.

O AHP permite aos avaliadores efectuar comparações relativas entre dois elementos, procurando quantificar em que medida um determinado elemento é mais importante que outro na prossecução dos objectivos em análise. Saaty propôs o uso da escala apresentada no Quadro 1, para avaliar a importância de todos os elementos, par-a-par [Saaty, 1991].

Intensidade da Importância	Definição	Explicação
1	Mesma importância	Dois elementos contribuem de igual modo para o objectivo
3 ou 1/3	Importância reduzida	A experiência e a avaliação favorecem levemente um elemento em relação ao outro
5 ou 1/5	Importância elevada	A experiência e a avaliação favorecem fortemente um elemento em relação ao outro
7 ou 1/7	Importância muito elevada	Um elemento é fortemente favorecido; a sua importância é demonstrada na prática.
9 ou 1/9	Importância absoluta	As evidências favorecem um elemento em relação ao outro com um grau elevado de certeza
2 (1/2), 4 (1/4), 6 (1/6), 8 (1/8)	Importâncias intermédias	Quando se deseja maior compromisso

Quadro 1: Escala Padrão de Avaliações Proposta por Saaty
(Adaptado de [Saaty, 1991])

O conjunto de todas as comparações par-a-par efectuadas é representado em matrizes da seguinte forma:

$$M = \begin{bmatrix} 1 & a_{12} & \dots & a_{1n} \\ \frac{1}{a_{21}} & 1 & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ \frac{1}{a_{n1}} & \frac{1}{a_{n2}} & \cdots & 1 \end{bmatrix}$$

O número de julgamentos necessários para a construção da matriz é $n \cdot (n-1)/2$, onde (n) é o número de elementos da matriz (M).

Após o preenchimento da matriz de comparação, primeiro calcula-se a soma para cada uma das (n) colunas, divide-se cada elemento da matriz pelo somatório da coluna a que pertence o elemento, e calculam-se as somas de cada linha, depois efectua-se a normalização das somas das linhas, mediante a divisão da soma de cada linha pelo número de elementos. O resultado destes cálculos é referido como a matriz de prioridades [Karlson e Ryan, 1997].

$$W = \begin{bmatrix} \sum W_1 / n \\ \sum W_2 / n \\ \dots \\ \sum W_n / n \end{bmatrix}$$

Onde, (W) é uma matriz normalizada dos valores médios na matriz de comparações, utilizada para quantificar e ponderar a importância dos vários elementos analisados.

A fim de testar a consistência das respostas – razão de consistência (RC), que indica se os dados da matriz (M) foram relacionados de forma lógica, é preciso calcular um índice de consistência (IC), através da seguinte expressão:

$$IC = (\lambda \text{máx} - n) / (n-1)$$

A partir da multiplicação das matrizes (M) e (W), obtém-se uma nova matriz, de nx1, em que o primeiro elemento da matriz resultante (T), é dividido pelo primeiro elemento da matriz (W), e assim sucessivamente; deste modo, λ máx é dado pela equação:

$$\lambda máx = \frac{T_1/W_1 + \ldots + T_n/W_n}{n}$$

Com base no cálculo do índice de consistência (IC) e no índice de consistência aleatória (ICA), para (n) elementos, mostrado no Quadro 2, é possível calcular a razão de consistência (RC), dado pela equação $RC = IC / ICA$, em que se considera aceitável uma razão de consistência menor do que 0,10; caso contrário, as comparações estabelecidas entre os elementos da matriz (M) deverão ser revistas [Karlson e Ryan, 1997].

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ICA	0	0	0,58	0,90	1,12	1,24	1,32	1,41	1,45	1,49	1,51	1,48	1,56	1,57	1,59

Quadro 2: Índice de Consistência Aleatória
(Adaptado de [Karlson e Ryan, 1997])

O índice de consistência aleatória (ICA), apresentado no Quadro 2, é proveniente de uma amostra de matrizes recíprocas, com uma escala de 1 a 9, geradas aleatoriamente [Karlson e Ryan, 1997].

3.4.4. Secção C1 – Avaliação da Concorrência

Aqui é avaliada a concorrência do cliente, com vista a identificar como é que este percebe o desempenho do produto, em comparação com os principais concorrentes [Cohen, 1995].

É preciso saber exactamente qual é o desempenho do produto e quais as características que o condicionam. É a partir desta informação e da avaliação do cliente, que a equipa pode estabelecer um referencial para as características do produto e qual o grau de satisfação a alcançar, que servirá para classificar os produtos concorrentes e planear a qualidade do produto a desenvolver [Bossert, 1991].

Mais uma vez a importância dos requisitos pode ser estabelecida com base numa escala relativa ou absoluta.

3.4.5. Secção C2 – Metas para os Requisitos

Aqui são definidas as metas para a qualidade dos requisitos, isto é, estabelece-se a performance mínima a alcançar em cada requisito do cliente [Cohen, 1995].

3.4.6. Secção C3 – Objectivos de Vendas

Representa os principais benefícios em termos de vendas que o produto trará ao cliente, com base no grau satisfação das suas necessidades [Cohen, 1995].
Os resultados alcançados com as vendas do produto traduzem o grau de consonância dos requisitos do cliente, com a sua política para o mercado alvo [Akao, 1990; Cohen 1995].

3.4.7. Secção C4 – Índice de Melhoramento da Performance

Esta secção apresenta um índice de melhoramento, determinado pela divisão da performance desejada, pela que foi conseguida até à data, reflectindo em que medida o produto actual precisa de melhorar o seu desempenho.

3.4.8. Secção C5 – Peso Absoluto

Aqui é colocado o peso absoluto dos requisitos, que resulta da multiplicação da importância dos requisitos, pelas vendas a alcançar, estabelecendo diferentes prioridades para os requisitos, na perspectiva de que os esforços se devem concentrar nos requisitos mais importantes, nos que estão em consonância com a estratégia do cliente e naqueles que precisam de ser aperfeiçoados [Cohen, 1995].

3.4.9. Secção C6 – Peso Relativo

Apresenta o peso relativo dos requisitos, determinado pela conversão do peso absoluto em percentagem, através da divisão do peso absoluto de cada requisito pelo resultado da soma de todos os pesos absolutos, com o objectivo de facilitar a percepção da importância relativa dos requisitos [Cohen, 1995].

3.4.10. Secção B – Soluções Técnicas

Descreve, sob a forma de tabela, as respostas técnicas às necessidades do cliente expressas em termos linguísticos ou, desde logo, as características do

produto definidas directamente pela equipa de projecto, com base nas necessidades expressas pelo cliente (soluções técnicas).
A tabela de soluções técnicas, tem como função traduzir a "voz do cliente" para a "voz do engenheiro", ou seja, transformar os requisitos do cliente em características de projecto, de modo a que permitam constituir um produto com qualidade [Cohen, 1995; Mazur, 1996].

3.4.11. Secção E – Correlações Existentes

Representa a intensidade da relação entre a Secção A e a Secção B. A matriz de relações é composta por células, que indicam a medida em que cada solução técnica responde às necessidades identificadas [Cohen, 1995].
A intensidade das relações deve seguir uma escala de quatro níveis: inexistente, fraca, média e forte, expressos na forma de números {0,1,3,9} ou símbolos {Vazio, O, ©, ∇} [Akao, 1990].

3.4.12. Secção D – Soluções Prioritárias

As soluções prioritárias são calculadas a partir da importância dos requisitos (secção F) e das correlações obtidas (secção E). Se (Fj) representar a importância do requisito (j) do cliente e (En,j) a correlação entre a solução técnica (n) e o requisito (j), então, a importância da solução (n) será dada pela equação:

$$D_n = \sum_{j=1} E_{n,j} \cdot F_j$$

3.4.13. Secção H – Performance a Atingir

Aqui são definidas as metas, em valores e unidades de medida, a atingir com cada solução proposta, com dois objectivos em mente: (1) determinar se são mensuráveis e (2) clarificar o que é considerado um valor ideal.
Existem soluções, cujos valores de performance "quanto maiores, melhor", outras em que "quanto menores, melhor" e há ainda situações em que o melhor resultado consiste em alcançar um valor concreto [Cohen, 1995].
Nesta fase ainda não se está a procurar definir o valor ideal, mas descobrir como determiná-lo. Caso não seja possível definir os objectivos técnicos para uma

determinada solução, significa que esta não é quantificável, pelo que, o desdobramento deverá prosseguir, até que se obtenha uma solução técnica mensurável [Cohen, 1995; Barnard, 2005].

3.4.14. Secção J – Impactos entre as Soluções

Esta secção identifica a influência que as soluções têm entre si, contribuindo deste modo para a resolução de conflitos entre as soluções propostas; são comparadas entre si, duas a duas, permitindo identificar como se relacionam. As correlações entre as soluções poderão seguir uma escala idêntica à usada na secção E, representando a medida em que o desempenho positivo de uma contribui para a performance positivo da outra solução, ou a performance positiva de uma prejudica outra. Em alternativa, poderá ser usada a escala de intensidades {Positiva_Forte, Positiva, Negativa, Negativa_Forte} [Xie et al., 2003].

A matriz de correlações também pode ser usada para identificar correlações de influência entre os requisitos do cliente [Cohen, 1995].

3.4.15. Secção I – Orientação das Soluções

Esta secção revela a direcção que as soluções devem tomar, isto é, se os valores da Secção H, devem aumentar, diminuir ou atingir um valor específico face à concorrência do cliente.

3.4.16. Secção G – Dificuldades Técnicas

Este factor expressa as dificuldades tecnológicas em conseguir obter uma solução com o valor especificado, isto é, este factor determina quais são as soluções que, provavelmente, exigirão maior esforço e recursos.

A matriz de planeamento, ou casa da qualidade, contém, assim, os requisitos gerais do cliente, sendo usada para traduzir os requisitos extraídos de pesquisas de mercado, de comparações com os concorrentes ou de avaliações internas do cliente, em requisitos técnicos do produto [Cohen, 2005].

A casa da qualidade pode ser definida como a matriz de execução do projecto da qualidade, sistematizando as qualidades práticas exigidas pelo cliente por meio de expressões linguísticas, convertendo-as em características substitutas

(soluções técnicas) e mostrando a correlação existente entre as soluções e as qualidades práticas [Akao, 1990].
A importância da matriz da qualidade revela-se, assim, na tradução das frases qualitativas do cliente em informações mensuráveis.

3.5. QFD e o Software

No contexto do desenvolvimento de software, a qualidade refere-se a um conjunto de actividades, executadas com a finalidade de satisfazer as necessidades dos clientes, apesar de diferentes clientes terem perspectivas antagónicas da qualidade [Evans, 2004].
O software possui qualidade, quando dá uma resposta fiável, acessível, segura e em tempo útil, às necessidades do cliente [McConnell, 1996].
A qualidade exige a identificação regular das necessidades e expectativas do cliente, dado que estas evoluem constantemente. Sem um sistema de qualidade, as organizações perdem a capacidade de comunicar com os clientes [Evans, 2004].
A qualidade está também associada à superioridade concorrencial. Para conseguir uma vantagem competitiva, o primeiro passo é implementar um conjunto de acções, que dêem a certeza de que os produtos possuem a qualidade desejada. É preciso desenvolver e implementar processos de garantia da qualidade dos produtos [Porter, 2001-a]; um produto que precise de menos recursos poderá ser vendido a um preço mais reduzido [Cohen, 1995].
As organizações precisam de mudar, no entanto, é elementar que as mudanças devem ser direccionadas pelos clientes [Porter, 1998].
Os problemas do software resultam, em parte, da dificuldade de tradução dos requisitos dos clientes em requisitos técnicos e, bem assim, na adopção de processos adequados [Ewusi-Mensah, 2003; Stepanek, 2005].
O aumento da concorrência e as rápidas mudanças tecnológicas provocaram uma drástica redução do ciclo de vida do software [Koch, 2005]; as organizações procuram métodos que consigam um menor tempo de resposta, aumento da produtividade, elevada qualidade dos produtos e satisfação dos clientes [Koch, 2005].

O QFD é uma resposta possível para esta procura – é um processo que permite ouvir e traduzir a voz dos clientes, vertendo-a nos processos de software e, bem assim, estabelecer prioridades para a sua implementação. É um mecanismo que impulsiona o processo de planeamento, dirigindo-o para o mercado, revestindo-se de grande importância para qualquer projecto de software.

3.6. QFD e o Software - Desdobramento da Qualidade (QD)

É possível aplicar o modelo QFD de quatro fases, descrito anteriormente, às especificidades da produção de software. As matrizes seguem uma estrutura, que nas linhas apresentam um conjunto de requisitos, com origem no cliente, ou que transitam de matrizes anteriores, representando o que é pretendido "O Quê", e nas colunas apresentam o "Como" responder a esses requisitos [Herzwurm et al., 2002].

Com o intuito de transmitir uma noção geral do funcionamento do processo, é apresentado de forma simplificada o modelo do QFD adoptado para a gestão de projectos de software. Assim, o modelo conceptual é composto por seis matrizes (ver Figura 17):

- Desdobramento de necessidades do cliente, em funções do software e em requisitos do cliente;
- Desdobramento dos requisitos do cliente em requisitos do software;
- Desdobramento dos requisitos de software em requisitos dos subsistemas (ou definição da arquitectura);
- Desdobramento dos requisitos do software em possíveis riscos.
- Desdobramento dos requisitos do software em actividades dos processos;
- Desdobramento das actividades em requisitos dos recursos;

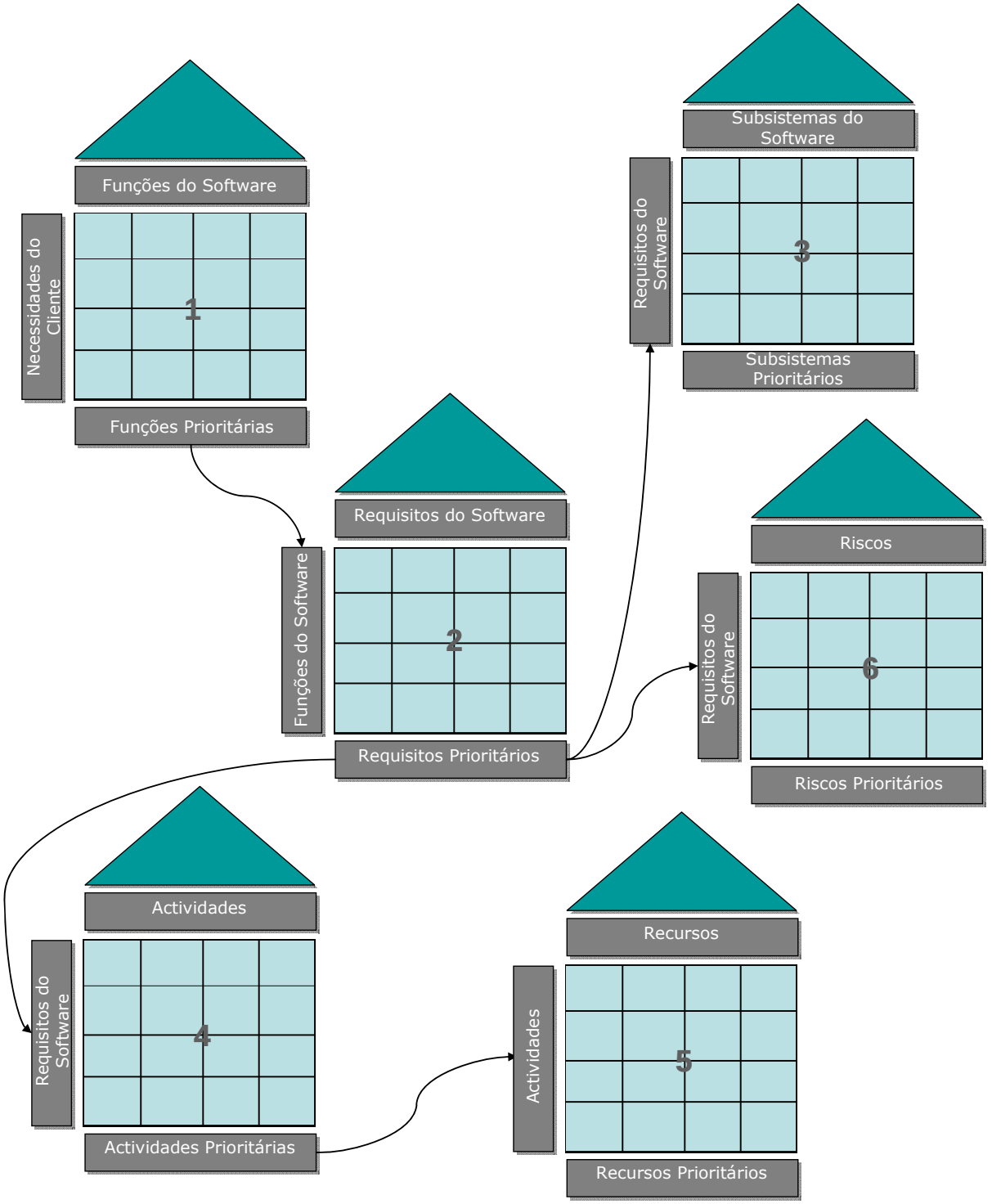

Figura 17: Modelo QFD para Software

O princípio básico do relacionamento entre as matrizes centra-se no facto de que os elementos mais importantes, obtidos numas matrizes, servirão como input para outras.

Contudo, por uma questão de manutenção da carga administrativa num nível razoável, somente os elementos das matrizes considerados mais importantes, obtidos em qualquer uma das matrizes, serão transformados em inputs para as restantes matrizes.

3.6.1. Matriz de Funções

A primeira das matrizes do modelo QFD para software é, também, conhecida como a casa da qualidade, descrita na secção anterior.

O objectivo desta matriz é identificar as funções do software, na perspectiva do cliente. É necessário que a voz do cliente seja ouvida, identificados os seus problemas e consideradas as diversas soluções com possibilidade de satisfazer as necessidades identificadas [Cohen, 1995].

No caso do desenvolvimento de um produto para um cliente específico, que também utilize o QFD, os resultados das matrizes do cliente, poderão servir como input para as matrizes do modelo QFD proposto.

As principais metas a estabelecer para o software (redução dos custos, aumento da quota de mercado, aumento de produtividade, etc.), que servirão de input para a matriz da casa da qualidade, deverão ter origem no plano estratégico do cliente, ou num processo de análise prévia dos requisitos, enquadrada no cenário concorrencial [Bossert, 1991].

As fases de construção da matriz da qualidade consistem essencialmente em (1) conhecer o cliente, as suas necessidades, e expectativas sobre o software, (2) transformá-los em funções, mediante questionários, entrevistas, ou qualquer outro método adequado ao problema em consideração, (3) ajudar o cliente a estabelecer prioridades para as suas necessidades, (4) identificar as funções do software que contribuem para satisfazer o cliente, (5) sistematizar, sob a forma de matriz, a correlação entre as necessidades e as funções, e (6) estabelecer prioridades para a implementação das funções (ver Figura 18).

A identificação das necessidades do cliente é fundamental para definir o público-alvo e o mercado a ser estudado [Akao, 1990; Bossert, 1991]. É a partir desta informação que é possível fixar metas para as funções do software.

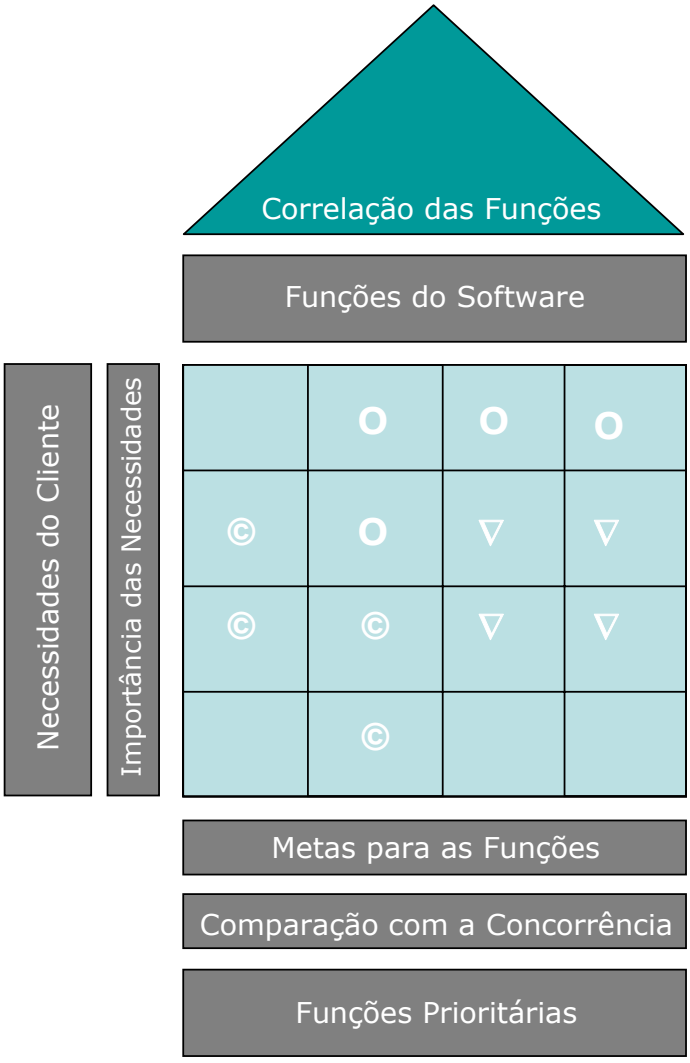

Figura 18: Primeira Matriz – Necessidades *versus* Funções do Software

A definição de prioridades para os requisitos obriga a que sejam estabelecidos vários indicadores, a saber:

- Qual a importância que o cliente atribui às necessidades que deseja colmatar (I_i);
- A estratégia do cliente, traduzida pelas metas estabelecidas para as funções (M_j);
- A avaliação da concorrência (C_j).

Assim, a prioridade das funções (I_j) resulta da correcção da importância definida pelo cliente (I_i), mediante a aplicação dos indicadores (M_i) e (C_j) e, bem assim, do factor de correlação entre as funções e necessidades ($I_{j,i}$), identificadas pela equipa que define o âmbito do software;

$$I_j = \left[\sum_{i=1} I_{j,i} \cdot Ii \right] \cdot \frac{M_j}{C_j}$$

As funções são simples descrições abreviadas do software. Os requisitos do software, descritos como "use cases" ou sob qualquer outra forma, expressam as funções em termos muito mais detalhados. Por outras palavras, as funções ajudam a compreender e a comunicar num nível de abstracção elevado, mas provavelmente não conseguem descrever cabalmente o software de modo a que se possa escrever código a partir dessas descrições. São demasiado abstractas para esse propósito [Leffingwell e Widrig, 1999].

3.6.2. Matriz de Requisitos do Software

Os requisitos do software traduzem as funções em requisitos técnicos, mensuráveis e objectivos. Esta tradução requer obrigatoriamente a colaboração dos analistas de sistemas, programadores, representantes da gestão da organização e, sempre que possível, dos utilizadores finais [Evans, 2004].
A partir do cruzamento da informação sobre as funções desejadas e dos requisitos do software, é possível determinar as prioridades para os requisitos, considerando, no entanto, a avaliação dos seguintes indicadores (ver Figura 19):

- A intensidade da correlação entre as funções e os requisitos ($I_{k,j}$);
- A importância relativa das funções (I_j);
- A avaliação dos requisitos do software em relação aos concorrentes mais directos, comparado o produto actual do cliente (ou um de referência) com os da concorrência (C_k);

- Com base na especificação técnica actual e na análise da concorrência são definidas metas a atingir para cada um dos requisitos de software (M_k);

- E, finalmente, é considerado o nível de dificuldade que a organização tem em implementar cada uma das soluções propostas (D_k) [Carpinetti e Peixoto, 1998; Lee, 2005].

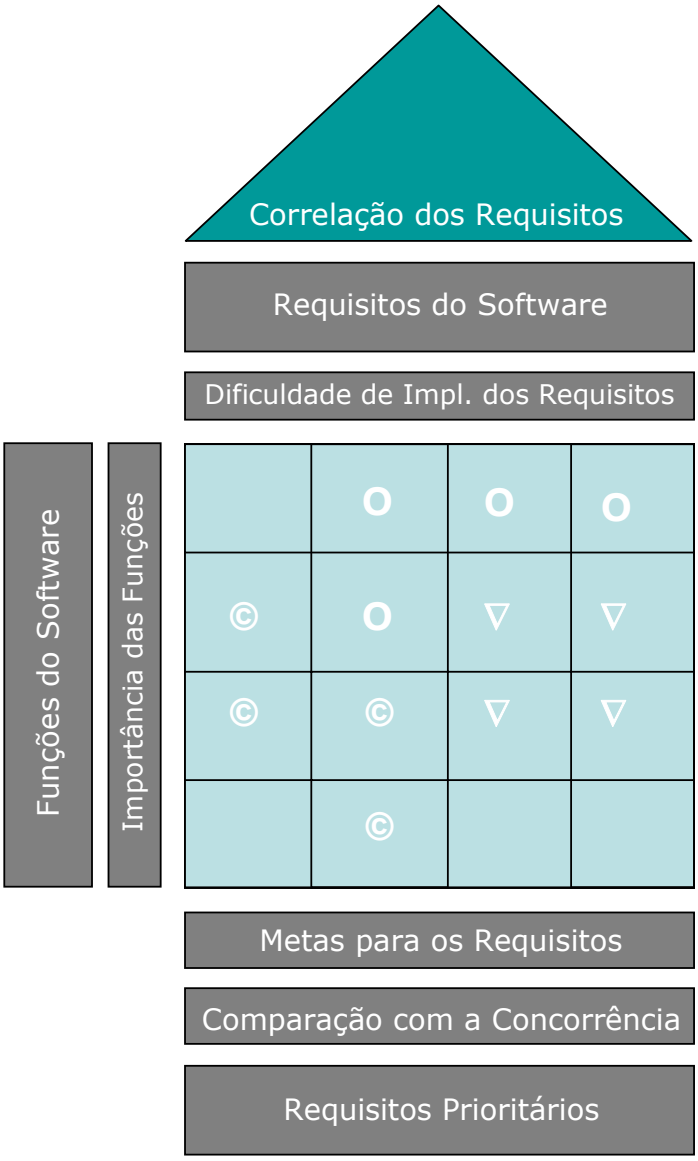

Figura 19: Segunda Matriz – Funções *versus* Requisitos do Software

A prioridade dos requisitos do software é obtida mediante a soma de todas as contribuições para a satisfação do cliente com uma dada solução, corrigido pelos indicadores M_k, C_k e D_k.

Se I_j for a importância relativa das funções do software, então, a importância de cada solução técnica, é dada pela equação:

$$I_k = \left[\sum_{j=1} I_{k,j} \cdot I_j \right] \cdot \frac{M_k}{C_k} \cdot \frac{1}{D_k}$$

3.6.3. Matriz de Subsistemas

Esta fase caracteriza-se pela especificação da relação estabelecida entre os requisitos de software, identificados na matriz anterior e os subsistemas do software a ser desenvolvido.

A matriz de subsistemas desdobra o software nos componentes que compõem o software – definição da arquitectura – procurando (1) identificar os subsistemas funcionais do software e (2) relacioná-los com os requisitos do software, com o objectivo de conceder prioridade aos subsistemas que estão fortemente relacionados com as funções desejadas pelo cliente (ver Figura 20).

Esta relação é quantificada com a identificação dos seguintes indicadores:

- A intensidade do relacionamento ($I_{k,l}$) entre os subsistemas e os requisitos, ou seja, a medida em que as principais soluções técnicas propostas influenciam cada um dos principais componentes do software;

- A importância dos subsistemas para o sucesso do software é determinada pelo cálculo do factor (I_l);

$$I_l = \sum_{k=1} I_{l,k} \cdot I_k$$

- É ainda possível considerar a concorrência para cada um dos subsistemas a desenvolver (C_l) e as metas (M_l) a atingir, com o objectivo de avaliar os subsistemas globalmente, e não apenas com base na importância dos requisitos que o compõem;

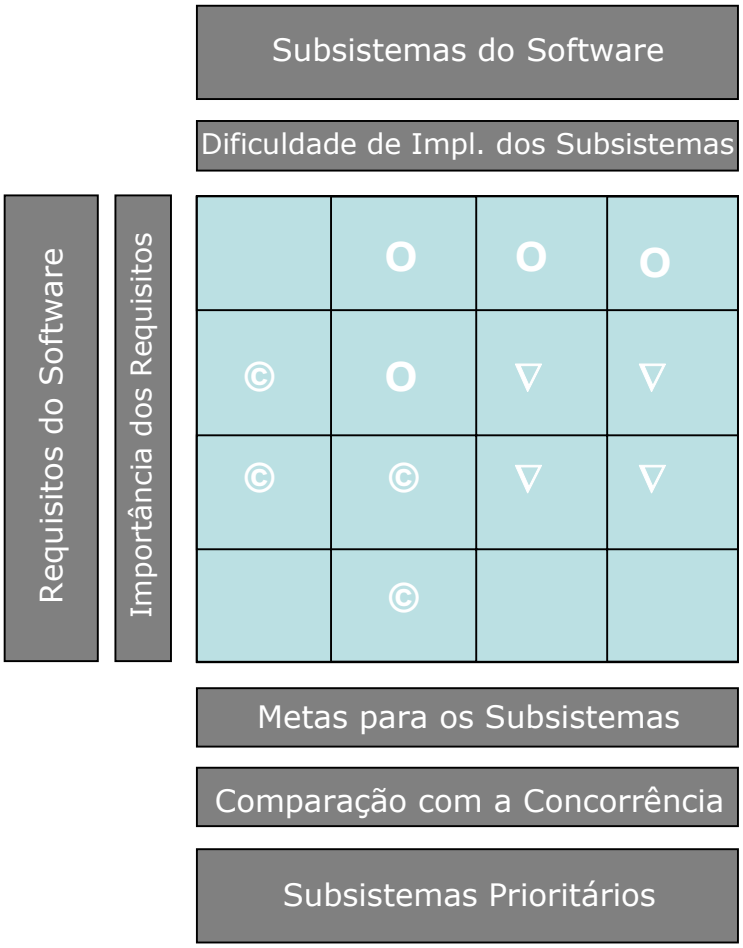

Figura 20: Terceira Matriz – Requisitos *versus* Subsistemas

Assim, a prioridade na implementação dos subsistemas é estabelecida em função da dependência dos requisitos e, eventualmente, da avaliação da concorrência.
A definição de prioridades para os subsistemas do software, através do cálculo do (I_I), encerra o desdobramento da qualidade (QD) do software, da divisão do QFD, preconizada por Akao, conforme foi discutido anteriormente.

3.7. QFD e o Software - Desdobramento da Função Qualidade (QFDr)

Em função da metodologia de desenvolvimento adoptada para o projecto de software e com base nos dados obtidos anteriormente, a equipa de projecto pode iniciar de imediato o processo criativo, porém, as metodologias apenas dão indicações genéricas sobre os procedimentos a seguir [Armour, 2004].

É preciso que a equipa concretize e adapte as orientações dadas pelas metodologias ao projecto concreto e às suas necessidades reais. Este é o objectivo da próxima matriz QFD.

3.7.1. Matriz de Processos / Actividades

Esta fase procura identificar os processos/actividades (processos e respectivas actividades) mais adequados para o planeamento, concepção, codificação e teste do software, para que consiga incorporar os requisitos mais importantes para o cliente, isto é, nesta fase são definidos os processos de produção do software em geral, bem como os indicadores de qualidade e produtividade de cada actividade ou conjunto de actividades (quando não for possível estabelecer os indicadores para cada actividade individualmente) que compõem o processo.

Os processos/actividades a definir não se limitam somente aos da produção do software, mas consideram-se, também, os processos de gestão do projecto.

O objectivo desta matriz é identificar os processos/actividades que melhor se adequam à gestão e implementação dos requisitos do software identificados.

A matriz identifica as actividades mais críticas para o software, permitindo ao gestor implementar medidas que melhorem os processos. É preciso não esquecer, porém, que os processos/actividades estão directamente relacionados com a metodologia adoptada.

A construção da matriz do processo inicia-se com o desdobramento do processo de software nas actividades que o compõem, para depois se correlacionar os requisitos com as actividades identificadas (ver Figura 21). O principal objectivo da aplicação desta matriz é atribuir uma prioridade mais elevada às actividades do processo que estão fortemente correlacionadas com a concretização dos requisitos do software, de modo a que os esforços de monitorização e controlo possam ser concentrados nas actividades mais importantes, evitando a dispersão de recursos.

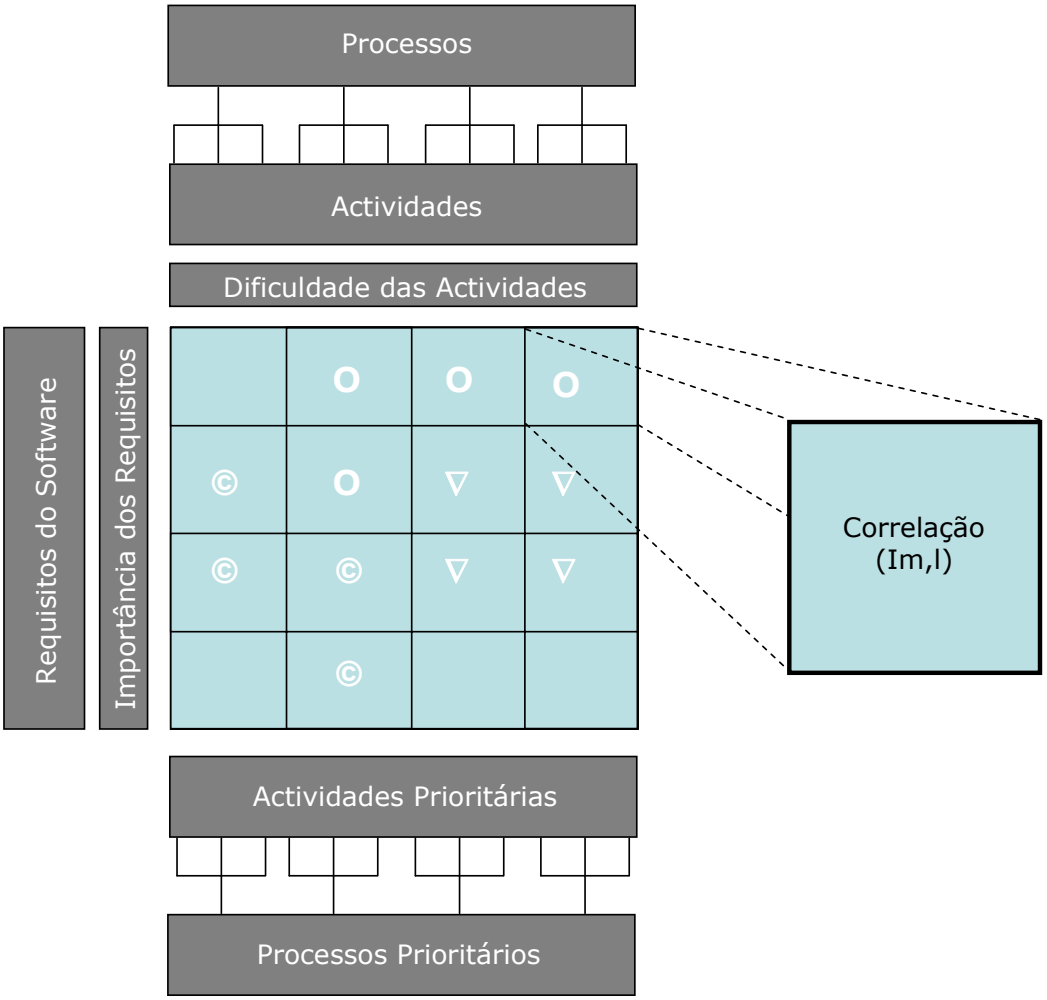

Figura 21: Quarta Matriz – Requisitos do Software *versus* Actividades

Os indicadores necessários à quantificação da prioridade são idênticos aos usados anteriormente, a saber:

- A intensidade da correlação das actividades dos processos com os requisitos do software ($I_{m,l}$), procurando avaliar em que medida cada actividade é essencial para a implementação dos requisitos de software;

- Atribuição de um nível de dificuldade à implementação das actividades (D_m), que avalia as actividades, na perspectiva de eventuais dificuldades com a sua implementação, monitorização, controlo e, bem assim, o custo das actividades.

A prioridade a dar às actividades (I_m) é conferida em função da importância da actividade em questão na concretização dos requisitos, bem como dos aspectos práticos da sua execução, monitorização e controlo.

$$I_m = \left[\sum_{l=1} I_{m,l} . I_l \right] \cdot \frac{1}{D_m}$$

Com o cálculo das prioridades das actividades do processo de fabricação do software, conclui-se o desdobramento da matriz de processos/actividades.

3.7.2. Matriz de Recursos

Se a matriz anterior definiu quais as actividade que serão executadas, nesta fase deverão ser identificados os recursos necessários à execução dos processos/actividades de construção do software (ver Figura 22).

É preciso identificar as capacidades necessárias (especialidades e níveis de formação), equipamentos, materiais, tangíveis ou não, consumidos pelos processos/actividades, pois os recursos a empregar variam muito em função do tipo de software e do processo usado.

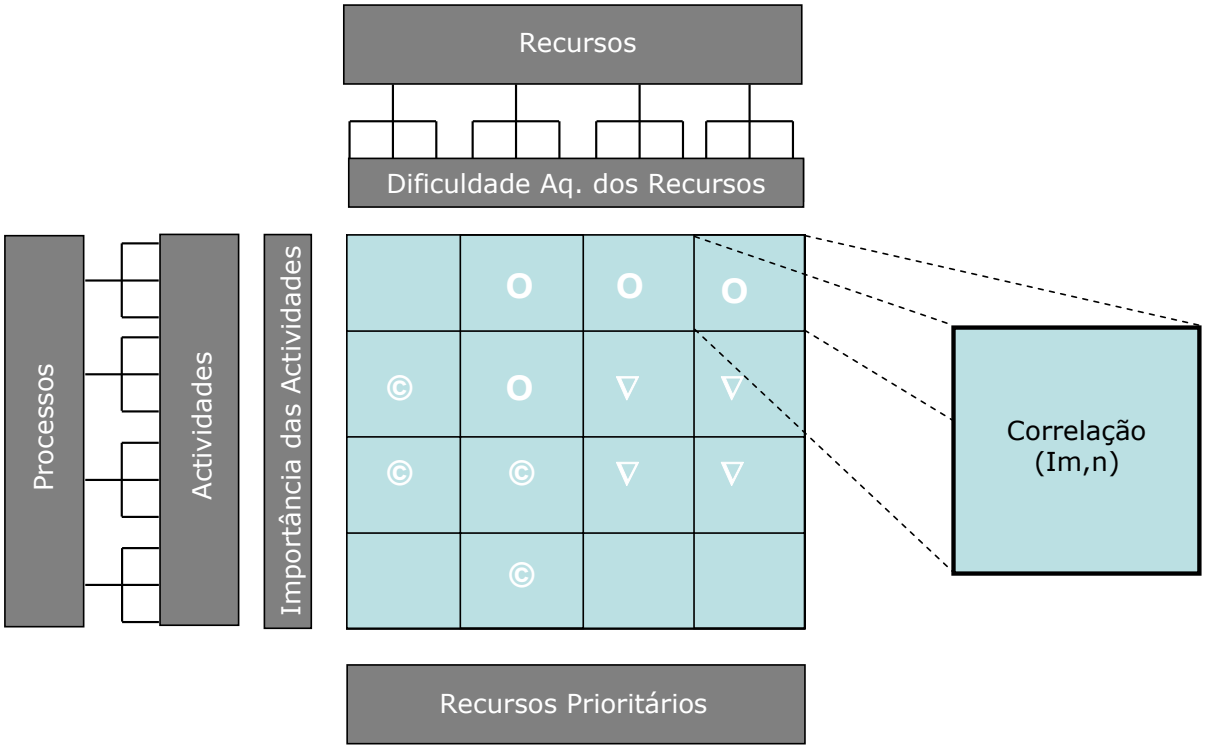

Figura 22: Quinta Matriz – Processos/Actividades *versus* Recursos

Esta matriz correlaciona as actividades do processo de produção com os recursos necessários à produção do software. A Matriz de Recursos requer (1) o desdobramento dos recursos, (2) o relacionamento dos recursos com os processos/actividades e a (3) definição de prioridades para a aquisição ou atribuição de recursos.

O desdobramento dos recursos tem como objectivo catalogar todos os recursos humanos, tecnológicos, matérias, etc., necessários à produção do software, para que então se possa correlacionar os processos/actividades com os recursos.

A ligação dos processos/actividades aos recursos, pretende dar prioridade à aquisição de mão-de-obra e tecnologias, e bem assim, de outros recursos considerados importantes, que estejam directamente relacionados com a concretização dos requisitos do software.

O conhecimento dos seguintes indicadores permite quantificar o nível de relacionamento entre os requisitos de software e os recursos:

- O grau de intensidade da correlação entre os recursos e as actividades dos processos ($I_{n,m}$);

- A escassez dos recursos ou custos elevados devem ser considerados na definição de prioridades para os recursos, através da aplicação de um factor de dificuldade (D_n);

A prioridade na aquisição dos recursos é determinada a partir da importância destes e dos aspectos práticos da sua implementação.

$$I_n = \left[\sum_{m=1} I_{n,m} \cdot I_m \right] \cdot \frac{1}{D_n}$$

A matriz de recursos permite identificar aqueles que podem contribuir para aumentar a eficiência económica dos processos/actividades de software e, bem assim, prestar indicações preciosas aos gestores sobre a equipa a adquirir, ou qual será a formação mais adequada dos recursos humanos da organização.

Esta matriz conclui o processo de desdobramento do produto em requisitos, processos/actividades e recursos. Mas, tal como foi discutido no Capítulo 2, a

gestão de projectos não seria eficaz, sem uma gestão adequada dos riscos, inerentes a qualquer projecto.

3.7.3. Matriz de Riscos

A matriz de riscos permite identificar as ameaças presentes nos ambientes interno e externo à produção do software, analisá-las e estabelecer prioridades de acordo com os factores risco de maior importância.

O processo de avaliação deverá ser periódico, uma vez que num ambiente dinâmico podem surgir e desaparecer novos factores de risco que, também, podem variar quanto ao grau de impacto [Heldman, 2005].

É possível quantificar o risco do projecto, mediante a identificação dos principais factores e subfactores de risco que, eventualmente, poderão afectar o projecto [Dey, 2002]. A identificação e quantificação de risco através da especificação de um conjunto de factores de risco constituem um dos métodos mais usados na identificação dos riscos [Heldman, 2005].

O processo de avaliação dos riscos consiste basicamente em (1) identificar os mais comuns, de modo a criar um padrão aplicável à maioria dos projectos de software da organização, (2) desdobrá-los em factores de risco, específicos do projecto em análise, (3) relacionar os factores de risco com os diversos riscos e (4) quantificar os riscos e relacioná-los com os requisitos do software, de modo a estabelecer prioridades para uma análise individualizada mais pormenorizada (ver Figura 23).

A identificação e quantificação do risco passam pela definição de possíveis cenários, que incluem alguns dos riscos mais comuns, tais como o custo, o prazo, os recursos, os componentes, os processos e os requisitos de qualidade. Em cada cenário existem factores que podem ser categorizados, por exemplo, económicos, técnicos e comportamentais [Bohem, 1991; Heldman, 2005; Kendrick, 2003].

A estes factores será atribuída a mesma importância ou, então, poderá ser usado novamente o AHP, para atribuir níveis de importância diferentes para cada um dos factores de risco, diminuindo, desta forma, a arbitrariedade na definição de escalas de importância [Dey, 2002; Korpela et al., 2002; Panthi, 2007].

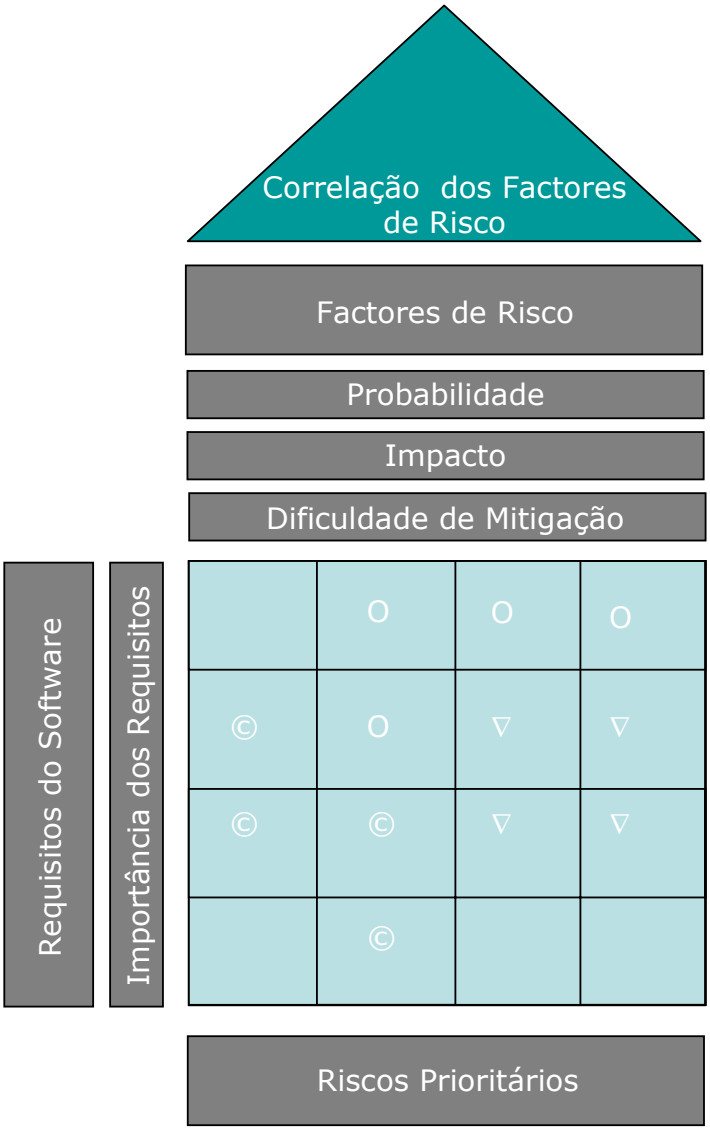

Figura 23: Sexta Matriz – Factores de Risco *versus* Requisitos do Software

A identificação dos riscos deverá ser específica para o software ou cliente em questão, contrariamente aos factores de risco, que também dependem da organização [Heldman, 2005].

A forma mais directa de avaliar os factores de risco consiste em determinar a exposição dos requisitos do software ao risco, em termos de probabilidade e do impacto da ocorrência de cada um dos factores padrão.

A exposição ao risco é uma medida quantitativa largamente conhecida e utilizada para avaliar e definir prioridades para os riscos [SWEBOK, 2001; Kendrick, 2003; PMI, 2004]. A sua simplicidade e aplicabilidade fazem com que esta técnica quantitativa permanece sem rival.

A exposição ao risco representa o potencial de perda para um indivíduo, projecto ou organização, em função da probabilidade de ocorrência do risco e da magnitude das suas consequências [Kendrick, 2003; Heldman, 2005].
O risco pode assim ser definido como a probabilidade de um evento indesejável ocorrer e a importância que terá, caso se verifique a sua ocorrência [Bohem, 1991; Kendrick, 2003].
O nível de exposição ao risco traduz-se na seguinte equação genérica: $R = P \cdot L$.
A probabilidade do risco é representada pelo valor P, que se deve situar no intervalo de 0 a 1. Se a probabilidade for 0, o risco não acontecerá, portanto não existe, mas se a probabilidade for igual a 1, significa que risco é uma certeza e consequentemente, deixou de ser risco e transformou-se num problema. O impacto L pode ser expresso em unidades monetárias ou qualquer outra.[1]
Em projectos de software, um evento ou acção para que possa ser considerado um risco, deve ter sempre associada uma perda, uma hipótese ou alguma escolha, ou seja, o risco pode ser modificado por uma acção [Fairley, 1994].
Os indicadores necessários à quantificação da prioridade dos riscos são muito idênticos aos usados nas análises anteriores, a saber:

- A correlação dos factores de risco com os requisitos do software é expressa pelos factores probabilidade (P_o) e impacto (I_o), cujo produto (R_o), determina o nível de exposição dos requisitos do software a cada um dos factores de risco;

- A exposição ao risco obtida para cada um dos factores deve ser corrigida pela importância relativa de cada factor, definida anteriormente, para toda a organização ou por classes de projectos;

- A mitigação dos riscos poderá implicar custos adicionais, que deverão ser considerados no momento de escolher os riscos a tratar. Estas limitações são representadas por um indicador (D_o);

[1] Por exemplo, na situação em que a probabilidade de ocorrência de um determinado risco seja de 15% (P = 0,15), com uma perda média estimada em 12000 €, a exposição ao risco é definida da seguinte forma: $R = P \cdot L = 0{,}15 \cdot 12000 = 1800€$.

O cálculo da importância dos riscos (I_o), permite definir uma medida de avaliação do grau de relação dos riscos com a concretização dos requisitos do software. Este indicador é obtido pela expressão:

$$I_o = \left[\sum_{k=1} \left[P(I_{o,k}) \cdot L(I_{o,k}) \right] \cdot I_k \right] \cdot \frac{1}{D_o}$$

A prioridade a dar aos riscos (I_o) é conferida em função da influência que têm nos requisitos, em termos de probabilidade e impacto, e dos aspectos práticos da sua mitigação, monitorização e controlo, bem como da importância que os requisitos têm na satisfação das necessidades do cliente.
O cálculo de prioridades no controlo dos riscos encerra o desdobramento da função qualidade do ponto de vista de gestão do projecto de software (QFD restrito).

3.8. Conclusões do Capítulo

Este capítulo procurou mostrar como o QFD pode ser usado para incorporar no software as reais necessidades dos clientes. Este é um instrumento de fácil utilização, que mantém o gestor de projecto e a equipa de desenvolvimento, em qualquer fase do projecto, focados na "voz do cliente", resultando em produtos com menos problemas, que são provocados, quer por uma definição insuficiente dos requisitos, quer por problemas de comunicação. A utilização de matrizes implica que o cliente especifique claramente, com a ajuda dos engenheiros de software, quais as suas necessidades, isto é, as matrizes estabelecem uma ponte entre clientes e técnicos.
O envolvimento dos técnicos na definição dos requisitos do cliente permitirá, por seu lado, que os requisitos do cliente sejam verdadeiramente compreendidos e traduzidos nas melhores soluções técnicas.
A partir dos requisitos do cliente, sob a forma de funções do produto – requisitos do software de alto nível – é possível passar para a definição dos requisitos do software propriamente ditos e, a partir daqui, eventualmente, poderá ser criada uma matriz que relacione os requisitos do software com o código que, no entanto, será extremamente difícil, senão impossível de usar.

Mas, quando é exigido um elevado nível de controlo ou quando este é exigido, por entidades públicas ou relações contratuais, os relacionamentos entre os requisitos do software e o código, devem ser especificados sob a forma de matriz. O sucesso deste objectivo depende muito da sofisticação das ferramentas usadas na definição dos requisitos, análise e concepção, pois sem uma ferramenta que possibilite a automatização deste processo, rapidamente surgirão problemas, dado que, por exemplo, num processo de desenvolvimento orientado a objectos, é provável que sejam produzidas muitos use cases e classes.

O modelo QFD para software contempla a análise dos riscos, que poderá ser abordada sob perspectivas qualitativas e quantitativas; a primeira, requer apenas que o gestor do projecto efectua uma avaliação da probabilidade e do impacto com base na sua experiência, bem como na importância dada pelos clientes e pelos técnicos, às funções e aos requisitos do software, respectivamente, enquanto a segunda perspectiva requer dados mais concretos, como os custos estimados dos requisitos do software e simulações probabilísticas, a discutir no próximo capítulo.

As organizações sempre possuíram sistemas para contabilizar os custos dos seus projectos, pelo que o registo dos custos directos, principalmente os de mão-de-obra directa e dos custos com materiais, não constitui um conceito novo para estas organizações. Mas a surpreendente mudança, no volume e magnitude, dos custos directos para os indirectos, conduziu a que as organizações comecem a questionar se os custos dos métodos de gestão actuais, não excedem os benefícios [Cokins, 2001].

No próximo capítulo, são apresentados os processos ABC/ABB e TDABC/TDABB para identificar os custos do software, de uma forma mais exacta e com um custo de gestão menor, procurando dar resposta às preocupações actuais das organizações produtoras de software.

CAPÍTULO 4 – Custeio por Actividades

Até este momento houve, com a aplicação do QFD ao software, uma preocupação em auscultar o cliente, procurando compreender quais as necessidades que deseja ver respondidas pelo software, sob a forma de funções, que posteriormente serão traduzidas em requisitos do software, subsistemas e código. Estas operações são executadas, necessariamente, por actividades, que consomem recursos. Os requisitos do software, as actividades e os recursos estimados, precisam, no entanto, de uma gestão de custos, adequada a uma nova realidade.

As organizações têm diferentes opções quando se trata de decidir como medir e reportar os seus custos. O método e a abordagem escolhidos dependem de uma grande variedade de factores. Mas uma coisa é certa: no ambiente actual, mais competitivo e de recursos escassos, há um maior escrutínio da forma como os custos são imputados [Cokins, 2001].

Na origem desta mudança encontra-se, sem dúvida, a grande evolução tecnológica, que criou situações de enorme concorrência entre as organizações [Bossidy et al., 2002]. Os gestores e as equipas sentem-se cada vez mais responsabilizados, o que tem consequências nas suas acções e resultados – as pessoas não toleram que lhes sejam imputados custos, quando sentem que outra área organizacional, equipa de trabalho, produto ou serviço, esteve na origem dos custos [Cokins, 2001].

É justamente aqui, que os mecanismos de controlo de custos e, bem assim, a produção de informação para apoio à decisão, possuem uma importância fundamental. Mas, também aqui, a evolução tecnológica teve um papel importante, pois veio facilitar a implantação de controlos, com a possibilidade da realização de controlos de custos mais detalhados [Cokins, 2001].

4.1. Evolução e Estrutura do ABC

O custeio por actividades (ABC) compila um conjunto de informações financeiras e operacionais sobre as actividades da organização, considerando os custos dos

atributos das actividades, tais como a duração, qualidade, etc., para assim calcular a importância real das actividades para o processo de produção [Hicks, 1999]. O ABC permite obter um conhecimento profundo do negócio, pois analisa minuciosamente os processos, identificando a combinação de actividades, o consumo de recursos e os produtos produzidos (objectos de custo) [Kaplan e Cooper, 1998].

A lógica do custeio por actividades está justamente na identificação das actividades e na afectação de recursos às mesmas. Somente a partir do momento em que os recursos são transferidos para os objectos de custo é que é possível dizer que há um custo, cuja magnitude, está relacionada com a percentagem de consumo de actividades [Kaplan e Cooper, 1998].

O custeio por actividades implica a adopção de uma nova forma de pensar e, principalmente das questões a responder pelo sistema de custeio da organização [Hicks, 1999]. Enquanto os sistemas tradicionais de custeio, procuram determinar a que elementos do projecto deverão ser imputados os custos, o ABC propõe novos desafios [Cokins, 2001]:

- Conhecer as actividades do projecto realizadas com os recursos da organização;

- Quantificar os recursos usados nas actividades e nos processos do projecto;

- Conhecer as razões que legitimam a execução das actividades e dos processos;

- Quantificar o consumo de actividades por produto e cliente da organização.

O sistema ABC, depois de implementado, proporciona estas informações, relacionando os custos da organização com as actividades que executa, permitindo, assim, determinar a origem dos custos, quais as actividades executadas, quantificar o consumo dos recursos e, ainda, verificar de que modo

estas actividades estão relacionadas com os proveitos da organização [Kaplan e Cooper, 1998]. O ABC é, portanto, um formato de custeio que presta informações sobre o valor das actividades, monitoriza o consumo de recursos e quantifica as actividades, de modo a permitir a actuação sobre as actividades com um impacto expressivo nos custos da organização.

Este conhecimento permite eliminar actividades ou tarefas redundantes, optimizar processos, estabelecer novas formas de relacionamento, redefinir a prioridade dos clientes, abandonar determinadas opções técnicas, e definir novas estratégias para o futuro [Gunasekaran et al., 2000].

A utilidade do custeio por actividades não se limita, porém, ao custeio dos produtos, é sobretudo, uma ferramenta de gestão dos custos [Cokins, 2001]. A adaptação dos sistemas de informação das organizações para a aplicação do modelo ABC torna a gestão do negócio mais fácil, uma vez que as decisões passarão a assentar em informações sobre a situação real da organização [Kaplan e Cooper, 1998].

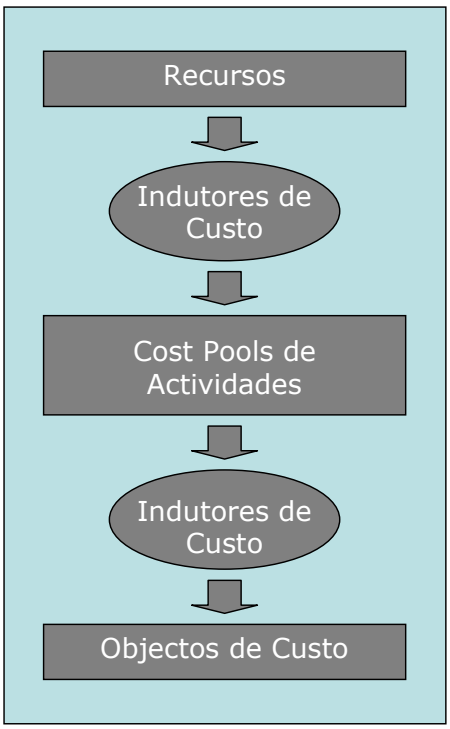

Figura 24: Primeira Versão do ABC
(Adaptado de [Cokins, 2001])

Apesar de os seus princípios básicos serem conhecidos há décadas, o modelo conceptual do sistema de custeio por actividades, somente viu a sua primeira versão desenhada no final dos anos 80 [Hicks, 1999].

A primeira abordagem apresenta uma visão funcional do custeio dos produtos – mede o custo por grupos de actividades (Cost Pools), os recursos gastos e a performance dos objectos de custo – sem no entanto, descriminar os custos das actividades individuais, consumidas pelos objectos de custo [Cokins, 2001] (ver Figura 24). Esta versão, também não conseguiu dar resposta às exigências colocadas pela introdução do conceito da qualidade total nas organizações, que defende uma estruturação dos negócios por processos [Cokins, 2001].

A fim de dar resposta à necessidade de análise dos processos de negócio, o ABC evoluiu para a gestão por actividades (ABM) [Kaplan e Cooper, 1998].

Esta viragem veio permitir às organizações considerarem nas suas decisões, o valor acrescentado pelas actividades e, em simultâneo, o esforço despendido com actividades que não produzem valor [Cokins, 2001]. O ABC foi, assim, redesenhado para proporcionar (1) uma visão dos custos e (2) uma visão dos processos da organização [Cokins, 2001] (ver Figura 25).

A primeira – a componente vertical do modelo – é usada para responder às necessidades identificadas na primeira versão do ABC, proporcionando informações mais precisas sobre os custos de concepção, planeamento e execução do produto [Kaplan e Cooper, 1998].

A segunda – a componente horizontal do modelo – procura, com base nas informações recolhidas, identificar oportunidades para melhorar os processos, permitindo aos gestores obter uma percepção dos problemas operacionais, relacionados com a criação de valor [Kaplan e Cooper, 1998].

4.2. Indutores de Custo

O ABC recorre a indutores de custo (ver Figura 25) para estabelecer uma relação causal, entre a medida do custo das actividades, e a medida do custo dos objectos de custo, o que lhe permite determinar qual o valor adicionado por cada uma das actividades ao produto, e quanto custa a sua execução. Os indutores de recursos medem a utilização dos recursos pelas actividades e, apresentam-se,

quase sempre, sob a forma de um indicador não financeiro [Kaplan e Cooper, 1998].

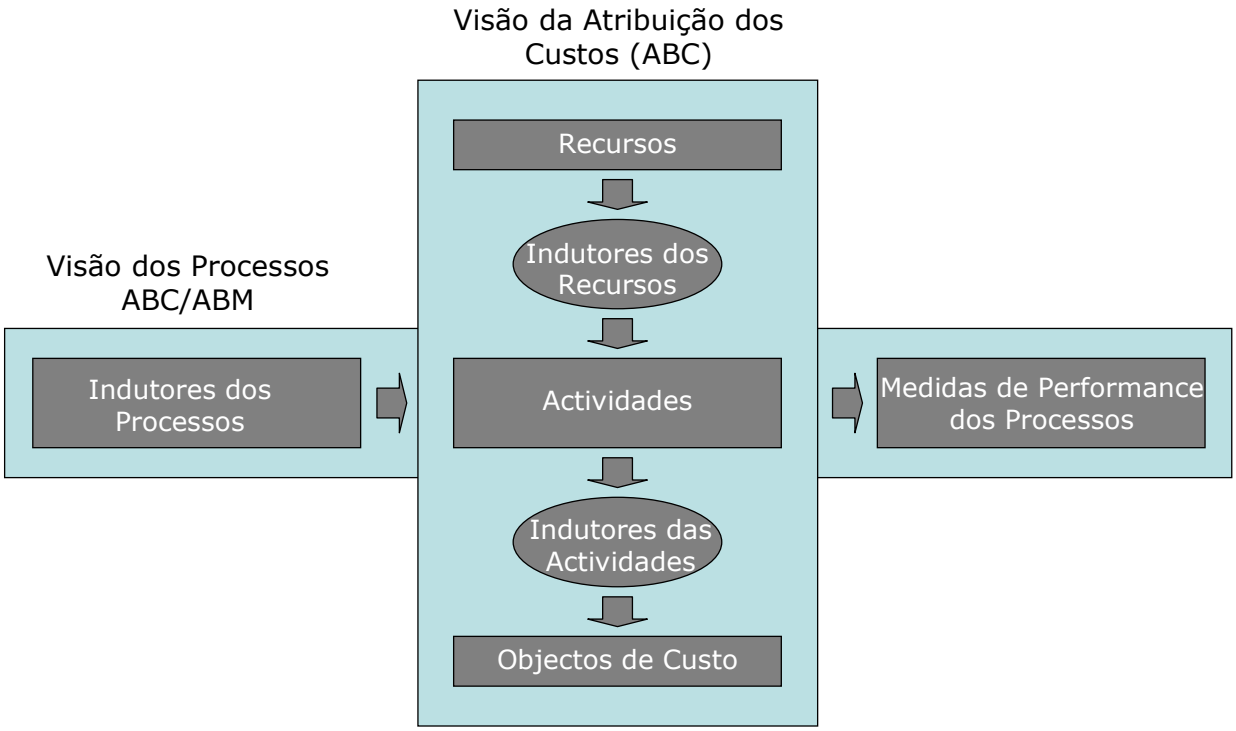

Figura 25: Segunda versão do ABC
(Adaptado de [Cokins, 2001])

A identificação e a medição dos indutores de recursos, assim como dos indutores de actividades, são de extrema importância para o sucesso da implementação do ABC [Cokins, 2001]. Na sua definição, é considerado o custo da medição e a facilidade na obtenção dos dados, pois, ainda que um número elevado de indutores garanta uma maior precisão na informação gerada, acarreta custos superiores; por outro lado, a simplificação dos indutores pode causar distorções no apuramento dos custos das actividades e dos objectos de custo [Cokins, 2001].

4.3. Orçamentação por Actividades (ABB)

O ABB não é mais do que o ABC invertido [Kaplan e Cooper, 1998]. Enquanto o ABC atribui os recursos às actividades e, posteriormente, aos objectos de custo, num fluxo vertical de cima para baixo, o ABB inverte as relações causais do

modelo ABC e estabelece um fluxo de baixo para cima [Kaplan e Cooper, 1998] (ver Figura 26).

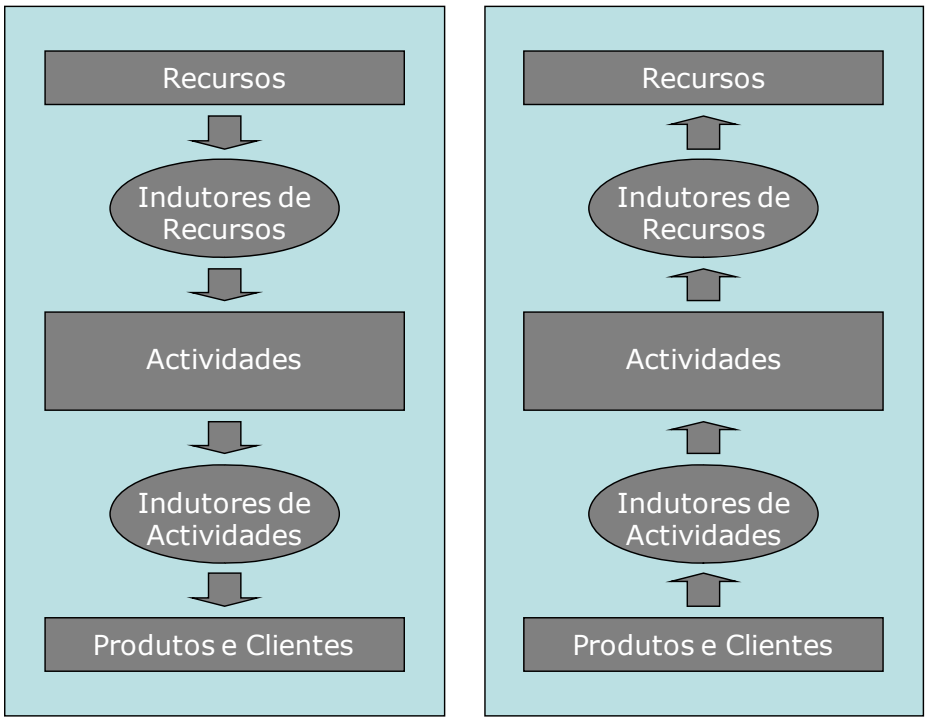

Figura 26: Inversão das Relações Causais do ABC
(Adaptado de [Kaplan e Cooper, 1998])

O ABB utiliza o modelo ABC como estrutura de custos, para representar as actividades necessárias à execução do produto [Brimson e Antos, 1999].
É usado durante o planeamento dos produtos, para atribuir os recursos necessários à execução das actividades, eliminando desperdícios e garantindo a manutenção da capacidade produtiva da organização [Kaplan e Cooper, 1998]. O ABB proporciona deste modo a oportunidade de controlar os recursos, mas principalmente de planear a sua procura, proporcionando aos gestores as informações necessárias para adquirir apenas os recursos necessários à execução das actividades [Kaplan e Cooper, 1998].
O ABB permite à organização construir um plano integrado, com uma visão clara, do modo como serão consumidas as actividades e os recursos, na prossecução das metas estabelecidas para o produto. Esta sistematização de informações

sobre o consumo de recursos permite aos gestores identificar a escassez ou o excesso de capacidade [Brimson e Antos, 1999].

Ao estimar o nível de actividade futura e a redução dos custos pretendida, o ABB contribui para aumentar a eficiência da aplicação dos recursos, fazendo com que o orçamento funcione como um mecanismo disciplinador do planeamento [Brimson e Antos, 1999].

O orçamento é aplicado com diferentes graus de complexidade e detalhe, adaptando-se aos diferentes níveis de conhecimento da organização, incorporando, sempre que possível, a opinião das pessoas sobre as metas orçamentais a estabelecer para os produtos, de forma a comprometê-las com os resultados a alcançar [Evans, 2004].

4.4. Gestão de Projectos e ABC/ABB

As organizações que fazem dos projectos de software o seu negócio, começam a sentir a necessidade de implementar um conjunto de actividades específicas, que lhes permitam obter um maior detalhe dos custos e uma visão mais clara dos processos de produção do software [Cokins, 2001; Armour, 2004].

Uma possível solução passa pela utilização do ABC/ABB, para efectuar o planeamento e controlo das actividades relacionadas com o projecto [Fichman e Kemerer, 2001; Ooi e Soh, 2003; Tuan et al., 2006].

O ABB permite identificar os recursos atribuídos a cada actividade e o modo como estas contribuem para maximizar o valor do produto [Cokins, 2001]; centra a atenção do gestor no custo das actividades executadas por produto, utilizando um critério de causa-efeito para identificar os indutores de custo [Kaplan e Cooper, 1998]. O ABB é, portanto, uma expressão quantitativa das actividades esperadas pela organização [Brimson e Antos, 1999].

A contribuição do ABB reside principalmente nas informações que propicia ao gestor, e na clarificação do impacto que as diversas actividades terão nos resultados económicos da organização. Esta informação é particularmente útil para planear e calcular os custos dos projectos [Fichman e Kemerer, 2001; Tuan et al., 2006].

É mais simples reagir a um orçamento baseado em actividades, pois exprime o trabalho que as pessoas desenvolvem. Com esta informação, o gestor pode

decidir a melhor forma de usar o dinheiro da organização e quais os resultados a alcançar [Brimson e Antos, 1999].
A implementação do ABB traz outras vantagens, designadamente:

- Coloca nas pessoas a responsabilidade de gestão das respectivas actividades com vista a alcançarem a performance delineada [Succi et al. 2000];

- Permite discernir quais as causas da variação dos outputs dos processos [Cokins, 2001];

- Permite compreender como é que os produtos criam uma demanda de actividades específicas que, por sua vez, requerem recursos [Ooi e Soh, 2003];

- Torna mais visível o resultado da gestão do excesso, ou deficiência, de capacidades dos recursos [Brimson e Antos, 1999];

- Permite estabelecer prioridades e eliminar actividades supérfluas [Kaplan e Cooper, 1998; Cokins, 2001];

- Monitoriza e avalia continuamente a performance dos projectos [Cokins, 2001].

4.4.1. Identificação dos Processos e Actividades

O processo ABB inicia-se com a estimativa do volume e da combinação de produtos (objectos de custo) esperados. As estimativas poderão incluir não apenas os produtos que serão produzidos, mas também os clientes (ou tipos de clientes) [Kaplan e Cooper, 1998].
Os métodos de estimativa da procura das actividades organizacionais no ABB são idênticos aos utilizados no planeamento tradicional do orçamento. A diferença reside somente na abrangência – o ABB alarga o orçamento tradicional, incluindo também o consumo de todas as actividades indirectas [Emblemsvåg, 2003].

Segue-se a identificação das actividades e processos, com vista a criar uma representação da organização, que documente a sequência de fases executadas pelas diferentes áreas funcionais envolvidas nos projectos. Estes dividem-se em processos e sub processos, que se dividem em actividades numa estrutura em árvore [Cook, 2005] (ver Figura 27).

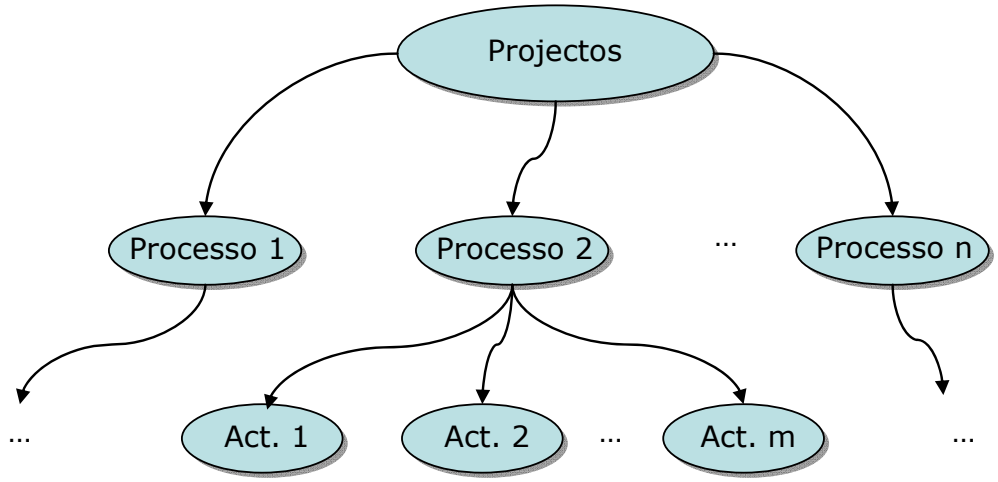

Figura 27: Estruturação em Árvore de Projectos, Processos e Actividades

Na identificação dos processos e das actividades, podem ser usadas técnicas específicas, como entrevistas, pesquisas sobre a documentação existente, emprego de questionários ou a observação das práticas de trabalho [Report #95-139, 1995; Moore, 2000; Fichman e Kemerer, 2001; Emblemsvåg, 2003]. É importante compreender todo o encadeamento dos processos, porém, as actividades a detalhar dependem, certamente, do fim a que se destina a informação.

Na definição dos processos e no levantamento das actividades, é importante dar atenção ao nível de detalhe das informações, pois dispor de poucos detalhes é tão prejudicial como tê-los em excesso [Cokins, 2001].

4.4.2. Identificação dos Recursos

O conhecimento do consumo esperado de actividades, permite ao gestor estimar os recursos necessários para sustentar a execução das actividades pretendidas.

Conseguida uma representação da organização por projectos, processos e actividades, é essencial medir a sua performance. Mas primeiro é preciso

identificar e medir os recursos consumidos, para poder determinar qual a relação entre os recursos e as actividades. Estes dados podem ser conseguidos com a realização de entrevistas com os gestores de projecto ou analisando os sistemas de informação da organização [Kaplan e Cooper, 1998; Brimson e Antos, 1999; Cokins, 2001]. No entanto, nem sempre o nível de detalhe dos dados recolhidos é o desejável, o que pode provocar atrasos na compilação da informação [Cokins, 2001].

O processo de orçamentação converte a procura de actividades estimada, em estimativas de recursos [Brimson e Antos, 1999]. Após a identificação dos recursos gastos por actividade, é então, determinada a sua capacidade prática, isto é, a taxa máxima efectiva que a organização poderá disponibilizar por actividade e projecto [Kaplan e Cooper, 1998].

A indivisibilidade da capacidade de alguns dos recursos resulta, geralmente, numa oferta que ultrapassa a procura estimada. Para solucionar este problema o sistema ABB trabalha de cima para baixo (Figura 26), para identificar os elementos do processo que restringem a utilização máxima da capacidade dos recursos [Kaplan e Cooper, 1998].

4.4.3. Relacionamento entre Recursos e Actividades

Apurados os valores dos recursos empregues nos projectos e respectivos processos é, então, possível relacioná-los com as actividades, mediante os indutores de recursos [Kaplan e Cooper, 1998; Brimson e Antos, 1999; Cokins, 2001]. A Figura 28 apresenta uma matriz para recolha de dados, durante o processo de levantamento do custo das actividades [Roztocki et al., 1999; Gerlach et al., 2002].

Estes dados podem, no entanto, ser registados directamente da matriz QFD – Recursos *versus* Processos/Actividades [Gonzalez et al., 2005] (ver Figura 28).

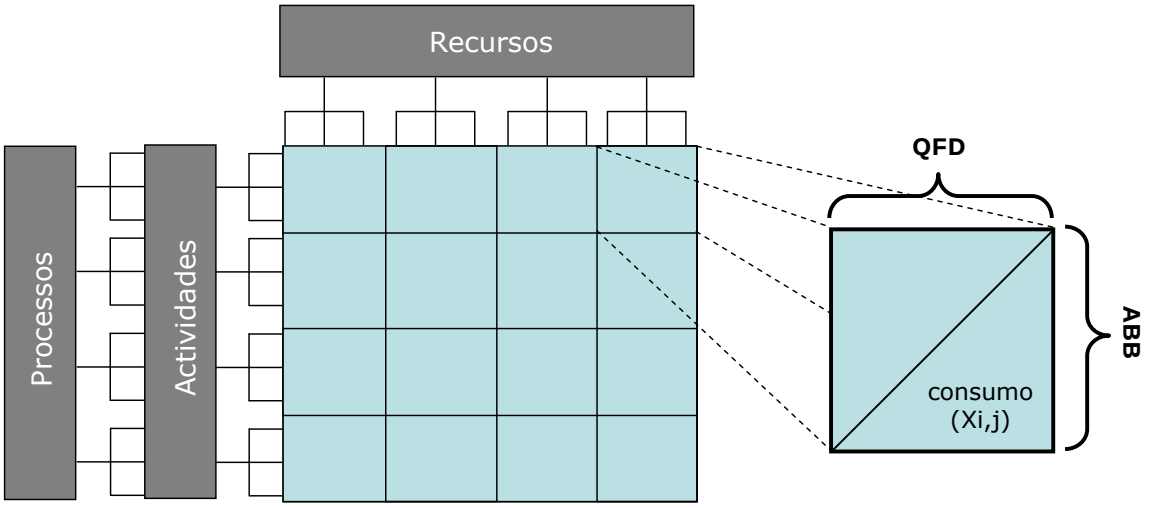

Figura 28: Matriz de Recursos *versus* Actividades
(Adaptada de [Roztocki et al., 1999])

As células da matriz central (Xi,j) contêm dois campos – um destinado ao registo da correlação entre os processos/actividades (i=1,…,n) e os recursos (j=1,…,m) e outro campo para registar o consumo esperado de recursos, por parte dos processos/actividades.

4.4.4. Identificação dos Objectos de Custo

Os objectos de custo da organização consistem, geralmente, nos seus produtos ou subprodutos. Contudo, também podem ser considerados como objectos de custo os clientes ou fornecedores [Cokins, 2001].

Na perspectiva de uma estratégia de outsourcing, o custo da gestão de um fornecedor também pode constituir um dado precioso. Assim, na definição dos objectos de custo, devem-se considerar todos os interessados nas informações relativas aos custos, como a administração, os gestores, etc. [Cokins, 2001].

4.4.5. Relacionamento entre Actividades e Objectos de Custo

Uma vez identificados os objectos de custo, é então possível atribuir-lhes os custos das actividades, por meio de indutores de custo por actividade [Kaplan e Cooper, 1998; Brimson e Antos, 1999; Cokins, 2001].

A repartição dos custos de uma actividade entre os objectos de custo faz-se com base no consumo de recursos por essa actividade, e não segundo o volume dos objectos de custo [Kaplan e Cooper, 1998].

Assim como na definição dos indutores de recursos, a escolha dos indutores de actividades também deve ser racional, para que a medição não se torne desgastante ou onerosa [Cokins, 2001].

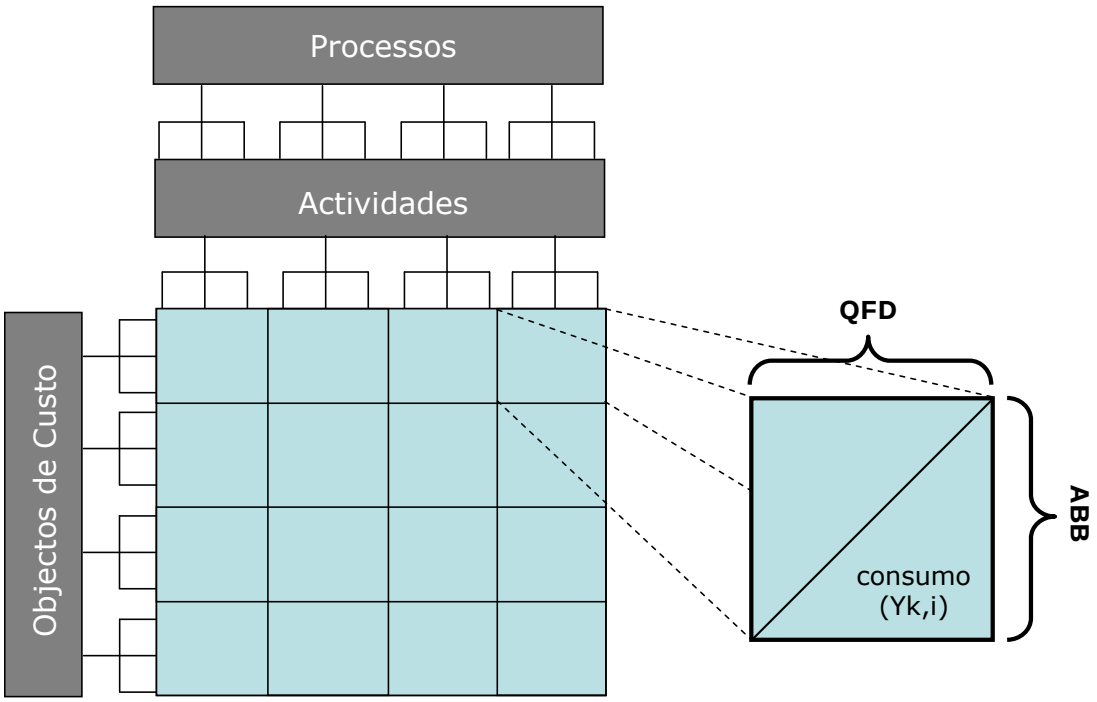

Figura 29: Matriz de Actividades *versus* Objectos de Custo
(Adaptada de [Roztocki et al., 1999])

As técnicas de recolha de informação (entrevistas, análise de documentos, questionários, etc.) usadas na identificação dos processos e das actividades podem ser usadas nesta fase, para criar uma matriz de actividades *versus* objectos de custo (ver Figura 29).

As células da matriz (Yk,i), para além de demonstrarem, ou não, a existência de uma correlação entre as actividades (i=1,…,n) e os objectos de custo (k=1,…,o), exibem o consumo de actividades pelos objectos de custo.

Na Figura 29, pode-se visualizar a matriz usada no levantamento do custeio dos objectos de custo. As colunas da matriz representam as actividades e as linhas representam os produtos de software (requisitos e subsistemas). Quando um

produto consumir determinada actividade, a célula que revela esta relação deverá ser marcada com um marcador numérico que traduza o consumo de actividades pelos objectos de custo.

4.5. Estimativa de Custos com ABC/ABB

A informação contida na matriz de recursos *versus* processos/actividades é o ponto de partida para a construção de um modelo de cálculo dos custos do software. A única operação requerida é a da multiplicação de matrizes, que tem como única condição, que o número de colunas da primeira matriz seja igual ao número de linhas da segunda, o que ocorrerá sempre, pois se for considerado um sistema de $m_x n$ equações:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \ldots \quad \ldots \quad \ldots \quad \ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m \end{cases}$$

A sua representação em termos matriciais será:

$$\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{m1} & a_{m2} & & a_{mn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \ldots \\ b_m \end{bmatrix}$$

Se a primeira matriz (A), for de $m_x n$ e a segunda (X), de $n_x 1$, então o seu produto será uma matriz de $m_x 1$, traduzida por $A \cdot X$:

$$[A \cdot X]_i = [B_i] = \sum_{k=1}^{n} a_{ik} \cdot x_k = a_{i1} \cdot x_1 + a_{i2} \cdot x_2 + \ldots + a_{in} \cdot x_n \quad (1 \le i \le m)$$

Os custos por actividade resultam da multiplicação da matriz recursos *versus* actividades, pela matriz de custos dos recursos, a saber:

$$CA_i = \sum_{j=1}^{m} X_{i,j} \cdot CR_j$$

Onde:

- CRj = Matriz de custos dos recursos disponíveis, j=1,…, m;

- Xi,j = Percentagem de recursos (j), consumidos pela actividade (i), i=1,…,n, j=1,…, m.

Após o apuramento dos custos por actividade, pode-se dar início à fase seguinte do processo ABC – o apuramento do custo dos objectos de custo.
Os custos por objecto de custo resultam da multiplicação da matriz actividades *versus* objectos de custo, pela matriz de custos por actividade, obtida na fase anterior.

$$CO_k = \sum_{i=1}^{n} Y_{k,i} \cdot CA_i$$

Onde:

- CAi = Matriz de custos das actividades disponíveis, i=1,…, n;

- Yk,i = Percentagem de actividades (i), consumidas pelos objectos de custo (k), i=1,…,n, k=1,…,o.

O cálculo da matriz (COk) processada em duas fases pode ser realizado num só passo. Se a matriz (Zk,j) representar a quantidade do recurso (j), consumido pelo objecto de custo (k), então, o produto das (Yk,i) e (Xi,j), dá lugar a uma nova matriz de (k) linhas e (n) colunas: a matriz recursos *versus* objectos de custo (Zk,j); não esquecendo, no entanto, que os recursos primeiro são consumidos pelas actividades e depois pelos objectos de custo.

$$Z_{k,j} = \sum_{j=1}^{m} Y_{k,i} \cdot X_{i,j}$$

Onde:

- Zi,j = Percentagem de recursos (j), consumidos pelo objecto de custo (k), j=1,…,m, k=1,…, o;

- Xi,j = Percentagem de recursos (j), consumidos pela actividade (i), i=1,…,n, j=1,…, m;

- Yk,i = Percentagem de actividades (i) consumidas pelos objectos de custo (k), i=1,…,n, k=1,…,o.

Então:

$$CO_k = \sum_{j=1}^{m} Z_{k,j} \cdot CR_j = \sum_{j=1}^{m} \sum_{i=1}^{n} Y_{k,i} \cdot X_{i,j} \cdot CR_j$$

Assim, construído o modelo ABC/ABB e estabelecidas as relações entre os recursos e as actividades, e entre estas e os objectos de custo, será somente necessário recolher os dados sobre a actividade produtiva, ou seja o volume dos indutores, e colocá-los na forma matricial, isto é, para se obter uma estimativa dos custos dos objectos resultantes do projecto, somente é preciso preencher as matrizes (Yk,i) e (Xi,j).
Estas mesmas matrizes podem ser usadas para recolher os custos dos objectos de custo, durante as várias fases atravessadas pelo desenvolvimento do produto, ou seja, o mesmo sistema pode ser usado para estimar os custos e determinar os custos reais dos produtos, daí a razão de o modelo de custos se designar globalmente por ABC/ABB.

4.6. Problemas do ABC/ABB

A crescente oferta de produtos e a diversidade de clientes dificulta muito o processo de modelação dos custos, apesar de muitas organizações conseguirem desenvolver com sucesso sistemas de gestão, custeio e orçamentação por actividades [Anderson et al., 2002].
Os modelos ABC/ABB estabelecem uma relação directa entre os custos de execução das actividades e os produtos, procurando aumentar a precisão da informação gerada, em relação aos sistemas tradicionais de custeio, incorporando um certo número de indutores de recursos e de custo [Cokins, 2001].

O ABC torna-se mais exacto à medida que as actividades são subdivididas e refinadas. Mas este processo, conduz ao crescimento do número de actividades e, em consequência, a um aumento do tempo de desenvolvimento do sistema, isto é, o refinamento do sistema de custo, provoca um aumento dos custos, do próprio sistema [Anderson et al., 2002; Max, 2005]. Assim, nos ambientes mais complexos e dinâmicos, as organizações começaram simplesmente a abandonar o ABC dadas as dificuldades encontradas para representar a complexidade das suas operações – a implementação demorava muito tempo e possuía um custo elevado [Kaplan e Anderson, 2004].
Com o objectivo de resolver estes problemas Kaplan e Anderson [2004] introduziram o conceito Time-Driven ABC (TDABC).

4.7. Time-Driven ABC (TDABC)

O TDABC possui a capacidade de identificar e reportar as transacções complexas de uma forma simples, recorrendo a equações do tempo para conseguir moldar a complexidade actual das actividades de produção [Bruggeman et al., 2005].
O conceito do TDABC foi desenvolvido originalmente em 1997, por Steven Anderson, que o aplicou na sua empresa, a Acorn Systems, em 2001, onde, em equipa com Robert Kaplan da Harvard Business School, o aperfeiçoou para conseguir moldar a complexidade actual das actividades de produção [Bruggeman et al., 2005].
Este sistema requer a estimativa de apenas dois indutores: o custo unitário dos recursos consumidos (indutores de recursos) e as unidades de tempo necessárias para executar os vários eventos das múltiplas actividades (indutores de duração) [Kaplan e Anderson, 2004].
A inovação do TDABC reside na estimativa do tempo necessário para a execução dos eventos de uma actividade, de modo a reflectirem as diferentes características das actividades. O TDABC usa indutores de duração (por exemplo, horas/homem), em vez de indutores da transacção (por exemplo, linhas de código ou pontos de função implementados).
Em ambientes muito competitivos, uma actividade particular, nem sempre consome a mesma quantidade de recursos, para as mesmas situações. Em vez de definir uma actividade para cada situação, a aproximação TDABC estima a

procura de recursos mediante uma equação temporal. O TDABC é implementado em seis fases [Kaplan e Anderson, 2004]:

- Identifica os vários recursos necessários à execução das actividades;
- Estima o custo de cada recurso;
- Estima a capacidade em unidades de tempo, para cada um dos recursos (por exemplo, número de horas de trabalho);
- Calcula o custo unitário dos recursos (em unidades de tempo), dividindo o custo total de cada um dos recursos pela sua capacidade prática;
- Determina o tempo consumido por cada evento das actividades;
- Multiplica o custo unitário dos recursos pelo tempo consumido pelos eventos associados a cada objecto de custo, a fim de determinar o seu custo total.

Em seguida é apresentado um exemplo ilustrativo da formulação do TDABC, adaptado de [Kaplan e Anderson, 2004].

4.7.1. Exemplo Ilustrativo

Considere-se um departamento de vendas, em que foi implementado o ABC, em que é preciso estimar mensalmente o tempo gasto com três actividades: registo de um novo cliente, entrada de ordens e a expedição de ordens urgentes. Com a abordagem TDABC, a equipa do ABC começa por estimar o tempo que é preciso para executar cada uma das actividades. No presente exemplo, o registo de um novo cliente demora 15 minutos a executar e, por cada ordem registada, são dispendidos cerca de 5 minutos, acrescidos de 3 minutos, por cada item que compõe a ordem; as ordens para entrega urgente resultam em 10 minutos adicionais, necessários para coordenação do processo de expedição.

As três actividades podem ser congregadas numa só, definida pela seguinte equação:

$$\text{tempo de processamento de ordens} = 5 + 15 \cdot [\text{novo cliente}] + 3 \cdot [\text{número de itens}] + 10 \cdot [\text{pedido urgente}]$$

O custo da actividade é determinado pela multiplicação do tempo de processamento, pelo custo unitário dos recursos, consumidos por esta actividade.

A principal vantagem do TDABC, é considerar vários indutores na definição do custo de uma actividade, por contraponto ao ABC tradicional, que somente considera um indutor por actividade. Se forem necessários vários indutores de actividade para um custeio mais exacto, o ABC tradicional precisará de introduzir diferentes actividades para detalhar o modelo (registo de clientes, entrada de ordens e a expedição de ordens urgentes).

4.8. Estimativa de Custos com TDABC/TDABB

O custo de uma actividade, no contexto do TDABC, é uma função da duração da actividade e do custo. No exemplo ilustrado, o custo de execução do processamento de ordens é calculado em função das características de três indutores de tempo específicos: (1) registo de um novo cliente (2) registo de itens e (3) tratar do pedido com urgência.

Assim, o custo de um evento (l), da actividade (i), é dado por ($CA_{i,l} = T_{i,l} . CA_i$), onde (CAi) é o custo da actividade (i), por unidade de tempo, determinado anteriormente, e (Ti,l) o tempo consumido pela actividade (i), para um evento (l) [Bruggeman et al., 2005].

Se o custo unitário da actividade (i), que consome (m) recursos (j), é dado por:

$$CA_i = \sum_{j=1}^{m} X_{i,j} \cdot CR_j$$

O custo de uma actividade resulta, então, da soma dos custos de (p) eventos, ou seja, o custo por actividade (i), para o evento (l), passa a ser dado por:

$$CA_{i,l} = T_{i,l} \cdot CA_i = \sum_{j=1}^{m} T_{i,l} \cdot \left(X_{i,j} \cdot CR_j\right)$$

O custo total de um objecto do custo pode, assim, ser calculado mediante a equação:

$$CO_k = \sum_{l=1}^{p} \sum_{i=1}^{n} T_{k,i,l} . CA_i = \sum_{l=1}^{p} \sum_{j=1}^{m} \sum_{i=1}^{n} T_{k,i,l} \cdot \cdot \left(X_{i,j} \cdot CR_j\right)$$

Onde:

- CAi = Custo unitário da actividade (i);
- Tk,i,l = Tempo consumido pelo evento (l), da actividade (i), na execução do objecto de custo (k);
- CRj = Custo por unidade do tempo (horas ou minutos) do recurso (j);
- Xi,j = Consumo de recursos (j) pela actividade (i);
- m = Número de recursos disponíveis;
- n = Número de actividades executadas;
- p = Número de eventos da actividade (i);
- COk = Custo do objecto (k), que consome (p) eventos da actividade (i), que por sua vez, consome (Xi,j) recursos (j), que têm um custo (CRj).

Com estas equações, o tempo consumido por uma actividade (Ti,l), pode ser expresso em função de diferentes características (indutores de duração).
O tempo necessário para um evento (l), da actividade (i), com (q) indutores de tempo, é dada pela seguinte equação [Bruggeman et al., 2005]:

$$T_{i,l} = \beta_0 + \beta_1 \cdot X_1 + \cdots + \beta_p \cdot X_q$$

Onde:

- Ti,l = Tempo consumido pelo evento (l), da actividade (i);
- B0 = Tempo consumido pela actividade (i), independentemente das características do evento (l);
- β1 = Consumo de tempo por cada unidade do indutor de tempo (X1);
- X1 = Indutor de tempo número 1, Xq = indutor de tempo (q);
- l =1,…,p.

No exemplo ilustrado anteriormente,

$$T_{\text{proc. ordens}}, l = 5 + 15 \cdot X_1 + 3 \cdot X_2 + 10 \cdot X_3$$

Em que (B0=5) é uma constante e (X1, um novo cliente) é um indutor de tempo indicador (0-1), (X2, o número de itens), é um indutor discreto e (X3, um pedido urgente) é outro indutor do tipo indicador.
A actividade de processamento de ordens possui quatro eventos, a saber:

- $T_{\text{proc. ordens}}, 1 = 5 + 3 \cdot X_2$: É um pedido normal e somente é necessário recolher informação relativamente aos itens que constituem a ordem de encomenda;
- $T_{\text{proc. ordens}}, 2 = 5 + 3 \cdot X_2 + 10 \cdot X_3$: Pedido efectuado por um cliente já registado, mas que solicita urgência no seu processamento;
- $T_{\text{proc. ordens}}, 3 = 5 + 15 \cdot X_1 + 3 \cdot X_2$: É um pedido de um novo cliente, que é necessário registar e preencher a respectiva ordem;

- $T_{proc.\ ordens}, 4 = 5 + 15 \cdot X_1 + 3 \cdot X_2 + 10 \cdot X_3$: Trata-se de um novo cliente, que efectua um pedido com urgência;

As equações do tempo podem considerar ainda as interacções entre indutores (por exemplo, a urgência de uma ordem pode limitar o número de itens associados) [Bruggeman et al., 2005]:
No TDABC o número de indutores do tempo (ou o número de termos na equação do tempo) não tem quaisquer restrições [Kaplan e Anderson, 2004].

4.8.1. Relacionamento entre Processos/Actividades e Objectos

A utilização do TDABC/TDABB obriga a uma modificação da matriz de recolha de dados (ver Figura 30); na matriz de actividades *versus* objectos de custo, são registados os eventos, despoletados pelas diversas actividades, para a execução dos objectos de custo.
As células da matriz (k,i), para além de demonstrarem, ou não, a existência de uma correlação entre as actividades (i=1,…,n) e os objectos de custo (k=1,…,o), registam o somatório dos eventos das actividades consumidas pelos objectos de custo (∑Tk,i,l), em que l=1,…,p.

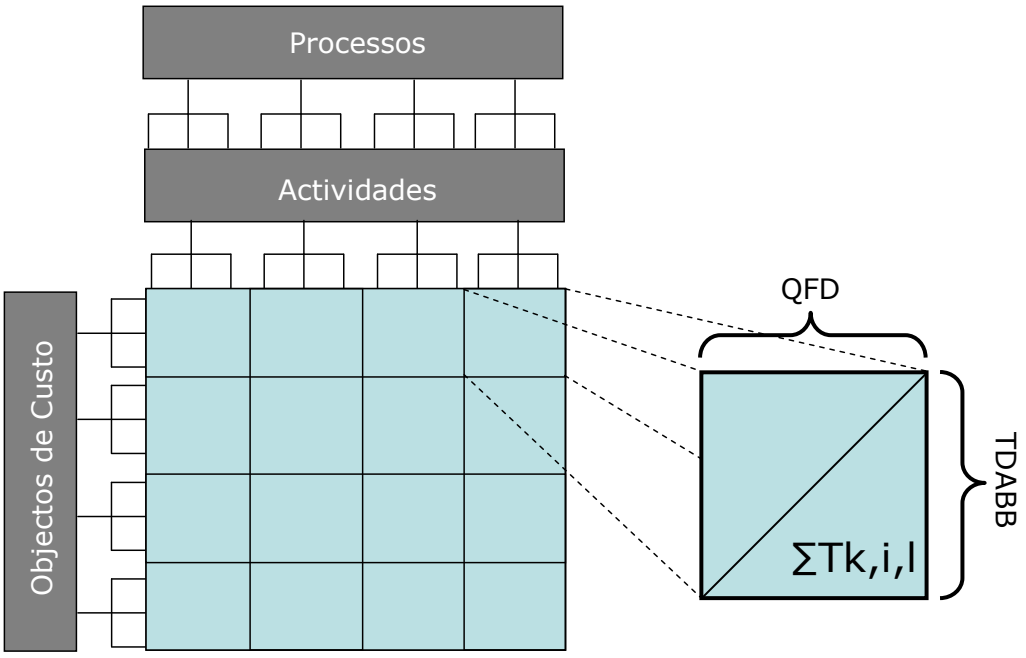

Figura 30: Matriz de Actividades *versus* Objectos de Custo
(Quando é usado o Processo Time-Driven ABB)

4.9. Conclusões do Capítulo

Neste capítulo considerou-se o ABC/ABB como resposta ao problema do controlo dos custos dos projectos de software, principalmente pela capacidade que possui, de identificar os custos indirectos do software que, considerando o crescimento de componentes de software reutilizáveis, ou passíveis de aquisição a terceiros, para depois serem integrados nos produtos a desenvolver, se transformou numa das principais preocupações das organizações produtoras de software.

Mas esta não é a única vantagem do ABC/ABB, pois esta abordagem, também responde ao problema do controlo da produtividade, uma vez que, associa às actividades conduzidas durante o projecto, os recursos necessários, permitindo subsequentemente, verificar a performance das actividades e identificar eventuais subaproveitamentos de recursos, isto é, na prática o ABC/ABB constitui, um sistema de controlo da produtividade dos projectos de software.

A gestão por actividades, integrada com o desdobramento da qualidade, dá a conhecer as actividades que têm um papel mais activo na concretização dos requisitos do software. Esta informação é muito útil ao gestor de projecto, no momento em que tem de seleccionar onde deverá concentrar os esforços de monitorização e controlo e, bem assim, as actividades que poderão acarretar um risco maior.

Apesar de o ABC/ABB ter inúmeras vantagens, possui algumas debilidades, que o TDABC/TDABB vem colmatar, procurando reduzir o número de actividades e de indutores, bem como reduzir o tipo de indutores usados, recorrendo apenas a indutores de tempo que registam o tempo gasto com os diferentes eventos das várias actividades. O conceito de evento na execução das actividades é uma novidade, mas relativamente simples – consiste na identificação das múltiplas variantes de uma actividade, expressas sob a forma de uma equação em unidades de tempo. A única exigência do TDABC/TDABB é que o custo dos recursos seja traduzido para unidades de tempo, usadas em todo o sistema, incluindo os objectos de custo, o que é consistente com a prática de gestão de projectos em muitas organizações, que registam o trabalho efectuado, em horas/homem.

No próximo capítulo ver-se-á como é possível integrar o QFD e o TDABB, com a Engenharia do Valor ou a Programação Linear, para redefinir a importância dos requisitos do software e das actividades, de modo a que a selecção das opções técnicas e de gestão reflictam, não só as preferências dos clientes, como também considerem os custos das diferentes soluções.

CAPÍTULO 5 – Framework de Integração

Até a este momento verificou-se que os modelos ABC/ABB, ou o TDABB/TDABB, que foram propostos, possuem pontos de convergência com o QFD, designadamente, a matriz de processos/actividades *versus* recursos e a matriz de objectos de custo (requisitos do software) *versus* processos/actividades. A convergência limita-se à utilização do mesmo instrumento, neste caso as matrizes QFD, (1) para recolher dados sobre as necessidades e expectativas dos clientes, bem como a correlação entre estas e as soluções propostas pelos engenheiros de software, e (2) os custos dessas soluções e actividades de suporte, mas sem qualquer relação causal.

Ao longo da construção das matrizes QFD, certamente que algumas das soluções propostas serão abandonadas, em detrimento de outras consideradas mais importantes. Assim sendo, numa primeira fase, a estimativa dos custos não precisa de ser muito elaborada, pois somente depois de decidido quais as melhores opções, é que a estimativa dos custos deve ser refinada; na primeira fase, as estimativas de custos poderão basear-se apenas na avaliação do gestor de projecto para, depois, numa segunda fase, criar uma WBS, que examine minuciosamente os componentes que compõem cada um dos requisitos do software.

Ainda assim, poderá acontecer que as estimativas de custos ultrapassem o orçamento previsto para o projecto, obrigando a que alguns dos requisitos sejam descartados, ou então, apenas possam ser implementados parcialmente.

Este capítulo aborda a problemática das limitações orçamentais, apresentando duas opções, que permitem incluir a ponderação dos custos estimados pelo processo ABB/TDABB nas matrizes QFD: a Engenharia do Valor, um método menos preciso, mas de aplicação muito simples; e um modelo de Programação Linear, um método mais recente, que tira partido do poder computacional disponível actualmente, para resolver sistemas de equações lineares mais complexas.

Actualmente, não há dúvidas de que o controlo dos custos possui uma importância primordial para o sucesso de qualquer projecto de software. O

conceito do valor ganho, isto é, a produtividade do projecto, tem vindo a ganhar espaço na gestão de projectos, e mais recentemente, na área do software, com reconhecidas vantagens. Seguindo essa tendência, é abordado, também neste capítulo, a gestão do valor ganho – EVM, para monitorização e controlo dos custos dos requisitos do software e das actividades do projecto.

5.1. QFD e os Custos do Software

O custo é, por regra, um ponto de grande preocupação na gestão de projectos de software. A introdução de limites aos custos, implica que a concepção do software, ambicione a satisfação do cliente, mas procure, também, optimizar o processo de desenvolvimento. É muito frequente que as decisões sobre a construção do software ou a implementação do processo obriguem o gestor de projecto a decidir que requisitos deverão ser modificados ou eliminados.

A performance de um produto tem por base, as funções que executa, o custo de implementação e o tempo que leva a desenvolver. Existem várias funções que podem ser adicionadas ao software, das quais algumas representam o valor primário do produto e outras o valor subsidiário [Fujita et al., 2003]. A um equilíbrio entre funções e custos está, sem dúvida, associado o sucesso de um projecto.

O QFD convencional é usado para maximizar a satisfação do cliente, através da identificação dos atributos do produto, que preenchem as expectativas do cliente e cruzando os requisitos do cliente com as características técnicas do software. O QFD estabelece, então, prioridades para os requisitos técnicos, de acordo com o impacto que terão nos requisitos do cliente, usando uma matriz de correlações. A fim de concretizar as metas estabelecidas, são atribuídos recursos ao desenvolvimento do produto, porém, as questões financeiras são, por regra, abordadas de uma forma subjectiva e adhoc [Tang et al., 2002; Kaldate et al., 2006]. Esta perspectiva do QFD, apenas considera o valor do software conseguido com o alcance das metas estabelecidas, contrariamente à perspectiva económica, que pondera o valor ganho por unidade monetária investida [Xie et al. 2003].

Segundo Bode e Fung [1998], Chen e Weng [2003], Xie et al. [2003], é preciso um modelo QFD mais avançado, que balanceie os custos e a satisfação do

cliente, pois as restrições orçamentais e de outros recursos, como a mão-de-obra especializada e o próprio tempo, limitam muito as funções que podem ser incorporadas no produto. O QFD convencional é, portanto, limitado tecnicamente, uma vez que, assume uma disponibilidade infinita de recursos para atingir as metas definidas, ignorando as restrições orçamentais reais.

5.2. Engenharia do Valor

Em conjunto com o QFD, poderá ser usada a Engenharia do Valor – VE, para estabelecer prioridades, para as actividades e requisitos do produto, tendo em consideração os respectivos custos [Akao, 1997, Beiter e Ishii, 1999, Fujita et al., 2003; Stenbeck e Svensson, 2004].

A Engenharia do Valor é usada para identificar os requisitos ou actividades relacionadas com o produto, que adicionam valor, com vista a criar produtos mais satisfatórios, ao mais baixo custo possível [Dimsey e Mazur, 2002].

É um método com provas dadas ao longo do tempo e conhecido mundialmente, permitindo às organizações reduzir os custos de um produto, mediante a procura de formas alternativas, mais eficientes, para a sua realização, sem comprometer a performance ou as funções desejadas [Sinbari e Hari, 1992].

Esta técnica potência a optimização do produto, permitindo visualizar a relevância, em valor, de cada função, requisito ou actividade, por meio de um índice de valor, que compara a importância relativa de cada item em análise (função, requisito, actividade, etc.) com o custo relativo do mesmo item [Stenbeck e Svensson, 2004]:

$$\text{Índice} = \frac{I_j}{C_j} \qquad (1)$$

Onde:

- Ij = Importância relativa do item (j);
- Cj = Custo relativo do item (j);

Os resultados deste índice são posteriormente inseridos num gráfico [Beiter e Ishii, 1999; Fujita et al., 2003] (ver Figura 31).

Este método pressupõe que há uma relação directa entre a relevância e o custo, e que, a razão entre o primeiro e o segundo é, idealmente, igual à unidade, isto é, que o custo de um determinado item deverá ser proporcional à sua relevância. As exigências de uma razão unitária são, no entanto, muito elevadas, pelo que, na prática são permitidas tolerâncias à linha que expressa a relação ideal entre a relevância e o custo [Stenbeck e Svensson, 2004] (ver Figura 31).

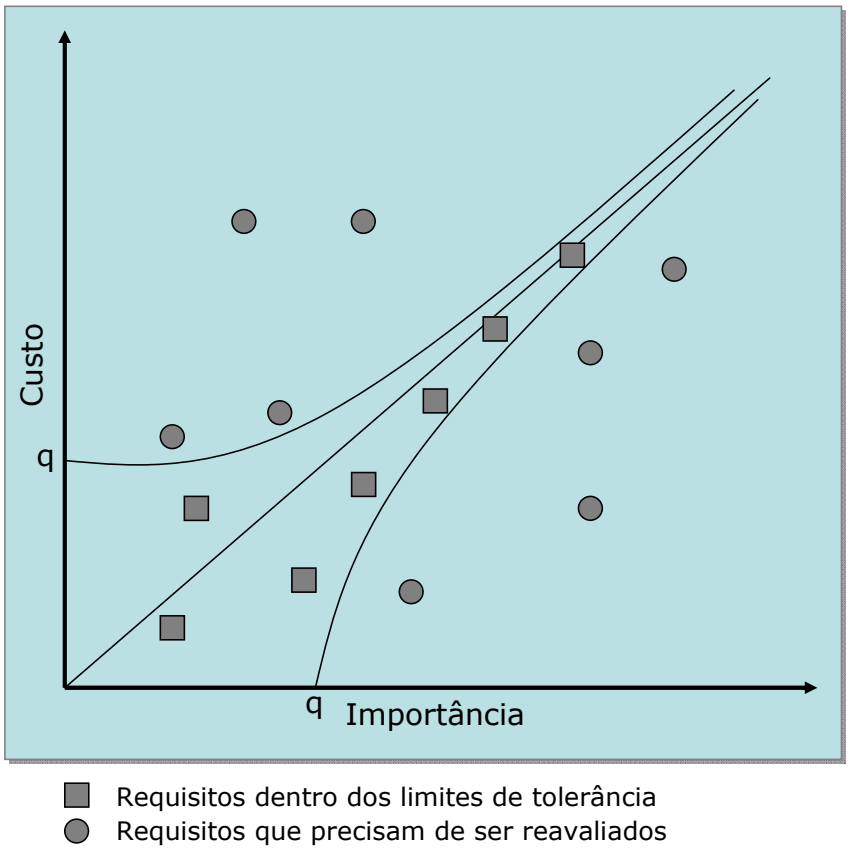

Figura 31: Custo *versus* Importância
(Adaptado de [Beiter e Ishii, 1999])

A área que delimita os índices, considerados aceitáveis, é definida pela aplicação de uma equação elaborada por Tanaka, em 1985 [Stenbeck e Svensson, 2004]:

$$CL = \sqrt{x^2 \pm q^2} \qquad (2)$$

O valor (q), é definido pelo gestor de projecto, todavia, um valor inferior a 20% do valor máximo relativo é, geralmente, considerado como adequado [Stenbeck e Svensson, 2004].

A relativização dos custos e da importância facilita a compreensão e a apreciação da distribuição do valor pelas funcionalidades do software e o gráfico que daí resulta permite a identificação de custos desproporcionados. Mas dado que, no início, o gráfico produzido tem por base estimativas vagas, é importante que esta análise seja refinada ao longo das sucessivas fases do projecto [Chao e Ishii, 2004].

5.3. Programação Linear

Wasserman, em 1993, a partir do trabalho desenvolvido por Lyman, em 1990, desenvolveu um método de definição de prioridades que, propunha a normalização das correlações, entre os requisitos do cliente e soluções técnicas, de modo a criar uma nova correlação, que incluísse a interdependência entre os requisitos técnicos, de acordo com a seguinte equação [Chen e Weng, 2003; Chen e Weng, 2006]:

$$R_{i,j}^{*} = \frac{\sum_{k=1}^{n} R_{i,k} \cdot \gamma_{k,j}}{\sum_{j=1}^{n}\sum_{k=1}^{n} R_{i,k} \cdot \gamma_{k,j}} \qquad (1)$$

Onde:

- $R_{i,j}^{*}$ = Correlação normalizada entre a função (i) e o requisito (j), em que i=1,…,m, j=1,…,n;

- $R_{i,k}$ = Correlação quantificada entre as funções (i) e os requisitos (j), em que i=1,…,m e k=1,…,n;

- $\gamma_{k,j}$ = Correlação quantificada entre os requisitos (k) e (j), em que k,j= 1,…,n.

O factor $R_{i,j}^{*}$ pode ser interpretado como a contribuição do requisito (j), para a implementação da função (i), quando as metas estabelecidas para o requisito (j) forem alcançadas [Tang et al., 2002]. Este modelo assume que as funções são independentes entre si, o que se aplica à maior parte dos casos práticos [Xie et al., 2003].

Wasserman propôs ainda a criação de um índice de valor, calculado com base na importância atribuída aos requisitos do cliente, na correlação normalizada e nos custos, para distribuir o orçamento atribuído ao projecto.

Com base na normalização de Wasserman, Xie et al. [2003], propõem um modelo de Programação Linear (LP), para solucionar o problema das decisões condicionadas por uma disponibilidade de recursos limitada, com vista a melhorar a qualidade do produto. O objectivo é determinar como o produto ou processo pode ser melhorado, mediante uma melhor distribuição dos recursos pelos diferentes requisitos do software ou actividades, para que a satisfação global do cliente seja a maior possível.

Considerando, por exemplo, as correlações entre as funções do software, desejadas pelo cliente, e os requisitos do software, o modelo de programação linear para a atribuição de recursos, pode ser formulado da seguinte forma [Xie et al., 2003]:

$$\text{Max } Z = \left[\sum_{i=1}^{n} I_i \cdot \sum_{j=1}^{m} R_{i,j}^{*} \cdot \frac{X_j}{C_j} \right] \qquad (2)$$

Onde:

- Z = Objectivo de satisfazer o cliente;
- Xj = Variável de decisão sobre os recursos a atribuir ao requisito (j), j=1,…,m;
- Ii = Importância da função (i), i=1,…,n;
- $R_{i,j}^{*}$ = Correlação normalizada entre a função (i) e o requisito (j), i=1,…,m, j=1,…,n;

- Cj = O custo estimado para a implementação do requisito de software (j), j=1,…,m.

Na equação (2):

- X_j / C_j = Nível de concretização das metas estabelecidas para o requisito (j);

- $\sum_j R_{i,j}^{*} \cdot X_j / C_j$ = Impacto que o requisito (j) tem na satisfação da função (i);

- $\sum_i Ii \cdot \sum_j R_{i,j}^{*} \cdot X_j / C_j$ = Impacto de todos os requisitos na satisfação global do cliente com as funções do produto.

5.3.1. Formulação de Restrições

Segundo Fung et al. [2003] e Xie et al. [2003], na prática existem algumas restrições que se colocam no processo de optimização do processo QFD. A primeira é desde logo a escassez dos recursos atribuídos ao projecto, apresentados sob a forma de estimativas orçamentais, isto é, o somatório das estimativas de custos, de todas as actividades ou requisitos do software, não deverá exceder o orçamento disponível.

$$\sum_j X_j \leq C \qquad (3)$$

Onde:

- $\sum_j X_j$ = Somatório dos custos dos recursos atribuídos à execução das actividades ou requisitos do software do projecto;

- C = O orçamento disponível para a execução do projecto.

Os clientes poderão exigir um nível mínimo de satisfação para um dado requisito, ou a própria organização pode especificar metas a alcançar para as actividades de desenvolvimento (metas de produtividade), considerando, ou não, a performance da concorrência mais directa.
Xie et al., [2003] propõem que seja incluído no modelo de Programação Linear uma restrição que, por exemplo, traduza os padrões mínimos de satisfação do cliente com uma determinada função do software:

$$Ii \cdot \sum_j R_{i,j}^{*} \cdot X_j / C_j \geq Si \qquad (4)$$

Onde:

- Si = Satisfação mínima com a função (i).

Fung et al. [2003] e Xie et al. [2003], propõem ainda que sejam impostos limites ao grau de cumprimento das metas estabelecidas, isto é, as restrições de alguns dos requisitos ou actividades, poderão limitar os valores das metas a alcançar. É recomendável, portanto, a definição de limites inferiores para as metas sempre que possível; podem ser usados valores sem dimensão, situados num intervalo de [αi, 1], de modo a representar a subjectividade dos analistas, restrições técnicas, a performance dos produtos concorrentes, etc., podendo ser definidos diferentes (αi), para cada um dos requisitos [Fung et al., 2003].

$$\alpha_i \leq \frac{X_j}{C_j} \leq 1 \qquad (5)$$

Se na equação (5), a variável de decisão Xj = Cj, significa que o orçamento determinado pelo ABB ou TDABB, será 100% atribuído ao requisito (j).
A fim de ilustrar este método é apresentado um pequeno exemplo, adaptado de Xie et al. [2003].

5.3.2. Exemplo Ilustrativo

Supondo que o orçamento disponível para melhorar a performance de um lápis é de €3000. Existe um total de cinco requisitos técnicos. As variáveis de decisão e os custos assumidos para a realização completa dos requisitos são apresentadas

no Quadro 3. Partindo do princípio que as correlações entre os requisitos e as funções foram normalizados (Quadro 4), então:

Requisitos Técnicos (j)	**Variáveis de Decisão (Xj)**	**Custos Estimados (Cj)**
Comprimento	X1	€500
Desgaste	X2	€200
Resíduo Produzido	X3	€700
Hexagonalidade	X4	€1000
Resíduo da Rasura	X5	€1000

Quadro 3: Variáveis de Decisão e Custos Estimados para os Requisitos

Funções (i)	**Requisitos Técnicos (j)**					**Importância das Funções (Ii)**
	Comprimento	Desgaste	Resíduo Produzido	Hexagonalidade	Resíduo da Rasura	
Ergonómico	0.250			0.750		15
Sem Resíduos		0.190	0.405		0.405	25
Durável	0.023	0.185	0.396		0.396	45
Não Rolar	0.100			0.900		15

Quadro 4: Correlações Normalizadas entre Funções e Requisitos

A maximização função de satisfação do cliente (Z) é dada por:

$$\text{Max } Z = \left[\sum_i I_i \cdot \sum_j R_{ij}^{*} \cdot X_j / C_j \right]$$

$$= \frac{15 \cdot 0.25 \cdot X1}{500} + \frac{15 \cdot 0.75 \cdot X4}{1000} + \frac{25 \cdot 0.19 \cdot X2}{200} + \frac{25 \cdot 0.405 \cdot X3}{700}$$

$$+ \frac{25 \cdot 0.405 \cdot X5}{1000} + \frac{45 \cdot 0.023 \cdot X1}{500} + \frac{45 \cdot 0.185 \cdot X2}{200} + \frac{45 \cdot 0.396 \cdot X3}{700}$$

$$+ \frac{45 \cdot 0.396 \cdot X5}{1000} + \frac{15 \cdot 0.1 \cdot X1}{500} + \frac{15 \cdot 0.9 \cdot X4}{1000}$$

As várias restrições são:

$$X1 + X2 + X3 + X4 + X5 \leq 3000$$

Supondo que as percentagens de melhoria dos requisitos técnicos, face à concorrência, são estabelecidas de acordo com o Quadro 5, o segundo conjunto de restrições ficará:

$$\frac{15 \cdot 0.25 \cdot X1}{500} + \frac{15 \cdot 0.75 \cdot X4}{1000} \geq 150\%$$

$$\frac{25 \cdot 0.19 \cdot X2}{200} + \frac{25 \cdot 0.405 \cdot X3}{700} + \frac{25 \cdot 0.405 \cdot X5}{1000} \geq 200\%$$

$$+ \frac{45 \cdot 0.023 \cdot X1}{500} + \frac{45 \cdot 0.185 \cdot X2}{200} + \frac{45 \cdot 0.396 \cdot X3}{700} + \frac{45 \cdot 0.396 \cdot X5}{1000} \geq 150\%$$

$$\frac{15 \cdot 0.1 \cdot X1}{500} + \frac{15 \cdot 0.9 \cdot X4}{1000} \geq 150\%$$

	Incremento na Satisfação do Cliente
Ergonómico	150%
Sem Resíduos	200%
Durável	150%
Não Rola	150%

Quadro 5: Satisfação do Cliente com as Funções do Produto

Para cada requisito técnico, é ainda possível definir um intervalo com o grau de execução admissível para os requisitos:

$$0 \leq \frac{X1}{500} \leq 1 \quad 0 \leq \frac{X2}{200} \leq 1 \quad 0 \leq \frac{X3}{700} \leq 1 \quad 0 \leq \frac{X4}{1000} \leq 1 \quad 0 \leq \frac{X5}{1000} \leq 1$$

Com recurso a um pacote de software, apropriado para a resolução das equações lineares, o problema de optimização pode ser solucionado, obtendo-se para as diversas variáveis de decisão os seguintes resultados:

$$X1 = 300, \quad X2 = 200, \quad X3 = 500, \quad X4 = 1000, \quad \text{e} \quad X5 = 1000$$

Como se pode verificar, os requisitos técnicos respeitantes ao comprimento e ao resíduo produzido pelo lápis, apenas são financiados parcialmente, de modo a que sejam cumpridos os limites orçamentais impostos (X1 = €300 e X3 = €500, contrariamente aos custos estimados, em €500 e €700).

5.4. Engenharia do Valor *versus* Programação Linear

O modelo de Xie et al. [2003] aplica a Programação Linear para determinar os recursos a atribuir às actividades ou aos requisitos; a satisfação do cliente é maximizada, atribuindo recursos aos requisitos que, para uma mesma unidade monetária, temporal ou outra, conseguem uma maior satisfação do cliente [Bode e Fung, 1998]. Este modelo considera, não somente a satisfação do cliente, mas também a satisfação da organização com o orçamento do projecto. Em função do orçamento disponível, as metas estabelecidas para os requisitos do software ou actividades, poderão ser totalmente financiadas, apenas parcialmente, ou poderão nem sequer ser financiadas, conforme se pode verificar no exemplo apresentado.

A Engenharia do Valor é um modelo mais simples, pois utiliza somente o índice (Ij/Cj), que reflecte a importância dos itens em análise (funções, requisitos, actividades, ou outros) e os respectivos custos, para estabelecer prioridades na distribuição dos recursos. O orçamento disponível é então distribuído pelos itens numa sequência descendente do rácio (Ij/Cj).

A importância dos itens (Ij) pode, por si só, ser considerada como uma forma de estabelecer prioridades. No entanto, apesar de parecer mais fácil, esta opção poderá conduzir a prioridades que resultem na atribuição de orçamentos sub optimizados. Por exemplo, poderá ser economicamente mais sensato dar uma prioridade aos requisitos com menor importância técnica, se as respectivas metas poderem ser alcançadas com muito menos esforço e recursos [Bode e Fung, 1998].

5.5. Earned Value Management (EVM)

Mesmo depois de estarem claramente definidos os orçamentos atribuídos a cada função, requisito ou actividade do projecto, é inevitável que ocorram alguns problemas durante a sua execução, que causam derrapagem dos custos, atrasos

na execução do calendário, ou mesmo a impossibilidade de alcançar os resultados desejados [Brownsword e Smith, 2005]. É assim indispensável, implementar mecanismos de monitorização e controlo, que mantenham o projecto no percurso planeado [Kendrick, 2003; Verzuh, 2005; Hallows, 2005].
O Departamento de Defesa Norte-Americano (DoD), na década de 1960, criou o C/SCSC – Cost/Schedule Control Systems Criteria, constituído por um conjunto de 35 critérios, para controlo dos seus projectos e, bem assim, a forma como deveriam ser apresentados os relatórios e os resultados [Christensen e Ferens, 1995]. Em 1997 o DoD, reviu alguns dos critérios, e o C/SCSC passou a denominar-se EVMS – Earned Value Management System, reduzindo o número de critérios de 35 a 32. Com esta mudança, o EVMS começou a ganhar a aceitação na indústria privada [Staley et al., 2002].
O EVM, como é conhecido o EVMS no sector privado, pode ser usado pelos gestores de projecto para planear, monitorizar e controlar o desenvolvimento de software, pois tem subjacente um conceito muito simples: comparar o que foi conseguido (valor ganho) com o que foi gasto e com o que se planeava gastar [Bechtold, 1999].

5.5.1. Parâmetros do EVM

A EVM assenta sobre três parâmetros, que constituem os pilares deste conceito, a partir dos quais são deduzidas informações sobre o projecto, designadamente, a situação actual e a história do projecto e, bem assim, projecções sobre a evolução dos respectivos custos e prazos [Brownsword e Smith, 2005].
Uma das melhores formas de interpretar estas informações é através de gráficos. A Figura 33 apresenta três curvas: (1) a linha base do custo estimado (em unidades de tempo) para os requisitos ou as actividades (BCWS ou Budget Cost of Work Scheduled), percorrendo todo o período de planeamento do projecto; (2) os custos efectivos (ACWP ou Actual Cost of Work Planned); (3) o valor ganho, considerando as actividades planeadas (BCWP ou Budget Cost of Work Planned).
Numa situação ideal, a curva dos custos reais (ACWP) e a curva do valor ganho (BCWP) deveriam sobrepor-se à curva do orçamento, o que representaria um desempenho de 100% para os custos e prazos do projecto.

Sabendo, no entanto, que na prática este objectivo é muito difícil de concretizar, a ideia será considerar a curva do orçamento (BCWS) como uma referência.

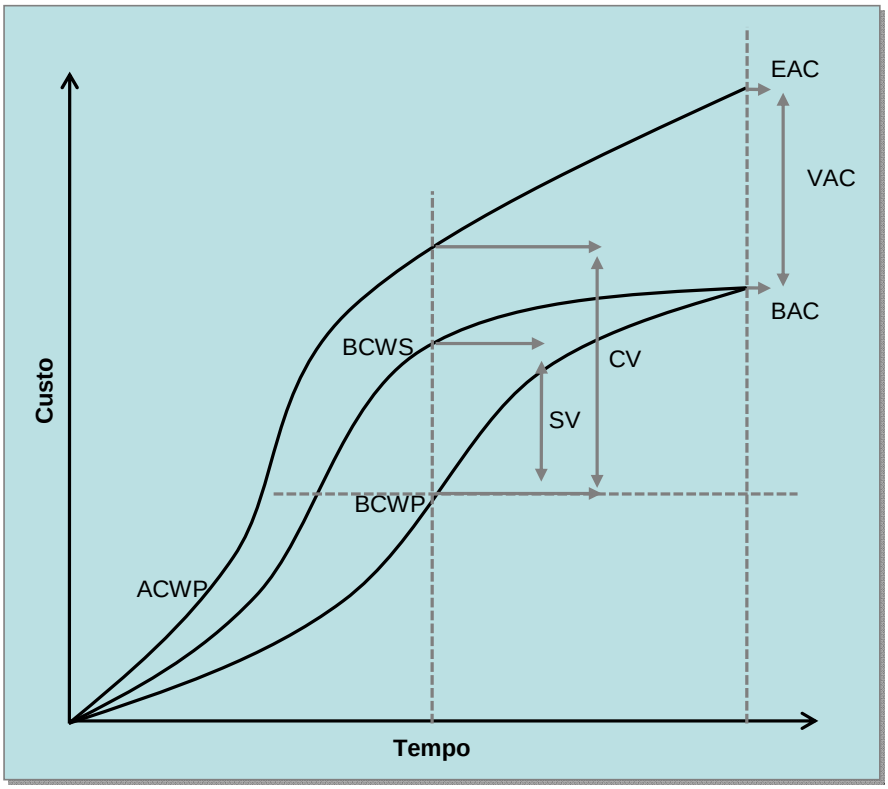

Figura 33: Gráfico para Análise do Valor Ganho
(Adaptado de [Forsberg et al., 2005])

A análise crítica sobre o progresso das actividades e dos requisitos é possível através da correlação entre os parâmetros citados, ao longo do ciclo de vida do projecto. As principais correlações, definidas pelo EVM, são apresentadas em seguida [PMI, 2004; Taylor, 2004; Forsberg et al., 2005]:

- CV (Cost Variance ou Variância do Custo): É a diferença entre o valor real e o custo efectivo, até a uma determinada data (CV = BCWP – ACWP); se CV for positivo, a actividade ou requisito em análise tem um custo abaixo do valor previsto; se for negativo, significa que foi ultrapassada a estimativa;

- SV (Schedule Variance ou Variância do Prazo): É a diferença, em termos de custo, entre o valor da actividade ou requisito, isto é, o valor ganho, e o valor planeado (SV = BCWP – BCWS); se SV for positivo, o projecto está adiantado; se for negativo, o projecto está atrasado;

- SPI (Schedule Performance Index ou Índice de Performance do Prazo): Representa a relação entre o valor ganho e o valor planeado numa determinada data (SPI = BCWP/ BCWS). O SPI mostra a taxa de conversão do valor planeado em valor ganho até à data[2];

- CPI (Cost Performance Index ou Índice de Performance do Custo): Exprime a relação entre o valor ganho e o custo real do projecto (CPI = BCWP/ACWP). O CPI mostra qual a taxa de conversão dos custos planeados em custos efectivos para o mesmo período[3].

Uma das principais vantagens da gestão por valor ganho é poder calcular o desempenho do projecto e traçar projecções e estimativas quanto aos prazos e custos das actividades e dos requisitos. Quanto mais fiéis e realistas forem as medições, melhores serão as projecções, e melhores serão as hipóteses de sucesso das acções intentadas pelo gestor de projecto para recuperar os atrasos ou custos imprevistos [Budd e Budd, 2005].
Através dos parâmetros do EVM podem ainda ser usadas várias fórmulas para estimar os custos futuros do projecto. Estas fórmulas recorrem-se basicamente

[2] Um SPI igual a 1 indica que o valor planeado foi integralmente adicionado ao projecto; se o SPI for menor que 1, o projecto está atrasado; se o SPI é superior a 1, o projecto está adiantado. Por exemplo, um SPI = 0,80 indica que somente 80% do tempo previsto no orçamento foi convertido em trabalho planeado, resultando numa perda de 20% do tempo útil disponível.

[3] Analogamente ao SPI, um CPI igual a 1 indica que o valor gasto pelo projecto foi integralmente adicionado ao projecto (o projecto está dentro do orçamento previsto); se o CPI for menor que 1, o projecto está a gastar mais do que o previsto (provavelmente haverá mais custos no final do projecto); se o CPI é superior a 1, o custo do projecto está abaixo do orçamento previsto. Por exemplo, um CPI = 0,80 indica que por cada €1 consumido, apenas €0,80 são convertidos em produto efectivo. A perda, portanto, é de €0,20 por cada €1 gasto.

do índice de performance dos custos (CPI) e do índice de performance do calendário (SPI), os quais podem ser utilizados independentemente ou em conjunto. A premissa utilizada nas fórmulas é a de que a performance do projecto no futuro segue a tendência até à data [Fleming e Koppelman, 1999-b]. Em relação à previsibilidade e às projecções dos custos e prazos dos projectos, aplicam-se as seguintes correlações [PMI, 2004; Taylor, 2004; Forsberg et al., 2005]:

- VAC (Variation at Completion ou Variação Final dos Custos): Exprime a diferença entre o custo final orçado para o projecto (BAC - Budget at Completion) e o custo final estimado (EAC - Estimated at Completion).

- ETC (Estimated to Complete ou Custo para Completar): É o valor financeiro necessário para completar o projecto (ETC = BAC-BCWP);

- EAC (Estimated at Completion ou Custo Final Estimado): É o valor financeiro do custo final previsto do projecto, considerando a performance actual (EAC = ACWP + ETC).

O ETC pode, também, considerar a opinião do gestor sobre a performance do projecto, mediante a aplicação de um novo índice de desempenho, que resulta da combinação do índice de performance do custo e do índice de performance do prazo [Christensen, 1998; Christensen, 1999]:

$$EAC = ACWP + ETC / Índice$$

Quando o gestor assume que o trabalho remanescente será executado em conformidade com o plano estabelecido e que o desvio registado não representa uma perda de controlo do orçamento, o índice de performance será igual a 1 – é uma perspectiva optimista; o gestor de projecto poderá também entender que o trabalho a executar seguirá o desempenho financeiro indicado pelo CPI – uma perspectiva mais provável; o gestor poderá ainda assumir que o restante trabalho seguirá simultaneamente a projecção financeira do CPI e do SPI ($CPI \cdot SPI$) – uma perspectiva pessimista [Christensen, 1999].

Uma vez determinadas as três perspectivas dos custos finais (EAC), poderá aplicar-se uma função probabilística triangular[4] aos dados, de modo a permitir inferir, com o grau de confiança desejado, qual o custo final estimado para o projecto. É possível criar uma função triangular de distribuição de probabilidade com três pontos, correspondentes às três perspectivas do EAC: optimista, provável e pessimista (ver Figura 34).

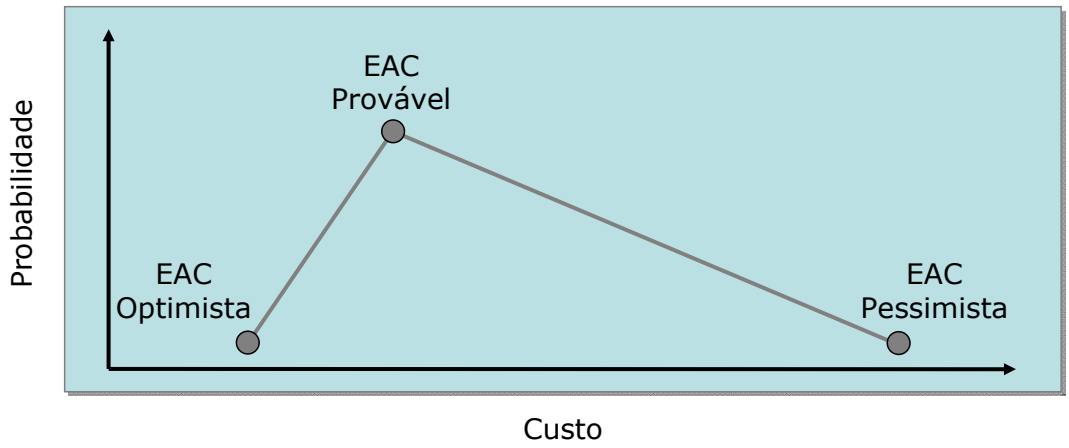

Figura 34: Função de Probabilidade Triangular
(Adaptado de [Browning et al., 2002])

Através da utilização de software de simulação, por exemplo, o pacote @Risk, é possível determinar o EAC final a partir da função triangular com os parâmetros ($EAC_{optimista}$, $EAC_{provável}$ e $EAC_{pessimista}$) [Browning et al., 2002].
Uma vez construída a função de densidade de probabilidade, são especificados os parâmetros da simulação, bem como o número de iterações e repetições da simulação. O número de iterações tem grande importância na determinação da qualidade dos resultados, uma vez que, quanto mais iterações são realizadas, mais a função de densidade final se aproxima das funções originais, porém, requer mais tempo de processamento computacional [Emblemsvåg, 2003].

[4] Esta função necessita, somente, de 3 pontos para a sua construção [Moran, 1996; Browning e Honour, 2005; Panthi, 2007].

5.5.2. Baseline das Actividades e dos Requisitos do Software

A elaboração da curva BCWS para o projecto (ou baseline, como também é frequentemente designada) é umas funções mais importantes, e das mais difíceis a executar pelo gestor de projecto [Christensen e Ferens, 1995].

Consiste, basicamente, na disposição em sequência de todas as actividades e objectos de custo (funções ou requisitos do software), ao longo de um determinado período e da estimativa dos respectivos custos [Bainey, 2004; Verzuh, 2005]. Esta curva servirá de referência para a monitorização e controlo do progresso e para as estimativas dos custos do projecto, pelo que deve existir algum cuidado na sua elaboração, de modo a representar com precisão o planeamento de todas as actividades do projecto [Bainey, 2004].

Sem esta preocupação o EVM é prejudicado, já que depende da comparação entre, o progresso do projecto (BCWP) e os custos reais incorridos (ACWP), com a baseline, ou seja, com o orçamento planeado (BCWS) [Taylor, 2004]. Obviamente, uma má correlação entre as informações do planeamento e do EVM causará a perda do controlo.

A partir daqui, o gestor de projecto poderá determinar quais os recursos necessários para cada actividade e requisito do software. É na orçamentação por actividades que ocorre a transformação dos resultados esperados em unidades de tempo comparáveis, que permitirá efectuar análises sobre custos actuais e futuros do projecto.

5.6. Framework de Integração

O desdobramento da função qualidade é eficiente na identificação das necessidades dos clientes e na sua tradução em características técnicas. A Engenharia de Valor ou os modelos de Programação Linear, são eficientes na procura de alternativas de menor custo para a concretização dos requisitos do software ou das actividades.

O QFD e o ABC/ABB, ou a sua versão mais recente, o TDABC/TDABB, podem convergir, proporcionando às organizações a possibilidade de definir a performance de soluções técnicas, que assegurem a realização de um produto de software com o máximo valor possível, com custos perfeitamente controlados e identificados. A definição de um produto de software que considere os desejos e

as expectativas dos clientes e, bem assim, as restrições orçamentais, isto é, que dê uma resposta aos problemas dos custos, da produtividade, da comunicação, da qualidade e do risco, poderá ser conseguida mediante a integração do QFD do TDABC/TDABB, da Engenharia do Valor ou a Programação Linear; o controlo dos custos e dos prazos dos requisitos ou das actividades é assegurado pelo EVM. A Figura 32 apresenta um framework, que poderá ser seguido para orientar o processo de integração do QFD, ABB/ABC ou TDABC/TDABB, EVM e Engenharia do Valor ou a Programação Linear.

É usado o QFD para identificar as necessidades dos clientes e traduzi-las em funções do software, que, por sua vez, são transformadas em requisitos do software. Em seguida, usando as técnicas de Programação Linear, Engenharia do Valor ou simplesmente a Importância, é desenvolvida uma solução inicial (solução proposta com base nos resultados do processo QFD), com a qual se faz uma estimativa dos custos para as actividades e para os requisitos do software, com base no ABB/TDABB;

A estimativa resultante permitirá verificar onde deverão efectuar-se modificações, com vista a optimizar a concepção do software. Ao fim de várias iterações, pode ser conseguido um equilíbrio entre a satisfação da organização e a satisfação do cliente, podendo, deste modo, ser conseguidas soluções mais adequadas, sob diferentes critérios [Tang et al., 2002].

As diferentes vias para a implementação das funções pretendidas pelo cliente, também, deverão ser analisadas, tendo em consideração os riscos identificados na matriz QFD do risco. A matriz inicial apenas fazia uma avaliação qualitativa dos riscos, porém, a estimativa dos custos, associada às projecções proporcionadas pelo EVM, permitem agora refazer a matriz de riscos, de forma a efectuar uma análise de risco quantitativa, logo mais precisa.

Nos modelos de análise de valor apresentados (Engenharia do Valor e modelos de Programação Linear), a variável de custo de um requisito de software é determinada pelo processo TDABB, descrito no terceiro capítulo. Esta variável é, no léxico do ABC, um objecto de custo. Como se viu no terceiro capítulo o custo de um objecto de custo (k) pode ser dado por:

$$CO_k = \sum_{l=1}^{p} \sum_{i=1}^{n} T_{k,i,l} . CA_i = \sum_{l=1}^{p} \sum_{j=1}^{m} \sum_{i=1}^{n} T_{k,i,l} \cdot \left(X_{i,j} \cdot CR_j \right),$$

Esta equação traduz o custo do objecto de custo (k), que consome actividades (i), com eventos (l), que por sua vez, consomem recursos (j). Note-se que, se uma actividade possuir apenas um evento, então, $T_{k,i,l} = Y_{k,i}$, logo, (COk) será igual ao custo estimado pelo processo ABB tradicional:

$$CO_k = \sum_{j=1}^{m} \sum_{i=1}^{n} Y_{k,i} \cdot X_{i,j} \cdot CR_j$$

A estimativa de custos das actividades, em qualquer dos processos, ABB ou TDABB, é dada por:

$$CA_i = \sum_{j=1}^{m} X_{i,j} \cdot CR_j$$

Com os dados (∑Tk,i,l), no caso de ser usado o modelo de custos TDABB ou (Yk,i), quando é usado o ABB, e bem assim (Xi,j), retirados das matrizes de processos/actividades e recursos, preenchidas durante a execução do processo QFD (ver Figuras 21 e 22), podemos verificar, então, que a variável de custo usada nos modelos de análise de valor, é dada por (CAi), quando se trata de avaliar o valor das actividades, e que é dado por (COk), quando se pretende avaliar os requisitos do software.

Esta premissa é válida para os modelos de Programação Linear e Engenharia do Valor, já que, a análise da Importância, não tem em consideração os custos das actividades ou dos requisitos (objectos de custo).

O orçamento atribuído às actividades ou aos requisitos não é, necessariamente, igual ao custo estimado, pois o projecto está sujeito a restrições orçamentais. Com base na informação da estimativa dos custos das actividades e dos requisitos do software, obtida pelo processo ABB/TDABB, há um regresso à definição dos requisitos, para reconsiderar os custos face à sua importância e decidir que alterações poderão ser efectuadas. Assim, é possível convergir numa solução consensual após algumas iterações.

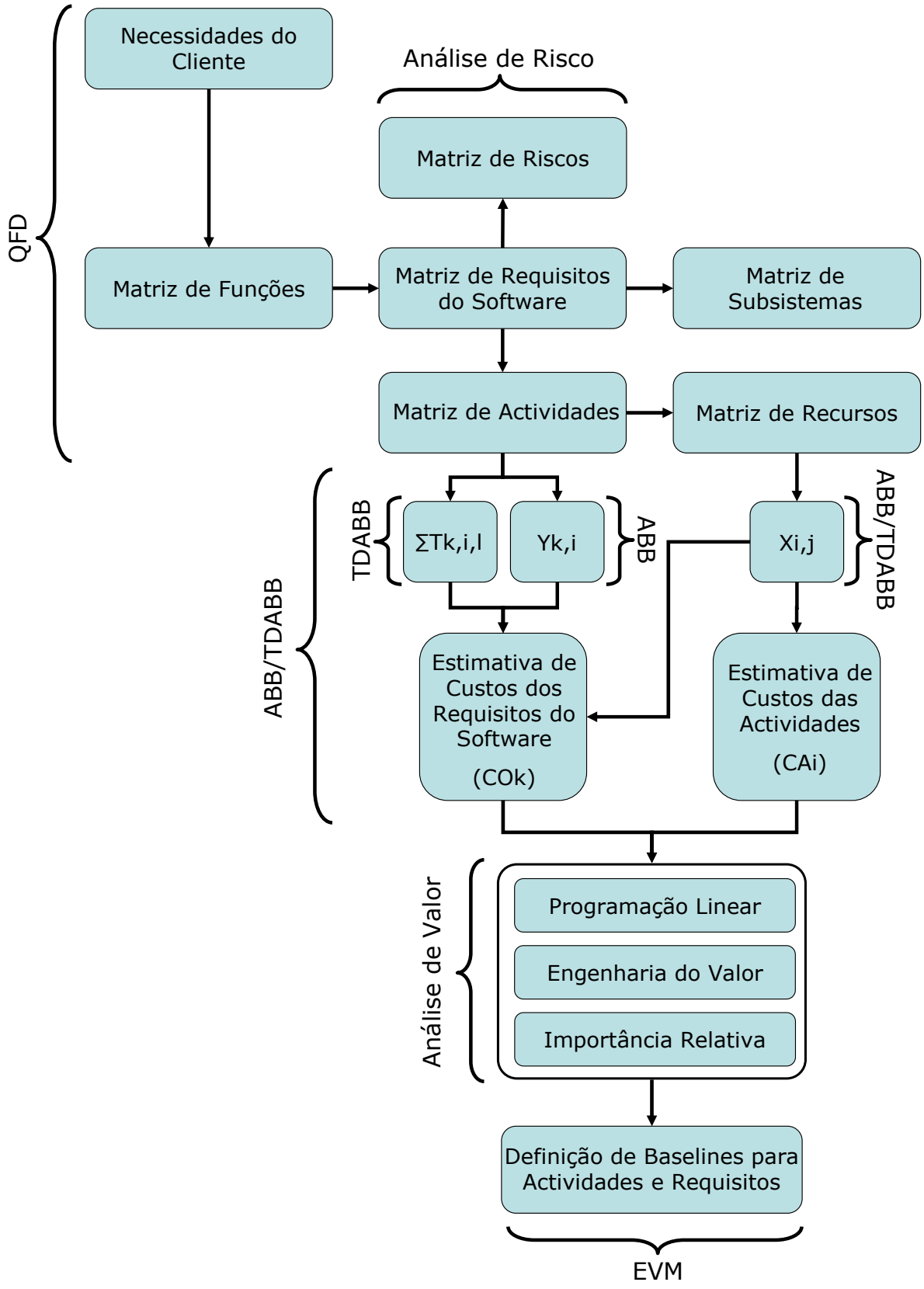

Figura 32: Framework de Integração

5.6.1. Lidar com a Complexidade

Nos sistemas mais complexos, é fundamental dividir o projecto em subsistemas. Estes poderão ser entregues a várias equipas que funcionarão em paralelo, coordenadas por uma equipa principal, uma vez que um sistema complexo exige um grande esforço da gestão.

A solução deste problema poderá passar pela criação de células que agreguem os requisitos do software correlacionados, para que possam ser controlados como um todo, com a consequente economia de esforços e pragmatismo.

Cada célula poderá ser responsável por um subsistema, com recursos e um orçamento próprios [Budd e Budd, 2005].

A quantidade ideal de células de controlo e a quantidade de actividades por célula depende do compromisso entre a viabilidade/capacidade de controlo e a precisão dos resultados. Quanto mais células e actividades forem definidas, maiores os esforços necessários para controlo e administração destes, por outro lado, mais precisos serão os resultados, e vice-versa [Budd e Budd, 2005].

A equipa principal poderá iniciar o seu trabalho antes de se constituírem as equipas celulares. Esta equipa terá como responsabilidade identificar as necessidades de cliente, definir os requisitos do software a um nível macro, agrupados pelo grau de correlação, e distribui-los pelas diferentes equipas. As equipas celulares analisam, então, em maior detalhe os requisitos do software, desenvolvem as actividades necessárias, procedendo, também, a uma análise de valor e à definição das baselines próprias.

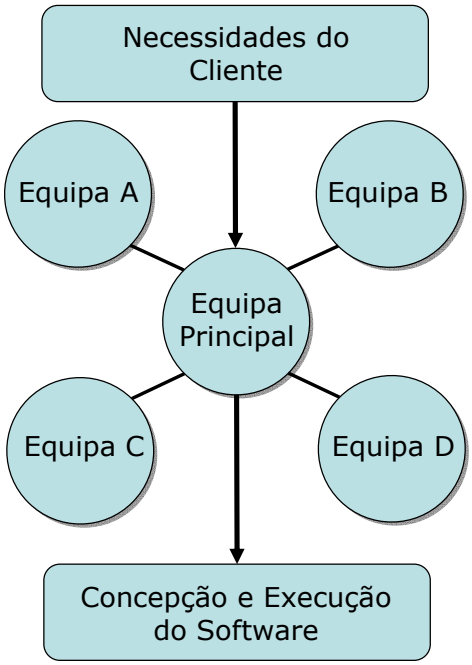

Figura 35: Distribuição Celular dos Projectos de Software
(Adaptado de [Sinbar e Hari, 1992])

A informação produzida ao nível das células é remetida, em tempo real, à equipa principal, que examina os custos dos requisitos do software propostos, com vista a efectuar eventuais alterações. Este processo é totalmente interactivo, com a informação a circular entre as equipas celulares e a equipa principal [Sinbar e Hari, 1992].

5.7. Conclusões do Capítulo

O custo é um factor que não é possível escamotear na gestão de projectos, pois é a prioridade número um de entre as preocupações dos clientes, e por consequência, a principal preocupação dos gestores de projecto. O QFD revela-se muito útil em colocar no software, aquilo que os clientes querem, de forma directa ou indirecta, isto é, quer pela comunicação directa do cliente, ou mediante a interpretação dos engenheiros do software das necessidades do cliente. É impossível, no entanto, dissociar as vontades dos clientes dos custos que acarreta a sua satisfação. O QFD convencional, mais preocupado em identificar as melhores soluções do ponto de vista do cliente, não considera os custos, mas como se viu, o seu controlo é, cada vez mais, um factor primordial

para o sucesso do qualquer projecto, e das próprias organizações produtoras de software.

Neste contexto, a Engenharia do Valor ou os modelos de Programação Linear (PL), revelam-se extraordinariamente úteis, na distribuição do orçamento do projecto, pelos requisitos ou actividades com maior importância na concretização dos desejos dos clientes. Os modelos PL levam, no entanto, a distribuição de recursos, um passo mais à frente, pois permitem estabelecer condições que deverão ser satisfeitas, para que um determinado item (requisito ou actividade) seja candidato a receber parte do orçamento, ou confirmada essa possibilidade, em que medida o item em causa pode ser apenas parcialmente realizado, sendo-lhe atribuído um orçamento menor do que o inicialmente estimado.

Concluído o processo de distribuição pelos vários itens em análise nas matrizes QFD, fundamentalmente requisitos e actividades, verificou-se que seria preciso um sistema para controlar a performance dos custos.

A gestão de valor ganho – EVM, é uma perspectiva de gestão orientada para o controlo da produtividade da execução do projecto, isto é, procura que no decorrer do projecto se verifique, não só o que foi gasto, mas como foi gasto. Esta abordagem é ideal para o controlo de uma gestão por actividades, pois permite identificar perturbações nas actividades conduzidas e, bem assim, desvios na implementação dos requisitos do software. Porém, na prática o EVM requer uma aplicação parcimoniosa, já que é mais um esforço de gestão a suportar, com custos associados. Aqui, o QFD poderá revelar, mais uma vez a sua utilidade, uma vez que é um mecanismo que estabelece prioridades, com base nas quais os gestores podem decidir quais os requisitos ou actividades que deverão ser monitorizados via EVM.

A integração do QFD, ABC/ABB, TDABC/TDABB, EVM, Engenharia do Valor e modelos de Programação Linear, possui, certamente alguma dificuldade. Com o objectivo de facilitar a implementação desta abordagem, propôs-se um framework de integração, que poderá servir de mapa para a sua aplicação prática. Em sistemas mais complexos o framework poderá ser repartido por várias equipas celulares interdependentes, responsáveis pela execução de um ou mais módulos que compõem o framework ou então, pela execução parcial dos vários módulos.

CAPÍTULO 6 – Conclusões

Na perspectiva da conclusão da pesquisa desenvolvida, são apresentadas neste capítulo a síntese do estudo efectuado, as suas contribuições, limitações e, bem assim, produzidas algumas indicações sobre possíveis estudos, que possam contribuir para a evolução da investigação conduzida.

Não há dúvidas de que a competição crescente está a forçar as organizações a lançarem no mercado novos produtos, com maior qualidade e num espaço de tempo cada vez menor. Como resultado, as organizações viram-se obrigadas a procurar novas formas de rentabilizar os seus esforços. A procura de vantagens competitivas, obrigou as organizações a reduzirem o ciclo de vida dos projectos. O ciclo de vida de um projecto envolve vários processos, conforme se pode verificar no segundo capítulo, comuns à maioria dos projectos; o ciclo de vida do software possui os seus próprios processos, que evidentemente, envolvem, numa ou noutra fase, os processos de gestão de projectos.

Mas os processos de gestão do projecto, e os processos do software, constituem apenas orientações para a prossecução do projecto, constatando-se que na prática, as organizações adoptam processos próprios, na maioria dos casos variantes dos mais conhecidos. Verifica-se, no entanto, que os modelos de ciclo de vida possuem pontos-chave comuns, isto é, independentemente do modelo adoptado, é necessário identificar os requisitos do cliente, os requisitos do software, conceber a arquitectura, proceder à codificação e testes do software, com vista a colocar em operação um produto, que satisfaça o mais possível, as expectativas do cliente e, bem assim, identificar as actividades (processos de gestão) que lhes dêem suporte. Os modelos em cascata, incremental, evolutivo, espiral, ou outros, podem variar na sua implementação, em função das características das organizações, porém, os pontos-chave mantêm-se sempre presentes, bem como as actividades de gestão.

Assim, entendeu-se que, para dar resposta aos problemas identificados, seria necessário começar por propor uma abordagem diferente ao ciclo de vida do software, e do projecto, aplicando aos pontos-chave, processos usados com sucesso noutras áreas, mas cujo emprego na indústria do software, ainda está a

dar os primeiros passos, designadamente o desdobramento da função qualidade (QFD) e o custeio por actividades (ABC), e as suas variantes, a orçamentação por actividades (ABB), Time-Driven ABC (TDABC) e Time-Driven ABB (TDABB); é claro que o software possui as suas especificidades próprias, mas afigura-se que existe um potencial enorme nestes processos.

O QFD permite desdobrar as necessidades dos clientes, em funções do software, requisitos do software, subsistemas e, no próprio código, quando existam ferramentas que o permitam, estabelecendo prioridades, em função da importância e relevância estratégica para os clientes, durante todo o ciclo de vida do software. São fixadas metas para a performance das funções do software, requisitos do software – gestão do projecto na perspectiva do cliente (desdobramento da qualidade – QD) e estabelecidas metas para a performance das actividades a desenvolver, bem como quais os recursos a adquirir – gestão do projecto na perspectiva da organização (desdobramento da função qualidade em sentido restrito – QFDr). É precisamente nesta segunda área de actuação, que o ABC se situa. A partir da estratégia da organização, é implementado um processo de gestão de actividades, que dá suporte aos processos do software, isto é, o ABC é usado na gestão das actividades de desenvolvimento do software.

O QFD, de modo geral, permite melhorar a qualidade do software, ao estruturar de forma eficaz a identificação das necessidades do cliente e o planeamento da resposta a dar, correlacionando as decisões sobre o concepção do produto com os processos de software, determinando quais os processos/actividades que mais contribuem para a satisfação dos clientes (materializada na implementação dos requisitos do software adequados); identifica facilmente quais os pontos-chave que precisam de ser melhorados, e em que medida, pois possui uma lógica de causa e efeito, permitindo que as decisões tomadas durante o desenvolvimento, sejam compatíveis com as metas estabelecidas; facilita, também, a comunicação entre as pessoas, um dos grandes problemas das organizações, e intra-organizações, como foi referido anteriormente, o que contribui para melhorar a performance dos processos e aumentar a satisfação dos clientes, quer externos, quer mesmo internos à organização.

O ABC é integrado com o QFD para planear, monitorizar e controlar os projectos de software, recorrendo a indicadores, sob a forma de indutores de custo de

recursos e actividades, em unidades de tempo ou monetárias – dependendo do uso do ABC/ABB ou do TDABC/TDABB – para estimar custos e/ou obter indicações sobre os custos actuais do projecto, contribuindo para que o gestor tome as decisões mais acertadas. O ABC proporciona uma medida dos custos dos processos do software, pois atribui com precisão, os custos das actividades, aos objectos de custo responsáveis, permitindo ao gestor efectuar uma análise crítica do valor acrescentado pelas actividades, ou pelos próprios objectos, que assumem, por regra, a forma de requisitos do software. A análise dos custos através do modelo ABC propicia, deste modo, condições para a identificação, análise e melhoria dos processos, e consequentemente, o aumento da qualidade e produtividade dos projectos de software. Mas as vantagens do ABC não terminam aqui, pois continuando mais para montante, é possível determinar o valor acrescentado por cada recurso atribuído ao projecto. Em conjunto os indicadores de actividades e recursos (indutores de custo) proporcionam informações precisas sobre a produtividade, uma das grandes dificuldades actuais da gestão de projectos de software.

Esta formatação do projecto por recursos, actividades e objectos de custo, permite uma grande flexibilidade, com a introdução de novos objectos considerados relevantes, e/ou a exclusão de objectos menos importantes com base nos dados recolhidos, acomodando as necessidades dos processos de software mais modernos, globalmente designados por ágeis (Scrum, XP, Crystal, RUP, DSDM, etc.), que preconizam, de uma forma geral, a implementação do software em iterações sucessivas, com a possibilidade de introduzir alterações aos requisitos durante a fase de execução.

A complexidade do software actual implica que sejam desenvolvidas muitas actividades durante o ciclo de vida de um projecto, que precisam de ser ajustadas quando é iniciado um novo projecto, mesmo que seja semelhante. Ora, o ABC parte do princípio de que qualquer actividade idêntica requer o mesmo tempo de execução, pelo que, se as actividades sofrerem alterações, é necessário criar novas actividades. O TDABC não exige tal simplificação, levando o ABC um passo mais à frente. Suporta a complexidade e transmutação das operações actuais, recorrendo a equações de tempo, que reflectem as características específicas das actividades que causam variações no tempo de

execução. As informações sobre os custos e a rentabilidade do software são portanto conseguidas de uma forma rápida e barata, com uma implementação simples e pouco onerosa. O TDABC pode, do mesmo modo que o ABC, ser usado para estimar os custos das actividades e dos recursos, passando nesse caso a designar-se por TDABB.

A aplicação do EVM às estimativas de custos, obtidas pelo ABB ou TDABB, permite consolidar as informações recolhidas, numa única unidade medida, preferencialmente em unidades de tempo (unidade usada pelo TDABB), o que facilita a visualização do progresso e a realização de projecções futuras, contribuindo deste modo para facilitar a tomada de decisões. As vantagens da integração do EVM com o TDABB são, contudo, mais abrangentes – a interacção entre os gestores de projecto e as pessoas envolvidas na construção da baseline de actividades ou requisitos do software, necessária à aplicação da EVM, implica a atribuição de responsabilidades, funcionando assim como um agente de comunicação.

Os indicadores de performance do EVM podem ser utilizados, inclusive, para verificar o planeamento, permitindo analisar a qualidade do plano estabelecido, ainda que, apenas apontem os pontos a serem analisados com maior detalhe, e que, provavelmente, precisarão de acções correctivas. Estas acções de observação e melhoria constante colocam a organização no caminho para conseguir a excelência do software.

A relação entre QFD e TDABC/TDABB é cimentada, pela aplicação da Engenharia do Valor ou dos modelos de Programação Linear, ao incorporar as considerações financeiras resultantes do TDABC/TDABB no QFD, estabelecendo assim, prioridades para os requisitos do software ou para as actividades, que contribuem para a satisfação dos clientes, mas que, simultaneamente satisfazem as restrições orçamentais.

Esta investigação conclui com a proposta de um framework de gestão de projectos de software, capaz de orientar as organizações na implementação do QFD, ABB ou TDABB, EVM, Engenharia do Valor e modelos de Programação Linear para os custos, com vista a alcançar uma melhoria da performance dos projectos de software. O framework proposto, procura facilitar a cooperação das pessoas envolvidas, estando particularmente adaptado, à gestão de múltiplos

projectos, com evidentes sinergias; a sua aplicação inicial em condições controladas, por exemplo, num só projecto, contribuirá para motivar a organização para a sua adopção. Através do framework proposto, a transição do processo de custos de uma organização, para o ABC/TDABC e ABB/TDABB, pode ocorrer de forma suave, sem grandes investimentos em recursos computacionais ou recolha de dados, nem grandes reestruturações organizacionais.

Em suma, espera-se que o framework funcione como guia para a implementação do custeio e orçamentação por actividades, visando uma gestão de actividades e recursos mais eficiente e eficaz, respondendo desta forma aos problemas da gestão dos custos dos projectos de software, mas não só, pois são estabelecidas orientações para a integração do ABC/ABB com o QFD, para se conseguir uma correlação profunda entre as necessidades dos clientes actuais e o software resultante, reduzindo as falhas na comunicação entre clientes e engenheiros do software, para se obter um software com maior qualidade. O framework aplica ainda o método de decisão AHP, usado nas mais variadas áreas industriais, criando uma abordagem à gestão do risco, muito semelhante ao processo QFD, correlacionando os riscos com os requisitos do software, ou caso se considere necessário com os processos/actividades. O módulo EVM do framework estreita ainda mais, a correlação do ABC (ABB, TDABC/TDABB) com o QFD, pois é usado para monitorizar as actividades usadas na implementação dos requisitos do software (que irão satisfazer as necessidades dos clientes), funcionando simultaneamente, como indicador de risco, que influencia o preenchimento da matriz de riscos. Finalmente, o framework apresenta um módulo de análise de valor, dos requisitos do software e das actividades, apesar de poder ser usado para avaliar o software em qualquer nível: funções, requisitos do software, subsistemas, blocos de código, etc. Este módulo é muito interessante, pois permite instituir as mais variadas restrições (ou condições) na distribuição dos recursos pelas actividades, e destas pelos designados objectos de custo, que para manter a simplicidade do framework, se limitam aos requisitos do software, podendo, no entanto, alargar-se ao código do software.

6.1. Trabalhos Futuros

A validação através do estudo de um caso prático poderá demonstrar que as propostas apresentadas podem ser uma solução atractiva para a indústria de software actual, que precisa de métodos simples e objectivos, que garantam a fiabilidade da informação usada na tomada de decisões, sem elevados custos de concepção e implementação.
Entre as principais características, ou necessidades de trabalhos futuros, em conformidade com o discorrido anteriormente, destacam-se:

- A implementação do framework numa organização típica, que desenvolva projectos de software;

- O refinamento do framework, com base nos resultados alcançados com a aplicação prática da metodologia;

- O desenvolvimento de um sistema de informação que automatize o framework proposto, com vista a facilitar e agilizar o registo de dados e os cálculos matriciais, bem como o rastreamento dos objectos ao longo das diversas fases do ciclo de vida do software, permitindo que se efectue uma gestão de configurações.

- Finalmente, é possível desenvolver modelos matemáticos da metodologia, que simulem o comportamento da organização face à dinâmica da concorrência, obtendo-se assim diferentes perspectivas que darão suporte à tomada de decisões estratégicas.

Referências

[Akao e Mazur, 1998], Yoji Akao e Glenn H. Mazur, Using QFD To Assure QS-9000 Compliance, ISQFD 98, 1998

[Akao, 1990], Yoji Akao, Quality Function Deployment – Integrating Customer Requirements into Product Design, Productivity Press, 1990

[Akao, 1997], Yoji Akao, QFD: Past, Present, and Future, International Symposium on QFD 97, 1997

[Albuquerque et al., 2003], Alfram Albuquerque, Edilson Ferneda, Marcelo Barros e Agenor Martins, The Innovation Planning Task for Products and Services an Architecture Using QFD and CBR, ICEIS 2003 - Artificial Intelligence and Decision Support Systems, 2003

[Anderson et al., 2002], S. Anderson, J. Hesford e M.Young, Factors influencing the Performance of Activity-Based Costing Teams: A field study of ABC Model Development Time in the Automobile Industry, Accounting, Organizations and Society, 2002

[Armour, 2004], Philip G. Armour, The Laws of Software Process: A New Model for the Production and Management of Software, Auerbach Publications, 2004

[Bainey, 2004], Kenneth R. Bainey, Integrated IT Project Management - A Model-Centric Approach, Artech House, 2004

[Bechtold, 1999], Richard Bechtold, Essentials of Software Project Management, Management Concepts, 1999

[Beiter e Ishii, 1999], Kurt A. Beiter e Kosuke Ishii, Incorporating the Voice of the Customer in Preliminary Component Design, Proceedings of DETC ’99, ASME Design for Manufacturing Symposium, 1999

[Bode e Fung, 1998], Bode Jürgen e Richard Y.K. Fung, Cost Engineering with Quality Function Deployment, Elsevier, 1998

[Bohem, 1991], Barry W. Bohem, Software Risk Management: Principles and Practices, IEEE Software, 1991

[Bossert, 1991], James L. Bossert, Quality Function Deployment – A Practitioner's Approach, ASQC Quality Press, 1991

[Bossidy et al., 2002], Larry Bossidy, Ram Charan e Charles Burck, Execution - The Discipline Of Getting Things Done, Crown Business, 2002

[Brimson e Antos, 1999], James A. Brimson e John Antos, Driving Value Using Activity-Based Budgeting, John Wiley & Sons, 1999

[Brooks, 1987], Fred P. Brooks, No Silver Bullet: Essence and Accidents of Software Engineering, IEEE Computer, 1987

[Browning e Honour, 2005], Tyson R. Browning e Eric C. Honour, Measuring the Lifecycle Value of a System, Browning & Honour, 2005

[Browning et al., 2002], Tyson R. Browning, John J. Deyst, Steven D. Eppinger, Daniel E. Whitney, Adding Value in Product Development by Creating Information and Reducing Risk, IEEE Transactions on Engineering Management, 2002

[Brownsword e Smith, 2005], Lisa Brownsword e Jim Smith, Using Earned Value Management (EVM) in Spiral, Development, Integration of Software-Intensive Systems Initiative, 2005

[Bruggeman et al., 2005], Werner Bruggeman, Patricia Everaert, Steven R. Anderson e Yves Levant, Modelling Logistics Costs using Time-Driven ABC: A Case in a Distribution Company, Ghent University, Faculty of Economics and Business Administration, 2005

[Budd e Budd, 2005], Charles I. Buddd e Charnele S. Budd, A Practical Guide to Earned Value Project Management, Management Concepts, 2005

[Buglione e Abran, 2001], Luigi Buglione e Alain Abran, QF2D: Quality Factor Through QFD Application, Proceedings of Qualita2001 (Fourth International Congress on Quality and Reliability), 2001

[Buglione et al., 2002], Luigi Buglione, Nihal Kececi e Alain Abran, An Integrated Graphical Assessment For Managing Software Product Quality, International Conference on Software Quality, 2002

[Carpinetti e Peixoto, 1998], Luiz C. R. Carpinetti e Manoel O. C. Peixoto, Merging Two QFD Models Into One: An Approach of Application, International Journal of Manufacturing Technology and Management, 1998

[Chao e Ishii, 2004], Lawrence P. Chao e Kosuke Ishii, Design Process Error-Proofing: Project Quality Function Deployment, Proceedings of DETC '04, 2004 ASME Design Engineering Technical Conferences, 2004

[Chen e Weng, 2003], Liang-Hsuan Chen e Ming-Chu Weng, A Fuzzy Model for Exploiting Quality Function Deployment, Elsevier, 2003

[Chen e Weng, 2003], Liang-Hsuan Chen e Ming-Chu Weng, An evaluation approach to engineering design in QFD processes using fuzzy goal programming models, Elsevier, 2006

[Christensen e Ferens, 1995], David S. Christensen e Daniel V. Ferens, Using Earned Value For Performance Measurement on Software Development Projects, Acquisition Review, 1995

[Christensen, 1998], David S. Christensen, The Costs And Benefits Of The Earned Value Management Process, Acquisition Review Quarterly, 1998

[Christensen, 1998], David S. Christensen, The Costs And Benefits Of The Earned Value Management Process, Acquisition Review Quarterly, 1998

[Christensen, 1999], David S. Christensen, Value Cost Management Report To Evaluate The Contractor's Estimate At Completion, Acquisition Review Quarterly, 1999

[Cohen, 1995], Lou Cohen, Quality Function Deployment – How to Make QFD Work For You, Addison Wesley Longman, 1995

[Cokins, 2001], Gary Cokins, Activity-Based Cost Management – An Executives Guide, John Wiley & Sons, 2001

[Cook, 2005], Curtis R. Cook, Just Enough Project Management: The Indispensable Four-Step Process for Managing Any Project Better, Faster, Cheaper, McGraw-Hill, 2005

[Denne e Cleland-Huang, 2003], Mark Denne e Jane Cleland-Huang, Software by Numbers: Low-Risk, High-Return Development, Prentice Hall, 2003

[Denney, 2005], Richard Denney, Succeeding with Use Cases Working Smart to Deliver Quality, Addison Wesley Professional, 2005

[Dey, 2002], Prasanta Kumar Dey, Project Risk Management: A Combined Analytic Hierarchy Process and Decision Tree Approach, Cost Engineering, 2002

[Dimsey e Mazur, 2002], Jim Dimsey e Glenn Mazur, QFD to Direct Value Engineering in the Design of a Brake System, QFD Institute, 2002

[Emblemsvåg, 2003], Jan Emblemsvåg, Life-Cycle Costing Using Activity-Based Costing And Monte Carlo Methods To Manage Future Costs And Risks, John Wiley & Sons, 2003

[Evans, 2004], Isabel Evans, Achieving Software Quality through Teamwork, Artech House, 2004

[Ewusi-Mensah, 2003], Kweku Ewusi-Mensah, Software Development Failures: Anatomy of Abandoned Projects, The MIT Press, 2003

[Fairley, 1994], Richard Fairley, Risk Management For Software Projects, IEEE Computer Society Press, 1994

[Fehlmann, 1998], Thomas M. Fehlmann, Quality Function Deployment for the full Business Life Cycle, Euro Project Office, 1998

[Fichman e Kemerer, 2001], Robert G. Fichman e Chris F. Kemerer, Activity Based Costing For Component-Based Software Development, Journal of Systems and Software, 2001

[Fleming e Koppelman, 1999-a], Quentin W. Fleming and Joel M. Koppelman, Earned Value Project Management - An Introduction, The Journal of Defence Software Engineering, 1999

[Fleming e Koppelman, 1999-b], Quentin W. Fleming and Joel M. Koppelman, Earned Value Project Management - A Powerful Tool for Software Projects, The Journal of Defence Software Engineering, 1999

[Forsberg et al., 2005], Kevin Forsberg, Hal Mooz e Howard Cotterman, Visualizing Project Management - Models and frameworks for mastering complex systems, Third Edition, John Wiley & Sons, 2005

[Fujita et al., 2003], K. Fujita, H. Takagi e T. Nakayama, Assessment Method of Value Distribution for Product Family Deployment, International Conference On Engineering Design, 2003

[Gerlach et al., 2002], James Gerlach, Bruce Neumann, Edwin Moldauer, Martha Argo e Daniel Frisby, Determining The Cost of IT Services, Communications of The ACM, 2002

[Gonzalez et al., 2005], Marvin E. Gonzalez, Gioconda Quesada, Rhonda Mack e Ignacio Urrutia, Building Na Activity-Based Costing Hospital Model Using Quality Function Deployment and Benchmarking, Emerald Group Publishing Limited, 2005

[Gunasekaran et al., 2000], A. Gunasekaran, R. McNeil e D. Singh, Activity-based management in a small company: a case study, Production Planning & Control, 2000

[Hallows, 2005], Jolyon Hallows, Information Systems Project Management: How to Deliver Function and Value in Information Technology Projects, Second Edition, AMACOM, 2005

[Hamilton, 1999], Marc Hamilton, Software Development Building Reliable Systems, Prentice Hall, 1999

[Heldman, 2005], Kim Heldman, Project Manager's Spotlight on Risk Management, Sybex, 2005

[Herzwurm et al., 2002], Georg Herzwurm, Sixten Schockert, Ulrike Dowie, Michael Breidung, Quality Planning for E-Commerce Applications, Proceedings of 1st Turkish National Symposium on Quality Function Deployment in Izmir, Turkey, 2002

[Hicks, 1999], Douglas T. Hicks, Activity-Based Costing – Making it Work For Small and Mid-Sized Companies, John Wiley & Sons, Inc, 1999

[Hierholzer et al., 1998], Andreas Hierholzer, Georg Herzwurm, Harald Schlang, Applying QFD For Software Process Improvement At Sap Ag, Walldorf, Germany, Proceedings of the World Innovation and Strategy Conference in Sydney, Australia, 1998

[Hoyle, 2005], David Hoyle, Automotive Quality Systems Handbook: ISO/TS 16949:2002 Edition, Butterworth-Heinemann, 2005

[Ioannou et al., 2003], George Ioannou, Katherine C. Pramataris e Gregory P. Prastacos, A quality Function Deployment Approach to Web Site Development: Applications For Electronic Retailing, Les Cahiers du Management Technologique, 2003

[Jagdev et al., 1997], H. Jagdev, P. Bradley e O. Molloy, A QFD Based Performance Measurement Tool, Elsevier, 1997

[Jiang et al., 2007], Jui-Chin Jiang, Ming-Li Shiu e Mao-Hsiung Tu, QFD's Evolution in Japan and the West, ASQ, 2007

[Kaldate et al., 2006], A. Kaldate, D. Thurston, H. Emamipour e M. Rood, Engineering parameter selection for design optimization during preliminary design, Journal of Engineering Design, 2006

[Kaplan e Anderson, 2004], Robert S. Kaplan e Steven R. Anderson, Time-Driven Activity-Based Costing, Harvard Business Review, 2004

[Kaplan e Cooper, 1998], Robert S. Kaplan e Robin Cooper, Cost & Effect – Using Integrated Cost Systems to Drive Profitability and Performance, Harvard Business Scholl Press, 1998

[Karlson e Ryan, 1997], Joachim Karlson e Kevin Ryan, A Cost-Value Approach for Prioritizing Requirements, IEEE Software, 1997

[Kendrick, 2003], Tom Kendrick, Identifying and Managing Project Risk: Essential Tools for Failure-Proofing Your Project, AMACOM, 2003

[Koch, 2005], Alan S. Koch, Agile Software Development - Evaluating the Methods for Your Organization, Artech House, 2005

[Korpela et al., 2002], Jukka Korpela, Antti Lehmusvaara, Kalevi Kyläheiko e Markku Tuominen, Adjusting Safety Stock Requirements with an AHP-based Risk Analysis, Proceedings of the 36th Hawaii International Conference on System Sciences, 2002

[Krogstie, 1999], John Krogstie, Using Quality Function Deployment in Software Requirements Specification, In proceedings of the Fifth International Workshop on Requirements Engineering: Foundations for Software Quality, 1999

[Larman, 2003], Craig Larman, Agile and Iterative Development: A Manager's Guide, Addison Wesley, 2003

[Lee, 2005], Insub Leo Lee, Quality Function Deployment - QFD, 2005

[Leffingwell e Widrig, 2003], Dean Leffingwell, Don Widrig, Managing Software Requirements: A Use Case Approach, Second Edition, Addison Wesley, 2003

[Liu et al., 2007], Xiaoqing (Frank) Liu, Yan Sun, Praveen Inuganti, Chandra Sekhar Veera e Yuji Kyoya, A Methodology for Tracing the Requirements in the Object-Oriented Software Design Process Using Quality Function Deployment, ASQ, 2007

[Lyman, 1990], D. Lyman, Deployment Normalization, Transactions from 2nd Symposium on Quality Function Deployment, 1990

[Marchewka, 2003], Jack T. Marchewka, Information Project Management – Providing Measurable Organizational Value, John Wiley & Sons, 2003

[Max, 2005], Mitchell Max, SOX + ABC = VALUE!, The Performax Group, 2005

[Mazur, 1996], Glenn Mazur, Voice Of Customer Analysis: A Modern System Of Front-End QFD Tools, With Case Studies, AQC, 1996

[McConnell, 1996], Steve McConnell, Rapid development: taming wild software schedules, Microsoft Press, 1996

[Mogollón, 2000], Ruth Maritza Avila Mogollón, El AHP (Proceso Analítico Jerárquico) y su Aplicación Para Determinar Los Usos de Las Tierras - El Caso de Brasil, FAO 2000 - Proyecto Regional "Información Sobre Tierras Y Aguas Para Un Desarrollo Agrícola Sostenible" (Proyecto Gcp/Rla/126/Jpn), 2000

[Moore, 2000], Kevin R. Moore, Using Activity-Based Costing To Improve Performance: A Case Study Report, Maxwell Air Force Base, Alabama, 2000

[Moran, 1996], S.R. Moran, Observations on AF/SMC Proposal Cost-Risk Evaluation Techniques, Business Development, Advanced Programs & Technology - Lockheed Martin Missiles & Space, 1996

[Newell e Grashina, 2004], Michael W. Newell e Marina N. Grashina, The Project Management Question and Answer Book, AMACOM, 2004

[Ooi e Soh, 2003], Ginny Oii e Christina Soh, Developing an Activity-Based Costing Approach For System Development and Implementation, ACM SIGMIS Database - ACM Press, 2003

[Ouyang et al., 1997], Sheng Ouyang, Jasper Fai, Qian Wang e Kim Jhonson, Quality Function Deployment, Department of Computer Science, University of Calgary, 1997

[Panthi, 2007], Kamalesh Panthi, Prioritizing and Estimating Hydropower Project Construction Risks: A Case Study of Nyadi Hydropower Project, Himalayan Research Papers Archive, 2007

[PMI, 2004], Project Management Institute, A Guide to the Project Management Body of Knowledge, Third Edition, Project Management Institute, 2004

[Porter, 1998], Michael E. Porter, Competitive Advantage – Creating and Sustaining a Superior Performance, The Free Press, 1998

[Porter, 2001-a], Michael E. Porter, From Competitive Advantage to Corporate Strategy, Harvard Business Review, 2001

[Porter, 2001-b], Michael E. Porter, How Information Gives You Competitive Advantage, Harvard Business Review, 2001

[Porter, 2007], Michael E. Porter, Estratégia e Vantagem Competitiva, Público, 2007

[Report #95-139, 1995], Sate of Texas Audit Office, Report #95-139, Guide to Cost-Based Decision-Making, 1995

[Roztocki et al., 1999], Narcyz Roztocki, Jorge F. Valenzuela, José D. Porter, Robin M. Monk e Kim LaScola Needy, A Procedure for Smooth Implementation of Activity-Based Costing in Small Companies, Proceedings of the 1999 American Society of Engineering Management (ASEM) National Conference, 1999

[Saaty, 1980], T. L. Saaty, The Analytic Hierarchy Process, McGraw-Hill, 1980

[Sinbar e Hari, 1992], Gabriel Sinbar e Amihud Hari, Quality Improvement With an Integrated, Method of QFD and Value Engineering, 9th International Conference of the Israel Society for Quality Assurance, 1992.

[Staley et al., 2002], Mary Jo Staley, Patricia Oberndorf e Carol A. Sledge, Using EVMS with COTS-Based Systems, Carnegie Mellon University - Software Engineering Institute, 2002

[Stenbeck e Svensson, 2004], Carl Stenbeck e Johan Svensson, Value Balancing Method for Product Development – A Case Study at Volvo Corporation, School of Economic and Commercial Law, Göteborg University, 2004

[Stepanek, 2005], George Stepanek, Software Project Secrets: Why Software Projects Fail, Apress, 2005

[Stylianou et al., 1997], Antonis C. Stylianou, Ram L. Kumar e Moutaz J. Khouja, Total Quality Management – Based Systems Development Process, The Data Base for Advances in Information Systems, 1997

[Succi et al. 2000], Giancarlo Succi, Luigi Benedicenti, Stefano De Panfilis e Tullio Vernazza, Activity-Based OO Business Modelling and Control, IEEE TI Pro, 2000

[SWEBOK, 2004], SWEBOK, Guide to the Software Engineering Body of Knowledge, IEEE Computer Society, 2004

[Tang et al., 2000], Jiafu Tang, Richard Y.K. Fung, Baodong Xu e Dingwei Wang, A New Approach To Quality Function Deployment Planning With Financial Consideration, Elsevier, 2000

[Taylor, 2004], James Taylor, Managing Information Technology Projects: Applying Project Management Strategies to Software, Hardware, and Integration Initiatives, AMACOM, 2004

[Tian, 2005], Jeff Tian, Software Quality Engineering - Testing, Quality Assurance, and Quantifiable Improvement, John Wiley & Sons, 2005

[Tran e Sherif, 1995], Tuyct-Lan Tran e Joseph S. Sherif, Quality Function Deployment (QFD): An Effective Technique For Requirements Acquisition, Proceedings of the 2nd IEEE Software Engineering Standards Symposium, 1995

[Tuan et al., 2006], Han-Wen Tuan, Chia-Yi Liuy e Chiou.Mei Chen, Using ABC Model For Software Process Improvement: A Balanced Perspective, Proceedings of the 39th Annual Hawaii International Conference on System Sciences, 2006

[Turney, 1991], P. B. B. Turney, Common Cents: The ABC Performance Breakthrough, Hillsboro - Cost Technology, 1991

[Vahadilla e Kumar, 2003], Omkarprasad S. Vahadilla e Sushil Kumar, Analytic hierarchy process: An overview of applications, Elsevier, 2003

[Verzuh, 2005], Eric Verzuh, The Fast Forward MBA in Project Management, Second Edition, John Wiley & Sons, 2005

[Wasserman, 1993], G.S. Wasserman, On how to prioritize design requirements during the QFD planning process, IEEE Transactions, 1993

[Wilson, 2000], Peter B. Wilson, Sizing Software With Testable Requirements, Auerbach Publications, 2000

[Wysocki e McGary, 2003], Robert K. Wysocki and Rudd McGary, Effective Project Management, Third Edition, John Wiley & Sons, 2003

[Xie et al., 2003], M. Xie, K. C. Tan e T. N. Goh, Advanced QFD Applications, American Society for Quality, 2003

[Xiong et al., 2005], Xiong Wei, Yoshimichi Watanabe e Hisakazu Shindo, A Description Approach to Software by HoQ Extension Concept and Its Quantitative Structuralization, Journal of Software, 2005

[Yacuzzi e Martín, 2003], Enrique Yacuzzi e Fernando Martín, QFD: Conceptos, Aplicaciones y Nuevos Desarrollos, Universidad del CEMA, 2003

[Yamamoto et al., 2005], Chigako Yamamoto, Kunihiro Kishi, Fumihiro Hara e Keiichi Satoh, Using Quality Function Deployment To Evaluate Government Services From The Customer's Perspective, Journal of the Eastern Asia Society for Transportation Studies, 2005

[Zrymiak, 2003], Dan Zrymiak, Software Quality function Deployment – Modifying The House of Quality for Software, iSixSigma.com, 2003