

**Universidade Federal de Pernambuco
Centro de Informática**

**Seleção de Produtos de Software
Utilizando uma Abordagem Baseada
em Engenharia de Requisitos**

**Por
Carina Frota Alves
Dissertação de Mestrado**

**Recife
Março, 2001**

CARINA FROTA ALVES

Seleção de Produtos de Software Utilizando uma Abordagem Baseada em Engenharia de Requisitos

Dissertação apresentada à Coordenação da Pós-Graduação em Ciência da Computação do Centro de Informática como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação.

Orientador: Jaelson Freire Brelaz de Castro

Recife
Março, 2001

*Dedico este trabalho aos meus pais,
Edna e Ferreira, minha eterna fonte
de inspiração e força.*

AGRADECIMENTOS

A sensação de dever cumprido é realmente uma das mais gratificantes. Sinto-me profundamente feliz por ter realizado este trabalho e finalizado mais uma etapa.

Considero que a verdadeira conquista foi ter tido a oportunidade de vivenciar e aprender com os caminhos que trilhei para chegar até aqui. Foram momentos cheios de alegrias e tristezas, certezas e dúvidas, diversão e solidão, enfim foram experiências singulares.

Seria injusto da minha parte citar aqui apenas alguns poucos nomes. Agradeço então a todos que passaram e ficaram marcados no meu coração e que contribuíram de alguma forma para tornarem esses dois anos um dos períodos mais ricos e felizes da minha vida. Muito obrigada!

RESUMO

À medida que o tamanho e a complexidade dos sistemas de software crescem, aumenta o interesse de desenvolver sistemas baseados em componentes reusáveis, conhecidos na literatura como COTS (*Commercial-Off-The-Shelf*). Os principais benefícios prometidos por esta nova tecnologia são diminuição de custos e tempo de desenvolvimento.

O desenvolvimento de sistemas baseado em produtos COTS compreende basicamente as atividades de avaliação, seleção, adaptação, integração e atualização de componentes. Em particular, o sucesso desses sistemas depende largamente do processo de seleção pois é nesse momento que serão analisadas as características de cada produto candidato e sua conformidade com os requisitos dos *stakeholders*. Nesse contexto, o processo de engenharia de requisitos assume um importante papel durante a seleção dos produtos candidatos. Se um produto inadequado for selecionado, ou seja, se ele não atender aos requisitos dos stakeholders, sérios riscos e problemas poderão ser ocasionados nas fases seguintes, podendo até comprometer todo o sistema.

Apesar da evidente importância da fase de seleção de COTS, poucos métodos têm sido propostos para assistir esse processo, onde a maioria dos métodos existentes na literatura ainda não trata o processo de requisitos de forma satisfatória. Nesse contexto, o presente trabalho propõe uma abordagem baseada em engenharia de requisitos para assistir o processo de seleção de pacotes COTS.

A principal contribuição dessa pesquisa é o método CRE (*COTS-based Requirements Engineering*), ele suporta a modelagem e análise dos requisitos para auxiliar o processo de tomada de decisão entre produtos concorrentes e assim contribuir para o sucesso do processo de seleção. A fim de validar o método proposto, é apresentado um estudo de caso realizado numa organização que utilizou o método CRE para assistir a seleção de uma ferramenta para desenvolvimento em Java.

ABSTRACT

As the size and complexity of software systems grows, the interest in developing systems based on reusable components increases, known in literature as COTS (Commercial-Off-The-Shelf). The potential benefits of this new technology are reduced costs and shorter development time.

The system development based on COTS products basically focuses on the evaluation, selection, adaptation, integration and evolution of components. In particular, the success of COTS-based systems largely depends on the selection process because at this moment the characteristics of each candidate product are evaluated as well as their conformity with stakeholder's requirements. Therefore, if an inadequate product is selected, serious risks and problems can propagate to the subsequent phases, affecting the quality of the resulting system.

Although the evident importance of COTS selection phase, few methods have been proposed to assist this process. Most current methods do not treat the requirements process effectively. In this context, the present work proposes an approach based on requirements engineering to assist the process of COTS packages selection.

The main contribution of this research is the CRE (COTS-based Requirements Engineering) method. It emphasizes the modeling and analysis of requirements in order to help the decision making process between candidate products and contribute for the selection process success. A case study is presented to provide the method validation, in which the organization used the CRE method to assist the selection of a Java development environment.

CONTEÚDO

1	<i>Introdução</i>	1
1.1	Motivações	2
1.2	Estágio Atual do Desenvolvimento Baseado em COTS	3
1.3	Escopo da Dissertação	4
1.4	Objetivos e Abordagem	5
1.5	Contribuições	6
1.6	Estrutura da Dissertação	7
2	<i>Desenvolvimento Baseado em COTS</i>	9
2.1	Motivações para o uso de COTS	10
2.2	Princípios e Características do Desenvolvimento Baseado em COTS	11
2.3	Comparação Entre o Desenvolvimento Convencional e o Baseado em COTS	13
2.3.1	Arquitetura	15
2.3.2	Componentes	16
2.3.3	Metodologia/Processo	17
2.4	Atividades Básicas do Ciclo de Vida COTS	18
2.4.1	Mercado COTS	19
2.4.2	Seleção	20
2.4.3	Adaptação	22
2.4.4	Integração	23
2.4.5	Atualização	24
2.5	Desafios no Processo de Desenvolvimento Baseado em COTS	24
2.6	Considerações Finais	25
3	<i>O Processo de Seleção de COTS</i>	27
3.1	Visão Geral	28

3.2	Desafios e Problemas Encontrados no Processo de Seleção de COTS.....	28
3.3	Trabalhos Relacionados	31
3.3.1	Método OTSO	32
3.3.2	Método PORE	32
3.3.3	Framework STACE	33
3.3.4	Método IusWare	34
3.3.5	Método Análise de Características	35
3.3.6	Aspectos Relevantes.....	36
3.4	Técnicas para Tomada de Decisão	37
3.4.1	Técnica MAUT	38
3.4.2	Técnica WSM.....	39
3.4.3	Técnica AHP	40
3.4.4	Principais Limitações das técnicas para tomada de decisão	41
3.5	Considerações Finais	42
4	<i>Engenharia de Requisitos para Desenvolvimento Baseado em COTS</i>	<i>45</i>
4.1	Visão Geral da Engenharia de Requisitos	46
4.2	O Processo de Engenharia de Requisitos.....	47
4.2.1	Elicitação de Requisitos	48
4.2.2	Análise de Requisitos.....	50
4.2.3	Documentação de Requisitos	51
4.2.4	Validação de Requisitos.....	52
4.3	Uma Proposta do Processo de Engenharia de Requisitos para DBC.....	53
4.3.1	Elicitação de Requisitos para DBC	54
4.3.2	Análise de Requisitos para DBC	55
4.3.3	Documentação de Requisitos para DBC	56
4.3.4	Validação de Requisitos para DBC	57
4.4	Aspectos Relevantes do Processo de Engenharia de Requisitos para DBC	58
4.5	Modelagem de Requisitos Não-Funcionais	59

4.6	Considerações Finais	63
5	<i>O Método CRE</i>	65
5.1	Descrição Geral.....	66
5.2	As Fases do Método CRE.....	67
5.2.1	Identificação	69
5.2.2	Descrição	76
5.2.3	Avaliação.....	87
5.2.4	Aceitação	94
5.3	Considerações Finais	98
6	<i>Estudo de Caso</i>	100
6.1	Objetivos.....	101
6.2	Utilização do Método CRE.....	102
6.2.1	Identificação	102
6.2.2	Descrição	104
6.2.3	Avaliação.....	116
6.2.4	Aceitação	121
6.3	Considerações Finais	121
7	<i>Conclusões e Trabalhos Futuros</i>	125
7.1	Contribuições do Trabalho	126
7.2	Trabalhos Futuros	128
	<i>Glossário</i>	131
	<i>Referências Bibliográficas</i>	133

ÍNDICE DE FIGURAS

<i>Figura 1.1 - Foco da Dissertação.....</i>	<i>5</i>
<i>Figura 2.1 - Diferenças entre o desenvolvimento convencional e o baseado em COTS.....</i>	<i>14</i>
<i>Figura 2.2 - Desenvolvimento de sistemas baseado em COTS.....</i>	<i>18</i>
<i>Figura 2.3 - Atividades do processo de desenvolvimento baseado em COTS.....</i>	<i>19</i>
<i>Figura 2.4 - Técnica Wrapper.....</i>	<i>23</i>
<i>Figura 3.1 - Atividades de avaliação durante o ciclo de vida COTS</i>	<i>28</i>
<i>Figura 3.2 - Dimensões do Processo de Seleção de COTS</i>	<i>30</i>
<i>Figura 3.3 - Princípios do processo de tomada de decisão.....</i>	<i>38</i>
<i>Figura 4.1 - Atividades do Processo da Engenharia de Requisitos no Modelo Espiral</i>	<i>48</i>
<i>Figura 4.2 - Exemplo de grafo SIG.....</i>	<i>62</i>
<i>Figura 5.1 - Processo Iterativo de Aquisição de Requisitos e Seleção de Produtos</i>	<i>66</i>
<i>Figura 5.2 - Visão Geral do Método CRE</i>	<i>68</i>
<i>Figura 5.3 - Elaboração do Critério de Seleção</i>	<i>70</i>
<i>Figura 5.4 - Modelo para lista de produtos candidatos</i>	<i>73</i>
<i>Figura 5.5 – Template para Identificação dos Produtos</i>	<i>76</i>
<i>Figura 5.6 - Mecanismo de Feedback.....</i>	<i>80</i>
<i>Figura 5.7 - Modelo do Documento de Requisitos</i>	<i>81</i>
<i>Figura 5.8 - Modelo para lista de produtos candidatos atualizada</i>	<i>84</i>
<i>Figura 5.9 – Template para descrição dos produtos.....</i>	<i>86</i>
<i>Figura 5.10 - Árvore Hierárquica usando a Técnica AHP</i>	<i>89</i>
<i>Figura 5.11 - Árvore indicando o ranking dos produtos em relação aos critérios.....</i>	<i>91</i>
<i>Figura 5.12 - Template para avaliação dos produtos</i>	<i>93</i>
<i>Figura 5.13 – Template para aceitação dos produtos</i>	<i>98</i>
<i>Figura 6.1 - Grafo SIG do requisito Usabilidade</i>	<i>110</i>
<i>Figura 6.2 - Grafo SIG do requisito Integrabilidade.....</i>	<i>110</i>
<i>Figura 6.3 - Grafo SIG do requisito Manutenibilidade</i>	<i>111</i>
<i>Figura 6.4 - Grafo SIG do requisito Manutenibilidade</i>	<i>111</i>
<i>Figura 6.5 - Grafo SIG do requisito Produtividade.....</i>	<i>112</i>

<i>Figura 6.6 – Grafo SIG com os relacionamentos entre os requisito.....</i>	<i>113</i>
<i>Figura 6.7 - Uso de ferramenta ExpertChoice</i>	<i>118</i>
<i>Figura 6.8 – Modos de comparação aos pares das ferramentas candidatas</i>	<i>119</i>
<i>Figura 6.9 - Gráfico para análise de resultados (critérios X alternativas).....</i>	<i>120</i>
<i>Figura 6.10 - Percentual de tempo gasto nas fases do método CRE.....</i>	<i>123</i>
<i>Figura 7.1 - Protótipo Assistente de Avaliação</i>	<i>129</i>

ÍNDICE DE TABELAS

<i>Tabela 2.1 - Comparação entre modelos de desenvolvimento.....</i>	<i>15</i>
<i>Tabela 3.1 - Resumo das etapas cobertas pelos métodos descritos.....</i>	<i>37</i>
<i>Tabela 5.1 - Escore de conformidade entre produtos e requisitos</i>	<i>74</i>
<i>Tabela 5.2 - Checklist de Descrição do Produto</i>	<i>83</i>
<i>Tabela 5.3 - Valores dos Atributos Peso e Escore</i>	<i>89</i>
<i>Tabela 5.4 - Escala de importância usada nas comparações da técnica AHP</i>	<i>90</i>
<i>Tabela 5.5 - Matriz de comparação entre critérios</i>	<i>90</i>
<i>Tabela 5.6 - Ranking entre os critérios.....</i>	<i>91</i>
<i>Tabela 5.7 - Matriz de comparação entre produtos em relação ao critério segurança ..</i>	<i>91</i>
<i>Tabela 5.8 - Ranking entre os produtos candidatos.....</i>	<i>92</i>
<i>Tabela 6.1 - Requisitos críticos que os IDEs Java devem atender.....</i>	<i>103</i>
<i>Tabela 6.2 Lista Inicial de Ferramentas Candidatas</i>	<i>104</i>
<i>Tabela 6.3 - Lista atualizada das ferramentas candidatas</i>	<i>105</i>
<i>Tabela 6.4 - Avaliação preliminar das ferramentas</i>	<i>106</i>
<i>Tabela 6.5 - Documento de Requisitos.....</i>	<i>107</i>
<i>Tabela 6.6 - Prioridades dos Requisitos</i>	<i>108</i>
<i>Tabela 6.7 - Refinamento do Requisito “Possuir editor de código”</i>	<i>109</i>
<i>Tabela 6.8 - Motivos da Eliminação das ferramentas</i>	<i>116</i>
<i>Tabela 6.9 - Avaliação das ferramentas usando a técnica WSM.....</i>	<i>117</i>
<i>Tabela 6.10 - Tempo gasto para aplicar cada fase do Método CRE</i>	<i>122</i>

Capítulo 1

Introdução

Este capítulo mostra as principais motivações para realização deste trabalho. Em seguida, apresenta o estágio atual do desenvolvimento baseado em COTS. As seções seguintes descrevem o escopo da dissertação, os objetivos e a abordagem utilizada, as principais contribuições e finalmente, a estrutura do trabalho.

1.1 MOTIVAÇÕES

À medida que o tamanho e a complexidade dos sistemas de software crescem, aumenta o interesse de desenvolver sistemas baseados em componentes reusáveis. Os principais benefícios prometidos por esta nova tecnologia são diminuição de custos e tempo de desenvolvimento [OBE98] [TRA97b]. Atualmente, existe uma grande variedade de produtos reusáveis disponíveis no mercado, esses produtos são chamados de COTS (*Commercial-Off-The-Shelf*). Exemplos típicos de produtos COTS são pacotes de serviços de e-mail, tais como: Eudora, Pegasus e Outlook Express; sistemas antivírus, por exemplo: Dr. Solomon, F-Prot e Sophos; ferramentas de gerenciamento de requisitos, tais como: RequisitePro, DOORS e Cradle; ferramentas para desenvolvimento em Java, por exemplo: VisualAge, JBuilder e JDeveloper.

Num processo de Desenvolvimento Baseado em COTS (DBC), as atividades de avaliação e seleção de produtos candidatos são alguns dos aspectos fundamentais do seu ciclo de vida [KON95]. Visto que é nesse momento onde serão analisadas as características de cada COTS candidato para em seguida escolher um ou mais produtos que se mostraram adequados. Se um produto inadequado for selecionado, isso pode acarretar sérios problemas nas fases seguintes. O sucesso de sistemas baseados em COTS depende largamente de um preciso entendimento das capacidades e limitações de cada produto [NCU00a]. A fim de atingir esse sucesso, a avaliação dos produtos deve ser um processo simultâneo com a aquisição de requisitos do usuário [ALV00].

Inúmeros exemplos presentes na literatura comprovam que o desenvolvimento de software ainda está num estágio inadequado e propenso à falhas. O desastre ocorrido com o Ariane 5 é um exemplo de falha ocasionada por conflitos entre especificações. Sob o ponto de vista da engenharia de requisitos, o desastre foi devido à uma análise incorreta de requisitos que mudaram na especificação do Ariane 5 em relação aos modelos anteriores [NUS97][GIG96]. Outro exemplo clássico de falha de sistema ocasionada por vários erros cometidos durante o processo de desenvolvimento, inclusive por uma incorreta

especificação de requisitos, é o caso da pane que aconteceu no sistema de controle de ambulâncias de Londres, ocorrido em 1992 [FIN96] [BRE99].

A comunidade de engenharia de software concorda que os erros ocasionados durante a fase de engenharia de requisitos estão entre os mais caros para corrigir depois que o sistema está em funcionamento [BOE84] [KOT97] [WEI96] [LOU95]. Analisando esses sintomas em relação a sistemas baseados em COTS, observa-se que as conseqüências são muito mais sérias [NCU99] [KUN00]. Uma vez que o sucesso de qualquer sistema baseado em COTS depende largamente de uma seleção correta dos produtos candidatos, onde esses produtos são selecionados de acordo com o grau de conformidade com os requisitos dos usuários. Dessa forma, é verificável que a engenharia de requisitos adquire um papel de destaque no processo de seleção de produtos COTS. Como será mostrado nos capítulos seguintes, a seleção de produtos que realmente atendam aos requisitos dos usuários pode contribuir consideravelmente para uma diminuição de gastos e esforço nas fases seguintes.

1.2 ESTÁGIO ATUAL DO DESENVOLVIMENTO BASEADO EM COTS

Apesar dos sistemas baseados em COTS oferecerem às organizações a oportunidade de reduzir custos e tempo de desenvolvimento dos sistemas de software [OBE97a], existem várias dificuldades que essas organizações precisam superar. Como qualquer outra abordagem de engenharia de software, o desenvolvimento de sistemas baseado em COTS também possui inúmeros problemas e desafios associados. Ou seja, desenvolvimento baseado em COTS não é a solução mágica que irá resolver todos os problemas existentes na engenharia de software [OBE98].

O mercado de produtos COTS está sempre em contínua mudança, ele abrange diversas áreas de software, por exemplo: sistemas operacionais, aplicações WEB, sistemas distribuídos, aplicações para gerenciamento organizacional, etc. Existem inúmeros vendedores atuando nesse mercado e aparentemente eles dispõem de soluções semelhantes e com preços competitivos. Entretanto, é necessário fazer uma análise mais profunda de cada produto a fim de avaliar quais os reais benefícios e implicações de cada opção

disponível [OBE97a]. Nesse contexto, outros aspectos devem ser levados em consideração, tais como: garantias, direito de uso, licenças, entre outros [CHA98].

O uso de DBC representa importantes mudanças nas organizações, pois não se trata apenas de um problema técnico mas também de uma necessidade de reeducação em vários aspectos organizacionais [KUN00]. Quase sempre há profundas mudanças de responsabilidades, uma vez que o papel do programador deve ser trocado pelas tarefas de adaptação e integração. A fim de obter ao máximo todos os benefícios da abordagem COTS, as organizações precisam passar por um amplo processo educativo. Em particular, a forma como os incentivos são dados pode desencorajar o uso de COTS uma vez que muitas organizações estipulam o salário dos desenvolvedores de acordo com a quantidade de linhas de código que eles desenvolvem [LYN97]. Além disso, normalmente o salário dos gerentes é baseado em resultados anuais enquanto sistemas baseados em COTS podem necessitar de algum tempo até que os benefícios sejam observados.

Grande parte das pesquisas realizadas na área de DBC focaliza nos processos de adaptação [VOA97] [MAC98], integração [BRO95] [VID96] [DEA97] [DEL97] [HAI97] e arquitetura [GAR95] [FOX97] [CLE95]. Recentemente, a comunidade de DBC tem verificado a importância da fase de seleção e desde então alguns métodos têm sido propostos, dentre os mais importantes estão: OTSO [KON96a] PORE [NCU00a] e STACE [KUN99]. Entretanto, como será visto mais adiante, a maioria deles não aborda em detalhes o processo de engenharia de requisitos o que resulta numa grave restrição a esses métodos.

1.3 ESCOPO DA DISSERTAÇÃO

Esta dissertação enfoca as fases de avaliação e seleção de produtos COTS, as demais fases do ciclo de vida COTS serão descritas mas serão abordadas superficialmente. Nesse contexto, a engenharia de requisitos assume uma posição de destaque pois ela irá guiar todo o processo de seleção de COTS. Já que, como foi visto na seção 1.1, a aquisição dos requisitos dos *stakeholders* é uma etapa fundamental durante a seleção de COTS. Será dada ênfase às atividades de elicitação e especificação dos requisitos não-funcionais, pois como

será visto nos próximos capítulos os requisitos de qualidade, tais como: confiabilidade, performance e usabilidade são importantes critérios de seleção de produtos. A Figura 1.1 mostra o escopo dessa dissertação que engloba as fases de avaliação e seleção do desenvolvimento baseado em COTS assim como enfoca o processo de engenharia de requisitos.

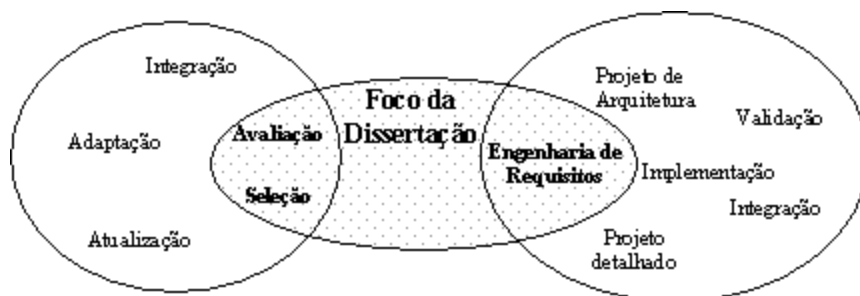


Figura 1.1 - Foco da Dissertação

Vale salientar que esse trabalho não aborda reuso de software. Já que durante o desenvolvimento de sistemas baseados em COTS, o código-fonte dos produtos normalmente não está disponível. A seleção de múltiplos COTS, onde vários produtos são adquiridos a fim de obter as funcionalidades desejadas também não será tratada aqui, essa dissertação enfoca o processo de seleção de um único produto COTS. O motivo para essa decisão é porque primeiramente precisa-se entender os problemas encontrados durante a seleção de um único produto antes de considerar os problemas e desafios para selecionar inúmeros produtos.

1.4 OBJETIVOS E ABORDAGEM

O principal objetivo dessa dissertação é fornecer um método baseado em requisitos para assistir os processos de avaliação e seleção de produtos COTS. Esse método enfatiza o processo de aquisição e modelagem dos requisitos não-funcionais. Além disso, serão associadas outras técnicas e ferramentas presentes na literatura nas fases que compõem o método. A fim de verificar a viabilidade dessa proposta, será realizado um estudo de caso para seleção de ferramentas para desenvolvimento Java. Essa pesquisa é estruturada a partir

de 4 hipóteses básicas que deverão ser examinadas no decorrer dos próximos capítulos, são elas:

H1 Os métodos para seleção de COTS existentes na literatura não abordam de forma satisfatória os requisitos dos stakeholders.

H2 Novos desafios surgem no processo de engenharia de requisitos para desenvolvimento baseado em COTS que não existiam no processo tradicional.

H3 Requisitos não-funcionais são aspectos fundamentais durante o processo de seleção.

H4 O método proposto é simples de ser usado e proporciona assistência nas tarefas de avaliação e seleção de produtos COTS.

1.5 CONTRIBUIÇÕES

Esta dissertação fornece várias contribuições para a área de Desenvolvimento Baseado em COTS (DBC), em particular elas estão diretamente relacionadas com a melhoria do processo de seleção de produtos COTS. As contribuições mais significativas são:

(i) O método CRE (*COTS-based Requirements Engineering*) é proposto para fornecer um processo sistemático de aquisição dos requisitos durante as atividades de avaliação e seleção de COTS;

(ii) O método proposto permite uma especificação mais detalhada dos requisitos não-funcionais dos usuários a fim de serem usados no processo de tomada de decisão entre os produtos candidatos;

(iii) Diretrizes para descrever o critério de seleção de forma que ele possibilite a descrição e análise de todos os aspectos importantes para seleção de COTS;

(iv) Estratégias para guiar o processo de seleção usando modelos para aquisição de requisitos, *templates* para descrição dos produtos e técnicas para tomada de decisão;

(v) O método CRE permite que os resultados obtidos em avaliações anteriores sejam documentados e reutilizados em casos semelhantes.

1.6 ESTRUTURA DA DISSERTAÇÃO

Além deste capítulo introdutório, esse trabalho consiste de mais seis capítulos, são eles:

Capítulo 2 – Desenvolvimento Baseado em COTS

Esse capítulo mostra as principais motivações para usar a abordagem COTS, assim como apresenta algumas definições e conceitos relacionados com os termos componentes e COTS. Depois será feita uma comparação entre o processo de desenvolvimento convencional e o baseado em COTS. Além disso, o ciclo de vida COTS será mostrado e cada uma das fases será descrita em seções individuais.

Capítulo 3 – O Processo de Seleção de COTS

Esse capítulo descreve uma visão geral da área, onde serão mostrados os problemas e desafios associados com o processo de seleção de COTS. Em seguida, serão examinados alguns dos trabalhos mais significativos da área. Finalmente, algumas técnicas de tomada de decisão serão descritas e avaliadas quanto à sua usabilidade para auxiliar o processo decisório entre pacotes COTS candidatos.

Capítulo 4 – Engenharia de Requisitos para Desenvolvimento Baseado em COTS

Nesse capítulo, é mostrada uma visão geral da engenharia de requisitos, onde serão descritas cada fase desse processo. Em seguida, o processo de engenharia de requisitos para DBC será analisado, onde deverão ser mostradas algumas técnicas para aquisição e modelagem de requisitos e será dada ênfase especial aos requisitos não-funcionais.

Capítulo 5 – O Método CRE

Neste capítulo é apresentada a principal contribuição desse trabalho, o Método CRE (*COTS-Based Requirements Engineering*). Esse método é orientado a requisitos e auxilia o processo de seleção de COTS. Ele consiste das fases de identificação, descrição, avaliação e aceitação.

Capítulo 6 – Estudo de Caso

Esse capítulo mostra um estudo de caso real com o objetivo de validar o método proposto. Será realizado um estudo de como o método CRE auxilia o processo de seleção de ferramentas para desenvolvimento em Java. Nesse contexto, serão avaliadas a viabilidade e facilidade de cada fase do método.

Capítulo 7 – Conclusões e Trabalhos Futuros

Este capítulo mostra as conclusões obtidas durante o desenvolvimento desse trabalho, assim como as principais contribuições que ele fornece para área de DBC. Serão mostrados alguns possíveis trabalhos futuros. Assim como será feito um direcionamento para futuras pesquisas nessa área.

Capítulo 2

Desenvolvimento Baseado em COTS

Este capítulo aborda o desenvolvimento de sistemas baseado em COTS. Serão mostradas algumas motivações para o uso de COTS, em seguida são fornecidas várias definições para esse termo. Além disso, o processo de desenvolvimento baseado em COTS será comparado com o processo tradicional. Finalmente, as fases do ciclo de vida COTS serão examinadas.

2.1 MOTIVAÇÕES PARA O USO DE COTS

Desenvolver sistemas através do uso de componentes reusáveis tem sido um sonho acalentado pela comunidade de engenharia de software desde algumas décadas atrás. Em 1968, durante uma conferência da NATO, McIlroy [MCL68] já vislumbrava uma indústria de componentes de software reutilizáveis. A necessidade do mercado em desenvolver produtos confiáveis e estáveis, num curto intervalo de tempo e com custos reduzidos, tem conduzido um crescente interesse para o desenvolvimento de sistemas baseados em COTS. O uso recente dessa abordagem é baseado numa longa história de trabalho em sistemas modulares, projeto estruturado e mais recentemente em sistemas orientados a objeto [BRO97].

Esta abordagem, conhecida como CBSE (*Component-Based Software Engineering*), focaliza basicamente na seleção e integração de componentes de software reusáveis para prover soluções de sistemas a um custo de desenvolvimento mínimo [BRO98]. Existem diversos fatores que explicam esse interesse no desenvolvimento de sistemas usando componentes COTS, entre eles [TRA97a]:

- O aumento da competição para prover soluções de software com maior qualidade e desenvolvidas rapidamente;
- A crescente demanda por softwares extensos e complexos, que usualmente não podem ser desenvolvidos por uma única organização;
- A diversidade no mercado tanto de produtos COTS de domínio específico quanto de componentes genéricos;
- O aumento do grau de interoperabilidade e de concordância de padrões entre os produtos COTS, o que garante uma significativa redução de tempo e esforço gastos durante a fase de integração do produto;
- O crescente esforço em pesquisas na área, fornecendo novas técnicas e métodos para suportar DBC.

Além de todos esses aspectos, outro fator que incentiva o desenvolvimento de sistemas baseado em COTS é que a abordagem de utilizar componentes reusáveis já é realizada com maturidade em setores industriais como telefonia, eletrônica e hardware. O aumento considerável no número de organizações que adotam o uso de COTS no desenvolvimento de software fortalece a credibilidade dessa abordagem. Governo, indústria, comércio e instituições de pesquisa têm demonstrado interesse pelo tema [SOU99].

Na década de 90, importantes organizações como NASA (*National Aeronautics and Space Administration*) e DoD (Departamento de Defesa dos Estados Unidos), declararam ter obtido sucesso com redução de custos e prazos de desenvolvimento de software devido ao uso de componentes COTS [OBE97b]. Tanto que em 1994 foi estabelecida uma política de incentivo ao uso de COTS no DoD. Em 1998, McDermid declarou que dependendo da aplicação, o uso de COTS pode suportar até 80% da funcionalidade com custo de apenas 2% do desenvolvimento tradicional [MCD98].

2.2 PRINCÍPIOS E CARACTERÍSTICAS DO DESENVOLVIMENTO BASEADO EM COTS

Ao mesmo tempo em que a abordagem de desenvolvimento baseado em COTS tem ganhado popularidade, tem-se criado uma crescente confusão entre os termos componentes e COTS. Nesse contexto, o termo componente é bastante genérico, ele pode se referir a diferentes tipos e níveis de software [MOR00]. Contudo, ainda não existe um consenso sobre a caracterização do que seria um componente de software [WER00]. Durante o 1º Workshop Internacional de CBSE (*Component-Based Software Engineering*), ocorrido em 1998, Brown e Wallnau [BRO98] mostraram diversas definições do que seria um componente. Algumas das visões discutidas pela comunidade da área e apresentadas durante esse encontro são:

- “Um componente como um elemento arquitetural – este componente deve prover e estar em conformidade com um conjunto de interfaces”;

- “Um componente como um elemento implementacional – acessado através de interfaces bem documentadas que podem ser descobertas em tempo de execução”;
- “Um componente como um elemento implementacional – que faz parte de um contexto arquitetural específico”;
- “Um componente de negócio – representa a implementação de um conceito autônomo de negócio ou processo”.

Atualmente, a comunidade de CBSE tem direcionado a conceituação do que seriam componentes de software para a seguinte definição [CBSE00]:

- “Componentes são artefatos autocontidos, claramente identificáveis, que descrevem ou realizam uma função específica e têm interfaces claras em conformidade com um dado modelo de arquitetura de software, documentação apropriada e um grau de reutilização definido”.

Um dos focos da pesquisa atual em CBSE é nas soluções de *middleware* e padrões para ambientes distribuídos. Nos últimos anos, diversas tecnologias para infra-estrutura de componentes surgiram no mercado e a tendência é que tais abordagens sejam padronizadas. As principais tecnologias presentes no mercado são: CORBA (*Common Object Request Broker Architecture*), Java Beans e Enterprise Java Beans, COM (*Common Object Model*) e DCOM (*Distributed Common Object Model*). Dentre essas tecnologias de componentes, CORBA, COM e DCOM são direcionadas especialmente para o desenvolvimento de aplicações distribuídas. Como apelo para sua utilização, estes padrões propõem um alto nível de abstração para viabilizar a interoperabilidade dos componentes através da utilização de linguagens intermediárias, geradores de código e linguagens para especificação de interfaces.

Na literatura disponível, existem inúmeras definições para o termo COTS (*Commercial-Off-The-Shelf*), por exemplo:

- “Software de terceiros” [VOA98];

- “Software comercial, desenvolvido sem nenhuma aplicação específica em mente” [MCD98];
- “Software comercial, sem código fonte disponível e com versões periódicas” [DEI99];
- “Software que existe, está disponível para o público, pode ser comprado ou licenciado e que é suportado e atualizado pelo vendedor que retêm os direitos de propriedade” [OBE97a].

Este trabalho adota a última definição pois ela é considerada a mais abrangente e aceita pela comunidade da área. Outro aspecto importante relativo ao desenvolvimento baseado em COTS diz respeito ao número de componentes que são integrados para fornecer uma solução de software. Segundo Foreman [FOR99], existem dois tipos de sistemas baseados em COTS. O primeiro e mais simples é chamado de CSS (*COTS-Solution Systems*), onde somente um produto é usado para prover a solução. Nesse tipo de sistema, a organização deve se adaptar ao produto e toda infra-estrutura e manutenção é provida pelo vendedor. O segundo tipo de sistema baseado em COTS é chamado de CIS (*COTS-Intensive Systems*), nesse caso múltiplos produtos são integrados para prover a funcionalidade do sistema. Nesse contexto, considera-se que sistemas CIS permitem maior flexibilidade aos processos de mudança organizacional pois a própria organização é responsável pela infra-estrutura e manutenção [WAL98]. Entretanto, os processos de seleção, integração e manutenção desses sistemas são muito mais complexos. O desenvolvimento baseado em COTS possui várias características diferentes do desenvolvimento convencional. Para um melhor entendimento da abordagem COTS é necessário que tais diferenças sejam analisadas com cuidado. Assim, este assunto será tratado na próxima seção.

2.3 COMPARAÇÃO ENTRE O DESENVOLVIMENTO CONVENCIONAL E O BASEADO EM COTS

Como discutido anteriormente, existe um crescente interesse em usar COTS como um caminho viável para desenvolver soluções de software num curto espaço de tempo. Uma

grande variedade de ferramentas, técnicas, modelos de processo e aplicações tem sido desenvolvidos para suportar o desenvolvimento baseado em COTS. Existe uma considerável mudança de paradigma entre o desenvolvimento tradicional onde o software é modelado e implementado e o desenvolvimento baseado em COTS que sugere a aquisição de produtos já implementados e disponíveis no mercado. A Figura 2.1 mostra a mudança entre essas duas formas de desenvolvimento, no desenvolvimento convencional existe uma seqüência linear das atividades, enquanto no desenvolvimento baseado em COTS existem *tradeoffs* simultâneos entre o processo de requisitos, a arquitetura de software e os produtos COTS que serão adquiridos. Em [ROS01] é fornecida uma abordagem para tratar esses *tradeoffs* usando uma abordagem formal para descrever arquiteturas de software que atendem aos requisitos especificados e assim auxiliam o processo de seleção de componentes COTS.

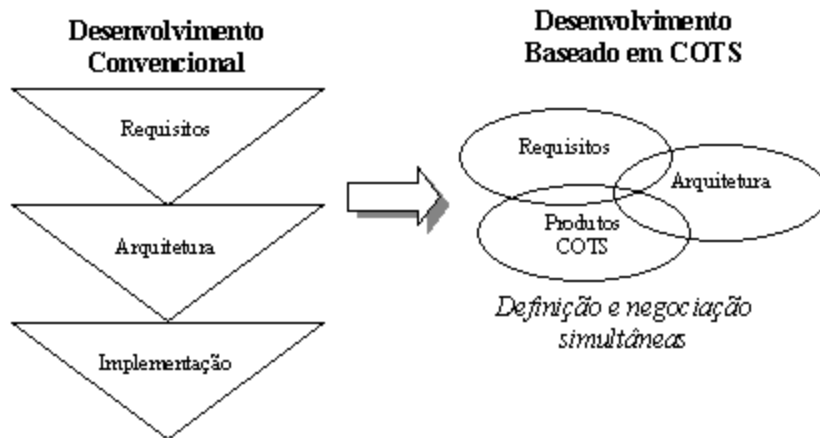


Figura 2.1 - Diferenças entre o desenvolvimento convencional e o baseado em COTS

Além disso, a própria natureza dos COTS sugere que o modelo de desenvolvimento de software baseado em COTS deve ser diferente do modelo convencional [ALV00]. A Tabela 2.1[AOY98] resume as principais características entre o desenvolvimento de software convencional e o baseado em COTS, onde tais características serão descritas em seguida.

Características	Convencional	COTS
Arquitetura	Monolítica	Modular

Componentes	Implementação	Interface
	Caixa-branca	Caixa-preta
Metodologia/Processo	Desenvolvimento (total)	Composição

Tabela 2.1 - Comparação entre modelos de desenvolvimento

2.3.1 Arquitetura

Em geral, definir uma arquitetura de software consistente, extensiva e robusta tem sido uma tarefa árdua durante o desenvolvimento de software convencional. Em sistemas baseados em COTS, essa tarefa se torna muito mais complexa. Inicialmente, como discutido em [CLE95] e [TRA97b], a arquitetura desses sistemas quase sempre sofre significativa influência da arquitetura dos COTS que integram o sistema final. Dessa forma, a seleção de diferentes pacotes de software tem quase sempre um grande impacto na arquitetura global do sistema. Segundo Garlan [GAR95], superar “enganos arquiteturais” (*architectural mismatches*) entre COTS selecionados é outra complexa tarefa não encontrada no desenvolvimento convencional [TRA97a]. São descritos alguns aspectos com possibilidade de ocorrer enganos:

- Natureza do componente, incluindo suporte de infra-estrutura que o componente pode necessitar, modelo de controle de sincronização e modelo de dados;
- Natureza dos conectores, onde problemas podem ser encontrados nos protocolos utilizados e tipos de dados a serem transmitidos pelos conectores;
- Estrutura da arquitetura, que pode apresentar problemas na forma que um componente pensa que a arquitetura se comporta em termos das iterações entre os componentes;
- Processo de construção, principalmente relacionados a problemas de heterogeneidade dos componentes.

Para tentar minimizar estes possíveis enganos, Garlan propõe alguns *guidelines*:

- Utilizar componentes previamente desenvolvidos para serem reutilizados, onde os pontos de conexão em suas interfaces estejam bem documentados e explicitados;
- Utilizar *wrappers* sempre que problemas de compatibilidade de interfaces ocorrerem;
- Usar sempre que possível padrões arquiteturais de sucesso e bem documentados.

A abordagem COTS enfatiza o uso de uma arquitetura modular, onde se pode desenvolver parcialmente um sistema e incrementalmente adicionar novas funcionalidades através da introdução ou substituição de componentes [AOY98]. Normalmente, tais sistemas são complexos sendo assim necessário uma representação detalhada da arquitetura dos mesmos.

2.3.2 Componentes

No desenvolvimento convencional, normalmente toda a funcionalidade do sistema é implementada e o código-fonte está acessível, ou seja, ele é considerado uma caixa-branca. Esta última característica permite que os desenvolvedores executem testes com bom nível de precisão em relação às possíveis vulnerabilidades do software [MAC98]. Já os componentes COTS são considerados caixas-pretas, quer dizer, na maioria dos casos são entregues como objetos executáveis e possuem mecanismos ou licenças que proíbem a análise do código-fonte. A falta desse código dificulta o entendimento preciso de como o componente se comporta internamente. Essa característica torna difícil a avaliação da vulnerabilidade desses componentes e conseqüentemente a avaliação da confiabilidade do sistema final.

Dessa forma, os testes de sistemas baseados em COTS são mais complexos do que os testes de sistemas desenvolvidos de forma convencional [WEY98]. Entretanto, existem alguns métodos baseados em testes de caixa-preta que podem ser usados para avaliar a confiabilidade de produtos COTS [MAC98]. Uma metodologia bastante utilizada é a técnica de injeção de falhas. O objetivo desta técnica é prever o impacto que a falha de um componente COTS poderá causar ao resto do sistema [VOA97].

Um dos conceitos mais importantes da abordagem COTS é a sua interface e portanto a interação entre os componentes através de suas interfaces. As interfaces são particularmente importantes para a composição e customização dos componentes. Através delas pode-se entender as funcionalidades e restrições sobre as quais os componentes devem operar. A descrição da interface de um componente consiste basicamente de duas partes [BER98]:

- Assinatura – descreve as operações que o componente provê;
- Comportamento – descreve o comportamento do componente.

Componentes podem implementar (exportar) uma ou mais interfaces e eles também podem usar (importar) interfaces provenientes de outros componentes. Conseqüentemente, exportar interfaces corresponde aos serviços que o componente fornece ao ambiente e importar interfaces corresponde aos serviços que o componente necessita do ambiente para implementar os serviços exportados. Juntamente, as interfaces de importação e exportação de todos os componentes restringem a possibilidade de conexão entre componentes. Para construir um sistema composto de componentes, é necessário definir um mapeamento através da conexão compatível de interfaces de importação e exportação [BER98].

2.3.3 *Metodologia/Processo*

A maioria das metodologias de desenvolvimento, incluindo os populares métodos orientados a objetos, assume que o desenvolvimento de sistemas é feito quase totalmente a partir da implementação de todas as funcionalidades necessárias [AOY98]. Entretanto, esses modelos não são considerados adequados ao desenvolvimento baseado em COTS. Nesse tipo de desenvolvimento a noção de construir um sistema através de implementação tem sido substituída pela idéia de compor e integrar componentes de software já existentes no mercado.

É importante lembrar que existem diversas metodologias para desenvolvimento dos próprios componentes. O método Catalysis [WIL98], por exemplo, é um dos mais aceitos pela comunidade de DBC pois o detalhamento do seu processo é bastante consistente. Ele

cobre todas as fases do desenvolvimento de um componente, desde a especificação até sua implementação. Esse método provê especial ênfase na parte de especificação das interfaces. Essa abordagem é interessante pois a composição e colaboração entre os componentes para formar a arquitetura é dada pelas suas interfaces [WER00].

2.4 ATIVIDADES BÁSICAS DO CICLO DE VIDA COTS

Um princípio básico do desenvolvimento de software baseado em COTS é procurar adquirir produtos para a maior parte da funcionalidade desejada, ao invés de desenvolvê-la [SOU99]. O processo de desenvolvimento, de uma forma geral, corresponde à identificação dos produtos disponíveis no mercado, seleção de um ou mais COTS considerados adequados e integração deles ao sistema final, a Figura 2.2 ilustra esse processo. Dessa forma, a maioria dos modelos de ciclo de vida para sistemas baseados em COTS considera as atividades essenciais de identificação, seleção e integração. Existem poucas variações entre os diversos modelos propostos na literatura.

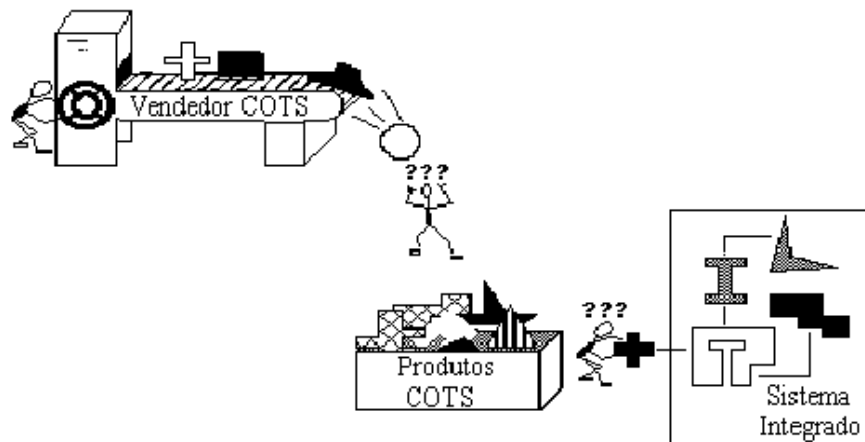


Figura 2.2 - Desenvolvimento de sistemas baseado em COTS

Esse trabalho adota o modelo proposto por Brown [BRO96] pois ele é um dos mais aceitos pela comunidade de DBC, Figura 2.3. As seções seguintes tratam separadamente as cinco

atividades do ciclo de vida COTS segundo esse modelo, que são: *Mercado COTS*, *Seleção*, *Adaptação*, *Integração* e *Atualização*.

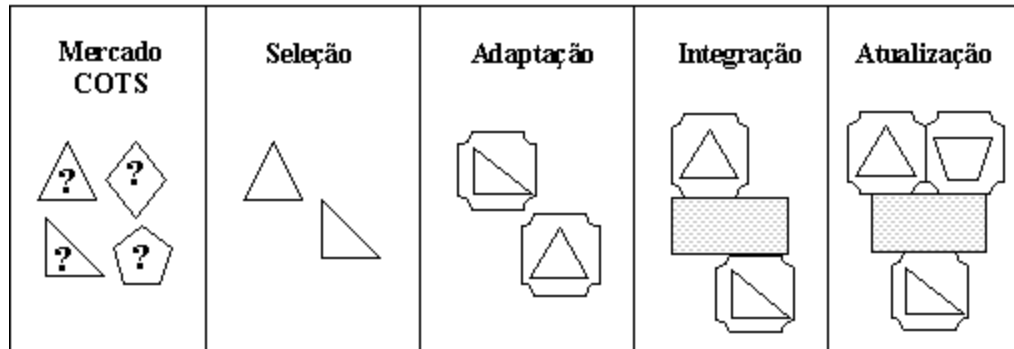


Figura 2.3 - Atividades do processo de desenvolvimento baseado em COTS

2.4.1 Mercado COTS

A fase de avaliação do mercado COTS é considerada um processo complexo e fracamente definido, onde grande parte das informações é difícil de obter [HAI97]. Essa fase inclui as seguintes atividades: elicitar com os usuários os requisitos gerais do sistema, identificar e classificar os COTS em conjuntos de produtos com características semelhantes e finalmente avaliá-los para subsequente seleção. Normalmente, esse processo inicia com poucas informações sobre os produtos. A informação disponível pode consistir do nome do produto, uma descrição superficial das suas principais funcionalidades e alguma idéia sobre o ambiente operacional no qual ele deverá operar [BRO97].

Ao longo dessa fase, as propriedades de cada produto devem ser identificadas e analisadas. Tais propriedades incluem uma descrição das funcionalidades e restrições dos possíveis candidatos. Outro aspecto importante é obter uma descrição do comportamento de cada produto no que diz respeito à sua interação com o restante do sistema. Esse tipo de informação diminui os riscos de possíveis erros e conseqüentemente torna o processo de avaliação menos ad hoc [KUN99]. Durante o processo de avaliação, um importante critério de eliminação de pacotes é analisar a sua dependência em relação a outros produtos. Candidatos COTS dependentes de outros produtos que podem violar algum requisito

essencial do sistema devem ser eliminados logo cedo no processo de avaliação [TRA97a]. Por exemplo, produtos que são incompatíveis com o sistema operacional requerido, deveriam ser rejeitados logo no início do processo.

Segundo Wallnau [WAL98], existem duas visões de avaliar COTS, a primeira e mais simples é chamada *superficial*. Esse tipo de avaliação consiste em adquirir os produtos que pelo menos aparentemente são adequados para uso. Essa abordagem garante custo mínimo durante o esforço de avaliação pois os produtos que falham durante um estágio particular do processo de avaliação são logo eliminados e o primeiro produto que passar por todos os estágios é escolhido para posterior integração. É importante lembrar que os estágios do processo de avaliação são definidos, na maioria das vezes, por cada organização e normalmente não existe um processo padrão.

A outra visão de avaliação é chamada de *exaustiva*. Nessa abordagem todos os candidatos são avaliados detalhadamente ao longo de todos os estágios do processo de avaliação. O objetivo da avaliação exaustiva é garantir que um ou mais produtos apropriados e de garantida qualidade serão selecionados e posteriormente integrados ao sistema. Tal abordagem requer um custo adicional de tempo e recursos. Portanto, em situações onde existe um alto grau de incerteza e um grande número de candidatos, a avaliação superficial oferece menos riscos.

2.4.2 Seleção

O objetivo dessa fase é selecionar um ou mais candidatos que passaram com sucesso pela fase de avaliação e que posteriormente deverão integrar o sistema final. Normalmente, a seleção de um produto requer uma análise de prioridades e conseqüente negociação. Como resultado, estabelecer um critério de seleção de produtos é considerado como uma das atividades mais importantes desse processo. Esse critério deve ter outras informações além de simplesmente procurar satisfazer os requisitos dos *stakeholders*. É necessário também avaliar aspectos relativos aos vendedores dos produtos [TRA97a], como por exemplo:

- Reputação e maturidade do vendedor;

- Garantias de suporte do vendedor;
- Estabilidade do vendedor;
- Modificações do produto em futuras versões.

Além desses aspectos citados acima, existem outros fatores não-técnicos que influenciam o critério de seleção [HAI97]:

- Prazo para obter uma decisão final da seleção dos produtos;
- Número de candidatos que estão sendo avaliados;
- Recursos disponíveis para as atividades de avaliação e seleção;
- Utilização do produto em outros sistemas anteriores.

As atividades de priorização e negociação são normalmente realizadas a partir da análise e avaliação dos benefícios que cada produto pode trazer ao sistema. Por exemplo, um produto que atende todos os requisitos críticos do sistema deve ter prioridade sobre aquele que provê somente parte dos requisitos. Entretanto, se o vendedor desse primeiro produto não tiver uma boa reputação no mercado, deve-se tomar cuidados extras no momento de selecionar o primeiro produto em detrimento de outro.

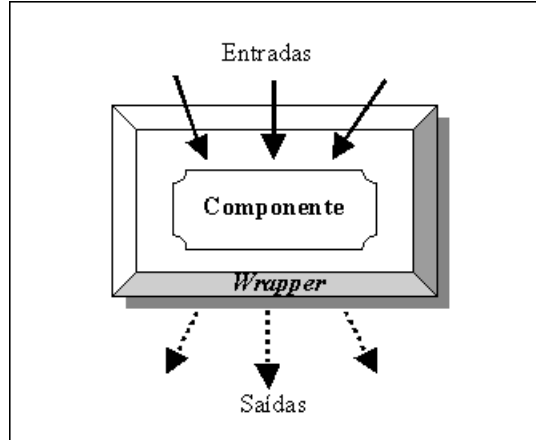
Na maioria dos casos, os processo de avaliação e seleção de produtos COTS ocorrem em paralelo. Ou seja, enquanto são feitas a identificação e avaliação de alguns produtos, outros já podem estar sendo priorizados e negociados. Por esse motivo que muitos autores [TRA97b] [NCU99] [NWO99] consideram essas duas atividades como um processo único. A idéia de que COTS são *plug and play* e que podem ser integrados ao sistema como peças de Lego não descreve a realidade desse tipo de desenvolvimento. Quase sempre é preciso adaptar os produtos para que eles funcionem corretamente e em conformidade com as necessidades dos usuários. Assim, a adaptação é a próxima atividade do ciclo de vida COTS.

2.4.3 Adaptação

Normalmente, cada produto é desenvolvido para atender diferentes tipos de usuários e aplicações, ao invés de um único usuário. Esse fato gera, em alguns casos, a necessidade de adaptar os COTS para cada sistema em particular. Muitas vezes, a adaptação também é necessária por causa de defeitos dos próprios COTS, que podem gerar problemas com a conexão entre dois produtos e conseqüentemente comprometem a integração geral do sistema [SOU99]. A atividade de adaptação inclui o desenvolvimento de todos os adaptadores de software e complementos necessários para que se possa posteriormente realizar uma integração com sucesso. A adaptação de um produto deve ser baseada em regras que garantem a diminuição de conflitos entre eles. Uma técnica bastante simples é o uso de scripts, que são escritos como uma proteção entre as entradas e saídas [BRO97].

Entretanto, existem várias outras estratégias mais sofisticadas para fazer a adaptação de COTS, como por exemplo: capas de proteção ou *wrappers*, análise de atenuação de riscos e localização de falhas [MAC98]. As técnicas de injeção de falhas podem ser usadas para identificar comportamento inaceitável exibido por um COTS [VOA97], porém elas não corrigem automaticamente esses problemas. Como esses produtos não possuem o código disponível, normalmente não é possível reparar falhas detectadas.

Uma maneira de suavizar esse problema é desenvolver um *wrapper* ao redor do COTS, desabilitando-o de exibir funcionalidades não desejadas. Uma barreira de software é colocada ao redor do produto, limitando o que ele pode fazer. Geralmente uma forma de se implementar tal envoltório é testando as entradas antes de fazer uma chamada ao sistema e somente permitindo tal chamada se a entrada provavelmente produzir um comportamento aceitável [MAC98]. Uma visão geral dessa técnica é mostrada na Figura 2.4. Depois que qualquer tipo de adaptação tenha sido realizada para evitar futuros conflitos entre os COTS e o sistema, a fase seguinte do ciclo de vida COTS é a integração.

Figura 2.4 - Técnica *Wrapper*

2.4.4 Integração

A fase de integração inclui todo o esforço necessário para interconectar os diferentes produtos anteriormente selecionados num único sistema integrado. Normalmente, essa fase inclui o desenvolvimento de outras partes do sistema que não foram suportadas por produtos comercialmente disponíveis. Dessa forma, é conveniente utilizar também um modelo de desenvolvimento tradicional como o Cascata ou Espiral [TRA97a].

Um aspecto do desenvolvimento baseado em COTS que deve ser considerado com cuidado é o fato de que muitos dos problemas encontrados durante o processo de integração não podem ser determinados antes que essa fase comece. Assim, realizar uma estimativa consistente de prazos e recursos necessários pode ser uma tarefa extremamente difícil [VIG96]. Boa parte dos sistemas que utilizam COTS são grandes, complexos e possuem um longo ciclo de vida. Dessa forma, ao longo dos anos tais sistemas poderão sofrer inúmeras atualizações de seus produtos, assim como ter a inclusão de vários outros. Esse fato é devido principalmente à constante mudança que ocorre no mercado de software e nas organizações atuais. A fase de atualização dos componentes é a última atividade do processo de desenvolvimento baseado em COTS e será vista a seguir.

2.4.5 Atualização

A maioria dos sistemas, incluindo aqueles baseados em COTS, precisa de sucessivas atualizações. Por exemplo, para reparar um erro, pode ser necessário adquirir um novo produto para substituir aquele com defeito. Similarmente, quando é requerida uma nova funcionalidade, ela pode estar incluída num novo COTS que é adicionado ao sistema. Sem um planejamento cuidadoso, a substituição de um COTS pode ter extensa repercussão em muitos outros produtos [BRO97]. Por exemplo, quase sempre quando novas versões de um produto são adquiridas pode ser necessária também a substituição de versões de outros COTS, ou em alguns casos esses produtos recém adquiridos podem até ser incompatíveis com aqueles que já compõem o sistema [HAI97]. Dependendo do número de produtos e diferentes vendedores, o efeito dessas múltiplas dependências pode variar de pequenas inconveniências até a completa instabilidade do sistema [CAR97].

A atualização de sistemas baseados em COTS está diretamente ligada com a aquisição de novos lançamentos e versões dos produtos. Assim, um sistema que possui inúmeros COTS está fortemente dependente das futuras modificações e atualizações realizadas pelos vendedores desses produtos. Normalmente, é necessário revalidar licenças e contratos com os vendedores [CAR97]. A próxima seção trata alguns dos principais desafios relacionados com o desenvolvimento baseado em COTS, no qual a dependência de vendedores COTS é considerada um dos mais importantes.

2.5 DESAFIOS NO PROCESSO DE DESENVOLVIMENTO BASEADO EM COTS

Quando uma organização utiliza COTS, quase sempre ela perde uma certa autonomia e adquire algumas dependências adicionais. Os vendedores de COTS freqüentemente atualizam seus produtos baseados em relatórios de erros, percepção de novas necessidades do mercado e concorrência de outros produtos similares [KUN99]. Uma organização precisa julgar se é vantajoso incluir novas atualizações dos COTS que já estão integrados ao sistema. Se a decisão for mudar esses produtos por novas versões, isso implica num

esforço adicional de adaptação e realização de mais testes para verificar a confiabilidade do sistema. Se a organização decidir não incluir novas atualizações dos COTS, tais produtos provavelmente ficarão ultrapassados e podem não ser adequadamente suportados pelos fornecedores. O desenvolvimento de sistemas baseado em COTS é profundamente afetado por fatores externos que devem ser levados em consideração, por exemplo:

- Como determinar a qualidade do suporte a longo prazo que o vendedor irá prover?
- Quais as consequências que o sistema sofrerá se o vendedor de um COTS crítico sair do mercado?
- Como avaliar os custos associados à aquisição de determinado COTS?
- Quais atributos de um produto são críticos para um domínio particular?
- Como verificar que o produto está em conformidade com outros padrões?
- Como descrever e quantificar os aspectos de qualidade de um COTS?

Essas questões são extremamente relevantes e estão sendo estudadas pela comunidade de DBC há alguns anos [CAR97]. O impacto de alguns dos problemas mencionados acima pode ser minimizado se for dado mais enfoque às fases de avaliação e seleção [NCU00].

2.6 CONSIDERAÇÕES FINAIS

As principais motivações para desenvolver sistemas baseados em COTS são as possíveis oportunidades de desenvolver sistemas num curto intervalo de tempo e com custos reduzidos. Atualmente, a abordagem COTS está amplamente difundida na indústria de software e inúmeras organizações têm constatado as vantagens dessa forma de desenvolvimento. Vale lembrar que ainda não existe um consenso na comunidade de CBSE sobre a definição do termo COTS. Esse trabalho considera que COTS são componentes que possuem as seguintes características básicas: estão disponíveis no mercado, podem ser comprados e geralmente não permitem acesso ao código-fonte.

Existem várias diferenças entre o desenvolvimento convencional e o baseado em COTS. Nesse capítulo, foram analisados os aspectos referentes à arquitetura, componentes e metodologia desses dois tipos de desenvolvimento de software. De um modo geral, o ciclo de vida do desenvolvimento baseado em COTS consiste das fases de: avaliação do mercado de COTS, seleção, adaptação, integração e atualização.

As fases de avaliação e seleção são consideradas entre as mais críticas uma vez que é durante essa etapa que precisam ser analisados aspectos relacionados à funcionalidade e características gerais dos produtos, custos de aquisição, garantias do vendedor, etc. Além disso, realizar um processo de seleção bem estruturado e consistente pode evitar futuros problemas durante as fases seguintes do processo de desenvolvimento baseado em COTS. A fim de esclarecer melhor tais aspectos, o próximo capítulo trata detalhadamente o processo de seleção de produtos COTS.

Capítulo 3

O Processo de Seleção de COTS

Este capítulo aborda o processo de seleção de COTS. Inicialmente, será mostrada uma visão geral da área, em seguida os principais desafios desse processo serão examinados. A seção seguinte descreve os trabalhos mais relevantes na área de seleção de COTS. Logo depois, serão examinadas algumas técnicas para tomada de decisão.

3.1 VISÃO GERAL

As fases de avaliação e seleção de COTS estão intimamente relacionadas. Intuitivamente, o processo de avaliação deve anteceder o de seleção. Todavia, é errado considerar que a avaliação é um evento único para cada produto selecionado. Ao contrário, existem atividades de avaliação que precedem o processo de seleção e outras que ocorrem depois que o produto já foi selecionado [FOR99], onde algumas dessas atividades podem ser realizadas em paralelo [CAR98]. A Figura 3.1 mostra os esforços do processo de avaliação ao longo do ciclo de vida de sistemas baseados em COTS. Entretanto, de acordo com o contexto e objetivos dessa dissertação, a atividade de avaliação será examinada somente durante o processo de seleção.

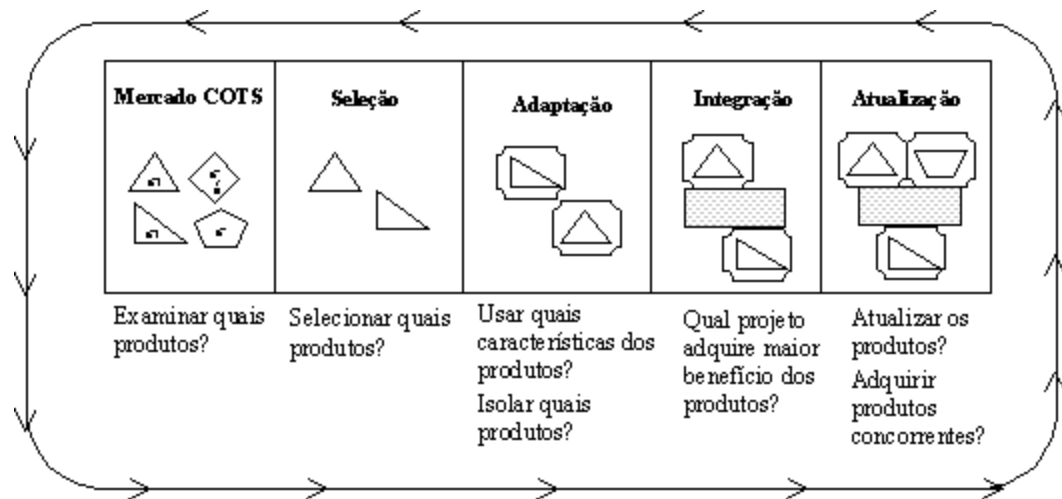


Figura 3.1 - Atividades de avaliação durante o ciclo de vida COTS

3.2 DESAFIOS E PROBLEMAS ENCONTRADOS NO PROCESSO DE SELEÇÃO DE COTS

A seleção de componentes COTS pode ser considerada o processo para avaliar se determinado produto previamente desenvolvido é apropriado para o uso no contexto de um novo sistema [HAI97]. Inúmeras pesquisas nessa área têm enfatizado que um processo de

seleção bem definido e sistemático é fundamental para o sucesso de qualquer sistema baseado em COTS [KUN99] [NCU99] [FOX97] [KON96a]. Além disso, as decisões realizadas nesse momento proporcionarão impacto considerável nas fases seguintes. Entretanto, o processo de seleção está propenso a diversos riscos e problemas. Existem alguns fatores que dificultam a realização de um processo de seleção bem sucedido, entre eles:

- **Falta de um processo bem definido** – Normalmente, as empresas sofrem pressões para realizar o processo de seleção num curto intervalo de tempo e quase sempre esse processo é feito de forma ad-hoc [KON96a], onde não são usados métodos e ferramentas apropriados e isso dificulta o planejamento e o aprendizado a partir de lições anteriores;
- **Critério de seleção** – Existem problemas com a definição do critério de seleção, às vezes são incluídos atributos desnecessários e não apropriados para descrever os produtos candidatos;
- **Natureza caixa-preta dos COTS** – A falta de acesso ao código interno dos COTS dificulta o seu entendimento e dessa forma torna a avaliação complicada, muitas vezes até mesmo a documentação desses componentes está incompleta ou errada;
- **Rápida mudança no mercado** – A avaliação de COTS é difícil porque o mercado de software muda rapidamente. Por exemplo: uma nova versão de um produto que está sendo avaliado pode ter algumas características diferentes e que podem gerar conflito com o resto do sistema.

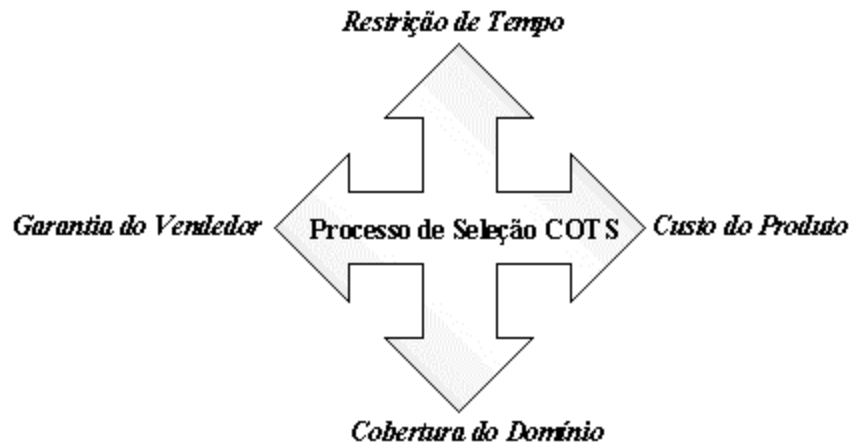


Figura 3.2 - Dimensões do Processo de Seleção de COTS

Normalmente, a seleção de produtos não é considerada uma tarefa trivial, ela requer consideração de múltiplos fatores e análise cuidadosa entre requisitos dos usuários, características técnicas e fatores financeiros [KON96a]. Foram identificadas quatro dimensões (Figura 3.2) que precisam ser avaliadas em detalhes durante a fase de seleção, são elas [ALV00]:

- **Cobertura do domínio** – Os componentes precisam prover totalmente ou parte das capacidades necessárias para atender os requisitos críticos dos *stakeholders*, em alguns casos, componentes extras precisam ser desenvolvidos ou adquiridos para atender tais exigências;
- **Restrição de tempo** – As companhias normalmente operam com prazos curtos, onde sua competitividade e sucesso no mercado dependem de entregas dentro do prazo preestabelecido. O processo de seleção é uma atividade que consome uma considerável quantidade de tempo e esforços;
- **Custo associado** – Essa é uma variável extremamente importante, os gastos para desenvolver um sistema baseado em COTS vão além do custo de aquisição do produto, estão entre os gastos associados mais comuns: treinamento, suporte, adaptação e

manutenção. Aoyama [AOY98] descreve um modelo econômico para estimar o custo do desenvolvimento de um sistema baseado em COTS;

- **Garantias do vendedor** – Um importante aspecto para considerar durante a atividade de seleção é verificar o suporte técnico que o vendedor provê, algumas questões precisam ser analisadas, por exemplo: reputação e maturidade do vendedor, número e tipo de aplicações que já utilizam o produto, garantia de suporte e manutenção.

Esse trabalho considera detalhadamente cada uma dessas quatro dimensões associadas com o processo de seleção. Primeiramente, serão descritos alguns dos trabalhos mais significativos da área e em seguida será feita uma análise dos benefícios e restrições de cada método.

3.3 TRABALHOS RELACIONADOS

Existem basicamente três estratégias para seleção de COTS: filtragem progressiva, identificação por característica chave e montagem de quebra-cabeça [OBE97]. *Filtragem progressiva* representa uma estratégia onde um produto é selecionado a partir de um grupo de potenciais candidatos, esse processo começa com uma grande variedade de candidatos e através de mecanismos de filtragem, os produtos considerados inadequados são eliminados progressivamente. Na *identificação por característica chave*, uma característica particular é selecionada antes de iniciar o processo de seleção, essa característica pode ser um determinado vendedor ou uma tecnologia, por exemplo: o critério de seleção inicial pode ser procurar somente por produtos Microsoft ou que utilizam a tecnologia CORBA. Finalmente, o modelo de *montagem de quebra-cabeça* inicia com a premissa de que uma solução COTS válida precisará “encaixar” os vários componentes do sistema como um quebra-cabeça. Essa abordagem aplica técnicas de prototipagem evolucionária para construir inúmeras versões que são progressivamente próximas do sistema final. Existem vários trabalhos que propõem métodos para auxiliar o processo de seleção. A seguir, serão descritos alguns desses métodos.

3.3.1 Método OTSO

Uma das primeiras iniciativas para assistir e facilitar a seleção de COTS é o Método OTSO (*Off-The-Shelf Option*) [KON95]. Esse método suporta a localização, avaliação e seleção de componentes reusáveis, assim como fornece técnicas específicas para definir o critério de avaliação, comparando custos e benefícios das alternativas. Esse método decompõe o critério de avaliação em quatro áreas principais [KON96b]:

- Requisitos dos usuários;
- Características de qualidade do produto;
- Compatibilidade de domínio e arquitetura;
- Interesses estratégicos (custos, estabilidade do vendedor, etc.).

Uma das fases mais importantes do método é a *análise dos resultados*, essa fase faz uma estimativa dos custos para adquirir cada candidato COTS e compara custos e benefícios de cada alternativa. Além disso, usa a técnica AHP (*Analytic Hierarchy Process*) [SAA90] para consolidar os resultados da avaliação e suportar o processo de tomada de decisão. Apesar do método OTSO enfatizar que um dos problemas chave do processo de seleção de COTS é a falta de atenção aos requisitos dos *stakeholders*, esse método não provê nenhuma sugestão ou solução para atacar esse problema. O principal foco do método é definir o critério de avaliação, porém ele não oferece orientação de como adquirir e modelar os requisitos. O método assume que os requisitos já existem pois ele parte da especificação dos requisitos para em seguida utilizá-los durante o processo de seleção.

3.3.2 Método PORE

O Método PORE (*Procurement-Oriented Requirements Engineering*) [NCU00a] é baseado num processo iterativo de aquisição de requisitos e seleção de produtos, essa é uma abordagem recente e bastante aceita [FOX97] [KUN00]. O modelo de ciclo de vida PORE consiste de seis processos genéricos. O modelo de processo descreve os processos

fundamentais que precisam ser realizados durante a aquisição de produtos COTS, são eles [NCU99]:

- Processo de gerenciamento de aquisição do produto – onde os objetivos são planejar e controlar o processo de aquisição;
- Processo de aquisição de requisitos – esse processo adquire e valida os requisitos dos usuários, ele também determina a arquitetura que os produtos deverão seguir;
- Processo de seleção de fornecedor – o objetivo é estabelecer um critério para selecionar o fornecedor e fazer um ranking de acordo com o critério das melhores alternativas;
- Seleção de pacote de software – o objetivo é identificar os pacotes candidatos, estabelecer o critério de seleção usando os requisitos dos usuários e fazer um ranking para selecionar o produto que melhor se ajusta aos requisitos chave dos usuários;
- Produção do contrato - onde são negociados os contratos legais com o fornecedor, além disso são negociados os direitos e garantias;
- Aceitação do pacote – o objetivo é validar o pacote em relação aos requisitos chave dos usuários.

Uma das principais fraquezas desse método é a falta de uma análise mais detalhada de como deve ser realizado o processo de requisitos. Em particular, ele não aborda como elicitar e modelar os requisitos não-funcionais durante o processo de seleção de COTS. O Método PORE traz uma abordagem nova, porém não descreve como o critério de seleção deve ser obtido e utilizado, como também esquece de considerar aspectos não técnicos durante o processo seletivo, tais como: custo e treinamento. Outras limitações do método são a sua complexidade de uso e tempo necessário para realizar todas as fases propostas.

3.3.3 *Framework STACE*

O Framework STACE (*Social-Technical Approach to COTS Evaluation*) [KUN99] tem como objetivo descrever uma abordagem sócio-técnica para selecionar produtos. STACE

ênfatiza a importância da participação dos usuários no processo de seleção pois essa é a principal forma de trazer a dimensão social ao desenvolvimento de sistemas. Esse método usa a abordagem *identificação por característica chave* e dessa forma sugere selecionar a tecnologia que os produtos candidatos devem se basear antes de iniciar o processo seletivo. O Framework STACE é fortemente influenciado pelos trabalhos de Kontio (método OTSO) e Maiden (método PORE), ele possui alguns princípios fundamentais:

- Suporta uma abordagem sistemática de avaliação e seleção de COTS, permitindo a reutilização de conhecimento e casos futuros;
- Suporta a avaliação tanto de produtos COTS como de tecnologias subjacentes;
- Usa abordagem sócio-técnica para melhorar o processo de seleção;
- Suporta avaliação de fatores não técnicos.

STACE recomenda que o uso de um critério de seleção deve ser baseado em fatores sócio-técnicos, onde são analisadas características de funcionalidade, qualidade do produto segundo normas ISO e fatores econômicos. Durante o processo de avaliação dos produtos, a participação dos usuários é usada para aumentar a aceitação do produto, como também para resolver problemas de conflito de interesses. Uma das limitações desse método é a ausência de uma avaliação dos requisitos não-funcionais durante o processo de seleção de COTS. O método STACE se baseia em normas ISO, essas normas definem os requisitos de qualidade como um conjunto de atributos do produto porém não analisam o impacto que tais atributos proporcionarão em cada produto a ser selecionado.

3.3.4 Método IusWare

O método IusWare [MOR97] é baseado na abordagem auxílio de decisão de critérios múltiplos (ADCM). O método define um processo de avaliação que consiste de duas fases principais, a primeira é o projeto do modelo de avaliação e a segunda fase é chamada de fase de aplicação. A fase de projeto inclui as seguintes atividades:

- Identificar os atores relevantes no processo de avaliação, descrevendo seus papéis e objetivos, assim como os recursos disponíveis para esse processo;
- Identificar o tipo de avaliação necessária, onde pode ser feita uma descrição detalhada dos produtos ou *ranking* de produtos, indicando a preferência de cada um;
- Definir uma hierarquia não redundante dos atributos usados durante a avaliação;
- Associar uma medida, uma escala de critério e uma função para transformar a escala de medida numa escala de critério para cada atributo;
- Escolher uma técnica de agregação para agregar valores no critério de seleção.

Na fase de aplicação, os atributos dos produtos são medidos, em seguida essas medidas são transformadas em valores e agregadas de acordo com a recomendação dos atores. Apesar do método IusWare ser uma metodologia para avaliar e selecionar produtos de software, ele não trata a questão dos requisitos nesse processo. O método apresenta um modelo de avaliação que envolve a definição de uma hierarquia dos atributos juntamente com um modelo de tomada de decisão que é baseado na abordagem ADCM. A principal fraqueza desse método é que ele assume que os requisitos já estão disponíveis.

3.3.5 *Método Análise de Características*

As principais características do método de análise de características [KIT96] são:

- Ajuda a clarificar as características importantes do produto no contexto do ambiente que ele será usado;
- Ajuda a identificar as diferenças entre os produtos;
- Provê explicações do “porquê” foi tomada determinada decisão para selecionar um produto em detrimento das outras alternativas.
- O método é realizado de forma iterativa, onde a cada iteração diminui o número de produtos candidatos [KIT97]. Durante esse processo o conjunto de características sendo avaliadas pode variar assim como o próprio critério de avaliação. Esse método fornece

um *framework* comparativo para auxiliar as avaliações, esse *framework* é expresso em termos de um conjunto de propriedades, qualidades, atributos e características de cada produto, onde é associada uma prioridade (crítico, relevante, insignificante) a cada elemento. Os produtos são julgados a partir da análise do grau de concordância dos mesmos com os requisitos. Para isso, o método provê uma escala na qual é atribuído um escore a cada produto em relação a determinado atributo. O método análise de características é baseado na identificação dos requisitos dos usuários em relação a uma atividade ou tarefa em particular, em seguida esses requisitos são mapeados em características que o produto precisa suportar. Uma das restrições do método é que ele não oferece qualquer solução para tratar os problemas que surgem durante a aquisição dos requisitos.

3.3.6 Aspectos Relevantes

A partir da análise dos métodos descritos acima, pode-se observar que a maioria deles não trata de forma adequada os problemas relacionados com o processo de requisitos. Dessa forma, pode-se verificar a partir da Tabela 3.1 que a hipótese H1 é coerente.

H 1 Os métodos para seleção de COTS existentes na literatura não abordam de forma satisfatória os requisitos dos *stakeholders*.

Alguns métodos reconhecem a importância dos requisitos, porém a maioria deles não oferece soluções apropriadas. A única exceção entre os métodos apresentados é o método PORE [NCU00a]. Ele oferece uma metodologia detalhada, incluindo diversos métodos e técnicas para elicitar os requisitos dos usuários e em seguida utilizá-los para guiar o processo de seleção de COTS. Entretanto, a grande restrição desse método é que ele não oferece um tratamento efetivo para elicitar e modelar os requisitos não-funcionais.

	Identificação do produto	Aquisição dos requisitos	Descrição dos requisitos não-funcionais	Avaliação/ Seleção	Análise de decisão
OTSO	√	–	–	√	√
PORE	√	√	*	√	√
STACE	√	*	*	√	*
IusWare	–	–	–	*	√
Análise de característica	–	*	–	√	*

√ trata de forma satisfatória – não trata * trata parcialmente

Tabela 3.1 - Resumo das etapas cobertas pelos métodos descritos

A Tabela 3.1 mostra as principais etapas do processo de seleção que são cobertas pelos métodos descritos acima. A tabela mostra que praticamente todos métodos apresentam fraquezas no tratamento dos requisitos, enquanto os processos de avaliação/seleção são abordados por todos os métodos apresentados. Finalmente, nenhum dos métodos trata de forma adequada os requisitos não-funcionais. Nesse contexto, um dos principais objetivos desse trabalho é atacar essa restrição existente nos métodos descritos.

3.4 TÉCNICAS PARA TOMADA DE DECISÃO

Uma das etapas críticas da maioria dos processos de seleção de COTS é o processo de tomada de decisão. Com o objetivo de selecionar ou recomendar um produto COTS adequado para integrar o sistema, as alternativas disponíveis precisam ser avaliadas de acordo com sua relativa importância para atender os requisitos dos usuários. As técnicas de tomada de decisão são usadas para analisar essas alternativas e fazer um ranking das melhores opções e assim contribuir para obtenção de um processo de seleção consistente.

A Figura 3.3 mostra os princípios do processo de tomada de decisão para avaliar produtos. Esse processo é representado numa hierarquia de três níveis e foi adaptado de Saaty [SAA90]. No primeiro nível é apresentada a principal meta para o processo de tomada de decisão. No segundo nível são mostrados alguns critérios relevantes para selecionar o produto, estão entre eles: funcionalidades, garantias do vendedor, requisitos de qualidade como usabilidade, etc. Finalmente, no terceiro nível estão os produtos candidatos que serão avaliados a partir dos critérios definidos no nível 2 para atingir a meta principal que é *Selecionar produto adequado*. Uma vez que as alternativas disponíveis são avaliadas, uma abordagem comum é necessária para consolidar os resultados da avaliação e fazer o ranking das várias alternativas. As técnicas descritas a seguir têm como objetivo auxiliar essas tarefas.

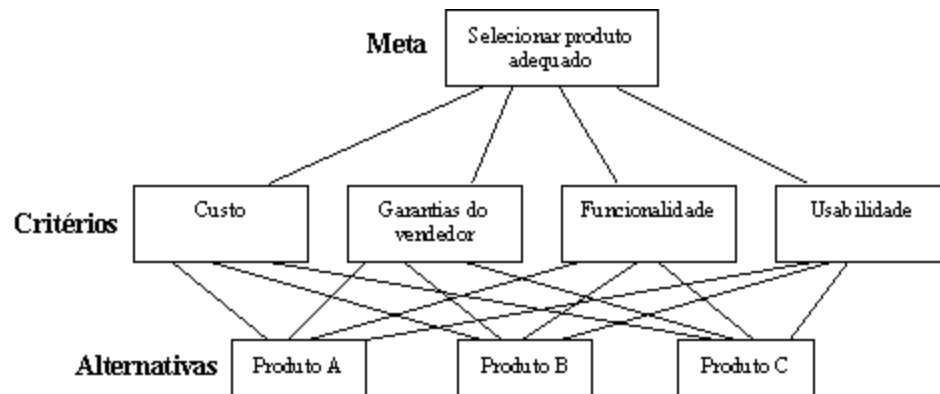


Figura 3.3 - Princípios do processo de tomada de decisão

3.4.1 Técnica MAUT

A técnica MAUT (*Multi-Attribute Utility Theory*) auxilia a escolha da melhor alternativa dentro de um conjunto de opções que são descritas em termos de seus atributos [CLE97]. Essa técnica precisa de algumas informações sobre as preferências dos usuários a fim de negociar situações com múltiplos atributos, por exemplo:

- A preferência entre valores de um dado atributo (preferência entre uma base de dados comercial ao invés de uma proprietária);

- A preferência entre atributos (qual a importância da interoperabilidade da base de dados sobre o seu custo).

Uma das vantagens da técnica MAUT é que ela força os usuários a considerarem todos os atributos referentes a uma decisão. Muitas vezes, durante o processo de tomada de decisão pode ocorrer que certos atributos importantes sejam esquecidos. Essa técnica auxilia os tomadores de decisão para criar uma lista detalhada de todos os atributos relevantes [YOU98]. A principal fraqueza da técnica é que ela representa os atributos de forma independente e assim não trata a relação de dependência que ocorre entre muitos atributos.

3.4.2 Técnica WSM

A técnica WSM (*Weighted Score Method*) é uma das abordagens mais usadas em situações de tomada de decisão [CLE97]. Essa técnica faz um ranking das alternativas em função dos critérios preestabelecidos, onde é definida uma escala que representa a preferência relativa de cada atributo. Essa técnica funciona da seguinte maneira:

- i. Vários critérios são descritos para avaliar cada alternativa (por exemplo: facilidade de uso, custo, segurança, performance, etc.);
- ii. Cada critério tem um peso associado que está diretamente relacionado com a sua importância;
- iii. As alternativas devem ser avaliadas segundo a observância de cada critério e depois é atribuído um escore numa escala pré-definida;
- iv. Finalmente, cada alternativa receberá uma pontuação que é calculada a partir da seguinte fórmula [KON96b]:

$$Pont_Produto_a = \sum_{j=1}^n (peso * escore_{aj})$$

onde: a representa uma alternativa e n o numero de critérios

O uso da técnica WSM é simples e apresenta resultados que pelo menos intuitivamente fazem sentido. Entretanto, existem diversos problemas com essa técnica que tornam

questionável se WSM consegue representar a real preferência entre alternativas [NCU00a]. Como o método WSM produz números reais como resultado, tais resultados podem ser facilmente interpretados como se eles representassem a real diferença entre duas alternativas. De fato, os escores resultantes apenas representam o *ranking* relativo entre as alternativas e as diferenças entre os valores não dá nenhuma indicação da sua relativa superioridade [KON96a]. Outra restrição de WSM é que a atribuição dos pesos para cada critério pode ser bastante complicada quando o número de critérios é grande.

3.4.3 Técnica AHP

A técnica AHP (*Analytical Hierarchy Process*) [SAA90] é um método baseado na decomposição hierárquica de um problema de tomada de decisão com múltiplos critérios. Essa técnica tem sido usada com sucesso em diversas áreas, incluindo engenharia de software [FIN95] e seleção de software [HON81] [NCU00] [MIN92] [KON96b] [KUN99]. O método AHP é suportado por uma ferramenta comercial chamada *Expert Choice* [EXP00] que auxilia nos cálculos e julgamentos necessários. AHP decompõe um problema numa hierarquia de critérios, onde em cada nível a importância relativa de cada fator é avaliada através de comparações aos pares dos critérios. Finalmente, as alternativas são comparadas aos pares em relação a cada critério. Os principais passos para aplicar a técnica AHP são [SAA90]:

- i. Definir uma hierarquia de fatores que influenciam a decisão, isso resulta numa estrutura hierárquica de fatores que tem as alternativas como nós finais dessa hierarquia, ver Figura 3.3;
- ii. Definir a importância dos fatores em cada nível;
- iii. Definir as preferências entre as alternativas;
- iv. Verificar a consistência do ranking e revisar os resultados;

- v. Apresentar o resultado da avaliação, onde a alternativa com maior prioridade será recomendada como a melhor escolha.

O *ranking* obtido através da comparação aos pares entre as alternativas é convertido para um *ranking* normalizado usando o método de autovetores, ou seja, o ranking relativo entre as alternativas é representado em valores cujo total é um. Segundo Saaty [SAA90], o tratamento de um problema com múltiplos atributos através de hierarquias é a forma natural como os seres humanos observam e representam o mundo. Usando essa técnica, as alternativas e critérios são comparados aos pares, o que resulta em resultados mais confiáveis [KON96b]. Dessa forma, o problema de ter que comparar valores absolutos é evitado uma vez que somente sua preferência relativa é comparada.

Um dos principais problemas da técnica AHP é a complexidade dos cálculos necessários para fazer as comparações aos pares e realizar as operações entre as matrizes de valores. Entretanto, com o uso da ferramenta *Expert Choice* [EXP00] esse problema pode ser minimizado. Uma importante limitação de AHP quando aplicada na avaliação de software é o fato dessa abordagem assumir independência entre os atributos dos produtos, ou seja, com o objetivo de fazer as comparações, essa técnica assume que os atributos são independentes [NCU00].

3.4.4 Principais Limitações das técnicas para tomada de decisão

Um problema inerente a qualquer processo de tomada de decisão é a dificuldade para atribuir pesos e escores aos atributos que estão sendo examinados. Uma possível solução é formar um grupo de avaliadores que possua experiência em tarefas dessa natureza e já tenha utilizado as técnicas propostas. Outro aspecto importante é que o processo de tomada de decisão seja realizado pelas mesmas pessoas, assim os riscos de inconsistência entre os resultados podem ser reduzidos.

As principais técnicas para tomada de decisão possuem algumas limitações quando utilizadas para selecionar produtos COTS. Além da complexidade de uso, outro aspecto relevante é a falta de uma abordagem que trate o inter-relacionamento entre os atributos dos

produtos. A maioria dos requisitos de software interage entre si, de forma que a avaliação de um produto em relação à satisfação de um requisito pode sofrer influência de outros requisitos que pelo menos aparentemente, não possuam nenhuma ligação.

Dessa forma, torna-se necessário desenvolver novas abordagens para tratar o inter-relacionamento entre os atributos durante o processo de tomada de decisão. Nesse contexto, é importante que tais abordagens sejam baseadas em requisitos pois essa estratégia propicia um melhor entendimento da importância de cada atributo, facilitando assim a atribuição consistente dos pesos. Outra vantagem de uma abordagem baseada em requisitos é que ela fornece informações importantes para auxiliar suposições e julgamentos durante o processo de tomada de decisão.

3.5 CONSIDERAÇÕES FINAIS

O processo de seleção é uma das fases mais importantes do processo de desenvolvimento baseado em COTS. Durante essa fase são escolhidos um ou mais produtos adequados para integrar o sistema final. Assim, a seleção apropriada é fundamental para o sucesso desses sistemas senão podem ser ocasionados inúmeros problemas nas fases seguintes como uma possível incompatibilidade com o restante do sistema.

Existem alguns fatores que dificultam o processo de seleção, entre os mais importantes estão: a natureza caixa-preta dos produtos e a rápida mudança do mercado de software. Além desses fatores, a própria natureza multidimensional do processo de seleção torna-o uma atividade complexa e demorada. Tais aspectos mostram que a seleção de COTS precisa de métodos bem definidos para assistir esse processo.

Existem diversos métodos presentes na literatura para auxiliar o processo de seleção de COTS, nesse capítulo foram descritos alguns dos mais importantes. A principal fraqueza da maioria desses métodos está relacionada com a falta de um processo de requisitos. Somente o método PORE [NCU00] aborda um pouco mais a fundo os requisitos dos *stakeholders*. Entretanto, esse método não fornece nenhuma atenção diferenciada aos requisitos não-

funcionais que por natureza são mais difíceis de especificar e que, entretanto possuem um papel fundamental durante a avaliação das capacidades dos produtos COTS.

Um aspecto importante durante as fases de avaliação e seleção de COTS é o processo de tomada de decisão. As técnicas mais relevantes para auxiliar o processo de seleção de software foram descritas nesse capítulo. O principal objetivo delas é prover informações para julgar e avaliar a adequabilidade de cada alternativa, tomando como critério vários atributos pré-definidos. Uma importante restrição das técnicas para tomada de decisão quando aplicadas para selecionar software é a falta de uma abordagem que trate o inter-relacionamento entre os atributos dos produtos, no qual esse aspecto está diretamente relacionado com a interação entre os requisitos dos usuários. Conseqüentemente, são necessárias abordagens baseadas em requisitos para auxiliar o processo de tomada de decisão.

Diante de todas essas observações, essa dissertação está fortemente fundamentada na importância de uma abordagem baseada em requisitos para assistir todas as etapas do processo de seleção de produtos COTS. Nesse contexto, o próximo capítulo mostra o processo de engenharia de requisitos tradicional e como ele pode ser adaptado para auxiliar o desenvolvimento de sistemas baseados em COTS.

Capítulo 4

Engenharia de Requisitos para Desenvolvimento Baseado em COTS

Este capítulo fornece uma visão geral das fases do processo de engenharia de requisitos. Em seguida será mostrada uma proposta do processo de engenharia de requisitos para desenvolvimento baseado em COTS. São examinados os processo de aquisição e especificação de requisitos e em particular é descrito um framework para descrição dos requisitos não-funcionais. Além disso, é examinado como a engenharia de requisitos pode auxiliar o processo de seleção de COTS.

4.1 VISÃO GERAL DA ENGENHARIA DE REQUISITOS

Uma das primeiras medidas do sucesso de um sistema de software é verificar se ele atende às necessidades dos *stakeholders* [NUS00]. Num dos primeiros trabalhos realizados na área, Bell e Thayer [BEL76] observaram que muitos requisitos são inadequados, inconsistentes, incompletos e ambíguos e que eles tem um grande impacto na qualidade do software final. A partir dessa observação eles concluíram que “requisitos para um dado sistema não podem ser levantados naturalmente, ao contrário eles precisam ser projetados e necessitam de contínuas revisões”.

Estudos [BOE84] indicam que, quando só detectados depois do software implementado, erros em requisitos de software são até 20 vezes mais caros de se corrigir que qualquer outro tipo de erro. Estudos recentes comprovam que os problemas relacionados aos requisitos do sistema afetam boa parte das organizações que desenvolvem e usam sistemas de software [STA95] [ESI96]. Portanto, pode-se considerar que a engenharia de requisitos é uma das fases mais importantes do processo de engenharia de software [LAM00]. A fim de melhorar a qualidade dos requisitos, inicialmente é necessário definir o que requisitos são realmente. Existem inúmeras definições disponíveis na literatura para o termo requisitos. Segundo Kotonya e Sommerville [KOT97], um requisito pode descrever:

- Uma facilidade no nível do usuário; por exemplo, um corretor de gramática e ortografia;
- Uma propriedade muito geral do sistema; por exemplo, o sigilo de informações sem a devida autorização;
- Uma restrição específica no sistema; por exemplo, o tempo de varredura de um sensor;
- Uma restrição no desenvolvimento do sistema; por exemplo: a linguagem que deverá ser utilizada para o desenvolvimento do sistema.

Uma definição bem simples para requisitos é dada por Macaulay [MAC96], segundo ele requisito é simplesmente algo que o cliente necessita. Segundo Jackson [JAC95], requisitos são fenômenos ou propriedades do domínio da aplicação que devem ser executados, normalmente expressos em linguagem natural, diagrama informal ou outra notação apropriada ao entendimento do cliente e da equipe de desenvolvimento.

Outra definição necessária é a de engenharia de requisitos. Uma definição genérica para o termo é uma atividade que objetiva estabelecer o que o cliente requer de um sistema de software. Segundo o IEEE [STD830], engenharia de requisitos corresponde ao processo de aquisição, refinamento e verificação das necessidades do cliente para um sistema de software, objetivando-se ter uma especificação completa e correta dos requisitos de software.

Atualmente, uma das definições para engenharia de requisitos mais aceita é dada por Zave [ZAV97], segundo ela a engenharia de requisitos está relacionada com a identificação de metas para serem atingidas pelo sistema a ser desenvolvido, assim como a operacionalização de tais metas em serviços e restrições. Essa área também está interessada com o relacionamento desses fatores para fazer uma especificação do comportamento do software e sua evolução ao longo do tempo. A engenharia de requisitos é uma área ampla e multidisciplinar, onde aspectos sociais e humanos desempenham um importante papel no processo de engenharia de requisitos [ZAV97] [NUS00]. Em seguida, esse assunto será abordado em detalhes.

4.2 O PROCESSO DE ENGENHARIA DE REQUISITOS

O processo de engenharia de requisitos pode ser considerado um conjunto estruturado de atividades que são seguidas com o objetivo de derivar, validar e manter um documento de requisito [KOT97].

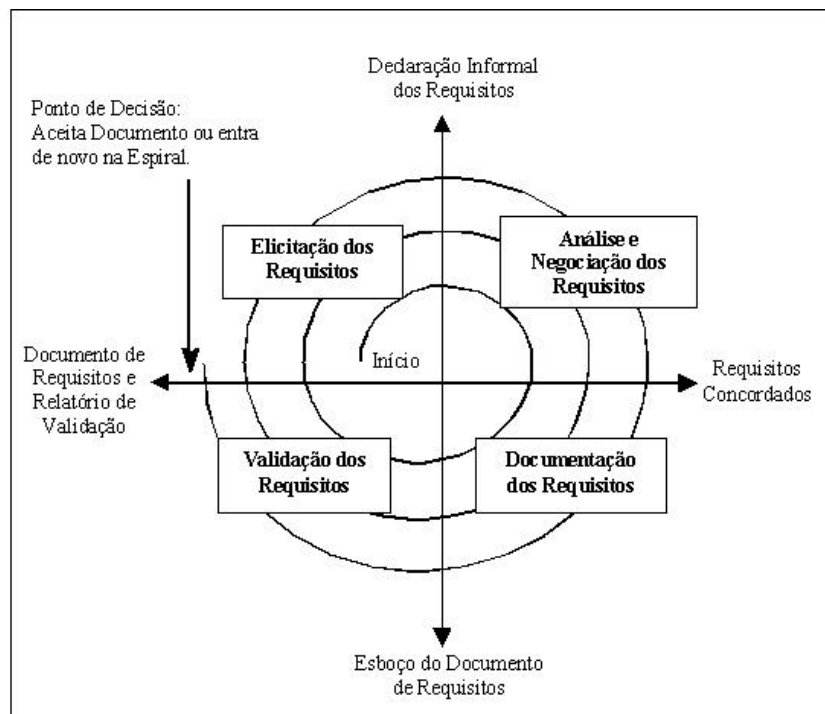


Figura 4.1 - Atividades do Processo da Engenharia de Requisitos no Modelo Espiral

Existem várias propostas para modelos de processo de engenharia de requisitos. Todavia, não existe um processo considerado ideal. Este trabalho adota o modelo proposto por Kotonya e Sommerville [KOT97], que descreve as seguintes fases do processo de requisitos: Elicitação, Análise, Documentação e Validação de Requisitos. A Figura 4.1 mostra essas atividades representadas num modelo espiral, no qual cada atividade do processo é repetida até que seja tomada a decisão de que o documento de requisitos pode ser aceito. Apesar dessas atividades serem normalmente descritas independentemente e numa ordem particular, na prática, elas consistem de processos iterativos e inter-relacionados que podem cobrir todo o ciclo de vida do desenvolvimento de sistemas de software [NUS00].

4.2.1 Elicitação de Requisitos

Esta é a primeira atividade no ciclo de vida da engenharia de requisito, o seu propósito geral é obter conhecimento relevante para o problema a ser resolvido. Elicitação de

requisitos é um processo de descoberta dos requisitos para um sistema através da comunicação com clientes, usuários finais e quem tiver interesse no desenvolvimento do sistema. Além de descobrir quais são as necessidades dos usuários, essa atividade também requer uma cuidadosa análise da organização, do domínio de aplicação e dos processos organizacionais [KOT97]. Assim, cabe a elicitação a tarefa de identificar os fatos que compõem os requisitos do sistema, de forma a prover o mais correto entendimento do que é esperado do sistema software. Outro papel do processo de elicitação é identificar as necessidades das diferentes classes de usuários, por exemplo: experientes, ocasionais, iniciantes, etc [SHA99].

Um dos principais problemas encontrados durante essa fase é a dificuldade de entender as reais necessidades dos usuários [CAS95][NUS00]. Em boa parte dos casos, isso é devido à formação distinta entre analistas e usuários, e portanto eles têm pontos de vista diferentes sobre o mesmo problema. Outro fator que dificulta o processo de elicitação é que muitas vezes os usuários não têm uma idéia precisa e explícita do sistema a ser desenvolvido, ou mesmo possuem dificuldades em descrever seu conhecimento sobre o domínio do problema. Uma técnica bastante utilizada é o uso de cenários para representar as tarefas que os usuários executam normalmente e aquelas que eles desejam executar [LEI97][SCH98]. A escolha da técnica apropriada para elicitar requisitos depende do tempo e dos recursos disponíveis, assim como do tipo de informação necessária. Algumas das classes de técnicas de elicitação são [NUS00]:

- Técnicas tradicionais – inclui o uso de questionários, entrevistas, análise de documentação existente [WEI96] [MAM99];
- Técnicas de elicitação de grupo – são técnicas de dinâmica de grupo com o objetivo de entender de forma mais detalhada as necessidades dos usuários, estão incluídas: brainstorming, sessões JAD e RAD [DAM97];
- Prototipação – é utilizada para elicitar requisitos quando há um alto grau de incerteza ou quando é necessário um rápido feedback dos usuários [DAV92];

- Técnicas de modelagem – fornece um modelo específico das informações que serão adquiridas, e usa esse modelo para orientar o processo de elicitação. Inclui métodos baseados em metas, tais como: KAOS [LAM98] e I* [YU95] e métodos baseados em cenários como CREWS [MAI98];
- Técnicas cognitivas – inclui uma série de técnicas originalmente desenvolvidas para aquisição de conhecimento para sistemas baseados em conhecimento, alguns exemplos são: Análise de protocolo, *laddering*, *card sorting*, *repertory grids* [SHA96];
- Técnicas contextuais – surgiu como uma alternativa para as técnicas tradicionais e cognitivas, inclui técnicas de etnografia e análise social [VIL99];

O processo de elicitação é uma atividade complexa e quase sempre requer uma quantidade considerável de tempo e recursos. Entretanto, ela é fundamental para garantir que o sistema final atenderá às expectativas e necessidades dos usuários. O resultado final da fase de elicitação é um esboço de documento que contém uma descrição abstrata dos requisitos.

4.2.2 Análise de Requisitos

O principal objetivo dessa fase é fornecer descrições abstratas dos requisitos que possam ser facilmente interpretadas. Existem inúmeras abordagens para modelagem de requisitos [DAV93], modelos podem ser usados para representar todo o processo de engenharia de requisitos. Além disso, muitas abordagens podem ser usadas como ferramentas de elicitação, onde as notações e modelos parciais são usados para direcionar novas aquisições de informações [NUS00]. Alguns exemplos de modelagem são: organizacional [ALE99], comportamental [POT97], de domínio [JAC93], de requisitos não-funcionais [SAN00] [CYS99] [CHU00]. Cada classe de modelagem se destina a analisar e resolver um determinado tipo de problema.

Durante os processos de análise e negociação são encontrados problemas com os requisitos, tais como: requisitos esquecidos, conflitantes, ambíguos e duplicados. A partir da identificação desses requisitos com problemas, os usuários devem discutir, priorizar e negociar esses requisitos até que se obtenha um acordo com possíveis modificações e

simplificações. O processo de negociação de requisitos tenta resolver conflitos entre usuários sem necessariamente comprometer a satisfação dos objetivos de cada usuário. Em geral, os modelos de negociação identificam as principais necessidades de cada usuário, ou seja, atribuem prioridades aos requisitos, em seguida analisam esses resultados para tentar garantir que os requisitos mais críticos sejam atendidos. Em [KAR98], Karlsson descreve um estudo de caso realizado num projeto Ericsson em que duas importantes técnicas para priorização e negociação de requisitos foram comparadas. As técnicas avaliadas foram AHP [SAA90] e QFD [HAU88], os resultados obtidos indicaram que a técnica AHP apesar de ter sido considerada complexa, apresentou melhores resultados.

Em princípio, a negociação de requisitos deveria ser um processo objetivo. As decisões a serem tomadas em relação aos requisitos do sistema deveriam ser baseadas nas necessidades técnicas e organizacionais [KOT97]. Entretanto, a realidade é normalmente diferente. Negociações raramente são conduzidas usando somente argumentos lógicos e técnicos, elas também são fortemente influenciadas pelas necessidades pessoais dos indivíduos.

4.2.3 *Documentação de Requisitos*

Depois que os requisitos do sistema são concordados, eles devem ser documentados com um nível apropriado de detalhes. Em geral, é necessário que o documento de requisitos seja entendível por todos os envolvidos no processo de engenharia de requisitos pois ele servirá como um contrato entre usuários e desenvolvedores.

O principal foco da pesquisa em documentação de requisitos é prover notações e linguagens de especificação. Desde linguagem natural [AMB97] à lógica [CAS96] [ANT98], diferentes linguagens têm sido propostas para expressar e descrever requisitos. Pesquisas atuais têm reconhecido que o gerenciamento de requisitos é uma atividade crucial no processo de engenharia de requisitos, ou seja, é necessário não somente escrever os requisitos de forma entendível, mas também permitir que eles possam ser rastreados e gerenciados ao longo da evolução do sistema [GOT94]. Rastreamento de requisitos é um

fator importante para prover uma documentação dos requisitos completa e com integridade, assim como ajudar no processo de gerenciamento de mudanças nesses requisitos [TOR99].

Segundo Ryan [RYA93], o documento de requisitos é o meio através do qual é possível descrever as restrições quanto às características do produto e quanto ao processo de desenvolvimento, a interface com outras aplicações, a descrição sobre o domínio e as informações de suporte ao conhecimento do problema.

4.2.4 *Validação de Requisitos*

Segundo Loucopoulus [LOU95], o processo de validação de requisitos é definido como a atividade na qual certifica que o documento de requisitos é consistente com as necessidades dos usuários. Descrever os requisitos de forma explícita é uma condição necessária não somente para validar os requisitos mas também para resolver conflitos entre usuários. Em geral, a validação de requisitos é considerada uma atividade complicada por dois principais motivos. Inicialmente, a própria natureza filosófica desse processo, onde é levado em consideração o que é verdadeiro, permite que muitos pesquisadores comparem a validação de requisitos com o problema de validação do conhecimento científico [NUS00]. O segundo motivo é social e está relacionado com a dificuldade de obter um consenso entre diferentes usuários com objetivos conflitantes [LAM98].

Outro grande desafio durante a validação de requisitos é demonstrar que a especificação dos requisitos do sistema está correta. Enquanto o projeto e a implementação possuem o documento de requisitos para verificar seus resultados, a validação de requisitos não possui nenhuma outra representação que pode ser usada como base. Ou seja, não existe uma forma para demonstrar formalmente que a especificação está correta [KOT97]. Nessa seção foram descritas as fases do processo de Engenharia de Requisitos para o desenvolvimento de sistemas de software tradicionais, a próxima seção trata esse mesmo processo para o desenvolvimento baseado em COTS.

4.3 UMA PROPOSTA DO PROCESSO DE ENGENHARIA DE REQUISITOS PARA DBC

Nessa seção serão analisadas as principais implicações da hipótese 2:

H 2 Novos desafios surgem no processo de engenharia de requisitos para desenvolvimento baseado em COTS que não existiam no processo tradicional.

Assim como no desenvolvimento de software convencional, o processo de engenharia de requisitos é fundamental para o sucesso de sistemas baseados em COTS [NCU00]. Todavia, como foi visto na seção 3.3.6 a maioria dos métodos para seleção de COTS existentes na literatura ainda não tratam o processo de requisitos de forma satisfatória. Além da própria falta de um processo consistente de aquisição e especificação de requisitos, existem novos problemas e desafios associados ao processo de engenharia de requisitos para DBC que precisam ser analisados e resolvidos.

É importante ressaltar que existem alguns trabalhos presentes na literatura que tratam o processo de requisitos para desenvolvimento dos próprios componentes COTS [POT95] [SOU99]. Deifel [DEI98], por exemplo, propõe um modelo de requisitos que consiste de quatro fases e descreve os principais problemas para desenvolvimento de CCOTS (*Complex COTS*) que ocorrem em cada fase. Ele mostra que um dos principais desafios desse processo é que os produtos são desenvolvidos para um mercado inteiro, ao invés de um único cliente. Assim não existe uma comunicação direta entre usuários e desenvolvedores.

Uma das principais diferenças entre o processo de engenharia de requisitos para desenvolvimento convencional e para baseado em COTS, é que no primeiro pode existir uma garantia maior de que os requisitos especificados sejam atendidos, uma vez que o sistema será implementado a partir do documento de requisitos. Assim, se o sistema não atender as necessidades dos *stakeholders*, isso pode ter sido devido a uma má especificação dos requisitos. Já no desenvolvimento de sistemas baseado em COTS, não se pode garantir que os requisitos dos *stakeholders* sejam integralmente atendidos pois os produtos

existentes no mercado já foram implementados a partir de requisitos preestabelecidos pela organização que desenvolveu o COTS. As próximas seções descrevem uma proposta do processo de engenharia de requisitos sob a perspectiva do desenvolvimento baseado em COTS. A fim de facilitar o entendimento e possibilitar possíveis comparações, será utilizado o mesmo modelo de fases descrito por Kotonya e Sommerville [KOT97] para o processo tradicional e mostrado nas seções acima.

4.3.1 Elicitação de Requisitos para DBC

Em geral, as atividades realizadas durante a elicitação de requisitos para sistemas baseados em COTS são semelhantes ao processo tradicional. Entretanto, vale ressaltar que no contexto de DBC, esse processo perde considerável importância uma vez que muitos requisitos são elicitados no momento que os produtos COTS são examinados. Assim, o processo de elicitação dos requisitos pode ocorrer ao longo do processo de seleção e não apenas na fase inicial desse processo.

Inicialmente, parte-se de uma descrição abstrata das principais funcionalidades que o produto deve prover, nesse momento deve-se focar principalmente os requisitos funcionais pois são mais fáceis de serem adquiridos e modelados. Em seguida, são feitos sucessivos refinamentos dos requisitos, onde os requisitos não-funcionais devem receber atenção especial já que eles são importantes critérios para avaliação de produtos semelhantes. Os requisitos contratuais são uma importante categoria de requisitos não-funcionais. Eles estão relacionados com os acordos e negociações que devem ser realizados entre os vendedores de COTS e a organização que está adquirindo esse produto. Segundo a classificação de Kotonya para os requisitos não-funcionais [KOT98], os requisitos contratuais fazem parte dos requisitos externos. Alguns exemplos desses requisitos são [NCU00]:

- Requisitos sobre o vendedor (estabilidade, reputação, maturidade);
- Requisitos sobre o treinamento disponível (qualidade, custo);

- Requisitos sobre o método de pagamento (fixo, prestações);
- Requisitos sobre o tipo de contrato (direitos, tempo de entrega, aspectos legais).

Normalmente, as técnicas para elicitação de requisitos para DBC são as mesmas usadas no desenvolvimento tradicional. Algumas das técnicas mais comuns são: entrevistas, *brainstorming*, questionários, *card sorting* [MAI96]. Cada técnica é particularmente útil para uma etapa do processo de aquisição de requisitos. Em geral, deve-se iniciar o processo de busca a partir da descrição dos requisitos considerados críticos pelos *stakeholders*. Normalmente, nem todos os requisitos adquiridos durante o processo de elicitação poderão ser atendidos. Assim a fase de análise e negociação tem um papel fundamental para avaliar a importância de cada requisito e contribuir para que esses requisitos sejam priorizados de forma consistente com as reais necessidades dos *stakeholders*.

4.3.2 *Análise de Requisitos para DBC*

Nesse momento, a partir dos requisitos adquiridos na fase de elicitação já é possível iniciar o processo de busca dos potenciais produtos candidatos. Durante o processo de localização, a análise dos requisitos deve ajudar a discriminar entre dois produtos semelhantes. É importante ressaltar que durante essa fase, os requisitos não-funcionais precisam ser explicitamente modelados para evitar possíveis inconsistências e tratar conflitos entre os mesmos. Dessa forma, a avaliação entre produtos que provêm funcionalidades semelhantes pode ser feita a partir da análise dos requisitos não-funcionais que cada produto suporta. A seção 4.5 descreve uma técnica largamente aceita pela comunidade de Engenharia de Requisitos [CYS99] [KOT97] [SAN00] para modelagem de requisitos não-funcionais e que está sendo utilizada nesse trabalho para avaliar os aspectos de qualidade dos produtos COTS.

No desenvolvimento convencional, a análise dos requisitos é geralmente direta pois os *stakeholders* descrevem o futuro sistema através de modelos que contêm um conjunto de atributos e condições específicas que o sistema deve suportar. Entretanto, a análise dos requisitos é diferente no desenvolvimento de sistemas baseados em COTS, pois a

especificação dos requisitos deve ser flexível a ponto de aceitar algumas funcionalidades já presentes nos produtos e que não foram inicialmente descritas [CAR97]. Assim a fase de análise para DBC deve suportar um extenso processo de modificação dos requisitos especificados.

Também existem casos onde os produtos presentes no mercado não atendem a todos os requisitos críticos descritos pelos stakeholders. Assim pode-se tomar basicamente duas decisões para resolver esse problema. A primeira sugere que a especificação dos requisitos sofra ligeiras alterações para suportar tais restrições. Já a segunda propõe que essas funcionalidades, que não podem ser obtidas em produtos COTS, sejam implementadas da forma tradicional. Assim, deve ser realizado um extenso processo de priorização e negociação de tais requisitos em relação aos produtos disponíveis. Esse trabalho usa a técnica AHP [SAA90] para auxiliar esse processo, ela tem sido usada em diversos trabalhos para priorização de requisitos para seleção de software [HON81] [NCU00] [MIN92] [KON96b] [KUN99], ver seção 3.4.3.

4.3.3 *Documentação de Requisitos para DBC*

Nessa etapa os requisitos dos usuários são descritos num documento que deve conter uma especificação detalhada dos requisitos funcionais e não-funcionais assim como mostrar as prioridades de cada requisito. Os requisitos priorizados podem se enquadrar em quatro categorias:

- Essencial e imperativo – prioridade muito alta;
- Importante – prioridade alta;
- Baixa importância – prioridade média;
- Sem importância – prioridade baixa;

É importante que o documento de requisitos permita um efetivo gerenciamento dos requisitos pois eles normalmente serão alterados ao longo do processo de seleção de COTS. Um dos principais motivos de mudança na especificação dos requisitos é devido a conflitos

entre requisitos, no qual a maioria dos conflitos é relativa aos requisitos não-funcionais. Assim é necessário que tais requisitos sejam explicitamente descritos no documento.

Outro aspecto que precisa ser tratado durante essa fase é o rastreamento dos requisitos. Uma vez que é fundamental documentar as decisões e modificações feitas nos requisitos durante todo o processo de desenvolvimento baseado em COTS. Em particular, é importante descrever os motivos porque os requisitos tiveram que ser modificados. Além disso, o rastreamento de requisitos deve permitir que os sucessivos refinamentos dos requisitos fiquem disponíveis para auxiliar o processo de seleção de COTS. Segundo [GOT94], o processo de rastreamento garante a satisfação dos stakeholders disponibilizando um documento que mostra que todos requisitos estão sendo atendidos. No caso do desenvolvimento baseado em COTS, o rastreamento de requisitos possibilita um entendimento dos motivos pelos quais os produtos foram priorizados e selecionados.

4.3.4 *Validação de Requisitos para DBC*

O processo de validação dos requisitos precisa ser feito em paralelo com a atividade de avaliação dos produtos COTS. Normalmente, a validação é diretamente afetada pela disponibilidade dos produtos existentes no mercado e que satisfazem os requisitos especificados. Uma possível atividade desse processo é validar, ou seja, aceitar as funcionalidades existentes nos produtos e que não foram especificadas pelos *stakeholders* ao longo das fases processo de requisitos.

Um dos principais desafios dessa fase é obter um consenso entre os *stakeholders* em relação aos requisitos que tiveram que ser simplificados ou até mesmo excluídos devido à falta de produtos que atendessem tais requisitos. Em casos em que um número mínimo dos requisitos críticos não puderem ser satisfeitos, os *stakeholders* devem concordar que eles deverão ser implementados da forma tradicional.

4.4 ASPECTOS RELEVANTES DO PROCESSO DE ENGENHARIA DE REQUISITOS PARA DBC

Em geral, é conveniente que todas as fases do processo de engenharia de requisitos sejam realizadas em paralelo e iterativamente com o processo de seleção. Uma vez que depois que o produto é adquirido pode ser difícil e bastante dispendioso modificar requisitos já especificados. Ao contrário do desenvolvimento tradicional, onde é possível alterar requisitos durante as fases seguintes do processo, mesmo que isso signifique um considerável custo adicional, no desenvolvimento baseado em COTS não é possível modificar as características do produto. Isso é devido à característica caixa-preta da maioria dos produtos disponíveis no mercado, que impossibilita acesso ao código-fonte. Normalmente os vendedores de produtos COTS disponibilizam uma boa descrição das funcionalidades dos produtos. Ao contrário, os aspectos de qualidade (i.e. requisitos não-funcionais) quase sempre são superficialmente descritos, onde boa parte das informações é relativa aos seguintes aspectos:

- Compatibilidade com determinado sistema operacional;
- Requisitos de Hardware (espaço e memória requeridos);
- Interoperabilidade com outros sistemas;
- Existência de tutorial para facilitar o uso, etc.

Entretanto, a maioria dos vendedores não fornece informações sobre outros aspectos de qualidade críticos, a menos que ela seja solicitada. Alguns requisitos não-funcionais são de extrema importância durante a seleção de produtos COTS e em caso de conflito, podem inclusive ter prioridade sobre outros requisitos funcionais, por exemplo:

- Conformidade com os padrões organizacionais;
- Usabilidade do produto;
- Flexibilidade para futuras mudanças, etc.

Além desses aspectos, vale ressaltar que em muitos casos os requisitos não-funcionais correspondem aos objetivos estratégicos da organização [DUK99]. Assim, não é uma mera coincidência que boa parte dos requisitos não-funcionais sejam solicitados pelos gerentes. Um outro fato que precisa ser considerado é que geralmente, os gerentes fazem parte da equipe de seleção e eles são os responsáveis pelas decisões finais para aquisição de determinado produto COTS. Portanto, a partir dessas observações, pode ser evidenciada a importância para realizar uma boa descrição dos requisitos não-funcionais a fim de obter um processo de seleção de COTS consistente e que satisfaça as reais necessidades dos *stakeholders*. A próxima seção aborda esse assunto e descreve um método para modelagem de requisitos não-funcionais.

4.5 MODELAGEM DE REQUISITOS NÃO-FUNCIONAIS

De um modo geral, os requisitos não-funcionais (*RNF*) definem qualidades e restrições sobre os sistemas de software [KOT97]. Eles possuem um papel crítico durante o desenvolvimento de software, entretanto, de forma surpreendente, eles têm recebido pouca atenção na literatura [CYS01] [SAN00] [CHU00]. Eles são quase sempre descritos informalmente durante a fase de análise de requisitos e isso aumenta os riscos de contradições e conflitos na especificação dos requisitos não-funcionais. Alguns aspectos fundamentais de tais requisitos são:

- Subjetividade – eles são interpretados e avaliados por diferentes pessoas que têm diferentes perspectivas e necessidades, assim eles podem ter diferentes significados para cada pessoa;
- Relatividade – sua interpretação e importância dependem diretamente de cada sistema em particular, além disso sua realização é relativa (ex. um RNF pode ser essencial para uma pessoa enquanto para outra não seja tão importante);
- Interatividade – eles interagem entre si, assim a realização de um RNF pode interferir positivamente ou negativamente outros requisitos.

Por todos esses motivos, quase sempre é bastante difícil e complexo elicitar, expressar e quantificar requisitos não-funcionais [DUK99]. Existem alguns métodos presentes na literatura para assistir a descrição de requisitos não-funcionais [DOB91] [KOT93] [LOU95]. Dentre esses trabalhos, o NFR *Framework* proposto por Chung [CHU00] é considerado um dos métodos mais completos e bem aceitos para modelar requisitos não-funcionais [CYS99].

O NFR *Framework* é uma abordagem orientada a processo onde os requisitos não-funcionais são explicitamente representados como metas a serem obtidas [CHU95a]. Uma importante característica dessas metas é que nem sempre elas podem ser inteiramente satisfeitas. Ao contrário, existem graus de satisfação que podem ser obtidos. Essa abordagem pode tratar melhor os aspectos de subjetividade, relatividade e interatividade inerentes dos RNF. Dessa forma, Chung [CHU00] propõe a noção de *softgoal* para descrever as metas que precisam obter níveis de satisfação.

Uma premissa fundamental dessa abordagem é que os *softgoals* possuem a propriedade de interagir entre si, em conflito ou sinergia [CHU95b]. Os *softgoals* são decompostos em outros *softgoals* e representados numa estrutura gráfica inspirada nas árvores E/OU para solução de problemas, esses grafos são chamados SIG (*Softgoal Interdependency Graphs*). A aplicação do NFR *Framework* consiste de dois estágios principais, Aquisição do Conhecimento e Aplicação do *Framework*. A seguir são descritas as atividades que deverão ser executadas em cada fase [CHU00].

Fase 1 - Aquisição do Conhecimento

- Adquirir conhecimento específico dos RNF – Essa atividade consiste de obter conhecimento sobre os requisitos não-funcionais a partir de experiências práticas ou literatura disponível. Em seguida, os RNF são incluídos em catálogos específicos e que serão usados como fonte de informação para as próximas fases;

- Adquirir conhecimento do domínio – Durante essa atividade são adquiridas informações sobre o domínio do problema e uma idéia geral dos principais requisitos do sistema.

Fase 2 – Aplicação do NFR Framework

- Identificar conceitos relacionados aos RNF – Nesse momento é feita uma estimativa inicial das prioridades da organização, isso inclui os requisitos críticos. Além disso, são identificados os possíveis métodos ou operacionalizações desses requisitos como também os argumentos para priorizar os requisitos;
- Refinar e inter-relacionar conceitos dos RNF – Essa fase inclui uma série de atividades, são elas:
 - Refinar e relacionar os *softgoals* para esclarecer possíveis ambigüidades;
 - Relacionar métodos para operacionalizar *softgoals*;
 - Identificar *tradeoffs*, interdependências (conflito/sinergia) entre os *softgoals*;
 - Identificar prioridades, assim como partes críticas e dominantes;
 - Fornecer argumentos para tomada de decisão e motivos para escolher determinado método;
 - Avaliar a realização dos *softgoals* (particularmente os prioritários).

É fundamental ressaltar que os passos referentes à fase 2, não precisam ser sequenciais como mostrado aqui, eles podem inclusive ser iterativos. Outro esclarecimento importante é a respeito dos métodos e operacionalizações abordados na fase 2. Eles estão no final do grafo SIG (são as “folhas” do grafo) e representam o meio pelo qual os *softgoals* podem ser operacionalizados em funções. Nesse estágio é desejável que todas as ambigüidades e conflitos tenham sido eliminados.

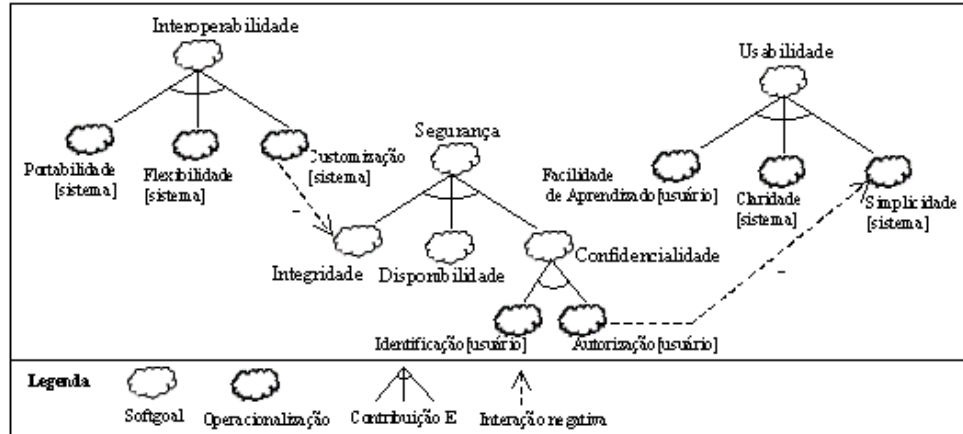


Figura 4.2 - Exemplo de grafo SIG

A Figura 4.2 mostra um exemplo de um grafo SIG representando a decomposição e a interação dos requisitos *Usabilidade*, *Segurança* e *Interoperabilidade*. De acordo com esse grafo, o requisito *Usabilidade*, por exemplo, foi decomposto em *Facilidade de aprendizado*, *Clareza* e *Simplicidade* que o usuário utiliza o sistema. Já o requisito *Segurança* foi decomposto nos requisitos *Integridade*, *Disponibilidade* e *Confidencialidade*. Em seguida, o requisito confiabilidade pode ser operacionalizado pelos métodos *Identificação* e *Autorização* dos usuários. Um aspecto importante desse grafo é que depois que os RNFs foram refinados, puderam ser identificados alguns relacionamentos entre requisitos que aparentemente não estavam relacionados. Nesse exemplo, o relacionamento é negativo, isto é, existe uma relação de conflito entre os requisitos. Em particular, a operacionalização *Autorização[usuário]* interage em conflito com *Simplicidade[sistema]*. Ou seja, os métodos para autorização dos usuários influenciam negativamente a simplicidade do sistema. Em situações como essa é necessário um extenso processo priorização e negociação desses requisitos, onde a participação dos *stakeholders* assume um papel fundamental.

4.6 CONSIDERAÇÕES FINAIS

Esse capítulo mostrou as fases do processo de engenharia de requisitos tradicional, são elas: elicitação, análise, documentação e validação. Em seguida, foi proposto um modelo para o processo de engenharia de requisitos específico para o desenvolvimento baseado em COTS. Com o objetivo de facilitar o entendimento e proporcionar comparações, essa proposta foi baseada nas mesmas fases descritas para o processo tradicional. Um dos aspectos mais relevantes abordados nesse modelo de engenharia de requisitos para DBC diz respeito à falta de garantia de que todos os requisitos especificados possam ser atendidos já que o produto foi implementado a partir de requisitos predeterminados. Na maioria dos casos, deve existir um extenso processo de priorização e negociação desses requisitos.

Durante a fase de elicitação de requisitos para DBC, foi descrito um importante tipo de requisito não-funcional conhecido como requisitos contratuais. Eles são relacionados com os acordos e negociações que devem ser realizados entre os vendedores de COTS e clientes. A fase de análise para DBC é considerada essencial pois é nesse momento que são tratados os aspectos de priorização e negociação entre requisitos conflitantes. A etapa de documentação se preocupa com o gerenciamento e rastreamento dos requisitos ao longo do processo de DBC. Finalmente, a fase de validação procura obter um consenso entre os *stakeholders* sobre as modificações necessárias na especificação dos requisitos para ficar em conformidade com as capacidades dos produtos disponíveis no mercado.

Nesse capítulo, foi ressaltada a necessidade de uma especificação consistente e detalhada dos requisitos não-funcionais. Pois, como foi mostrado na seção 4.4, eles são aspectos fundamentais para auxiliar o processo de seleção de COTS. Além da própria dificuldade para elicitar, expressar e quantificar os requisitos não-funcionais; ainda existem inúmeras questões relativas ao processo de seleção que precisam ser analisadas e resolvidas. Alguns dos aspectos de extrema relevância são:

- Como selecionar produtos COTS com base nos requisitos não-funcionais?
- Quais os atributos dos produtos precisam ser examinados?

- Como priorizar e negociar requisitos conflitantes?
- Como avaliar e priorizar produtos aparentemente semelhantes?
- Como decidir se é melhor implementar o sistema ao invés de adquirir um COTS?

A fim de proporcionar um direcionamento para futuras pesquisas na área, o próximo capítulo pretende analisar e discutir esses pontos. Além disso, será proposto um método para assistir o processo de seleção de COTS que enfatiza o tratamento dos requisitos não-funcionais a fim de obter uma seleção consistente e sistemática.

Capítulo 5

O Método CRE

Este capítulo pretende descrever a principal contribuição deste trabalho, o método CRE. Será mostrada individualmente cada uma das quatro fases que compõem o método, são elas: identificação, descrição, avaliação e aceitação.

5.1 DESCRIÇÃO GERAL

Este trabalho propõe um novo método para assistir o processo de seleção de COTS, ele é chamado Método CRE (*COTS-based Requirements Engineering*) [ALV00] [ALV01]. Esse método é orientado a requisitos e tem como uma das principais estratégias enfocar a descrição dos requisitos não-funcionais durante o processo seletivo. Essa abordagem é bastante promissora pois pesquisas recentes mostram que os requisitos não-funcionais têm um papel crítico durante o desenvolvimento de sistemas [CYS99] [CHU00] [SAN00].

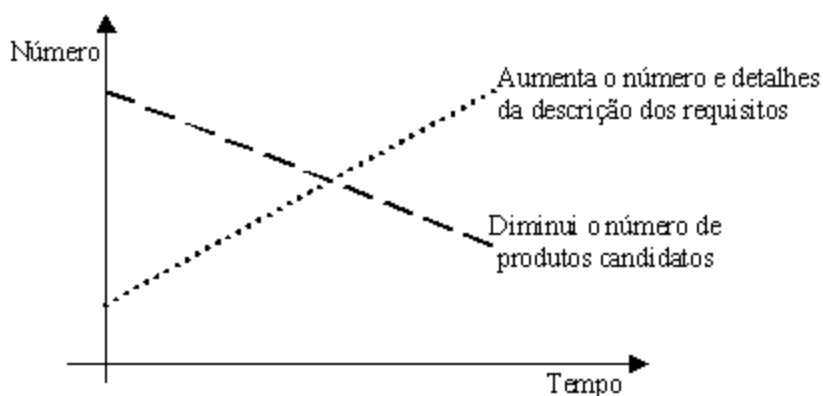


Figura 5.1 - Processo Iterativo de Aquisição de Requisitos e Seleção de Produtos

O método CRE é fundamentado no processo iterativo de aquisição de requisitos e seleção/rejeição de produtos. No início desse processo, existem poucos requisitos descritos e um grande número de produtos candidatos. À medida que ele evolui, a quantidade de requisitos aumenta e assim diminui o número de COTS que atendem à esses requisitos, ver Figura 5.1. Esse processo continua até que um produto satisfaça um número suficiente de requisitos. O método CRE possui um mecanismo de avaliação exaustiva (ver seção 2.4.1) pois os produtos são examinados detalhadamente em relação aos requisitos dos *stakeholders*. Além disso, a estratégia de seleção usada pelo CRE é considerada uma filtragem progressiva (ver Seção 3.3), como pode ser observado pelo processo iterativo entre aquisição de requisitos e eliminação de produtos. A próxima seção descreve as fases do método proposto.

5.2 AS FASES DO MÉTODO CRE

O método CRE é orientado a metas, isto é, as fases do processo são direcionadas por metas previamente definidas. Cada fase do CRE possui um *template* cujo objetivo é fornecer diretrizes para serem seguidas durante a realização das atividades de cada fase do método. Esses *templates* mostram as principais metas e o resultado final que deve ser obtido em cada fase. Eles também descrevem os principais passos que precisam ser executados, além de indicar as técnicas que serão usadas durante cada fase.

O método CRE possui quatro fases que interagem entre si, são elas: Identificação, Descrição, Avaliação e Aceitação. Em particular, a ordem apresentada aqui não é rígida e a atividade que será realizada depende do tipo de informação disponível no momento e da necessidade para obter determinado resultado. Dessa forma, a seqüência dos processos não é preestabelecida e cada um deles pode ser repetido inúmeras vezes. Além disso, os *templates* de cada fase podem ser usados inúmeras vezes durante o processo de seleção de COTS.

O método CRE sugere que o processo de engenharia de requisitos deve direcionar todo o processo de seleção. Portanto, as etapas do processo de requisitos podem ser realizadas e repetidas em cada fase do CRE. A Figura 5.2 fornece uma visão geral do método CRE, mostrando as atividades de cada uma das suas fases e os relacionamentos entre as mesmas. Além disso, é explicitada a constante interação que existe entre as fases do método. Nas próximas seções cada fase do método será descrita individualmente.

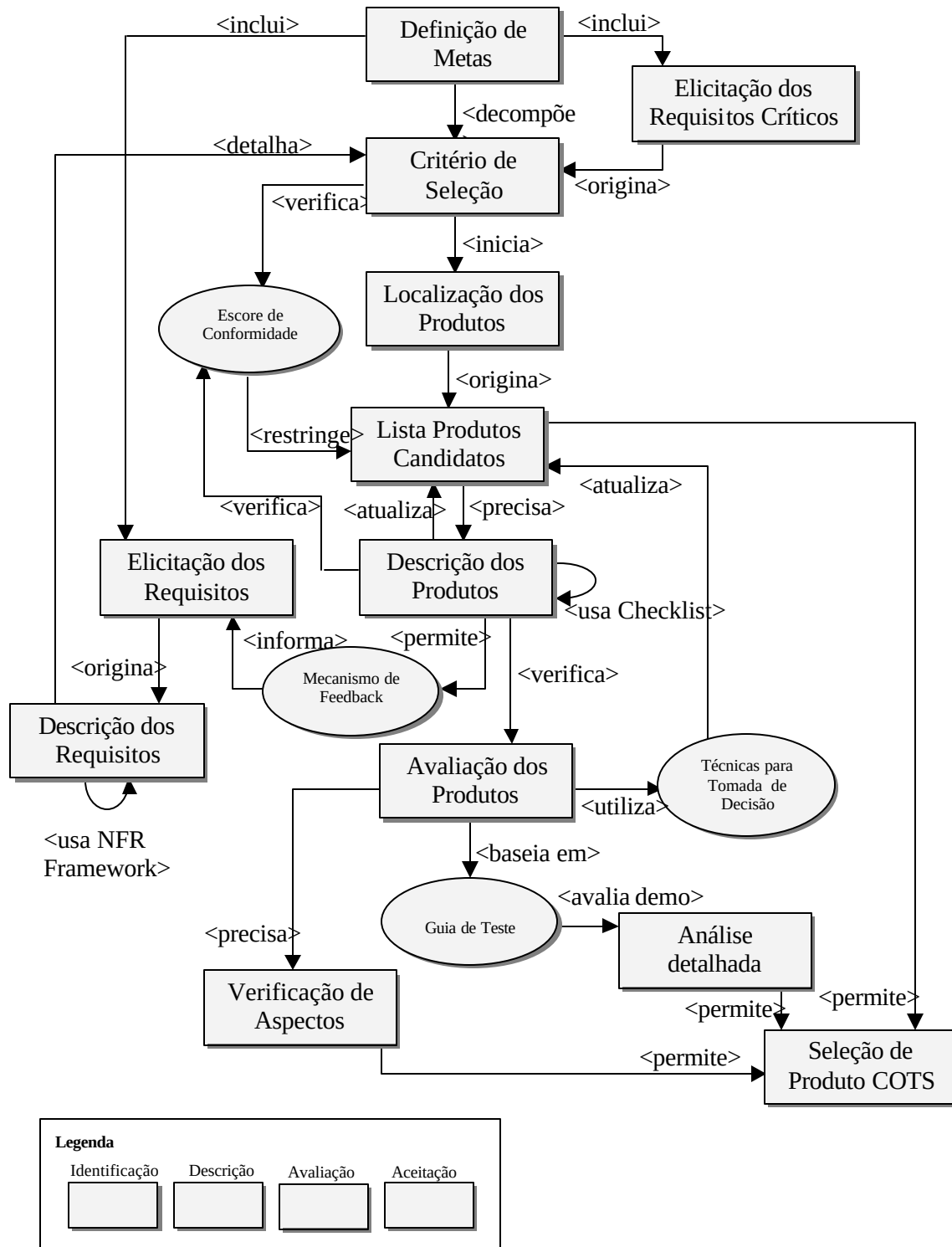


Figura 5.2 - Visão Geral do Método CRE

5.2.1 Identificação

O principal objetivo dessa fase é identificar os possíveis produtos candidatos. A primeira etapa do processo de identificação de COTS é definir as metas que precisam ser atingidas para obter um processo de seleção bem sucedido. Em geral, uma meta básica de qualquer processo de aquisição de COTS é selecionar um produto que atenda reais necessidades dos *stakeholders*. Nesse contexto, torna-se clara a importância da engenharia de requisitos para auxiliar o processo de especificação dos requisitos. A definição das metas corresponde à elaboração do critério de seleção, que precisa ser baseado numa análise detalhada de fatores que influenciam o processo de aquisição. Esse trabalho utiliza a classificação proposta por [KON96b] que identifica cinco grupos de fatores que influenciam a seleção de COTS, são eles:

- Requisitos dos *stakeholders* – inclui os requisitos funcionais e não-funcionais, inicialmente eles são responsáveis pelo processo de localização de produtos, em seguida o documento de requisitos é usado para fazer a análise de conformidade entre os produtos candidatos e os requisitos [ALV01];
- Arquitetura da aplicação – descreve as restrições arquiteturais que o produto deve atender, estão incluídos: padrões de comunicação e de interface, características de plataformas, etc. Todas essas características introduzem um conjunto de restrições que tornam a integração de algumas alternativas impraticáveis ou bastante caras. Em [ROS01] é proposta uma abordagem de arquitetura de software formal para tratar a seleção de componentes COTS;
- Objetivos e restrições do projeto – os objetivos do projeto descrevem os benefícios organizacionais [ALE99] que são esperados com a aquisição de um novo COTS, existem também restrições organizacionais que precisam ser consideradas pois normalmente são limitações de recursos e tempo disponível;
- Disponibilidade de produtos – é importante que as metas traçadas sejam realísticas com as opções existentes no mercado, por outro lado é útil examinar as funcionalidades que os produtos oferecem e que não foram incluídas no documento de requisitos;

- Infra-estrutura organizacional – inclui a maturidade e experiência da organização em relação a processos de aquisição de COTS. Nesse contexto, o conhecimento do domínio e a experiência da equipe de seleção são fatores importantes para o sucesso desse processo.

É aconselhável que o critério de seleção seja elaborado a partir de uma análise cuidadosa desses fatores. Num contexto geral, é importante que cada fator seja explicitamente avaliado e então usado como base para a definição inicial do critério de seleção. A Figura 5.3 mostra a elaboração do critério de seleção a partir das metas que devem ser baseadas nos cinco fatores vistos acima. Geralmente, os aspectos relacionados às restrições do projeto são críticos já que a maioria das organizações trabalha com limitações de tempo e recursos. Dessa forma, é necessário analisar quais os reais benefícios que serão obtidos com a utilização das técnicas propostas pelo método CRE.

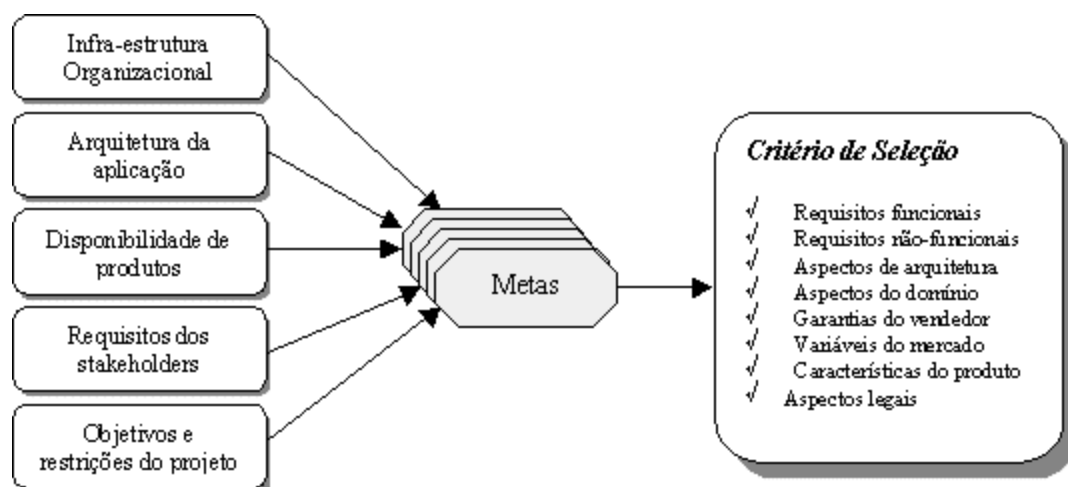


Figura 5.3 - Elaboração do Critério de Seleção

O primeiro passo para elaboração do critério de seleção é elicitar os requisitos críticos ou imperativos, onde estão incluídos os requisitos que descrevem as funcionalidades centrais (ex. o produto deve fornecer um corretor ortográfico) e algumas restrições chave (ex. o custo deve ser inferior a R\$ X) que os produtos precisam atender. Em geral, os requisitos críticos são quase sempre inflexíveis e raramente sofrem grandes alterações. Nesse estágio, as técnicas mais apropriadas para elicitar os requisitos críticos são entrevistas e

questionários. Já que não é necessário um alto nível de detalhamento dos requisitos e essas informações podem ser obtidas a partir de algumas reuniões entre os *stakeholders* e a equipe de seleção. Além disso, é aconselhável ter alguns especialistas do domínio fazendo parte da equipe de seleção, onde eles assumem um importante papel durante esse processo de elicitación inicial.

O propósito inicial do critério de seleção é identificar as principais características que os produtos precisam atender para iniciar o processo de busca. O objetivo desse processo é encontrar todos os potenciais produtos candidatos existentes no mercado. Basicamente, existem duas formas de realizar esse processo de localização de produtos COTS e cada organização deve adotar a mais adequada a fim de obter suas metas. Em casos onde não existe um consenso sobre quais são os requisitos críticos, uma possível abordagem é considerar todos os produtos encontrados que aparentemente atendem a uma funcionalidade global (ex. considerar todos os sistemas de informação em saúde existentes no mercado). Nesses casos, normalmente é obtida uma extensa lista de produtos candidatos que deve ser refinada ao longo das outras fases propostas pelo método. Já em casos onde é possível estabelecer os requisitos críticos de forma clara e bem definida, pode-se partir de um grupo reduzido de produtos que realmente atendem tais requisitos.

O processo de busca é uma atividade informal e extremamente dependente de fatores externos, assim não existe um processo padrão. Porém, existem algumas diretrizes que podem ser seguidas. Primeiramente, é importante usar várias fontes de informação para auxiliar a busca, além disso, é aconselhável documentar todo o processo de busca, mesmo que a organização não repita esse processo com frequência. Isso possibilita que o processo de seleção seja uma atividade sistemática e bem definida. As principais fontes para localização de produtos COTS são [ALV00]:

- Bibliotecas internas – grandes organizações possuem bibliotecas de produtos desenvolvidos por outros setores da mesma organização;
- Internet – existe uma grande variedade de informações atualizadas sobre os produtos comerciais existentes no mercado de software;

- Revistas e jornais especializados – essas publicações normalmente mostram comparações entre produtos e possuem uma grande quantidade de anúncios;
- Vendedores – depois que os principais vendedores foram descobertos, pode-se adquirir informações sobre os seus produtos e também identificar seus principais competidores e possíveis candidatos no processo de seleção;
- Consultores e especialistas – esses profissionais possuem um bom conhecimento da área e podem fornecer informações sobre os produtos existentes;
- Conferências e feiras – geralmente, esses eventos possuem várias exposições e stands de venda de produtos de software;
- Outras organizações – identificar organizações que já passaram por um processo de seleção semelhante.

Um dos grandes desafios durante a atividade de localização de COTS é decidir quando parar o processo de busca. Uma estratégia simples é utilizar uma boa diversidade de fontes de procura, verificar a frequência que novas alternativas são descobertas e analisar quando não existir um número considerável de novas descobertas. Assim, pode-se presumir que os benefícios de uma procura adicional serão irrelevantes. Uma situação que precisa ser analisada com cuidado é quando os requisitos críticos são bastante específicos e não podem ser atendidos por nenhum produto disponível no mercado, em casos dessa natureza deve-se negociar com os *stakeholders* como proceder. O método CRE sugere duas alternativas para contornar esse problema, são elas: optar por implementar o sistema da forma tradicional ou então “relaxar” as imposições dos requisitos críticos para que eles se adequem aos produtos disponíveis.

Durante o processo de identificação de COTS é feita uma priorização superficial dos requisitos pois existem poucos requisitos especificados e geralmente eles ainda estão pouco detalhados. Nessa etapa, a validação dos requisitos é parcial pois só foram descritos os requisitos críticos e não existe garantia que todos os requisitos especificados poderão ser totalmente atendidos pelos produtos COTS.

LISTA DE PRODUTOS CANDIDATOS

PRODUTO 1

Nome:

Descrição geral:

Informações importantes:

Escore de conformidade

Req1 –

Req2 –

...

ReqN –

Comentários:

...

...

Figura 5.4 - Modelo para lista de produtos candidatos

Vale ressaltar que durante o processo de identificação, é fundamental documentar todas as informações adquiridas sobre os requisitos e os produtos COTS pois elas serão utilizadas ao longo das fases seguintes do método CRE. Assim, uma das principais atividades dessa fase é elaborar uma lista com produtos candidatos, ver Figura 5.4. Além de fornecer informações gerais sobre os COTS, cada produto dessa lista possui um escore de conformidade, que verifica de forma quantitativa se o produto atende aos requisitos críticos. Esse trabalho propõe a seguinte escala de conformidade:

<i>Conformidade</i>	<i>Escore</i>
Não atende o requisito	0
Atende com sérias restrições	1
Atende parcialmente	2
Atende	3

Tabela 5.1 - Escore de conformidade entre produtos e requisitos

É importante ressaltar que os valores estipulados para o escore de conformidade foram baseados na indicação proposta pelo método WSM (ver Seção 3.4.2) que será usado mais adiante e utiliza tais escores. Pela própria natureza desse processo inicial, quase sempre o número de requisitos é relativamente pequeno e eles são pouco detalhados. Portanto, nesse estágio o documento de requisitos consiste somente de uma breve descrição dos requisitos críticos. O método CRE fornece o *template para identificação dos produtos* (vide Figura 5.5), que mostra algumas diretrizes básicas para assistir a fase de identificação. Esse *template* provê indicações de quais tipos de requisitos elicitar, assim como descreve o tipo de informação que deve ser adquirida sobre os produtos que compõem a lista de COTS candidatos.

Esse *template* recomenda que o critério de seleção seja inicialmente simples e permita avaliar a conformidade dos produtos a partir de informações quantitativas (ex. quantos usuários podem usar o produto simultaneamente?) e respostas booleanas (ex. o produto é compatível com Microsoft Word?). O *template* também fornece uma orientação sobre os possíveis passos que devem ser realizados durante essa fase. Um importante passo descrito nesse *template* é o processo de rejeição dos produtos que não atendem uma quantidade mínima de requisitos críticos. Esse tipo de decisão é específico para cada projeto e cabe à equipe de seleção decidir se vai estipular uma pontuação mínima que o produto candidato deve atingir. Essa pontuação é baseada num percentual de conformidade em relação à pontuação máxima. Por exemplo, supondo que para um dado processo de seleção existem 5 requisitos críticos, então a pontuação máxima que um produto pode obter em relação à conformidade com esses requisitos será 15. Se a equipe de seleção determinar que para um produto ser incluído na lista, ele deve atender no mínimo 80% dos requisitos críticos então a pontuação mínima será 12. Para tratar situações como essa, foram definidas regras de situação que são baseadas em implicações lógicas do tipo IF < condição> THEN < ação>. Abaixo é mostrada uma regra para tratar o processo de rejeição inicial de produtos.

```
If escore_total >= pontuação_min  

Then incluir produto na lista  

Else rejeitar produto
```

O resultado mais significativo da fase de identificação é uma lista com os produtos candidatos que será examinada detalhadamente nas próximas fases. Nesse estágio inicial, o critério de seleção descreveu somente os requisitos críticos. Entretanto, outros aspectos referentes ao critério de seleção (ver Figura 5.3) precisam ser considerados (ex. analisar se os produtos atendem os requisitos não-funcionais, verificar garantias do vendedor, avaliar o custo associado de cada alternativa, etc.) para realizar uma seleção de COTS consistente com as reais necessidades dos *stakeholders*. A Figura 5.5 mostra o *template para identificação dos produtos* que orienta essa fase.

Template para Identificação dos Produtos

METAS

- Elicitar os requisitos críticos dos stakeholders;
- Elaborar critério de seleção;

• Guiar o processo de localização de produtos; • Adquirir informações sobre os produtos identificados; • Atribuir escores de conformidade entre os produtos e os requisitos críticos; • Disponibilizar técnicas para guiar a aquisição de requisitos e de informações sobre os produtos. RESULTADO FINAL Compor uma lista com os produtos COTS candidatos para ser examinada nas próximas fases. REQUISITOS PARA ELICITAR Requisitos críticos que quase sempre são inflexíveis, nesse momento são elicitados os requisitos funcionais críticos e alguns requisitos não-funcionais essenciais. AQUISIÇÃO DE INFORMAÇÕES SOBRE OS PRODUTOS Nessa fase as informações são gerais e podem incluir: descrição geral sobre as capacidades do produto (ex. funcionalidades, requisitos de hardware, sistema operacional requerido) além de outras informações relevantes (ex. custo de aquisição, referências anteriores, tempo que o produto está no mercado). SEQUÊNCIA DE PASSOS 1. As técnicas para elicitação de requisitos mais adequadas nesse momento são entrevistas e questionários; 2. Usar uma estratégia para localização dos COTS (é específica para cada organização e tipo de produto); 3. Adquirir informações sobre os produtos (sugestões Web, revistas e questionários aplicados com vendedores); 4. Obter o escore de conformidade entre os produtos candidatos e os requisitos críticos; 5. Rejeitar os candidatos que não atingem a pontuação mínima; 6. Elaborar a lista de produtos candidatos e incluir informações obtidas anteriormente; CONSIDERAÇÕES IMPORTANTES • Durante a elicitação inicial é aconselhável usar o conhecimento que a equipe de seleção tem sobre o domínio; • Nesse estágio inicial os produtos são mais fáceis de avaliar em relação à sua funcionalidade; • O processo de tomada de decisão é simples e direto pois os produtos que não atendem a pontuação mínima são eliminados; • É fundamental documentar todas as informações obtidas e decisões tomadas.

Figura 5.5 – *Template* para Identificação dos Produtos

5.2.2 Descrição

Nessa etapa, o critério de seleção deverá ser detalhadamente elaborado, onde será dado destaque especial aos requisitos não-funcionais. Além das técnicas tradicionais de

elicitação, como entrevistas e questionários, o método CRE sugere o uso de outras técnicas como *brainstorming*, *card sorting* e *laddering*. Tais técnicas têm sido propostas por outros métodos de seleção e as duas últimas são particularmente úteis para adquirir informações sobre categorias de produtos, além de informações hierárquicas sobre requisitos e propriedades de produtos [NCU99].

A técnica *card sorting* é bastante simples para ser aplicada, ela é usada para adquirir requisitos que discriminam entre produtos, ou seja, requisitos que auxiliam a tomada de decisão. Essa técnica pode ser utilizada da seguinte forma: a equipe de seleção escreve o nome dos COTS candidatos em cartões e pede que os *stakeholders* separem os produtos em categorias, são usados critérios do tipo (ex. o produto deve ser compatível com Microsoft Word) para discriminar entre os produtos. As categorias de produtos (ex. compatível ou não compatível) indicam a conformidade desses produtos em relação aos requisitos segundo o ponto de vista dos avaliadores. A fim de obter melhores resultados, a técnica de *laddering* pode ser usada juntamente com *card sorting*. A principal vantagem de *laddering* é que ela fornece um tratamento padronizado e hierárquico dos requisitos. Para usar essa técnica, os *stakeholders* descrevem categorias comuns e tipos de produtos para descobrir requisitos atendidos por todos os produtos. Dessa forma, pode-se evitar desperdício de tempo detalhando requisitos que não discriminam entre produtos.

Depois que os requisitos foram elicitados, eles precisam ser efetivamente especificados de forma que diminuam possíveis ambigüidades e contradições entre os requisitos. Nesse contexto, a especificação dos requisitos não-funcionais é uma atividade complexa pois quase sempre eles são descritos pelos *stakeholders* de modo abstrato e informal. Além disso, a própria natureza do processo de requisitos para DBC dificulta ainda mais essa atividade. Um dos principais motivos é porque mesmo que os requisitos não-funcionais sejam explicitamente especificados, é difícil analisar se o produto atende a esses requisitos pois geralmente os vendedores não fornecem uma descrição detalhada e precisa das capacidades dos produtos, em especial dos aspectos relacionados aos requisitos de qualidade como: confiabilidade, estabilidade, flexibilidade, eficiência, etc. Além disso, não existe garantia alguma de que os produtos realmente atendam esses requisitos não-

funcionais especificados. Mesmo diante de todas essas dificuldades em relação à especificação de requisitos não-funcionais, esse processo é fundamental para seleção efetiva de produtos COTS. Uma das premissas desse trabalho (hipótese H3) trata exatamente dessa questão. Os principais benefícios para focar requisitos não-funcionais durante essa fase são:

- Eles ajudam a discriminar entre produtos semelhantes (i.e. que já atendem a todos requisitos críticos);
- Geralmente, eles são mais críticos do que outros requisitos funcionais que não fazem parte dos requisitos críticos;
- Eles incluem as restrições da arquitetura (ex. compatibilidade, portabilidade e interoperabilidade) e do projeto (ex. custo e tempo);
- Eles fornecem informações importantes sobre possíveis problemas de adaptação e integração dos produtos ao restante do sistema;
- Eles têm um impacto global na qualidade do sistema;
- Eles estão diretamente relacionados com a satisfação dos *stakeholders*.

Diante desses aspectos, uma das principais metas dessa fase é descrever explicitamente os requisitos não-funcionais e mostrar como eles serão usados para auxiliar o processo de decisão. Como foi visto na seção 4.5, esse trabalho propõe o uso do NFR *Framework* para especificar os requisitos não-funcionais. As principais vantagens dessa abordagem é que ela decompõe os requisitos, ajudando assim que a equipe de seleção entenda e quantifique a prioridade de cada requisito em relação às necessidades descritas (em alto nível) pelos *stakeholders*. Além disso, o NFR *Framework* mostra os possíveis conflitos entre requisitos e portanto contribui para o processo de tomada de decisão que será realizado na próxima fase. No estudo de caso é mostrado como essa abordagem é usada num processo de seleção real. Nesse contexto, uma abordagem bastante promissora para avaliar se os requisitos não-funcionais estão sendo atendidos é estabelecer uma arquitetura de software que descreva como os componentes do sistema devem estar conectados. Vale salientar que a utilização

dessa abordagem é indicada para sistemas complexos e com vários produtos interconectados, em especial, casos em que são selecionados múltiplos produtos. Esta é uma das limitações do método CRE pois ele fornece uma abordagem para seleção de um único produto. No entanto, essa é uma das áreas que se pretende cobrir em trabalhos futuros.

Um importante processo que ocorre durante essa fase é o mecanismo de *feedback*. Ele consiste de uma constante troca de informações entre o processo de requisitos e a descrição dos produtos. Inicialmente, os requisitos são elicitados e então é originada a descrição dos requisitos. A partir dessa descrição, os produtos candidatos são localizados e é obtida uma descrição desses produtos. O método CRE sugere que a análise das características dos produtos permite que novas funcionalidades, inicialmente não especificadas pelos *stakeholders*, sejam descobertas e incluídas no documento de requisitos, ver Figura 5.6. Em geral, o mecanismo de *feedback* é bastante positivo pois auxilia o processo de elicitação dos requisitos. Como essa atividade é reconhecidamente difícil de conduzir [MAM99], esse mecanismo fornece considerável benefício ao processo de engenharia de requisitos para DBC.

No entanto, é importante avaliar com cuidado os casos onde os produtos possuem funcionalidades que vão muito além das necessidades dos *stakeholders*. Nesse casos pode acontecer que a satisfação dos *stakeholders* em relação ao produto que está sendo adquirido seja comprometida pois esse COTS pode mostrar problemas como: exigir muita memória (baixa performance), ser complexo de usar (baixa usabilidade), entre outros. Como pode ser notado, esses problemas estão intimamente relacionados com os requisitos não-funcionais, fato que mais uma vez evidencia a importância de analisá-los com cuidado durante essa fase a fim de não comprometer a satisfação dos *stakeholders* e conseqüentemente o sucesso do processo de seleção.

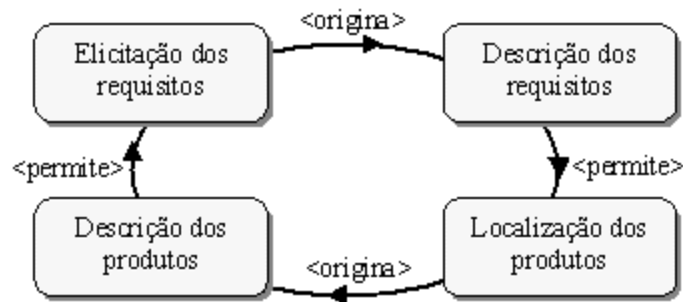


Figura 5.6 - Mecanismo de *Feedback*

Durante essa fase, o documento de requisitos é detalhadamente elaborado e deverá conter todas as informações relevantes sobre os requisitos que o COTS candidato deverá atender. Vale lembrar que a análise do documento de requisitos é uma parte fundamental para elaboração do critério de seleção. A seguir, cada parte do documento (vide Figura 5.7) é explicada:

- Req_id – cada requisito possui um identificador único;
- Tipo – os requisitos podem ser pertencer à dois tipos (funcionais e não-funcionais);
- Subtipo – eles podem ser (críticos ou desejáveis);
 - Críticos - requisitos imprescindíveis de serem atendidos
 - Desejáveis - requisitos que podem ou não ser atendidos
- Descrição – detalha as informações sobre o requisito;
- Prioridade – os requisitos podem ter prioridade variando de 1 até 4¹;
 - 4 - prioridade muito alta (essencial e imperativo)
 - 3 - prioridade alta (importante)
 - 2 - prioridade média (baixa importância)
 - 1 - prioridade baixa (sem importância)

¹ Essa escala é a mesma adotada por Karlsson [KAR98] no seu trabalho sobre priorização de requisitos.

- Contribuições – os requisitos não-funcionais podem interagir de forma positiva ou negativa com outros requisitos;
 - (+ +) make²
 - (+) help
 - (-) hurt
 - (- -) break
- Origem – explicita a origem do requisito, ou seja, qual o papel do *stakeholder* que o descreveu e assim fornece dados para realizar o rastreamento dos requisitos;
- Comentários – mostra as possíveis observações sobre esse requisito;

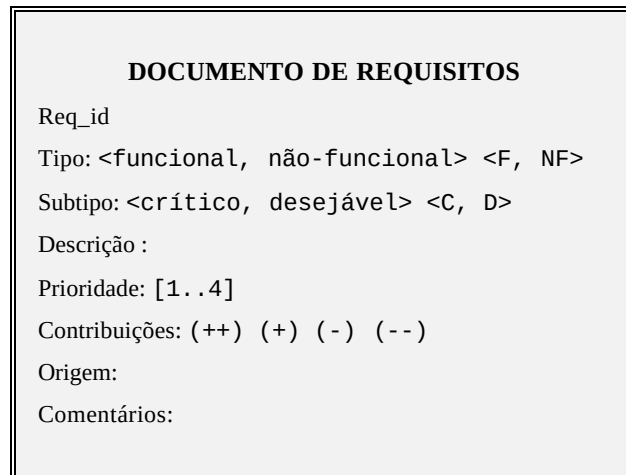


Figura 5.7 - Modelo do Documento de Requisitos

Uma das atividades críticas durante a especificação dos requisitos não-funcionais é o tratamento de conflito entre eles. Esse processo é fundamental para a seleção de COTS pois fornece subsídios para que a equipe de seleção determine quais requisitos são mais críticos para serem atendidos do que outros. A própria natureza dos requisitos não-funcionais sugere que eles estão sujeitos a inúmeros conflitos entre si [CHU00]. Um exemplo clássico é o conflito entre segurança e performance. Suponha que um sistema requer uma política de segurança muito forte, na qual devem ser usados vários mecanismos para segurança da informação, tais como: criptografia e validação de acesso com regras de elegibilidade. Por

² Essa notação é adaptada do NFR Framework [CHU99]

outro lado, o uso desses mecanismos aumenta o tempo de resposta, ocasionando um comprometimento da performance geral da aplicação. Assim os *stakeholders* devem ter em mente que é bastante comum acontecer tais conflitos e então avaliar qual requisito é mais crítico para a satisfação do seu problema. O Método CRE fornece regras de situação para tratar conflitos entre requisitos. A primeira trata do caso de conflito forte entre dois requisitos.

```
If Confl_Forte[Req1,Req2] and Req1_Prio > Req2_Prio  
Then Atender Req1  
Else If Confl_Forte[Req1,Req2] and Req2_Prio > Req1_Prio  
Then Atender Req2
```

Essa outra regra trata do conflito fraco entre dois requisitos. Nesse caso, é aconselhável analisar a situação mais detalhadamente para tomar a decisão mais coerente diante das informações disponíveis. Nesse sentido, a ação mais provável é comprometer o atendimento do requisito de mais baixa prioridade para atender o requisito de maior prioridade.

```
If Confl_Fraco[Req1,Req2] and Req1_Prio > Req2_Prio  
Then Analisar Situação e Comprometer Req2  
Else If Confl_Fraco[Req1,Req2] and Req2_Prio > Req1_Prio  
Then Analisar Situação e Comprometer Req1
```

Em geral, o processo de resolução de conflitos entre requisitos é dependente de cada situação em particular. Nesse contexto, é importante salientar que o NFR *Framework* auxilia a descoberta de possíveis conflitos entre requisitos, uma vez que ele explicita os relacionamentos entre os requisitos não-funcionais durante todo o seu processo de decomposição. Entretanto, essa abordagem não fornece diretrizes para resolver tais conflitos. Dessa forma, é aconselhável que a equipe de seleção tenha alguma experiência prévia sobre resolução de conflitos obtida em seleções anteriores ou pelo menos possua um bom conhecimento sobre o domínio. Outra atividade importante dessa fase é obter informações mais detalhadas sobre os produtos. Em particular, tais informações serão

usadas para avaliar a conformidade do produto com os aspectos descritos no critério de seleção. Algumas possíveis fontes de informação são: documentações, especificações, manuais de uso e de configuração, outras organizações que utilizam o produto, etc. O método CRE fornece um *checklist* para guiar o processo de aquisição de informações sobre os produtos. Esse *checklist* é dividido em duas categorias básicas: aspectos do produto e aspectos do vendedor. A Tabela 5.2 mostra um exemplo genérico de *checklist*, no qual geralmente a equipe de seleção precisa adaptá-lo para suas necessidades específicas.

Checklist	
<i>Aspectos do Produto</i>	<i>Aspectos do vendedor</i>
✓ Custos associados	✓ Maturidade
✓ Conformidade c/ normas de qualidade	✓ Período de entrega
✓ Capacidades	✓ Estabilidade
✓ Benefícios	✓ Treinamento
✓ Restrições	✓ Reputação
✓ Controle de versões	✓ Qualidade e custo do suporte

Tabela 5.2 - *Checklist* de Descrição do Produto

Um aspecto chave do *Checklist* é avaliar a conformidade dos produtos em relação às normas de qualidade. A ISO 9126 [ISO9126], por exemplo, define um conjunto de atributos que um software precisa ter para que sua qualidade seja descrita e avaliada. Já a ISO 12119 [ISO12119] é específica para pacotes de software, ela estabelece alguns requisitos de qualidade para pacotes de software e fornece instruções de como testar um produto em relação aos requisitos preestabelecidos. A ISO 12119 fornece informações importantes sobre aspectos de qualidade que os produtos devem apresentar para que eles estejam em conformidade com esta norma, tais como:

- Descrição explícita sobre as funcionalidades disponíveis;
- Descrição sobre aspectos de usabilidade, eficiência, confiabilidade, manutenibilidade e portabilidade;
- Presença de documentação detalhada;

- Declaração sobre fornecimento de suporte e manutenção por parte do vendedor.

A partir das informações obtidas com a utilização do *checklist*, é possível elaborar uma versão mais completa e detalhada da lista de produtos candidatos (ver Figura 5.8). Nesse estágio, deve ser feito um novo escore de conformidade entre cada produto candidato e os requisitos elicitados durante a fase de descrição. Além disso, os outros escores obtidos anteriormente, relativos aos requisitos críticos, podem ser reavaliados uma vez que o CRE é baseado na iteração contínua entre as suas fases. A análise dos dados obtidos no *checklist* também deve ser incluída nessa nova versão da lista.

LISTA DE PRODUTOS CANDIDATOS ATUALIZADA

PRODUTO 1

Nome:

Descrição geral:

Informações importantes:

Escore de conformidade

Req1 –

Req2 –

...

ReqN –

ReqN+1 –

ReqN+2 –

...

Análise do *checklist*

<i>Aspectos do produto</i>	<i>Aspectos do vendedor</i>
custos associados	maturidade
comentários -	comentários -
capacidades	estabilidade
comentários -	comentários -
...	...

Comentários finais:

Figura 5.8 - Modelo para lista de produtos candidatos atualizada

Obs: as informações dentro do retângulo pontilhado são obtidas nessa fase

Cada processo de seleção deve considerar quais aspectos são relevantes para seu domínio em particular que devem ser analisados. Desse modo, é aconselhável que a equipe de seleção avalie em detalhes somente os pontos do *checklist* que forem considerados importantes para auxiliar o processo de seleção. Vale ressaltar que em alguns casos, nem todos os aspectos poderão ser examinados pela própria falta de informação disponível. Em geral, vários produtos podem ser eliminados da lista de candidatos atualizada depois que eles são avaliados em relação ao critério de seleção. Os produtos que continuam, são priorizados levando-se em consideração principalmente os requisitos não-funcionais pois eles ajudam a discriminar entre produtos semelhantes que já atendem satisfatoriamente aos requisitos críticos. Normalmente, no desenvolvimento de sistemas baseados em COTS, os requisitos não-funcionais que descrevem restrições de arquitetura são críticos e necessitam de atenção especial [DUK99].

A seguir é mostrado o *template para descrição dos produtos*, que fornece diretrizes para serem seguidas durante a fase de descrição. Ele descreve que o resultado final dessa fase é uma lista atualizada dos produtos. Para obter esse resultado, é necessário elicitar e especificar em detalhes os requisitos desejáveis. Nesse contexto, é dado um destaque especial aos requisitos não-funcionais. Além disso, é usado um *checklist* para obter informações detalhadas sobre os produtos. Uma importante atividade descrita nesse *template* é o mecanismo de *feedback* entre a descrição dos produtos e a elicitação dos requisitos. Até esse estágio, geralmente a lista de produtos candidatos possui várias alternativas aparentemente semelhantes. Portanto, não é conveniente tomar uma decisão somente com base na lista de produtos e no critério de seleção, é necessário realizar uma avaliação mais criteriosa dos benefícios de cada alternativa. Entretanto, avaliar e analisar as características relevantes desses produtos requer um considerável intervalo de tempo, quase sempre maior do que a organização tem disponível [KON95]. Assim, o Método CRE recomenda que somente os candidatos mais promissores passem para a próxima fase, que é o processo de avaliação.

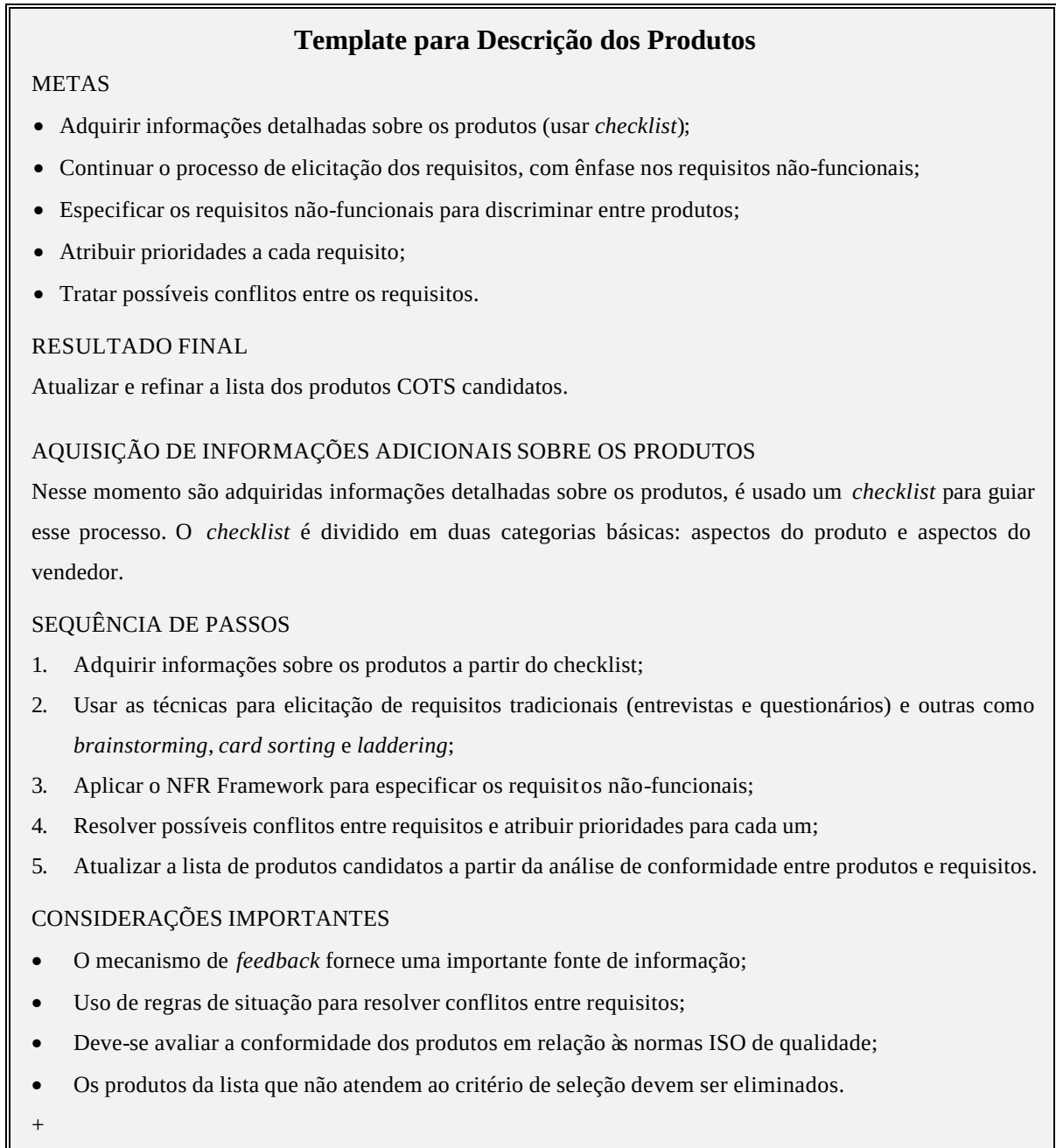


Figura 5.9 – Template para descrição dos produtos

5.2.3 Avaliação

O principal objetivo dessa fase é avaliar a adequabilidade de cada produto em relação ao critério de seleção. Nessa fase, é aconselhável que todas as informações relevantes já tenham sido obtidas e estejam disponíveis, porém nem sempre essa situação é possível. Assim, nesses casos é necessário obter os dados considerados fundamentais para o processo de tomada de decisão que estão faltando. Em geral, custo é um aspecto fundamental durante o desenvolvimento de qualquer sistema de software. Em particular, para desenvolvimento baseado em COTS, o custo é uma restrição crítica já que essa forma de desenvolvimento se baseia essencialmente na promessa de redução de custos e tempo de desenvolvimento. Além disso, o custo exerce grande impacto na realização de outros requisitos não-funcionais, ou seja, existe um *tradeoff* que precisa ser considerado. Num contexto geral, pode-se considerar que o objetivo de qualquer desenvolvimento de software é maximizar critérios relacionados aos requisitos funcionais e não-funcionais como performance e usabilidade; enquanto o critério custo deve ser sempre minimizado [KAR98].

Nesse sentido, é importante ressaltar que o custo de DBC geralmente não é somente o preço de aquisição do produto, também estão associados custos com adaptação, integração, treinamento de pessoal, aquisição de produtos complementares, entre outros. O Método CRE sugere o uso de modelos de custo para estimar o custo associado de cada produto COTS e assim fornecer mais subsídios para avaliar as alternativas. O modelo COCOTS (CONstructive COTS) [BOE98], por exemplo, é um avanço do modelo econômico COCOMO [BOE81] e fornece uma abordagem detalhada para estimar os custos associados para desenvolver sistemas baseados em COTS. Esse modelo é dividido em quatro sub-modelos para tratar as principais fontes de custos de DBC que estão presentes em cada fase do processo de DBC (seleção, adaptação, integração e manutenção). Em particular, o método CRE sugere que a estimativa de custo da fase de seleção seja realizada assim que as metas forem definidas, ou seja, no início da fase de identificação. Por outro lado, é mais apropriado realizar a avaliação de custos das fases seguintes nesse momento pois é necessário ter conhecimento dos requisitos dos *stakeholders* além de informações sobre os produtos. O modelo COCOTS descreve atributos que devem ser analisados em cada sub-

modelo e fornece escalas de importância para esses atributos. O processo de avaliação de custos é feito usando métricas para estimar o esforço necessário para realizar cada etapa do processo DBC. Outra abordagem é fornecida por Ellis [ELI95], ele provê uma solução parcial para estimar o custo de sistemas baseados em COTS pois é estimado somente o esforço de adaptação e integração dos produtos. Vale salientar que dependendo da complexidade do sistema pode-se decidir usar ou não esses modelos de custos.

Em geral, o processo de tomada de decisão é complexo e exige um bom entendimento das capacidades de cada produto candidato. Nesse sentido, é aconselhável que a equipe de seleção tenha um contato prévio com os produtos que estão sendo avaliados. O método CRE fornece duas estratégias para avaliar os benefícios de cada alternativa a fim de realizar o processo de tomada de decisão. A abordagem mais simples utiliza a técnica WSM [CLE97], vista na seção 3.4.2. Essa técnica fornece a seguinte fórmula para obter a pontuação de cada produto em relação à sua conformidade com os critérios examinados:

$$Pont_Produto_a = \sum_{j=1}^n (peso * score_{aj})$$

onde: a representa uma alternativa e n o número de critérios

O peso e o score dessa fórmula são na verdade a prioridade e o score de conformidade descritos nas fases de identificação e descrição. A Tabela 5.3 mostra os valores desses atributos³. O resultado final da aplicação dessa fórmula em todas alternativas é um *ranking* dos produtos candidatos em relação às características do critério de seleção que foram avaliadas. Vale ressaltar que utilização dos produtos pela equipe de seleção permite que os escores de conformidade sejam atribuídos com maior confiabilidade do que se forem usados somente manuais e documentação para realizar essa atividade. Em muitos casos os vendedores fornecem uma versão *freeware* dos produtos que pode ser usada nesse momento.

³ Esses valores atribuídos foram baseados nos trabalhos de Karlsson [KAR98] e na técnica WSM.

<i>Conformidade</i>	<i>Escore</i>	<i>Prioridade</i>	<i>Peso</i>
Não atende o requisito	0	Baixa (sem importância)	1
Atende com sérias restrições	1	Média (baixa importância)	2
Atende parcialmente	2	Alta (importante)	3
Atende	3	Muito alta (essencial e imperativo)	4

Tabela 5.3 - Valores dos Atributos Peso e Escore

O método WSM é adequado para usar em processos de tomada de decisão simples, onde existe um pequeno número de critérios. Entretanto, é importante lembrar que existem várias restrições a esse método, tais limitações estão descritas na seção 3.4.2. A outra estratégia para assistir o processo de tomada de decisão é a técnica AHP [SAA90], também já descrita anteriormente na seção 3.4.3. A utilização dessa técnica é mais complexa que a WSM, porém os resultados obtidos são mais consistentes e confiáveis pois ela permite que a equipe de seleção analise os resultados sob várias perspectivas a partir dos cenários que são montados com a técnica AHP, como já foi constatado em [KON95].

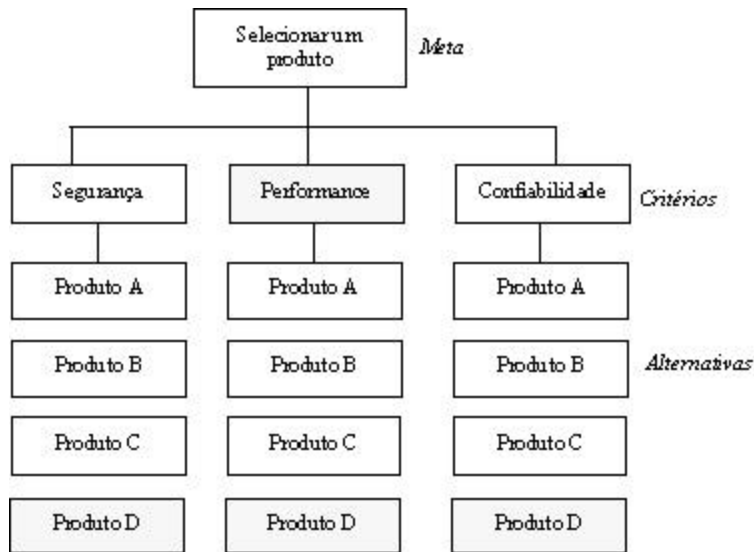


Figura 5.10 - Árvore Hierárquica usando a Técnica AHP

O primeiro passo para usar a técnica AHP é decompor as informações numa hierarquia de metas, critérios e alternativas, ver Figura 5.10. Em seguida, é determinada a importância

relativa de cada critério, onde são usadas comparações aos pares para avaliar a importância de um critério em relação ao outro. A Tabela 5.4 mostra uma escala de importância proposta por Saaty [SAA90]. A partir desses valores preestabelecidos, pode-se comparar aos pares a importância de um critério em relação a outro. Nesse processo, os requisitos não-funcionais são particularmente importantes pois é preciso avaliar quantitativamente suas prioridades para obter um processo de decisão consistente.

<i>Intensidade de Importância</i>	<i>Definição</i>
1	Igual
3	Moderada
5	Forte
7	Muito forte
9	Extrema
2, 4, 6, 8	Valores intermediários

Tabela 5.4 - Escala de importância usada nas comparações da técnica AHP

A Tabela 5.5 mostra um exemplo de comparação entre critérios usando uma matriz de dados. Nesse exemplo, considera-se que os critérios relevantes durante o processo de tomada de decisão são somente: segurança, performance e confiabilidade. O próximo passo é usar o método de autovetores para obter o ranking relativo de cada critério. A Tabela 5.6 mostra um possível resultado do ranking entre esses critérios, onde o somatório desses valores deve ser igual a 1.

	Segurança	Performance	Confiabilidade
Segurança	1/1	1/2	3/1
Performance	2/1	1/1	4/1
Confiabilidade	1/3	1/4	1/1

Tabela 5.5 - Matriz de comparação entre critérios

Critério	Autovetor	Ranking
Segurança	.3196	} $S=1$ 2°
Performance	.5584	
Confiabilidade	.1220	

Tabela 5.6 - Ranking entre os critérios

Em seguida, todas alternativas devem ser comparadas aos pares em relação à sua conformidade com cada critério, ver Tabela 5.7. Finalmente, pode ser obtida a árvore hierárquica mostrando o ranking dos produtos em relação a cada critério, ver Figura 5.11.

Segurança				
	Produto A	Produto B	Produto C	Produto D
Produto A	1/1	1/4	4/1	1/6
Produto B	4/1	1/1	4/1	1/4
Produto C	1/4	1/4	1/1	1/5
Produto D	6/1	4/1	5/1	1/1

Tabela 5.7 - Matriz de comparação entre produtos em relação ao critério segurança

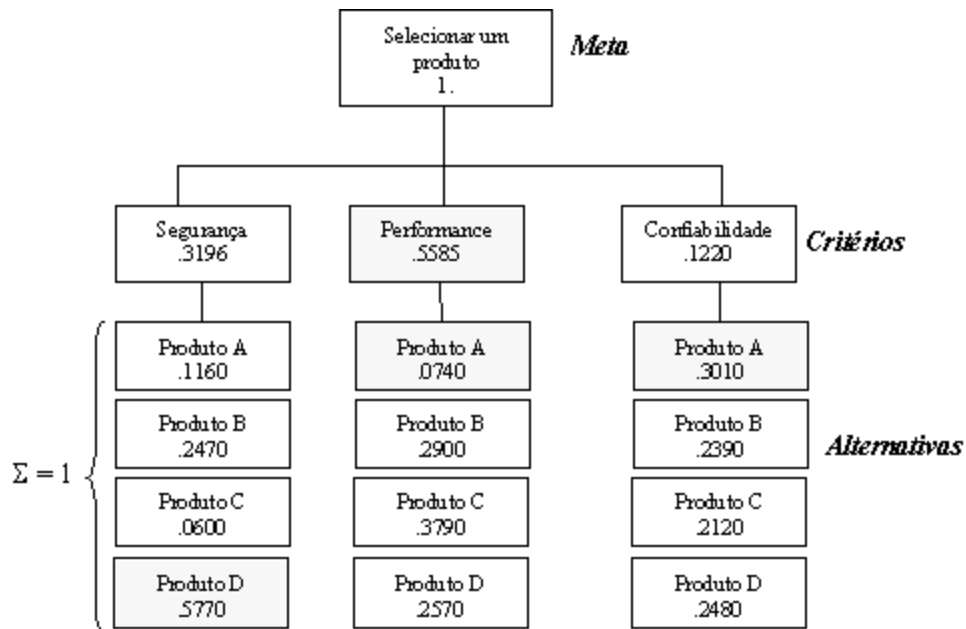


Figura 5.11 - Árvore indicando o ranking dos produtos em relação aos critérios

Multiplicando-se a matriz das alternativas pelo vetor dos critérios, obtém-se o *ranking* entre os produtos, ver Tabela 5.8. Dessa forma, a técnica AHP fornece uma análise sob várias

perspectivas para determinar os benefícios de cada alternativa. O resultado final da fase de avaliação é um *ranking* dos produtos que estão na lista de candidatos.

<i>Alternativas</i>	<i>Autovetor</i>	<i>Ranking</i>
Produto A	.3060	2°
Produto B	.2720	3°
Produto C	.0940	4°
Produto D	.3280	1°

Tabela 5.8 - Ranking entre os produtos candidatos

Como foi mostrado, a aplicação dessa técnica requer várias operações entre matrizes que podem ser assistidas pela ferramenta ExpertChoice⁴. No final do processo obtêm-se um *ranking* dos produtos em relação aos critérios avaliados. Uma das principais vantagens da técnica AHP é que ela tem sido utilizada com sucesso tanto em processos de priorização de requisitos [KAR96] como para seleção de produtos [KON95] [NCU00a]. Além disso, essa técnica pode ser usada em processos decisórios extremamente complexos, onde os critérios precisam ser decompostos em vários sub-critérios hierarquicamente relacionados. Essa característica é compatível com os resultados obtidos do NFR *Framework* que decompõe os requisitos não-funcionais em vários sub-níveis.

É importante salientar que as técnicas para tomada de decisão apresentadas devem ser usadas como ferramentas de suporte e não como uma solução. Assim é fundamental que a equipe de seleção faça uma análise crítica dos resultados obtidos usando tais técnicas. O resultado final dessa fase um *ranking* dos produtos COTS candidatos em relação ao critério de seleção. A seguir é fornecido o *template para avaliação dos produtos*.

⁴ Existe uma versão *freeware* dessa ferramenta disponível no site <http://www.expertchoice.com>

Template para Avaliação dos Produtos

METAS

- Avaliar adequabilidade de cada produto em relação ao critério de seleção;
- Usar modelos de custo para estimar o custo associado de cada alternativa;
- Usar técnicas de tomada de decisão.

RESULTADO FINAL

Obter ranking dos produtos candidatos, a partir das técnicas de tomada de decisão.

PROCESSO DE AVALIAÇÃO DE CUSTOS

- Critério crítico para qualquer sistema;
- Impacto em outros requisitos não-funcionais;
- Modelo de custo COCOTS é apropriado para realizar essa tarefa.

PROCESSO DE TOMADA DE DECISÃO

- Requer esforço e tempo consideráveis;
- Equipe de seleção precisa ter amplo conhecimento do domínio;
- Pode-se usar, entre outras, as seguintes técnicas:
 - WSM (fácil uso, adequada para processos de seleção simples)
 - AHP (uso complexo mas fornece resultados mais detalhados)

SEQUÊNCIA DE PASSOS

1. Adquirir informações importantes que estão faltando;
2. Avaliar o custo associado de cada alternativa;
3. Usar técnicas para tomada de decisão;
4. Obter ranking dos produtos.

CONSIDERAÇÕES IMPORTANTES

- Fazer uma análise criteriosa dos resultados obtidos com as técnicas de tomada de decisão;
- Documentar todos os passos realizados durante essa fase.

Figura 5.12 - *Template* para avaliação dos produtos

5.2.4 Aceitação

Um dos principais objetivos da fase de aceitação é realizar sessões de demonstração dos produtos com os *stakeholders*. Geralmente, os vendedores disponibilizam versões demonstrativas (*freeware*) de seus produtos que podem ser usadas durante os testes de aceitação. Em particular, a norma ISO 12119 fornece instruções de como realizar tais testes. É recomendável que a equipe de seleção instale os produtos candidatos em ambiente operacional para simular sua execução numa situação real. Durante esse estágio, alguns dos aspectos de qualidade a serem avaliados são:

- Usabilidade;
- Integrabilidade;
- Interoperabilidade;
- Compatibilidade;
- Performance.

Os testes de aceitação permitem que a equipe de seleção observe a adequabilidade de cada produto num ambiente mais realístico do que analisando simples descrições encontradas em documentações e manuais. Além de permitir que os *stakeholders* avaliem os produtos na prática. Um aspecto importante dessa fase é verificar possíveis conflitos entre o COTS que será escolhido e o restante do sistema. Dessa forma, é possível verificar a necessidade de futuras adaptações entre o produto e o sistema, já podendo especificar alguns *scripts* ou *wrappers* (ver Seção 2.4.3). Um outro aspecto fundamental para ser examinado são as restrições de arquitetura que os produtos devem estar em conformidade. Em particular, essa atividade está diretamente relacionada com a verificação de que os requisitos de qualidade estão sendo atendidos. Existem situações em que o conflito é tão crítico que pode necessitar de um complexo e dispendioso processo de adaptação do produto, acarretando a inviabilidade de sua integração. Nesses casos pode-se optar pelo próximo candidato no *ranking*.

É importante salientar que normalmente os testes de aceitação são demorados e portanto nem sempre é recomendável realizar testes em todos os produtos candidatos. A equipe de seleção deve levantar quais os custos necessários e considerar quais são os reais benefícios para realizar testes de aceitação em todos os produtos candidatos. Uma alternativa para casos com limitações de recursos é realizar testes somente com alguns produtos, aqueles candidatos mais fortes ou onde existe um alto grau de incerteza sobre os vendedores e as características do produto. O *template para aceitação dos produtos* fornece um guia para realização dos testes de aceitação.

Durante a fase de aceitação acontece a validação final dos requisitos. Nesse momento pode ser verificado quais requisitos são atendidos pelo produto que será selecionado e quais não puderam ser completamente satisfeitos. É importante salientar que existem casos em que nenhum produto disponível satisfaz todos os requisitos críticos dos *stakeholders*. Assim deve haver um acordo entre a equipe de seleção e os *stakeholders* para seguir uma das duas alternativas:

- Diminuir o número e detalhamento das restrições impostas;
- Implementar esses componentes de acordo com a especificação dos requisitos.

Uma importante atividade dessa fase é a verificação dos aspectos legais e contratuais que não puderam ser obtidos e avaliados com o *checklist* da fase de descrição. A equipe de seleção precisa verificar com cuidado os aspectos contratuais de cada alternativa, a fim de realizar julgamentos coerentes sobre os riscos e dependências que eles estão dispostos a assumir quando o produto for adquirido. Uma licença entre o vendedor e a organização que está comprando o produto COTS precisa especificar pelo menos alguns aspectos chave, por exemplo [CHA98]:

- Direitos que o vendedor autoriza (concessão de uso);
- Formas de pagamento;
- Riscos e responsabilidade que cada parte assume;

- Suporte, manutenção e garantias que o vendedor fornece;
- Confidencialidade do produto licenciado.

Geralmente, o vendedor não dá direito de acesso ao código-fonte pois ele faz parte dos seus direitos de propriedade intelectual. Entretanto, segundo dados do SEI (*Software Engineering Institute*) [SEI00], na prática, cerca de 30% dos produtos COTS adquiridos como caixa-preta necessitam de algumas mudanças no seu código durante o processo de adaptação. Nesses casos, é necessário negociar com o vendedor formas de obter acesso a parte do código-interno ou então sugerir que ele realize as alterações necessárias.

Em [CHA98] é fornecida uma análise detalhada de todos aspectos que deveriam estar referenciados numa licença de aquisição de produtos COTS. Uma estratégia proposta é que os vendedores forneçam uma garantia durante um curto intervalo de tempo e disponibilizem um suporte técnico pago depois que a garantia se expirar. Nesse período inicial, a organização certamente irá integrar o produto no seu sistema, dando a oportunidade do vendedor resolver possíveis defeitos que apareçam como resultado da integração. Depois que o período de garantia termina, o vendedor pode fornecer suporte e manutenção pagos. As potenciais vantagens dessa estratégia são que o comprador é motivado a integrar o produto rapidamente, além de ter garantia durante o período mais crítico que é o processo de adaptação e integração. Sob o ponto de vista do vendedor, ele é incentivado a eliminar defeitos e melhorar a qualidade do produto nas versões futuras. É importante salientar que na prática, muitos fornecedores já cobram taxa de manutenção no momento do licenciamento.

Template para Aceitação dos Produtos

METAS

- Realizar testes de aceitação dos produtos num ambiente operacional;
- Realizar validação final dos requisitos;
- Verificar conflitos entre o produto e o sistema;
- Analisar aspectos legais e contratuais.

RESULTADO FINAL

Selecionar o produto mais adequado para integrar o sistema.

GUIA PARA REALIZAÇÃO DOS TESTES DE ACEITAÇÃO

Deve incluir informações detalhadas sobre as seguintes questões:

- Objetivo do teste;
- Funções a serem testadas;
- Ambiente de testes e outras condições (detalhes de configuração);
- Dados de testes;
- Procedimento;
- Comportamento esperado do sistema.

RECOMENDAÇÕES DA ISO 12119 PARA REALIZAR OS TESTES DE ACEITAÇÃO

- É importante que todos os pacotes estejam disponíveis, assim como os documentos de requisitos identificados na descrição de produto;
- Se o treinamento for mencionado na descrição do produto, o responsável pelo teste deve ter acesso ao material de treinamento;
- A descrição do produto, a documentação de usuário e os programas devem estar em conformidade com os requisitos dos stakeholders;
- Os detalhes que não puderem ser testados devem ser mencionados nos relatórios de teste, assim como os motivos para não testá-los;
- Os guias de teste devem ser construídos sistematicamente, eles devem cobrir todas as funções descritas na descrição do produto e a documentação de usuário;
- O relatório de teste deve conter o plano de teste executado, todos os resultados associados com os guias, incluindo todas as falhas que ocorreram durante o teste, além da identificação do pessoal envolvido.
- Os aspectos relacionados com licenciamento devem ser analisados.

Continuação
• A estrutura do relatório deve ser a seguinte: – Identificação do produto, – Sistemas usados para o teste, – Documentos usados (com suas identificações), – Resultados dos testes de descrição, – Documentação e programas, – Lista das não-conformidades aos requisitos, – Data de encerramento dos testes. CONSIDERAÇÕES IMPORTANTES • Testes de aceitação fornecem informações sobre adequabilidade do produto ao sistema; • Durante a validação final é possível verificar quais requisitos não foram atendidos pelo produto; • Se possível, garantir que um representante do vendedor esteja presente durante os testes de aceitação; • Os aspectos legais precisam ser examinados cuidadosamente antes do produto ser adquirido.

Figura 5.13 – *Template* para aceitação dos produtos

5.3 CONSIDERAÇÕES FINAIS

Normalmente, as organizações realizam o processo de seleção de forma ad-hoc, pois ele é direcionado por critérios subjetivos como indicação de amigos e custos de aquisição mais baixos. Além disso, é destinado um curto intervalo de tempo para realizar a seleção já que essa atividade não é considerada prioridade para a organização que deseja ter resultados rápidos. Nesse sentido, o objetivo principal é integrar o COTS o mais rápido possível ao sistema. Por outro lado, como foi discutido na Seção 3.2, é necessário ter um processo de seleção bem definido a fim de maximizar os benefícios da abordagem de desenvolvimento baseado em COTS. O método CRE fornece uma abordagem detalhada para sistematizar o processo de seleção de produtos COTS, onde são propostos diversos métodos e técnicas úteis para cada fase do CRE. Nesse contexto, a proposta do método é descrever um conjunto de diretrizes para sistematizar o processo de seleção de produtos COTS.

O método CRE é baseado no processo iterativo de aquisição de requisitos e seleção/rejeição de produtos. Ele é orientado a metas e cada fase possui um *template* cujo objetivo é facilitar

a realização das atividades propostas. A principal contribuição do CRE em relação aos outros métodos de seleção de COTS existentes na literatura é a ênfase no tratamento dos requisitos não-funcionais como forma de melhorar o grau de adequabilidade do produto a ser escolhido em relação às necessidades dos *stakeholders*, ou seja, garantir que o produto satisfaça ao máximo os requisitos especificados.

O CRE possui quatro fases que interagem entre si, são elas: identificação, descrição, avaliação e aceitação. Como o próprio nome sugere, a fase de identificação consiste basicamente da localização e identificação dos possíveis COTS candidatos. A fase seguinte fornece direcionamentos de como detalhar o critério de seleção, com ênfase especial nos requisitos de qualidade. Além disso, deve ser realizada uma descrição detalhada de cada produto da lista de candidatos. Nessa etapa, ocorre o mecanismo de *feedback* que fornece uma troca importante de informações entre a descrição dos produtos e a especificação dos requisitos. Na fase de avaliação são fornecidas técnicas para tomada de decisão, cujo objetivo é obter um *ranking* dos produtos candidatos. Finalmente, a fase de aceitação descreve os testes de aceitação que deverão ser realizados com os candidatos mais promissores. A fim de obter todos os benefícios do método CRE, aconselha-se documentar e disponibilizar todas as informações adquiridas e decisões tomadas ao longo de todo o processo de seleção. Em particular, tais informações poderão ser necessárias durante as fases de adaptação e integração.

Uma das possíveis limitações do método CRE é o tempo necessário para realizar todas as fases propostas, que muitas vezes pode ser maior do que a empresa tem disponível. Além disso, o uso de algumas técnicas, como NFR *Framework* e AHP, pode necessitar de um treinamento prévio da equipe de seleção. O próximo capítulo fornece um estudo de caso real que utiliza o método CRE para selecionar ambientes de desenvolvimento em Java. Assim poderá ser visto como são usadas as fases do método proposto num contexto prático.

Capítulo 6

Estudo de Caso

Este capítulo fornece um estudo de caso para mostrar a utilização do método CRE num projeto de seleção de ferramentas para desenvolvimento em Java. Inicialmente, são descritos o objetivo e a metodologia aplicada durante a realização do estudo. Em seguida, é mostrado como cada fase do método CRE foi usada para conduzir essa seleção. Finalmente, os resultados desse estudo são descritos, assim como as principais lições aprendidas.

6.1 OBJETIVOS

O principal objetivo desse capítulo é examinar a hipótese H4 que é a seguinte:

H4 O método proposto é simples de ser usado e proporciona assistência nas tarefas de avaliação e seleção de produtos COTS.

Esse capítulo apresenta um estudo de caso realizado com o intuito de mostrar a utilização do método CRE num processo de seleção de ferramentas para desenvolvimento de aplicações em Java. Esse estudo foi realizado no ecossistema acadêmico ligado ao Centro de Informática da Universidade Federal de Pernambuco. Dessa forma, a equipe de seleção contou com a participação de engenheiros de software e pesquisadores desse ambiente cooperativo entre empresas e universidade.

Atualmente, existe uma grande variedade no mercado de software dessas ferramentas que são conhecidas como IDE (*Integrated Development Environment*). Nesse estudo foram avaliadas várias alternativas de IDEs disponíveis no mercado, onde foi aplicado o método CRE para assistir esse processo de seleção. A proposta do método é fornecer uma abordagem bem definida e sistemática que enfatiza a avaliação dos requisitos não-funcionais como forma de obter um processo de seleção de produtos COTS consistente com as necessidades dos *stakeholders*. A metodologia aplicada para guiar esse estudo foi baseada nas indicações descritas nos *templates* fornecidos pelo método CRE. Nesse contexto, foram seguidos os passos indicados em cada *template* a fim de assistir cada fase do processo de seleção. Em particular, nas próximas seções será examinada a aplicabilidade de cada fase proposta pelo método para assistir a aquisição dessa ferramenta para desenvolvimento em Java.

6.2 UTILIZAÇÃO DO MÉTODO CRE

Este processo de seleção foi realizado seguindo passo a passo as fases do método CRE, dessa forma as próximas seções mostram como este estudo foi conduzido em relação às metas propostas por cada fase do método. Em particular, é fornecida uma descrição detalhada de como foram realizadas as atividades relativas a cada uma das fases do CRE.

6.2.1 *Identificação*

A primeira atividade dessa fase é elaborar o critério de seleção inicial a fim de iniciar o processo de localização dos potenciais candidatos. Dessa forma, a meta básica desse processo de seleção é adquirir uma ferramenta que forneça um ambiente integrado para o desenvolvimento de aplicações em Java. O processo de localização iniciou com uma pesquisa na Internet dos diversos IDEs Java existentes no mercado, foram usados *sites* especializados como JavaWorld¹, PC Magazine² e Java Developer Journal³ para realizar essa pesquisa. A partir das informações obtidas nesses *sites*, foram localizadas 30 ferramentas como sendo possíveis candidatas. Em particular, foi suficiente realizar uma busca somente na Internet já que existe uma extensa variedade de informações sobre tais ferramentas disponível nesse meio. Vale salientar que essas ferramentas localizadas ainda não fazem parte da lista de produtos candidatos pois é necessário que as mesmas estejam em conformidade com os requisitos críticos dos *stakeholders*.

O *template para identificação dos produtos* (ver página 76) mostra que uma das principais metas dessa fase é elicitare os requisitos críticos que as ferramentas precisam atender. Para elicitare tais requisitos, foram realizadas entrevistas com os *stakeholders* que forneceram informações sobre os requisitos que eram considerados críticos de acordo com suas necessidades. Os *stakeholders* que participaram desse estudo foram analistas de negócio,

¹ <http://www.javaworld.com>

² <http://www.pcmagazine.com>

³ <http://www.sys-con.com/java/>

engenheiros de software e gerentes de projeto. O critério de seleção preliminar que direcionou o processo de eliminação inicial dos IDEs Java, consistiu de seis requisitos considerados críticos (vide Tabela 6.1). Dentre as ferramentas candidatas, a grande maioria delas era proveniente de pequenos fornecedores que não possuem estabilidade e reputação no mercado de software. Nesse sentido, os *stakeholders* consideraram que esse era um importante critério para eliminação de ferramentas pois a boa reputação do vendedor no mercado pode ser um indicativo positivo da maturidade e qualidade geral da ferramenta.

Requisitos Críticos	
√	Suportar desenvolvimento de aplicações Web
√	Suportar rastreamento e controle de versões
√	Fornecer <i>Wizards</i> para tecnologias Java
√	Possuir ferramenta de <i>debug</i>
√	Possuir editor de código
√	Fornecedor ter estabilidade no mercado

Tabela 6.1 - Requisitos críticos que os IDEs Java devem atender

Vale salientar que por causa do grande número de candidatos que claramente não atendiam ao critério de seleção inicial, a eliminação dos mesmos foi realizada sem ter a necessidade de estipular uma pontuação mínima necessária para que os produtos continuassem na lista de candidatos. O processo de eliminação foi feito da seguinte forma: as ferramentas que não atendiam a todos os requisitos mostrados na Tabela 6.1 foram logo eliminadas. Esse fato confirma uma das considerações levantadas no *template para identificação dos produtos*, enfatizando que nessa etapa o processo de tomada de decisão deve ser simples e direto. Esse *template* sugere ainda que sejam aplicados questionários com os fornecedores para adquirir mais informações sobre os produtos. No entanto, essa tarefa não foi necessária pois já existia um bom repositório de informações disponível nos *sites* desses produtos. A Tabela 6.3 mostra a lista das ferramentas candidatas.

Lista inicial de ferramentas candidatas				
<i>Nome da ferramenta</i>	<i>Fornecedor</i>	<i>Versão</i>	<i>100% Puro Java</i>	<i>Site na Web</i>
VisualAge	IBM	3.5	Sim	hppt://www.ibm.com
Jbuilder	Borland	4.0	Sim	hppt://www.borland.com
Jdeveloper	Oracle	3.2	Sim	hppt://www.oracle.com
Forte for Java	Sun	2.0	Sim	hppt://www.sun.com
VisualCafé	Webgain	4.0	Não	hppt://www.webgain.com
PowerJ	Sybase	3.0	Sim	hppt://www.sybase.com
CodeWarrior	Metrowerks	6.0	Sim	hppt://www.metrowerks.com

Tabela 6.2 Lista Inicial de Ferramentas Candidatas

A partir da lista com os produtos candidatos foi verificada a necessidade de obter informações detalhadas sobre as capacidades de tais ferramentas para que o processo de avaliação continuasse. Além disso, a fim de obter critérios para discriminar entre as ferramentas candidatas, foi necessário que os requisitos descritos no critério de seleção inicial fossem refinados, além de adquirir outros requisitos considerados relevantes para efetivar o processo de tomada de decisão. Dessa forma, a próxima seção trata desses assuntos.

6.2.2 Descrição

De acordo com o *template para descrição dos produtos* (ver página 86), uma das principais atividades dessa fase é adquirir informações detalhadas sobre os produtos. Em particular, esse foi o primeiro passo seguido pois a equipe de seleção considerou que uma análise preliminar sobre as capacidades das ferramentas poderia ajudar na especificação dos questionários que em seguida foram aplicados com os *stakeholders* a fim de obter os requisitos e suas prioridades. Nesse contexto, a análise da documentação dessas ferramentas serviu como importante fonte para o mecanismo de *feedback* que consiste de um auxílio para o processo de aquisição de requisitos a partir das informações contidas na descrição dos produtos candidatos. As principais fontes de informação sobre essas ferramentas foram a Internet, além de entrevistas com *stakeholders* que utilizam tais IDEs.

Nesse sentido, o *checklist* (ver Tabela 5.2) orientou o processo de levantamento das informações sobre as capacidades das ferramentas avaliadas. Vale salientar que durante esse processo foi possível eliminar algumas das ferramentas que se mostraram inadequadas. Por exemplo, durante uma entrevista com *stakeholders*, foi observado que a ferramenta VisualCafé possui uma forte tendência a instabilidade que foi relatada por uma equipe de desenvolvimento. Como será visto adiante, esse aspecto foi considerado um requisito não-funcional crítico pelos *stakeholders*. Dessa forma, a lista atualizada das ferramentas candidatas depois desse processo de eliminação é mostrada na Tabela 6.3. Essa lista fornece os requisitos do sistema básicos que essas ferramentas necessitam para operar.

Descrição Geral	JDeveloper	Jbuilder	VisualAge	Forte for Java
Fornecedor	Oracle	Borland	IBM	Sun
Versão	v. 3.2	Enterprise v.4	Enterprise v.3.5	Internet Edition 2.0
Sistema Operacional	- Windows NT 4, 2000	- Windows 98, 2000 ou NT 4 - Linux kernel 2.2.6 - Solaris 7	- Windows 98, 2000 ou NT 4	- Windows NT 4, 2000 - Red Hat Linux 6.2 - Solaris 7.8
Código 100% Java	Sim	Sim	Sim	Sim
Requisitos de Hardware	- CD-ROM drive - RAM 96 MB mínimo 256 MB recomendado - Espaço em disco 280 MB mínimo 500 MB ou mais recomendado - Mouse ou outro dispositivo	- CD-ROM drive - RAM 128 MB recomendado - Espaço em disco 250 MB mínimo - Mouse ou outro dispositivo	- CD-ROM drive - RAM 96 MB mínimo 160 MB recomendado - Espaço em disco 400 MB mínimo 700 MB ou mais recomendado - Mouse ou outro dispositivo	- CD-ROM drive - RAM 128 MB mínimo 512 MB recomendado - Espaço em disco 128 MB mínimo 384 MB ou mais recomendado - Mouse ou outro dispositivo

Tabela 6.3 - Lista atualizada das ferramentas candidatas

Como resultado do levantamento de informações sobre as ferramentas candidatas, a Tabela 6.4 fornece uma descrição dos principais pontos positivos e negativos encontrados em cada ferramenta. Nesse estágio foram considerados dados encontrados na documentação obtida nos *sites* Web dos próprios fornecedores das ferramentas candidatas, em *sites*

especializados sobre a linguagem Java⁴, além de informações obtidas a partir de entrevistas com os *stakeholders* que utilizam esses IDEs.

Ferramentas	Pontos Fortes	Pontos Fracos
JBuilder	• Recebeu diversos prêmios de melhor IDE Java no ano de 2000 • Considerado um dos melhores compiladores do mercado; • Pode ser executado em diversas plataformas com as mesmas interfaces e ferramentas; • Possui interface bastante compreensível; • Suporta bem Enterprise JavaBeans; • Permite alteração no código HTML sem precisar mudar código Java.	• Não fornece bom suporte para JSP • Ferramenta que exige muito da máquina (hardware)
VisualAge	• Eficientes mecanismos de controle de versão e debugger; • Boa interoperabilidade com outras ferramentas; • Forte integração com WebSphere; • Possui bom suporte e documentação disponíveis na Web; • Templates para estender sistemas ERP (SAP R/3) para e-business (só versão Enterprise); • Líder no mercado de tecnologias VoiceXML e WML; • Suporta outras ferramentas de controle de versão como Microsoft Visual SourceSafe.	• Ambiente de desenvolvimento restrito à plataforma Windows • Demora considerável para incluir novas tecnologias • Compilação de JSP deixa a desejar • Curva de aprendizado grande para usuários iniciantes • Ferramenta que exige muito da máquina (hardware)
Forte for Java	• Código inteiramente aberto; • Arquitetura extensível; • Permite desenvolvimento em diversas plataformas; • Eficiente ambiente para edição de código; • Fornece Wizard JDBC-Swing.	• Ferramenta bastante nova no mercado; • Falta suporte à Enterprise JavaBeans e edição de HTML; • A versão Community não possui controle de versão integrado;
Jdeveloper	• Fornece ferramentas gráficas simples e úteis para iniciantes; • Bom suporte para JSP; • Interface semelhante com o JBuilder; • Dispõe de <i>business components</i> que facilita o desenvolvimento de Enterprise JavaBeans;	• Ambiente de desenvolvimento restrito à plataforma Windows; • Boa integração com banco de dados Oracle mas deixa a desejar em outros ambientes; • É necessário adquirir outros produtos Oracle.

Tabela 6.4 - Avaliação preliminar das ferramentas

Segundo o *template para descrição dos produtos*, nessa fase deve ser feito um refinamento dos requisitos descritos no critério de seleção, assim como tratar os requisitos não-

⁴ <http://www.sun.com>, <http://www.javaworld.com>

funcionais. Nesse momento foram realizadas entrevistas individuais com os *stakeholders*, onde foram aplicados questionários com o objetivo de obter informações mais detalhadas sobre os requisitos funcionais e não-funcionais que as ferramentas devem atender. Tais questionários foram elaborados a partir dos dados obtidos durante o processo de levantamento das informações sobre os produtos.

Req_ID	Tipo/ Sub-tipo	Descrição	Origem	Priori- dade (1..4)	Relaciona- mentos
Req_1	F/C	Suportar desenvolvimento de aplicações Web	Eng. Soft.	3	Req_14
Req_2	F/D	Suportar servidor de aplicação	Gerente	2	Req_13 Req_18
Req_3	F/C	Possuir rastreamento e controle de versões	Eng. Soft.	2	Req_18 Req_17
Req_4	F/C	Fornecer <i>Wizards</i> para tecnologias Java	Analista	2	Req_1 Req_14
Req_5	F/C	Possuir ferramenta de Debug	Eng. Soft.	4	Req_18
Req_6	F/C	Possuir editor de código	Eng. Soft.	4	Req_18
Req_7	F/D	Dispor de help on-line	Eng. Soft.	2	Req_12
Req_8	NF/D	Facilidade para troca de <i>releases</i>	Gerente	3	Req_16(+)
Req_9	NF/D	Suportar reuso de componentes	Analista	4	Req_18(+)
Req_10	NF/D	Ter boa performance	Eng. Soft.	4	Req_18(+)
Req_11	NF/C	Prover estabilidade de uso	Analista	4	Req_18(+)
Req_12	NF/C	Ser fácil de usar	Eng. Soft.	4	Req_4 Req_6 Req_7
Req_13	NF/D	Ser adaptável a novas tecnologias	Gerente	4	Req_1 Req_4 Req_12(-)
Req_14	NF/C	Ter boa integrabilidade com outras aplicações	Analista	3	Req_1 Req_3 Req_5
Req_15	NF/D	Prover customização de funcionalidades	Eng. Soft.	2	Req_14(+)
Req_16	NF/D	Ser de fácil manutenção	Analista	1	Req_8(+)
Req_17	NF/D	Prover segurança do controle de versões	Eng. Soft.	2	Req_3 Req_10(-)
Req_18	NF/C	Prover alto nível de produtividade	Analista	4	Req_3 Req_5 Req_6 Req_7 Req_9(+) Req_10(+)
Req_19	NF/C	Fornecedor ter estabilidade no mercado	Analista	4	-

Legenda: F – Funcional NF – Não-Funcional C – Crítico D – Desejável

Tabela 6.5 - Documento de Requisitos

Num primeiro momento, não foram citados quais produtos estavam sendo examinados para não influenciar a opinião dos *stakeholders*, já que em geral cada desenvolvedor tem uma preferência pessoal por determinada ferramenta. Dessa forma, cada *stakeholder* analisou a importância dos critérios, isto é, atribuiu prioridades aos requisitos sem necessariamente avaliar uma ferramenta em particular. Durante essas entrevistas, os *stakeholders* também sugeriram novos requisitos que eles consideravam relevantes e que não foram

discriminados nos questionários. A lista com os requisitos obtidos durante as entrevistas foi condensada num documento que fornece a descrição dos requisitos, sua origem, prioridade e relacionamentos com outros requisitos. Em particular, a origem dos requisitos foi fundamental para realizar o rastreamento dos mesmos ao longo do processo de seleção. Vale salientar que os relacionamentos descritos no documento de requisitos foram obtidos a partir da aplicação do NFR *Framework*, onde uma das principais vantagens dessa abordagem é explicitar relacionamentos entre requisitos que não eram observados inicialmente. Para obter a prioridade de cada requisito foram aplicados questionários com os *stakeholders* envolvidos, onde cada participante atribuiu prioridades aos requisitos e em seguida foi obtida uma média final para cada requisito. Se durante esse processo forem verificadas algumas discrepâncias entre os resultados, devem ser realizadas reuniões de *brainstorming* com os *stakeholders* envolvidos para chegar a um consenso final. A Tabela 6.6 mostra um resumo de como esse levantamento de prioridades foi realizado.

Requisito	Prioridade (1..4)				
	Stakeholder1 (gerente)	Stakeholder2 (analista)	...	StakeholderN (eng. soft)	Média
Req_1	4	4	...	4	4
Req_2	4	3	...	2	3
Req_3	4	4	...	4	4
...	...	...	...	...	...
Req_18	4	4	...	4	4
Req_19	2	2	...	1	2

Tabela 6.6 - Prioridades dos Requisitos

De acordo com o *template para descrição dos produtos*, nessa fase deve ser usado o NFR *Framework* para refinar os requisitos não-funcionais e assim obter informações sobre os possíveis relacionamentos e conflitos existentes entre eles. Segundo Chung [CHU00], o primeiro passo para usar o NFR *Framework* é adquirir conhecimento específico sobre os requisitos não-funcionais e o domínio. Essa primeira etapa foi realizada usando a própria documentação das ferramentas e livros sobre requisitos, enquanto o conhecimento sobre o domínio foi obtido durante as entrevistas com os *stakeholders*. Nesse contexto, muitos dos requisitos especificados precisaram ser refinados para um melhor entendimento do seu significado, diminuir possíveis ambigüidades e assim facilitar a análise do critério de

seleção. Por exemplo, o critério *Possuir editor de código* foi considerado um requisito funcional imperativo segundo os *stakeholders*. Em particular, foi explicitada a importância que as ferramentas possuam um editor poderoso e eficiente, onde estes atributos estão relacionados com aspectos não-funcionais. Dessa forma, foi necessário investigar o significado dos termos para os *stakeholders*. O refinamento desses requisitos não-funcionais permitiu que novas funcionalidades mais detalhadas fossem obtidas em relação ao requisito *Possuir editor de código*, ver Tabela 6.7.

<i>Possuir um poderoso e eficiente editor de código</i>	
√	Localizar texto no código escrito;
√	Realçar palavras-chave no código;
√	Fornecer <i>templates</i> de códigos;
√	Exibir possíveis expressões para completar o código editado;
√	Fornecer auto-formatação do código.

Tabela 6.7 - Refinamento do Requisito “*Possuir editor de código*”

A equipe de seleção decidiu que dentre os requisitos não-funcionais especificados no documento de requisitos, seria interessante refinar cinco deles a fim de facilitar a análise dos mesmos e assim identificar os relacionamentos (interdependências) existentes entre eles. O requisito Req_13 refere-se à usabilidade da ferramenta, esse requisito foi decomposto em Facilidade de aprendizado dos usuários avançados e iniciantes; Acessibilidade para gerenciar projetos e capacidades da ferramenta; e Simplicidade da documentação, interface e instalação. Em seguida, o *softgoal* Facilidade de aprendizado foi refinado em compreensibilidade do código gerado e das capacidades; disponibilidade de help on-line e de suporte na Web; e Claridade da interface de uso e da documentação. A Figura 6.1 mostra esse grafo SIG (*Softgoal Interdependency Graph*).

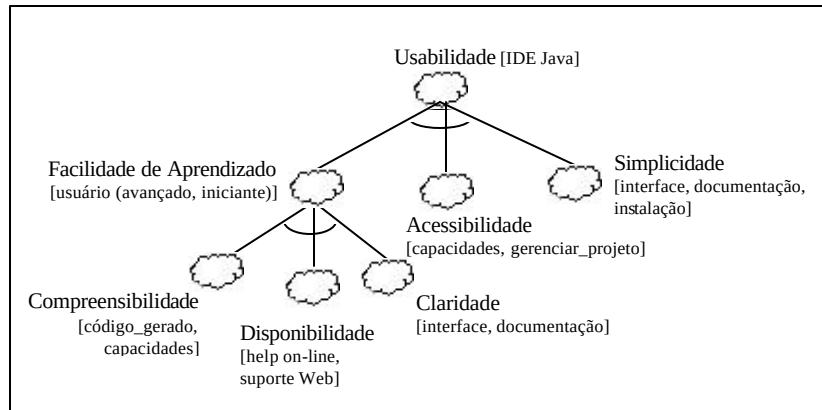


Figura 6.1 - Grafo SIG do requisito *Usabilidade*

O requisito Req_15 relaciona-se com a Integrabilidade da IDE Java, de acordo com as necessidades dos *stakeholders* esse *softgoal* foi decomposto em Integrabilidade com várias tecnologias: servidor de aplicação, XML e XHTML, Servlet, JSP (vide Figura 6.2).

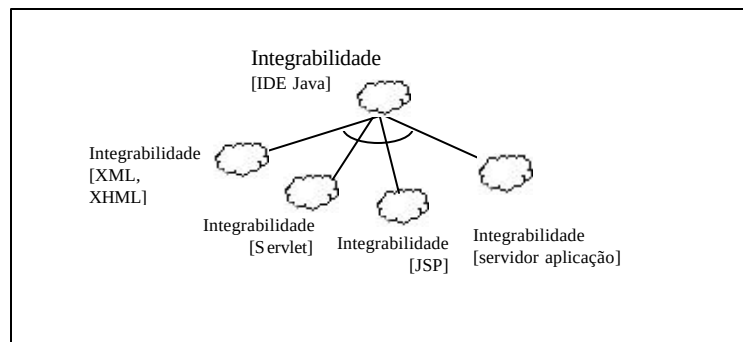


Figura 6.2 - Grafo SIG do requisito *Integrabilidade*

O requisito Req_17 refere-se ao *softgoal* manutenibilidade da IDE Java que foi decomposto em Adaptabilidade da ferramenta a novas tecnologias; Flexibilidade para atualizar-se a novos componentes Web e de banco de dados; e Extensibilidade a novas tecnologias.

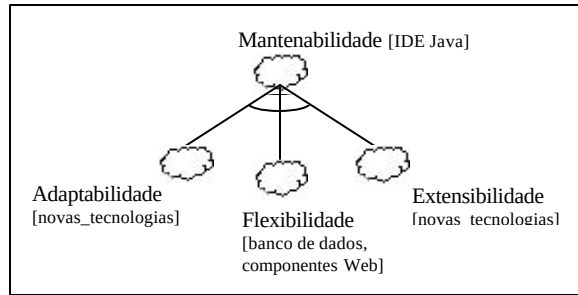


Figura 6.3 - Grafo SIG do requisito *Manutenabilidade*

O Req_18 refere-se à segurança do mecanismo de controle de versão, esse *softgoal* foi decomposto em Disponibilidade de histórico das alterações e nome do usuário que realizou a modificação; Integridade do código alterado; e Confidencialidade de acesso ao código. Esse último *softgoal* sofreu as seguintes operacionalizações: Identificação do usuário e Autorização do usuário.

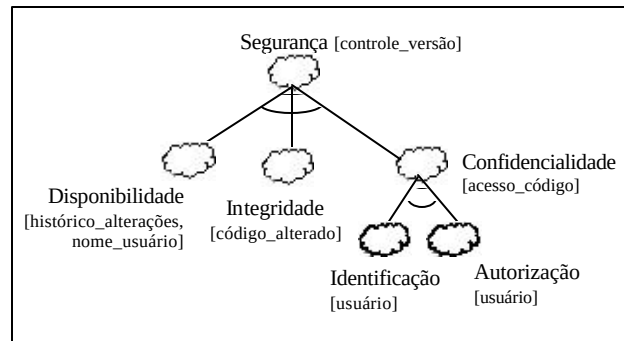


Figura 6.4 - Grafo SIG do requisito *Segurança*

O Req_19 diz respeito ao nível de produtividade que a IDE Java deve prover aos desenvolvedores. Esse *softgoal* foi refinado em Disponibilidade de Wizards e mecanismo de controle de versão; Capacidade para identificar erros no código gerado; e Eficiência da IDE Java. Em seguida esse último *softgoal* foi operacionalizado em Eficiência da ferramenta de debugger e Eficiência do editor de código. Vale salientar que não foram feitos os grafos SIG referentes aos outros requisitos não-funcionais pois a equipe de seleção não verificou a necessidade de refiná-los. Seguindo as indicações do NFR *Framework*, o próximo passo a ser realizado é identificar os inter-relacionamentos entre os requisitos.

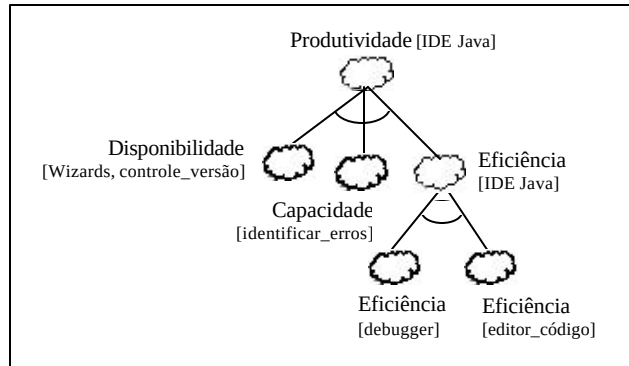


Figura 6.5 - Grafo SIG do requisito Produtividade

A Figura 6.6 fornece o grafo SIG completo, descrevendo todos os requisitos não-funcionais e a interação entre eles. Foram encontrados 16 relacionamentos entre os requisitos que à primeira vista não tinham relação alguma. A fim de facilitar o entendimento desse grafo, será analisado o relacionamento entre os requisitos não-funcionais (RNF) de cada árvore do grafo individualmente.

- Usabilidade[IDEJava]

1. O RNF Customização[capacidades] fornece uma contribuição positiva para usabilidade geral da ferramenta pois a capacidade de customizar o ambiente de trabalho facilita o uso dos usuários;
2. O RNF Disponibilidade[wizards, controle_versão] também tem uma contribuição positiva para satisfazer a usabilidade da ferramenta já que essas capacidades auxiliam o desenvolvimento das aplicações, os wizards permitem que o desenvolvedor não precise se preocupar com aspectos triviais do código, já o controle de versão facilita o desenvolvimento de aplicações feitas por grandes equipes;

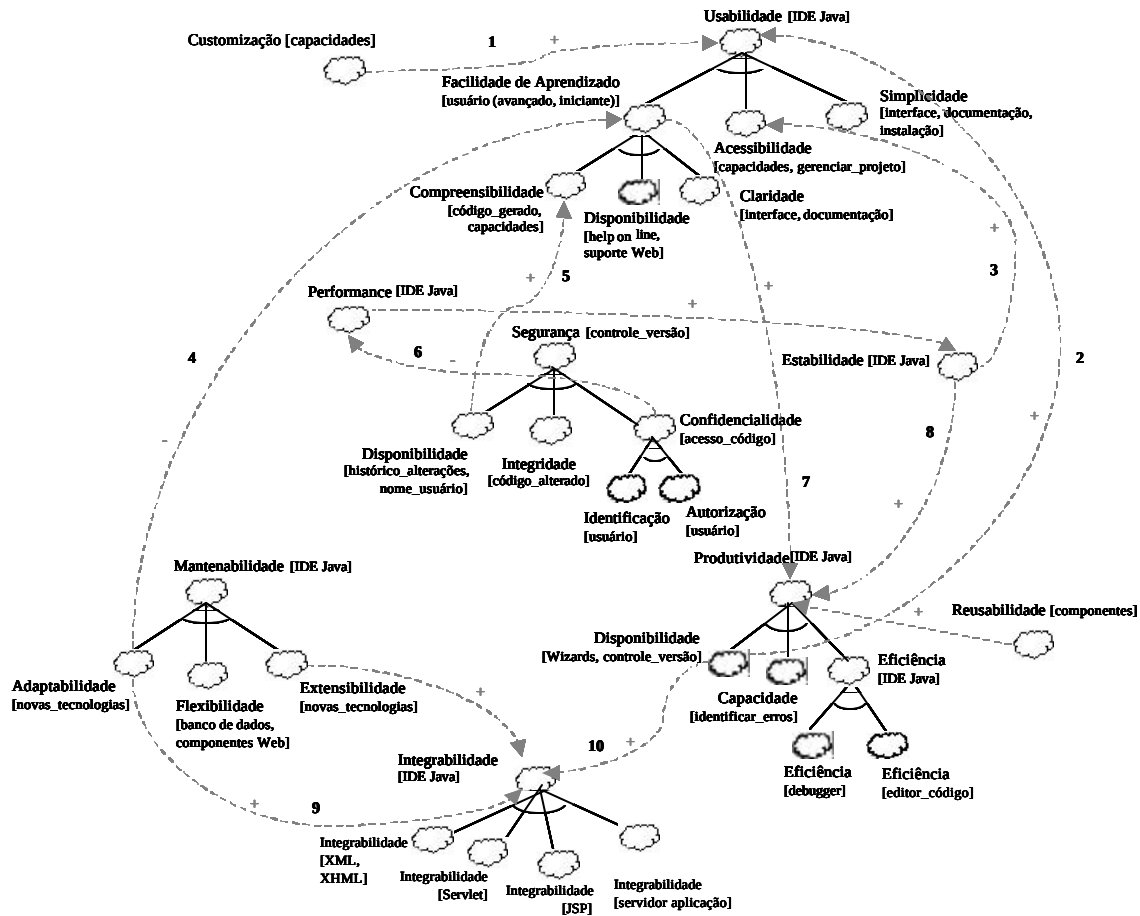


Figura 6.6 – Grafo SIG com os relacionamentos entre os requisitos

3. A Estabilidade[IDEJava] influencia positivamente o RNF Acessibilidade pois se o sistema for estável, i.e. tiver baixo índice de falhas, de um modo geral o acesso às capacidades da ferramenta pode ser facilitado;
4. O RNF Adaptabilidade[novas_tecnologias] tem uma contribuição negativa com a facilidade de aprendizado, em especial dos usuários iniciantes já que a inclusão de novas tecnologias à ferramenta pode aumentar a curva de aprendizado e dificultar o aprendizado dos usuários;
5. O RNF Disponibilidade[histórico_alterações, nome_usuário] contribui positivamente para a compreensibilidade do código gerado pois a disponibilidade

dessas informações ajuda que o desenvolvedor entenda melhor o código gerado por outros usuários que estão desenvolvendo o sistema;

- Performance[IDEJava]

6. A Confidencialidade[acesso_código] também afeta de forma negativa a performance da ferramenta já que as operacionalizações de identificação e autorização de usuários desse RNF podem comprometer o tempo de resposta da ferramenta;

- Produtividade[IDEJava]

7. O RNF Facilidade de Aprendizado[usuário(avançado,iniciante)] fornece uma contribuição positiva para o aumento da produtividade;
8. A Estabilidade[IDEJava] também influencia positivamente a produtividade que os desenvolvedores podem obter já que quanto mais estável for o uso da ferramenta maior será o ganho de produtividade;

- Integrabilidade[IDEJava]

9. O RNF Adaptabilidade[novas_tecnologia] fornece uma contribuição positiva para integração (i.e. suporte) da ferramenta com tecnologias de ponta pois ferramenta deve ter uma boa capacidade de adaptação para suportar a integração de novas tecnologias;
10. O RNF Disponibilidade de Wizards também fornece uma contribuição positiva para integração do IDE já que muitos wizards que as ferramentas disponibilizam facilitam o desenvolvimento de aplicações que usam, por exemplo, tecnologias de Servlet e XML.

A equipe de seleção usou as regras de situação a fim de resolver alguns *tradeoffs* entre os requisitos não-funcionais já refinados. Dessa forma, levou-se em consideração as prioridades estabelecidas pelos *stakeholders* para cada requisito (vide Tabela 6.5). Por exemplo, existe uma contribuição negativa entre o RNF Confidencialidade[acesso_código] e Performance[IDEJava] que precisa ser avaliada e resolvida. De acordo com a regra de

situação para resolver conflitos entre os requisitos, deve ser considerada a prioridade de cada requisito para em seguida decidir qual deverá ser comprometido a fim de satisfazer o outro. Por exemplo, se um requisito tem prioridade maior em relação a outro requisito e existe um conflito entre eles, deve-se optar por satisfazer o requisito de maior prioridade. Nesse caso, o Req_18 (*Prover segurança do controle de versões*) tem prioridade 2, enquanto o Req_11 (*Ter boa performance*) tem prioridade 4. Dessa forma, é indicado que num processo de tomada de decisão para seleção entre ferramentas candidatas, onde uma delas possui um poderoso mecanismo de segurança do controle de versões mas que compromete a performance da IDE, possa ser eliminada pois como foi visto anteriormente, esse RNF entra em conflito com a performance, que por sua vez tem prioridade mais alta.

Nesse sentido, os requisitos não-funcionais atuaram como critérios discriminatórios entre produtos com funcionalidades semelhantes. A análise dos RNFs foi essencial para realizar o processo de eliminação das ferramentas uma vez que quanto aos aspectos funcionais todas as quatro ferramentas atendiam os requisitos de maneira razoável. Durante o processo de análise da conformidade entre as ferramentas candidatas e o critério de seleção, a equipe de seleção chegou a um consenso que somente três produtos deveriam passar para a fase seguinte pois a outra alternativa não conseguiu satisfazer alguns dos requisitos não-funcionais críticos. Em particular, a decomposição obtida com o uso do NFR *Framework* facilitou esse processo já que alguns dos critérios que serviram para eliminação do Forte for Java foram obtidos a partir do refinamento dos requisitos não-funcionais. Por exemplo, a equipe de seleção observou que a produtividade dessa ferramenta podia ser comprometida pela baixa eficiência da sua ferramenta de debugger. Como foi mostrado na Figura 6.5, a eficiência da ferramenta de debugger é uma operacionalização do RNF Produtividade[IDEJava], onde ele possui prioridade 4. A Tabela 6.8 mostra o motivo pelo qual a ferramenta Forte for Java foi eliminada. Vale ressaltar que nesse momento foi levada em consideração principalmente a experiência prévia dos *stakeholders* com o uso dessas ferramentas para verificar a conformidade das mesmas em relação ao critério de seleção.

Ferramenta	Motivo da Eliminação
Forte for Java	- Essa ferramenta é mais recente no mercado que as outras e não possui boa adaptabilidade a novas tecnologias (Req_14), isto é, foi verificado que ela demora um tempo considerável para incluir novas funcionalidades; - A performance do tempo de execução e de inicialização foram consideradas baixas (Req_11); - A ferramenta não fornece suporte a EJB e HTML (Req_15); - A produtividade (Req_19) da ferramenta é comprometida por causa da baixa eficiência da ferramenta de debugger.

Tabela 6.8 - Motivos da Eliminação das ferramentas

6.2.3 Avaliação

Segundo o *template para avaliação dos produtos* (ver página 93), nessa etapa foi realizado o processo de tomada de decisão entre as ferramentas que continuaram na lista de produtos candidatos, que são VisualAge, Jdeveloper e Jbuilder. O método CRE sugere o uso das técnicas WSM e AHP. Inicialmente, foi aplicada a técnica WSM por ser considerada mais simples e direta. A Tabela 6.9 mostra o uso da técnica WSM para obter a pontuação (i.e. *ranking*) das três ferramentas em relação ao critério de seleção, inicialmente foram atribuídos escores de conformidade entre as ferramentas e cada um dos requisitos, em seguida utilizou-se a fórmula abaixo para obter o *ranking* entre as ferramentas.

$$Pont_Produto_a = \sum_{j=1}^n (peso * score_{aj})$$

onde: *a* representa uma alternativa e *n* o numero de critérios

A equipe de seleção considerou o uso dessa técnica bastante intuitivo e fácil, além disso, alguns *stakeholders* já estavam familiarizados com a técnica pois eles tinham usado variações da WSM em outros processos de seleção. Por outro lado, como foi visto na Seção 3.4.2, essa técnica mostrou sérias restrições, por exemplo:

- Como são produzidos números inteiros como resultado pode-se interpretar que eles correspondem à real diferença entre as ferramentas, porém a pontuação obtida representa o *ranking* relativo entre as mesmas;

- A técnica fornece uma avaliação independente entre os critérios, ou seja, considera o escore de conformidade de cada critério individualmente, no entanto os requisitos quase sempre relacionam-se como foi mostrado no documento de requisitos (vide Tabela 6.5).

Requisito	VisualAge			JBuilder			JDeveloper		
	Prioridade (1..4)	Confor – midade (0..3)	Pontua – ção	Prioridade (1..4)	Confor – midade (0..3)	Pontua – ção	Prioridade (1..4)	Confor – midade (0..3)	Pontua – ção
Req_1	3	3	9	3	3	9	3	3	9
Req_2	2	3	6	2	2	4	2	1	2
Req_3	2	3	6	2	3	6	2	2	4
Req_4	2	2	4	2	2	4	2	2	4
Req_5	4	3	12	4	3	12	4	3	12
Req_6	4	3	12	4	3	12	4	3	12
Req_7	2	3	6	2	3	6	2	3	6
Req_8	3	1	3	3	3	9	3	2	6
Req_9	4	3	12	4	3	12	4	2	8
Req_10	4	3	12	4	3	12	4	3	12
Req_11	4	3	12	4	3	12	4	3	12
Req_12	4	2	8	4	2	8	4	1	4
Req_13	4	3	12	4	3	12	4	3	12
Req_14	3	3	9	3	3	9	3	2	6
Req_15	2	2	4	2	2	4	2	1	2
Req_16	1	2	2	1	3	3	1	1	1
Req_17	2	3	6	2	3	6	2	2	4
Req_18	4	2	8	4	3	12	4	1	4
Req_19	4	3	12	4	3	12	4	3	12
Pontuação Final	155			164			132		

Tabela 6.9 - Avaliação das ferramentas usando a técnica WSM

De acordo com o ranking obtido com essa técnica, a ferramenta Jbuilder ficou em 1 lugar, vindo logo em seguida VisualAge e por fim Jdeveloper. A fim de comparar os resultados obtidos com a técnica WSM, também foi aplicada a técnica AHP. O principal problema encontrado nesse momento foi relativo à alta curva de aprendizado dessa técnica que é bem mais complexa. No entanto, essa técnica tem algumas vantagens em relação a WSM, a mais relevante é quanto a capacidade de avaliar a importância relativa entre os critérios fazendo comparações aos pares da importância de cada um, onde essa abordagem que está de acordo com a natureza dos requisitos de interagir entre si. Nesse contexto, a técnica AHP permite um tratamento hierárquico entre os critérios que coincide com a decomposição dos requisitos não-funcionais obtida com os grafos SIG do NFR *Framework*.

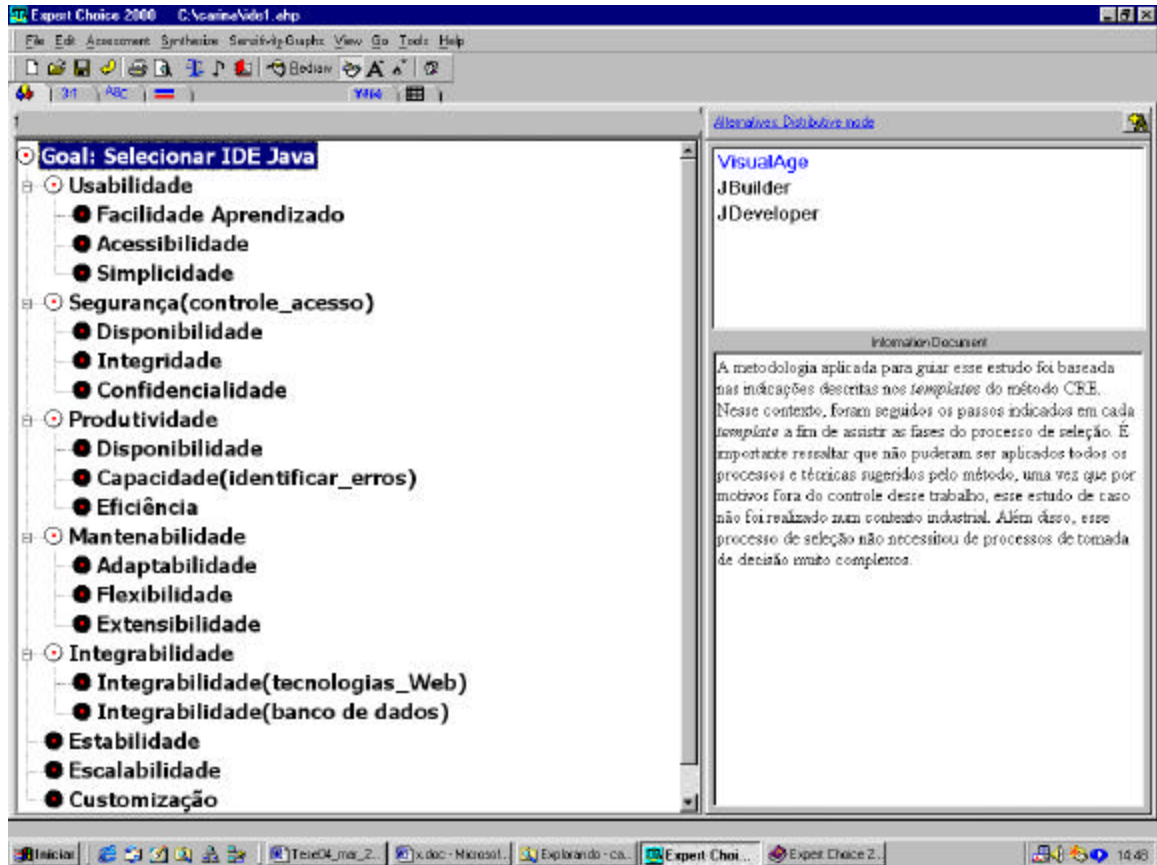


Figura 6.7 - Uso de ferramenta ExpertChoice

Durante a aplicação dessa técnica foi usada uma ferramenta *freeware* chamada ExpertChoice⁵. Ela se mostrou bastante adequada para consolidar o processo de tomada de decisão, além de ser fácil de usar. A fim de simplificar a demonstração de seu uso, foram considerados somente os requisitos não-funcionais para atuarem como critérios de seleção. Essa decisão foi aceitável já que observou-se que nesse estágio as ferramentas candidatas atendem de forma aceitável os requisitos funcionais especificados. Dessa forma, sua análise não iria influenciar muito o processo de tomada de decisão. O primeiro passo para utilizar a técnica AHP foi especificar a meta a ser obtida com esse processo de tomada de decisão, que nesse caso foi *Selecionar IDE Java*. Em seguida os critérios (i.e. RNF) e sub-critérios

⁵ <http://www.expertchoice.com>

foram descritos numa estrutura de árvore hierárquica e finalmente foram mostradas as alternativas avaliadas (ver Figura 6.7). O próximo passo seguido foi priorizar os requisitos fazendo comparações aos pares em relação a adequação dos produtos. Por exemplo, a Figura 6.8 mostra uma comparação entre as ferramentas Jbuilder e Jdeveloper em relação ao sub-critério “*facilidade de aprendizado*”. Uma das vantagens da ferramenta ExpertChoice é que ela permite três formas de comparações (verbal, numérica e gráfica) entre as ferramentas, o que torna esse processo bastante intuitivo. A comparação verbal fornece uma escala com gradações para avaliar as ferramentas (*extreme, very strong, strong, moderate, equal*). Por exemplo, na comparação entre elas em relação ao requisito *facilidade de aprendizado*, a ferramenta Jbuilder recebeu a pontuação *strong* em relação a Jdeveloper. Na comparação numérica, são propostos números de 1 a 9 para comparar as ferramentas. Finalmente, na comparação gráfica os *stakeholders* podem comparar as ferramentas em relação a uma escala gráfica.

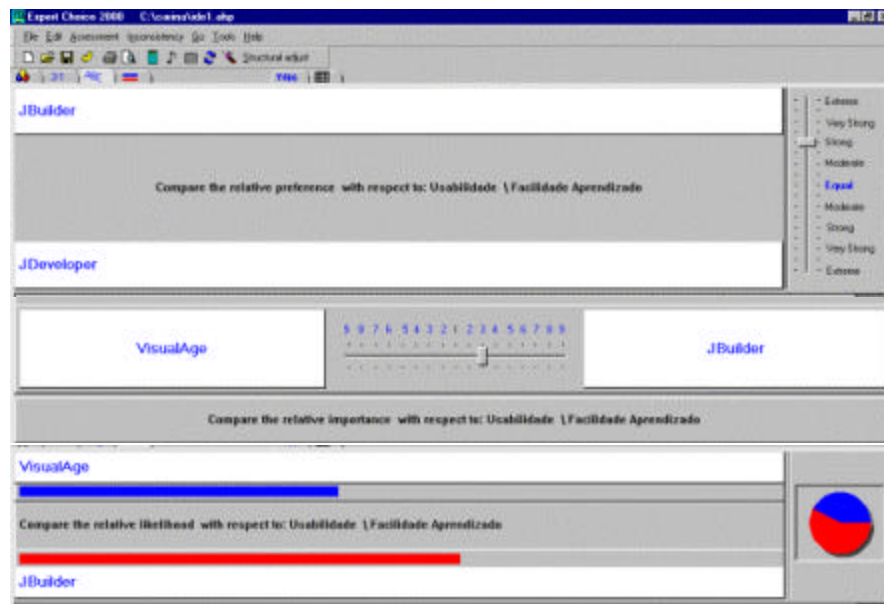


Figura 6.8 – Modos de comparação aos pares das ferramentas candidatas

Ao final do processo de julgamento sobre a importância relativa dos critérios, sub-critérios e alternativas, foi obtido o ranking entre as ferramentas. Houve quase um empate entre as ferramentas Jbuilder (40,4%) e VisualAge (37,8%), por último ficou Jdeveloper com 21,8%. A Figura 6.9 mostra um dos gráficos que podem ser gerados para analisar os

resultados obtidos que fornece o ranking final entre as ferramentas o desempenho de cada uma em relação aos requisitos avaliados. Esses resultados são similares aos obtidos com o uso da técnica WSM, porém a técnica AHP fornece mais subsídios para apoiar o processo de tomada de decisão.

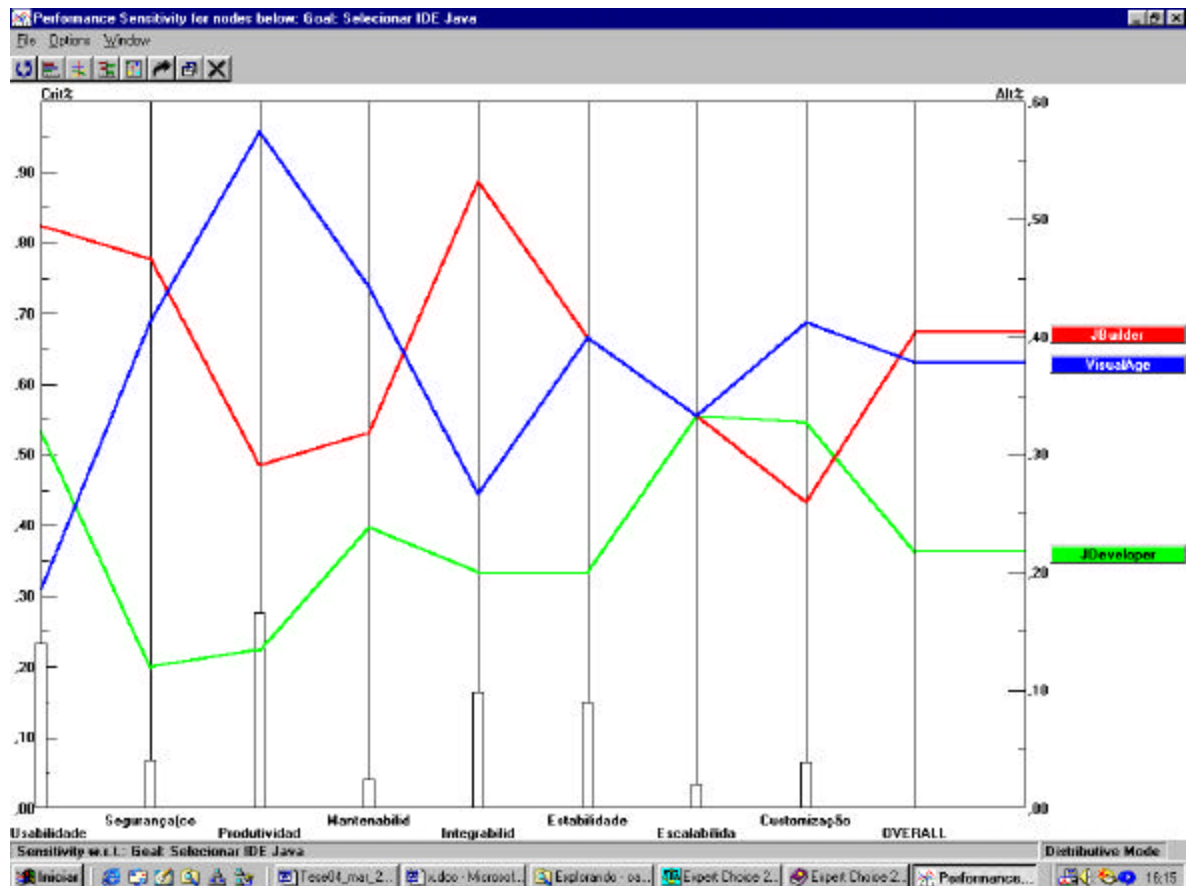


Figura 6.9 - Gráfico para análise de resultados (critérios X alternativas)

Segundo o *template para avaliação dos produtos*, nessa fase deve ser feita a estimativa de custos de cada ferramenta. Entretanto, para esse estudo de caso em particular, não foi necessário realizar essa atividade pois no contexto desse ecossistema acadêmico que o processo de seleção foi realizado, existem várias parcerias com fornecedores e muitos softwares são obtidos de forma gratuita.

6.2.4 Aceitação

Segundo o *template para aceitação dos produtos*, nessa fase são realizados os testes de aceitação com os futuros usuários e a verificação dos aspectos legais. No entanto, essa etapa não foi realizada pois a ferramenta escolhida poderá ser adquirida gratuitamente. Finalmente, nessa etapa deve ser escolhida a ferramenta que melhor atende as necessidades dos *stakeholders*. Levando em consideração os resultados obtidos com a utilização das técnicas para tomada de decisão, isto é sem incluir a estimativa de custos e os testes de aceitação, a ferramenta Jbuilder seria selecionada pois obteve a maior pontuação nas duas técnicas aplicadas.

6.3 CONSIDERAÇÕES FINAIS

Esse estudo de caso foi útil para mostrar a utilização do método CRE numa situação real para seleção de IDEs Java. Em particular, foi descrito passo a passo como cada atividade foi realizada durante esse processo de seleção. A Tabela 6.10 fornece o esforço de tempo necessário para realizar cada fase do método. A fase de identificação foi realizada sem consumir muitos esforços, necessitando de somente 12 horas, isso foi devido em grande parte à diversidade de informações disponíveis na Web sobre os IDEs Java.

A fase de descrição consumiu 67% do tempo gasto, isso mostra que a principal dificuldade desse processo de seleção foi relativa à elaboração do critério de seleção, onde a especificação dos requisitos não-funcionais consumiu apenas 10 horas, já que o engenheiro de requisitos dominava o uso dessa técnica. Nesse sentido, como pesquisa futura é interessante verificar a curva de aprendizado dessa abordagem por um usuário iniciante. Vale ressaltar que o levantamento de informações sobre as ferramentas foi um processo demorado (18 horas) pois apesar da grande quantidade de material disponível, foi difícil encontrar informações consistentes sobre as capacidades e limitações das ferramentas. Curiosamente, a fase de avaliação consumiu bem menos tempo do que o previsto. Em particular, a utilização da técnica AHP foi realizada em apenas 7 horas, graças principalmente ao uso da ferramenta *ExpertChoice* que apoiou a técnica, realizando todos

os cálculos necessários e fornecendo tabelas e gráficos. O tempo total para realizar esse estudo de caso foi de 65 horas, considerando a participação de 3 pessoas na equipe de seleção, o esforço foi de 21 horas/homem de trabalho. Dessa forma, analisando todas as vantagens obtidas com a utilização do método CRE para sistematizar e apoiar detalhadamente todo o processo de seleção pode-se considerar que o esforço necessário ficou dentro do limite aceitável pelos *stakeholders*. A Figura 6.10 fornece um gráfico que mostra o percentual gasto nas principais atividades das três fases do método CRE que foram realizadas nesse estudo de caso.

Fases do CRE	Tempo gasto	%	Observações
Identificação <i>Principais Atividades</i> - Definição do critério de seleção inicial - Localização dos produtos - Elaboração da lista de candidatos	12 h 3 h 4 h 5 h	19 5 6 8	O tempo gasto para realizar as atividades dessa fase foi curto principalmente por causa da grande diversidade de IDEs Java disponível na Web.
Descrição <i>Principais Atividades</i> - Levantamento de informações sobre as ferramentas - Elaboração do documento de requisitos - Uso do NFR Framework para refinar os requisitos não-funcionais	43 h 18 h 15 h 10 h	67 28 23 16	A parte mais difícil foi obter informações detalhadas sobre as capacidades de cada ferramenta, já em relação à elaboração do documento de requisitos a demora foi devida ao tempo gasto com entrevistas com os <i>stakeholders</i> .
Avaliação <i>Principais Atividades</i> - Uso da técnica WSM - Uso da técnica AHP	10 h 3 h 7 h	14 3 11	Ao contrário do que se imaginava, essa fase foi realizada num curto intervalo de tempo, principalmente por causa da utilização de uma ferramenta usada para auxiliar o uso da técnica AHP.
Aceitação	-	-	Essa fase não foi realizada pois não se adequava no enfoque desse estudo.
Total	65 h	100	

Tabela 6.10 - Tempo gasto para aplicar cada fase do Método CRE

A principal limitação desse estudo de caso foi o fato de não ter sido possível aplicar todas as técnicas propostas nos *templates*, já que o escopo desse processo de seleção foi acadêmico e não industrial. Em particular, a participação dos *stakeholders* foi restrita pois eles não aplicaram diretamente o método no seu ambiente de trabalho. Dessa forma, é importante realizar estudos mais complexos para avaliar com maior rigor a viabilidade do método CRE num contexto organizacional. Uma proposta interessante é observar duas equipes trabalhando no mesmo processo de seleção, uma delas usando o método CRE e a

outra não, e então comparar os resultados obtidos. Dessa forma, seria possível analisar com maior precisão as vantagens do método proposto. Basili [BAS84], por exemplo, propõe uma abordagem chamada replicação de projetos que pode ser usada para acompanhar esse estudo de caso futuro.

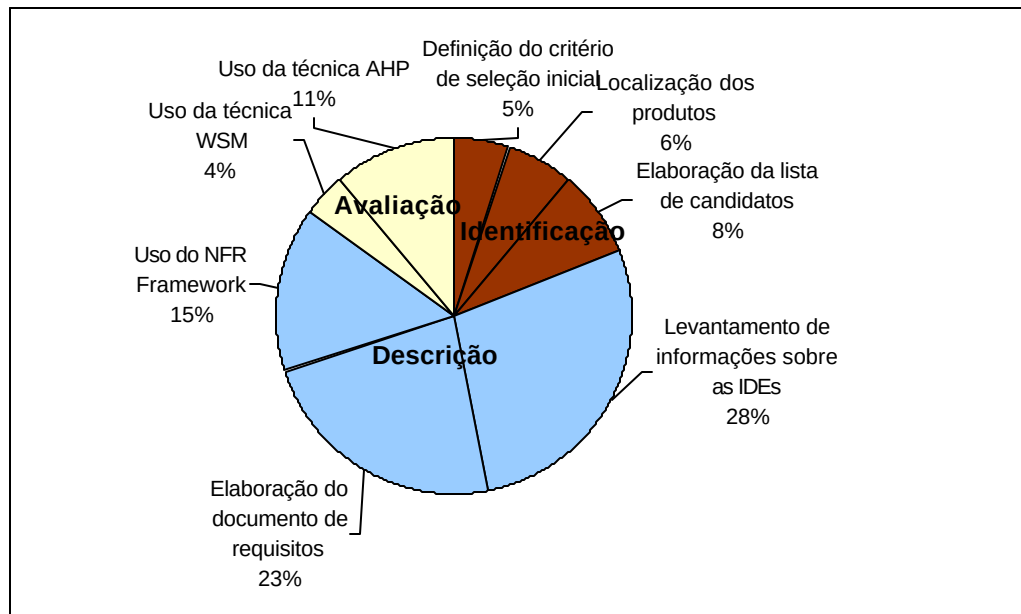


Figura 6.10 - Percentual de tempo gasto nas fases do método CRE

Durante a realização desse estudo foram obtidas algumas lições que serão consideradas o ponto de partida para realizar trabalhos futuros a fim de melhorar o método CRE.

- Lição_1: A participação de especialistas no domínio é fundamental durante o processo de avaliação das ferramentas;
- Lição_2: Foi difícil encontrar informações significativas sobre as ferramentas na documentação disponível, assim foi necessário usar o conhecimento de especialistas que utilizavam os IDEs;
- Lição_3: O uso das técnicas propostas foi direto pois o engenheiro de requisitos já tinha conhecimento prévio e direcionou a aplicação das mesmas, assim não foi possível observar a facilidade de uso dessas técnicas por usuários iniciantes;

- Lição_4: Os *stakeholders* mostraram interesse em participar das reuniões e entrevistas mas consideraram que iriam perder tempo realizando algumas das técnicas propostas.

É importante ressaltar a relevância dessa última lição já que ela evidencia um dos grandes problemas enfrentados pela engenharia de software que refere-se à falta de interesse dos engenheiros em utilizar técnicas e metodologias para assistir os processos de desenvolvimento de sistemas de software. Nesse sentido, essa pesquisa considera que além da necessidade de novos métodos para apoiar o processo de desenvolvimento baseado em COTS, como é o caso do método CRE, é necessário uma mudança de comportamento dos desenvolvedores no que diz respeito às práticas propostas pela engenharia de software.

Capítulo 7

Conclusões e Trabalhos Futuros

Este capítulo mostra as principais contribuições do trabalho para a área de desenvolvimento baseado em COTS. Em seguida, descreve trabalhos futuros que poderão ser realizados a partir dessa dissertação e finalmente fornece alguns possíveis direcionamentos para pesquisas futuras na área.

7.1 CONTRIBUIÇÕES DO TRABALHO

A área de desenvolvimento baseado em COTS é bastante recente, no entanto inúmeras pesquisas estão sendo realizadas nessa área. Nesse contexto, as perspectivas futuras são bastante otimistas para o mercado de produtos COTS. Esse trabalho forneceu importantes contribuições na área de pesquisa de engenharia de requisitos para desenvolvimento de sistemas baseado em COTS, são elas:

- Um estudo comparativo das principais vantagens e limitações dos métodos existentes na literatura para seleção de COTS;
- Um levantamento dos principais desafios e problemas relacionados com a aquisição e análise dos requisitos dos *stakeholders* para obter um processo de seleção de COTS bem sucedido;
- Uma proposta do processo de engenharia de requisitos específico para desenvolvimento baseado em COTS;
- Um método baseado em requisitos que assiste o processo de seleção de produtos COTS;
- Um estudo de caso prático utilizando o método proposto para selecionar ambientes para desenvolvimento em Java.

Durante o desenvolvimento desse trabalho foram identificadas 4 hipóteses que estruturaram e direcionaram a presente pesquisa, a seguir cada hipótese é reexaminada a fim de verificar a sua efetividade.

H1 Os métodos para seleção de COTS existentes na literatura não abordam de forma satisfatória os requisitos dos stakeholders.

- ❖ Como foi investigado na seção 3.3.6, a hipótese **H1** é satisfeita pois a maioria dos métodos descritos não possui um processo bem definido de engenharia de requisitos. A única exceção é o método PORE [NCU00] que fornece uma abordagem

direcionada para requisitos. Entretanto, esse método também possui restrições no que diz respeito ao tratamento dos requisitos não-funcionais. A Tabela 3.1 mostra uma comparação dos principais métodos existentes em relação às atividades que os mesmos cobrem.

H2 *Novos desafios surgem no processo de engenharia de requisitos para desenvolvimento baseado em COTS que não existiam no processo tradicional.*

- ❖ De fato, como em qualquer abordagem de engenharia de software, o processo de desenvolvimento de sistemas baseados em COTS possui novos e diferentes desafios em relação ao processo tradicional. Como mostrado na Seção 4.3, um dos objetivos dessa tese foi analisar as implicações do processo de engenharia de requisitos para desenvolvimento baseado em COTS e mostrar as possíveis adequações no processo tradicional para atender as novas necessidades da abordagem COTS.

H3 *Requisitos não-funcionais são aspectos fundamentais durante o processo de seleção.*

- ❖ Tradicionalmente, os requisitos não-funcionais são mais complexos e difíceis de tratar do que os requisitos funcionais, porém eles são considerados críticos pois estão diretamente associados com a satisfação dos *stakeholders*, no desenvolvimento baseado em COTS eles assumem uma relevância ainda maior uma vez que eles serão importantes critérios de decisão para seleção de produtos concorrentes como foi mostrado na Seção 6.2.2 do estudo de caso.

H4 *O método proposto é simples de ser usado e proporciona assistência nas tarefas de avaliação e seleção de produtos COTS.*

- ❖ A partir do estudo de caso realizado no Capítulo 6, pode-se verificar a viabilidade do método CRE num processo de seleção de IDE Java. Nesse sentido, as atividades propostas foram aplicadas de forma bastante simples e intuitiva. Um aspecto importante foi que o tempo gasto na realização dessa seleção foi considerado aceitável pelos *stakeholders*.

O presente trabalho forneceu um avanço relevante para melhorar o estado da arte da engenharia de requisitos para desenvolvimento de sistemas baseados em COTS. Entretanto, os esforços realizados para o desenvolvimento dessa pesquisa podem ser considerados apenas o início de um trabalho mais abrangente. Nesse sentido, a próxima seção descreve alguns dos trabalhos futuros que poderão ser realizados a fim de continuar e aprofundar as propostas que foram discutidas nessa dissertação.

7.2 TRABALHOS FUTUROS

Inicialmente, pretende-se realizar um estudo de caso mais detalhado onde seja possível acompanhar o uso do método CRE num processo de seleção real sendo realizado por uma organização. Num contexto mais abrangente, as propostas de trabalhos futuros podem ser divididas em duas direções distintas. A primeira está relacionada com a implementação de um protótipo que fornecerá um assistente de avaliação para auxiliar a realização das atividades propostas por cada fase do método CRE. Como foi descrito no Capítulo 5, o CRE descreve um conjunto de *templates* que fornecem diretrizes e metas com o objetivo de melhorar e sistematizar o processo de seleção. Em geral, o uso desses *templates* requer que a equipe de seleção possua um vasto conhecimento do domínio e das técnicas utilizadas. A fim de facilitar a utilização do método, poderá ser desenvolvido um protótipo assistente que fornecerá auxílio para usuários inexperientes ou processos de seleção muito complexos. A proposta é que ele utilize a técnica de inteligência artificial de sistema especialista [JAC99] [AMA93]. Essa técnica emprega o conhecimento humano para resolver problemas que requerem a presença de um especialista. A proposta da arquitetura desse protótipo possui os seguintes componentes (ver Figura 7.1):

- Base de Conhecimento – Armazena o conhecimento obtido do especialista no domínio, além de guardar informações obtidas em processos de seleção anteriores;
- Máquina de Inferência – Utiliza as regras de situação para inferir que ação deve ser tomada em cada situação em particular, ou seja, usa as regras para guiar o processo de raciocínio lógico;

- Módulo Assistente – Analisa o conjunto de metas a serem atingidas e fornece conselhos sobre a melhor estratégia a seguir a partir de dados obtidos da máquina de inferência.

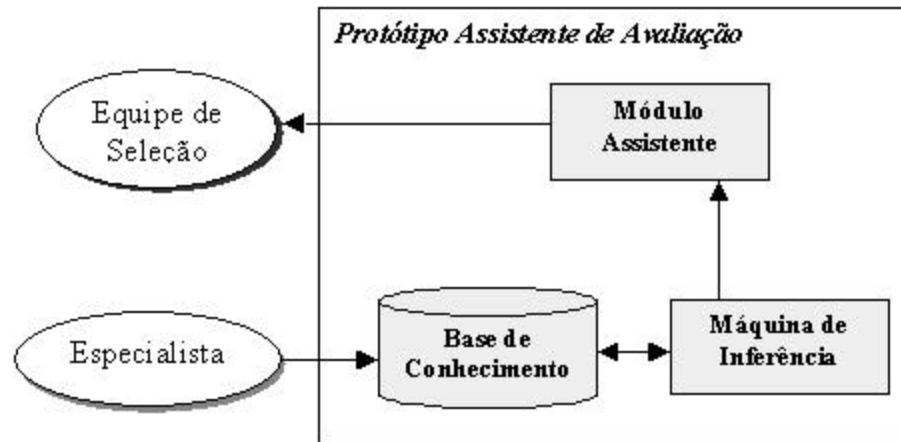


Figura 7.1 - Protótipo Assistente de Avaliação

Outra linha de pesquisa futura é estender o método CRE para assistir a seleção de múltiplos componentes. Nesse tipo de processo, vários produtos de diferentes vendedores são selecionados e integrados a fim de fornecer funcionalidades que um único produto é incapaz de prover [FOR99]. As decisões de seleção de produtos num sistema com múltiplos COTS são diretamente influenciadas pelos inter-relacionamentos existentes entre eles. Nesse contexto, a arquitetura de software exerce forte influência na seleção dos produtos já que a mesma deve ser um contínuo processo de *tradeoff* entre os requisitos dos *stakeholders* e os produtos candidatos. Além disso, é a partir da arquitetura de software que se pode especificar de forma mais detalhada como é feita a interconexão entre os componentes. Nesse contexto, são necessárias novas técnicas e metodologias para suportar o complexo processo de seleção de múltiplos COTS que atenda aos requisitos dos *stakeholders* e ao mesmo tempo satisfaça à arquitetura do sistema.

GLOSSÁRIO

AHP	<i>Analytical Hierarchy Process</i> é uma técnica de tomada de decisão baseada na decomposição de um problema com critérios múltiplos numa hierarquia e é feito um ranking das alternativas a partir de comparações aos pares entre elas.
Componente de software	São artefatos autocontidos, claramente identificáveis, que descrevem ou realizam uma função específica e têm interfaces claras em conformidade com um dado modelo de arquitetura de software, documentação apropriada e um grau de reutilização definido.
COTS	<i>Commercial Off-The-Shelf</i> são produtos de software disponíveis no mercado que podem ser comprados ou licenciados e são suportados e atualizados pelo vendedor que retêm os direitos de propriedade.
Critério	Requisito que os produtos precisam atender.
Critério de seleção	Conjunto de requisitos que os produtos precisam atender, pelo menos parcialmente. É uma forma objetiva de testar se determinado requisito está sendo atendido pelo produto.
DBC	<i>Desenvolvimento Baseado em COTS</i> é o processo de desenvolver sistemas a partir da seleção e integração de produtos já existentes.
Equipe de seleção	Grupo responsável pelo processo de seleção, normalmente é formado por gerentes, usuários finais, engenheiros de software e de requisitos.
ERP	<i>Enterprise Resource Planning</i> são soluções COTS que cobrem vários setores da organização e para isso necessitam de inúmeras mudanças organizacionais.
Escore de conformidade	Medida que avalia o grau de satisfação entre os produtos candidatos e os requisitos dos stakeholders.
IDE	<i>Integrated Development Environment</i>
Método CRE	<i>COTS-based Requirements Engineering</i> é um método para assistir a seleção de produtos COTS que enfatiza a aquisição e análise de requisitos não-funcionais.
Processo iterativo	Processo que rejeita produtos que não estão em conformidade com os requisitos à medida que descrições mais detalhadas dos requisitos são produzidas.
Produto candidato	Produto que está sendo considerado como possível solução.
Regras de situação	Regras que usam implicações lógicas para inferir possíveis soluções de uma situação.
Requisito	É uma propriedade geral, funcionalidade ou restrição do sistema de software.
Requisito contratual	É relacionado com os acordos e negociações realizados entre os vendedores de COTS e o comprador.
Requisito desejável	Característica que não é crítica para o sucesso do sistema.
Requisito crítico	Requisito dos stakeholders imprescindível de ser atendido.
Requisito funcional	Define as funções que o sistema precisa prover.
Requisito não-funcional	É uma propriedade de qualidade ou restrição que o software precisa satisfazer.

Requisitos discriminatórios	Requisitos que ajudam a discriminar entre produtos candidatos aparentemente semelhantes.
Softgoal	Representa metas que possuem uma relação E/OU com outras metas que podem ser satisfeitas ou não.
Stakeholder	Pessoa que esteja interessada com o sucesso do desenvolvimento do sistema ou qualquer pessoa que seja afetada pelo sistema, por exemplo: usuário final, desenvolvedor, gerente, etc.
Template	Modo sistemático de apresentar um processo de alto nível que fornece orientação sobre as atividades a serem executadas.
Tomada de decisão	Processo que decide se determinado produto é adequado para ser selecionado.
Tradeoff	Processo de negociação entre duas ou mais partes.
WML	Wireless Markup Language
JSP	Java Server Pages

REFERÊNCIAS BIBLIOGRÁFICAS

- [ALE99] ALENCAR, F. **Mapeando a Modelagem Organizacional em Especificações Precisas.** Tese de Doutorado, Centro de Informática, Universidade Federal de Pernambuco. Dec. 1999. pp. 194 – 208.
- [ALV00] ALVES, C. CASTRO, J. e ALENCAR, F. **Requirements Engineering for COTS Selection.** 3rd Workshop on Requirements Engineering, Rio de Janeiro. Jul. 2000. pp. 85 – 96.
- [ALV01] ALVES, C. CASTRO, J. **Um Método Baseado em Requisitos para Seleção de COTS.** Fourth Workshop Iberoamerican on Software Engineering and Software Environment (IDEAS'01) San Jose, Costa Rica, Apr. 2001.
- [AMA93] AMARAL, J. e CASTRO, J. **Assist;DS – Um Assistente Inteligente para o Desenvolvimento de Software.** Simpósio Brasileiro de Inteligência Artificial, Porto Alegre, Brasil. 1993. pp. 419 – 432.
- [AMB97] AMBRIOLA, V. e GERVASI, V. **Processing Natural Language Requirements.** 12th International Conference on Automated Software Engineering. Lake Tahoe, EUA. Nov. 1997.
- [ANT98] ANTONIOU, G. **The Role of Non-monotonic Representations in Requirements Engineering.** International Journal of Software Engineering and Knowledge Engineering, vol 8. 1998.
- [AOY98] AOYAMA, M. **New Age of Software Development: How Component-Based Software Engineering Changes the Way of Software Development.** International Workshop on Component-Based Software Engineering, Apr. 1998.
- [ARI5] <http://www.esrin.esa.it/esa/ariane/ariane5.html>
- [BAS84] Basili, V.R. and Perricon, B. **Software Errors and Complexity An Empirical Investigation.** ACM Communication 27 No 1. 1984. PP. 45-52.
- [BEL76] BELL, T. E. e THAYER, T. A. **Software Requirements: Are They Really a Problem?** Proceedings on 2nd International Conference on Software Engineering. San Francisco, 1976.
- [BER98] BERGNER, K. RAUSCH, A. e SIHLING, M. **Componentware--The Big Picture.** International Workshop on Component-Based Software Engineering, Apr. 1998.
- [BOE81] BOEHM, B. **Software Engineering Economics.** Prentice Hall, 1981.
- [BOE84] BOEHM, B. W. **Verifying and Validating Software Requirements and Design Specification.** IEEE software Vol 1 pp-94-103, Jan 1984.
- [BOE96] BOEHM, B. e IN, H. **Identifying Quality-Requirement Conflicts.** IEEE Software, Mar. 1996, pp. 25-35
- [BOE98] BOEHM, B. ABTS, C. BAILEY, E. **COCOTS Software Integration Cost Model: an Overview.** Proceedings of the California Software Symposium, Oct. 1998.
- [BRE98] BREITMAN, K.K.; LEITE, J.C.S.P. **A Framework for Scenario Evolution.** Third International Conference on Requirements Engineering - III. IEEE Computer Society Press. Los Alamitos, CA, U.S.A. 1998. pp. 214 - 221.

- [BRE99] BREITMAN, K. K. LEITE, J. C. P. e FINKELSTEIN, A. C. W. **The World's a Stage: A Survey on Requirements Engineering using a Real-life Case Study.** Jornal da Sociedade Brasileira de Computação. vol.6 nº1. Jul. 1999.
- [BRO95] BROWN, A.W. CARNEY, D. e McFALLS, M. D. **Technical Report CMU/SEI-95-SR-007** Proceedings SEI/MCC Symposium on Use of COTS in Systems Integration. Jun. 1995.
- [BRO96] BROWN, A. W. e WALLNAU, K. C. **Engineering of Component-Based Systems, Component-Based Software Engineering.** Software Engineering Institute, IEEE Computer Society Press, 1996.
- [BRO97] BROWN, A. W e SHORT, K. **On Components and Objects: The Foundation of Component-Based Development.** Proceedings of the Fifth International Symposium on Assessment of Software Tools. Jun. 1997. pp. 112-121.
- [BRO98] BROWN, A. e WALLNAU, K. C. **The Current State of CBSE.** IEEE Software, Sep./Oct. 1998.
- [CAR97] CARNEY, D. **Assembling Large Systems from COTS Components: Opportunities, Cautions, and Complexities.** SEI Monographs on the Use of Commercial Software in Government Systems, Jun. 1997.
- [CAR98] CARNEY, D. **Quotations from Chairman David: a little Red Book of Truths to Enlighten and Guide on the Long March Towards the COTS Revolution.** Software Engineering Institute. Jul. 1998.
- [CAS95] CASTRO, J. **The Process of Requirements Formalization: The FORMLAB Project.** Proceedings of Focus Symposium on Information Systems Analysis and Synthesis - ISAS 95. This event was part of the 5th International Symposium on Systems Research, Informatics and Cybernetics – INTERSYMP'95. Aug. 1995, Baden-Baden, Germany, pp. 1-5.
- [CAS96] CASTRO, J. GAUTREAU, J. e TORANZO, M. **Tool Suport for Requirements Formalization.** Proceedings of the ACM SIGSOFT Viewpoints 96: International Workshop on Multiple Perspective Software Development - ACM VP 96. San Francisco, USA, Oct. 1996, pp. 202-206.
- [CBSE00] <http://www.sei.cmu.edu/cbs/cbse2000/index.html>
- [CHA98] CHAVES, A. TORNABENE, C. e WIEDERHOLD, G. **Software Component Licensing: A Primer.** IEEE Software. Out. 1998.
- [CHU00] CHUNG, L. NIXON, B. YU, E. e MULOPOULOS, J. **Non-Functional Requirements in Software Engineering.** Kluwer Academic Publisher, 2000.
- [CHU95a] CHUNG, L. NIXON, B. e YU, E. **Using Non-Functional Requirements to Systematically Support Change.** Proceedings IEEE International Symposium on Requirements Engineering. Mar. 1995.
- [CHU95b] CHUNG, L. e NIXON, B. **Dealing with Non-Functional Requirements: Three Experimental Studies of a Process-Oriented Approach.** 17th International Conference on Software Engineering. Apr. 1995.
- [CLE95] CLEMENTS, P. C. **From Subroutines to Subsystems: Component-Based Software Development.** The American Programmer. Nov. 1995.
- [CLE97] CLEMEN, P. T. **Making Hard Decisions: An Introduction to Decision Analysis.** Hardcover 2nd edition, 1997.
- [COC98] COCOTS <http://sunset.usc.edu/research/COCOTS/index.html>

- [CYS01] CYSNEIROS, L. M. **Requisitos Não-Funcionais: da Elicitação ao Modelo Conceitual**. Tese de Doutorado, Departamento de Informática, PUC-RJ. Fev. 2001.
- [CYS99] CYSNEIROS, L. M. e LEITE, J. C. **Integrating Non-Functional Requirements into Data Modeling**. 4th International Symposium on Requirements Engineering. Limerick, Ireland. Jun. 1999.
- [DAM97] DAMIAN, A. HONG, D. e LI, H. **Joint Application Development and Participatory Design**. Disponível em www.cpsc.ucalgary.ca/~pan/seng/613/report.html
- [DAV92] DAVIS, A. **Operational Prototyping: A New Development Approach**. Software, 1992.
- [DAV93] DAVIS, A. M. **Software Requirements: Objects, Functions, and States**. Prentice Hall, Englewood Cliffs, New Jersey, USA, 1993.
- [DEA97] DEAN, C. J. e VIGDER, R. M. **System Implementation Using Commercial-Off-The-Shelf Software**. Proceedings of the 1997 Software Technology Conference. Mai. 1997.
- [DEI98] DEIFEL, B. **Requirements Engineering for Complex COTS**. Position Paper. REFSQ'98. 1998.
- [DEI99] DEIFEL, B. **A Model for version Planning of COTS**. Extended Abstract. SCE'99.
- [DEL97] DELLAROCAS, C. **Toward a Design Handbook for Integrating Software Components**. Proceedings of the Fifth International Symposium on Assessment of Software Tools. Jun. 1997. pp. 3-13.
- [DOB91] DOBSON, J. **A Methodology for Analyzing Human Computer-related Issues in Secure Systems**. International Conference on Computer Security and Integrity in our Changing World, 151-170, Espoo, Finland, 1991.
- [DUK99] DUKIC, L. **Non-Functional Requirements for COTS Software Components**. Workshop Ensuring Successful COTS Development, Los Angeles, EUA. Mai. 1999.
- [ELI95] ELLIS, T. **COTS integration in software solutions**. Systems Engineering in the Global Marketplace. Jul. 1995.
- [ERP00] <http://www.erpworld.org>
- [ESI96] European Software Institute, **European User Survey Analysis**. Report USV_EUR 2.1 ESPITI Project, 1996.
- [EXP00] <http://www.expertchoice.com>
- [FIN95] FINNIE, G. R. WITTIG, G. E. e PETKOV, D. I. **Prioritizing Software Development Productivity Factors Using the Analytic Hierarchy Process**. Journal of Systems and Software, vol 22, 1995.
- [FIN96] FINKELSTEIN, A. C. W. e DOWELL, J. **A Comedy of Errors: The London Ambulance Services Case Study**. Proceedings 8th International Workshop on Software Specification and Design. IEEE, Computer Society Press, 1996.
- [FOR99] FOREMAN, J. **On the Front Lines of COTS - Lessons Learned, Speculation for the Future**. Software Engineering Institute. 1999.
http://www.sei.cmu.edu/activities/cbs/cbs_slides/stc98/frontlines/sld001.htm
- [FOX97] FOX, G. MARCOM, S. e LANTNER, K. **A Software Development Process for COTS-based Information System Infrastructure**. Proceedings of the Fifth International Symposium on Assessment of Software Tools. Jun. 1997. pp. 133 – 142.

- [GAR95] GARLAN, D. ALLEN, R. e OCKERBLOOM, J. **Architectural Mismatch: or Why It's Hard to Build Systems Out of Existing Parts**. Proceedings of the 17th International Conference on Software Engineering, Seattle, EUA. 1995. pp. 179 – 185.
- [GIG96] GIGOU, J. ASTORG, J. M. BARREAU, F. **The Ariane 5 Booster Recovery System**. ESA Bulletin n. 85. Feb. 1996.
- [GOT94] GOTEL, O. e FINKELNSTEIN, A. **An Analysis of the Requirements Traceability Problem**. 1st International Conference on Requirements Engineering, Colorado Springs, EUA. Apr. 1994.
- [HAI97] HAINES, G. CARNEY, D. e FOREMAN, J. **Component-Based Software Development/COTS Integration**. Software Engineering Institute, Carnegie Mellon University, USA. Jan. 1997.
- [HAU88] HAUSER, J. e CLAUSING, D. **The House of Quality**. The Harvard Business Review, vol 3. 1988.
- [HON81] HONG, S. e NIGAM, R. **Analytic Hierarchy Process Applied to Evaluation of Financial Modeling Software**. Proceedings of the 1st International Conference on Decision Support Systems. Atlanta, EUA. 1981.
- [ISO12119] ISO/IEC 12119 **Information Technology: Software Packages – Quality requirements and testing**. Geneva, 1998.
- [ISO9126] ISO/IEC 9126 **Standard for Information Technology, Software Product Evaluation - Quality Characteristics and Guidelines for their use**. Geneva, 1991.
- [JAC93] JACKSON, M. e ZAVE, P. **Domain Descriptions**. 1st International Symposium on Requirements Engineering. San Diego, EUA. 1993.
- [JAC95] JACKSON, M. **Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices**. 1ed. Addison-Wesley. Massachusetts, USA. 1995.
- [JAC99] JACKSON, P. **Introduction to Expert Systems**. Addison-Wesley, 3rd edition, Janeiro 1999.
- [KAR96] KARLSSON, K. e RYAN, K. **Software Requirements Prioritizing**. 2nd International Conference on Requirements Engineering. Apr. 1996. pp. 110 – 116.
- [KAR98] KARLSSON, K. **A Systematic Approach for Prioritizing Software Requirements**. Department of Computer and Information Science, Linköping University, Suécia. 1998.
- [KIT96] KITCHENHAM, B. **DESMET: A Method for Evaluating Software Engineering Methods and Tools**. Technical Report TR96-09, Department of Computer Science, University of Keele. Aug. 1996.
- [KIT97] KITCHENHAM, B. JONES, L. **Evaluating Software Reengineering Methods and Tools**. Part 5: The Influence of Human Factors. Software Engineering Notes, 1997.
- [KON95] KONTIO, J. **A COTS Selection Method and Experiences of Its Use**. Proceedings of the 20th Annual Software Engineering Workshop, Maryland, Nov. 1995.
- [KON96a] KONTIO, J. **A Case Study in Applying a Systematic Method for COTS Selection**. Proceedings of the 18th International Conference in Software Engineering. Berlin, Mar. 1996.
- [KON96b] KONTIO, J. CALDIERA, G. e BASILI, V. **Defining Factors, Goals and Criteria for Reusable Component Evaluation**. CASCON'96. Nov. 1996.

- [KOT93] KOTONYA, G e SOMMERVILLE, I. **A Framework for Integrating Functional and Non-Functional Requirements**. International Workshop on Systems Engineering for Real Time Applications. UK. 1993. pp. 148-153.
- [KOT97] KOTONYA, G. e SOMMERVILLE, I. **Requirements Engineering – Processes and Techniques**. John Willy & Sons, 1997.
- [KUN00] KUNDA, D. e BROOKS, L. **Identifying and Classifying Processes (traditional and soft factors) that Support COTS Component Selection: A Case Study**. Proceedings of the ECIS 2000, Vienna University of Economics and Business Administration, 2000.
- [KUN99] KUNDA, D. e BROOKS, L. **Applying Social-Technical Approach for COTS Selection**. Proceedings of the 4th UKAIS Conference, University of York, Apr.1999.
- [LAM00] LAMSWEERDE, A. **Requirements Engineering in the Year 00: A Research Perspective**. 22nd Proceedings of International Conference on Software Engineering. Limerick, Ireland. Jun. 2000.
- [LAM98] LAMSWEERDE, A. DARIMONT, R. e LETIER, E. **Managing Conflicts in Goal-Driven Requirements Engineering**. IEEE Transactions on Software Engineering, Special Issue on Managing Inconsistency in Software Development. Nov. 1998.
- [LEI97] LEITE, J.C.S.P.; ROSSI, G.; MAIORANA, V.; BALAGUER, F.; KAPLAN, G.; HADAD, G.; OLIVEROS, A. **Enhancing a Requirements Baseline with Scenarios** Third IEEE International Symposium on Requirements Engineering - IEEE Computer Society Press. Los Alamitos, Ca, USA. 1997. pp. 44 -53.
- [LIC97] LICHOTA, R. W. e VESPRINI, R. L. **PRISM Product Examination Process for Component Based Development**. Proceedings of the Fifth International Symposium on Assessment of Software Tools. Jun. 1997. pp. 61 –69.
- [LOU95] LOUCOPOULOS, P. e KARAKOSTAS, V. **System Requirements Engineering**. London McGraw-Hill, 1995.
- [LYN97] LYNEX, A. e LAYZELL, P. J. **Understanding Resistance to Software Reuse**. Proceedings of 1997 Software Technology and Engineering Practice, IEEE Computer Society, 1997.
- [MAC96] MACAULY, L.A. **Requirements Engineering**. London, 1996
- [MAC98] MACGRAW, G. e VIEGA, J. **Why COTS Software Increases Security Risks**. Reliable Software Technologies. Sterling VA USA. Jun. 1998.
- [MAI96] MAIDEN, N. e RUGG, G. **ACRE: Selecting Methods for Requirements Acquisition**. Software Engineering Journal, vol 11. n° 3. 1996. pp. 183 – 192.
- [MAI98] MAIDEN, N. **CREWS-SAVRE: Scenarios for Acquiring and Validating Requirements**. Automated Software Engineering, 1998.
- [MAM99] MAMANI M. LEITE, J.C.S.P. **Elicit@99 um Protótipo de Ferramenta para a Elicitacao de Requisitos**. WER99 - Workshop en Requerimientos / Workshop de Engenharia de Requisitos - II. SADIO. Buenos Aires. 1999. PP. 45-55.
- [MCD98] MCDERMID, J. **The Cost of COTS**. IEEE Computer Society. Junho, 1998. Vol 31. N° 6. pp. 46–52.
- [MCL68] MCLLROY, M. D. **Mass-Produced Software Components**. Software Engineering Concepts and Techniques (1968 NATO Conference on Software Engineering) Van Nostrand Reinhold, 1976, pp. 88-98.

- [MEY97] MEYERS, C. e OBERNDORF, P. **Open Systems : The promises and the Pitfalls.** Software Engineering Institute, Carnegie Mellon University, USA. Mai. 1997.
- [MIN92] MIN, H. **Selection of Software: The Analytic Hierarchy Process.** International Journal of Physical Distribution and Logistics Management, vol 22, 1992.
- [MOR00] MORISIO, M. SEAMAN, C. PARRA, A. BASILI, V. KRAFT, S. e CONDON, S. **Investigating and Improving a COTS-Based Software Development Process.** Proceedings of the 22nd International Conference on Software Engineering, Limerick, Ireland. Jun. 2000. pp. 32 – 41.
- [MOR97] MORISIO, M. e CARROLL, J. M. **A methodology for the Evaluation and Selection of Software Products.** Dipartimento di Automatica e Informatica, Politecnico di Torino, Itália. 1997.
- [NCU00a] NCUBE, C. **A Requirements Engineering Method for COTS-Based Systems Development.** PhD thesis . School of Informatics, City University, London. Mai. 2000.
- [NCU00b] NCUBE, C. e MAIDEN, N. **COTS Software Selection: The Need to make Tradeoffs between System Requirements, Architectures and COTS/Components.** Workshop Ensuring Successful COTS Development. Mai. 2000.
- [NCU99] NCUBE, C e MAIDEN, N. A. M PORE: **Procurement-Oriented Requirements Engineering Method for the Component-Based Systems Engineering Development Paradigm.** International Workshop on Component-Based Software Engineering, Mai. 1999.
- [NUS00] NUSEIBEH, B. e EASTERBROOK, S. **Requirements Engineering: A Roadmap.** Proceedings of the 22nd International Conference on Software Engineering. Limerick, Ireland. Jun. 2000.
- [NUS97] NUSEIBEH, B. **Ariane 5: Who Dunnit?** IEEE Software, Mai./Jun. 1997.
- [NWO99] NWOSU, K. C. e SEACORD, R. C. **Workshop on Component-Based Software Engineering Processes.** I Technology of Object-Oriented Languages and Systems (TOOLS) Conference, Aug. 1999.
- [OBE97a] OBERNDORF, P. BROWNSWORD, L e MORRIS, E. **Workshop on COTS-Based Systems.** Software Engineering Institute, Carnegie Mellon University, Special Report CMU/SEI-97-SR-019, Nov. 1997.
- [OBE97b] OBERNDORF, P. **COTS and Open Systems – an Overview** Software Engineering Institute, Carnegie Mellon University, USA. Jan. 1997.
- [OBE97c] OBERNDORF, P. **Facilitating Component-Based Software Engineering: COTS and Open Systems.** Proceedings of the Fifth International Symposium on Assessment of Software Tools. Jun. 1997. pp. 143 –148.
- [OBE98] OBERNDORF, P. e BROWNSWORD, L. **Are You Ready for COTS?** Technical Report, Software Institute Engineering. Aug. 1997.
- [POT95] POTTS, C. **Invented Requirements and Imagined Customers: Requirements Engineering for Off-the-Shelf Software.** 2nd International Symposium on Requirements Engineering. York, Inglaterra. Mar. 1995. pp. 128 –130.
- [POT97] POTTS, C. **Requirements Models in Context.** 3rd International Symposium on Requirements Engineering. Annapolis, EUA. 1997.

- [ROS01] ROSA, N. S. ALVES, C. CUNHA, P. CASTRO, J. JUSTO, G. **Using Non-Functional Requirements to Select Components: A Formal Approach**. Fourth Workshop Iberoamerican on Software Engineering and Software Environment (IDEAS'01), San Jose, Costa Rica, Apr. 2001 (aceito para publicação).
- [RYA93] RYAN, M. **Defaults in Specifications**. Proceedings of IEEE International Symposium on Requirements Engineering. IEEE Computer Society Press. San Diego, EUA. Jan. 1993.
- [SAA90] SAATY, T. **The Analytic Hierarchy Process**. New York: McGraw-Hill, 1990.
- [SAN00] SANTOS, M. **O Uso da Análise de Requisitos como Estratégia para Apoio ao Projeto de Banco de Dados Distribuídos**. Dissertação de Mestrado, Centro de Informática, Universidade Federal de Pernambuco. 2000.
- [SCH98] SCHNEIDER, G. e WINTERS, J. **Applying Use Cases: a practical guide**. Addison-Wesley. 1998.
- [SEA99] SEACORD, R. C. e NWOSU, K. C. **Life Cycle Activity Areas for Component-Based Software Engineering Processes**. Workshop on Component-based Software, Aug. 1999.
- [SEI00] www.sei.cmu.edu
- [SHA96] SHAW, M. e GAINES, B. **Requirements Acquisition**. Software Engineering Journal, vol 11, 1995.
- [SHA99] SHARP, H. FINKELSTEIN, A. e GALAL, G. **Stakeholder Identification in the Requirements Engineering Process**. Workshop on Requirements Engineering Process. Florença, Itália. 1999.
- [SOM96] SOMMERVILLE, I. **Software Engineering**. 5º Edição. Addison-Wesley, 1996.
- [SOU99] SOUSA, F.B. CASTRO, J. F e ALENCAR, F. M **O Impacto dos COTS no processo de Engenharia de Requisitos**. II Workshop on Requirements Engineering. Buenos Aires, Argentina. Set. 1999.
- [STA95] The Standish Group, **Software Chaos**. www.standishgroup.com/chaos.html
- [STD830] IEEE Std. 830. **IEEE Guide to Software Requirements Specification**. The Institute of Electrical and Electronics Engineers. New York, EUA.1984.
- [SZY98] SZYPERSKI, C. **Component Software: Beyond Object-Oriented Programming**. Addison-Wesley, 1998.
- [TOR99] TORANZO, M. e CASTRO, J. **A Comprehensive Traceability Model to Support the Design of Interactive Systems**. International Workshop on Interactive System Development and Object Models - WISDOM 99. Lisboa, Portugal. Jun. 1999.
- [TRA97a] TRAN, V e LIU, D. **A Procurement-centric Model for Engineering Component-based Software Systems**. Proceedings of the Fifth International Symposium on Assessment of Software Tools. Jun. 1997. pp. 70 – 79.
- [TRA97b] TRAN, V e LIU, D. B. **A Risk-Mitigating Model for The Development of Reliable and Maintainable Large-Scale Commercial-Off-The-Shelf Integrated Software Systems**. Proceedings of the 1997 Annual Reliability and Maintainability Symposium, Jan. 1997.
- [VID96] VIDGER, M. R. GENTLEMAN, W. M. e DEAN, J. **COTS Software Integration: State-of-the-Art**. Disponível em www.sel.iit.nrc.ca/abstracts/NRC39198.abs, 1996.

- [VIL99] VILLER, S. e SOMMERVILLE, I. **Social Analysis in the Requirements Engineering Process: from ethnography to method.** 4th International Symposium on Requirements Engineering, Limerick, Ireland. Jun. 1999.
- [VOA97] VOAS, J. **A Defensive Approach to Certifying COTS Software.** Technical Report. Reliable Software Technologies Corporation, Sterling VA USA. Aug. 1997.
- [VOA98] VOAS, J. M. **The Challenges of Using COTS Software in Component-Based Development.** IEEE Computer Society. Vol 31. Nº 6. Jun. 1998. pp 44-45.
- [WAL98] WALLNAU, K.C. CARNEY, D. e POLLAK, B. **How COTS Software Affects the Design of COTS-Intensive Systems.** Software Engineering Institute, Carnegie Mellon University, USA. Jun. 1998.
- [WEI96] WEIRINGA, R. J. **Requirements Engineering – Frameworks for Understanding.** New York, John Wiley e Sons Ltd. 1996
- [WER00] WERNER, C. e BRAGA, R. **Desenvolvimento Baseado em Componentes.** Minicurso, XIV Simpósio Brasileiro de Engenharia de Software. João Pessoa. Out. 2000. pp. 297 – 329.
- [WEY98] WEYUKER, E. **Testing Component-Based Software: A Cautionary Tale.** IEEE Software, Sep. 1998.
- [WIL98] WILLS, A. C. e D’SOUZA, D. F. **Objects, Components and Frameworks with UML: The Catalysis Approach.** Addison Wesley, 1998. Disponível em <http://www.iconcomp.com>
- [YOU98] YOUNG, K. **The MAUT Analysis Case Study.** Disponível em <http://www.emunix.emich.edu/~kcyoung/course/DS605/project/furnace/termpap1.html>
- [YU95] YU, E. **Modeling Strategic Relationship for Process Reengineering.** PhD thesis, Computer Science Department, University of Toronto. Toronto, Canada. 1995.
- [ZAV97] ZAVE, P. **Classification of Research Efforts in Requirements Engineering.** ACM Computer Surveys, vol 29, n º 4. 1997.