Pós-Graduação em Ciência da Computação

“Um Processo para Seleção de Metodologias de Desenvolvimento de Software”

Por

Pedro Luciano leite Silva

Dissertação de Mestrado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

Recife, Fevereiro/2003

Universidade Federal de Pernambuco
CENTRO DE INFORMÁTICA
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Pedro Luciano Leite Silva

"Um Processo para Seleção de Metodologias de Desenvolvimento de Software"

ESTE TRABALHO FOI APRESENTADO À PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO DO CENTRO DE INFORMÁTICA DA UNIVERSIDADE FEDERAL DE PERNAMBUCO COMO REQUISITO PARCIAL PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIA DA COMPUTAÇÃO.

ORIENTADOR: Alexandre Marcos Lins de Vasconcelos

Recife, Fevereiro/2003

Silva, Pedro Luciano Leite

Um processo para seleção de metodologias de desenvolvimento de software / Pedro Luciano Leite Silva. – Recife : O Autor, 2003.

xvii, 129 p. : il., fig.,tab.

Dissertação (mestrado) – Universidade Federal de Pernambuco. CIn. Ciência da Computação, 2003.

Inclui bibliografia e anexos.

1. Software – Desenvolvimento – Seleção de metodologias. 2. Qualidade (Software) – Melhoria. 3. Requisitos de metodologia (Software). 4. Aquisição de software – Seleção de metodologias. 5. Desenvolvimento de software – Metodologias. I. Título.

004.05	**CDU (2.ed.)**	**UFPE**
005.12	**CDD (21.ed.)**	**BC2003-138**

Aos meus pais.

Agradecimentos

Agradeço a todos os que fazem o Centro de Informática da Universidade Federal de Pernambuco, particularmente a professora Judith Kelner e o professor Jaelson de Castro pelo incentivo que me deram para enfrentar o mestrado; a todos os professores; e em especial ao meu orientador, o professor Alexandre Marcos Lins Vasconcelos.

Agradeço a todos os que fazem a Chesf – Companhia Hidro Elétrica do São Francisco, empresa de destaque nacional na área de geração, transmissão e comercialização de energia elétrica, particularmente a toda a hierarquia que me permitiu desenvolver este trabalho e aos amigos do dia a dia.

Resumo

A sobrevivência de uma companhia está relacionada a questões sociais, econômicas e políticas. Ela deve lutar num mercado competitivo que cresce, pressionada pela necessidade de oferecer maior qualidade no desenvolvimento dos produtos, num curto espaço de tempo, de modo a satisfazer seus clientes.

As companhias gastam fortunas em projetos de *software* que são cancelados, que ultrapassam os custos previstos, que são entregues além do tempo estimado e que não atendem às necessidades quando concluídos. Há muitas pesquisas em andamento na tentativa de solucionar estes problemas, principalmente nos campos da tecnologia e de processos de desenvolvimento e gerenciamento de *software*, resultando em novas metodologias disponíveis no mercado.

As organizações de *software* precisam selecionar uma metodologia que suporte o desenvolvimento do projeto, visando a liberação de produtos dentro dos custos e prazos estimados, e com a qualidade exigida pelo cliente.

Este trabalho propõe um processo para selecionar uma metodologia para o desenvolvimento de um projeto de *software*. O processo de seleção se fundamenta em vários trabalhos correlatos, tais como a norma ISO/IEC-14102 e o modelo de Aquisição de *Software* (SA-CMM), proposto pelo *Software Engineering Institute*.

Além do processo de seleção, o trabalho também apresenta o protótipo de uma ferramenta de apoio ao mesmo, partindo do pressuposto que os requisitos da metodologia para o projeto foram definidos e que as metodologias disponíveis foram previamente avaliadas.

Palavras chave: Seleção de Metodologias de Desenvolvimento de *Software*, Processo de Desenvolvimento de *Software*, Qualidade de *Software*.

Abstract

The survival of an organization relates to social, economic and political issues. It needs to struggle in a competitive growing market pressed by the lack of time and aiming to offer greater quality on the products given to the clients.

Organizations spend their Money in software projects which are canceled, go beyond the estimated costs, are released late and do not satisfy the *stakeholder's* needs. There are many researches trying to get a solution to these problems, mainly in technologies and software development processes, and management of software, which result in new methodologies available on the market.

The software organizations need to select a methodology which supports the development of the project, focusing on the release of the products within the estimated costs and time, and with the required customer quality.

This work proposes a process for selecting a methodology for a software development project. The process is based in many related works such as the norm ISO/IEC-1402 and the Software Acquisition (SA-CMM) model proposed by the Software Engineering Institute.

Also, this work presents a prototype of a tool, taking into account that the methodology's requirements were defined and the available methodologies were evaluated.

Keywords: Software Development Methodology Selection, Software Development Process, Software Quality.

Conteúdo

Índice de Figuras

Índice de Tabelas

1. Introdução

Este capítulo discute as principais motivações deste trabalho e define alguns conceitos fundamentais. Também destaca a evolução e classificação das metodologias de desenvolvimento de *software***;** descreve o escopo do trabalho, seus objetivos, a abordagem usada e suas contribuições; e finalmente, apresenta a estrutura da dissertação.

1.1. Motivação

A sobrevivência de uma companhia está relacionada a questões sociais, econômicas e políticas. Para uma companhia sobreviver ela deve lutar num mercado competitivo que cresce, pressionada pela escassez de tempo para desenvolver os projetos e pela necessidade de oferecer maior qualidade[1] nos produtos de modo a satisfazer seus clientes.

Em 1998 o Standish Group publicou um relatório [63] referindo-se ao desenvolvimento de aproximadamente 200 mil projetos de *software* nos Estados Unidos, com as seguintes estatísticas:

- 26% dos projetos foram concluídos no tempo previsto e dentro dos custos esperados, com todas as características e funções originais especificadas;
- 28% dos projetos foram cancelados antes de serem concluídos;
- 46% dos projetos foram concluídos após os prazos previstos, ultrapassaram os custos estimados e apresentaram menos características e funções que as especificações originais.

Numa pesquisa similar, realizada em 1994 [62] no desenvolvimento de 175 mil projetos, apenas 16% deles foram classificados no primeiro item acima, e os dois ítens seguintes apresentaram taxas de 31% e 53%, respectivamente. A melhora do desempenho dos projetos, constatada na pesquisa de 1998, é resultado, principalmente, dos esforços realizados pelas inúmeras pesquisas desenvolvidas em todo o mundo na tentativa de solucionar os problemas mencionados.

Portanto, as tecnologias disponíveis estão se desenvolvendo a passos largos, desde o surgimento da World Wide Web [71] e da programação orientada a objetos. Plataformas como Java [61] e .Net [43], juntamente com XML [72], dispositivos web e wireless utilizam infra-estruturas padronizadas, as quais asseguram uma evolução contínua em direção a facilitar os processos de desenvolvimento e gerenciamento de projetos.

Como resultado das pesquisas e dos avanços tecnológicos, o mercado está cada vez mais agressivo e turbulento [30]. Os clientes de *software* exigem mudanças de forma mais freqüente a fim de manterem posições competitivas no mercado. Para responder às solicitações dos clientes, o desenvolvimento de *software* precisa selecionar uma metodologia de desenvolvimento adequada. Tal metodologia deve incentivar que o

projeto desenvolvido não ultrapasse os custos estimados, seja entregue dentro do prazo previsto e com uma qualidade suficiente de tal forma que satisfaça o cliente.

Segundo Osterweil [50] um processo de *software* é um *software* também (Software Process is a Software Too). Sendo assim, as técnicas existentes para seleção de *software* podem servir de base para a definição de uma técnica para seleção de um processo de *software*.

Selecionar uma metodologia (ou processo de *software*[2]) num mercado tão amplo é uma tarefa considerável, porque isto exige que as necessidades e perspectivas (de mercado) da organização sejam identificadas, e também porque há diversas metodologias disponíveis.

Estas necessidades da organização envolvem a adoção de ferramentas, a identificação das notações que serão usadas, padrões de desenvolvimento, a estrutura da equipe de desenvolvimento, as capacidades das pessoas, treinamentos necessários, custos, os propósitos do projeto, qualidade esperada, entre outros. As perspectivas da organização dependem da sua visão, missão, regras de negócios, estratégias, política e cultura.

O sucesso num desenvolvimento de *software* está baseado numa concentração de esforços, segundo Johnson [35], os quais se estendem desde o suporte executivo, envolvimento com o usuário, gerentes de projeto com experiência, objetivos claros de negócios e um escopo mínimo de projeto.

É desejável para a seleção de uma metodologia de desenvolvimento que se leve em consideração todos os interesses dos *stakeholders*[3] do projeto, bem como os aspectos técnicos e organizacionais.

O uso de uma metodologia adequada é importante para o sucesso de um projeto. Se a gerência não optar por uma metodologia apropriada, o projeto pode estar condenado ao fracasso; e se este processo se repetir, a sobrevivência de toda a organização pode ficar comprometida.

Uma metodologia de desenvolvimento é a chave para as boas práticas do desenvolvimento de *software* pois estabelece ordem nas atividades a fim de possibilitar a

[1] Qualidade é uma propriedade (ou um requisito não-funcional) que o produto deve ter para satisfazer o cliente 53.
[2] Nesta dissertação metodologia e processo de software serão usados como sinônimos, mas há algumas divergências que serão apresentadas na seção 1.2.
[3] *Stakeholder* é qualquer pessoa que tenha algum interesse no projeto, direta ou indiretamente, tais como os usuários, os clientes, o gerente de projeto, os gerentes de outros projetos que afetam ou são afetados pelo projeto, o financiador do projeto e os desenvolvedores.

conclusão de tarefas e/ou objetivos e oferece suporte ao gerenciamento permitindo a repetição do desenvolvimento de *software* para verificação de falhas e problemas que podem ser identificados, corrigidos e evitados. Isto é um aspecto fundamental na implantação de qualidade na organização e está relacionado a normas como, por exemplo, a ISO-9000 [31] e o modelo CMM [8].

Considerando estes aspectos, serão apresentados na próxima seção os principais problemas relacionados à seleção de uma metodologia.

1.2. Problemas Relacionados à Seleção de uma Metodologia

Para selecionar uma metodologia é necessário ter um conhecimento razoável sobre as que estão disponíveis, antes que o processo de seleção seja iniciado. Ler uma documentação disponível pode ajudar a entender uma metodologia, mas certamente só a leitura não é o suficiente para que uma pessoa possa afirmar que domina tal metodologia. Assim, é necessário observar algumas variáveis que só se manifestam durante o uso efetivo da metodologia num projeto.

Também é fundamental um conhecimento da organização que irá adotar a metodologia, bem como ter uma visão clara do mercado a que se destina o produto de *software* a ser desenvolvido, visto que o mercado impõe restrições à cultura da organização e esta influencia de forma decisiva na metodologia e vice-versa. Incompatibilidade entre a cultura da organização e a metodologia pode inviabilizar o seu uso ou prejudicar outros processos organizacionais. Algumas pessoas aceitam e até procuram as inovações enquanto outras detestam mudanças, muitas vezes por insegurança do desconhecido.

Segundo Thomsett [64], um projeto aborda várias considerações de qualidade, dependendo de como cada *stakeholder* enxerga o projeto. Os desenvolvedores seniores e a equipe de desenvolvimento estão preocupados com aspectos técnicos enquanto os gerentes de projetos, os analistas de negócios e os financiadores da metodologia estão preocupados com aspectos gerenciais. Os aspectos técnicos são relacionados com a especificação de funcionalidades, o modelo de dados, a documentação de projeto, os planos de testes, etc. Os aspectos gerenciais lidam com a alocação de recursos, a análise de riscos, as prioridades de projeto, as estimativas de custos, o retorno de investimento, os interesse dos *stakeholders*, e os projetos relacionados. Assim, a metodologia selecionada deve suportar ao máximo possível os interesses de todos os *stakeholders*.

É importante avaliar se o ambiente da organização suporta a metodologia, qual o impacto da metodologia no ambiente, se a metodologia possibilitará que o produto de *software* alcance a posição almejada no mercado, e assim por diante. A equipe de desenvolvimento necessitará de tempo para assimilar a nova metodologia, para conhecer os novos padrões de trabalho, e para se acostumar com toda a mudança.

Antes de implementar uma metodologia na organização como um todo, é aconselhável fazer um processo de transição na organização, de forma incremental. Por exemplo, para selecionar uma metodologia, é conveniente ter um projeto piloto e após a seleção da metodologia, é conveniente aplicá-la apenas em um projeto a fim de consolidar a sua aprendizagem. É fundamental observar o seu impacto na cultura da organização e vice-versa, bem como perceber as ações que foram tomadas para conduzir ao sucesso dos resultados. Se a organização decidir efetuar a transição de forma concorrente, ou seja, aplicar a metodologia em mais de um projeto ao mesmo tempo, isto necessitará de mais pessoas capacitadas na metodologia, e as dificuldades serão maiores, exigindo uma maior capacidade de absorção dos resultados e de comunicação entre as pessoas. Os resultados iniciais são fundamentais para evitar rejeição às mudanças e garantir uma segura aceitação da metodologia pela equipe de desenvolvimento.

Se uma equipe desenvolve um projeto usando uma metodologia inadequada, isto refletirá em dificuldades para o gerente do projeto, para o projetista e para todos os desenvolvedores, podendo resultar num projeto que oferece pouco valor para o negócio e conseqüentemente, não satisfazendo os interesses dos *stakeholders*. Portanto, toda a equipe deve ter conhecimento dos benefícios oferecidos pela metodologia e estar consciente das dificuldades a fim de garantir o sucesso no desenvolvimento do projeto.

Na próxima seção, será apresentada uma visão geral das metodologias de desenvolvimento de *software* a fim de fundamentar o assunto e levantar algumas questões a serem discutidas.

1.3. Visão Geral das Metodologias de Desenvolvimento de Software

Esta seção mostra conceitos importantes que precisam ser introduzidos para o bom entendimento desta dissertação. Ela também apresenta uma breve visão sobre as Metodologias de Desenvolvimento de *Software* (MDS's), seguida de uma classificação daquelas consideradas, neste trabalho, as mais importantes.

1.2.1. Conceitos

O termo metodologia, apresenta divergências quanto ao seu entendimento ou à sua abrangência. Alguns autores utilizam processo como sinônimo, quando se referem a uma metodologia. Contudo, há pelos menos duas abordagens diferentes que convém apresentar.

Henderson-Sellers [28 pp.4] se referem ao termo metodologia (ou método) no contexto de análise de projeto, como uma agregação de pelo menos três elementos: técnicas, processo e a linguagem, associados ao método. Onde o processo é suportado por técnicas com base numa linguagem padrão. Com base nesta definição, pode-se generalizar o conceito de metodologia para qualquer etapa do ciclo de vida (figura 1.1).

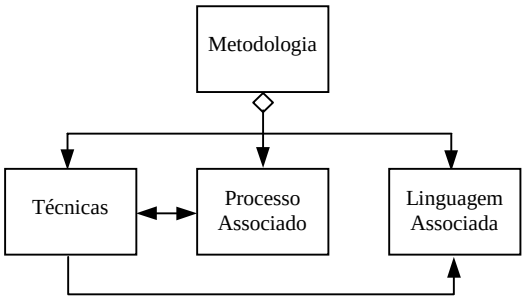


Figura 1. 1 – Os Três Principais Elementos de um Método.

Num foco mais amplo Henderson-Sellers definiram o conceito de Processo de Engenharia de *Software* - PES (figura 1.2) como sendo constituído pelos elementos: metodologia, pessoas / cultura organizacional, e ferramentas / tecnologias.

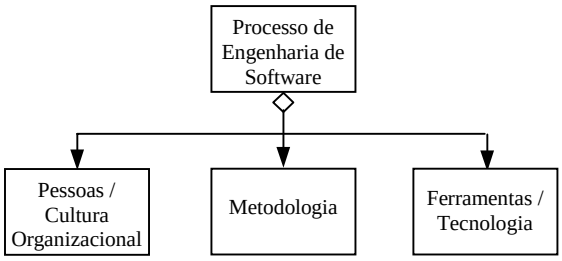


Figura 1. 2 – O Processo de Engenharia de *Software*.

De forma similar, Cockburn [15] define metodologia (figura 1.3), considerando um foco amplo, como constituída de elementos tais como: processo, atividades, técnicas, ferramentas, pessoas, papéis, padrões, valores da equipe (representando a cultura organizacional), entre outros. Para ele metodologia descreve a maneira "como a organização, repetidamente, produz e entrega sistemas". Uma interpretação para esta

figura é que as equipes desempenham papéis conforme suas capacidades para determinadas técnicas, as quais são utilizadas nas atividades para a produção de produtos, seguindo normas e padrões suportados por ferramentas, segundo os valores da equipe, ou seja, conforme a cultura da organização.

O conceito de metodologia proposto por Cockburn apresenta uma analogia à definição de processo engenharia de *software* estabelecido Henderson-Sellers, uma vez que ambos tratam de pessoas, ferramentas, processo, técnicas e cultura organizacional.

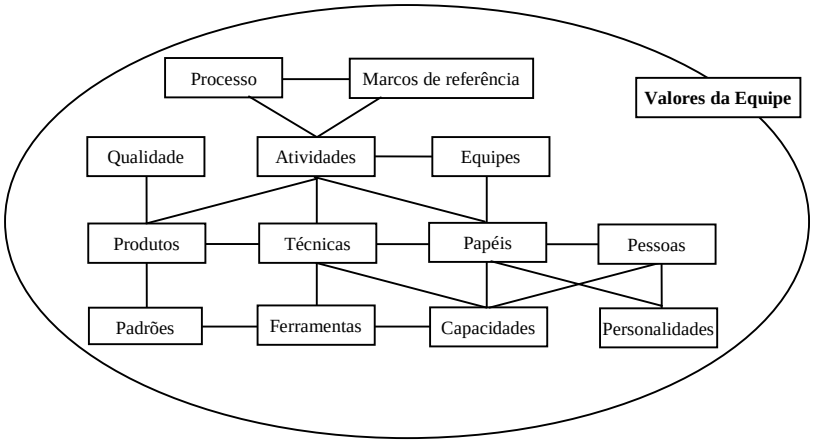


Figura 1. 3 – Elementos de uma Metodologia.

Cockburn ainda alerta que para se comparar metodologias é necessário observar os escopos das mesmas. Por exemplo, a figura 1.4 mostra que o ciclo de vida da metodologia em questão envolve por exemplo requisitos e testes; os papéis envolvem o financiador do projeto, o gerente do projeto, o treinador e as atividades envolvem o desenvolvimento do projeto e o preenchimento de time-sheet. Desse modo, não faz sentido comparar esta metodologia em outras áreas. Cada metodologia apresenta um escopo tanto em relação à extensão do ciclo de vida do projeto coberto pela metodologia, quanto nos papéis desempenhados pelas pessoas, ou ao subconjunto das atividades que as pessoas executam.

Pode-se dizer que ainda não há um conceito uniforme e completo com relação às metodologias de desenvolvimento de *software*, mas se percebe que o conceito apresentado por Henderson-Sellers sobre processo de engenharia de *software* é muito semelhante ao conceito apresentado por Cockburn quando se refere à metodologia num sentido mais amplo [11 pp.78, 14 pp.102].

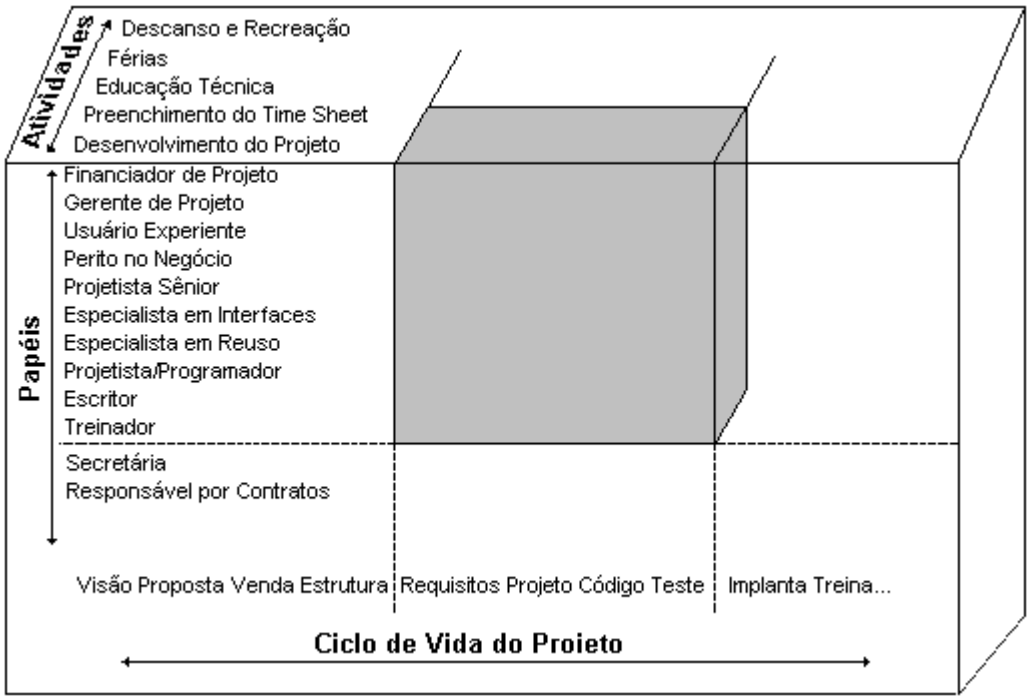


Figura 1. 4 – O Escopo de uma Metodologia.

Sendo assim, neste trabalho, estes termos são considerados como sinônimos.

Na próxima seção, segue uma rápida estória das MDS's.

1.2.2. Evolução das Metodologias de Desenvolvimento de Software

A evolução das MDS's está baseada num amplo esforço que combina técnicas para o entendimento dos requisitos; uso de tecnologias, ferramentas e padrões; técnicas para o gerenciamento e o desenvolvimento do produto de *software*.

Outras contribuições para as MDS's surgiram de métodos relacionados à orientação a objetos os quais podem ser classificados em três gerações.

Segundo Henderson-Sellers [26 pp.1] e Webster [67], a primeira geração, observada até 1992, se caracteriza por métodos inventados por indivíduos ou pequenos grupos de trabalho de forma isolada, tais como o método de Coad [9, 10] denominado OOA (Object Oriented Analysis), e o método de Booch [5] aplicado a OO.

Segundo Henderson-Sellers [26 pp.1], em 1994 surgiram os métodos da segunda geração, formados por coleções de outros métodos re-arrumados de modo ligeiramente diferente tais como MOSES [45], SOMA [59], Firesmith [22] e OMT [54].

Os métodos da terceira geração ao invés de re-arrumar, integraram outros métodos, têm o foco num processo de desenvolvimento de *software* mais amplo e foram inventados por

grandes grupos de forma colaborativa [27]. Como exemplos de métodos da terceira geração têm-se o Unified Method (Rumbaugh 1996 [55], atualmente conhecido como RUP– Rational Unified Process [38]), o método OPEN (atualmente o Consórcio OPEN) [47], e Catalysis [6, 19].

A próxima seção apresenta uma classificação das metodologias de desenvolvimento de *software*.

1.2.3. Classificação das Metodologias de Desenvolvimento de Software

De certa forma, os conceitos apresentados na seção 1.2.1 indicam a direção do caminho seguido por Henderson-Sellers e por Cockburn, uma vez que os primeiros deram ênfase a processo enquanto que o último se voltou para metodologia. Isto não quer dizer que uma direção esteja certa e a outra errada, mas mostra que são abordagens diferentes para tratar os problemas.

Henderson-Sellers [27] concluíram que cada projeto exige seu próprio processo e, como resultado, a tendência é a padronização das técnicas de modelagem visual e outros produtos que permitem grande visibilidade para a gerência do projeto. Devido à padronização, o processo de desenvolvimento torna-se mais independente dos desenvolvedores. Isto é bom para o gerente do projeto, uma vez que fica mais fácil controlar o processo. Por outro lado, produtos tais como a proposta de projeto, o documento de especificação de requisitos e o plano de testes, que permitem visibilidade para o gerenciamento, acarretam em processos mais pesados e demandam um maior número de pessoas para completar as tarefas que resultam naqueles produtos.

Cockburn [15] procurou dar ênfase na metodologia, tornando os processos mais leves e conciliando a necessidade dos projetos com as características de trabalho do ser humano. Com isto, o processo torna-se menos visível e há necessidade de maior comunicação entre as pessoas.

Na próxima sub seção, é apresentada uma classificação que divide as metodologias em pesadas[4] e ágeis[5].

[4] Também definida como metodologia grande, burocrática, densa, ou de cerimônia.
[5] Também definida como metodologia pequena, leve, ou adaptável.

Metodologias Pesadas

Uma **metodologia pesada** é usada por organizações que exigem um processo com uma comunicação baseando-se numa abordagem formal [28, pp.8], uma vez que elas geralmente lidam com projetos cujos requisitos têm que ser bem definidos, e com projetos críticos que envolvem risco de vida. Estes processos são executados com rigor (ideal para organizações com estruturas e culturas de controle) e geralmente suportam projetos acima de doze pessoas.

As metodologias pesadas são projetadas para possibilitar a padronização das ações das pessoas diante de projetos ou situações semelhantes, aumentando assim a produtividade da organização. Exemplos de metodologias pesadas são RUP, OPEN e Catalysis.

Metodologias Ágeis

Uma **metodologia ágil** é aquela que é usada pela organização na qual seu processo dá ênfase à colaboração, baseada numa abordagem flexível, visto que ela lida com projetos nos quais os requisitos mudam constantemente, dependendo das questões de mercado, necessidades da organização, propósitos do projeto, e domínio de conhecimento.

As metodologias ágeis [24] propõem uma comunicação muito próxima com o cliente, uso de menos produtos intermediários que são concluídos com poucas pessoas da equipe de desenvolvimento. Logicamente, isto impõe uma cultura bem mais dinâmica e exige muito mais capacitação de cada membro, uma vez que basicamente todos participarão de quase todas as fases do desenvolvimento de um projeto.

Como exemplos de metodologias ágeis tem-se XP [73, 74], *Crystal* [17], *SCRUM* [60], *Feature Driven Development* [21] e *Dynamic Systems Development Method* [18].

Em resumo, os conceitos de metodologia e processo de engenharia de *software* serão tratados como sinônimos; uma metodologia deve ser considerada observando-se o seu escopo; a evolução das metodologias continua, tendo um grande avanço com a orientação ao objeto; e finalmente as metodologias são classificadas como **pesadas** e **ágeis**.

A próxima seção apresenta o escopo, os objetivos e contribuição da dissertação.

1.4. O Escopo, Objetivos e Contribuição da Dissertação

O principal objetivo desta dissertação é propor um processo para selecionar uma metodologia para o desenvolvimento de *software* de um projeto.

Esta dissertação está baseada em algumas metodologias estudadas, as quais serão apresentadas no capítulo 2; na abordagem desenvolvida por Oakes [46] sobre a seleção de ferramentas CASE; nas estratégias de seleção de ferramentas Case da norma ISO/IEC-14102 [70]; na avaliação dos processos de *software* da futura norma ISO/ICE-15504 [32]; no modelo de Aquisição de *Software* SA-CMM [16] e nas publicações de Jim Higsmith [30], Alistair Cockburn [11, 12, 13, 14, 15], Clenio Salviano [57], Rob Thomsett [64] e Karina Alves [1].

A contribuição desta dissertação consiste na elaboração de um processo para selecionar uma metodologia para o desenvolvimento de *software* de um projeto. O processo de seleção leva em conta os interesses dos *stakeholders* os quais são originados da cultura da organização e prioridades do projeto. Além disso, o processo deve considerar os objetivos da organização no mercado, observar a criticidade do sistema, e averiguar a demanda do potencial humano envolvido, uma vez que todas estas questões afetam direta ou indiretamente os interesses dos *stakeholders*.

Os interesses dos *stakeholders* formam a base dos requisitos da metodologia procurada para o projeto. Uma vez determinados os interesses dos *stakeholders*, pode-se buscar identificar as metodologias que suportem estes interesses, ou seja, selecionar as metodologias que apresentam determinadas características que propiciam um melhor desenvolvimento do projeto.

Uma parte do processo de seleção proposto pode ser automatizada por uma ferramenta (MeSTool), a qual teve um protótipo implementado como parte da pesquisa relacionada à dissertação. Esta automação supõe que os requisitos da metodologia foram identificados.

Na próxima seção, é apresentada a estrutura da dissertação.

1.5. Estrutura da Dissertação

Além deste capítulo introdutório, este trabalho consiste de mais cinco capítulos que são descritos nesta seção.

Capítulo 2 – Metodologias de Desenvolvimento de *Software*

Este capítulo discute duas metodologias pesadas, OPEN e RUP, e duas metodologias ágeis, Crystal e XP. A escolha de tais metodologias para discussão nesta dissertação decorre dos seguintes fatos:

- RUP é uma das metodologias pesadas mais elaboradas, divulgadas, e usadas na atualidade [7].
- OPEN é uma das metodologias pesadas que trata todo o ciclo de vida [27] de um produto.
- XP é a metodologia ágil mais discutida nos últimos anos e também está difundida [7] em várias organizações de desenvolvimento de *software*.
- Crystal é uma das metodologias ágeis que, diferentemente de XP, se propõe a ser uma metodologia tolerante com relação à forma de trabalho do programador.

Capítulo 3 – Bases Conceituais para o Processo de Seleção de Metodologias

Este capítulo apresenta a motivação deste trabalho, aborda alguns problemas relacionados à seleção de uma metodologia e também alguns trabalhos correlatos, que foram utilizados como fundamentos para o processo de seleção de metodologias.

Capítulo 4 – Processo Proposto para Seleção de Metodologias

Este capítulo mostra o processo proposto que constitui a maior contribuição desta dissertação. O mercado, a cultura e o projeto, entre outros, são usados para determinar os interesses dos *stakeholders*. Estes interesses são utilizados na identificação das características das metodologias. Estas características permitem fazer comparações entre as metodologias bem como selecionar a candidata para o desenvolvimento de um projeto de *software*.

Capítulo 5 – O Protótipo MeSTool

Este capítulo apresenta um protótipo que automatiza parte do processo proposto a partir das características das metodologias. Este protótipo permite ao usuário cadastrar metodologias, cadastrar características das metodologias e selecionar uma metodologia cadastrada para um projeto.

Capítulo 6 – Conclusões e Trabalhos Futuros

Este último capítulo apresenta as conclusões e contribuições desta dissertação e mostra as possibilidades de continuação de trabalhos em outras pesquisas nesta área.

2. Metodologias de Desenvolvimento de *Software*

Este capítulo apresenta uma visão geral do estado atual das metodologias de desenvolvimento de *software*, que foram consideradas neste trabalho de pesquisa. São destacados os principais pontos de cada metodologia, tais como sua arquitetura, princípios e técnicas utilizados. Em seguida, discutem-se as semelhanças e diferenças entre as metodologias RUP, OPEN, XP e Crystal.

2.1. Alguns Exemplos de Metodologias de Desenvolvimento de Software

Nesta seção serão discutidas as metodologias RUP, OPEN, Crystal, e XP.

2.1.1. RUP

O Rational Unified Process [33, 56], desenvolvido pela Rational Software [51], é baseado em muitas das melhores práticas do desenvolvimento de *software* e apresenta estas práticas como um *framework* de processos, os quais atendem a uma grande variedade de projetos e organizações.

Este *framework* (figura 2.1) estabelece um processo centrado na arquitetura, baseado em componentes, e dirigido por casos de uso, através de Fluxos de Processo realizados por diversos responsáveis. Estes fluxos de processo são constituídos de atividades e artefatos que produzem o *software*. O desenvolvimento de *software* está organizado em duas dimensões: horizontal e vertical.

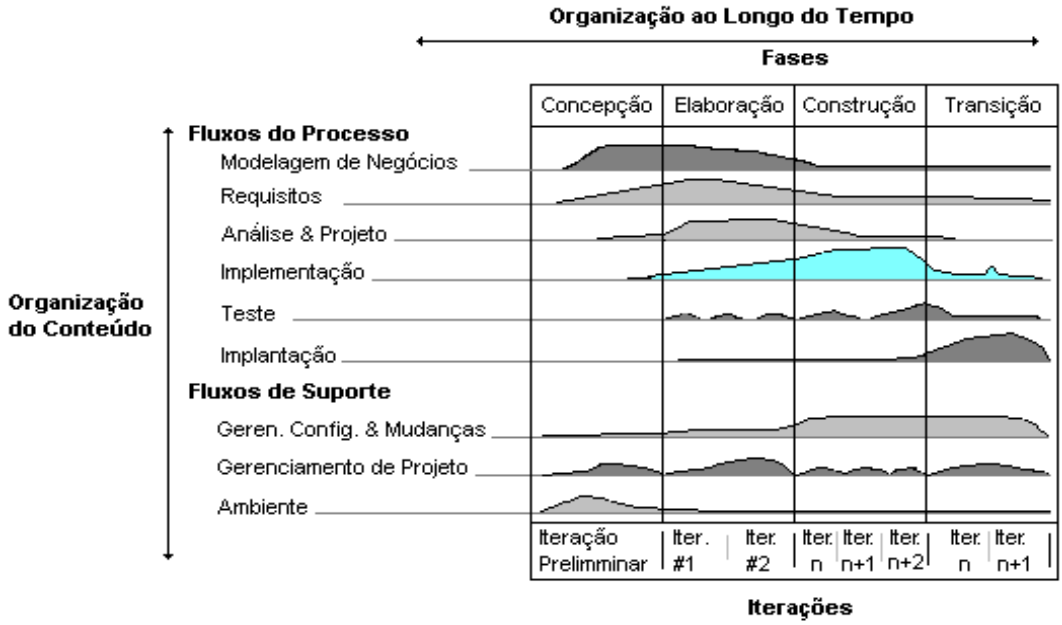


Figura 2. 1 – A Arquitetura do RUP com Duas Dimensões.

A dimensão horizontal apresenta o tempo e trata de aspectos dinâmicos do processo, expressos em termos de fases (Concepção, Elaboração, Construção e Transição), iterações, e marcos de referência.

A dimensão vertical apresenta os fluxos de processo e de suporte, e trata de aspectos estáticos do processo, descritos em termos de componentes de processo (atividades,

artefatos, e papéis), que logicamente agrupam as atividades de engenharia de *software* de acordo com sua natureza.

Os fluxos de processo [39] são: modelagem de negócios, requisitos, análise e projeto, implementação, teste e implantação. Os fluxos de suporte são gerenciamento de configuração e mudanças, gerenciamento de projeto, e de ambiente.

No RUP, um produto de *software* é projetado e construído numa sucessão de iterações incrementais. Isto permite testar e validar idéias de projeto, bem como reduzir os riscos, o mais cedo possível no ciclo de vida do produto.

Um conceito chave do RUP é o de instância de desenvolvimento [40], o qual é uma instância customizada do processo para um projeto específico. Esta descrição de processo define e identifica:

- O que será desenvolvido;
- Quais artefatos são realmente necessários;
- Quais templates devem ser usados;
- Quais artefatos já existem;
- Quais papéis serão necessários;
- Quais atividades serão executadas;
- Quais linhas mestras, padrões de projeto, e ferramentas serão utilizadas.

A instância de desenvolvimento é usualmente produzida pela:

- Seleção dos elementos relevantes no *framework* do RUP;
- Eliminação dos elementos de processos não necessários;
- Adição de qualquer elemento específico da organização, domínio ou tecnologia;
- Customização de alguns elementos para o contexto do projeto.

O fato do processo ser dirigido por casos de uso [34, 53] permite expressar o comportamento detalhado do sistema, o qual pode ser facilmente compreendido pelos usuários bem como pelos desenvolvedores.

A partir das necessidades dos *stakeholders*, é desenvolvido um documento de visão que contém características de alto nível do sistema (figura 2.2). Estas características são transformadas em requisitos de *software*, detalhados num nível que permita que os mesmos sejam projetados e implementados, bem como que sejam identificados casos de testes para o sistema. Estes requisitos detalhados são capturados no modelo de casos

de uso e em especificações complementares onde os requisitos não-funcionais[1] [1, 37, 53] são documentados. Com base nesses recursos e observando-se a estabilidade da arquitetura são elaborados, de forma iterativa, os modelos de projeto, os modelos de testes, a implementação, a documentação do usuário e o material de treinamento.

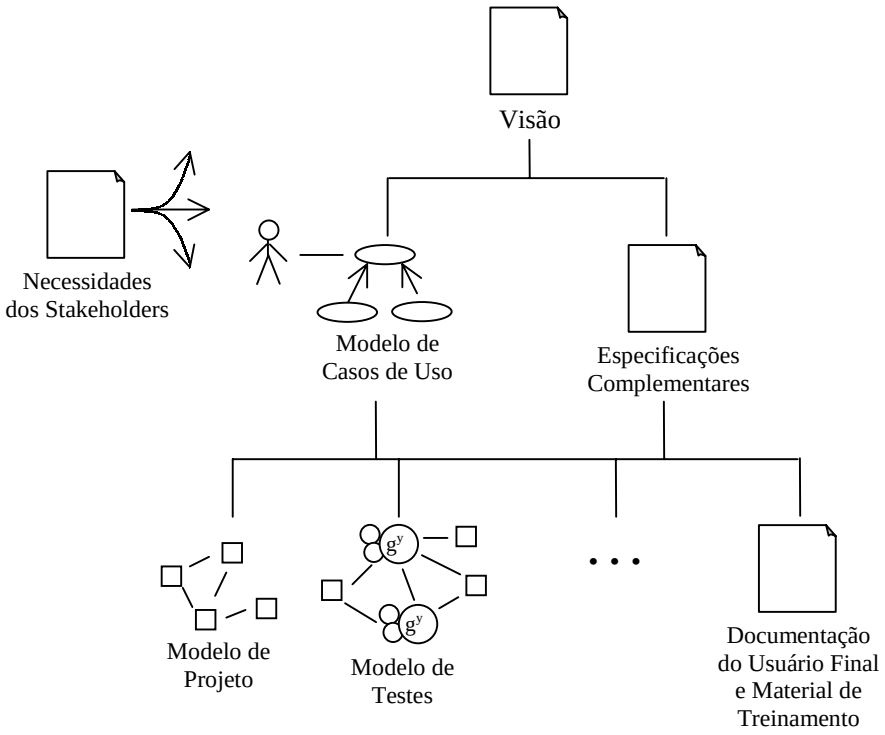


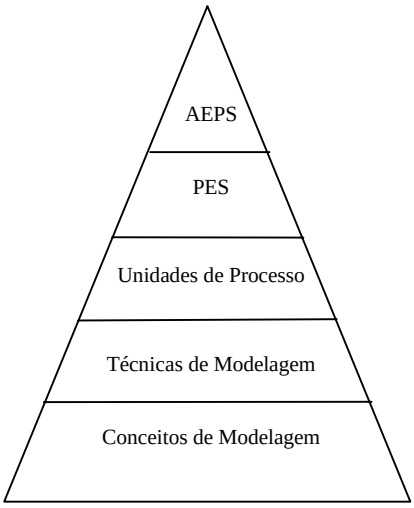
Figura 2. 2 – Principais Artefatos do RUP.

2.1.2. OPEN

A sigla OPEN significa Notação, Ambiente e Processo Orientados a Objeto (*Object-oriented Process, Environment and Notation*). OPEN usa um *framework* de processos, o qual se constitui de uma Arquitetura de Engenharia de Processo de *Software* (AEPS) que é uma generalização dos Processos de Engenharia de *Software* (PES). Como mencionado pelos autores Graham et. all [26], PES "é um conjunto de atividades seqüenciais e temporizadas que transformam os requisitos dos usuários em *software* e fornecem uma abordagem bem definida e testada para o desenvolvimento de sistemas OO". O processo de desenvolvimento OPEN consiste de atividades, e entrega de produtos por meio de conclusão das tarefas.

[1] Os requisitos não-funcionais são aqueles relacionados à qualidade ou restrição que um produto ou serviço deve apresentar para satisfazer às necessidades dos *stakeholders*.

Estas tarefas [26, 42] (figura 2.3) são realizadas através de técnicas de modelagem e são suportadas pelos conceitos de modelagem esquematizados. Esta figura pode ser entendida como uma arquitetura que suporta processos de engenharia de *software* os quais são formados por unidades de processo, através de técnicas baseadas em conceitos de modelagem.



Arquitetura de Engenharia de Processo de Software:
É um framework para guiar o desenvolvimento dos "Processos de Engenharia de Software"

Processo de Engenharia de Software:
É um conjunto de atividades executadas de forma seqüencial, num modelo de ciclo de vida específico.

Unidades de Processo - São conjuntos de atividades industriais que transformam entradas em saídas. e.g. modelagem de requisitos

Técnicas de Modelagem - São conjuntos de práticas e princípios para desenvolver alguma saída. e.g. modelagem de objetos, técnica CRC, Análise de Cenários

Conceitos de Modelagem – São conceitos fundamentais que suportam as técnicas de modelagem. e.g. conceitos de orientação a objetos tais como abstração, etc.

Figura 2. 3 – O Conceito Fundamental que Suporta o Process e-MeNtOR e o Próprio OPEN.

O *framework* de processos do OPEN é um modelo definido num meta-nível que consiste de atividades onde cada uma corresponde a vários trabalhos, que precisam ser executados. Estes trabalhos podem ser decompostos em tarefas, as quais podem ser executadas num curto espaço de tempo e podem ser identificados tanto como "completa" ou "incompleta". Como as tarefas precisam ser executadas por alguém (ou pelo *software*), é útil introduzir uma meta-classe denominada "Desempenho de Tarefa", a qual consiste de uma tarefa e seu executor. Desse modo, segundo Henderson-Sellers [28 pp. 9-10] (figura 2.4) no meta-modelo do OPEN, se diz que uma atividade consiste de desempenhos de tarefas diversas. Um fluxo de processo é então uma quantidade de atividades, conhecida como uma seqüência de desempenho de tarefa.

A fim de completar as tarefas, o "desempenho de tarefas" é apoiado pelo desenvolvedor num papel específico de uma ou mais técnicas onde cada uma contribui parcialmente ou de forma completa para um produto. Os produtos são geralmente concluídos no final das atividades, o que explica, na figura 2.4, a ligação entre atividade e produto. Em outras palavras, o PES consiste explicitamente de atividades, produtos, técnicas e executores e implicitamente de tarefas. A seqüência é conseguida adicionando asserções a cada tarefa. Estas asserções descrevem o contrato das atividades e tarefas objetivadas no

ciclo de vida o qual pode ser mais bem descrito por um modelo dirigido por contrato. Por sua vez, os produtos evoluem de tal forma que necessitam de um controle de versão.

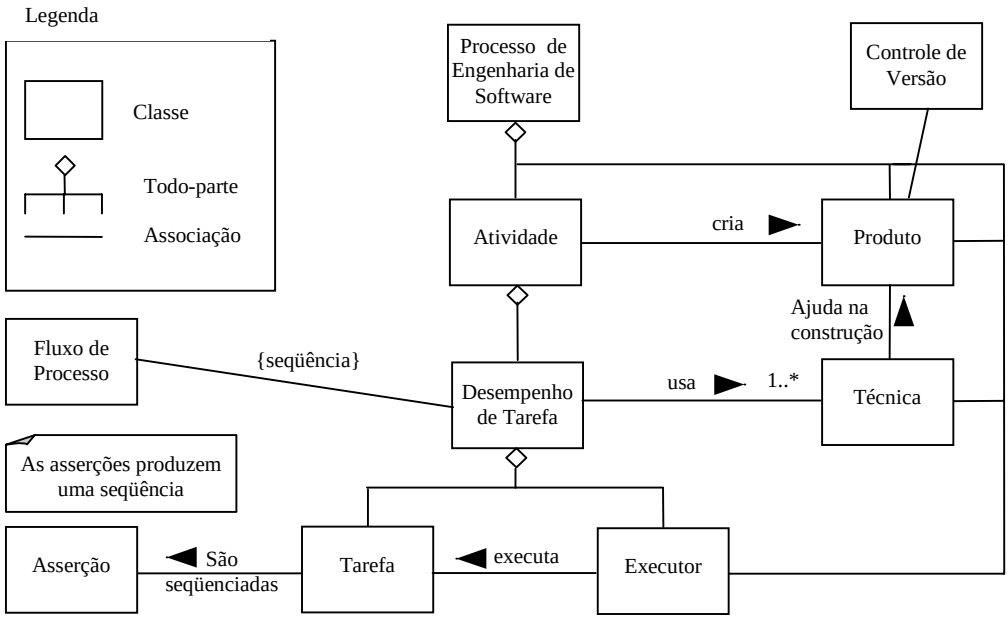


Figura 2. 4 – Atividades, Tarefas e Técnicas.

De acordo com Henderson-Sellers [28 pp. 4-5] (figura 2.5) OPEN trabalha com três níveis de processos: o Processo de Ciclo de Vida do Produto, o Processo de Engenharia de *Software* e o Processo Associado.

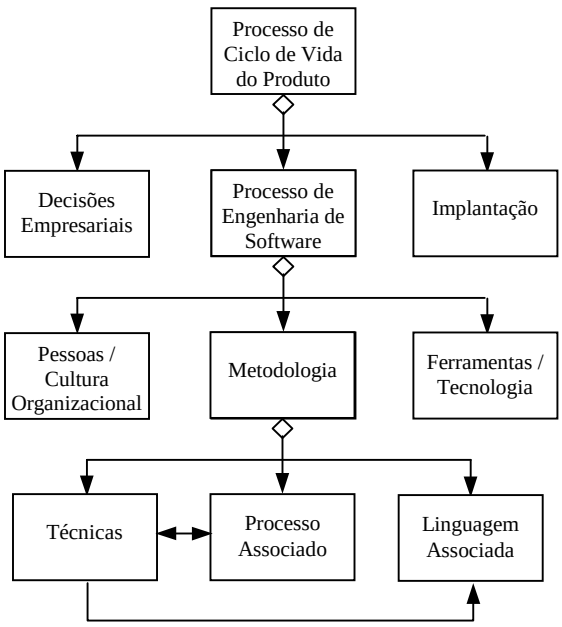


Figura 2. 5 – O Processo do Ciclo de Vida do Produto.

Este processo de ciclo de vida do produto vai além do domínio do *software* [28 pp.6] (figura 2.6), pois descreve como um negócio funciona em três estágios: o planejamento do negócio, a construção (que usa o modelo dirigido por contrato) e sua a implantação no cliente.

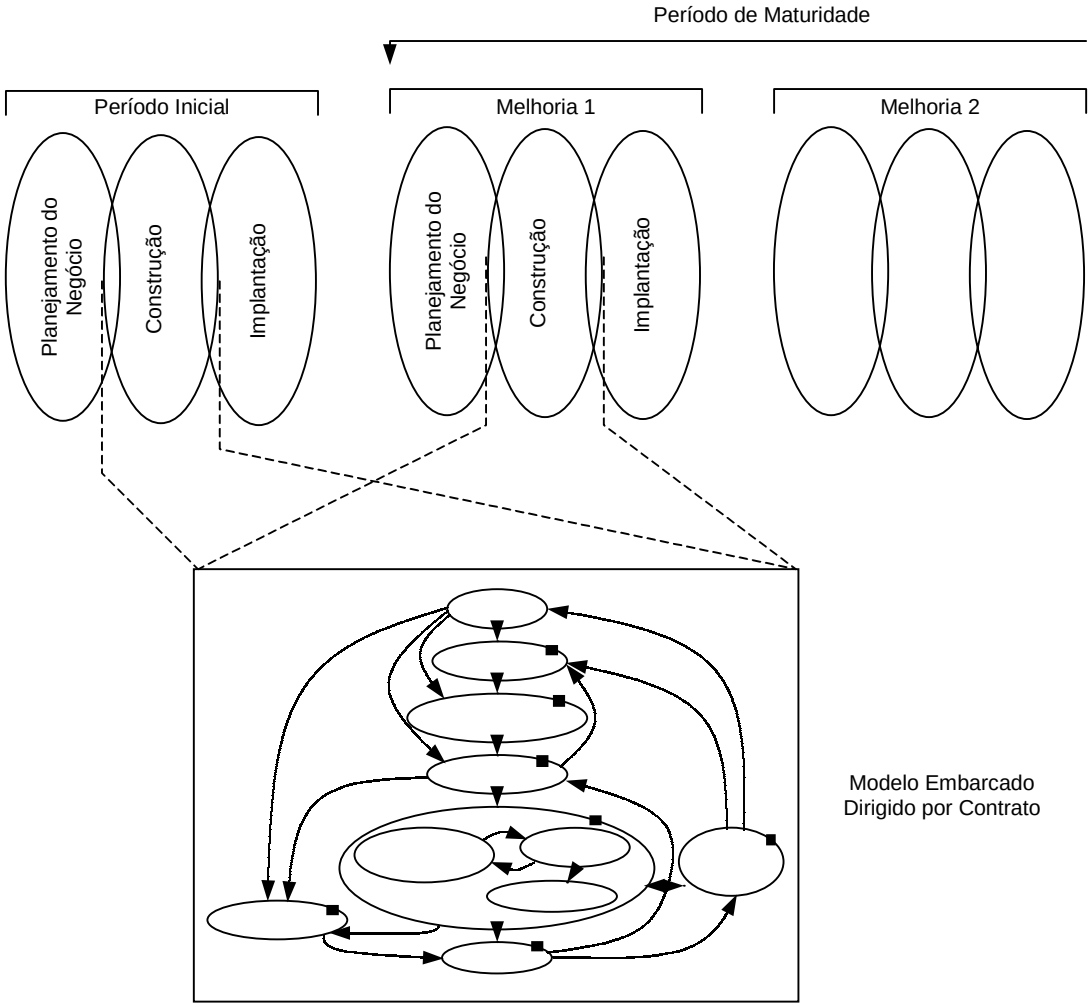


Figura 2. 6 – O Ciclo de Vida do Produto.

O modelo de ciclo de vida dirigido por contrato apresentado na figura 2.7 por Henderson-Sellers [28 pp. 12] permite que um programa de desenvolvimento seja decomposto em projetos os quais produzirão produtos de *software* disponíveis para serem entregues. Os projetos são compostos de atividades (representadas por elipse na figura 2.7), as quais permitem uma estrutura de gerenciamento do projeto, para o desenvolvimento do produto, com criação de *builds*[2] e *releases*[3].

[2] Uma versão executável de um projeto de software.
[3] Um build a ser liberado para uso.

Cada atividade apresenta um conjunto de metas e é modelada como um objeto, cujos métodos são as tarefas e possui pré e pós-condições que satisfazem as obrigações contratuais. Estas atividades estão conectadas umas às outras por linhas que indicam as transições potenciais que um desenvolvedor do processo pode fazer, mas somente quando as pós-condições da atividade atual estão atendidas. Uma vez atendidas estas pós-condições, a equipe de desenvolvimento pode seguir para outra atividade, novamente assumindo que as pré-condições da próxima atividade são satisfeitas.

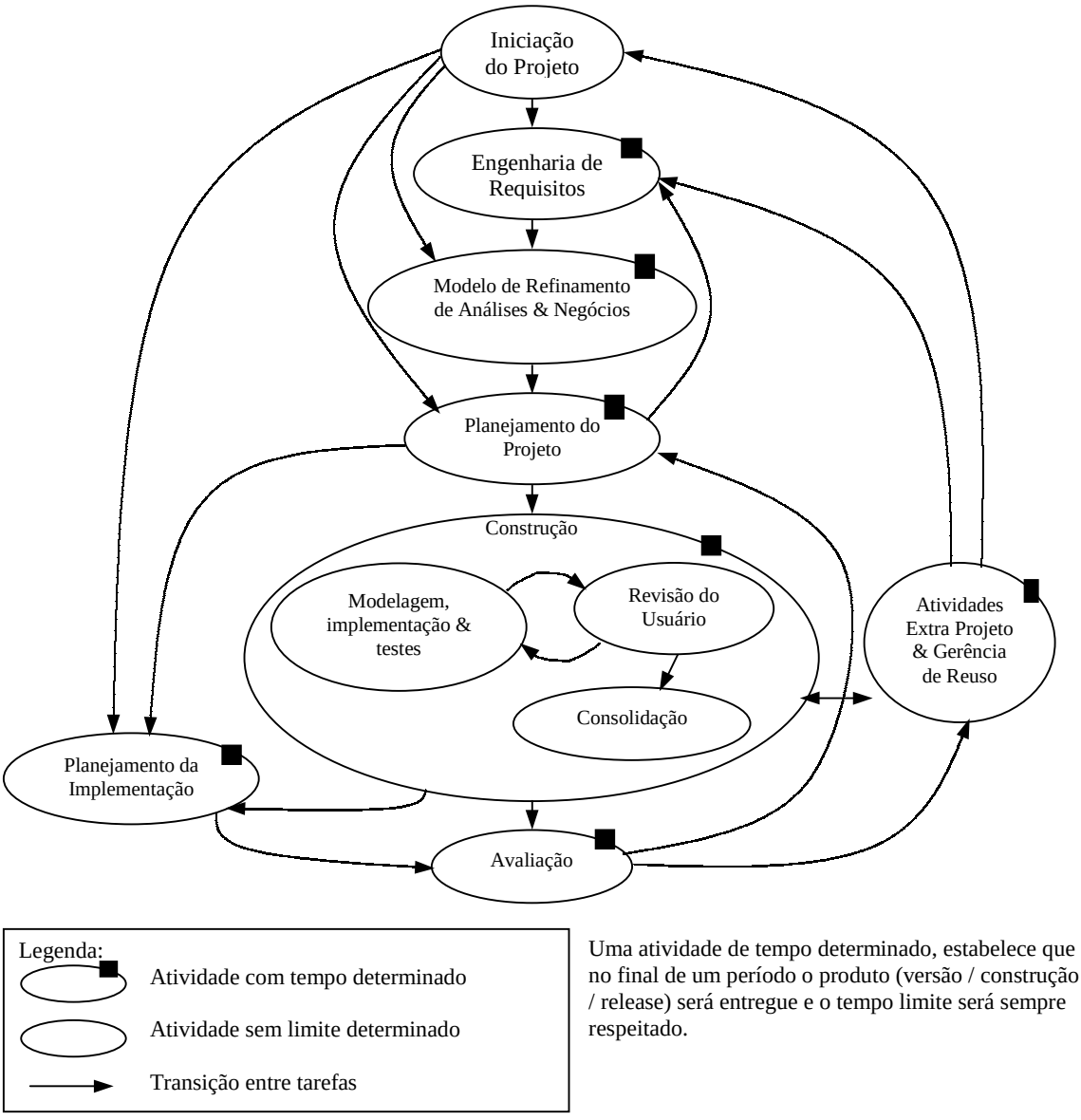


Figura 2. 7 – O Ciclo de Vida Dirigido por Contrato para Múltiplos Projetos.

Por exemplo, as tarefas relacionadas à atividade de construção da figura 2.7 são, conforme Henderson-Sellers [28 pp. 14], apresentadas na tabela 2.1

Tarefas	
A	Codificação
B	Construção do modelo de objetos
C	Avaliação da qualidade
D	Identificação de classes, objetos, tipos e papéis
E	Mapeamento dos papéis em classes
F	Aplicação dos Testes
G	Documentação

Tabela 2. 1 – As Tarefas Relevantes à Atividade de Construção.

Um elemento principal de OPEN é a sua relação bidimensional entre Tarefas e Técnicas. A conclusão de uma tarefa pode exigir a aplicação de uma ou mais técnicas e estas podem ser aplicáveis a várias tarefas. Considerando as seguintes técnicas [27]:

- 01 – Modelagem dos relacionamentos;
- 02 – Implementação da estrutura;
- 03 – Projeto interno das classes;
- 04 – Inspeções;
- 05 – Mapeamento para banco de dados.

Cada combinação de tarefa e técnica é identificada conforme sua ocorrência em:

- Mandatória (M)
- Recomendada (R)
- Opcional (O)
- Desencorajada (D)
- Proibida[4] (F).

Por exemplo, a tabela 2.2 [28 pp.15] mostra que para a tarefa A, são mandatárias as técnicas 01, 02 e 05, a técnica 03 é recomendada e a 04 é opcional.

Técnicas	**Tarefas**								
	A	**B**	**C**	**D**	**E**	**...**	**...**	**...**	**...**
01	M	R	D	O	O	...	...	...	...
02	M	D	F	F	D	...	...	...	...
03	R	R	D	O	O	...	...	...	...
04	O	O	M	O	M	...	...	...	...
05	M	O	D	O	D	...	...	...	...
...	...	...	...	...	...	...	...	...	...

Tabela 2. 2 – Um Ciclo de Vida de Customização de Tarefas em OPEN.

[4] Do inglês Forbidden.

2.1.3. Crystal

Crystal [11] usa a abreviatura PARTS para se referir à ***P****recisão*, exatidão (***A****ccuracy*), ***R****elevância*, ***T****olerância* e escala (***S****cale*) no gerenciamento do desenvolvimento de *software* e na entrega dos produtos.

Precisão é o quanto se deseja falar sobre um determinado item (a quantidade de detalhe) relacionado ao que vai ser entregue, ou relacionado ao que alguém deseja dizer. Para cada item há três versões de precisão: baixa, média e alta. *Exatidão* é o quão correto se deve ser com referência a um item, ou seja, *exatidão* alimenta o processo iterativo, pois os detalhes não são lembrados de imediato e para se chegar a um produto (intermediário) exato é necessário mais trabalho (ou mais iterações), tornando o processo mais lento. *Relevância* define quais itens, e como, serão abordados numa discussão. *Tolerância* é o quanto de variação, de um determinado item, pode ser permitido. *Escala* representa quantos itens do sistema são agrupados em uma única visão, num determinado momento.

Precisão, exatidão e também escala são enfatizados para o gerenciamento do desenvolvimento de *software* e para a entrega dos produtos. Por exemplo, um plano de projeto numa versão de precisão baixa mostra os itens fundamentais a serem produzidos, suas dependências e quais itens devem ser distribuídos juntos. Um plano de projeto numa versão de precisão média mostra as dependências entre as equipes e as respectivas datas de previsão de conclusão; isto requer mais trabalho para estabelecer os detalhes. Um plano de projeto numa versão de precisão alta é muito detalhado, mostrando suas tarefas relacionadas com o tempo previsto, atribuições, e dependências; isto exige ainda mais horas de trabalho.

Crystal baseia-se num desenvolvimento incremental através da seqüência de validação-V, esquematizado na figura 2.8.

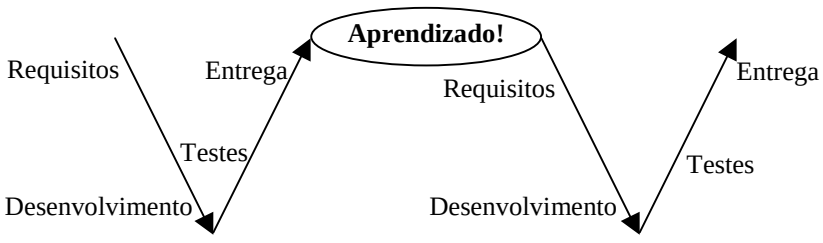


Figura 2. 8 – Desenvolvimento Incremental de um Projeto.

Ou seja, para fornecer qualquer parte do sistema é necessário primeiro o mesmo ser testado; e para testar ele deve primeiro ser projetado e desenvolvido; e o desenvolvimento ocorre após a elicitação dos requisitos. Quando ocorre a entrega, é dada ênfase ao que foi aprendido através das dificuldades superadas, ressaltando os pontos importantes e positivos que levaram ao sucesso alcançado, de forma que na próxima seqüência, ou seqüências futuras, estes pontos sejam levados em consideração.

Cockburn [11 pp.120-128] enfatiza que a validação-V, também referida como uma plataforma V-W, é diferente do desenvolvimento em cascata, o qual tem apenas um V, ou seja, consiste de requisitos, desenvolvimento, testes e entrega uma única vez.

Segundo Cockburn, “A plataforma V-W é uma estratégia geral que inclui desenvolvimento incremental e iterativo, espiral, e estratégias de prototipação”. Ela possibilita ao gerente de projeto:

- Distribuir as atividades de tal modo que as elas se acomodem nos tempos previstos no projeto.
- Arranjar os protótipos e iterações sem sacrificar a noção de progresso contínuo.
- Derivar indicadores de progresso efetivos baseados em incrementos entregue.
- Gerenciar os sub contratados através dos incrementos entregue.

A plataforma V-W é usada para um gerenciamento de alto nível no qual um segmento do sistema é separado em incrementos. Um possível cenário seria o seguinte, conforme destacado por Cockburn [11 pp. 121] (figura 2.9):

1) O financiador do projeto exige que um segmento do sistema esteja pronto nos próximos 10 meses.

   Representado pela parte do incremento visível para o financiador (corresponde ao trecho acima da linha tracejada). O financiador não se interessa em saber dos detalhes do desenvolvimento projeto, mas mantém (na sua agenda) o dia da entrega do produto.

2) O gerente do projeto mantém o controle do mesmo junto ao grupo de desenvolvimento.

   Dividindo o segmento do sistema em incrementos visíveis de quatro, três, e três meses, respectivamente para a equipe de desenvolvimento (corresponde ao trecho abaixo da linha tracejada), permite que o gerente acompanhe o projeto de forma mais efetiva e em intervalos de tempos

menores o que lhe possibilita tomar decisões corretivas para o sucesso do projeto.

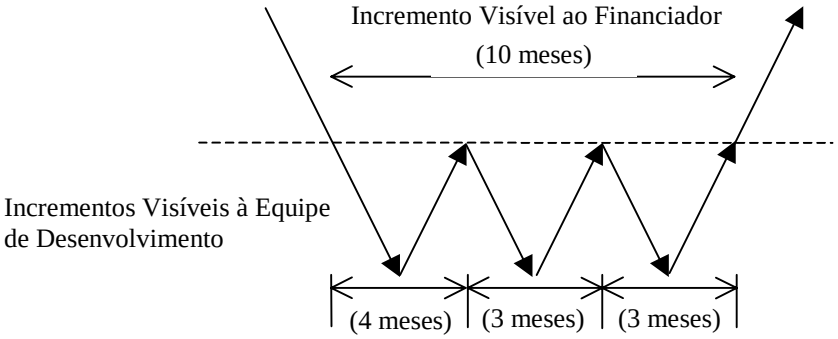


Figura 2. 9 – Separando um Incremento em Muitos.

Concorrência é outro conceito importante usado por Crystal. A questão chave no desenvolvimento concorrente é julgar a integração do processo com relação à estabilidade dos requisitos, o balanceamento das atividades com os pontos de gargalos, a capacidade de re-trabalho e eficácia da comunicação das equipes.

Num desenvolvimento serial, cada grupo de trabalho espera que o grupo de trabalho antecedente se torne estável (figura 2.10). O grupo responsável pela elaboração da interface de usuário aguarda que o grupo responsável pelo levantamento dos requisitos atinja um certo grau de estabilidade no processo antes de iniciar suas atividades. Em seguida, após concluir seus trabalhos este grupo passa seus produtos para o grupo de programação e este para o grupo de testes.

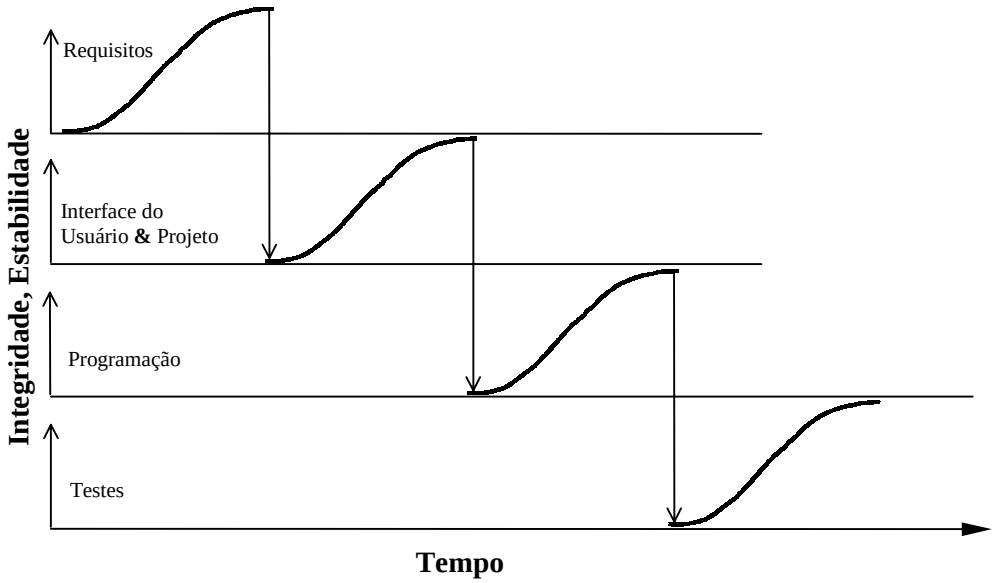


Figura 2. 10 – Um Desenvolvimento Serial.

Num desenvolvimento concorrente, cada grupo começa suas atividades assim que suas comunicações e capacidades de re-trabalho da equipe permitam (figura 2.11).

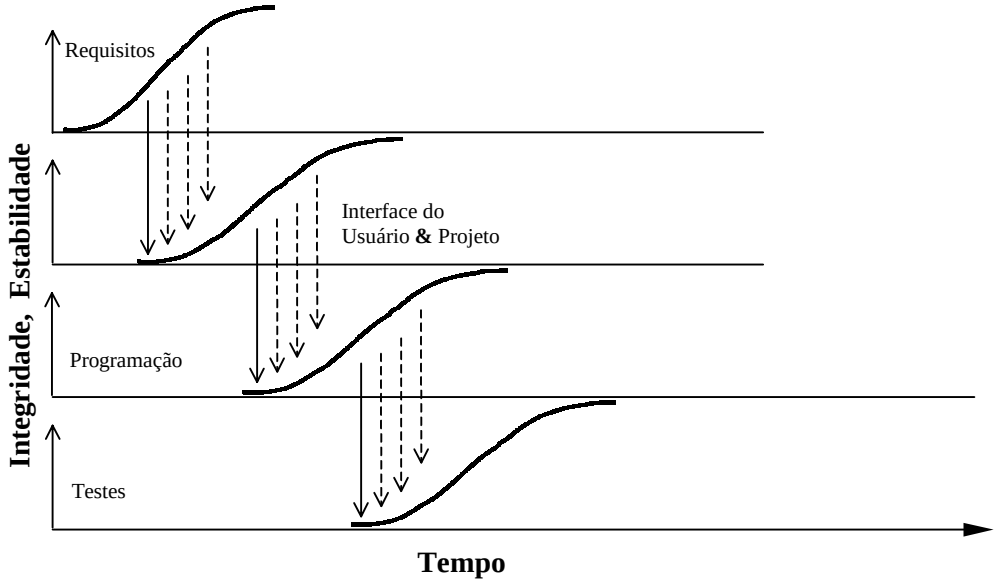


Figura 2. 11 – Um Desenvolvimento Concorrente.

Cada equipe de desenvolvimento tem suas próprias capacidades de trabalho, formas de comunicação para entender as artimanhas do projeto, e assim por diante, que são diferentes de outras equipes. No exemplo da figura 2.11, conforme o progresso, o grupo que obtém os requisitos passa, de forma contínua (como mostra as setas tracejadas), as informações atualizadas para o grupo responsável pela interface e projeto, o qual dará continuidade ao desenvolvimento com outras atividades. Por sua vez, o grupo de programação também recebe, de forma contínua, o projeto para ser codificado. Tão logo isto ocorra, o grupo de testes entra em ação. Num desenvolvimento concorrente os grupos têm que fazer re-trabalho, em virtude de, por exemplo, o grupo responsável pela elaboração de interface com o usuário ter iniciado suas atividades antes que as atividades do grupo responsável pelo levantamento dos requisitos se tornem estáveis, mas o resultado é obtido muito mais cedo devido a maior integração das equipes [11, 15]. Concorrência é um objetivo para muitas metodologias porque é uma das melhores técnicas para aumentar a velocidade de desenvolvimento. Concorrência é destinada a situações que exigem prioridades, como curto tempo para entrega do produto, cujos requisitos são demandados pelo mercado, com alta chance de alterações como uma conseqüência do sistema em desenvolvimento.

2.1.4. Extreme Programming

Extreme Programming (XP) faz uso de doze práticas e institui regras que definem os direitos do cliente e do desenvolvedor, no desenvolvimento de *software*. Estes direitos são baseados nos riscos do desenvolvimento de *software* e nos medos do cliente e do desenvolvedor.

Estas doze práticas, estabelecidas para neutralizar os riscos e medos inerentes a um projeto de *software*, são:

- *O Jogo do Planejamento.* Rapidamente determina o escopo do próximo *release* combinando as prioridades de negócios e estimativas técnicas. Quando a realidade ultrapassa o plano, o plano é atualizado.
- *Releases Pequenos*. Entrega rapidamente um sistema simples para ser colocado em produção e as novas versões são liberadas em ciclos pequenos. O sistema em funcionamento desperta outros requisitos de negócios, satisfaz ao financiador e mantém a equipe incentivada com o projeto.
- *A Metáfora* (Beck [3, 4], Jeffries [34], Wagner [65]). É uma integridade conceitual[5] baseada numa estória[6] simples, a qual é compartilhada pelos desenvolvedores, que representa como o sistema trabalhará e serve para moldar a arquitetura do mesmo. Segundo Jeffries [34], a metáfora passa por um processo de *brainstorming* que fornece uma base para a criação de nomes para todos os objetos no desenvolvimento do projeto e permite descrever as operações do projeto para qualquer pessoa.
- *Projetar de Forma Simples.* O sistema deve ser projetado tão simples quanto possível em qualquer momento. A complexidade extra é removida tão logo é percebida. Isto contribui para o entendimento das necessidades do projeto, permite que outros desenvolvedores entendam rapidamente o código produzido e permite que adaptações sejam efetuadas mais rapidamente.
- *Testes Automáticos* (Fowler [23], Jeffries [34], Wake [66]). São práticas eficientes, onde a codificação é guiada a partir dos testes. A eficiência se baseia em manter o foco do desenvolvedor exatamente nas partes do projeto necessárias para o desenvolvimento. Continuamente, os desenvolvedores escrevem testes de unidade, que devem ser executados sem falhas, antes do desenvolvimento prosseguir. É um modo de garantir que o código execute exatamente aquilo que o cliente espera. Estes testes são executados antes, durante e após a codificação. Já os clientes especificam

[5] A dificuldade é encontrar esta integridade conceitual que sirva como metáfora.
[6] Uma estória é alguma coisa que o cliente quer que o sistema faça [3].

os testes de aceitação que demonstram quando as características são concluídas. A elaboração dos testes antes da codificação contribui para o entendimento do processo, uma vez que os desenvolvedores sabem o resultado que estão procurando (sucesso nos testes) e os clientes ficam seguros da qualidade dos resultados. Quando o código é aprovado através dos testes de aceitação os desenvolvedores param de codificar a atividade.

- *Refactoring* (Martin [41], Wake [66], Orr [48], Fowler [23], Jeffries [34]). São técnicas aplicadas por programadores, no código de um projeto, as quais reestruturam o sistema sem alterar o seu comportamento de modo a remover duplicação de código, melhorar a sua comunicação, simplificar, ou adicionar flexibilidade de tal forma que facilite o entendimento dos outros programadores.
- *Programar em Pares* (Jeffries [34], Wake [66]). Toda a produção de código é escrita por dois programadores[7] em cada máquina. Enquanto um programador escreve o código, o outro apresenta alternativas de simplificação e padronização ao mesmo tempo em que faz uma revisão da codificação. Isto propicia que ambos aprendam, baseados na troca de experiência.
- *Propriedade Coletiva de Código* (Fowler [25]). Qualquer desenvolvedor pode alterar qualquer parte do código a qualquer momento. Isto permite que as soluções sejam implementadas quando percebidas, fazendo com que o desenvolvimento caminhe mais rápido.
- *Integração Contínua* (Fowler [25]). Integrar e construir o sistema várias vezes ao dia, sempre que uma tarefa for completada. Isto familiariza os desenvolvedores com as atividades de integração de sistema, tornando-a uma simples rotina diária.
- *40 horas semanais* – Como regra, não trabalhar mais que 40 horas na semana. Nunca trabalhar consecutivamente duas semanas onde cada semana ultrapasse às 40 horas semanais.
- *Cliente no Local* – Incluir um representante do cliente (com poder de selecionar as estórias mais adequadas ao negócio) como um membro da equipe de desenvolvimento, disponível durante todo o tempo para responder perguntas. Isto permite que o desenvolvedor obtenha respostas imediatas, contribuindo para uma rápida solução do problema.
- *Padrão de Codificação* – Os programadores escrevem todo o código de acordo com regras que dão ênfase à comunicação através da codificação. Não importa qual

[7] A dupla de programadores é periodicamente modificada, como forma de disseminar conhecimento sobre o domínio do projeto.

padrão será utilizado, desde que seja adotado por todos. Como o código é coletivo e a programação é feita em pares é fundamental que todos desenvolvam segundo um padrão estabelecido.

Excetuando-se os testes, nenhuma destas práticas se mantém eficaz de forma isolada. Por exemplo, manter o cliente no local apenas para entender os requisitos do sistema, ou programar em pares só para facilitar a programação e inspeção do código, podem não representar uma grande contribuição. Contudo, o conjunto delas produz um equilíbrio no desenvolvimento do projeto, pois as práticas se apóiam mutuamente (figura 2.12).

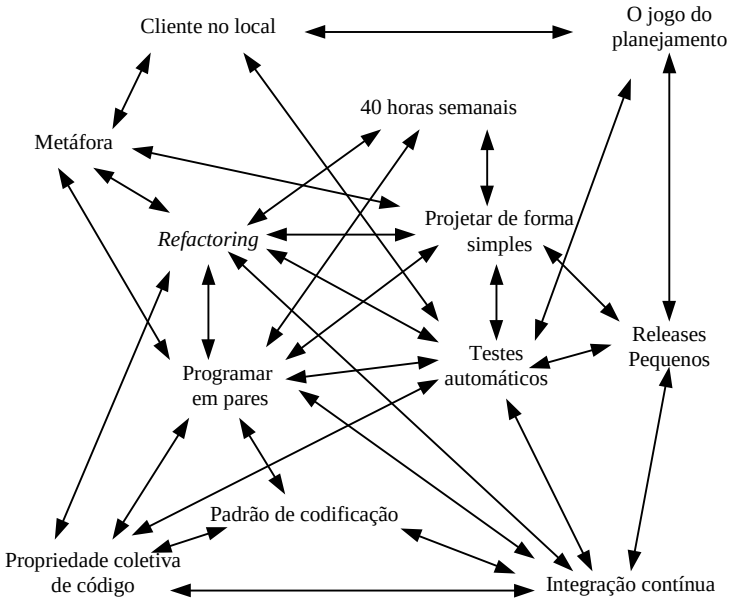


Figura 2. 12 – As Práticas se Suportam Mutuamente.

O cliente no local suporta mutuamente o jogo do planejamento, a metáfora e os testes automáticos, pois é o cliente quem indica o rumo a ser seguido pelo projeto (quando escolhe as estórias que serão desenvolvidas), especifica os testes de aceitação e tira as dúvidas dos desenvolvedores. Os testes por sua vez apóiam mutuamente *refactoring*, a programação em pares, a projetar de forma simples, a integração contínua e os "releases" pequenos. O *refactoring* se sustenta na metáfora, no projetar de forma simples, nos testes, na integração contínua, no padrão de codificação, na propriedade coletiva de código, etc.

Este apoio mútuo entre as práticas apresenta uma grande flexibilidade no desenvolvimento do projeto, pois contribui para a disseminação do conhecimento do domínio através da troca dos pares de programadores, permitindo que os desenvolvedores assumam qualquer parte do código. Segundo Beck [3], para se

perceber o bom resultado, de forma efetiva, é necessário que todas as práticas sejam executadas. Isto faz com que os processos de XP sejam rígidos, ou seja, se a programação não codificar primeiro os testes antes do código, ou se não programar em pares, isto pode ser programação baseada em XP, mas não é XP, pois não usufruirá dos benefícios da ajuda mútua entre todas as práticas descritas acima.

Com relação a risco Beck [3] diz que é um problema básico do desenvolvimento de *software* e destaca como riscos os seguintes:

- *Adiamento da data prevista.* Uma vez que o projeto não será entregue a tempo.
- *Projeto cancelado*. Após vários adiamentos sem ao menos entrar em produção.
- *Taxa de defeito.* É tão alta que o projeto não é usado.
- *Negócio mal entendido.* Ocorre quando o *software* não resolve o problema do negócio.
- *Mudança no negócio*. O *software* é entregue, mas os problemas do negócio que seriam solucionados foram substituídos por outros problemas mais importantes.
- *Falsa característica*. O *software* apresenta muitas características interessantes, do ponto de vista de programação, mas que não produzem rentabilidade para o cliente.
- *Insatisfação da equipe*. Resulta quando todos os bons desenvolvedores começam a se desinteressar pelo programa e deixam o projeto.

Beck também acredita que as pessoas envolvidas num projeto têm muitos medos e classifica os medos do cliente e os medos do desenvolvedor.

O cliente tem medo de:

- Não receber o que foi solicitado;
- Solicitar uma coisa errada;
- Pagar muito por tão pouco;
- Não ver um plano, de desenvolvimento de projeto, confiável;
- Ficar preso às decisões iniciais do projeto e não poder reagir às mudanças nos negócios;
- Não saber o andamento do desenvolvimento do *software*;
- Não saber da verdade (sobre as dificuldades de desenvolvimento do projeto).

O desenvolvedor tem medo de:

- Ser mandado a fazer mais do que ele sabe;
- Ficar desatualizado tecnicamente;
- Ser cobrado por responsabilidades sem ter recebido a autoridade necessária;
- Receber solicitações superficiais sobre o que precisa ser feito;

- Ter de solucionar problemas difíceis sem receber ajuda;
- Ter pouco tempo para conseguir os objetivos do desenvolvimento.

Para arcar com estes riscos e medos, XP institui certas regras que definem direitos do cliente e do desenvolvedor (Beck [4], Jeffries [34]).

O cliente tem o direito de:

- Ter um plano geral, para saber o que pode ser conseguido e a que custo;
- Receber o melhor resultado possível a cada semana;
- Ver o progresso no sistema, provando que o resultado do trabalho passa sucessivamente pelos testes especificados pelo cliente;
- Mudar seu modo de pensar, substituir funcionalidades, e mudar as prioridades sem ter que pagar custos exorbitantes;
- Ser informado das mudanças de planos a tempo de escolher como reduzir o escopo para garantir a data originalmente prevista de entrega do produto;
- Cancelar o projeto, quando quiser, e ficar com um sistema funcionando, refletindo seu investimento até o momento.

O desenvolvedor tem o direito de:

- Saber o que é necessário, com declarações explicitas de prioridades;
- Produzir trabalhos que satisfaça tanto ao cliente quanto ao seu desenvolvimento profissional;
- Pedir e receber ajuda de companheiros, gerentes, e clientes;
- Fazer e atualizar estimativas;
- Aceitar suas responsabilidades sem necessitar que elas lhe sejam impostas.

O XP não é um processo tolerante, todos estes valores, princípios e práticas devem ser rigorosamente seguidos para se obter um desenvolvimento efetivo, no qual o cliente é um participante do projeto (Jeffries [34], Wake [66]). A figura 2.13 indica que:

- O cliente define as estórias, ou seja, o que tem valor para o negócio e, portanto, deve ser desenvolvido;
- O programador estima os custos para desenvolver os valores para o negócio;
- O cliente escolhe as estórias que serão desenvolvidas[8];
- O programador constrói, e aprende a partir da última estimativa;
- O cliente, novamente, define os valores para o negócio e aprende com a repetição do processo e com os resultados obtidos em cada ciclo concluído.

[8] Para decidir o que fazer, e quando, é preciso saber o custo daquilo que foi solicitado.

Este processo é repetido através de estágios de aprendizagem tanto para o cliente (definindo e escolhendo valores) quanto para o programador (estimando e construindo valores).

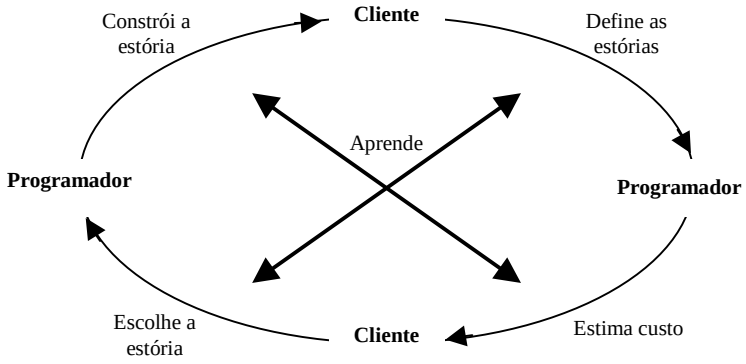


Figura 2. 13 – O Processo da Aprendizagem no Desenvolvimento.

2.2. Comparações entre as Metodologias Estudadas

Após a visão geral destas quatro metodologias, é possível observar que as mesmas apresentam muitas semelhanças e também algumas diferenças. A comparação será baseada em algumas fases de ciclo de vida do processo de desenvolvimento de *software* tais como planejamento, engenharia de requisitos, análise e projeto, implementação, testes e implantação.

Planejamento[9]

O RUP sugere o uso do Microsoft Project como ferramenta de planejamento, guiado por um template específico e procura estabelecer a estrutura do projeto, através da estimativa de esforços para cada fase de desenvolvimento, conforme exemplo apresentado por West [68] (figura 2.14). Para a maioria dos projetos o tempo de desenvolvimento em cada fase pode ser distribuído em 10% para Iniciação, 30% para Elaboração, 50% para Construção e 10% para Transição.

Krutchen [38] destaca também os marcos de referência que são elaborados no planejamento. No nível de planejamento das iterações (micro planejamento), procura-se identificar os *builds* e os *releases*.

[9] O Planejamento de um projeto se divide em macro planejamento e micro planejamento, ou seja o planejamento das iterações (ou fases).

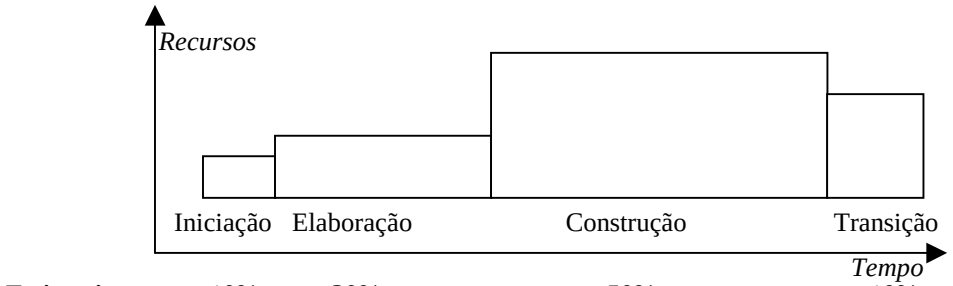


Figura 2. 14 – Alocação de Tempo Típica para as Quatro Fases de um Projeto.

Segundo Henderson-Sellers [28 pp. 172] (figura 2.15) há duas abordagens para o planejamento em OPEN: uma baseada em entrega de produtos ou pacotes; a outra, baseada nas atividades, tarefas, e técnicas. Em qualquer abordagem é destacada a natureza iterativa e incremental do plano do projeto.

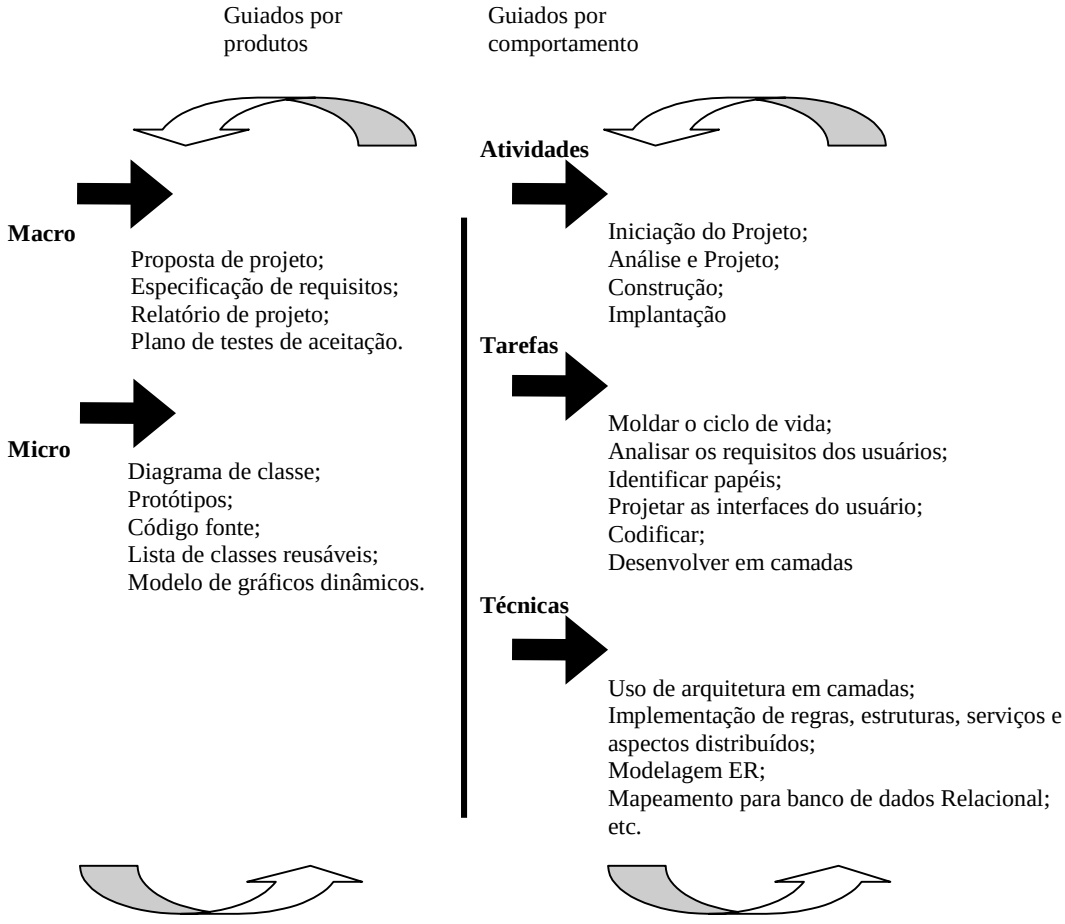


Figura 2. 15 – O Planejamento de um Projeto em OPEN.

O Crystal faz uso da técnica de precisão (PARTS), descrita na sub seção 2.1.3, usando casos de uso e auxiliado por recursos simples tais como a lista dos objetivos dos atores (tabela 5.1) e procura manter uma aproximação com o cliente até para compensar a falta de ferramenta adequada para sua metodologia.

O XP se baseia na metáfora e procura através das estórias estabelecer uma arquitetura para o sistema. O planejamento de XP é elaborado em conjunto pelos desenvolvedores e pelo cliente. Onde o cliente define as estórias que serão elaboradas em cada "release", baseando-se nas necessidades do negócio. Os desenvolvedores estimam quanto tempo levarão para desenvolver cada estória e alertam o cliente para as dificuldades técnicas relacionadas a cada estória.

Engenharia de Requisitos

O RUP se fundamenta nos casos de uso para a obtenção dos requisitos e utiliza o modelo de casos de uso para dirigir os modelos futuros do projeto. Os casos de uso servem também para: se formar um acordo entre o cliente e os usuários sobre o que o sistema fornecerá; dar aos desenvolvedores do sistema um entendimento dos requisitos do sistema; definir a funcionalidade do sistema; fornecer uma base para o planejamento das iterações; fornecer estimativas de custo e tempo para o desenvolvimento do sistema; e definir a interface do usuário do sistema.

O OPEN se fundamenta no projeto dirigido por responsabilidades [69] no lugar de caso de uso e utiliza uma série de técnicas [27] tais como modelagem dos relacionamentos, inspeções e projeto interno das classes, que são usadas nas tarefas para completar as atividades .

O Crystal procura identificar os requisitos da maneira mais conveniente para o projeto, ou seja, se fundamenta tanto nos casos de uso, como pode usar o projeto dirigido por responsabilidades.

O XP substitui os casos de uso pelas estórias, faz uso dos testes antes da codificação e utiliza o cliente no local tanto para esclarecer os requisitos bem como para elaborar os testes de aceitação.

Análise e Projeto

O RUP baseado em seu *workflow* procura traduzir os requisitos, guiados pelos casos de uso, numa especificação que descreva como implementar o sistema, através de um conjunto de classes e subsistemas os quais são restringidos por exemplo pelos requisitos não funcionais e pelo ambiente de implementação.

O OPEN usa como análise e projeto as diversas técnicas de modelagem e construção objetivando concluir tarefas tais como: a análise de requisitos; a lista inicial das classes de objetos; a identificação das classes persistentes; a identificação de papéis; o refinamento da lista de classes e o design da interface de usuários.

O Crystal usa UML e a técnica de cartões "Component-Responsibility-Collaborators" (CRC), a qual descreve princípios tais como alocação e avaliação de responsabilidades atribuídas a componentes de sistemas, combinando cenários com responsabilidades e interações. Esta técnica é associada à técnica de "modelagem baseada em responsabilidade" (RBM) [52] semelhante à técnica de "projeto dirigido por responsabilidade" (RDD) de Wirfs-Brock [69].

O XP usa de qualquer recurso na modelagem de um projeto, desde o rascunho num pedaço de papel até diagramas UML. A abordagem inicial se concentra nas partes essenciais do projeto a qual é codificada da forma mais simples. Em seguida o código sofre um processo de *refactoring,* modelando o projeto, antes de ser incrementado. Ao longo das iterações o código pode também sofrer novo refactoring a fim de manter o código (e o projeto) sempre pronto para um novo incremento e fácil de ser compreendido.

Implementação

O RUP segue o seu workflow específico de implementação o qual consiste na construção das partes do sistema, cujas unidades e componentes individuais são testados e combinados de forma incremental.

O OPEN se baseia na arquitetura, análise, projeto e usa as técnicas de modelagem para implementar o código, geralmente produzindo e testando componentes de forma incremental.

O Crystal se baseia nos casos de uso, nas técnicas RBM [52], e nos diagramas UML para a construção das partes do sistema (de forma iterativa e concorrente), mantendo contato, sempre que possível, com o cliente. O XP usa análise, projeto e implementação de forma tão iterativa, devido ao refactoring, que quase não se pode separar uma etapa da outra.

Testes

O RUP usa seu *workflow* de testes para verificar a interação entre objetos e componentes; a integração de todos os componentes do *software*; que todos os requisitos foram corretamente implementados; e identificar os defeitos e se certificar que todos foram tratados antes que o *software* seja implantado.

O OPEN e o Crystal aplicam processos semelhantes ao RUP. Os testes de unidade, de integração, bem como os testes de aceitação, competem exclusivamente aos desenvolvedores em contato com o cliente.

No caso do XP os testes de unidade e os de aceitação são de responsabilidade tanto dos desenvolvedores quanto do cliente. Os desenvolvedores elaboram os testes de unidade, auxiliados pelo cliente. Durante a implementação do código, efetuam também os testes de integração de forma contínua e param de codificar quando os testes de aceitação (elaborados pelo cliente juntamente com os desenvolvedores) são aprovados na execução do *build*.

Implantação

O RUP segue o seu workflow de implantação que procura entregar o produto ao cliente. Este fluxo de atividades envolve: a produção e montagem de um release do *software*; o empacotamento do *software*; a distribuição do *software*; a instalação do *software*; o treinamento dos usuários; o fornecimento de ajuda e assistência aos usuários; o planejamento e condução de testes beta; a migração dos *softwares* e dados existentes; e a aceitação formal pelo cliente.

O OPEN utiliza um plano de implantação, seguido de revisões de implementação. O Crystal não integra de forma contínua, mas estabelece builds periódicos para manter o controle do projeto focado nos interesses dos stakeholders. O XP utiliza a integração contínua. Se o desenvolvimento for no ambiente físico do cliente, a implantação ocorre de forma diária, uma vez que é constante a integração do sistema.

2.3. Considerações Finais

Este capítulo apresentou algumas metodologias destacando as práticas, princípios, similaridades e diferenças. Todas elas apresentam um processo iterativo de desenvolvimento.

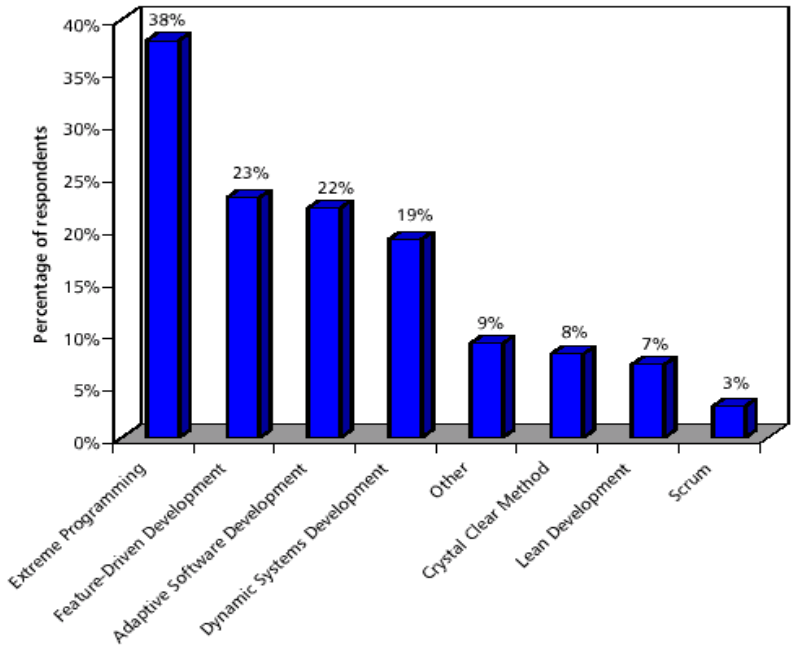
Segundo Charette [7], foi realizada uma pesquisa pelo consórcio Cutter, restrita a 200 gerentes de IS/IT[10], a qual mostrou que o Rational Unified Process é usado por 51% dos entrevistados que usam as metodologias, classificadas como pesadas. No entanto, por volta de 2003, aproximadamente 50% das pessoas que responderam a pesquisa esperam que a maioria dos seus projetos esteja usando abordagens ágeis

A tabela 2.3 apresenta um resumo destas considerações sobre as metodologias.

Etapa de Desenvolvimento	RUP	OPEN	Crystal	XP
Planejamento	Utiliza como recurso de planejamento o Microsoft Project.	Associa para cada Atividade uma relação de Tarefa x Técnica	Usa os conceitos de PARTS para estabelecer a arquitetura e planejamento.	Usa metáfora e estórias para a estabelecer a arquitetura do sistema.
Engenharia de Requisitos	Estas etapas são bem definidas e seguem fluxos de desenvolvimento específicos guiados pelos Casos de Uso.	Estas etapas são bem definidas e seguem o desenvolvimento dirigido por contrato de responsabilidade e implementado em camadas, de forma incremental.	Ocorre de forma iterativa, sempre validando com o cliente. Enfatiza a análise e o projeto, de forma iterativa, utilizando o desenvolvimento concorrente.	A interação utilizada por XP, entre cliente, desenvolvedor, e gerência, tornam difícil separa as etapas de requisitos, análise, projeto e implementação.
Análise				
Projeto				
Implementação				
Testes	Responsabilidade do desenvolvedor	Responsabilidade do desenvolvedor	Responsabilidade do desenvolvedor	responsabilidade do desenvolvedor e do cliente.
Implantação	Utiliza um fluxo específico para implantação.	Utiliza um plano seguido por revisões de implementação.	Estabelece builds periódicos.	Se o desenvolvimento for no ambiente físico do cliente, a implantação ocorre de forma diária.

Tabela 2. 3 – Resumo da Comparação das Metodologias.

Segundo a pesquisa, (figura 2.16) "para aqueles que não usam metodologias ágeis desenvolvidas por eles próprios, *Extreme Programming* é a mais popular, seguida pela *Feature-Driven Development*, *Adaptive Software Development*, e depois por *Dynamic Systems Development*".



[10] IS/IT significa Sistemas de Informação / Tecnologias de Informação.

Figura 2. 16 – As Metodologias Ágeis mais Usadas.

É desejável compreender que dependendo do contexto, a metodologia deve ser adaptada. Por exemplo, uma organização pode querer usar XP, mas devido à impossibilidade do cliente estar presente no local, tal decisão pode se tornar inviável. Neste caso, pode-se criar uma metodologia própria baseada em XP, ou então decidir por RUP, OPEN ou Crystal que, dependendo do contexto da organização, talvez sejam mais adequadas.

Algumas vezes um projeto requer uma metodologia “pesada” e não uma metodologia “ágil”. O fundamental é compreender os pontos positivos de cada metodologia, no contexto da organização, para usá-los quando necessário, não esquecendo os pontos negativos e procurando extrair o máximo proveito tanto das metodologias pesadas quanto das ágeis.

O próximo capítulo apresenta a proposição de um processo para selecionar uma metodologia de desenvolvimento de *software* para um projeto específico.

3. Bases Conceituais para o Processo de Seleção de Metodologias

Este capítulo apresenta alguns trabalhos correlatos, que foram utilizados como fundamentos para o processo de seleção de metodologias.

3.1. Trabalhos Correlatos

Esta dissertação está baseada em vários trabalhos correlatos. Os que mais se relacionam ou que mais influíram no desenvolvimento deste trabalho são descritos a seguir.

3.1.1. Guia Para Adoção de Ferramentas CASE

Em 1992, Oakes [46] adotou uma estratégia para seleção de ferramentas CASE na qual usou um processo cíclico que avalia a organização, avalia a tecnologia disponível e sua adequabilidade à organização, submete a ferramenta selecionada a um projeto piloto, efetua sua transição para um caso geral e finalmente institucionaliza a mesma na organização.

Pode-se dizer que avaliar a organização e a tecnologia são atividades, desempenhadas pelo corpo gerencial, que implicam em decisões e ações que representam custos e podem representar benefícios para a empresa; enquanto que submeter a ferramenta selecionada ao projeto piloto, efetuar a transição e sua institucionalização são atividades técnicas, que também devem ser executadas como um complemento das ações adotadas pelo corpo gerencial.

Este guia ajudará na avaliação da organização e da tecnologia no trabalho proposto.

3.1.2. Guia para Avaliação e Seleção de Ferramentas CASE

A norma ISO/IEC-14102 [70] estabelece quatro fases (figura 3.1): Iniciação, na qual são definidos os objetivos gerais, os requisitos da avaliação e seleção desejados (tendo como produtos o planejamento do projeto, os objetivos de alto nível e os critérios de seleção); Estruturação, na qual é estabelecido um conjunto de requisitos fundamentais, com base nas características das ferramentas e sobre os quais são tomadas as decisões da seleção, sendo por isso o processo mais importante (tendo como produtos o documento de requisitos e a lista de ferramentas candidatas); Avaliação, consiste num processo cujos resultados (plano de avaliação e relatórios produzidos) servirão como entradas do que serão usados no processo de seleção; e Seleção, na qual são identificadas as ferramentas mais apropriadas entre as candidatas (tendo como produto as recomendações de seleção).

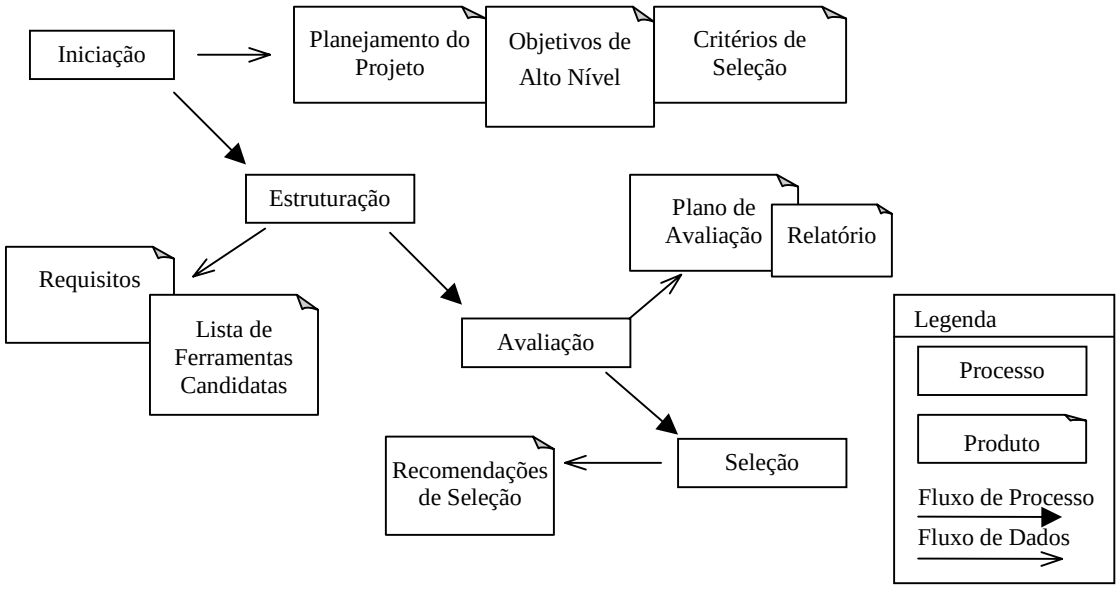


Figura 3. 1 – Avaliação e Seleção de Ferramentas CASE.

Estas fases serão utilizadas para estabelecer a estrutura do processo de seleção proposto.

3.1.3. Gerenciamento de Projetos com Foco em Inovação e Velocidade

Highsmith [30] utiliza os modelos de ciclos de vida de tecnologia e os tipos de culturas organizacionais propostos por Moore [44] para identificar os tipos de metodologia. Uma tecnologia não surge no mercado de forma aleatória. A sua difusão na sociedade [49] (figura 3.2) segue um processo através de dois modelos de mercados: o inovador e o principal, nos quais, as pessoas, gradualmente, exigem a garantia de benefícios relacionados às mudanças.

No mercado inovador [44] se encontram as pessoas (entusiastas e visionários) que são favoráveis às inovações tecnológicas e mais receptivas às mudanças. No mercado principal estão as pessoas menos favoráveis às inovações (pragmáticos, conservadores e desconfiados). O ciclo de vida da tecnologia tem início entre os entusiastas e segue através dos visionários, pragmáticos, conservadores e desconfiados, ao longo do tempo.

Os **entusiastas** são pessoas que valorizam a inovação e desejam aprender e trabalhar com as mais recentes tecnologias. Eles não estão preocupados se a tecnologia está sendo usada por outras pessoas ou se já tem suporte tecnológico. Os **visionários**,

baseando-se no sucesso inicial dos entusiastas, procuram soluções de negócios, particulares e imediatas.

Os **pragmáticos** confiam nos visionários e se preocupam com o suporte tecnológico, pois não pretendem fazer qualquer coisa de forma isolada. Os **conservadores** observam os pragmáticos e exigem absoluta confiança de que suas ações estarão corretas. Estão preocupados com detalhes e, especialmente interessados nos custos dos negócios, exigem baixos preços e serviços. Os **desconfiados** são os últimos a utilizar uma tecnologia, quase sempre quando a mesma está prestes a ser substituída.

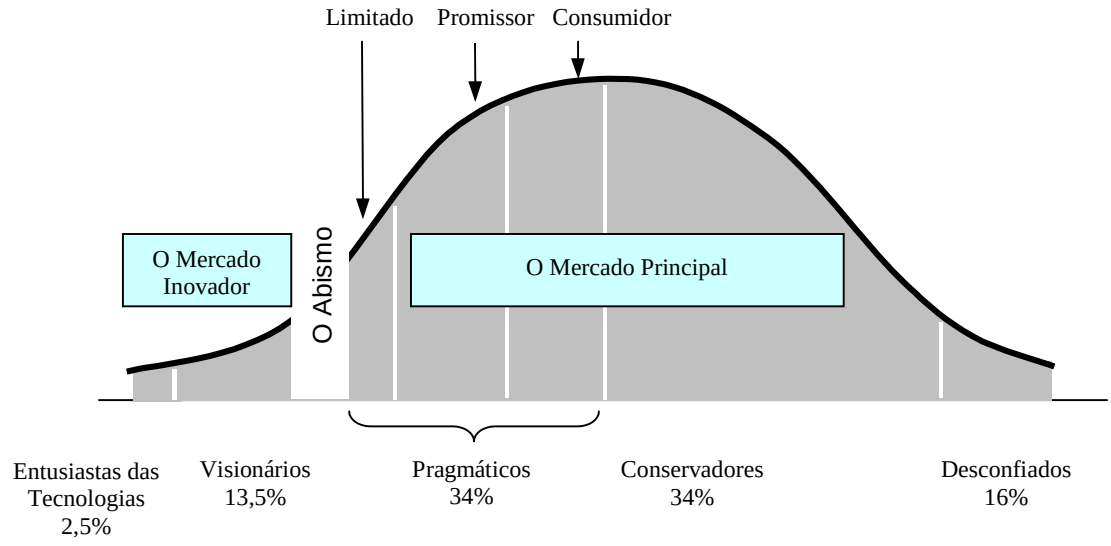


Figura 3. 2 – O Abismo do Mercado Inovador ao Tradicional.

A diferença entre estes dois mercados é tão grande que Moore [44] considera que existe um abismo entre ambos e, estimando que a chave para o sucesso no longo prazo está entre os pragmáticos (o que pode representar 84% uma vez que os conservadores, seguidos pelos desconfiados, confiarão nos pragmáticos), Moore subdivide-o em **limitado (***Bowling Alley)*, **promissor** (*Tornado)* e **consumidor** (*Main Street*).

- Limitado – através da analogia entre os pinos em um jogo de boliche e uma instância da tecnologia num nicho de mercado, Moore diz que o sucesso (derrubar um pino) obtido por uma companhia permite que outro cliente seja acessado no mesmo nicho, ou que outro nicho seja descoberto para a mesma tecnologia.
- Promissor – nesta fase, a tecnologia “alça vôo” e a chave é crescer o quanto puder (similar a um tornado), a ponto de Moore recomendar que as companhias

ignorem os clientes (não percam tempo com detalhes) e procurem obter o domínio do mercado. Moore sugere que no final da fase promissor haverá uma companhia dominante (gorila), representada pelo seu crescimento relativo, algumas companhias que cresceram confortavelmente (chipanzés) e uma porção delas que sobreviveram (macacos).

- Consumidor – nesta fase, o mercado se acomoda e os clientes começam a exigir redução de preços e descontos (a tecnologia torna-se uma *commodity* que pode ser comprada em qualquer rua principal – *main street*). A palavra chave passa a ser preço e serviço. A ameaça da competição já passou mas a companhia deve começar a procurar inovações a fim de recuperar o crescimento e retorno dos lucros.

Ainda de acordo com Moore [44], as quatro culturas organizacionais básicas são: competência, colaboração, controle, e cultivo,

A **cultura de competência** exalta o conhecimento e especialidades acima de títulos e cargos empresariais, exigindo a máxima capacitação técnica para os cargos gerenciais. Intensidade, velocidade nos negócios e trabalhos além do horário são características desta cultura. O perigo da cultura de competência é que levada ao extremo, pode resultar em metodologias onde a gerência de projetos tende a ignorar o mercado e manter o foco em medições e práticas internas de sucesso.

As **culturas de colaboração** são dominadas por grupos multifuncionais, onde os lideres se baseiam em papéis funcionais. Elas prestigiam as lideranças e as equipes de trabalho e estabelecem propósitos, limites, encorajam interação, e sabem quando tomar uma ação decisiva. Contudo, isto tudo pode resultar no adiamento de decisões e criação de grupos antagonistas.

As **culturas de controle** são baseadas no poder de decisão e são orientadas para a segurança e para a manutenção da estrutura hierárquica. As culturas de controle apresentam limitações, diante dos desafios da nova economia mundial, uma vez que as mudanças à que estão aptas, são àquelas previstas pela própria estrutura.

A **cultura de cultivo** motiva a autocapacitação e coloca as pessoas em primeiro plano, não como equipe, mas individualmente. Facilita a produção de novas idéias e produtos, mas os indivíduos relutam em fazer um trabalho quando perdem o interesse no mesmo. Dificulta a escalabilidade dos negócios uma vez que os participantes recusam qualquer forma de estrutura, processo, ou documentação.

A tabela 3.1 está baseada na análise de Highsmith [30] em relação à compatibilidade entre os tipos de metodologias (P = Pesada, A = Ágil e N = Nenhuma, ou seja uma abordagem ad hoc. para o desenvolvimento do projeto), os modelos de mercado e os tipos de cultura organizacional. Por exemplo, o mercado inovador é mais adequado para ad. hoc ou métodos muito leves e, portanto, é ideal para a cultura de cultivo.

	Competência	Colaboração	Controle	Cultivo
Embrionário	**N-A**	**N-A**		**N**
Limitado	**A**	**A**		**N-A**
Promissor	**A**	**A**		
Consumidor			**P**	

Legenda: ☐ Ideal ☐ Aceitável ☐ Desfavorável

Tabela 3. 1 – Tipos de Metodologias Versus os Modelos do Mercado e os Tipos de Cultura.

Para se obter sucesso no mercado limitado, é necessário um alto grau de conhecimento das necessidades do cliente e um plano para customizar os produtos. Uma cultura de colaboração, pelas suas características de trabalho coletivo, é a que melhor se adapta nesta fase de mercado, seguida pela cultura de competência. Para este caso, uma cultura de controle, resulta em confronto, pois seriam investidos grandes recursos.

Para o mercado promissor, o qual exige um alto desempenho devido à crescente demanda dos clientes é adequado para as culturas de competência, seguindo-se a de colaboração.

Para o mercado consumidor (e também o mercado conservador) em virtude de sua estabilidade, relacionada tanto à tecnologia quanto ao negócio, é adequado às culturas de controle que procuram impor ordem e previsibilidade, e conseqüentemente favorecem as metodologias pesadas.

Os desconfiados não são considerados, em virtude de não exercerem influência sobre os outros grupos. As tabelas 3.2 e 3.3, relacionam os modelos de mercado com o tipo de metodologia e o tipo de cultura organizacional com o tipo de metodologia respectivamente.

Modelo de Mercado	Tipo da Metodologia	Observações
Inovador	N	Ad hoc ou métodos muito leves são apropriados para pesquisa rápida e desenvolvimento. Uma metodologia ágil não se encaixa bem pois exige um trabalho colaborativo.
Limitado	A	Altos níveis de customização e interação intensa com o cliente são melhorados através da adoção de métodos mais leves orientados à colaboração.
Promissor	A	As reações com alta velocidade, que são requeridas neste modelo de mercado, vão de encontro a uma metodologia pesada, centrada em documentação, embora algum controle seja requerido para permitir atender a demanda do mercado.
Consumidor / Conservador	P	É recomendada uma abordagem pesada, enfatizando a eficiência operacional. Isto é possível em virtude deste mercado apresentar estabilidade, relacionada tanto à tecnologia quanto ao negócio.

Tabela 3. 2 – A Combinação do Tipo de Metodologia com o Mercado.

Cultura Organizacional	Tipo da Metodologia	Observações
Competência	A	As culturas de competência mantêm o foco na capacitação em detrimento do processo, e conseqüentemente obterão os benefícios de métodos leves. Elas não toleram uma documentação rigorosa e pesada.
Colaboração	A	As culturas de colaboração são receptivas a métodos que aumentem a interação do grupo, mas elas são bem menos receptivas a metodologias centradas no processo.
Controle	P	As culturas de controle procuram impor ordem e previsibilidade, e conseqüentemente favorecem as metodologias pesadas.
Cultivo	N	A natureza empreendedora da cultura de cultivo só tolera um mínimo de metodologia (muito leve), quando muito.

Tabela 3. 3 – A Combinação do Tipo de Metodologia com a Cultura Organizacional.

Estes conceitos serão usados para avaliar o mercado e a cultura da organização para a seleção da metodologia.

3.1.4. Uma Seleção de Produtos de Software Utilizando uma Abordagem Baseada em Engenharia de Requisitos

Alves [1] propõe o método CRE (*COTS-based Requirements Engineering*) para assistir o processo de seleção de produtos reusáveis disponíveis no mercado, conhecidos como COTS (Commercial-Off-The-Shelf). O seu método é orientado a requisitos e tem como uma das principais estratégias enfocar a descrição dos requisitos não-funcionais durante o processo seletivo. De forma similar, nesta dissertação serão usados requisitos não-funcionais nas diversas fases.

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 a esses requisitos (figura 3.3). Esse processo continua até que um produto satisfaça um número suficiente de requisitos.

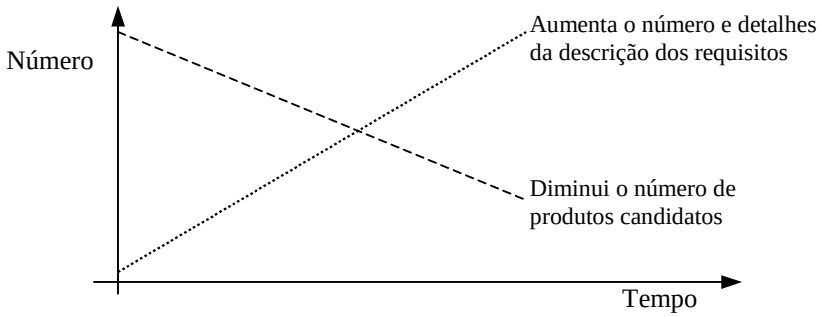


Figura 3. 3 – Processo Iterativo de Aquisição de Requisitos e Seleção de Produtos.

O método CRE é orientado a metas, isto é, as fases do processo são direcionadas por metas previamente definidas; possui quatro fases que interagem entre si (Identificação, Descrição, Avaliação e Aceitação); e sugere que o processo de engenharia de requisitos deve direcionar todo o processo de seleção. Sendo assim, as etapas do processo de requisitos podem ser realizadas e repetidas em cada fase do método CRE.

3.1.5. Uma Metodologia por Projeto

Cockburn [12, 13, 15] estabeleceu quatro princípios relacionados ao tamanho[1] de uma metodologia e definiu a criticidade dos sistemas em quatro faixas. Os quatro princípios são:

***Princípio Um**:* Para uma maior equipe de desenvolvimento, é necessária uma metodologia com um maior número de elementos de controle.

***Princípio Dois**:* Quanto maior a criticidade de um sistema, ou seja, quanto maior for o prejuízo provocado por defeitos não detectados, maior é a necessidade de tornar visível o processo.

[1] O tamanho de uma metodologia é o seu número de elementos de controle, incluindo produtos, padrões, atividades, marcos de referência, medidas de qualidade, entre outros.

***Princípio Três**:* Um pequeno aumento no tamanho da metodologia, ou na sua densidade[2], provoca relativamente um grande aumento no custo do projeto.

***Princípio Quatro**:* A forma mais efetiva de comunicação no sentido de facilitar a transmissão de idéias entre as equipes de desenvolvimento é através do contato frente a frente.

As quatro faixas de criticidade de um projeto são :

Perda de Conforto: significa que uma falha no sistema provocará desconforto nas pessoas, pois terão que efetuar um trabalho manualmente. Como exemplos desta classificação têm-se os de sistemas de suporte da rede corporativa de uma organização .

Perda de Quantia Monetária Discreta: significa que uma falha no sistema produz perda de dinheiro ou valores relativos. Exemplos típicos são sistemas de faturas de notas fiscais.

Perda de Quantia Monetária Essencial: significa que uma perda de dinheiro ou valores relacionados tem um efeito similar à falência empresarial. Um exemplo desta faixa é um sistema de controle bancário.

Perda de Vida: significa que pessoas muito provavelmente morrerão se houver uma falha no sistema. Exemplos desta faixa são sistemas para usinas nucleares, aeronaves, e controle de aviões.

Baseando-se nestes princípios, Cockburn estabeleceu um *framework* utilizado para selecionar instâncias de metodologias da família Crystal [12, 13, 15], as quais são denominadas por cores tais como claro, amarelo e verde (figura 3.4) resultando nas três instâncias *Crystal Clear*, *Crystal Yellow* e *Crystal Green*, respectivamente. O uso do *framework* consiste, segundo Cockburn, em determinar o número de pessoas num projeto, bem como identificar sua criticidade e prioridades. Por exemplo, para um projeto com três pessoas e com criticidade classificada como Conforto, a metodologia selecionada é uma instância de *Crystal Clear*. Para outro projeto, com dez pessoas e considerando uma criticidade envolvendo perda de Quantia Monetária Discreta, seria instanciada a metodologia *Crystal Yellow*.

[2] A densidade é o detalhe e a consistência requeridos nos elementos da metodologia. Uma maior densidade corresponde a um maior e mais rígido sistema de controle.

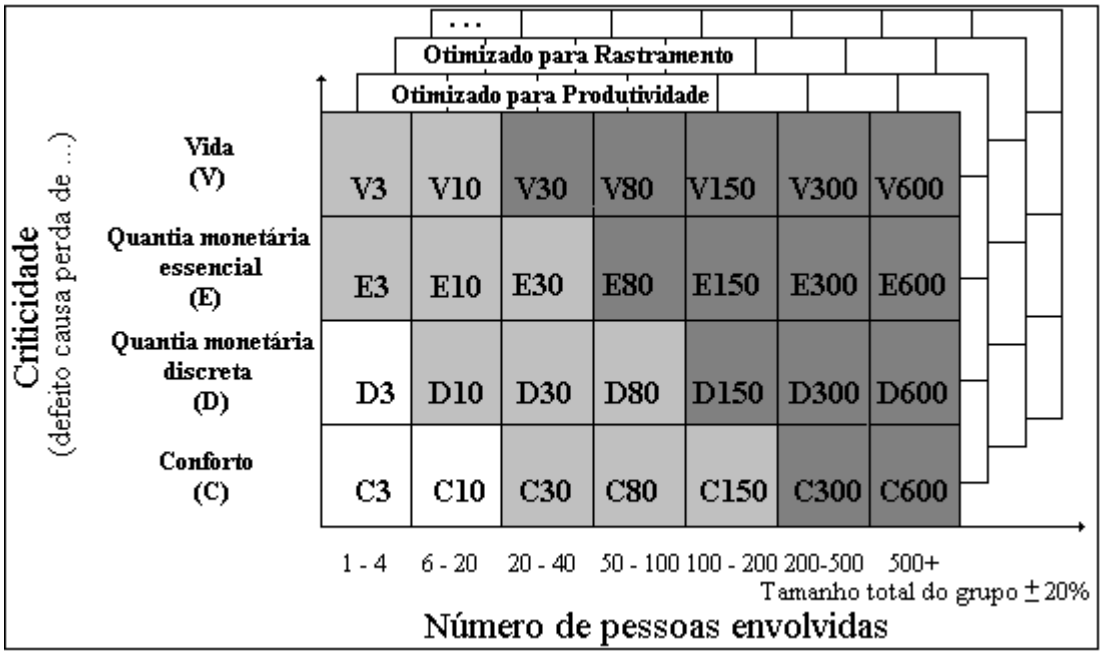


Figura 3. 4 – Um Processo de Seleção Considerando Pessoas x Criticidade x Prioridades do Projeto.

Os conceitos apresentados por Cockburn serão usados para identificar o projeto e a criticidade do sistema na seleção da metodologia.

3.1.6. O Modelo SA-CMM

O Modelo de Maturidade e Capacidade para Aquisição de *Software* (SA-CMM) [16] é um trabalho de cooperação entre o governo dos Estados Unidos, indústria, academia e o *Software Engineering Institute* (SEI) [58]. A experiência do SEI no desenvolvimento do Modelo de Maturidade e Capacidade para *Software* (SW-CMM) [8] foi aplicada no desenvolvimento do SA-CMM.

O SA-CMM fornece a estrutura necessária para facilitar a melhoria desejada nos processos de aquisição de *software* pelas organizações industriais e governamentais.

O SA-CMM descreve o papel do comprador, ou seja, de quem faz o processo de aquisição do *software*. Uma aquisição individual começa com o processo de definir a necessidade do sistema. O modelo inclui certas atividades pré-contratuais, tais como preparação do pacote de solicitação, desenvolvimento do conjunto inicial de requisitos e participação na seleção dos fornecedores. Uma aquisição individual termina quando o contrato para os produtos está concluído.

O SA-CMM identifica áreas chave de processos para quatro dos seus cinco níveis de maturidade. As áreas chave de processos estabelecem objetivos que devem ser

satisfeitos para cada nível de maturidade. Os níveis de maturidade e suas áreas chave de processos guiam a escalada para os níveis superiores de maturidade.

Uma parte dos objetivos ou atividades das áreas de processos dos níveis mais altos pode ser satisfeita ou executada num nível inferior. Contudo, uma área chave de processo não pode ser alcançada até que todos os seus objetivos sejam satisfeitos. Um nível de maturidade é alcançado quando se dominam todas as áreas chave de processos naquele nível. Uma vez que um nível de maturidade é alcançado, todos os objetivos nos níveis inferiores devem ser satisfeitos.

O SA-CMM é suficientemente genérico para ser usado por qualquer órgão governamental ou industrial, sem restrições ao tamanho da organização que efetua a aquisição dos produtos. Quando se aplica o SA-CMM a uma determinada empresa, algumas adaptações devem ser feitas de tal forma que se ajuste à estrutura empresarial. A equipe de projeto, apoiada por outros grupos interessados, é a entidade que tem a responsabilidade de conduzir a aquisição específica. O modelo deve ser interpretado no contexto das necessidades da organização; nem tudo que o modelo contempla deve ser automaticamente aplicado. Processos de aquisição efetivos e eficientes são críticos para o sucesso dos processos de melhorias, mas a qualidade dos resultados só pode ser determinada no contexto das necessidades do negócio da organização.

O uso do SA-CMM não está limitado a situações onde os produtos são adquiridos através de um contrato formal. A aquisição se aplica a todos os tipos de aplicações de *software* isolados e embarcados, incluindo aplicações comerciais de prateleira (COTS) e itens de *software* complementares adquiridos como parte de um sistema ou separadamente.

O SA-CMM é aplicado em todo o ciclo de vida do sistema. Durante o ciclo de vida de um sistema podem ocorrer muitas aquisições individuais, tanto na fase inicial do desenvolvimento quanto na fase de suporte. Quando itens são adquiridos para melhoria ou reengenharia de produtos, a organização responsável pelo suporte do sistema assume o papel de responsável pela aquisição. Desse modo, o modelo é aplicado, repetidamente, durante o ciclo de vida de um sistema.

O projeto de aquisição inicia com um requisito de alto nível. Quando a aquisição tem início, mais requisitos são identificados e refinados. Para os propósitos do SA-CMM, estes requisitos formam uma base (gerenciada e controlada) que incorpora novos requisitos. O gerenciamento e controle dos requisitos permanecem sob a responsabilidade da organização que efetua a aquisição, mesmo que a equipe responsável pelos fornecedores contribua para o processo de desenvolvimento.

Um projeto deve estabelecer a base para os requisitos dos produtos técnicos e não-técnicos, para os custos dos produtos que estão sendo adquiridos e a programação de entrega do projeto de aquisição.

Os Níveis de Maturidade do SA-CMM

A gerência da organização adquirente deve aumentar sua participação, liderança e disciplina no momento que ela procurar atingir um nível de maturidade maior. O SA-CMM define cinco níveis de maturidade. Cada nível de maturidade (exceto o nível 1) indica a capacidade do processo e as suas áreas chave do processo. Estes níveis fornecem um roteiro para a melhoria contínua do processo de aquisição de uma organização; não têm a finalidade de servir como mecanismos de classificação da posição da organização.
Os cinco níveis de maturidade do SA-CMM, são descritos a seguir:

1. *Inicial* – O processo de aquisição é caracterizado como ad hoc e, ocasionalmente, caótico. Poucos processos são definidos e o sucesso depende de esforço individual. Para que a organização amadureça, além do nível inicial, ela deve instalar controles básicos de gerenciamento.
2. *Repetível* – Os processos básicos de gerenciamento de aquisição são estabelecidos para planejar todos os aspectos de aquisição, gerenciar requisitos, acompanhar a equipe de projeto e o desempenho das equipes de fornecedores, gerenciar os custos de projetos e programar suas bases, avaliar os produtos e efetuar a sua transição para a organização de suporte. A equipe de projeto basicamente reage às circunstâncias da aquisição quando elas ocorrem. É estabelecida a disciplina necessária ao processo para permitir repetir os sucessos em projetos similares.
3. *Definido* – O processo de aquisição da organização está documentado e padronizado. Todos os projetos usam uma versão adaptada e aprovada do processo de aquisição padrão da organização para adquirir seus produtos. As atividades de projeto e gerência de contrato são pró-ativas, procurando se antecipar e negociar as circunstâncias da aquisição antes que elas apareçam. O gerenciamento de risco está integrado em todos os aspectos do projeto, e a organização fornece o treinamento requisitado pelo pessoal envolvido na aquisição. Para uma organização amadurecer além do nível definido, ela deve basear suas decisões em medições quantitativas de seus processos e produtos de tal forma que possa obter objetividade e efetuar decisões racionais.

4. *Quantitativo* – As medições detalhadas, dos processos e produtos de aquisição, são coletadas. Os processos e produtos são quantitativamente e qualitativamente compreendidos e controlados.

5. *Otimizado* – A melhoria contínua de processo é empregada através de *feedback* quantitativo dos processos, bem como pela implementação piloto de novas idéias e tecnologias. Ou seja, a organização reconhece que a melhoria contínua (e mudança contínua) é necessária para a sua sobrevivência.

A tabela 3.4 apresenta os cinco níveis descritos acima, com suas respectivas áreas chave de processo.

Nível	***Foco***	***Área Chave de Processo***
5 Otimizado	Melhoria de Processo Contínua	Gerência de Inovação na Aquisição
		Melhoria Contínua de Processo
4 Quantitativo	Gerenciamento Quantitativo	Gerência de Aquisição Quantitativa
		Gerência de Processo Quantitativo
3 Definido	Padronização de Processo	Gerência de Programa de Treinamento
		Gerência de Risco de Aquisição
		Gerência de Desempenho de Contrato
		Gerência de Desempenho de Projeto
		Requisitos do Usuário
		Definição e Manutenção de Processo
2 Repetível	Gerência Básica de Projeto	Transição para Suporte
		Avaliação
		Supervisão e Acompanhamento de Contrato
		Gerência de Projeto
		Desenvolvimento e Gerenciamento de Requisitos
		Solicitação
		Planejamento de Aquisição de *Software*
1 Inicial	Pessoas Competentes e Heróis	

Tabela 3. 4– Sinopse do Modelo SA–CMM.

O processo proposto, nesta dissertação, direciona a organização a atingir grande parte do nível 2 e algumas áreas chave de processo dos níveis superiores.

3.2. Considerações Finais

Neste capítulo foram apresentados a motivação deste trabalho, os problemas relacionados e alguns trabalhos correlatos.

Foi visto que o sucesso no desenvolvimento de um projeto de *software* depende de uma concentração de esforços, os quais devem ser apoiados pela metodologia selecionada. Os problemas para a seleção de uma metodologia vão desde a identificação do mercado a que se destina o produto até o conhecimento da cultura da organização como um todo, além da diversidade de metodologias disponíveis no mercado de *software*.

Dos trabalhos correlatos percebe-se a necessidade de avaliação da organização que deve ser estruturada também para o mercado em foco, considerando as reações às mudanças no contexto da organização, associado ao projeto de *software* considerado.

No próximo capitulo, o processo proposto é descrito com base no trabalho de Oakes para seleção de ferramentas CASE [46]; na cultura organizacional observada por Moore [44] e Cockburn [11, 15]; no relacionamento entre o mercado, cultura da organização e tecnologia, apresentado por Highsmith [30]; na capacidade da equipe de desenvolvimento, no tipo de projeto e na criticidade dos projetos definidos por Cockburn [12, 13]; nos interesses dos *stakeholders* segundo Thomsett [64]; no alinhamento das regras de negócios, estratégias e políticas com os objetivos de negócio da organização segundo Salviano [57]; e nos requisitos não funcionais [1].

4. Processo Proposto para a Seleção de Metodologias

Este capítulo apresenta um processo para selecionar uma metodologia a ser usada no desenvolvimento de um projeto de *software*, levando em conta os interesses dos *stakeholders* que são influenciados pelo mercado, tecnologia, estrutura organizacional, regras de negócio, estratégias e políticas da organização, capacitação das pessoas, criticidade do sistema, prioridades do projeto e cultura da organização.

4.1. A Proposta de uma Solução

Nesta seção é apresentado um processo de seleção cuja estrutura principal está baseada na norma ISO/IEC-14102 [70]. O processo proposto é composto por cinco fases e suas atividades, como apresentadas na tabela 4.1.

Fase	Atividades
Iniciação	– Estabelecer Objetivos Consiste em fixar claramente os objetivos do projeto e da organização em relação ao mercado. – Estabelecer Critérios de Seleção de Metodologias Especificar o nível de detalhe e a natureza das atividades, considerando o processo de desenvolvimento. – Planejar o Projeto de Seleção de Metodologias Designar responsabilidades, garantir orçamento e elaborar uma programação.
Estruturação	– Identificar os Interesses dos *Stakeholders* Avaliar a organização, o mercado, tecnologia, regras de negócios, políticas, capacitação, prioridades do projeto e cultura da organização. – Identificar as Metodologias Disponíveis Identificar as metodologias que satisfazem um conjunto de prioridades e requisitos críticos, observando-se os custos. – Definir os Requisitos da Metodologia Analisar o projeto e a cultura da organização os quais determinam os interesses que orientam os requisitos da metodologia.
Avaliação	– Avaliar as Metodologias Confrontar as metodologias disponíveis com os requisitos da metodologia procurada.
Seleção	– Selecionar uma Candidata Selecionar uma metodologia que melhor expresse os interesses dos *stakeholders* do projeto. – Validar a Metodologia Aplicar a metodologia no desenvolvimento de um projeto piloto.
Institucionalização	– Institucionalizar a Metodologia Implantar a metodologia, em toda a organização, com base no piloto.

Tabela 4. 1– As Fases do Processo Proposto.

As atividades presentes nestas fases são executadas na ordem indicada pelo diagrama de atividades, apresentado na figura 4.1.

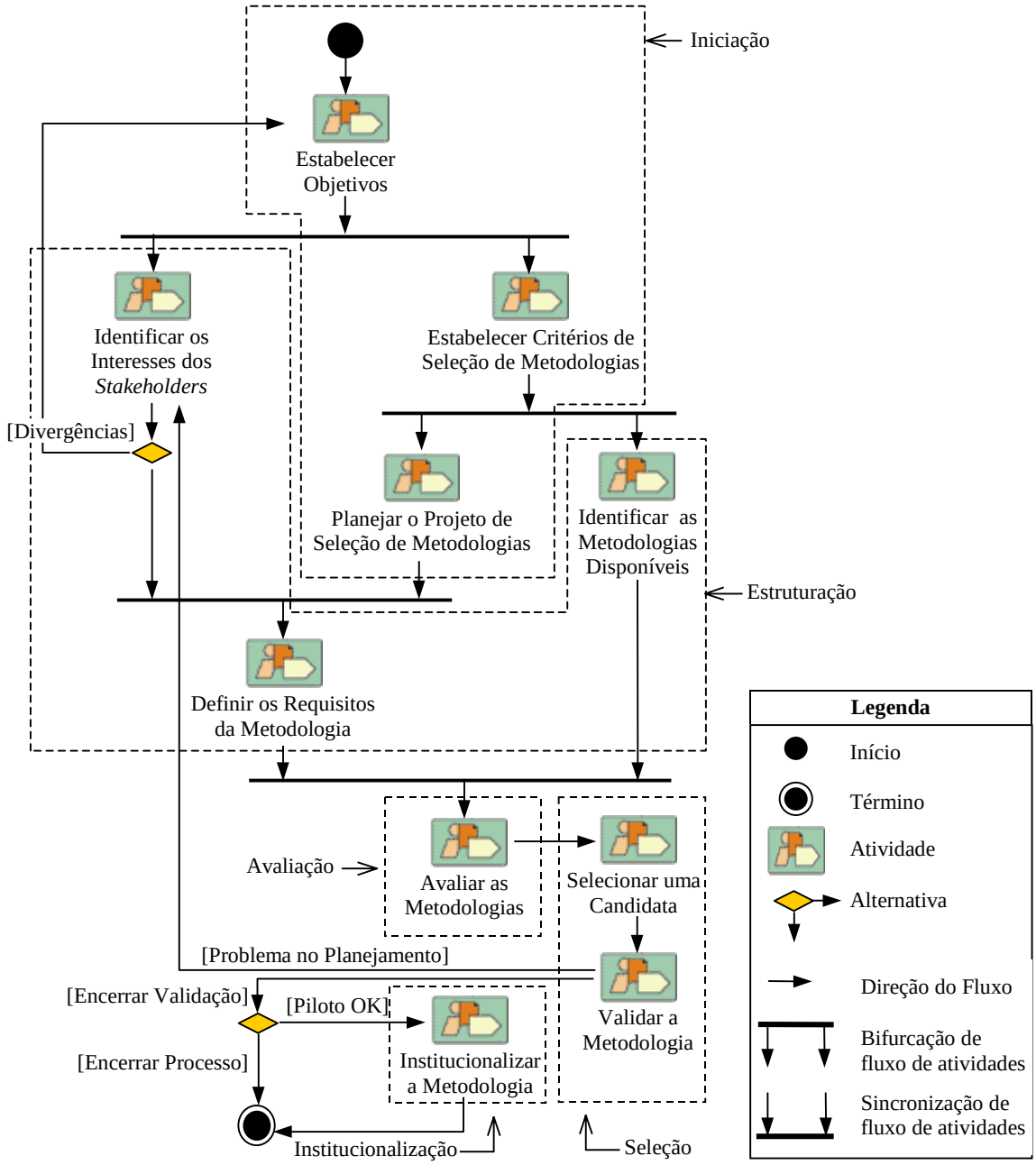


Figura 4. 1– A Proposta de uma Solução para a Seleção de Metodologias.

Para facilitar o entendimento do processo proposto, será apresentado um exemplo hipotético ao longo das próximas seções, o qual terá como cenário uma organização que desenvolve projetos de *software*, com uma pequena equipe, e decide atingir o mercado das organizações governamentais, onde os clientes têm como uma das exigências o acompanhamento dos projetos. Isto requer um maior número de pessoas na empresa

uma vez que os resultados das atividades desenvolvidas serão disponibilizados para acompanhamento pelos clientes e, portanto, haverá necessidade de confecção e garantia da qualidade dos produtos a serem publicados.

4.2. Fase de Iniciação

Nesta fase inicial do processo, os objetivos são estabelecidos, como ponto de partida, seguindo-se o estabelecimento dos critérios de seleção de metodologias e o planejamento do projeto de seleção de metodologias conforme mostra a figura 4.2.

Estas atividades, são descritas nas seções abaixo.

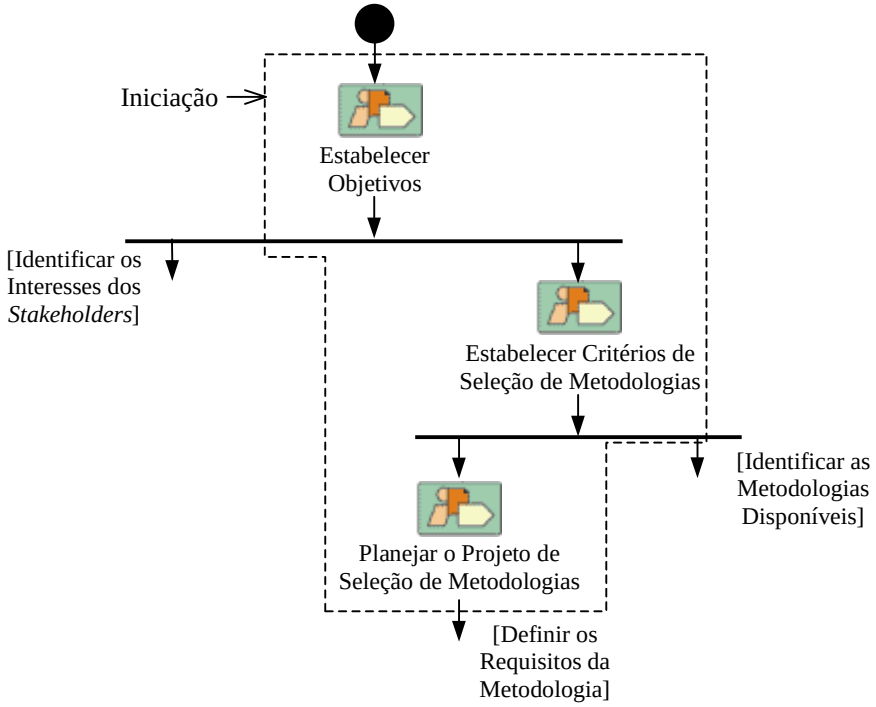


Figura 4. 2 – Atividades da Iniciação.

4.2.1. Estabelecer os Objetivos

Os objetivos para a seleção de uma metodologia de desenvolvimento de *software* devem estar relacionados com o projeto a ser desenvolvido. Este, por sua vez, se relaciona com:

- A visão e missão da empresa, seus interesses no mercado e seus recursos disponíveis, entre outras considerações, as quais servem para justificar a necessidade da aquisição (ou substituição) de uma metodologia;
- Estabelecer expectativas do mercado a ser atingido.

- A política organizacional a ser perseguida;
- Garantir orçamento, recursos e treinamentos;
- Designar responsabilidades.

No caso do exemplo, o objetivo da organização é atingir o mercado das grandes empresas, que se traduz em projetos que atendam àqueles clientes. Dessa forma os objetivos procurados estão relacionados a uma metodologia que ofereça uma boa documentação para o cliente bem como pontos de controle para o gerenciamento do projeto como diagramas de casos de uso, diagramas de classes, entre outros.

4.2.2. Estabelecer Critérios de Seleção de Metodologias

Os critérios de seleção decompõem os objetivos e expectativas a fim de permitir uma decisão de seleção, definir a importância do critério de seleção com base nos objetivos estabelecidos, determinar o cenário de avaliação e seleção, bem como o nível de detalhe e a natureza das atividades de avaliação que serão executadas.

O principal objetivo dos critérios de seleção é identificar os requisitos principais que a metodologia precisa atender a fim de subsidiar a identificação das metodologias disponíveis. O critério de seleção pode também estabelecer quais requisitos serão usados como critério de desempate entre as metodologias que apresentarem um mesmo resultado quantitativo.

Para alcançar os objetivos do exemplo, os critérios de seleção devem levar em conta não só o produto final como também o processo de desenvolvimento. Um dos critérios a considerar é que o processo deverá suportar várias equipes na execução das atividades. Sendo assim, é necessário que o mesmo possibilite uma forma de acompanhamento adequado destas atividades pelos gerentes de projetos. Em outras palavras, o critério de desempate levará em conta as facilidades para o gerenciamento do projeto.

4.2.3. Planejar o Projeto de Seleção de Metodologias

Após a definição dos objetivos e critérios de seleção, o planejamento do projeto deve ser desenvolvido, mantido atualizado e deve conter:

- Uma equipe da organização responsável pelo processo de seleção;
- Um conjunto de objetivos de seleção a serem alcançados, derivados dos objetivos predefinidos;
- Os cenários a serem utilizados e possíveis pesos para o critério de seleção;

- Uma programação de atividades com estimativas de recursos necessários e previsão de conclusão;
- Um projeto piloto, o qual será utilizado para validar a metodologia selecionada.
- Uma maneira de monitorar e controlar a execução do plano.

Para o exemplo, possivelmente haverá necessidade de contratação de profissionais para análise e desenvolvimento; treinamento adequado e serviços de consultoria a fim de alcançar os níveis de qualidade exigidos pelo mercado.

4.3. Fase de Estruturação

O ponto chave da fase de estruturação consiste no estabelecimento dos interesses dos *stakeholders*. A identificação destes interesses pode levar a divergências (que pode ser necessário retornar para os objetivos da fase inicial) relacionadas a: falha no entendimento dos objetivos; em virtude de novas oportunidades ou restrições do mercado; e incompatibilidades entre os requisitos não-funcionais estabelecidos [1].

As atividades desta fase são apresentadas na figura 4.3 e descritas a seguir.

4.3.1. Identificar os Interesses dos Stakeholders

O processo de seleção de uma metodologia depende de restrições que são impostas por fatores externos, tais como o mercado, bem como por fatores internos à organização, tais como a cultura organizacional. Os fatores externos são preponderantes uma vez que eles determinam a sobrevivência de uma organização, pois em primeiro lugar, uma organização depende de seus clientes (o mercado) para sobreviver, mas nem por isso os outros fatores são menos relevantes. Desse modo, os fatores internos devem ser tratados observando-se a direção imposta pelo mercado que se deseja atingir. Em outras palavras, os fatores internos devem seguir a estratégia da empresa visando atingir um determinado nicho de mercado. Para isto é necessário colher informações sobre a organização, o mercado e a tecnologia bem como sobre o projeto de *software* que será desenvolvido com a metodologia.

Os interesses dos *stakeholders* são constituídos pelas seguintes atividades: Avaliar a Organização; Identificar o Mercado; Avaliara a Tecnologia; Alinhar as Regras de Negócio, Estratégias e Políticas; Dimensionar a Capacitação das Pessoas; Determinar a Cultura da Organização; Definir a Criticidade do Sistema e as Prioridades do Projeto.

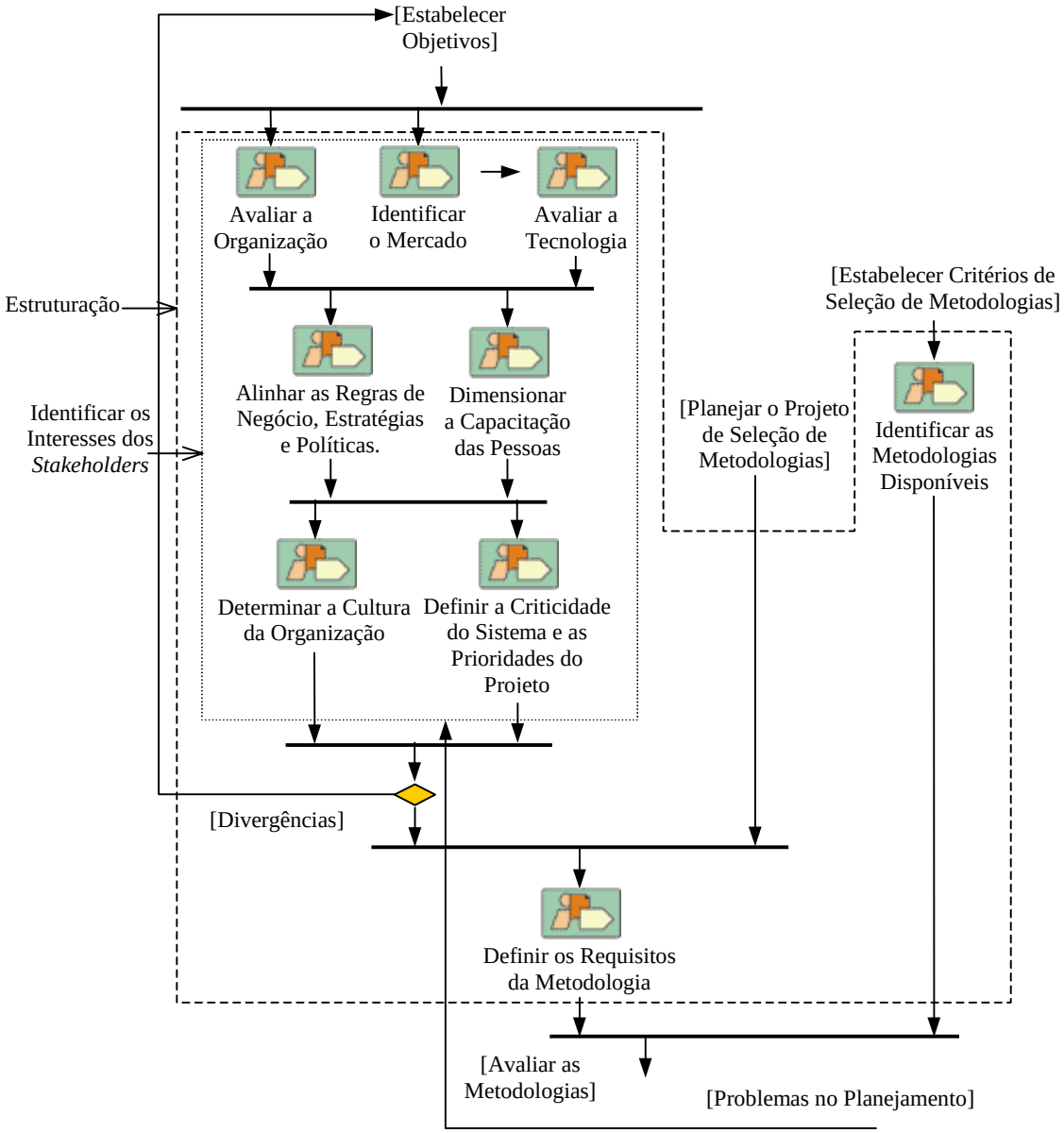


Figura 4. 3 – As Atividades da Estruturação.

Avaliar a Organização

A organização pode ser avaliada levando-se em conta tanto as influências da metodologia, no âmbito interno, como o mercado, no âmbito externo. No âmbito interno, a metodologia influencia nas decisões, propósitos, políticas, princípios, padrões, necessidades, expectativas, habilidades, cultura da organização e vice-versa. Por exemplo, há culturas que são inflexíveis com relação aos desenvolvedores e somente o projetista sênior toma as decisões. Isto faz com que a equipe fique desencorajada para inovação uma vez que está impedida de tomar decisões. Os desenvolvedores se sentem incapazes de se comunicarem com os clientes e mesmo com os outros desenvolvedores.

Se o projeto a ser desenvolvido for distribuído, entre equipes espalhadas geograficamente, é necessário que a cultura seja diferente ou o projeto poderá fracassar.

O ambiente externo exerce influência sobre a metodologia com os padrões, as políticas globalizadas, as tendências tecnológicas e o estágio do mercado no qual a organização pretende atuar. Para obter sucesso com uma nova metodologia num mercado em turbulência, primeiro é imperativa uma mudança de cultura organizacional para depois adaptar a cultura da equipe de desenvolvimento. Caso contrário, permanecendo com uma cultura incompatível com o mercado tecnológico e com a metodologia, o projeto pode não obter os resultados esperados.

O objetivo da avaliação organizacional é desenvolver um documento que identifique o estado atual do corpo funcional, do processo e da tecnologia, identifique um conjunto de melhorias a serem aplicadas dentro da organização, e antecipe as barreiras culturais relacionadas à nova metodologia. Tal documento deve identificar áreas onde a metodologia apresenta pontos positivos e onde ela é inadequada.

Como já foi visto no capítulo 2, o conceito de metodologia é abrangente e envolve questões relacionadas aos processos organizacionais que dão suporte ao processo de desenvolvimento de *software*; à capacitação das equipes de desenvolvimento; às ferramentas que são suportadas pela tecnologia na qual ocorrerá o desenvolvimento do projeto; bem como à tecnologia na qual o produto entrará em produção, padrões, e notações entre outras. Dessa forma certas questões relacionadas à escolha da metodologia podem incluir, por exemplo:

- *Qual a estabilidade, ou seja, a taxa de rotatividade, da equipe de desenvolvimento?*

  Uma alta taxa de rotatividade pode indicar insatisfação que deve ser investigada a fim de evitar a continuidade deste fluxo. No âmbito da metodologia existe a necessidade de que as atividades do processo de desenvolvimento produzam uma documentação suficiente para garantir a continuidade de um projeto na ausência de determinadas pessoas.

- *Há alguns obstáculos específicos no contexto? Quais?*

  Um obstáculo, por exemplo, pode ser rejeição à mudança. Isto pode indicar a necessidade de um bom planejamento, que leve em conta o treinamento adequado, esclarecimentos das dificuldades dos processos, as facilidades no manejo das ferramentas, entre outras medidas. Os obstáculos quando não removidos provocam insatisfação que podem se propagar para os demais *stakeholders*, de tal forma que inviabilizem a efetiva implantação da metodologia.

- *Há estruturas de apoio de comunicação entre os desenvolvedores? E entre os desenvolvedores e os clientes, ou entre os desenvolvedores e os gerentes? Quais?*

  Uma estrutura de apoio, por exemplo, pode ser a garantia de acesso do desenvolvedor ao cliente e vice-versa. É fundamental para o cliente ter uma visão clara do desenvolvimento do projeto, bem como saber o quanto vai lhe custar determinado requisito em termos de dificuldades de desenvolvimento. Para o desenvolvedor é fundamental que o mesmo saiba o que é mais importante a ser desenvolvido para o cliente. A metodologia, através do gerente do projeto e do seu financiador, é responsável pelo perfeito funcionamento das estruturas de apoio, as quais se não forem devidamente estabelecidas, resultam em obstáculos para o projeto.

- *Há mecanismos ou procedimentos estabelecidos que propiciem mudanças, inovações, ou sugestões?*

  A metodologia retrata a filosofia de trabalho da empresa, seus mecanismos disponíveis tais como flexibilidades para obter soluções, relacionamentos entre as pessoas, etc. Estes mecanismos são fundamentais, pois permitem que resultados sejam obtidos de forma mais eficaz para o desenvolvimento do projeto. Cabe à metodologia, através do gerente de projeto e do financiador, assegurar que tais mecanismos possam ser utilizados, para garantir sucesso ao processo de desenvolvimento de *software* como um todo.

- *Há colaboração interpessoal e entre grupos que propiciem tomadas de decisões? Que facilidades permitiriam aumentar a colaboração?*

  A tomada de decisão é tão importante quanto o gerenciamento do projeto, principalmente quando há pressões de tempo para a entrega do produto. A metodologia deve propiciar a colaboração e o gerente de projeto deve encorajar as tomadas de decisões, por parte dos desenvolvedores.

- *A organização apóia as colaborações ou prefere exercer um estilo de controle hierárquico sobre as atividades distribuídas?*

  O estilo de controle hierárquico adapta-se melhor às metodologias pesadas, pois apresentam estrutura de controle semelhante. Por outro lado, as metodologias ágeis exigem maior flexibilidade e estruturas de apoio que permitam maior colaboração entre os membros da organização a fim de compensar a falta de artefatos de comunicação entre os desenvolvedores.

Considerando-se o exemplo, a organização sofrerá o impacto do aumento no quadro de pessoal, o que poderá implicar em mudança cultural e organizacional. Possivelmente a organização terá que se adaptar a uma nova estrutura de controle de tal forma que permita o melhor gerenciamento dos projetos de desenvolvimento de *software*.

Identificar o Mercado

O projeto a ser desenvolvido deverá se encaixar em um dos modelos de mercado que a organização almeja atingir (seção 3.1.3). É imprescindível determinar para qual modelo a organização se dispõe a obter sucesso, pois cada modelo requer diferentes recursos, esforços e tempo de entrega.

Considere por exemplo a trajetória da tecnologia Java [61]. Em meados de 1996, a linguagem Java dava os primeiros passos e nem todo desenvolvedor de *software* reagiu positivamente. Esta foi a fase inovadora da tecnologia (ou do mercado para tal tecnologia), onde só os aficionados por tecnologias revolucionárias correram o risco de uso. Posteriormente, a linguagem Java foi galgando mais adeptos e oferecendo mais recursos até que, alcançou o mercado limitado, dominado pelo interesse dos visionários dos negócios. A partir deste ponto, os pragmáticos nos negócios decidiram investir na inovação quebrando o elo com o passado e dando início a um novo futuro, onde a linguagem oferecendo mais e mais recursos galgou uma nova posição no mercado. Naquele momento pode-se dizer que a tecnologia Java, na fase chamada *promissora*, já dava sinais de explosão no mercado onde continuava a cada três meses apresentando um novo release da linguagem.

Atualmente, a tecnologia Java está bem mais madura, próxima de uma nova fase chamada *consumidora*, onde os que ousaram na tecnologia começam a obter um diferencial considerável em relação aos concorrentes. É interessante perceber que enquanto Java estava na fase *inovadora*, outras linguagens como, por exemplo, C++ estavam na fase *conservadora*. Embora Java tenha atingido o mercado consumidor, continua com inovações tecnológicas como uma forma de permanecer por mais tempo na liderança do mercado.

Outro exemplo é a plataforma .Net [43], que no momento encontra-se na fase *inovadora*. Por outro lado, referindo-se a metodologias, pode-se dizer que RUP está na fase *consumidora* (ou atingiu o mercado consumidor), pois, segundo Charette [7], detém mais de 50% do mercado. Não se pode dizer o mesmo sobre a metodologia OPEN que talvez não tenha conseguido sair da fase do mercado *limitado*, uma vez que detém sua faixa,

mas não se percebe uma explosão de usuários migrando na sua direção. Quanto a XP pode-se dizer que se encontra na fase *promissora*, uma vez que está surgindo, com muita força, em muitas organizações de desenvolvimento de *software*. Crystal ainda é muito pouco conhecida, por isto pode-se dizer que está na fase *inovadora* ou, no máximo, *limitada*.

Para o exemplo, a organização pretende atuar no mercado consumidor, uma vez que é lá onde se encontram os clientes que ela pretende atingir. Dessa forma, é necessário que a mesma se apresente de forma adequada para os seus propostos clientes, os quais não estão dispostos a perder tempo com novas metodologias.

Avaliar a Tecnologia

É necessário avaliar a tecnologia disponível, uma vez que ela suportará o produto de *software* desenvolvido pelo projeto bem como as ferramentas utilizadas pela metodologia, além de influir na cultura da organização, pois exige capacitação das pessoas e impõe novas formas de trabalho. Portanto, cabe aos projetistas seniores interagirem com os desenvolvedores para que sejam analisados aspectos técnicos; as dificuldades de entendimento; as restrições impostas; e as facilidades de uso, de disseminação para os outros desenvolvedores, de performance nas redes, etc. Se a tecnologia apresentar muitas dificuldades de implantação, inviabilizará a adoção da metodologia.

Esta avaliação consiste de uma lista de tecnologias que se candidatam a refletir os benefícios esperados pelos *stakeholders* e que estejam dentro das estimativas de custo orçadas.

Algumas questões relacionadas podem ser incluídas, tais como:

- *Quanto tempo é necessário para que uma tecnologia seja adaptada ao ambiente organizacional, ou para se adequar às mudanças de requisitos da organização?*
  Se a metodologia usa uma tecnologia que exige um longo tempo para sua implantação ou apresenta grandes dificuldades de adaptação, pode inviabilizar sua utilização. A decisão, que envolve o tempo versus adequabilidade da tecnologia deve ser negociada com o financiador do projeto. Cabe a este decidir, baseado nas estratégias empresariais e nos riscos do projeto, o que é mais rentável para o seu negócio a curto, médio e longo prazo.
- *Quais são os aspectos mais significativos da tecnologia que levam em conta os requisitos da organização? Quais são os pontos fortes e fracos da tecnologia?*

A metodologia envolve tanto a cultura da organização quanto a tecnologia que a suporta. Portanto, quanto mais próximos forem os aspectos da tecnologia aos requisitos da organização, melhores condições existirão de se obter sucesso. Em outras palavras, uma metodologia que necessite de uma tecnologia que contraste com a sua cultura, terá que superar grandes dificuldades.

- *Qual o custo esperado (em termos de tempo e dinheiro) para adquirir, implantar, e manter uma tecnologia (incluindo custos de aquisição, manutenção, e capacitação)? Será que estes custos inviabilizarão o projeto de aquisição a longo prazo?*
  Uma metodologia baseada numa tecnologia de alto custo precisa justificar seu uso através dos resultados esperados. Dependendo do tipo do projeto, do foco e das exigências do mercado, justifica-se o uso de uma metodologia que faça uso de tais recursos, caso contrário, deve-se usar o mínimo de recursos possível.

- *Qual a freqüência de atualização da tecnologia?*
  Uma vez que a metodologia faz uso da tecnologia, se esta sofre constantes atualizações, é necessário que a equipe de desenvolvimento esteja apta a acompanhar a evolução para usufruir das vantagens. Isto requer recursos para capacitação que vão desde considerar um maior tempo de entrega do produto, para que a equipe se capacite às inovações, até contratação de consultoria ou terceirização a fim de acelerar o processo de aprendizagem.

- *Que diferencial de valores esta tecnologia pode propiciar aos projetos em relação aos concorrentes?*
  Deve ser analisado se a tecnologia oferece vantagens competitivas ao negócio. Em outras palavras, a metodologia deve ter como foco o negócio que, por sua vez, deve ser suportado pela tecnologia. Caso a tecnologia seja muito competitiva mas não atenda ao negócio, não deve ser adotada.

- *A* tecnologia *está em que fase de seu ciclo de vida no mercado tecnológico? Qual a tendência do mercado?*
  Uma tecnologia pode estar dominando o mercado, porém sendo profundamente ameaçada por novas tecnologias e o mercado pode estar dando sinais de mudança. Pode ser conveniente, se for possível, esperar um pouco para ver se a tecnologia representa uma tendência ou se não passa de um simples modismo no mercado. Esta decisão deve ser tomada pelo financiador que deve discutir com o gerente de projeto, observando o mercado, a cultura da organização e o tipo de projeto.

- *Qual o percentual de aceitação no mercado? Há outra tecnologia com potencial de superação?*

  Se uma tecnologia apresenta um baixo índice de aceitação no mercado, isto indica que a mesma está na fase embrionária, ou que apresenta dificuldades de capacitação, ou de adaptação a mudanças, entre outras. Devem ser observadas outras possibilidades tecnológicas ou aguardar, se possível, até que a mesma apresente-se mais estável no mercado.

- *Que informações podem* ser *utilizadas para que toda a empresa tire proveito da tecnologia?*

  Mecanismos que facilitem a colaboração, em todos os sentidos, contribuem significativamente para a consolidação de uma tecnologia.

Diante destas questões, as tecnologias que passarem por estes critérios são as pré-candidatas para uma análise mais criteriosa de adequabilidade, com base nos interesses dos *stakeholders*.

Com referência ao exemplo, dependendo da tecnologia a ser utilizada, tal como o desenvolvimento de serviços para *Web*, haverá necessidade de treinamento para o corpo gerencial bem como treinamento técnico para todos os desenvolvedores.

Alinhar as Regras de Negócio, Estratégias e Políticas

O mercado impõe restrições à organização afetando as regras de negócio, exigindo novas estratégias e políticas da organização, além de manter uma grande relação com a tecnologia de um modo geral. Isto tudo exige das pessoas capacitação e adaptações às mudanças, afetando a cultura da organização. Portanto, as regras de negócio, estratégias e políticas precisam estar alinhadas com a visão, missão e objetivos traçados pela alta cúpula da organização conforme ressalta Salviano [57].

Se uma organização não se preocupar no alinhamento das regras de negócio, estratégias e políticas, o que pode ocorrer é um desperdício de esforços, dificuldades de identificar os interesses dos *stakeholders*, queda na qualidade dos produtos ofertados aos clientes, entre outras.

Para o exemplo em questão, é preciso destacar que o objetivo da organização, em decorrência do novo projeto, é atingir o mercado das grandes empresas. Devem-se identificar as ações a serem adotadas de forma que todos participem do mesmo objetivo e cada um possa contribuir para o sucesso do projeto. Também pode ser necessário estabelecer responsabilidades, acompanhadas de eventuais mudanças na estrutura.

Dimensionar a Capacitação das Pessoas

A determinação da capacitação das pessoas é de fundamental importância para a organização pois são as pessoas (capacitadas) que elaboram os produtos os quais garantem a sobrevivência da organização. A capacitação é um processo constante uma vez que as atividades diárias permitem, na maioria das vezes, um aprendizado através das relações com outras pessoas, com clientes, através de pesquisa em grupo ou de forma isolada, com treinamento na própria empresa, em instituições de ensino, ou através de um mentor, entre outros. Uma forma de dimensionar a capacitação de uma pessoa é acompanhar o trabalho desenvolvido por ela ou pelo grupo e verificar a satisfação dos seus clientes internos e/ou externos.

Uma metodologia desenvolvida por pessoas com baixa capacitação provoca gargalhos no desenvolvimento, insatisfação entre os desenvolvedores, problemas para o gerente do projeto, atrasos na programação e, insatisfação do cliente, entre outros.

No exemplo em questão, a organização busca crescer num novo mercado. Conseqüentemente é necessário adquirir recursos humanos a fim de fazer frente às novas atividades. Estes recursos precisam ser escolhidos de forma criteriosa, pois eles se constituirão no maior patrimônio da organização, pois serão o elo entre a organização e o cliente. Dessa forma, todos precisam estar conscientes da missão e dos objetivos da organização, bem como devem ser continuamente capacitados para os novos desafios, sendo avaliados através de seus produtos e serviços.

Determinar a Cultura da Organização

As reações das pessoas se manifestam conforme a sua cultura. Assim, o projeto que será desenvolvido e que se encaixará em um modelo de mercado, deve também ser adequado à cultura organizacional. Caso haja divergências, a metodologia terá dificuldades para sobreviver.

O projeto exemplo tem por objetivo atingir as organizações governamentais, as quais geralmente parte de um mercado consumidor e trabalham segundo uma rígida hierarquia de controle. Como visto na tabela 3.1, para um mercado consumidor/conservador, a cultura recomendada é a de controle, que se integra à hierarquia da organização, e a metodologia a ser usada é do tipo pesada.

Definir a Criticidade do Sistema e Prioridades do Projeto

Baseando-se nos quatro princípios e nas quatro faixas de criticidade de Cockburn descritos na seção 3.3.4, deve-se identificar:

- O número de pessoas no projeto (tamanho do projeto);
- A criticidade do sistema;
- As prioridades do projeto.

As prioridades dos projetos dependem de quem os financiam, e segundo Cockburn, o financiador pode priorizar, entre outros:

- Que o projeto seja disponibilizado o mais breve possível no mercado;
- Que o projeto seja livre de defeito;
- Que o projeto apresente visibilidade do processo.

Dependendo do tipo do projeto, o cliente exigirá um maior ou menor acompanhamento dos produtos intermediários. Podem ocorrer casos em que uma falha no projeto represente perda monetária discreta e, portanto, o tempo de entrega poderá ser priorizado no lugar do acompanhamento do processo do projeto desenvolvido. Contudo, o tempo de entrega é um objetivo que não deve ser desprezado mesmo para projetos nos quais uma falha represente perda monetária essencial.

A criticidade impacta na metodologia no sentido de torná-la mais ou menos pesada (ou ágil). Quanto maior a criticidade, mais necessário será definir uma metodologia mais pesada, a fim de evitar perdas consideráveis.

A prioridade do projeto também impacta na metodologia. Por exemplo um projeto, que apresente uma baixa criticidade, mas que necessite de muitas funcionalidades (um grande escopo) e precise ser entregue num prazo relativamente curto, pode ser que a solução seja aumentar o número de pessoas, usando uma metodologia também mais pesada.

Assim, o tempo versus o escopo, juntamente com as criticidades e prioridades do projeto e segundo a experiência do projetista, determinam o tamanho da metodologia, ou seja, o número de elementos de controle utilizados na metodologia.

Para o exemplo da organização, a criticidade do sistema, depende de como o *software* será usado. Caso o *software* apenas apresente resultados os quais serão analisados para que, posteriormente, sejam tomadas algumas ações, é possível considerar uma criticidade de perda monetária discreta (ou essencial, segundo as implicações da falha no sistema). Se o *software* for responsável por tomar decisões em tempo real, a organização

deve inicialmente considerar o sistema com uma criticidade de perda monetária essencial e, se constatar após analise, que vidas humanas possam ser afetadas, o sistema deve utilizar a criticidade de perda de vida.

4.3.2. Identificar as Metodologias Disponíveis

Após estabelecer os critérios de análise de seleção de metodologias deve-se identificar o conjunto de metodologias disponíveis para o processo de seleção, através das seguintes tarefas:

- Estabelecer um conjunto de prioridades e requisitos críticos que devem ser atendidos pelas metodologias;
- Comparar o conjunto de prioridades e requisitos críticos com as metodologias disponíveis, observando-se os custos relacionados;
- Identificar as metodologias que satisfazem um número suficiente de requisitos críticos, as quais serão as metodologias candidatas para uma avaliação formal.

Esta atividade de coleta de informações e identificação das metodologias pode exigir diversas iterações como a forma mais rápida e eficiente de identificar as metodologias mais promissoras para uma posterior avaliação, a qual necessitará de informações mais detalhadas.

Referente ao exemplo, as metodologias identificadas foram o RUP, o OPEN, o Crystal e o XP.

4.3.3. Definir os Requisitos da Metodologia

Os requisitos da metodologia são coletados e organizados de modo que permitam a seleção da metodologia mais apropriada bem como para facilitar o processo de avaliação. Com base no projeto e na cultura, deve-se identificar e registrar os interesses dos *stakeholders* uma vez que estes guiam os requisitos da metodologia. Thomsett [64], por exemplo, utiliza aspectos técnicos e comerciais para identificar os objetivos e o escopo do projeto que retratam os interesses comuns dos *stakeholders*. Estes objetivos incluem estratégias do desenvolvimento do projeto, qual o nível de qualidade requerido pelos *stakeholders*, quais as considerações e restrições associadas que devem ser documentadas, e finalmente quais os serviços requeridos dos *stakeholders* e gerentes de outros projetos.

Os requisitos das metodologias são não-funcionais. Segundo [1] alguns aspectos 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 requisito não-funcional pode ser essencial para uma pessoa enquanto para outra não seja tão importante);
- Interatividade – eles interagem entre si, assim a realização de um requisito não-funcional pode interferir positivamente ou negativamente em outros requisitos.

Por esses motivos, quase sempre é difícil e complexo elicitar, expressar e quantificar requisitos não-funcionais [20].

Um requisito de uma metodologia pode ser, por exemplo, sua capacidade de adaptação às mudanças no negócio. Em outras palavras, define como a metodologia se relaciona com a cultura da organização e com o ciclo de vida da tecnologia.

Considerando-se o exemplo, os *stakeholders* precisam garantir o alinhamento da metodologia com o mercado Consumidor, sem esquecer de identificar os pontos importantes que refletirão na cultura da organização. Como requisitos do exemplo, serão usados os seguintes:

1. Ciclo de Vida da Tecnologia;
2. Cultura da Organização;
3. Criticidade do Sistema;
4. Prioridade do Projeto;
5. Documentação.

Os valores destes requisitos serão especificados na seção 4.5.1.

4.4. Fase de Avaliação

De posse das metodologias disponíveis e com base nos requisitos da metodologia, parte-se para a avaliação das mesmas (figura 4.4). A avaliação consiste em confrontar cada metodologia disponível com os requisitos da metodologia, analisando e registrando os requisitos observados.

Esta avaliação é fundamental para definir quais os requisitos de metodologia que serão usados no processo de seleção. Esta definição consiste numa decisão gerencial,

baseada unicamente nos interesses dos *stakeholders*, uma vez que não há nenhuma restrição para os requisitos encontrados.

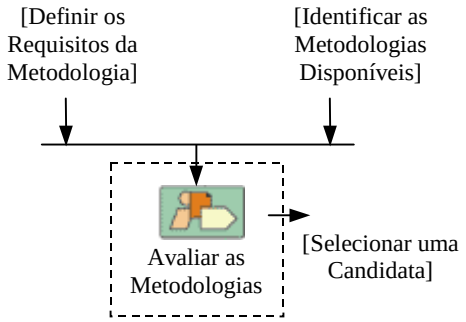


Figura 4. 4 – Atividade da Avaliação.

Apesar de terem sido apresentadas as metodologias de desenvolvimento de *software* RUP, OPEN, Crystal e XP nas subseções de 2.1.1 a 2.1.4, respectivamente, nesta fase serão utilizadas apenas o RUP e o XP, para efeito de simplificação.

4.5. Fase de Seleção

Seleção é um processo de análise de alternativas na qual a pessoa, sozinha ou em grupo, deve escolher a melhor solução para um determinado contexto.

4.5.1. Selecionar uma metodologia

Selecionar uma candidata consiste em atribuir peso às características conforme interesses dos *stakeholders* de tal forma a permitir a seleção de uma metodologia, conforme apresentado na figura 4.5.

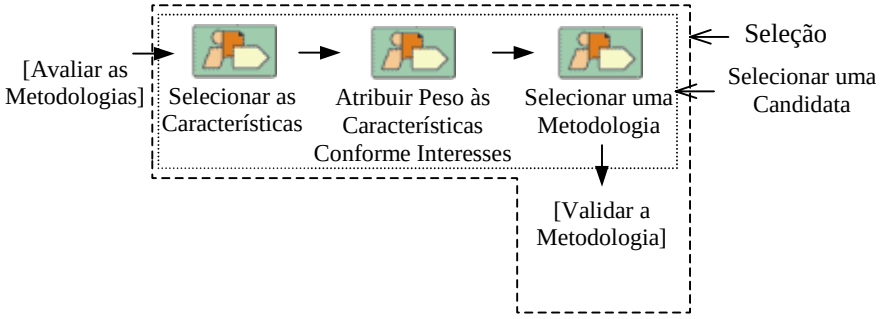


Figura 4. 5 – Selecionar uma Metodologia.

A estratégia para selecionar uma metodologia é um processo amplo e deve ser suportado por importantes decisões gerenciais. A metodologia deve refletir a cultura da organização, os objetivos do mercado a ser atingido e a tecnologia utilizada para o desenvolvimento, que é um dos alicerces para alcançar os objetivos. A seleção deve ser

um esforço de colaboração entre todos os *stakeholders* que buscam alcançar os objetivos do projeto.

Considerando a influência do mercado sobre a cultura organizacional e adaptando a estrutura organizacional de tal modo que facilite o estabelecimento da cultura desejada, bem como observando os quatro princípios estabelecidos por Cockburn na seção 3.3.4, é possível selecionar as características que mais se adequam à equipe de desenvolvimento.

A seleção de uma metodologia pode ser obtida entre um conjunto de metodologias candidatas que dão suporte ao conjunto de características[1] representando os interesses dos *stakeholders* do projeto.

Considere, por exemplo, a metodologia RUP (tabela 4.2). Sabe-se que ela corresponde a um tipo de metodologia pesada que domina o mercado numa proporção superior a 50% conforme Charette [7] e, conseqüentemente, pode-se também dizer que o ciclo de vida da sua tecnologia está na fase consumidora sendo, portanto, adequada para a cultura organizacional baseada em controle.

RUP	
Característica	**Valores**
Ciclo de Vida da Tecnologia	Consumidor
Cultura da Organização	Controle
Criticidade do Sistema	Perda de Vida
	Perda Monetária Essencial
	Perda Monetária Discreta
	Perda de Conforto
Prioridade do Projeto	Processo Visível
	Exatidão
Documentação	Completa

Tabela 4. 2 – Algumas Características da Metodologia RUP.

De forma semelhante, pode-se considerar a metodologia XP (tabela 4.3) a qual corresponde a um tipo da metodologia ágil. O ciclo de vida da tecnologia atende ao mercado inovador, ao mercado limitado e ao mercado promissor; é adequada para organizações que apresentam cultura de competência ou colaboração; seu processo comporta criticidade de projeto que impliquem em perdas de conforto e perda de quantia monetária discreta; atende a projetos com curto espaço de tempo para sua conclusão e

[1] É importante compreender que estas características são apenas sugeridas. Não constituem, de forma alguma, um conjunto completo, ou definitivo, das características de qualquer das metodologias descritas nas tabelas 4.2 e 4.3.

mesmo assim garante sua exatidão, sem oferecer uma completa documentação para o cliente.

XP	
Características	**Valores**
Ciclo de Vida da Tecnologia	Embrionário
	Limitado
	Promissor
Cultura da Organização	Competência
	Colaboração
Criticidade do Sistema	Perda Monetária Discreta
	Perda de Conforto
Prioridade do Projeto	*Software* o mais breve
	Exatidão
Documentação	Apenas o código

Tabela 4. 3 – Algumas Características da Metodologia XP.

Assim, quando os interesses dos *stakeholders* são identificados, os mesmos são relacionados às características das metodologias que os suportam. Para cada característica atribui-se um peso, onde quanto maior o peso maior a sua importância para os interesses dos *stakeholders* do projeto. Para que os valores fiquem próximos e permitam uma visualização gráfica mais adequada, é conveniente utilizar uma escala com poucos valores inteiros, por exemplo de 0 a 4, onde:

- Peso 0. A característica não representa nenhum interesse para o grupo;
- Peso 1. A característica representa pouco interesse para o grupo;
- Peso 2. A característica representa um regular interesse para o grupo;
- Peso 3. A característica representa um grande interesse para o grupo;
- Peso 4. A característica representa o maior interesse para o grupo.

Os pesos podem ser atribuídos de tal forma que representem o consenso dos interesses dos *stakeholder*.

No exemplo, o mercado consumidor está relacionado às organizações governamentais, devido à sua estabilidade, e a rigidez decorrente dos órgãos regulamentadores. Desse modo, para a característica ciclo de vida da tecnologia, deve-se selecionar o valor consumidor atribuindo-lhe o peso 4, de forma que se relacione à estrutura hierárquica de uma organização governamental e para a cultura da organização de controle. A característica criticidade do sistema, de um modo geral, não envolve risco de vida mas, pode ser encarada como um valor de perda monetária essencial, face os prejuízos advindos de alguma ação errada com base nas informações do *software*. Portanto, deve ser atribuído o peso 3 para a perda monetária essencial. Para a característica prioridade

do projeto, serão utilizados três valores: processo visível; *software* o mais breve; e exatidão. Uma vez que será usada uma metodologia pesada, pode-se atribuir peso 2 para o processo visível. Contudo, é fundamental atribuir o peso 4 para a entrega do *software* o mais breve, devido a importância para os negócios governamentais. Como a criticidade não envolve risco de vida, será atribuído o peso 3 para a exatidão do projeto. Com relação à característica de documentação, foi visto que é uma exigência do cliente, mas será atribuído o peso 2, uma vez que, para os projetos, as organizações governamentais dispõem de recursos financeiros limitados.

Baseando-se nos valores das características das tabelas 4.2 e 4.3, os resultados (tabela 4.4), serão os pesos de cada valor de característica, quando a metodologia satisfaz esta característica ou zero, caso contrário. Assim, os totais das metodologias analisadas são 18 para RUP e 10 para XP.

Características	**Valores**	**Peso**	**RUP**		**XP**	
			S	**R**	**S**	**R**
Ciclo de Vida da Tecnologia	Consumidor	4	1	4	0	0
Cultura da Organização	Controle	4	1	4	0	0
Criticidade do Sistema	Perda Monetária Essencial	3	1	3	1	3
Prioridade do Projeto	Processo visível	2	1	2	0	0
	Software o mais breve	4	0	0	1	4
	Exatidão	3	1	3	1	3
Documentação	Completa	2	1	2	0	0
Resultado da Seleção			**18**		**10**	

Legenda: **S** – Satisfaz **R** – Resultado

Tabela 4. 4 – O Peso Representa os Interesses dos *Stakeholders*.

É importante observar que o algoritmo utilizado não descarta uma metodologia em virtude da mesma não satisfazer uma determinada característica. Por exemplo, XP não satisfaz todas as características apresentadas na tabela 4.4. O algoritmo de seleção combina os resultados de avaliações de várias metodologias candidatas, fornecendo uma comparação para facilitar a tomada de decisão pelos gerentes responsáveis pela seleção. O resultado da seleção pode ser resumido pela fórmula seguinte:

Resultado da Seleção = **Σ** (**Peso** x **Satisfaz**) o qual aplica-se a cada metodologia considerada. Este resultado pode ser também expresso em um gráfico de barras como mostra a figura 4.6.

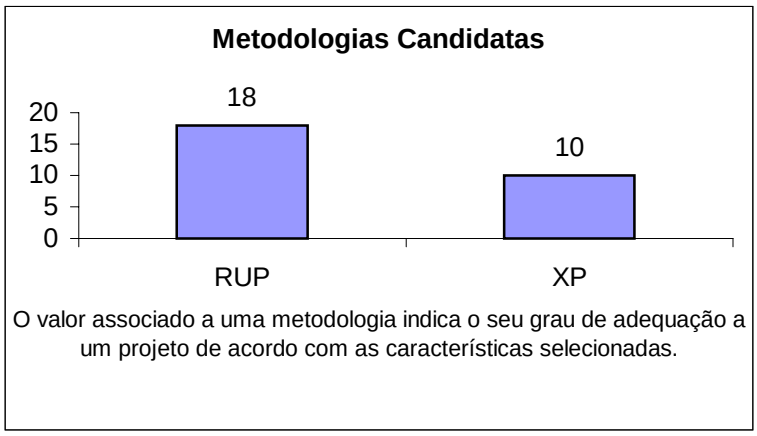


Figura 4. 6 – As Metodologias Candidatas.

Caso o resultado da seleção seja um empate, algumas possibilidades podem ser consideradas, tais como:

- Efetuar uma revisão dos pesos atribuídos para algumas características;
- Alterar alguns dos valores das características;
- Adicionar mais características;
- Eliminar algumas características.

Caso o empate persista, a seleção passa a ser uma decisão gerencial. Vale salientar que a metodologia selecionada deve estar de acordo com a cultura da organização e com o mercado a ser atingido, deve ser justificada e validade contra os objetivos originais.

No exemplo, a metodologia selecionada é o RUP, em virtude de apresentar um maior valor associado quando comparado com a outra metodologia.

A validação da metodologia deve ser feita em um projeto piloto, descrita na próxima seção.

4.5.2. Validar a Metodologia

O processo de validação se baseia na aplicação da metodologia em um projeto piloto de desenvolvimento de *software*, o qual pressupõe que a mesma já foi selecionada. Posteriormente, a metodologia será institucionalizada na organização.

A validação da metodologia (figura 4.7) se dá através da aplicação da mesma no desenvolvimento de um projeto piloto, cujo desenvolvimento consta das atividades planejar, executar, verificar e atuar (no projeto), de tal forma que os erros detectados sejam corrigidos e bloqueados. A figura se mostra adequada para a aplicação do

desenvolvimento baseado tanto no modelo cascata (onde são executados as atividades de Planejar o Projeto, Executar o Projeto e Verificar o Projeto) como no modelo incremental, no qual além das atividades do modelo cascata também contempla "Atuar sobre o Projeto[2]", fechando o ciclo quantas vezes forem necessárias.

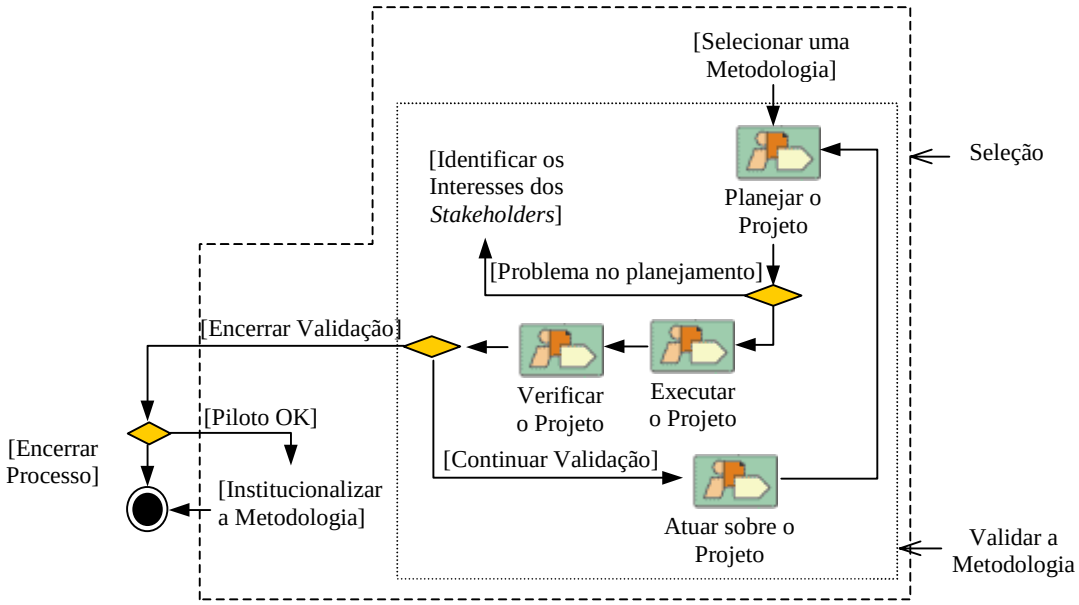


Figura 4. 7 – Validar a Metodologia.

Durante o planejamento do projeto pode ser necessário retornar a Identificar os Interesses dos *Stakeholders* (e rever os requisitos da metodologia), devido a algum problema detectado, como por exemplo, mudanças no negócio, ou alguma incompatibilidade dos requisitos não-funcionais.

O objetivo do projeto piloto é verificar o comportamento da metodologia num curto espaço de tempo, com um número reduzido de pessoas, para que, caso a metodologia se mostre inviável, o custo empregado seja o menor possível.

Como a metodologia aplicada ao projeto piloto aborda vários aspectos da organização, tais como o departamento de recursos humanos preocupa-se com a necessidade de treinamento ou recrutamento, o gerente de projetos procura verificar os benefícios da metodologia para o seu processo, entre outros, merece atenção diária de todos, particularmente do corpo gerencial.

O piloto deve ser tratado com cuidado e consciência, procurando detectar os bons e maus resultados que a metodologia deixa transparecer. O grupo de pessoas no projeto piloto deve estar desejoso de obter os bons resultados que a metodologia possa oferecer,

[2] E também, eventualmente, sobre o processo.

e atento aos possíveis resultados indesejados. É importante dedicar muita atenção, procurar facilitar a comunicação entre os desenvolvedores, e entre estes e o corpo gerencial. Todos os resultados devem ser registrados bem como as razões que contribuíram para os mesmos. Todos devem estar conscientes de que a metodologia pode não ser perfeita. As falhas devem ser devidamente tratadas desde o momento que aparecerem no projeto piloto.

Segundo Oakes [46], e Andriole [2], o projeto piloto deve ter um escopo suficiente para permitir que lições válidas sejam aprendidas a fim de serem aplicadas em projetos maiores. Por exemplo, o piloto deve contemplar um gerente de projeto que elabore o plano do projeto contendo marcos de referência, estimativas de custos e tempo. É importante que o piloto não seja um projeto que possa colocar em risco os processos da organização, mas que seja representativo do tipo de trabalho para o qual a metodologia está sendo avaliada.

É necessário que a metodologia seja aplicada ao projeto piloto, tornando claro para os desenvolvedores os pontos positivos e eventuais dificuldades que ela trará para a organização.

A seguir, as atividades do processo de validação são descritas.

Planejar o Projeto

Consiste em elaborar um plano de desenvolvimento para o projeto, com uma programação dos recursos (tais como pessoal, equipamentos e consultoria) que serão utilizados e com marcos de referência para entrega dos produtos intermediários e entrega dos releases a fim de garantir que o objetivo do projeto seja alcançado dentro dos prazos previstos, sem ultrapassar as estimativas de custos e funcionando de acordo com as expectativas do cliente.

Durante o planejamento, podem surgir informações que levam à modificação dos interesses dos *stakeholders*. Por exemplo, podem ser identificados pontos negativos na metodologia, incompatibilidades entre esta e os recursos disponíveis, ou alterações na disponibilidade dos recursos, de tal forma que se torne imperativo retornar para a identificação dos interesses dos *stakeholders*.

Com referência ao exemplo, o projeto piloto terá um escopo reduzido a fim de possibilitar resultados num prazo máximo de trinta dias, face restrições imposta pela organização a fim de possibilitar a institucionalização da metodologia, caso a mesma seja aprovada ou

reavaliar os requisitos da metodologia. A organização considera que neste prazo poderá, com a ajuda de uma consultoria obter resultados satisfatórios.

Executar o Projeto

A execução do projeto usa as técnicas e processos adequados a cada metodologia, definidos no planejamento. Tal atividade proporciona um aprendizado nas técnicas, através de um relacionamento cooperativo entre todos os desenvolvedores.

No caso do exemplo, a metodologia RUP faz uso de iterações que serão executadas nas atividades de Planejar o Projeto, Executar o Projeto, Verificar o Projeto, e Atuar sobre o Projeto

Verificar o Projeto

Esta atividade pode ser feita, ao longo do desenvolvimento do projeto, usando testes; em reuniões onde os participantes procuram identificar falhas do código em desenvolvimento, nos requisitos especificados na arquitetura; etc.

Durante o projeto, as capacitações das pessoas podem ser alteradas; o tempo pode impor novas necessidades ao mercado; os interesses dos *stakeholders* podem ser modificados, provocando uma mudança na cultura da organização; etc. Tudo isso contribui para um fluxo contínuo de pequenas mudanças que, se não forem atendidas num curto ou médio espaço de tempo, podem comprometer o resultado final do produto esperado.

As conclusões desta atividade podem levar a encerrar o projeto, a atuar sobre o projeto; ou simplesmente a institucionalizar a metodologia conforme os planos.

No caso do exemplo, graças ao escopo reduzido, e à consultoria, foi verificado que, após pequenas atuações sobre o projeto, concluiu-se a validação da metodologia com um build do piloto.

Atuar sobre o Projeto

De posse dos possíveis problemas, levantados na etapa anterior, deve-se pesquisar meios de impedir que os mesmos se repitam.
Os resultados dos esforços do projeto piloto devem ser avaliados para verificar se é necessário aplicar um novo projeto piloto a fim de dirimir dúvidas. Caso não se faça necessário, o próximo passo consiste na institucionalização da metodologia.

No caso do exemplo, as dificuldades do desenvolvimento do projeto se concentraram na elicitação dos requisitos, na identificação das interfaces do sistema, nos casos de uso, e na passagem dos casos de uso para a codificação. Estes problemas superados, servirão como base para enfrentar as dificuldades na institucionalização da metodologia, sem deixar de lado o apoio da consultoria.

4.6. Fase de Institucionalização

A institucionalização do uso da metodologia é uma etapa que dará continuidade ao desenvolvimento de projetos com a diferença que desta vez, a metodologia será implantada em toda a organização (figura 4.8).

Nesta fase, é importante que as dúvidas, em sua grande maioria, estejam esclarecidas e que haja uma quantidade suficiente de pessoas empenhadas a dar suporte às dificuldades que surgirão durante esta fase de implantação da metodologia.

A mesma equipe que participou do projeto piloto deve continuar no projeto de institucionalização, pois isto facilita a disseminação de pontos importantes que foram identificados.

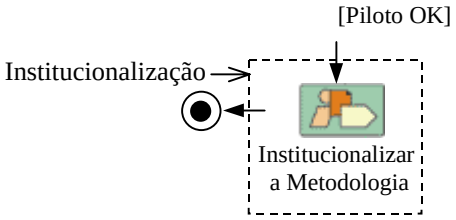


Figura 4. 8 – Institucionalizar a Metodologia.

Uma completa penetração da metodologia na organização pode ser lenta, e uma interrupção do suporte gerencial em qualquer estágio pode comprometer a institucionalização. Os objetivos dos esforços devem ser no sentido de encorajar a adoção da metodologia, manter suporte gerencial, e disseminar a metodologia o mais cedo possível. Isto pode ser conseguido, por exemplo, através de um “clima agradável” para solucionar os problemas, ou de recompensas, por meio de bonificação ou premiação para indivíduos e equipes que obtiverem bons resultados para a organização fazendo uso da metodologia.

No exemplo, a institucionalização da metodologia foi concluída após oito meses, nos quais as dificuldades foram enfrentadas e superadas graças ao apoio da consultoria,

juntamente com a disposição e colaboração de todos que evitaram a rejeição à mudança, e procuraram resolver os problemas e transmitir os sucessos alcançados em cada etapa.

4.7. Considerações Finais

Pressionadas pelo tempo, algumas organizações realizam o processo de seleção de uma metodologia de forma ad-hoc, destinando um curto período de tempo para isto.

O processo proposto é baseado nos requisitos identificados para as metodologias, os quais são indicados pelos interesses dos *stakeholders*, levando em conta o mercado, a tecnologia, a organização, o alinhamento das regras de negócio, as estratégias e políticas, a capacitação das pessoas, a cultura da organização, a criticidade do sistema e as prioridades do projeto de *software*.

Foi visto que o mercado restringe a cultura organizacional, influencia nas prioridades do projeto, e indica a tecnologia a ser adotada. A tecnologia exige capacitação das pessoas que modificam a cultura organizacional. A cultura organizacional e as prioridades dos projetos contribuem para a formação dos interesses dos *stakeholders*. Os interesses comuns buscam os recursos disponíveis para o desenvolvimento do projeto e apontam para as características da metodologia a ser utilizada.

Os interesses dos *stakeholders* representam os objetivos do negócio, a criticidade do projeto, a cultura da organização entre outros são, a base dos requisitos da metodologia usados para selecionar uma entre as candidatas.

As cinco fases do processo proposto, com suas atividades (figura 4.9).

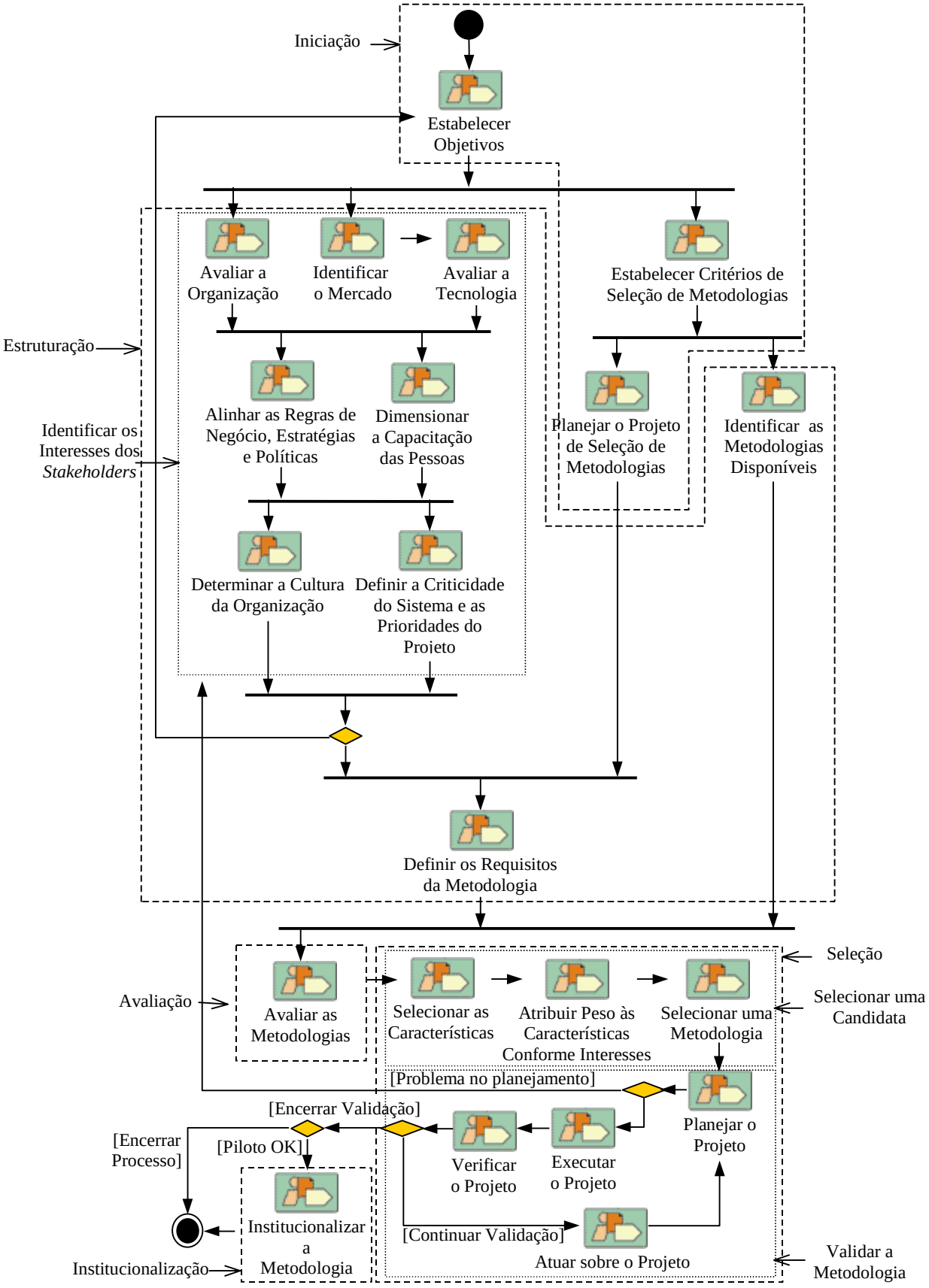


Figura 4. 9 – As Atividades do Processo Proposto.

A tabela 4.5 mostra o relacionamento entre o modelo SA-CMM e o processo proposto. Este relacionamento consistiu na identificação das áreas chave de processo, explícitas no modelo SA-CMM, as quais foram confrontadas com as fases do modelo proposto.

SA-CMM			***Processo para a Seleção de Metodologias***
Nível	***Foco***	***Área Chave de Processo***	***Fases***
5 Otimizado	Melhoria de Processo Contínua	Gerência de Inovação na Aquisição	Iniciação, Estruturação, Avaliação, Seleção, Validação e Institucionalização.
		Melhoria Contínua de Processo	
4 Quantitativo	Gerenciamento Quantitativo	Gerência de Aquisição Quantitativa	
		Gerência de Processo Quantitativo	
3 Definido	Padronização de Processo	Gerência de Programa de Treinamento	
		Gerência de Risco de Aquisição	
		Gerência de Desempenho de Contrato	
		Gerência de Desempenho de Projeto	
		Requisitos do Usuário	
		Definição e Manutenção de Processo	
2 Repetível	Gerência Básica de Projeto	Transição para Suporte	Estruturação, Avaliação, Seleção e Validação.
		Avaliação	Iniciação, Estruturação, Avaliação, Seleção, Validação.
		Supervisão e Acompanhamento de Contrato	Validação.
		Gerência de Projeto	Iniciação, Estruturação, Avaliação, Seleção, Validação e Institucionalização.
		Desenvolvimento e Gerenciamento de Requisitos	
		Solicitação	Iniciação, Estruturação, Avaliação, Seleção e Validação.
		Planejamento de Aquisição de *Software*	Iniciação.
1 Inicial	Pessoas Competentes e Heróis		

Tabela 4. 5 – Mapeamento entre o Modelo SA-CMM e o Processo Proposto.

No próximo capítulo, será apresentado um protótipo para automação da etapa Seleção de uma Metodologia Candidata. O protótipo parte do princípio de que os requisitos da metodologia já foram definidos.

5. Protótipo MeSTool

Este capítulo descreve a ferramenta MeSTool, Methdology Selection Tool, usada para a seleção de uma metodologia a ser implantada. O MeSTool é descrito através de suas janelas, focando nas funcionalidades mais importantes e cenários.

5.1. Visão geral do MeSTool

O protótipo do MeSTool foi desenvolvido em Access-97. Em sua implementação não houve preocupação com relação a requisitos não funcionais, tais como requisitos de desempenho e considerações de segurança. Seu escopo está bem focado na seleção de uma metodologia, no contexto do processo proposto no capítulo quatro, partindo do pressuposto de que os requisitos da metodologia já foram definidos e as metodologias avaliadas, restando apenas efetuar a seleção conforme a figura 5.1.

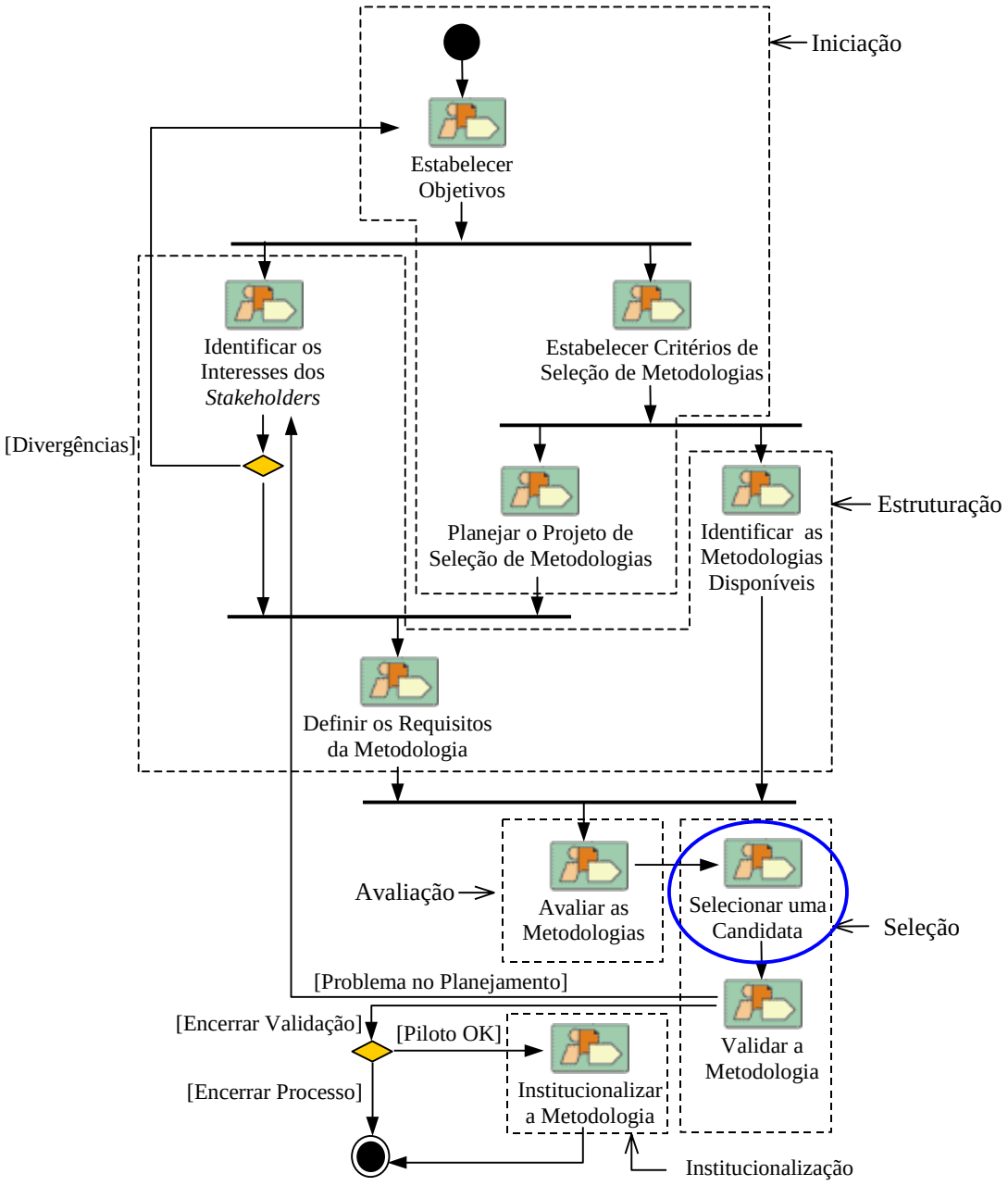


Figura 5. 1 – O Escopo do MeSTool no Processo Proposto.

Para selecionar uma metodologia para um determinado projeto, é necessário que as metodologias candidatas, com as suas características e os requisitos do projeto, estejam cadastradas na base de dados. Cadastrar um projeto na base de dados consiste em definir os requisitos da metodologia que darão suporte ao projeto. O protótipo confronta os requisitos do projeto, expressos em termos de características da metodologia procurada, com as características suportadas pelas metodologias na base de dados. A seleção da metodologia constará da identificação de uma metodologia qualquer que melhor atenda a estes requisitos.

Em outras palavras, selecionar uma metodologia para um projeto de *software* (figura 5.2) consiste em determinar as características do projeto atribuindo-se um peso, para cada uma, representando a importância da mesma para os interesses do projeto. Após esta etapa, seleciona-se a metodologia que melhor represente estes interesses.

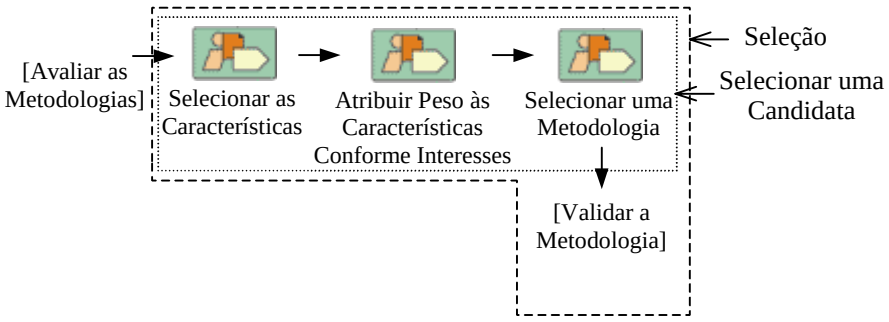


Figura 5. 2 – Os Passos para a Seleção de uma Metodologia.

As próximas seções detalham o protótipo do MeSTool, descrevendo suas funções disponíveis, alguns cenários, e as considerações finais.

5.2. As Funcionalidades e Cenários do Protótipo MeSTool

O MeSTool considera dois atores, um metodologista e um gerente de projeto/projetista, e oferece quatro funcionalidades básicas (tabela 5.1): Cadastrar uma Característica, Cadastrar uma Metodologia, Cadastrar um Projeto e Selecionar uma Metodologia. No anexo 1 estas funcionalidades são descritas na forma de casos de uso.

O principal objetivo do MeSTool é ajudar ao gerente de projeto a selecionar uma metodologia entre as candidatas, a qual suporte os interesses dos *stakeholders.* Estes interesses são representados pelas características escolhidas pelo gerente do projeto, atribuindo-se seus respectivos pesos.

Ator	Funcionalidade	Casos de Uso
Metodologista	Cadastrar Característica (e seus Valores)	Inserir Característica
		Inserir Valor da Característica
		Excluir Valor da Característica
		Excluir Característica
	Cadastrar Metodologia	Inserir Metodologia
		Definir Valores da Metodologia
		Atualizar Metodologia
Gerente de Projeto/Projetista	Cadastrar Projeto	Inserir Projeto
		Definir Valores do Projeto
		Atualizar Projeto
	Selecionar uma Metodologia	Selecionar uma Metodologia (para um Projeto)

Tabela 5. 1 – A Lista dos Objetivos dos Atores.

Inicialmente, Cadastrar Característica, Cadastrar Metodologia e Cadastrar Projeto devem ser executados nesta ordem, uma vez que uma metodologia necessita indicar as características que a mesma satisfaz e um projeto necessita selecionar uma metodologia na base de dados.

No caso do projeto, é necessário indicar as características de interesse dos seus *stakeholders* e confrontá-las com as características das metodologias disponíveis, de modo que se possam identificar quais metodologias suportam um maior número de características, resultando no melhor retorno para o processo.

Estas funcionalidades do MeSTool permitem ao metodologista (pessoa que define uma metodologia) cadastrar uma metodologia, bem como inserir características de metodologia; e ao projetista sênior ou gerente de projeto selecionar uma metodologia que seja mais adequada ao seu projeto.

O funcionamento básico do MeSTool gira em torno das características cadastradas na sua base de dados. Dessa forma, para compreender o processo como um todo, é importante entender que:

1. Para que o MeSTool possa ser utilizado pela primeira vez, é necessário que algumas características (de metodologias) sejam inseridas através da função Cadastrar Característica. Logo após a inserção de uma característica no MeSTool, ela torna-se disponível exclusivamente para que as metodologias possam usá-la. Quando pelo menos uma metodologia usar uma característica, esta se torna disponível também para a função Cadastrar Projeto.
2. Quando um metodologista insere uma metodologia no MeSTool, ele deve indicar quais das características disponíveis a metodologia faz uso. Caso necessite de uma característica não disponível, ele pode aumentar a base de dados, inserindo uma

nova característica para a sua metodologia. Estas características formam o conjunto das características da metodologia. Desse modo, cada metodologia terá seu conjunto de características. É importante lembrar que só as características das metodologias estão disponíveis à função Cadastrar Projeto, para evitar que sejam usadas aquelas características ainda não suportadas por pelo menos uma metodologia.

Tanto a função Cadastrar Projeto (figura 5.3) quanto Selecionar uma Metodologia (para um projeto) usam as características das metodologias. Cadastrar Projeto permite que o usuário escolha as características do seu interesse. Selecionar uma Metodologia é uma decisão gerencial, de responsabilidade do gerente do projeto e, portanto, o MeSTool disponibiliza, para que seja analisada, qualquer metodologia que atenda a pelo menos uma característica do projeto em questão.

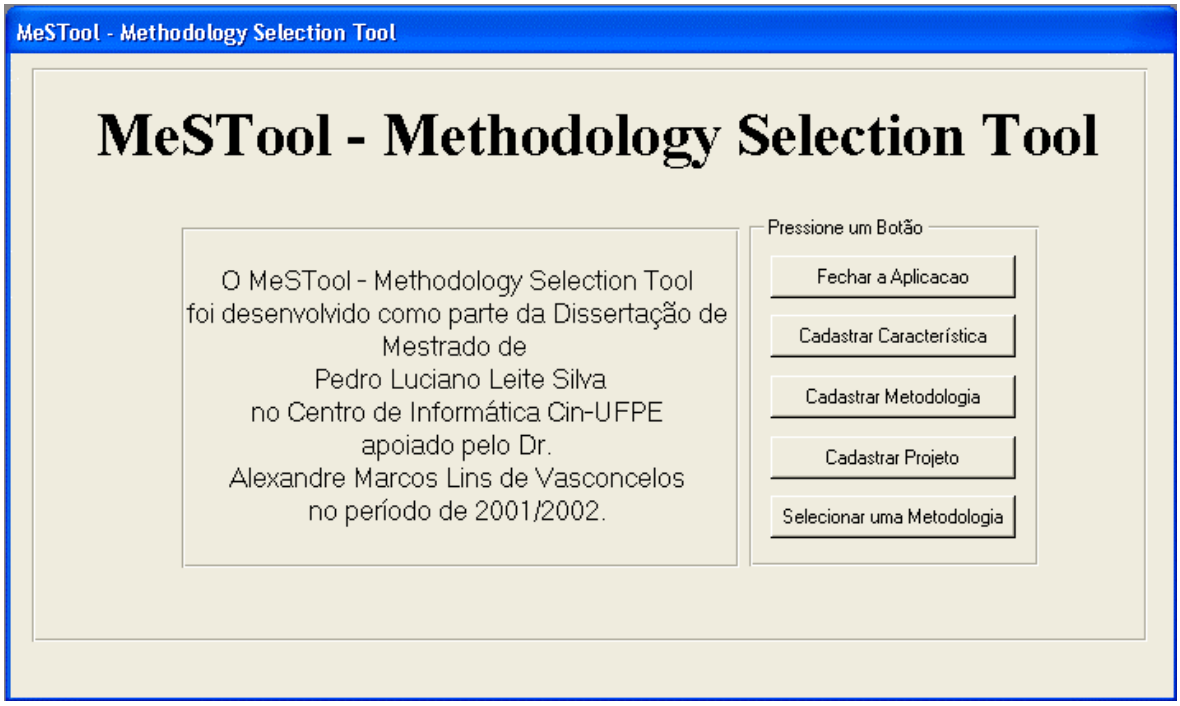


Figura 5. 3 – A Janela Principal do MeSTool.

Para ilustrar as funcionalidades e cenários de uso do MeSTool, será considerado o mesmo exemplo do capítulo quatro, ou seja, uma organização que desenvolve projetos de *software* com uma pequena equipe e decide atingir o mercado das organizações governamentais onde os clientes têm como uma das exigências o acompanhamento dos projetos.

As seções a seguir descrevem as funções no MeSTool, supondo o cenário do exemplo proposto.

5.2.1. Cadastrar Característica

Para cadastrar uma característica, é preciso primeiro, na janela principal do MeSTool, selecionar o botão Cadastrar Característica. Na primeira vez que se usa o MeSTool, ou seja, considerando que não há nenhum dado na base de dados, o sistema apresentará a janela da figura 5.4.

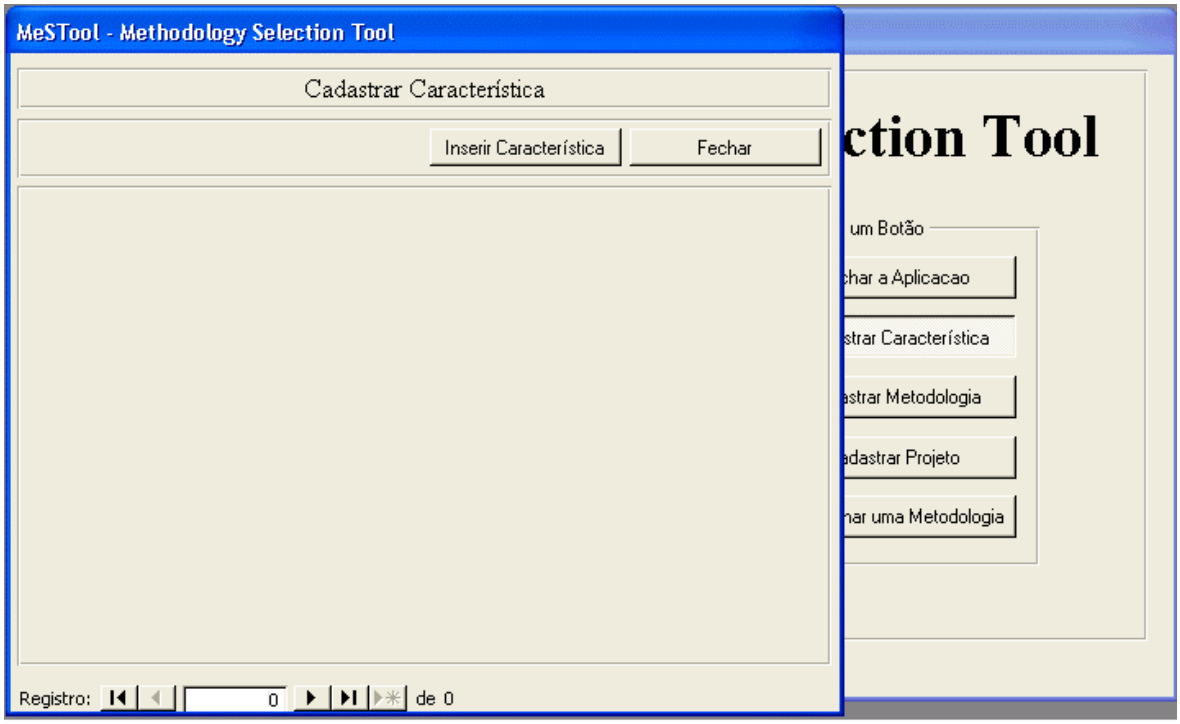


Figura 5. 4 – A Janela Inicial para Cadastrar Característica.

Enquanto não for cadastrada nenhuma característica na base de dados, na janela cadastrar característica permanecerão visíveis os botões para Inserir Característica e Fechar.

Pressionando-se o botão Inserir Característica, aparece a janela Inserir Característica (figura 5.5). Cada característica tem sua Descrição e a indicação Valores Excludentes para uma metodologia. Por exemplo, a característica Documentação apresenta os valores excludentes **completa** e **apenas o código**; já a característica Cultura da Organização permite que a organização assuma mais de um valor entre **colaboração**, **competência**, **controle** e **cultivo**. Dessa forma, ao se cadastrar uma característica deve-se indicar se seus valores são ou não excludentes.

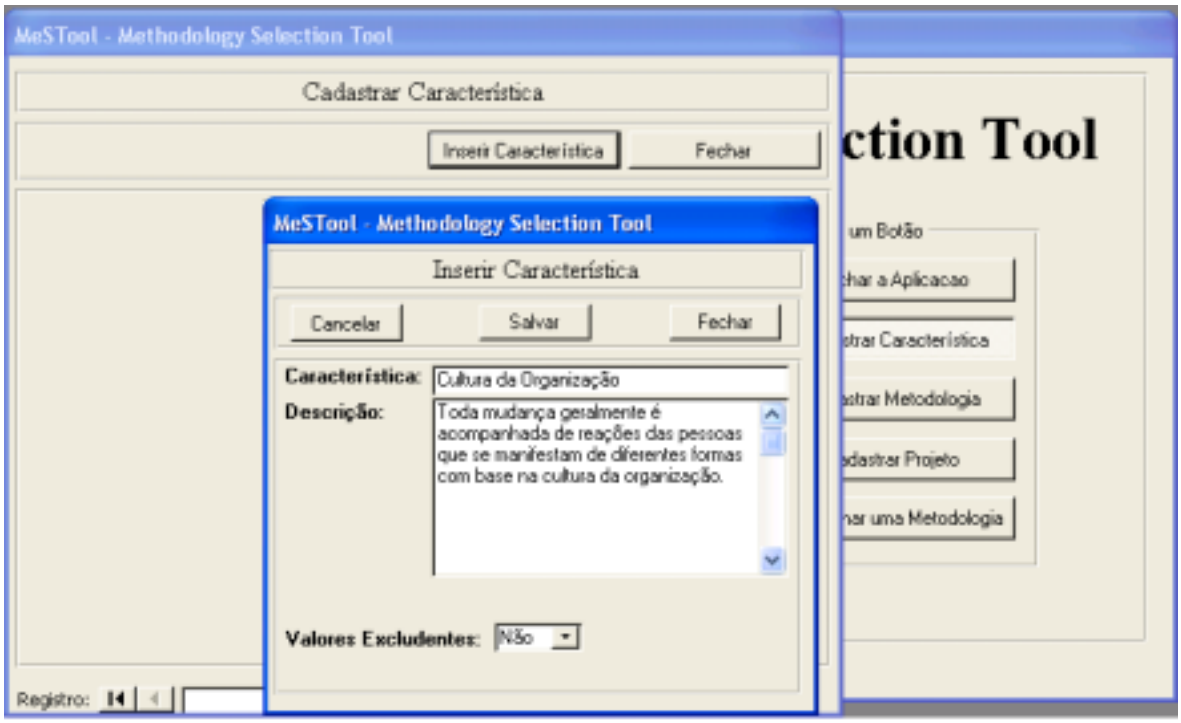


Figura 5. 5 – Cadastrar uma Característica na Base de Dados.

Ao Salvar os dados o MeSTool permite inserir os valores da característica (figura 5.6).

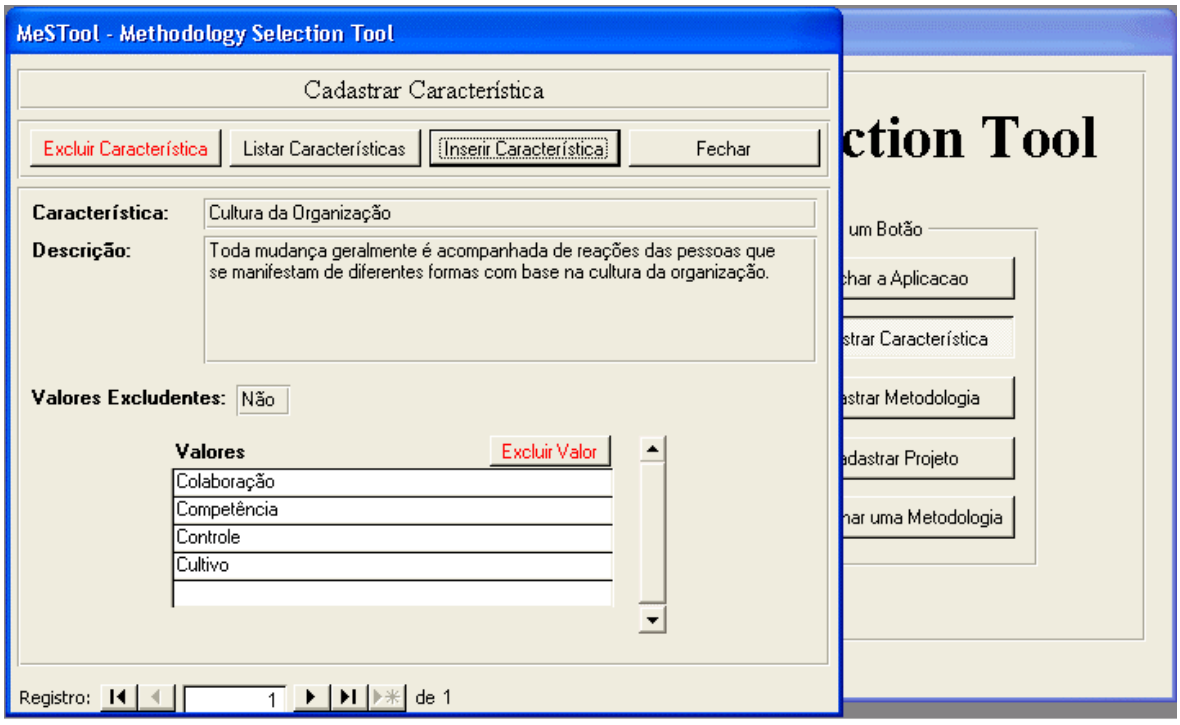


Figura 5. 6 – Inserir Valores da Característica.

Para inserir os valores da característica Colaboração, Competência, Controle e Cultivo, simplesmente digita-se um valor e a tecla “entra”. Então o cursor desloca-se para a linha

seguinte, onde é digitado outro valor. Para excluir um valor, seleciona-se o mesmo e pressiona-se Excluir Valor. Para excluir uma característica, com todos os seus valores, pressiona-se Excluir Característica.

A figura 5.6, indica que há apenas esta característica na base de dados (ainda não utilizada por nenhuma metodologia). Quando uma característica é inserida na base de dados, tornam-se visíveis os botões Excluir Característica e Listar Características.

A lista das características (figura 5.7) é obtida pressionando-se Listar Características na janela Cadastrar Característica. A lista apresenta o nome da característica, sua descrição, a indicação valores excludentes e, o valor da característica acompanhado da sua descrição.

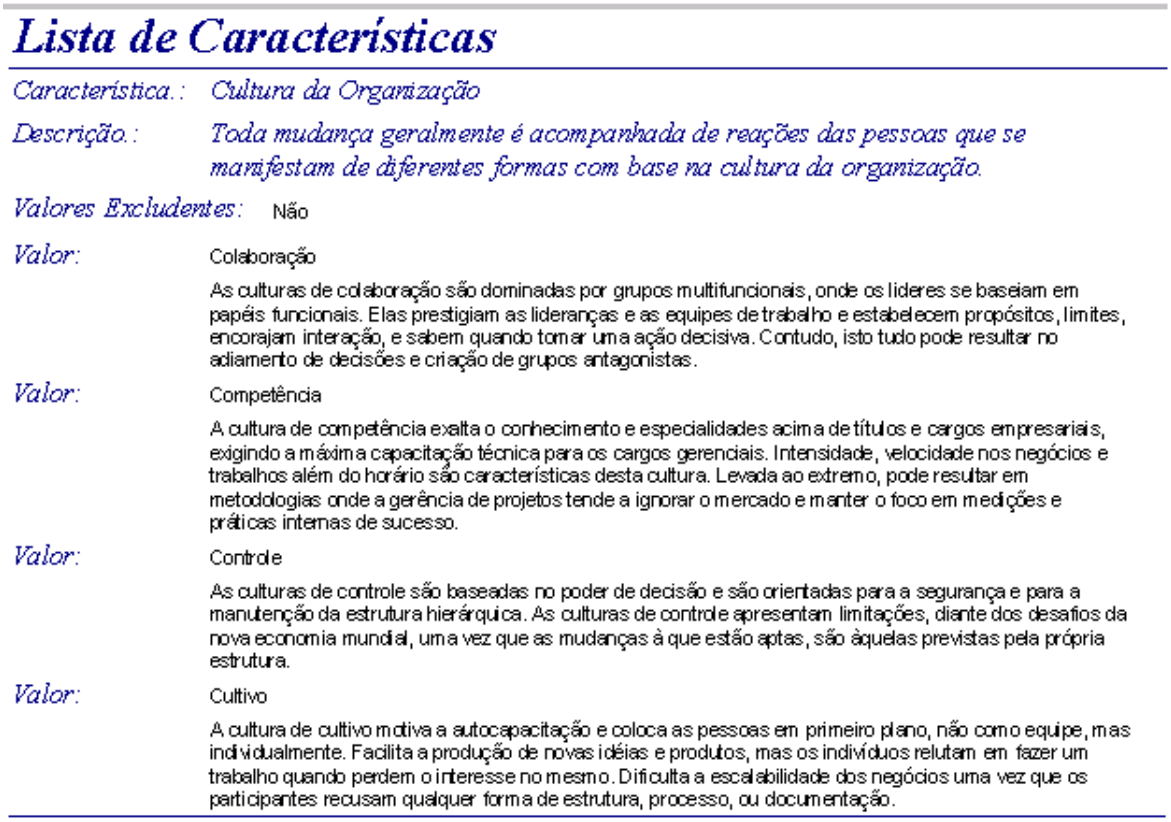


Figura 5. 7 – Lista das Características.

A figura 5.8 mostra que um duplo clique num valor da característica, por exemplo, em Controle, permite que informações complementares sejam adicionadas ao mesmo. Estas informações são fundamentais a fim de dirimir dúvidas tanto para o cadastramento das características, quanto para o cadastramento de uma metodologia feito por um metodologista, ou mesmo quando um gerente de projeto for especificar as características que seu projeto deve exibir.

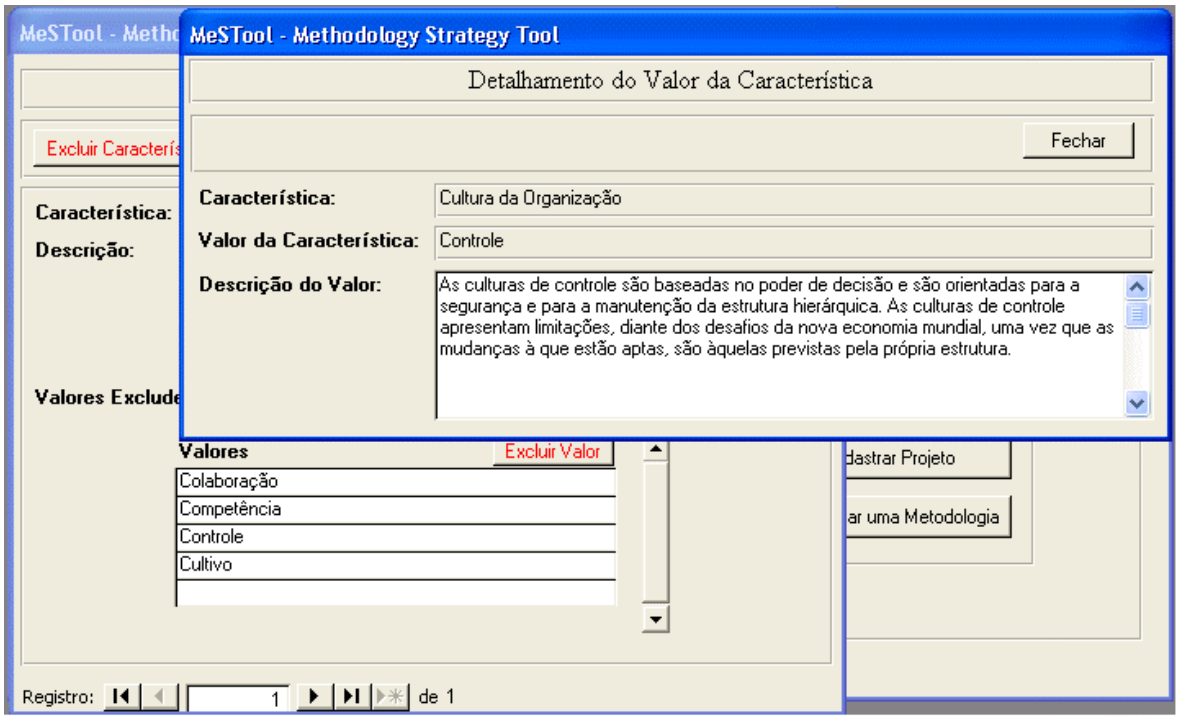


Figura 5. 8 – Detalhe do Valor da Característica.

Na próxima seção, será apresentada a função Cadastrar Metodologia.

5.2.2. Cadastrar Metodologia

Todas as características cadastradas na base de dados do MeSTool estão disponíveis para o metodologista indicar quais são suportadas por sua metodologia.

A figura 5.9 mostra a janela Cadastrar Metodologia na base de dados do MeSTool, onde foi necessário selecionar na janela principal do MeSTool a opção Cadastrar Metodologia. Quando se usa a janela Cadastrar Metodologia pela primeira vez, ainda não há metodologia cadastrada na base de dados do MeSTool, portanto, as únicas opções são pressionar o botão para Inserir Metodologia ou Fechar a janela.

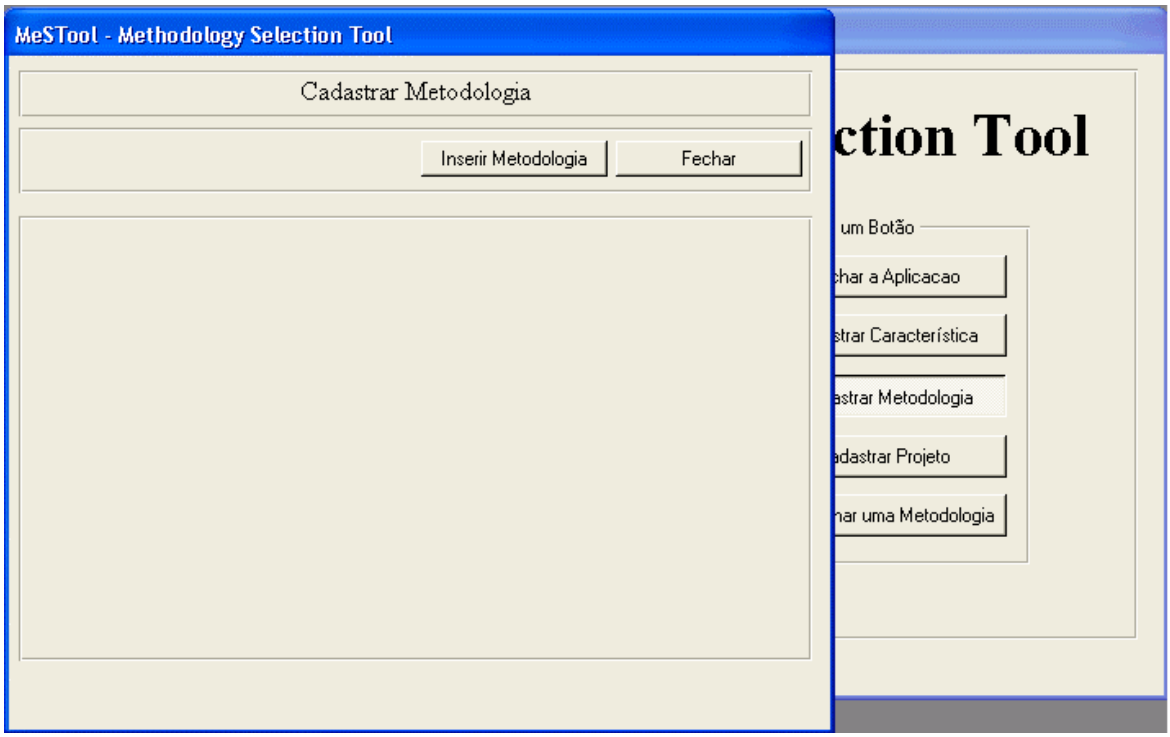


Figura 5. 9 – A Janela Inicial para Cadastrar Metodologia.

Pressionando-se o botão Inserir Metodologia, o sistema mostra a janela para o cadastramento, conforme a figura 5.10.

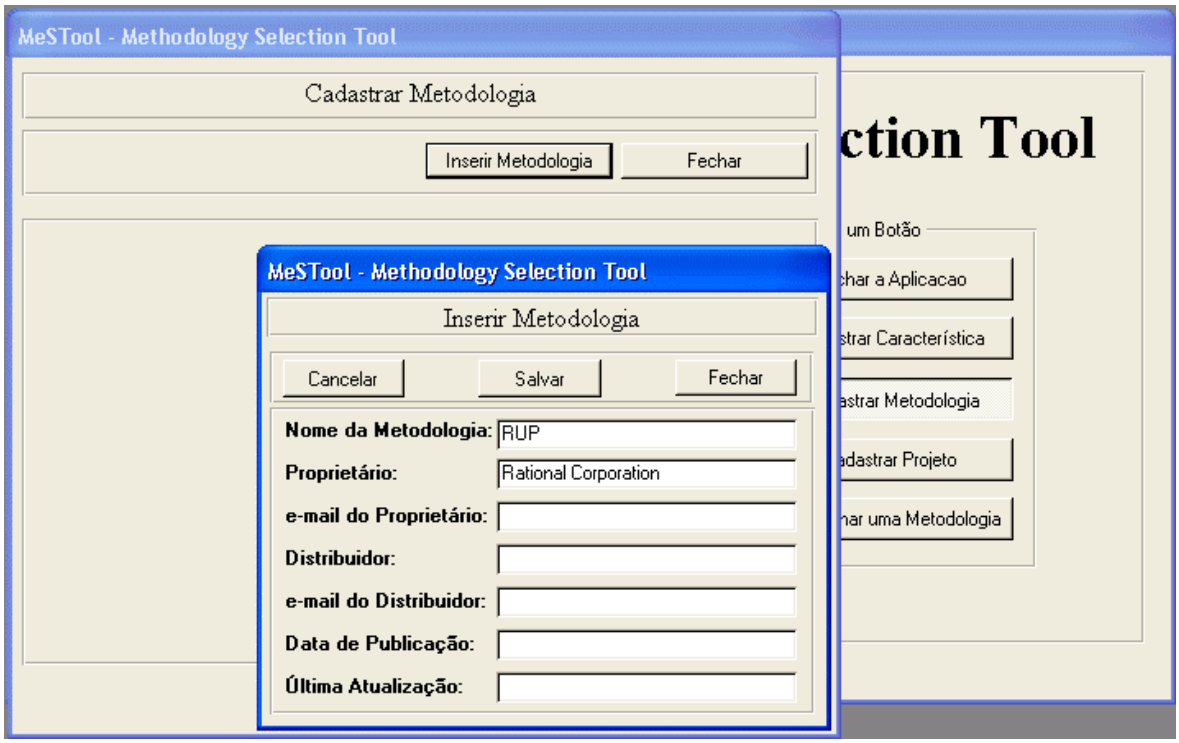


Figura 5. 10 – Inserindo uma Metodologia.

Após as informações sobre a metodologia serem inseridas, pode-se na janela Cadastrar Metodologia (figura 5.11), selecionar uma metodologia e uma dada característica e pressionar o botão Definir Valores da Metodologia selecionada. Esta ação abre a janela, de mesmo nome, a qual mostra, para a característica Cultura da Organização todos os seus valores (Colaboração, Competência, Controle e Cultivo) bem como a informação se a metodologia satisfaz ou não cada valor.

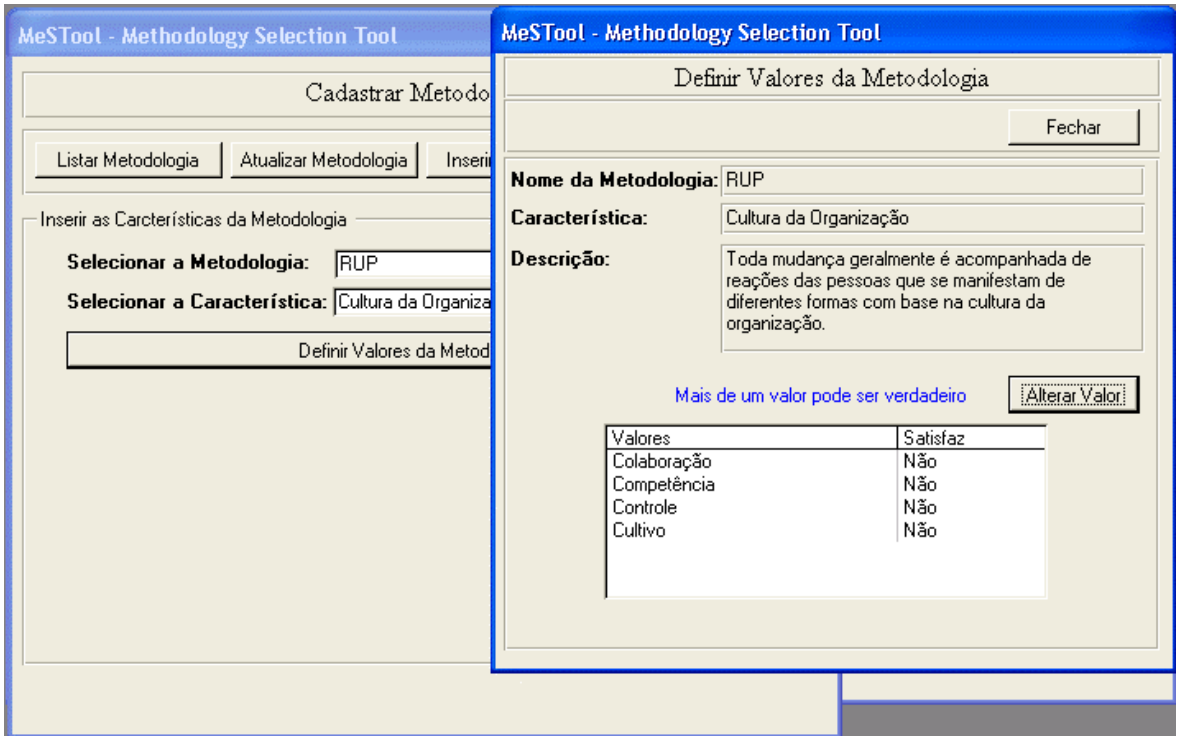


Figura 5. 11 – A Janela Definir Valores da Metodologia.

Para indicar que uma metodologia satisfaz um determinado valor de uma característica, é necessário selecionar o valor desejado, por exemplo Controle (figura 5.12), e pressionar o botão Alterar Valor na janela Definir Valores da Metodologia. A mensagem “Mais de um valor pode ser verdadeiro” ressalta que se Controle permanecer na condição Sim e for feita uma alteração da condição do valor Colaboração para Sim, a condição do valor Controle permanecerá na condição Sim.

Para os casos com a mensagem “Apenas um valor pode ser verdadeiro” indica que se um valor permanecer na condição Sim e for feita uma alteração da condição de outro valor para Sim, a condição do primeiro valor retornará imediatamente para a condição Não.

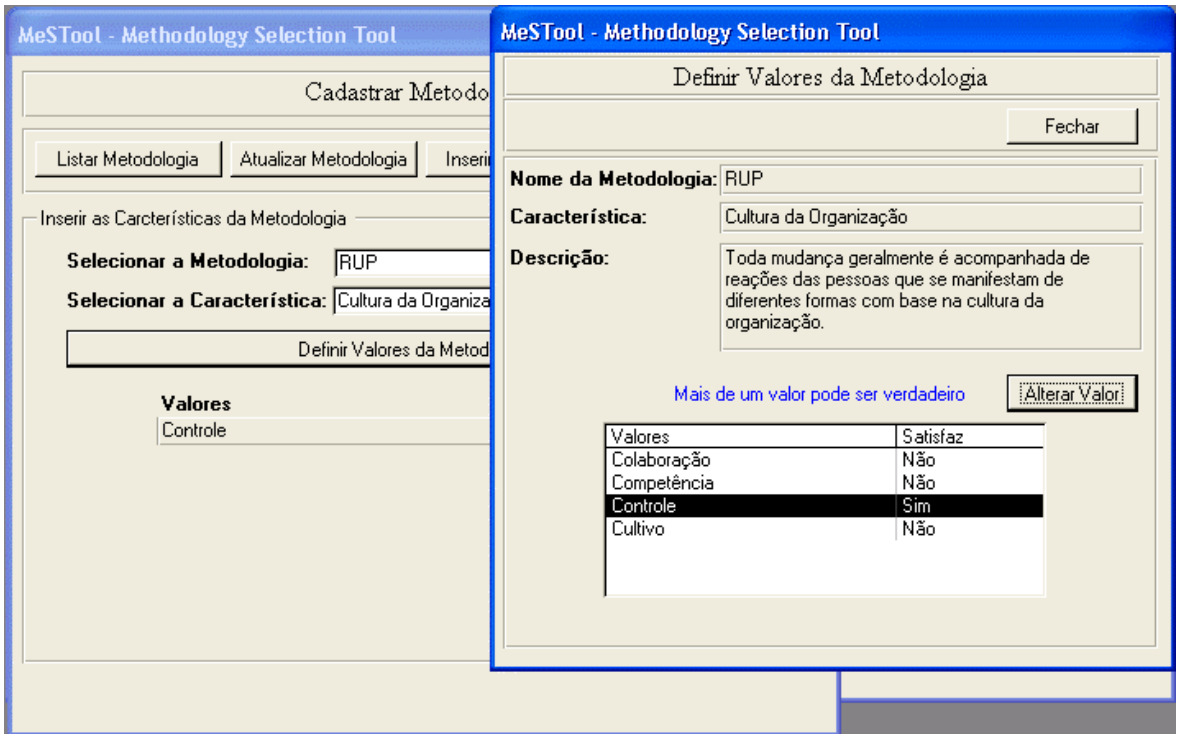


Figura 5. 12 – Definir Valores da Metodologia.

Ao se pressionar Listar Metodologia na janela Cadastrar Metodologia, obtém-se o relatório da mesma, como mostra a figura 5.13 para o caso da metodologia RUP, já cadastrada.

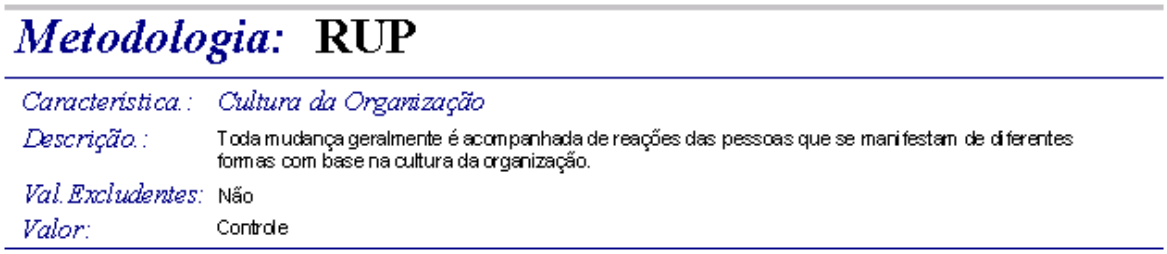


Figura 5. 13 – A Lista das Características da Metodologia.

Os dados do perfil de qualquer metodologia podem ser atualizados em qualquer momento do projeto. Para tanto é necessário selecionar a metodologia e pressionar o botão Atualizar Metodologia, conforme figura 5.14.

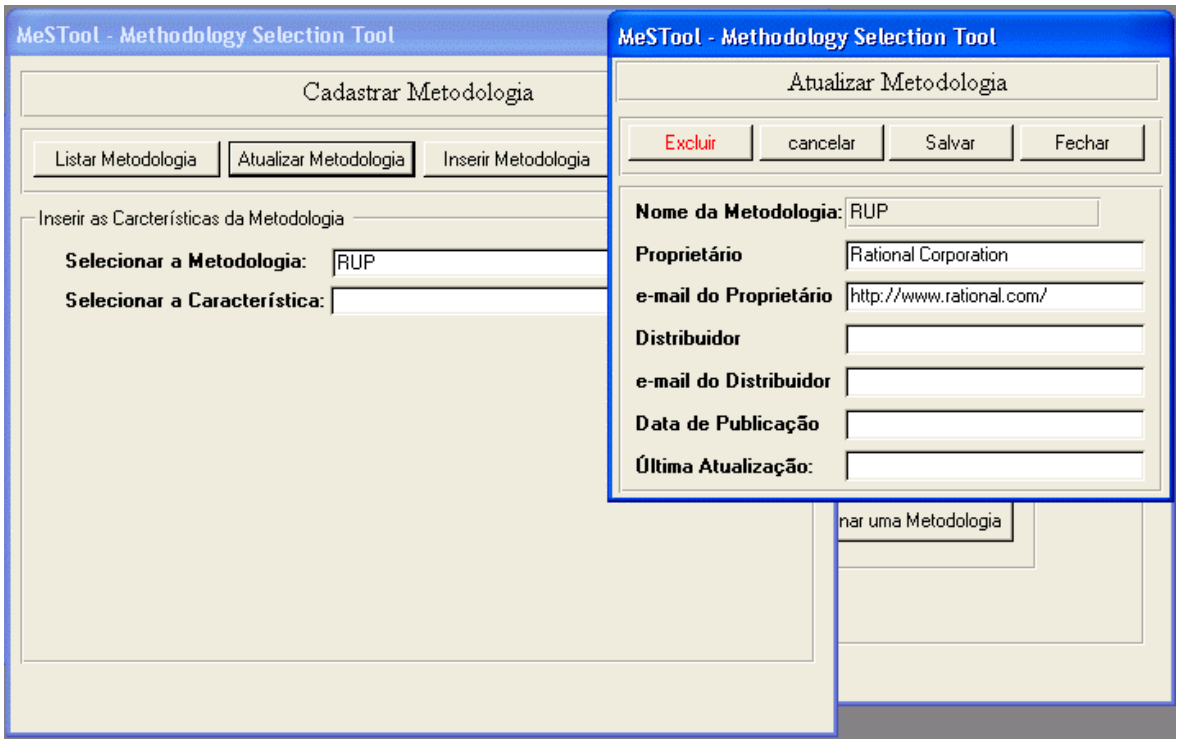


Figura 5. 14 – Atualizar Perfil da Metodologia.

Na próxima seção, será apresentada a função Cadastrar Projeto.

5.2.3. Cadastrar Projeto

Ao se pressionar o botão Cadastrar Projeto na janela principal, o sistema mostra a tela da figura 5.15. No exemplo, como ainda não há projeto na base de dados, só há duas alternativas na janela Cadastrar Projeto, Inserir Projeto e Fechar.

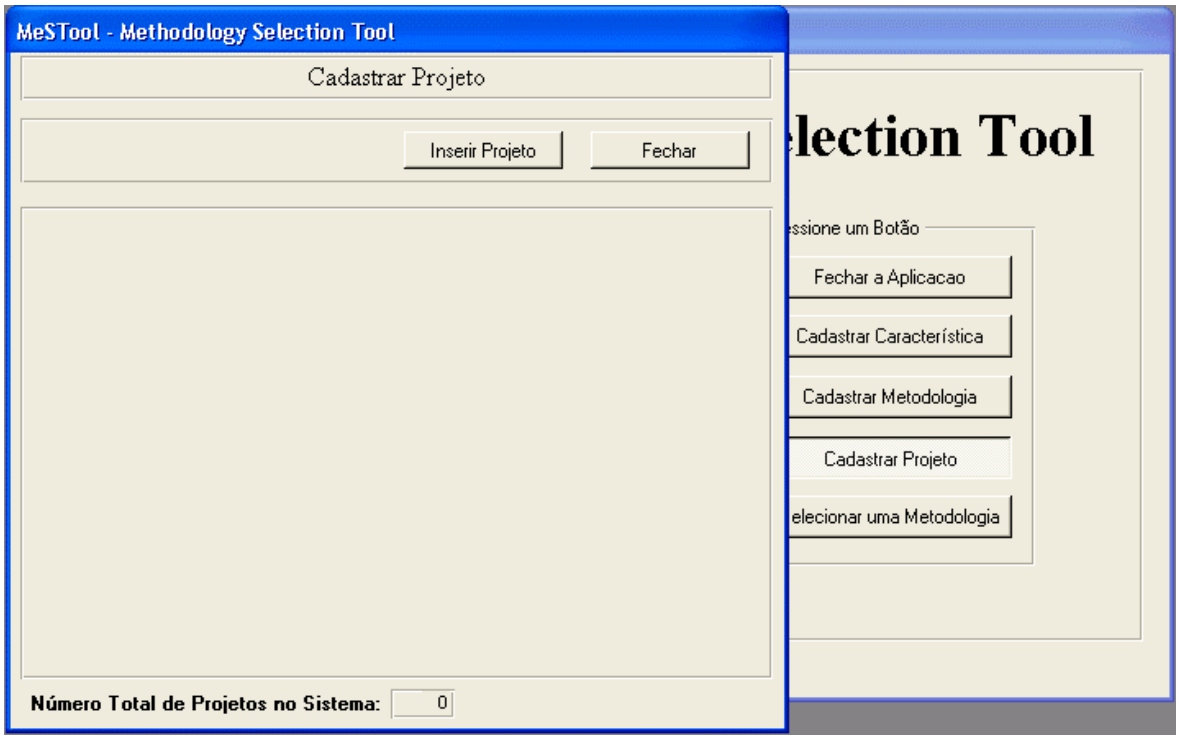


Figura 5. 15 – A Janela Inicial para Cadastrar Projeto.

Ao pressionar o botão Inserir Projeto, a figura 5.16 mostra que o perfil do projeto exige o nome da organização, o qual corresponde ao nome de uma diretoria, superintendência, departamento, divisão ou qualquer outro órgão da empresa que patrocina o projeto; o nome da pessoa para contato; o telefone de contato; etc.

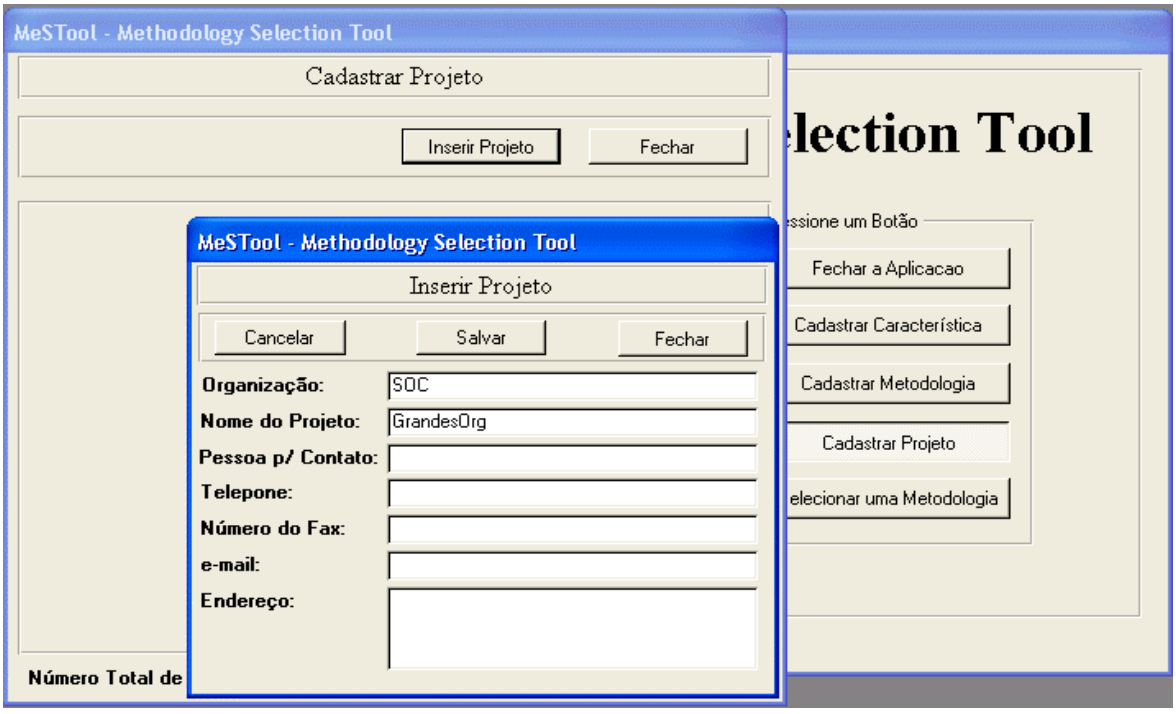


Figura 5. 16 – Cadastrando um Projeto.

Após serem inseridas as informações do perfil do projeto (figura 5.17), a janela Cadastrar Projeto disponibiliza os campos organização e projeto, os quais definem unicamente o mesmo. Pode-se então:

- Selecionar uma Característica (e definir seus valores no projeto);
- Atualizar Projeto, ou seja, atualizar o perfil do projeto.

Ao selecionar uma característica pode-se pressionar o botão Definir Valores do Projeto, para o projeto selecionado. Esta ação abre a janela, de mesmo nome, a qual mostra todos os valores da característica com o peso "0", uma vez que estes valores ainda não foram definidos como requisitos do projeto.

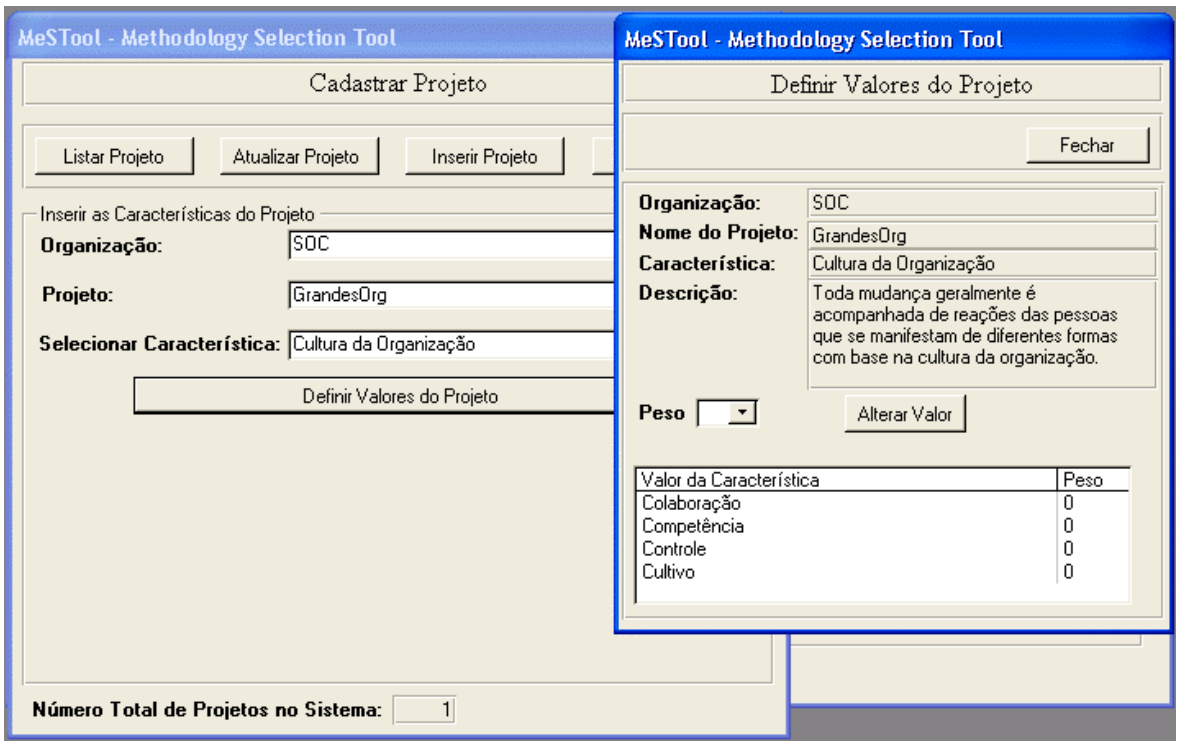


Figura 5. 17 – Definir Valores do Projeto.

Para indicar um valor como requisito do projeto (figura 5.18), efetuam-se as seguintes ações:

- Seleciona-se o valor da característica desejado;
- Escolhe-se o peso que melhor represente os interesses dos *stakeholders*.

Em seguida pressiona-se Alterar Valor.

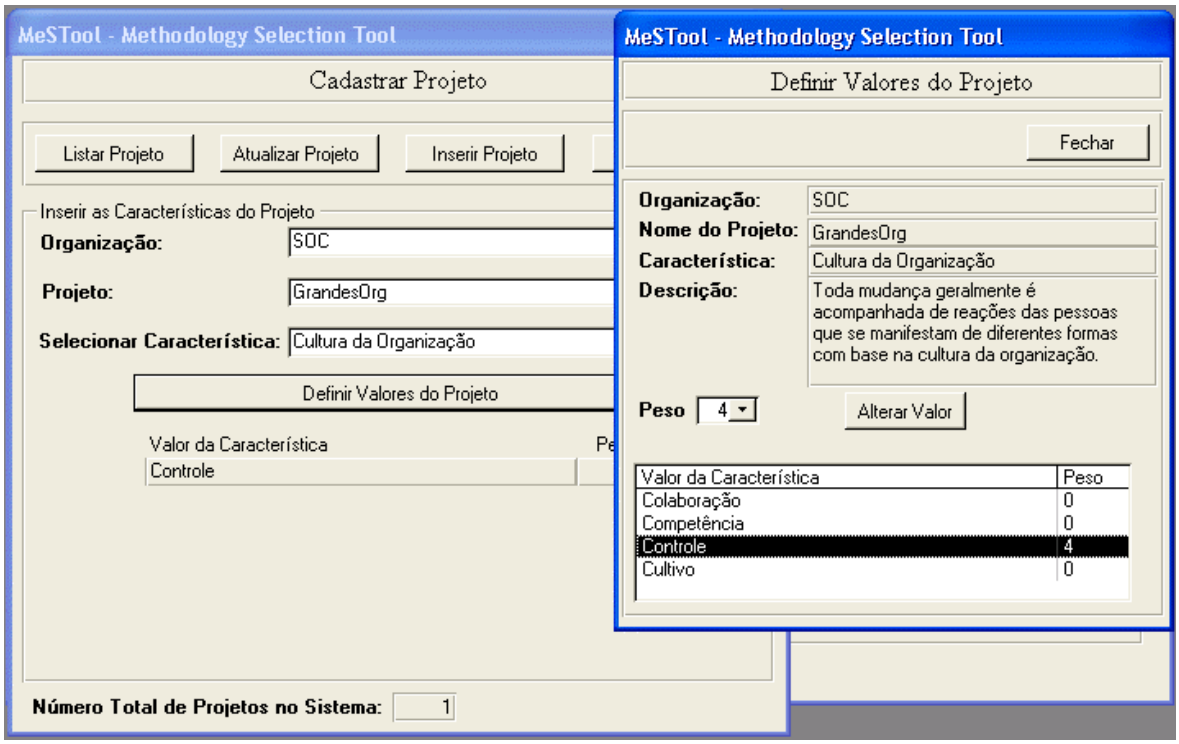


Figura 5. 18 – Definir Valores de uma Característica no Projeto.

Pressionando o botão Atualizar Projeto é possível atualizar os dados do perfil do projeto ou mesmo excluí-lo, conforme figura 5.19.

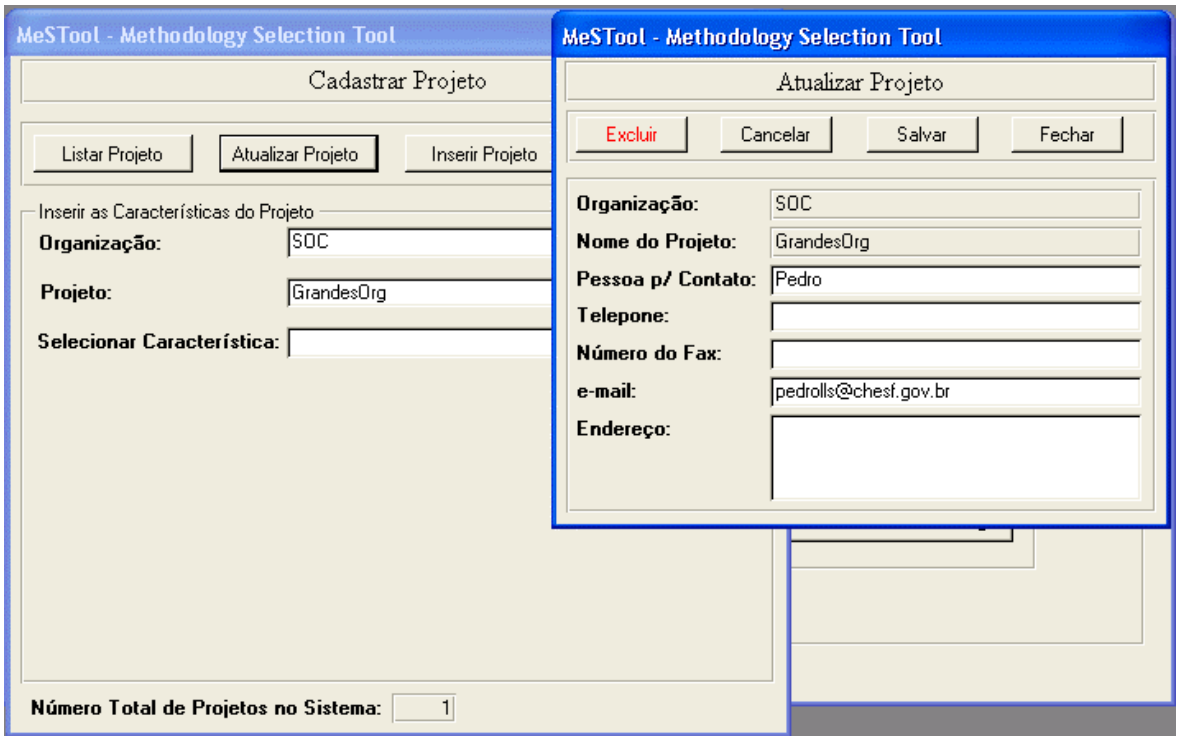


Figura 5. 19 – Atualizar Perfil do Projeto.

Uma vez que pelo menos uma característica esteja indicada no projeto, pode-se obter a lista de características do mesmo, pressionando-se o botão Listar Projeto na janela Cadastrar Projeto, cujo resultado é apresentado conforme mostra a figura 5.20. A lista apresenta, para cada organização e projeto, as características com seus valores.

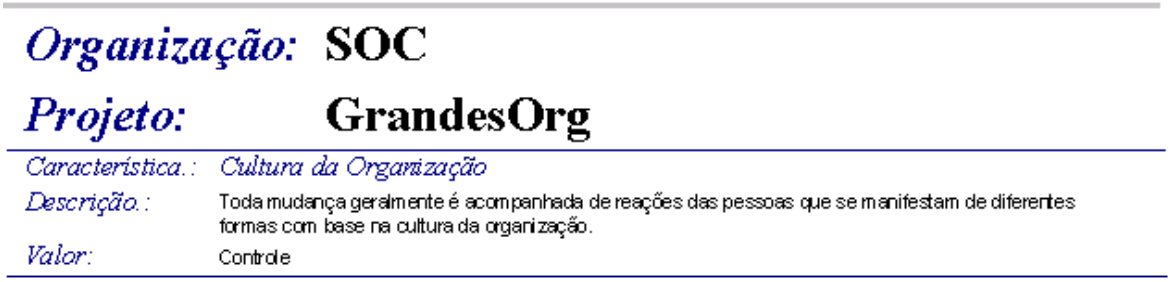


Figura 5. 20 – A Lista das Características do Projeto.

Na próxima seção, será apresentada a função Selecionar uma Metodologia.

5.2.4. Selecionar uma Metodologia

Ao ser pressionado o botão Selecionar uma Metodologia, na janela principal do MeSTool, o sistema mostra a janela, conforme a figura 5.21.

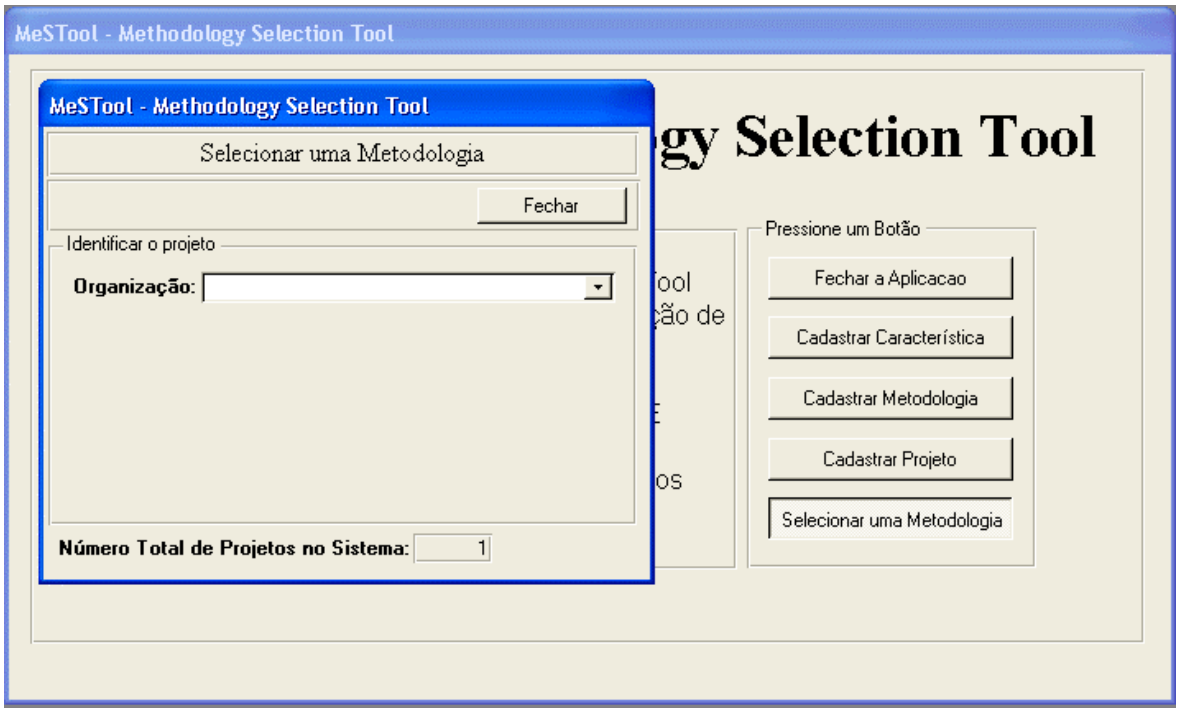


Figura 5. 21 – A Janela Selecionar uma Metodologia.

Para selecionar uma metodologia, é necessário indicar a organização e o projeto (figura 5.22) para o qual se busca a metodologia. Após indicar a organização e o projeto, o botão Selecionar uma Metodologia fica visível, permitindo que o mesmo seja pressionado.

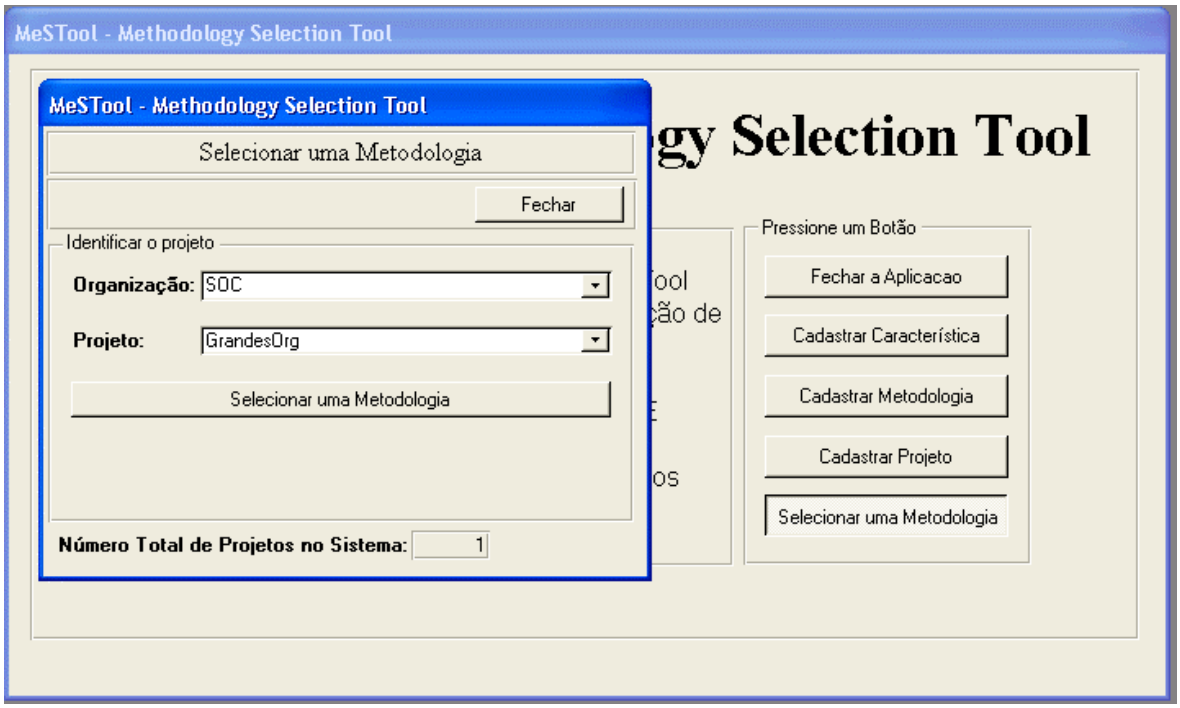


Figura 5. 22 – Selecionar uma Metodologia.

O gráfico da figura 5.23 mostra que, no momento, RUP é a única metodologia que atende os requisitos do projeto.

A janela com o Resultado das Metodologias Candidatas está dividida em cinco blocos:

- O Titulo da janela, na parte superior, logo abaixo do logotipo MeSTool;
- Os dados de identificação do projeto, abaixo e à direita do título;
- As características do Projeto, abaixo e à esquerda do título, mostram os valores das características com os respectivos pesos;
- Os Dados das Metodologias Candidatas mostram as características com seus valores relacionados às metodologias;
- O Gráfico, em forma de barra, das Metodologias Candidatas mostra, para cada metodologia o **Resultado da Seleção** = **Σ** (**Peso** x **Satisfaz**), conforme apresentado na seção 4.5.1.

À medida que outras metodologias estiverem disponíveis na base de dados, aparecerão no gráfico desde que atendam a pelo menos uma característica do projeto.

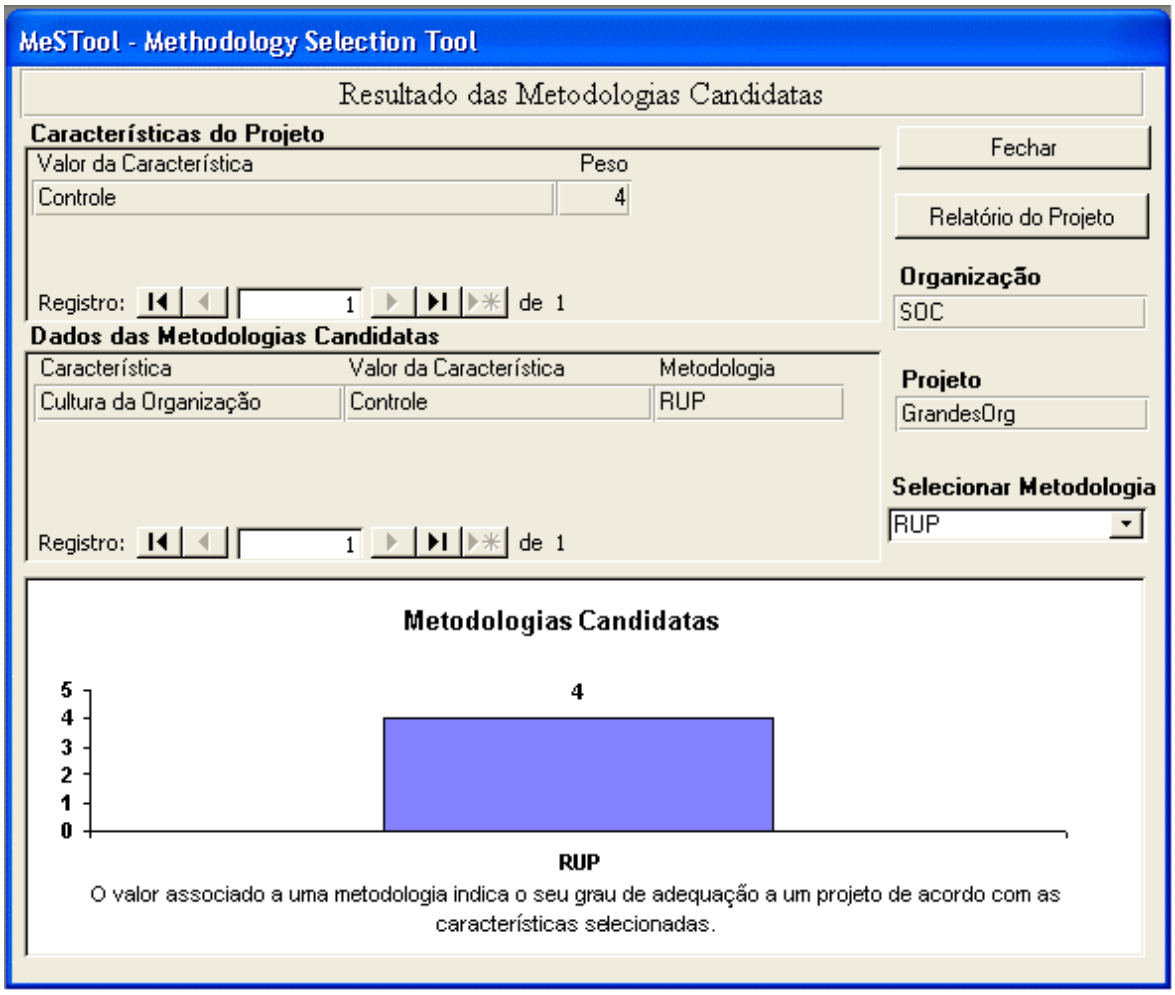


Figura 5. 23 – Metodologias Candidatas.

Quando o botão Relatório do Projeto é pressionado, o relatório com a metodologia selecionada para um projeto é apresentado na figura 5.24. Ele contém o nome da organização que financia o projeto, os dados do perfil do projeto, logo abaixo e à esquerda, e à direita estão os dados do perfil da metodologia selecionada. Na última parte do relatório vem o gráfico de barras apresentando a metodologia com seu respectivo total na parte superior da barra.

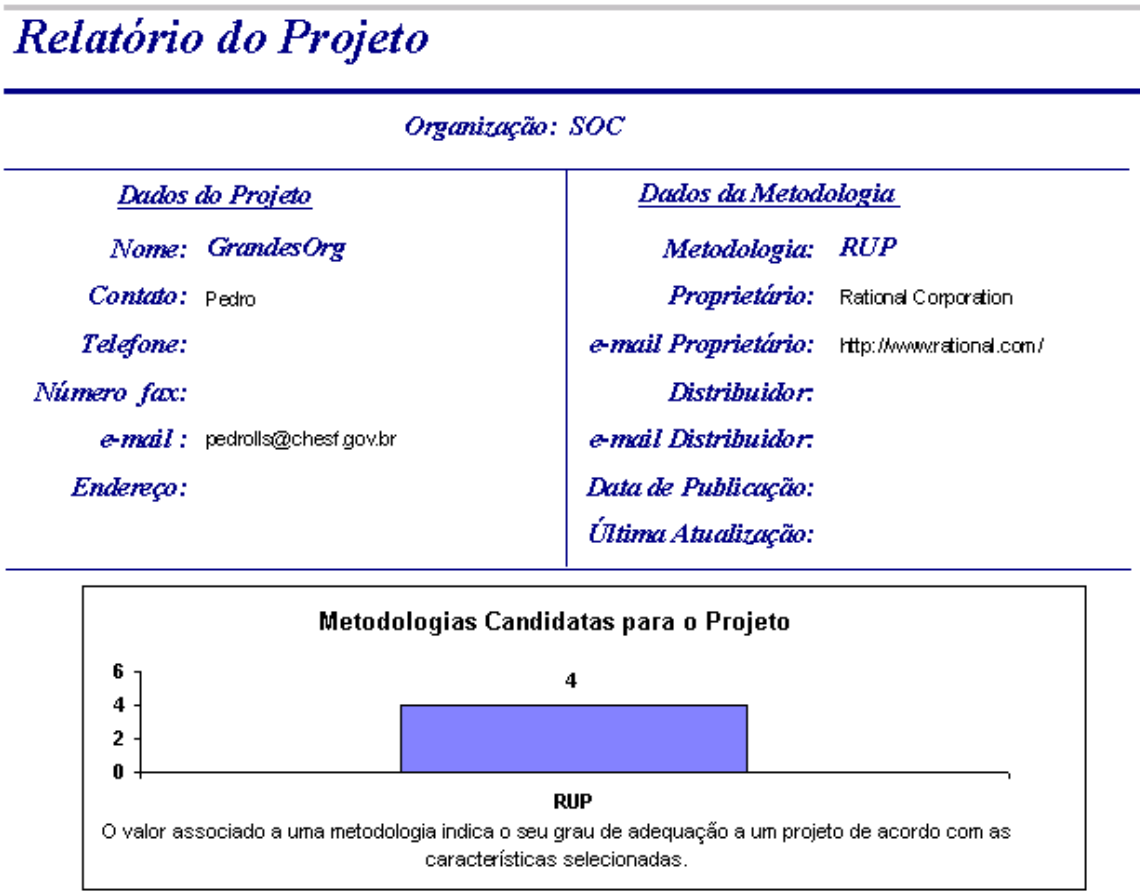


Figura 5. 24 – Relatório com a Metodologia Selecionada para um Projeto.

Tomando como base as tabela 4.2 e 4.3, que descrevem as características das metodologias RUP e XP, respectivamente, serão repetidas as ações da seção 5.2.1 para inserir novas características, bem como as da seção 5.2.2 para complementar as características da metodologia RUP e XP.

Novamente, fazendo uso da janela 5.22 e pressionando-se o botão Selecionar uma Metodologia, obtêm-se os resultados da seleção, apresentados na figura 5.25. O gráfico mostra as metodologias RUP e XP com seus respectivos totais na parte superior das barras.

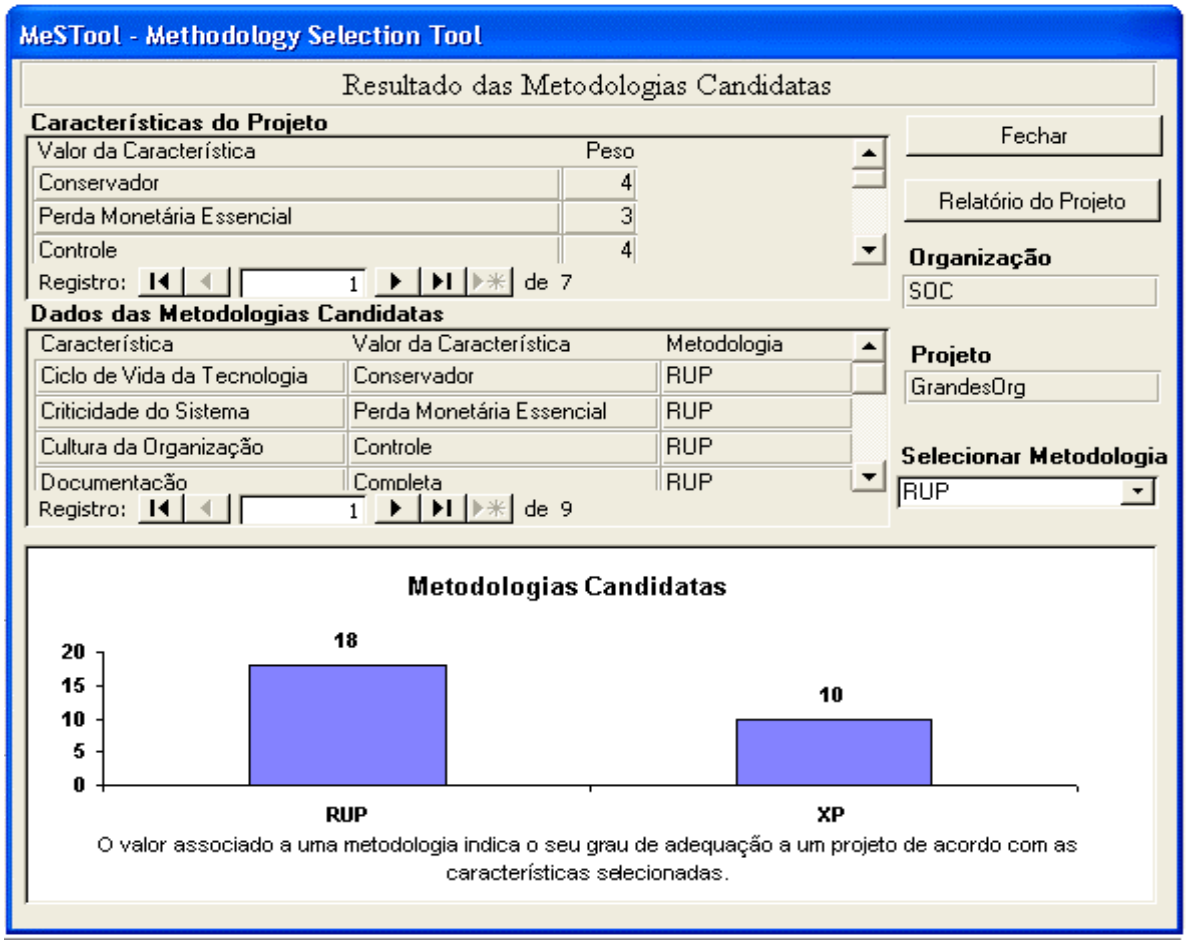


Figura 5. 25 – Resultado do Processo de Seleção de Metodologia.

Pressionando-se o botão Relatório do Projeto, na figura 5.25, o sistema mostra o relatório apresentado na figura 5.26, o qual é semelhante ao Resultado das Metodologias Candidatas. Contudo, pode ser impresso e serve como um resumo executivo.

Nota - Quando mais de uma metodologia está cadastrada na base de dados do MeSTool, toda aquela que satisfizer a pelo menos uma característica, aparecerá nos gráficos das figuras 5.25 e 5.26. Em outras palavras, uma metodologia não será descartada por não atender a todas as características. A decisão final sempre será de responsabilidade do gerente do projeto/projetista. Mesmo que ocorra um empate entre as metodologias o relatório apresentará os dados da metodologia selecionada.

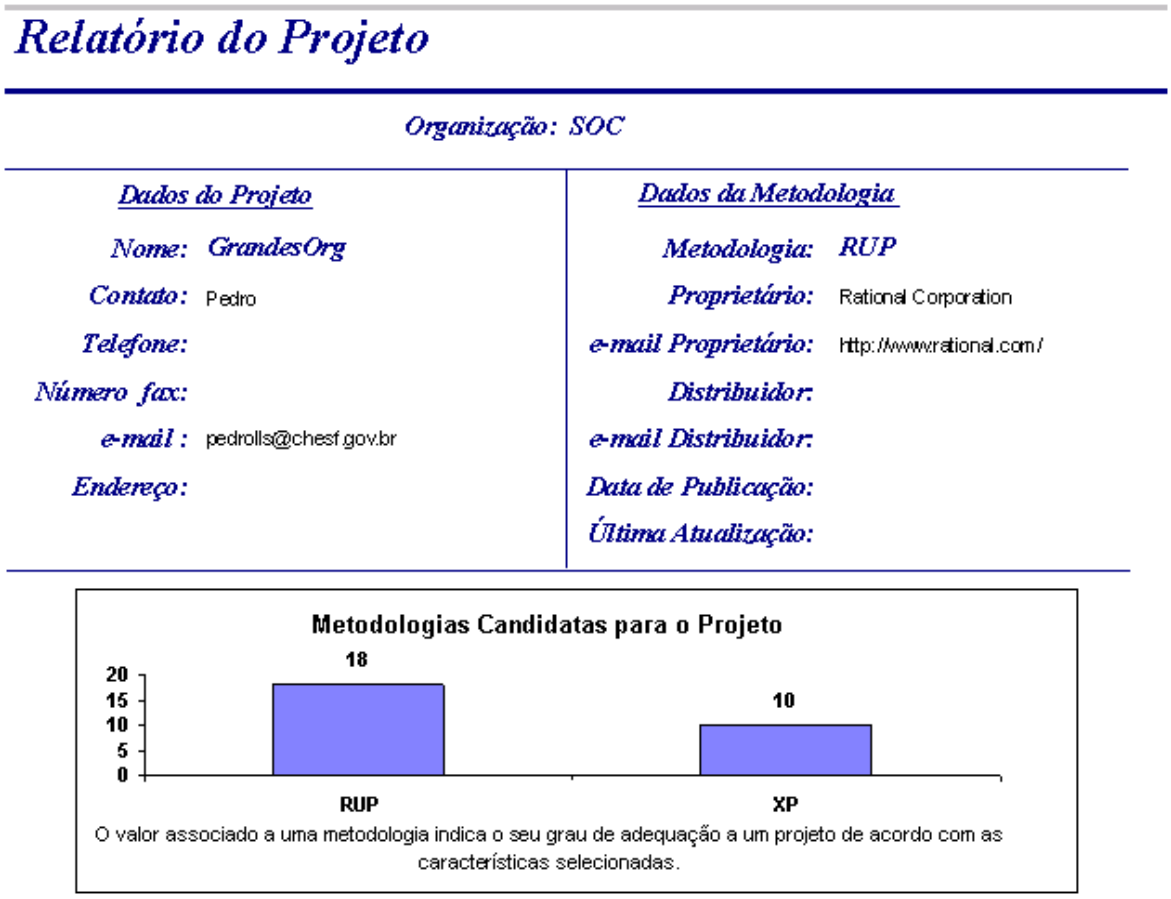


Figura 5. 26 – Relatório da Metodologia Selecionada para o Projeto.

A próxima seção apresenta a conclusão deste capítulo.

5.3. Considerações Finais

Este capítulo apresentou a ferramenta MeSTool, que auxilia o gerente de projeto a identificar uma metodologia que melhor satisfaz aos interesses dos *stakeholders.*

Basicamente o MeSTool atende a dois atores, o Gerente de Projeto/Projetista e o Metodologista abrangendo quatro funções básicas: Cadastrar Característica, Cadastrar Metodologia, Cadastrar Projeto e Selecionar uma Metodologia.

Estas funções devem ser usadas nesta ordem, pois o MeSTool funciona relacionando-se às características cadastradas na sua base de dados. Estas características são inicialmente só visíveis à função Cadastrar Metodologia. Só quando uma metodologia faz uso de uma característica, a função Cadastrar Projeto consegue “enxergar” aquela característica.

Cadastrar Projeto significa identificar os requisitos da metodologia que se busca para o projeto, os quais representam os interesses dos *stakehloders*. Estes interesses são representados por pesos atribuídos aos valores das características.

Tendo um projeto sido cadastrado no MeSTool, a função Selecionar uma Metodologia mostra as metodologias candidatas, ou seja, aquelas metodologias que satisfazem a pelo menos um dos requisitos da metodologia para o projeto. No entanto, a Seleção de uma Metodologia é uma decisão gerencial baseada nas informações apresentadas pelo MeSTool.

As limitações do MeSTool estão principalmente relacionadas à dependência de que os requisitos da metodologia já estejam definidos e as metodologias avaliadas, ou seja, o MeSTool não suporta o processo completo definido no capítulo quatro, mas apenas o processo de seleção.

Contudo, mesmo diante desta limitação, o MeSTool facilita a seleção de uma metodologia para o desenvolvimento do projeto, visto que apresenta, para qualquer metodologia cadastrada na base de dados do MeSTool, o resultado do algoritmo que considera os interesses dos *stakeholders* do projeto.

No próximo capítulo, serão apresentadas as conclusões e possibilidades de trabalhos futuros.

6. Conclusões e Trabalhos Futuros

Este capítulo apresenta as principais conclusões e as contribuições deste trabalho, e mostra trabalhos futuros que podem ser desenvolvidos baseados nesta dissertação.

6.1. Sumário do Trabalho

É necessário selecionar uma metodologia da que mais se adequa ao desenvolvimento de um *software*. A metodologia utilizada no desenvolvimento de *software* é de fundamental importância para o sucesso do produto uma vez que uma divergência entre esta e o objetivo empresarial ou entre esta e a cultura organizacional geram barreiras as quais são difíceis de transpor. Se estas barreiras não forem removidas, elas crescerão em outros projetos podendo comprometer a sobrevivência da empresa.

Cada vez mais aumentam a importância do entendimento do mercado, suas restrições, ameaças e oportunidades para a sobrevivência das organizações. Não adianta se conseguir um produto eficiente que não atenda ao mercado. Por outro lado, não adianta entender o mercado sem conseguir fornecer as respostas que o mesmo exige. Selecionar uma metodologia adequada ao *software* é um ponto de partida para o sucesso do projeto.

Este trabalho mostrou um processo de seleção de metodologias de desenvolvimento de *software*, considerando o mercado; a tecnologia; a cultura da organização; as regras de negócios, estratégias e políticas da organização; a capacitação das pessoas; a criticidade do sistema e as prioridades do projeto. Estas considerações são as bases dos requisitos da metodologia a ser selecionada para o projeto.

O MeSTool possui quatro funções básicas: Cadastrar Características, Cadastrar Metodologia, Cadastrar Projeto e Selecionar uma Metodologia. As duas primeiras formam a estrutura que possibilita o cadastramento das metodologias de desenvolvimento de *software*. As duas últimas permitem o cadastramento do projeto e a seleção da metodologia que satisfaz os interesses dos *stakeholders*.

O capítulo um, tratou das dificuldades encontradas num projeto de *software* e, apresentou uma visão geral sobre algumas metodologias de desenvolvimento. Foram abordados conceitos tais como processo de engenharia de *software* e metodologia num sentido amplo, o qual foi adotado nesta dissertação. Em seguida discutiu-se a evolução das metodologias e apresentou-se uma classificação para as mesmas em pesadas e ágeis. Foram estabelecidos o escopo e objetivos desta dissertação, discutiu-se a sua contribuição e sua estrutura de capítulos.

O capítulo dois descreveu quatro metodologias, o RUP, o OPEN, o Crystal e o XP.

O RUP é um *framework* que estabelece um processo centrado na arquitetura, baseado em componentes, e dirigido por casos de uso, através de Fluxos de Processo realizado por diversos trabalhadores.

O OPEN é um *framework* de processos cujas atividades seqüenciais e temporizadas transformam os requisitos dos usuários em *software*, utilizando um modelo de ciclo de vida dirigido por contrato.

O Crystal apresenta níveis de precisão, exatidão, relevância, tolerância e escala. É enfatizada a precisão, em todos os estágios de desenvolvimento, para o gerenciamento do desenvolvimento de *software* e para a entrega dos produtos. Crystal utiliza a seqüência de validação V-W, a qual é uma estratégia que inclui o desenvolvimento incremental e iterativo e permite ações gerenciais. Crystal usa o desenvolvimento concorrente para aumentar a velocidade de desenvolvimento.

O XP estabelece doze práticas baseadas para o seu processo de desenvolvimento e institui regras que definem os direitos do cliente e do desenvolvedor. XP é um processo rígido, pois todos os valores, princípios e práticas devem ser aplicados para se obter um desenvolvimento efetivo, no qual o cliente é um participante do projeto.

O capítulo três apresentou os problemas relacionados à seleção de uma metodologia. Foram apresentados os trabalhos correlatos tais como o Guia para Adoção de Ferramentas CASE [46]; a norma ISO/IEC-14102 [70]; o gerenciamento de projetos com foco em inovação e velocidade [30]; uma metodologia por projeto [12, 13]; Uma Seleção de Produtos de *Software* Utilizando uma Abordagem Baseada em Engenharia de Requisitos [1]; e o SA-CMM [16] que trata de processo de aquisição de *software*.

O capítulo quatro, com base nos trabalhos correlatos, apresentou o processo de seleção proposto nesta dissertação. Este processo é dividido cinco em fases. A fase de estruturação tem os objetivos de identificar os interesses dos *stakeholders*, identificar as metodologias disponíveis e definir os requisitos da metodologia. A fase de avaliação consiste em analisar as metodologias disponíveis segundo os requisitos identificados. A fase de seleção é responsável pela seleção da metodologia para o desenvolvimento do projeto. Após a seleção de uma metodologia, a mesma deve ser validada, através do desenvolvimento de um projeto piloto. Caso sejam positivos os resultados obtidos no projeto piloto, a metodologia é institucionalizada em toda a organização.

O capítulo cinco apresentou uma visão geral do MeSTool, uma ferramenta que suporta a seleção de uma metodologia, admitindo que os requisitos da metodologia já tenham

sido definidos. O MeSTool atende a dois atores, um metodologista e um gerente de projeto/projetista. O metodologista tem como objetivos Cadastrar Característica e Cadastrar Metodologia, enquanto o gerente de projeto/projetista tem os objetivos de Cadastrar Projeto e Selecionar uma Metodologia. As atividades do metodologista precedem as atividades do gerente de projeto uma vez que é necessário primeiro cadastrar características na base de dados, em seguida cadastrar as metodologias, e finalmente selecionar a metodologia mais adequada ao projeto.

Na próxima seção, será apresentada a contribuição deste trabalho.

6.2. Contribuições do Trabalho

A principal contribuição deste trabalho é o processo de seleção de uma metodologia para o desenvolvimento de um projeto de *software*, baseando-se em diversas fontes de consulta.

Outra contribuição é a construção de um protótipo de uma ferramenta (MeSTool), que automatiza parte do processo proposto, especificamente a fase de seleção, considerando que as metodologias foram avaliadas e os requisitos da metodologia que suporta o projeto foram definidos.

Na próxima seção serão apresentadas as possibilidades de trabalhos futuros baseados nesta dissertação.

6.3. Trabalhos Futuros

O MeSTool trata da seleção de metodologias de forma muito limitada, apesar de ser bastante flexível, pois permite que qualquer metodologia ou característica de metodologia seja inserida na base de dados. O MeSTool poderia gerenciar informações importantes sobre a organização, tais como a capacitação das pessoas, a tecnologia na empresa e a avaliação das mudanças de mercado.

O MeSTool poderia considerar as outras fases, por exemplo, o processo de iniciação sendo controlado pelo MeSTool facilitaria a tomada de decisão pois as informações poderiam ser obtidas de forma mais rápida; o processo de validação, permitiria um gerenciamento do projeto e do processo de desenvolvimento que ajudaria de forma mais efetiva o processo de institucionalização da metodologia na organização.

A tecnologia adotada no MeSTool ficou limitada a janelas de cadastramento e consulta, devido ao tempo disponível. Outra opção é implementar o MeSTool para a Internet,

usando a plataforma Java ou .Net, as quais permitem o acesso à aplicação através do uso de “browsers”, que são de fácil penetração nas organizações. Poderiam ser usados também recursos baseados em XML [72] para o armazenamento de dados e troca de informações entre sistemas, uma vez que em se tratando de transferência de dados, XML está se tornando o padrão da indústria. Os recursos da Internet facilitariam as trocas de informações e permitiriam que a base de dados do MeSTool fosse de acesso público, onde metodologistas pudessem cadastrar suas metodologias e receber sugestões e solicitações para atualizações, em virtude de novas características. Enquanto gerentes de projeto (ou projetistas) poderiam consultar o MeSTool para selecionar uma metodologia para seus projetos.

Apesar de acreditar-se que o processo proposto é consistente, pois se baseia em normas e padrões bem aceitos na comunidade acadêmica e industrial, é preciso avaliar a sua aplicação em um projeto de *software* real.

Referências Bibliográficas

1 Alves, Carina, “Seleção de Produtos de Software Utilizando uma Abordagem Baseada em Engenharia de Requisitos”, Dissertação de Mestrado, Centro de Informática, Universidade Federal de Pernambuco, Março 2001

2 Andriole, Steve, “New Opportunities for Business-Technology Integration”, Business-IT Strategies Advisory Service, Executive Report, Vol.5, No.5, 2002, Cutter Consortium

3 Beck, Kent, “Extreme Programming Explained: Embrace Change”, Addison-Wesley, Third printing, May 2000, ISBN 0201616416

4 Beck, Kent, Fowler, Martin, “Planning Extreme Programming”, Addison-Wesley, First printing, October 2000, ISBN 0-201-71091-9

5 Booch, Grady, “Object-Oriented Analysis and Design with Applications”, Addison-Wesley, Published 1993, ISBN 0201360462, http://www.slac.stanford.edu/~marino/html/booch/method.html

6 “Catalysis”, http://www.catalysis.org, Último acesso em 10 de Janeiro de 2002

7 Charette, Robert, “The decision is in: Agile versus Heavy Methodologies”, Vol.2, No.19, 2002, Cutter Consortium

8 “Capability Maturity Model for Software (SW-CMM)”, SEI - Software Engineering Institute, http://www.sei.cmu.edu/cmm/cmm.html, Último acesso em 27 de Setembro de 2001

9 Coad, Peter, Yourdon, Edward, “Análise Baseada em Objetos”, Segunda Edição, 1992, Editora Campus Ltda, ISBN 8570017006

10 Coad, Peter, Yourdon, Edward, “Projeto Baseado em Objetos”, 1993, Editora Campus Ltda, ISBN 857001760X

11 Cockburn, Alistair, “Surviving Object-Oriented Projects: A Manager's Guide”, Addison-Wesley, Sixth printing, November 2000, ISBN 0201498340

12 Cockburn, Alistair, “A Methodology Per Project, Humans and Technology Technical Report”, TR 99.04, Oct.1999, 7691 Dell Rd, Salt Lake City, UT 84121 USA, arc@acm.org, http://crystalmethodologies.org/articles/mpp/methodologyperproject.html, Último acesso em 31 de Agosto de 2001

13 Cockburn, Alistair, “Selecting a Project’s Methodology”, IEEE, July/August 2000

14 Cockburn, Alistair, “Encyclopedia of Work Products”, http://hometown.aol.com/humansandt/crystal/clear/w-encyclo.html, Último acesso em 26 de Outubro de 2001

15 Cockburn, Alistair, “Agile Software Development”, (Draft version, 2000), Addison-Wesley, ISBN 0201699699

16 Cooper, Jack, Fisher, Matthew, “Software Acquisition Capability Maturity Model (SA-CMM)”, Version 1.03, March 2002, Technical Report, CMU/SEI-2002-TR-010, ESC-TR-2002-010

17 “Crystal”, http://crystalmethodologies.org, Último acesso em 06 de Maio de 2002

18 “Dynamic Systems Development Method – DSDM”, http://www.dsdm.com, Último acesso em 04 de Maio de 2002

19 D´Souza, Desmond Francis, Wills, Alan Cameron, “Objects, Components, and Frameworks with Uml: The Catalysis Approach”, Second printing, December 1998, Addison-Wesley Object Technology Series, ISBN 0201310120

20 Dukic, L., “Non-Functional Requirements for COTS Software Components”, Workshop Ensuring Successful COTS Development, Los Angeles, EUA, May 1999

21 “Feature Driven Development – FDD”, http://www.togethersoft.com, Último acesso em 04 de Maio de 2002

22 “Firesmith”, http://www.donald-firesmith.com/Firesmith.html#1986, Último acesso em 06 de Março de 2003

23 Fowler, Martin, “Refactoring: Improving the Design of Existing Code”, Addison-Wesley, Fifth printing, September 2000, ISBN 0201485672

24 Fowler, Martin, Highsmith, Jim, “The Agile Manifesto”, http://www.sdmagazine.com/documents/s=844/sdm0108a/0108a.htm, Último acesso em 31 de Outubro de 2001

25 Fowler, Martin, Foemmel, Matthew, “Continuous Integration”, http://www.thoughtworks.com/library/index.html, Último acesso em 04 de Maio de 2002

26 Graham, Ian, Henderson-Sellers, Brian, Younessi, Houman, “The OPEN Process Specification”, Addison-Wesley, First printed 1997, ISBN 0201331330

27 Henderson-Sellers, Brian, Simons, Anthony, Younessi, Houman, “The OPEN Toolbox of Techniques”, Addison-Wesley, First printed 1998, ISBN 0201331349

28 Henderson-Sellers, Brian, Unhelkar, Bhuvan, “Open Modeling with Uml”, Addison-Wesley, First printed 2000, ISBN 0201675129

29 Heuser, Carlos Alberto, “Projeto de Banco de Dados”, Instituto de Informática da UFRGS, Sagra Luzzatto, Primeira edição, 1998, ISBN 85.241.0590-9

30 Highsmith, Jim, “E-Project management: Harnessing Innovation and Speed”, Software Management, Volume I, No.1, 2001, Cutter Consortium, http://www.cutter.com/consortium/advisory_epm.html, Último acesso em 20 de Novembro de 2001

31 “ISO-9000”, http://www.iso.ch/iso/en/ISOOnline.frontpage

32 “ISO/ICE-15504”, ISO/IEC JTC 1/CS 7/WG 10, ISO/IEC TR 15504:1998(E), Draft-Version, “Information technology - Software process assessment”

33 Jacobson, Ivar, Booch, Grady, Rumbaugh, James, “The Unified Software Development Process”, Addison-Wesley, Second printing, April 1999, ISBN 0-201-57169-2

34 Jeffries, Ron, Anderson, Ann, Hendrickson, Chet, “Extreme Programming Installed”, Addison-Wesley, Second printing, December 2000, ISBN 0-201-70842-6

35 Johnson, James H., “Micro Projects Cause Constant Change”, 2001, The Standish Group International, Inc., 198 Old Townhouse Road, West Yarmouth, MA 02673

36 Korth, Henry F., “Sistema de Banco de Dados”, São Paulo, Makron Books, 1995. ISBN 8534610738

37 Kotonya, G. and Sommerville, I., “A Frame work for Integratin Functional and Non-Functional Requirements”, International Workshop on Systems Engineering for Real Time Applications, UK, 1993, pp. 148-153

38 Kruchten, Philippe, “The Rational Unified Process – An Introduction”, Addison-Wesley, 1998 – ISBN 0-201-60459-0

39 Kruchten, Philippe, “What Is the Rational Unified Process?”, http://www.therationaledge.com/content/jan_01/f_rup_pk.html, Último acesso em 10 de Abril de 2002

40 Kruchten, Philippe, http://www.cutter.com/itjournal/itj0112e.html, Último acesso em 13 de Maio de 2002

41 Martin, Robert C., “Continuous Care Vs. Initial Design”, http://www.objectmentor.com/resources/articleIndex, Último acesso em 13 de Fevereiro de 2002

42 “Process MeNtOR”, http://www.processmentor.com, Último acesso em 07 de Maio de 2002

43 “Microsoft”, http://www.microsoft.com/ms.htm, Último acesso em 07 de Maio de 2002

44 Moore, Geofrey, “Inside the Tornado: Marketing Strategies from Silicon Valley's Cutting Edge”, HapperCollins, 1999, ISBN 0887308244

45 “MOSES”, http://panoramix.univ-paris1.fr/CRINFO/dmrg/OODOC/oodoc/oo-14.html, Último acesso em 06 de Março de 2003

46 Oakes, Kimberly Stepien, Smith, Dennis, Morris, Ed., “Guide to CASE Adoption”, Technical Report CMU/SEI-92-TR-15, ESC-TR-92-015, November 1992

47 “OPEN”, http://www.open.org.au/Introduction/main.html, Último acesso em 07 de Maio de 2002

48 Orr, Ken, “CMM versus Agile Development: Religious Wars and Software Development Development”, Agile Project Management Advisory Service, Executive Report, Vol.3, No.7, Cutter Consortium

49 Orr, Ken, “Managing Technology Decisionmaking”, Business-IT Strategies Advisory Service, Executive Report, Vol. 5, No. 9, Cutter Consortium

50 Osterweil, L., “Software Process is Software Too”, In Proc. ICSE 9, 9th International Conference on Software Engineering, Monterey, CA, USA, March 1987, IEEE Computer Society Press

51 “Rational Software”, http://www.rational.com, Último acesso em 08 de Janeiro de 2002

52 “Responsibility Based Modeling”, http://members.aol.com/humansandt/techniques/responsibility.htm, Último acesso em 20 de Janeiro de 2003

53 Robertson, Suzanne, Robertson, James, “Mastering the Requirements Process” Addison-Wesley, Published 1999, ISBN 0201360462

54 Rumbaugh, J. et al, “Object-Oriented Modeling and Design”, Prentice-Hall, 1991, Objectory, http://www.iconixsw.com/Jacobson.html, Último acesso em 06 de Março de 2003

55 Rumbaugh, James, http://www.rational.com/products/whitepapers/396.jsp#p5, Último acesso em 20 de Janeiro de 2003

56 “RUP”, http://www.rational.com, Último acesso em 06 de Maio de 2002

57 Salviano, Clenio F., “Um Método para escolha dos Processos para uma Melhoria Alinhada aos Objetivos de Negócio”, Workshop de Qualidade de Software, 2001, Instituto Nacional de Tecnologia da Informação – ITI, Campinas – São Paulo

58 "Software Engineering Institute – SEI", http://www.sei.cmu.edu, Último acesso em 27 de Setembro de 2001

59 "SOMA", http://ourworld.compuserve.com/homepages/grahami/SOMA.htm, Último acesso em 06 de Março de 2003

60 "SCRUM", http://www.controlchaos.com, Último acesso em 04 de Maio de 2002

61 "Sun Microsystems Inc.", http://www.sun.com , http://java.sun.com , Último acesso em 04 de Abril de 2002

62 The Standish Group International, Inc., "The Chaos Report (1994)", 1995, http://www.standishgroup.com/sample_research/chaos_1994_4.php, Último acesso em 06 de Maio de 2002

63 The Standish Group International, Inc., "A Recipe for Success", 1999, http://www.standishgroup.com/sample_research/PDFpages/chaos1998.pdf, Último acesso em 06 de Maio de 2002

64 Thomsett, Rob, "Extreme Project Management", Business-IT Strategies Advisory Service, Executive Report, Vol.2, No.2, 2001, Cutter Consortium

65 Wagner, Larry, "Extreme Requirements Engineering", The Great Methodologies Debate: Part 1, Cutter Consortium, IT Journal, December 2001, http://www.cutter.com/itjournal/itj0112f.html, Último acesso em 13 de Maio de 2002

66 Wake, William C., http://users.vnet.net/wwake, Último acesso em 08 de Maio de 2002

67 Webster, Steve, "On the evolution of OO methods", Department of Computing, Bournemouth University, Tablot Campus, Fern Barrow, Poole, Dorset, BH12 5BB U.K., http://dec.bournemouth.ac.uk/staff/swebster/OO_meth_evol_complete.html, Último acesso em 29 de Janeiro de 2003

68 West, David, "Planning a Project with the Rational Unified Process", The Rational Software White paper TP151, 08/02, http://www.rational.com/products/whitepapers/453.jsp?SMSESSION=NO, Último acesso em 17 de Janeiro de 2003

69 Wirfs-Brock, Rebecca, Wilkerson, Brian and Wiener, Lauren, "Designing Object-Oriented Software", Prentice Hall PTR, 1990, ISBN 0136298257

70 Wollman, Thomas, "Information technology – Guideline for the evaluation and selection for CASE tools", ISO/IEC-14102, JTC1/SC7/WG4, Project 7.25, 1995

71 "World Wide Web Consortium", http://www.w3.org, Último acesso em 03 de Maio de 2002

72 "Extensible Markup Language", http://www.w3.org/XML, Último acesso em 04 de Maio de 2002

73 "XP", http://xprogramming.com, Último acesso em 07 de Maio de 2002

74 "XP", http://www.extremeprogramming.org, Último acesso em 08 de Maio de 2002

Anexos

1. Os Casos de Uso do MeSTool

O MeSTool admite dois papéis para o usuário. Em um papel, o usuário comporta-se como um metodologista e no outro como um gerente de projeto/projetista. A seguir são descritos os casos de uso, conforme mostra a figura A.1 e classificados com as funções do MeSTool: Cadastrar Características; Cadastrar Metodologia; Cadastrar Projeto; e Selecionar uma Metodologia.

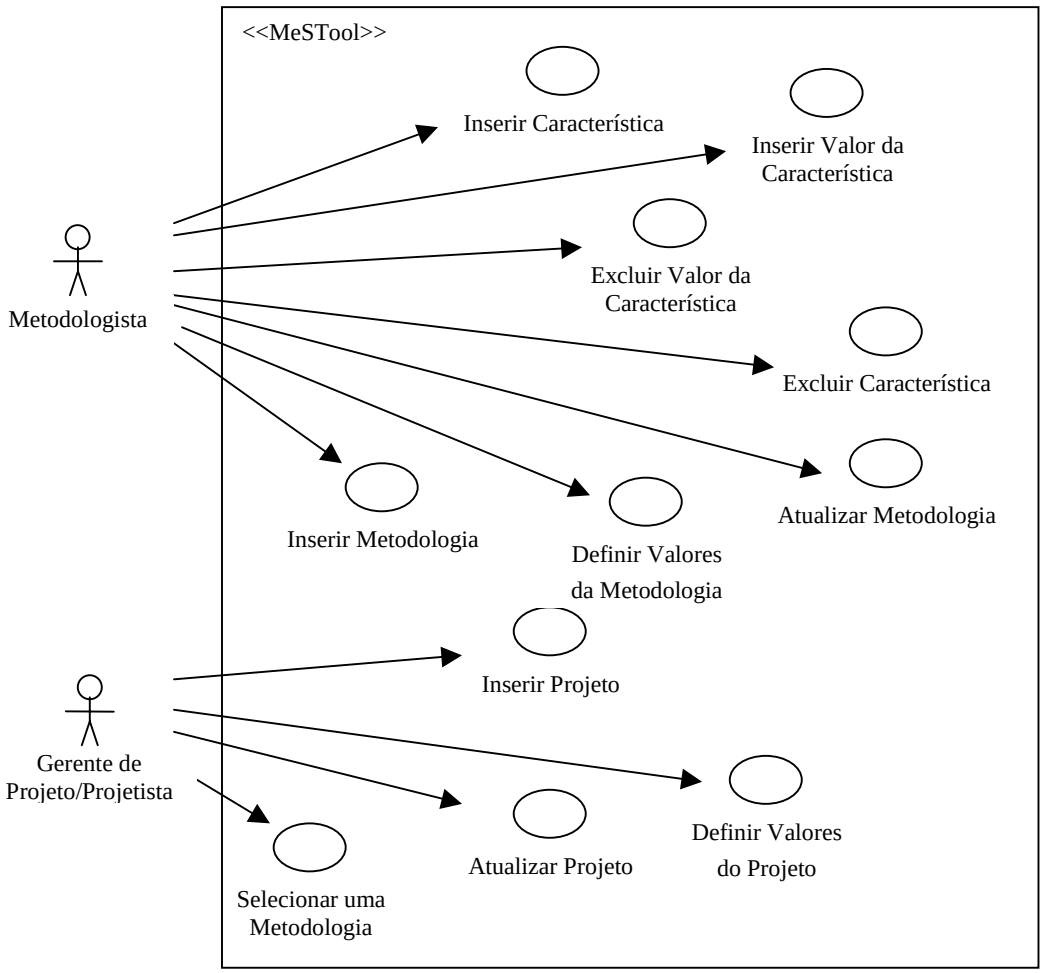


Figura A. 1– Os Casos de Uso.

1.1. Cadastrar Característica

São dois os casos de uso, relacionados à função cadastrar característica: Inserir Característica e Inserir Valor da Característica.

1.1.1. Caso de Uso: Inserir Característica

Iniciado por (Ator):
Metodologista

Descrição:
O metodologista registra as informações inerentes ao perfil da característica no banco de dados.

Entrada:
Informações inerentes ao perfil da característica.

Pré-Condições:
O perfil da característica a ser registrada no MeSTool não é do conhecimento do mesmo.

Saída:
Registro do perfil da característica na base de dados do MeSTool.

Pós-Condição:
O perfil da característica fica apto a receber valores.

Fluxo Principal de Eventos:
Na efetuação do registro do perfil de uma característica, as informações deste perfil de característica tornam-se disponíveis para receber valores.

Fluxo Excepcional de Eventos:
Se o perfil da característica a ser registrada for de conhecimento do MeSTool, este deve informar ao usuário de tal ocorrência.

1.1.2. Caso de Uso: Inserir Valor da Característica

Iniciado por (Ator):
Metodologista

Descrição:
O metodologista registra o valor da característica no banco de dados MeSTool.

Entrada:

Informações inerentes ao valor da característica.

Pré-Condições:

O perfil da característica é do conhecimento do MeSTool.

Saída:

Registro do valor da característica na base de dados do MeSTool.

Pós-Condição:

O perfil da característica com o seu valor ficam disponíveis para as metodologias.

Fluxo Principal de Eventos:

Na efetuação do valor de uma característica, a informação deste valor de característica torna-se disponível para as metodologias.

Fluxo Excepcional de Eventos:

Se o valor da característica a ser registrada for de conhecimento do MeSTool, este deve informar ao usuário de tal ocorrência.

1.1.3. Caso de Uso: Excluir Característica

Iniciado por (Ator):

Metodologista

Descrição:

O metodologista elimina as informações inerentes à característica no banco de dados.

Entrada:

Nenhuma.

Pré-Condições:

A característica encontra-se registrada no MeSTool.

Saída:

A característica é eliminada da base de dados do MeSTool.

Todos os registros nas tabelas relacionadas a esta característica são eliminados da base de dados.

Pós-Condição:

A característica fica desconhecida no MeSTool.

Fluxo Principal de Eventos:

Na efetuação da eliminação de uma característica, as informações desta característica desaparecem da base de dados.

Fluxo Excepcional de Eventos:

Nenhum.

1.1.4. Caso de Uso: Excluir Valor da Característica

Iniciado por (Ator):

Metodologista

Descrição:

O metodologista elimina o valor da característica no banco de dados MeSTool.

Entrada:

Nenhuma.

Pré-Condições:

O valor da característica é do conhecimento do MeSTool.

Saída:

O valor da característica é eliminado da base de dados do MeSTool.

Pós-Condição:

O MeSTool desconhece o valor da característica.

Fluxo Principal de Eventos:

Na efetuação da eliminação do valor de uma característica, a informação deste valor de característica torna-se indisponível para as metodologias.

Fluxo Excepcional de Eventos:

Nenhum.

1.2. Cadastrar Metodologia

São três os casos de uso, relacionados à função cadastrar metodologia: Inserir Perfil da Metodologia, Definir Valores da Metodologia e Atualizar Perfil da Metodologia.

1.2.1. Caso de Uso: Inserir Metodologia

Iniciado por (Ator):

Metodologista.

Descrição:

O metodologista registra as informações inerentes ao perfil da metodologia no banco de dados.

Entrada:

Informações inerentes ao perfil da metodologia.

Pré-Condições:

O perfil da metodologia a ser registrada no MeSTool não é do conhecimento do mesmo.

Saída:

Registro do perfil da metodologia na base de dados do MeSTool.

Pós-Condição:

O perfil da metodologia fica apto a receber dados.

Fluxo Principal de Eventos:

Na efetuação do registro do perfil de uma metodologia, estas informações tornam-se disponíveis para os projetos.

Fluxo Excepcional de Eventos:

Se o perfil da metodologia a ser registrada for de conhecimento do MeSTool, este deve informar ao usuário de tal ocorrência.

1.2.2. Caso de Uso: Definir Valores da Metodologia

Iniciado por (Ator):

Metodologista.

Descrição:

O metodologista registra as informações inerentes aos valores da metodologia, para determinada característica, na base de dados do MeSTool.

Entrada:

Informações inerentes aos valores da metodologia para uma determinada característica.

Pré-Condições:

O perfil da metodologia e os valores das características são do conhecimento da base de dados do MeSTool.

Saída:

Registro do valor da metodologia, para uma determinada característica, na base de dados do MeSTool

Pós-Condição:

O valor da metodologia fica disponível para os projetos.

Fluxo Principal de Eventos:

Na efetuação do valor de uma determinada característica de uma metodologia, este valor de metodologia torna-se disponível para os projetos.

Fluxo Excepcional de Eventos:

A janela é fechada e nenhuma alteração é efetuada no valor da metodologia.

1.2.3. Caso de Uso: Atualizar Metodologia

Iniciado por (Ator):

Metodologista.

Descrição:

O metodologista altera as informações inerentes ao perfil da metodologia no banco de dados.

Entrada:

Informações inerentes ao perfil da metodologia.

Pré-Condições:

O perfil da metodologia a ser alterada no MeSTool é do conhecimento do mesmo.

Saída:

Registro do perfil da metodologia, devidamente modificado, na base de dados do MeSTool.

Pós-Condição:

O perfil da metodologia continua apto a receber dados.

Fluxo Principal de Eventos:

Na efetuação de modificação do registro do perfil de uma metodologia, estas informações permanecem disponíveis para os projetos.

Fluxo Excepcional de Eventos:

O metodologista cancela a ação de atualização. Todos os dados permanecem inalterados na base de dados do MeSTool.

1.3. Cadastrar Projeto

São três os casos de uso, relacionados ao objetivo cadastrar projeto: Inserir Projeto, Definir Valores do Projeto e Atualizar Projeto.

1.3.1. Caso de Uso: Inserir Projeto

Iniciado por (Ator):
Gerente de Projeto/Projetista.

Descrição:
O gerente de projeto/projetista registra as informações inerentes ao perfil do projeto no banco de dados MeSTool.

Entrada:
Informações inerentes ao perfil do projeto.

Pré-Condições:
O perfil do projeto a ser registrado no MeSTool não é do conhecimento do mesmo.

Saída:
Registro do perfil do projeto na base de dados do MeSTool.

Pós-Condição:
O perfil do projeto fica apto a receber dados.

Fluxo Principal de Eventos:
Na efetuação do registro do perfil de um projeto, estas informações tornam-se disponíveis para selecionar uma metodologia.

Fluxo Excepcional de Eventos:
Se o perfil do projeto a ser registrado for de conhecimento do MeSTool, este deve informar ao usuário de tal ocorrência.

1.3.2. Caso de Uso: Definir Valores do Projeto

Iniciado por (Ator):
Gerente de Projeto/Projetista

Descrição:
O gerente de projeto/projetista registra as informações inerentes aos valores do projeto, para determinada característica, na base de dados do MeSTool.

Entrada:

Informações inerentes aos valores do projeto para uma determinada característica.

Pré-Condições:

O perfil do projeto e da metodologia bem como os valores das características são do conhecimento da base de dados do MeSTool.

Saída:

Registro do valor do projeto, para uma determinada característica, na base de dados do MeSTool.

Pós-Condição:

O valor do projeto fica disponível para selecionar uma metodologia.

Fluxo Principal de Eventos:

Na efetuação do valor de uma determinada característica de um projeto, este valor de projeto torna-se disponível para a seleção de uma metodologia.

Fluxo Excepcional de Eventos:

A janela é fechada e nenhuma alteração é efetuada no valor do projeto.

1.3.3. Caso de Uso: Atualizar Projeto

Iniciado por (Ator):

Gerente de Projeto/Projetista

Descrição:

O gerente de projeto/projetista altera as informações inerentes ao perfil do projeto no banco de dados.

Entrada:

Informações inerentes ao perfil do projeto.

Pré-Condições:

O perfil do projeto a ser alterado no MeSTool é do conhecimento do mesmo.

Saída:

Registro do perfil do projeto, devidamente modificado, na base de dados do MeSTool.

Pós-Condição:

O perfil do projeto continua apto a receber dados.

Fluxo Principal de Eventos:

Na efetuação de modificação do registro do perfil de um projeto, estas informações permanecem disponíveis na base de dados do MeSTool.

Fluxo Excepcional de Eventos:

O gerente de projeto cancela a ação de atualização. Todos os dados permanecem inalterados na base de dados do MeSTool.

1.4. Selecionar uma Metodologia

Esta função corresponde a apenas ao caso de uso Selecionar uma Metodologia

1.4.1. Caso de Uso: Selecionar uma Metodologia

Iniciado por (Ator):
Gerente de Projeto/Projetista.

Descrição:

O gerente de projeto/projetista analisa as informações sobre as metodologias candidatas e seleciona a que considerar mais conveniente para o projeto.

Entrada:

Metodologia escolhida.

Pré-Condições:

O perfil do projeto e da metodologia bem como os valores das características são do conhecimento da base de dados do MeSTool.

Saída:

Registro, na base de dados do MeSTool, da metodologia escolhida para o projeto.

Pós-Condição:

A metodologia selecionada para o projeto fica registrada na base de dados do MeSTool.

Fluxo Principal de Eventos:

Na efetuação do registro de uma metodologia para o projeto, este valor torna-se disponível para visualização.

Fluxo Excepcional de Eventos:

A janela é fechada e nenhuma alteração é efetuada relacionada à seleção de uma metodologia candidata para o projeto.

Na próxima seção é apresentado o modelo de dados.

2. O Modelo da Base de Dados

Um modelo conceitual e um modelo lógico formam o modelo de dados. Segundo Heuser [29] um modelo conceitual descreve o modelo de dado de um modo independente da implementação em um Sistema de Gerência de Banco de Dados (SGBD), e um modelo lógico descreve o modelo de dados de forma dependente do SGBD.
O modelo conceitual da base de dados do MeSTool é formado pelos relacionamentos entre seis tabelas principais e uma tabela auxiliar conforme mostrado na figura A.2.

A linha espessa da figura A.2 mostra que o relacionamento entre as tabelas characteristicsValue e methodology está selecionado. Um duplo clique neste relacionamento abre a janela editar relacionamentos conforme mostrado na figura A.3. É importante observar que como o grupo Impor integridade referencial está selecionado, as duas caixas de seleção contidas no mesmo ficam disponíveis, ou seja, é possível escolher Propagar atualização dos campos relacionados e Propagar exclusão dos registros relacionados. Em ambos os casos a integridade referencial da base de dados está garantida.

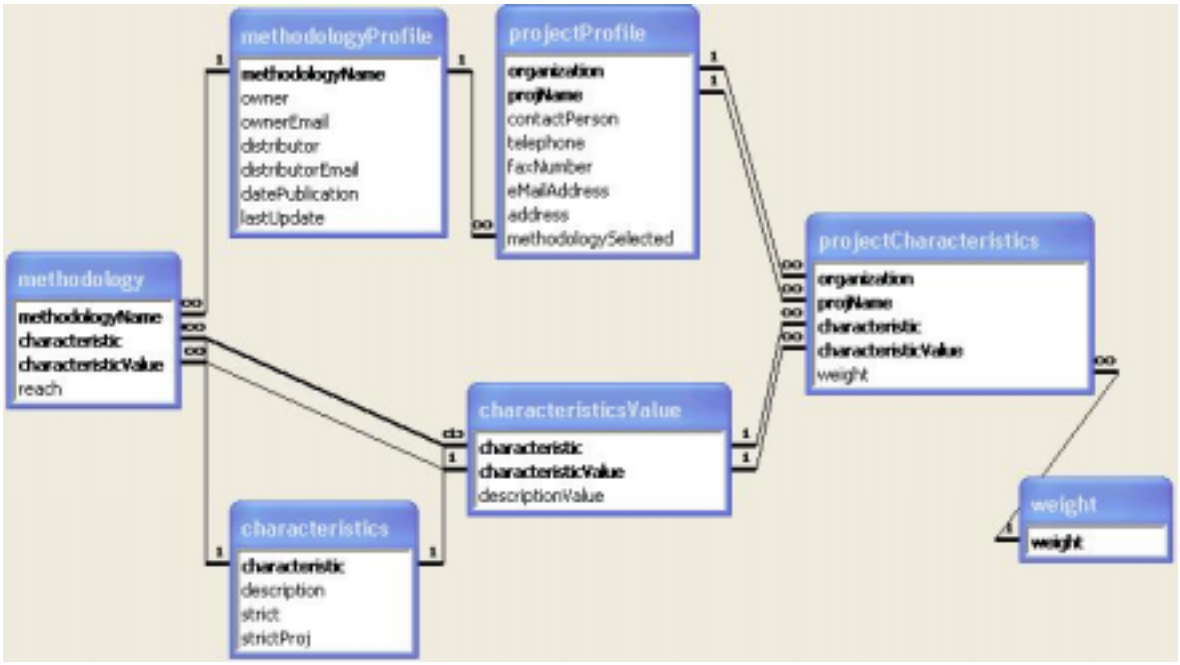


Figura A. 2 – O Modelo de Dados Conceitual do MeSTool.

Figura A. 3 – A Janela Editar Relacionamentos.

No caso da figura A.3, a propagação da exclusão dos registros relacionados não está selecionada. Isto indica que se for feita uma tentativa de excluir um registro da tabela characteristicsValue que esteja em uso pela tabela methodology, a base de dados do MeSTool responderá com a mensagem da figura A.4, garantindo a integridade referencial.

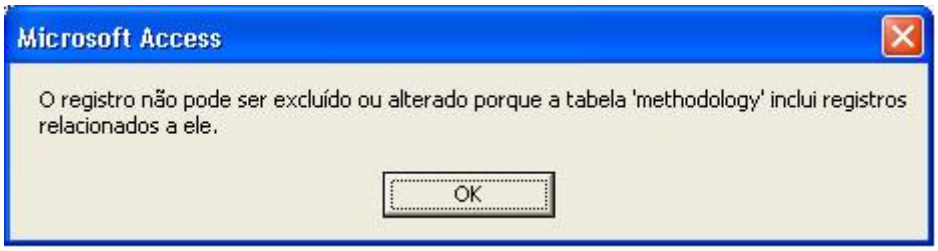


Figura A. 4 – A Integridade Referencial para um Registro.

Um duplo clique no relacionamento entre as tabelas characteristics e methodology abre a janela editar o relacionamento correspondente. Se for feita tentativa de excluir um registro da tabela characteristics que esteja em uso pela tabela methodology a base de dados do MeSTool responderá com a mensagem da figura A.5, em virtude da seleção do campo Propagar exclusão dos registros relacionados, garantindo a integridade referencial dos dados.

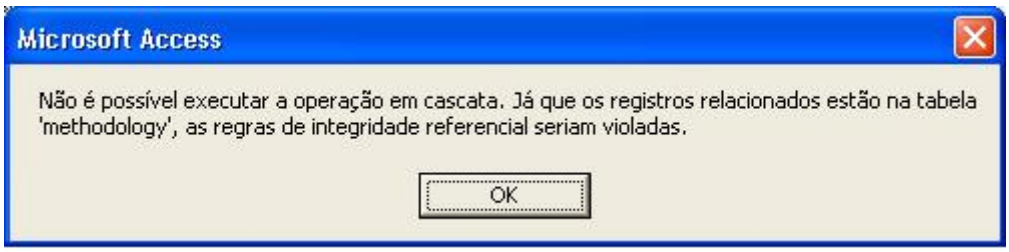


Figura A. 5 – A Integridade Referencial para a Propagação de Exclusão de Registros.

Segundo Korth [36] o modelo lógico pode ser representado por um esquema em forma de tabela. O esquema, representado em forma de tabela deve informar seus atributos, sua chave primária e, no caso de chaves estrangeiras é indicado em quais tabelas essa chave é primária. A tabela A.1 define os nomes das colunas para representar as tabelas dos esquemas.

Nome da Coluna	Significado	Valores Possíveis
Nome	Nome do Atributo	Texto
Descrição	Descrição do Atributo	Texto
Tipo	Tipo do Atributo	N (Numérico), T (Texto), M (Memorando)
Tamanho	Tamanho do Atributo	Medido em bytes
CP	Chave Primária	S (sim) ou N (não)
CE	Chave Secundária (Estrangeira)	S (sim) ou N (não)
Referência	Tabela onde o atributo é chave primária	Qualquer uma das tabelas, mencionada no esquema, a seguir.

Tabela A. 1 – Descrição das Colunas das Tabelas dos Esquemas.

Para cada entidade do modelo conceitual, segue abaixo a sua tabela correspondente.

Characteristics

Nome	Descrição	Tipo	Tamanho	CP	CE	Referência
Characteristics	Nome da característica	T	50	S	N	
Description	Descrição da característica	M	Sem limite	N	N	
Strict	Indica se permitem múltiplos valores	T	3	N	N	

CharacteristicsValue

Nome	Descrição	Tipo	Tamanho	CP	CE	Referência
Characteristics	Nome da característica	T	50	S	S	Characteristics
CharacteristicsValues	Nome do valor da característica	T	50	S	N	
DescriptionValue	Descrição do valor da característica	T	Sem limite	N	N	

MethodologyProfile

Nome	Descrição	Tipo	Tamanho	CP	CE	Referência
MethodologyName	Nome da metodologia	T	50	S	N	
Owner	Nome do proprietário da metodologia	T	50	N	N	
OwnerEmail	e-mail do proprietário da metodologia	T	50	N	N	
Distributor	Nome do distribuidor da metodologia	T	50	N	N	
DistributorEmail	e-mail do distribuidor da metodologia	T	50	N	N	
DatePublication	data de publicação da metodologia	T	50	N	N	
LastUpdate	Última data de atualização	T	50	N	N	

Methodology

Nome	Descrição	Tipo	Tamanho	CP	CE	Referência
MethodologyName	Nome da metodologia	T	50	S	S	MethodologyProfile
Characteristics	Nome da característica	T	50	S	S	Characteristics
CharacteristicsValues	Nome do valor da característica	T	50	S	S	CharacteristicsValue
Reach	Indica se a metodologia satisfaz o valor da característica	T	3	N	N	

Weight[1]

Nome	Descrição	Tipo	Tamanho	CP	CE	Referência
Weight	O peso	N	8	S	N	

ProjectProfile

Nome	Descrição	Tipo	Tamanho	CP	CE	Referência
Organization	Nome da metodologia	T	50	S	N	
ProjName	Nome do proprietário da metodologia	T	50	S	N	
ContactPerson	e-mail do proprietário da metodologia	T	50	N	N	
Telephone	Nome do distribuidor da metodologia	T	50	N	N	
FaxNumber	e-mail do distribuidor da metodologia	T	50	N	N	
eMailAddress	data de publicação da metodologia	T	50	N	N	
Address	Última data de atualização	T	50	N	N	
MethodologySelected	Nome da metodologia selecionada para o projeto	T	50	N	N	

ProjectCharacteristics

Nome	Descrição	Tipo	Tamanho	CP	CE	Referência
Organization	Nome da organização	T	50	S	S	ProjectProfile
ProjName	Nome do projeto	T	15	S	S	ProjectProfile
Characteristics	e-mail do proprietário da metodologia	T	50	S	S	Characteristics
CharacteristicsValues	Nome do distribuidor da metodologia	T	50	S	S	CharacteristicsValue
Weight	Peso atribuído ao valor da característica	N	8	N	S	Weight

[1] Tabela auxiliar.

Dissertação de Mestrado apresentada por **Pedro Luciano Leite Silva** a Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título, **"Um Processo para Seleção de Metodologias de Desenvolvimento de Software"**, orientada pelo **Prof. Alexandre Marcos Lins de Vasconcelos** e aprovada pela Banca Examinadora formada pelos professores:

Prof. Jaelson Freire Brelaz de Castro
Centro de Informática / UFPE

Profa. Ana Cristina Rouiller
Departamento de Ciência da Computação / UFLA

Prof. Alexandre Marcos Lins de Vasconcelos
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 27 de fevereiro de 2003.

Prof. AUGUSTO CESAR ALVES SAMPAIO
Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.