ANDRÉ LUIZ DE CASTRO LEAL

UMA PROPOSTA DE TAXONOMIA DE BOAS PRÁTICAS EM DESENVOLVIMENTO DE SOFTWARE

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

VIÇOSA
MINAS GERAIS - BRASIL
2009

ANDRÉ LUIZ DE CASTRO LEAL

UMA PROPOSTA DE TAXONOMIA DE BOAS PRÁTICAS EM DESENVOLVIMENTO DE SOFTWARE

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

APROVADA: 03 de julho de 2009.

Profª. Íris Fabiana de Barcelos Tronto
(Co-orientadora)

Prof. Alcione de Paiva Oliveira

Prof. Mauro Nacif Rocha

Prof. Marco Antônio Pereira Araújo

Prof. José Luís Braga
(Orientador)

Dedico a meu filho e esposa,
para que nossos esforços sempre nos
rendam bons frutos.
E principalmente para meu filho,
para que a arte de aprender possa lhe
ser tão prazerosa quanto a de brincar.

AGRADECIMENTOS

Agradeço principalmente ao Prof. José Luis Braga, pela oportunidade oferecida de ser meu orientador e poder aprender não só sobre questões técnicas, mas também poder desfrutar de seus conhecimentos gerais, paciência e colaboração.

Agradeço aos membros da banca, principalmente a disponibilidade dos membros externos, da sua paciência e das observações que irão tornar o trabalho mais completo.

E claro, a todos os colegas de turma.

BIOGRAFIA

ANDRÉ LUIZ DE CASTRO LEAL, filho de Oldemar Baccara Leal e Ligia Maria de Castro, brasileiro nascido em 09 de Agosto de 1971 na cidade de Juiz de Fora, no estado de Minas Gerais.

Atua no mercado desde 1992. É especialista em Ciência da Computação pela UFV e especialista em Gestão de TI pela Faculdade Machado Sobrinho.

Já atuou ou atua como professor em cursos superiores de graduação e pós-graduação em instituições como Faculdade Estácio de Sá, Doctum, Unipac Ubá e Faculdade de Ciências Gerenciais da cidade de Santos Dumont.

No ano da defesa da dissertação atuava como coordenador de projetos em uma fábrica de software na cidade de Juiz de Fora. Teve oportunidade de atuar como profissional de TI em empresas como PNUD/ONU, Embrapa Gado de Leite, Unimed, MRS Logística, Rhealeza Informática, Grupo Mult, CAEd, entre outros.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

LEAL, André Luiz de Castro, M.Sc., Universidade Federal de Viçosa, junho de 2009. **Uma proposta de taxonomia de boas práticas em desenvolvimento de software.** Orientador: José Luis Braga. Co-Orientadores: Iris Fabiana de Barcelos Tronto e Alcione de Paiva Oliveira.

Desde o início da década de 90, as empresas de software têm passado por transformações no seu modo de desenvolver produtos. Conseqüentemente, as abordagens de Engenharia de Software ganharam força no mercado e a busca por utilização de métodos científicos e ferramentas que façam aumentar a qualidade dos serviços têm crescido consideravelmente. O foco em abordagens que auxiliem a melhoria da qualidade de processos organizacionais e dos recursos humanos ganharam impulso e o setor direciona seus investimentos para a qualidade dos processos de serviços e produtos de software. Diante deste contexto, as micro e pequenas empresas passam por dificuldades em se adaptar às exigências de um mercado mais competitivo, suas carências de mão-de-obra e limitação de recursos impedem que implantem processos burocráticos e muito formais. Desta forma, é importante que essas empresas tenham uma alternativa no mercado para ser aplicada na sua realidade de desenvolvedora de software. A presente pesquisa apresenta uma taxonomia de boas práticas para auxílio no desenvolvimento de software para micro e pequenas empresas.

ABSTRACT

LEAL, André Luiz de Castro, M.Sc., Universidade Federal de Viçosa, june, 2009. **A proposed taxonomy of best practices in software development.** Adviser: José Luis Braga. Co-Advisers: Iris Fabiana de Barcelos Tronto and Alcione de Paiva Oliveira.

Since the early 90s, software companies have gone through changes in the way they develop products. Consequently, the approaches of Software Engineering gained strength in the market and the search for use of scientific methods and tools that can increase the quality of services has grown considerably. The focus on approaches that help to improve the quality of organizational processes and human resource gained momentum and the sector directs its investments in the quality of process services and software products. In this context, the micro and small businesses go through difficulties in adapting the requirements of a more competitive market, its lack of labor and limited resources prevent the implement of bureaucratic and very formal procedures. Thus it is important that they have an alternative in the market to be applied in their reality as a software developer. This research presents a taxonomy of best practices to aid in software development for micro and small enterprises.

1 INTRODUÇÃO

Diante da exigência das empresas contratantes, as empresas produtoras de software têm sido forçadas a aumentar a qualidade dos serviços prestados para atender às necessidades de seus clientes. Tais exigências de mercado estão diretamente relacionadas a projetos cada vez mais complexos e simultâneos gerenciados por essas empresas. Além de tratar com o gerenciamento de risco dos projetos, as empresas têm que tratar com a evolução, manutenção, dinamismo dos requisitos, complexidade e integração dos softwares atuais. Outro fator que impulsiona a profissionalização do setor é que clientes têm exigido prazos de entrega e custos menores. Diante disso, as empresas de software estão forçadas a concorrer em um mercado mais competitivo. Os investimentos de clientes estão voltados às empresas que garantam a entrega dos cronogramas financeiros e de prazos dentro do planejado e que tenham capacidade de lidar com o dinamismo dos requisitos de software e garantir a produtividade e qualidade.

Para garantir essas exigências de mercado, além de aplicar corretamente técnicas e métodos, as empresas de software necessitam de melhorar a qualidade dos processos de desenvolvimento de software. Os processos devem ser revistos e adequados às novas exigências de qualidade, prazo e custo, a fim de se produzir software com excelência. A qualidade dos produtos está fortemente ligada à qualidade do processo (Sebrae, 2004).

Nesse contexto em particular, encontram-se as micro e pequenas empresas (MPEs). Há vários conceitos de Micro e Pequena Empresa no Brasil, em geral, eles levam em consideração critérios quantitativos, como número de empregados ou faturamento anual bruto. Um dos conceitos que pode ser citado é o que estabelece a Lei 9.841, de 05 de outubro de 1999, que instituiu o atual Estatuto da Microempresa e da Empresa de Pequeno Porte, dispondo sobre o tratamento jurídico diferenciado, simplificado e favorecido, previsto nos arts. 170 e 179 da Constituição (1988), e revogando as leis anteriores que tratavam do assunto (Brasil, 1999). Para efeito de enquadramento de micro e empresas de pequeno porte, conforme estabelecido pela

Lei 9.841/99, o referencial é a receita bruta anual, cujos limites foram alterados em 01 de abril de 2005, pela Lei 5.028 (Brasil, 2005), conforme apresentado na **Tabela 1**.

Tabela 1: Caracterização de MPEs segundo receita bruta.

	Lei 9.841(5/10/99) Art 2o. I, II	**Lei 5.028(1/04/05) Art.1o. I, II**
Microempresa	I – receita bruta anual igual ou inferior a R$ 244.000,00	I - receita bruta anual igual ou inferior a R$ 433.755,14
Empresa de Pequeno Porte	II - superior a R$ 244.000,00 e igual ou inferior a R$ 1.200.000,00	II - superior a 433.755,14 e igual ou inferior a R$ 2.133.222,00

Fonte: (Brasil, 1999) e (Brasil, 2005).

Diante do contexto de exigências tais empresas necessitam não apenas de garantir a qualidade mencionada e imposições mercadológicas, mas também lidar freqüentemente com as dificuldades inerentes à sua realidade, tais como: inexistência de processos organizados e implementados, recursos humanos e financeiros escassos, baixo nível de conhecimento no domínio gerencial e inexistência de políticas de curto, médio ou longo prazos que auxiliem na melhoria dos processos de desenvolvimento.

Há atualmente no mercado inúmeras pesquisas e esforços para melhorar os processos de desenvolvimento de software. É importante se preocupar tanto com a qualidade do produto quanto com a qualidade dos processos de desenvolvimento para que o produto final esteja de acordo com as expectativas de qualidade exigidas pelo cliente (Rocha et al., 2001). Como exemplo de esforço na área de pesquisa, podemos citar o que propõe a Sociedade Brasileira de Computação (SBC). Em um de seus Grandes Desafios da Pesquisa em Computação no Brasil ela apóia o Desenvolvimento Tecnológico de Qualidade com foco concentrado em sistemas disponíveis, corretos, seguros, escaláveis, persistentes e ubíquos (SBC, 2006). O desafio está relacionado às pesquisas sobre ambientes, métodos, técnicas, modelos, dispositivos e padrões de arquitetura e de projeto capazes de auxiliar os projetistas e desenvolvedores de grandes sistemas de software e hardware a atingirem esses objetivos (SBC, 2006).

1.1 O cenário das MPEs no Brasil

Em um contexto de produção, empresas de diferentes portes passam por dificuldades para obter qualidade em seus produtos finais. Em particular as MPEs tendem a ter maior dificuldade por trabalharem com processos informais e a falta de recursos humanos e financeiros. No setor de software a dificuldade ainda é maior devido à ubiqüidade do produto e a complexidade dos processos produtivos. As principais dificuldades encontradas por empresas no desenvolvimento de software são, segundo Jones (1996) e Yourdon (1997):

1. incompreensão das necessidades do usuário final;
2. software difícil de manter e entender;
3. baixo desempenho e baixa qualidade;
4. recursos humanos atuando nas diversas fases do ciclo de vida do software sem um planejamento de atuação;
5. inexistência de processos de rastreabilidade para melhor manutenção de códigos fontes;
6. falta de conhecimento em métodos e técnicas de ES.

Além das dificuldades inerentes ao desenvolvimento do produto software, as MPEs têm presentes dificuldades do tipo:

1. investimentos focados em linguagens de programação e desenvolvimento e não em ES;
2. inexistência de processos organizados e aplicados;
3. recursos humanos e financeiros limitados;
4. baixo nível de conhecimento no domínio gerencial;
5. inexistência de políticas de curto ou longo prazo que auxiliem na melhoria dos processos de desenvolvimento.

Com isso empresas de porte pequeno encontram grandes problemas na sua produtividade, qualidade e competitividade no mercado, ocasionando na grande maioria das vezes o fechamento dessas empresas.

A falta de um processo sistemático de desenvolvimento de software é citada no relatório de 2001 do Ministério da Ciência e Tecnologia (MCT) (2001) como um dos pontos que prejudica a pequena empresa na qualidade e produtividade do processo e do produto desenvolvido. Um outro fator importante no insucesso das empresas frente a sua sobrevivência no mercado é o acesso limitado aos incentivos

fiscais e de políticas de crédito para ciência e tecnologia. Além disso, freqüentemente ignoram a existência destes incentivos (Rovere, 2000).

Em (Vos et al., 1998), cita-se que os administradores dessas empresas com tamanho reduzido, tendem a ter um horizonte de planejamento de curto prazo, ficando presos aos problemas do dia a dia e impedidos de atuar na definição de estratégias de longo prazo e de inovação.

Segundo pesquisas do MCT (2005), a maior concentração de empresas de informática é de MPEs com até 9 pessoas. A **Figura 1** e a **Tabela 2** apresentam a estatística por porte de empresa, segundo pesquisa do MCT (2005) dividida nas regiões do Brasil.

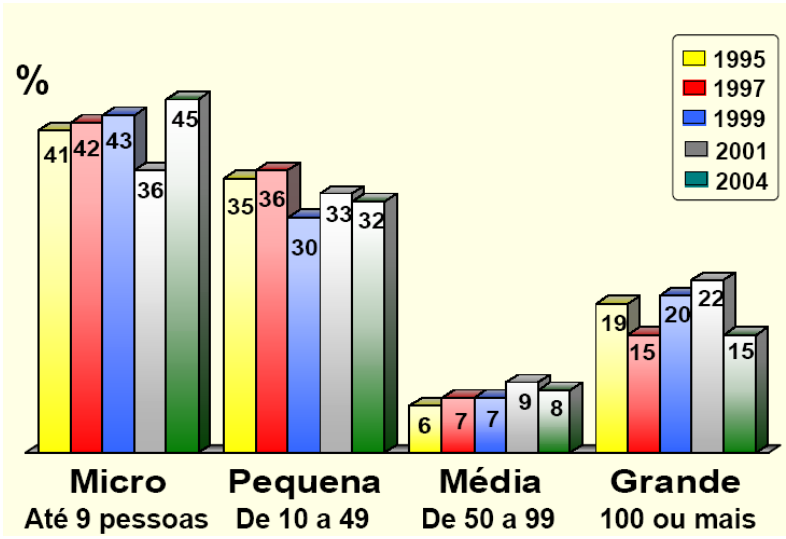

Figura 1: Porte das empresas, segundo força de trabalho efetiva.
Resultados da Pesquisa de 2005 do MCT (2005).

Tabela 2: Atividades de informática e serviços relacionados.
Distribuição percentual de estabelecimentos por porte segundo regiões
Resultados da Pesquisa de 2005 do MCT.

Regiões	**PORTE**				**TOTAL**
	Micro	**Pequena**	**Média**	**Grande**	
Norte	85,8	11,8	1,2	1,2	100
Nordeste	84,5	12,4	1,4	1,7	100
Sudeste	82,1	14,9	1,6	1,4	100
Sul	87,1	11,4	0,9	0,7	100
Centro-oeste	85,3	11,4	1,2	2,1	100
BRASIL	83,8	13,5	1,4	1,3	100

Fonte: MCT, 2005.

Com o intuito de auxiliar as empresas de software na busca pela melhoria na qualidade de seus processos produtivos existem no mercado nacional e internacional,

vários modelos de qualidade: MPS.BR (2005), a série da Norma ISO 9000 (ABNT, 2001), CMM/CMMI (SEI, 2002), ISO/IEC 15504 (2003) , entre outros.

Apesar das propostas de avaliação para melhoria dos processos organizacionais de desenvolvimento de software, as empresas envolvidas necessitam de um grande esforço até a avaliação oficial. Os modelos sugeridos não vêm acompanhados de técnicas ou métodos para que sejam atingidos os objetivos. Por exemplo, para se atingir o nível G do modelo MPS.BR é necessário que a empresa cumpra uma série de passos, mas não há na abordagem a sugestão de práticas ou mesmo ferramentas computacionais sugeridas para se atingir uma melhor gerência de projetos e de requisitos. Portanto, o caminho até a avaliação requer da empresa muita dedicação, disciplina e a implantação de métodos e técnicas formais que sejam satisfatórias às exigências do nível requerido.

As pesquisas do MCT apresentam um baixo número de MPEs que atingem o nível de avaliação desejado. Isso é causado pela falta de aderência dos modelos à realidade dessas MPEs. Segundo pesquisas do MCT (2001), a parcela de MPEs com certificação ISO 9000 ou CMM até o ano de 1999 eram apenas 7% do total. Apesar da possibilidade da adaptação dos modelos existentes para serem aderentes as MPEs, há uma necessidade de um maior esforço e de grande experiência para sua implantação, pois os modelos atuais atendem a um perfil de empresas que estão mais estruturadas em termos financeiros e organizacionais, caracterizando-se em um nível acima da grande maioria das empresas consideradas como MPEs.

Apesar da recompensa pelo ganho da qualidade, maturidade e controle dos processos de desenvolvimento de software, a opção pela implantação do modelo dever ser bem avaliada, pois a curva de aprendizado de um método formal é longa. Para Aggarwal (1998), a taxa de sobrevivência das pequenas empresas no mercado depende do ambiente tecnológico onde a empresa atua e do seu tempo de atuação. Curvas de aprendizado longas podem trazer dificuldades na implantação de políticas que requerem maior esforço de adoção, uma vez que a taxa da mortalidade das empresas no Brasil é elevada.

Conforme estatísticas do SEBRAE (2004), os índices de encerramento de empresas no Brasil é alto, 49,4% acabam antes de completar dois anos, 56,4% fecham durante os três primeiros anos e 59,9% nos quatro primeiros anos. Diante dessa realidade, são altas as chances das empresas encerrarem antes de terminarem o processo de certificação de melhoria de qualidade.

Portanto, há uma necessidade premente dessas empresas de iniciarem suas atividades de melhoria da qualidade com passos mais curtos a partir de adoção de boas práticas da ES. Após a escolha e utilização das boas práticas, devem iniciar seu esforço de implantação dos níveis mais básicos de avaliação disponíveis nos atuais modelos no mercado. Com base no contexto, entende-se que serão minimizados os impactos iniciais das exigências desses níveis de avaliação nas empresas onde a cultura do formalismo é inexistente. Objetivando a implantação inicial de métodos mais aderentes à sua realidade as MPEs terão condições de aumentar seu controle dos processos produtivos e melhorar as chances de sobrevivência no mercado, além de estarem mais preparadas para aumentar a competitividade.

1.2 Perfil de risco para adoção de boas práticas de desenvolvimento de software nas empresas

O grupo de pesquisas em ES do Departamento de Informática da Universidade Federal de Viçosa (GPES-DPI-UFV) e o APL TI-Viçosa, apoiados pelo SEBRAE, iniciaram em 2007 um trabalho de parceria com vistas à avaliação do perfil de risco associado à adoção de boas práticas e processos em desenvolvimento de software. O trabalho foi executado em oito empresas da APL TI-Viçosa, e na NoBugs - Empresa Júnior do curso de computação do DPI da UFV.

O foco principal da pesquisa de campo foi identificar os riscos de adoção de métodos dirigidos por planejamento (MDP) ou métodos ágeis (MA) ou híbridos. Segundo Boehm e Turner (2004), a combinação de características de MA e MDP, originando métodos híbridos, dão uma maior aderência a diferentes contextos de projetos de software. Sendo conhecidos os riscos, determinados a partir do perfil de cada empresa, é possível determinar um conjunto de boas práticas a serem adotadas pelas empresas individualmente ou coletivamente, permitindo às empresas subir um ou mais degraus no rumo da qualidade na produção de software.

Pelo porte das empresas participantes e do contexto empresarial em que estão inseridas, o grupo de pesquisa entendeu que essas fazem parte de um perfil de empresas consideradas como MPEs, pois seu porte, em temor do número de profissionais é inferior a 10, e os projetos a que se dedicam são também de pequeno porte. O termo empresa, nesse trabalho, se refere à micro-empresa, que foi nosso objeto de estudo.

1.2.1 Os fatores avaliados

De acordo com a pesquisa de Soares et al. (2008), os cinco fatores sugeridos no trabalho de pesquisa para a avaliação das empresas são uma extensão do *framework* de Cockburn (2000). Segundo Cockburn (2000), considerar múltiplos métodos é apropriado e necessário devido às distinções entre os vários projetos que as empresas costumam desenvolver. Nesse contexto, a opção por avaliar MDP, MA e híbridos é uma escolha que facilita a aderência das propostas aos diferentes projetos das empresas. Cockburn (2000) considera duas dimensões em seu framework: o tamanho da equipe e a criticalidade do sistema.

O trabalho desenvolvido nas empresas foi embasado nos resultados apresentados por Boehm e Turner (2003) que estendem o *framework* de Cockburn em (2000) e Boehm e Turner (2003 e 2004), com uma proposta de abordagem baseada em risco para balancear adoção de MA e MDP. Desta forma, pode-se obter uma abordagem híbrida que se adeque melhor às necessidades, objetivos, restrições e prioridades de um projeto ou organização (Soares, 2007).

Os cinco fatores críticos avaliados nas empresas podem ser discriminados, segundo Boehm e Turner (2004), como:

- Tamanho: refere-se ao tamanho da equipe envolvida e ao esforço necessário no desenvolvimento;

- Criticalidade: refere-se às perdas ocasionadas em função de erros presentes no produto. Relaciona-se ao risco a vidas humanas pelo uso do software e a outros riscos menos graves, com impacto direto na exigência do nível de qualidade;

- Dinamismo: refere-se à possibilidade de mudanças nos requisitos em função de mudanças no ambiente do problema, ou em função de um trabalho de extração mal executado, implicando em alterações no software;

- Competência Pessoal: refere-se à competência dos profissionais envolvidos no desenvolvimento. Um profissional pode ser associado a cinco níveis de experiência, variando de pouco experiente a especialista;

- Cultura: refere-se à cultura dos desenvolvedores em relação ao desenvolvimento e gerência de projetos de software. Os desenvolvedores podem possuir a cultura na qual se sentem mais confortáveis tendo muitos graus de liberdade para produzir o software, ou podem ter a cultura na qual se sentem mais confortáveis trabalhando com regras, procedimentos e políticas bem definidas e

estabelecidas. Os fatores foram organizados graficamente conforme é apresentado na **Figura 2**.

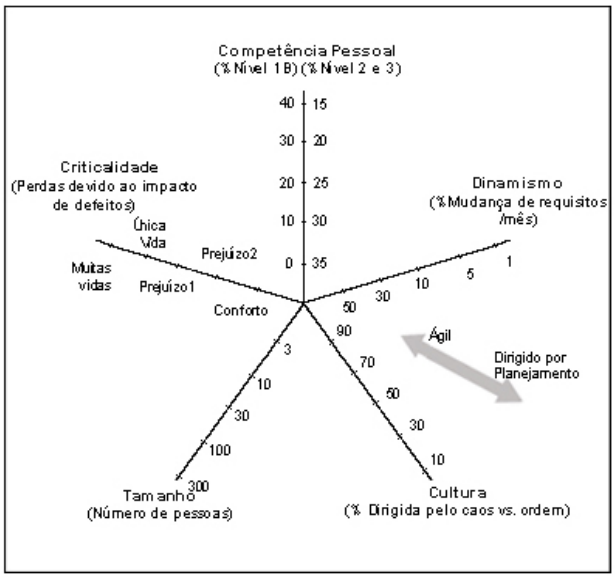

Figura 2: Modelo de Gráfico Polar (Boehm e Turner, 2004).

No eixo do fator *Criticalidade* são encontrados itens que podem ser considerados como:

- Prejuízo 1: prejuízo maior, capaz de provocar impacto negativo no projeto;

- Prejuízo 2 e Conforto: prejuízo menor, com pouco impacto financeiro e nenhum risco à vida humana;

- Única Vida e Muitas Vidas: se referem aos riscos à vida humana, caso o software resultante apresente erros. Adaptação de Soares (2007) do original proposto por Boehm e Turner (2004).

O eixo de *Competência* foi medido e plotado com base nos níveis de habilidades dos profissionais envolvidos. Cockburn (2002) identificou os níveis 1, 2 e 3 (3 é o mais experiente) como níveis de habilidades dos profissionais em desenvolvimento de software. Boehm e Turner (2004), dividiram o nível 1 em -1, 1B e 1A para permitir uma melhor identificação de riscos de adoção de MA ou MDP e lidar com métodos *ad-hoc*. Assim, têm-se os cinco níveis de habilidade mostrados na **Tabela 3** e pelos quais os profissionais têm sua competência pessoal definida.

Tabela 3: Classificação dos níveis de habilidade dos indivíduos.

Nível	Características
3	Capaz de revisar um método (quebrar suas regras) para ajustá-lo a uma nova situação sem precedentes.
2	Capaz de construir um método que se ajuste a novas situações com precedentes.
1A	Com treinamento, é capaz de seguir os passos de um método discreto (dimensionar histórias para ajustar incrementos, definir padrões, realizar refatorações complexas). Com experiência, pode alcançar o nível 2.
1B	Com treinamento, é capaz de seguir os passos de um método procedural (codificar um método simples, realizar refatorações simples, seguir padrões de codificação). Com experiência, pode atingir habilidades do nível 1A.
-1	Possui habilidades técnicas, mas não é capaz ou não está disposto a colaborar ou seguir métodos compartilhados.

Fonte: Elaborado pelo autor, adaptado de (Boehm e Turner, 2004).

1.3 O problema e sua importância

Para Soares (2007), com tantas técnicas, métodos e abordagens disponíveis na aplicação de um processo, a escolha desses elementos a serem usados em cada tipo de problema ainda é feita de maneira empírica, sem bases científicas que justifiquem a adoção de uma delas ou que procure a melhor aderência do método ou técnica ao tipo de problema em questão. Um amplo número de abordagens está disponível hoje no mercado para serem utilizados como diferencial na produção de software com qualidade. Apesar disso, as empresas continuam não cumprindo prazos de entrega, provocando aumento no custo de produção e produzindo sem atender as reais expectativas dos clientes (Sommerville, 2003). Nesse contexto, há uma resposta à pergunta a seguir será importante na influência da produção de softwares com qualidade nas MPEs, contexto desse trabalho:

Como selecionar boas práticas de Engenharia de Software (ES), baseado em critérios técnicos que levem em consideração o risco de adoção e que se ajustem à realidade de cada empresa baseado em diagnóstico de suas características?

1.4 Justificativa para desenvolvimento do tema

Segundo pesquisas do MCT (2001), as MPEs de desenvolvimento de software com certificação ISO 9000 ou CMM até o ano de 1999 eram apenas 7% do total. Apesar da possibilidade da adaptação dos modelos existentes para serem aderentes às MPEs, há uma necessidade de um maior esforço e de grande experiência para sua implantação, pois os modelos atuais atendem a um perfil de empresas que estão mais estruturadas em termos financeiros e organizacionais, caracterizando-se em um nível

acima da grande maioria das empresas consideradas como MPEs. Diante desta realidade, faz-se necessário uma pesquisa que apresente resultados que possam ser uma alternativa de melhoria de qualidade para as MPEs na produção de software.

1.5 Objetivos da dissertação

O objetivo da presente pesquisa é a criação de uma taxonomia de boas práticas de ES que melhore o entendimento das relações entre seus elementos aumentando assim as chances de adotar práticas corretas para os problemas e causas identificados na produção de software em MPEs.

Especificamente pretende-se:

- Pesquisar e conhecer técnicas que possam ser utilizadas para a identificação de boas práticas em desenvolvimento de software;

- Criar uma taxonomia de boas práticas justificando suas relações utilizando abordagem sistêmica;

- Apresentar, de forma justificada e relacionada, os principais problemas encontrados em qualidade de software, suas causas e a adoção de boas práticas que auxiliem a solucionar tais problemas;

- Apresentar e discutir um estudo de caso, mostrando como a proposta pode ser utilizada em micro-empresas.

1.6 Metodologia

A realização do trabalho foi composta de atividades que podem ser discriminadas como:

- **Atividade 1 (1A)**: participação na consultoria da empresas do Arranjo Produtivo Loca de Tecnologia da Informação de Viçosa (APL-TI-Viçosa) com a aplicação de questionários para a identificação de diagnóstico inicial de risco de adoção de métodos dirigidos por planejamento, ágeis ou híbridos. Os questionários foram aplicados com o objetivo de se medir os cincos fatores críticos sugeridos por Boehm e Turner (2004) e que impactam o desenvolvimento de software. Análise dos questionários e apresentação dos resultados para as empresas do APL. Tais questionários, e todo o trabalho de consultoria tiveram como base a pesquisa bibliográfica e dos estudos do trabalho científico de Soares (2007).

- Atividade 2 (2A): avaliação do potencial das empresas para a adoção de processos mais longos e burocráticos para o desenvolvimento de software e a verificação de falta de recursos necessários para tal abordagem.

- Atividade 3 (3A): estudo bibliográfico sobre as principais abordagens organizacionais disponíveis atualmente no mercado para desenvolvimento de software.

- Atividade 4 (4A): estudo bibliográfico sobre ontologias.

- Atividade 5 (5A): apresentação dos conceitos que envolvem a teoria do trabalho de Boehm e Turner (2004) com relação aos fatores críticos de desenvolvimento de software.

- Atividade 6 (6A): criação de uma taxonomia de boas práticas de desenvolvimento de software, com apresentação de principais problemas, causas e relação com boas práticas de software. Apresentação de inter-relações de elementos da taxonomia e o seu grau de influência recíproco justificado através de diagramas de influência.

- Atividade 7 (7A): apresentação dos resultados de duas empresas oriundas da consultoria na APL-TI-Viçosa e a justificativa do uso da taxonomia para melhorar a qualidade de produção de softwares.

A **Figura 3** apresenta o diagrama das atividades envolvidas neste trabalho de pesquisa.

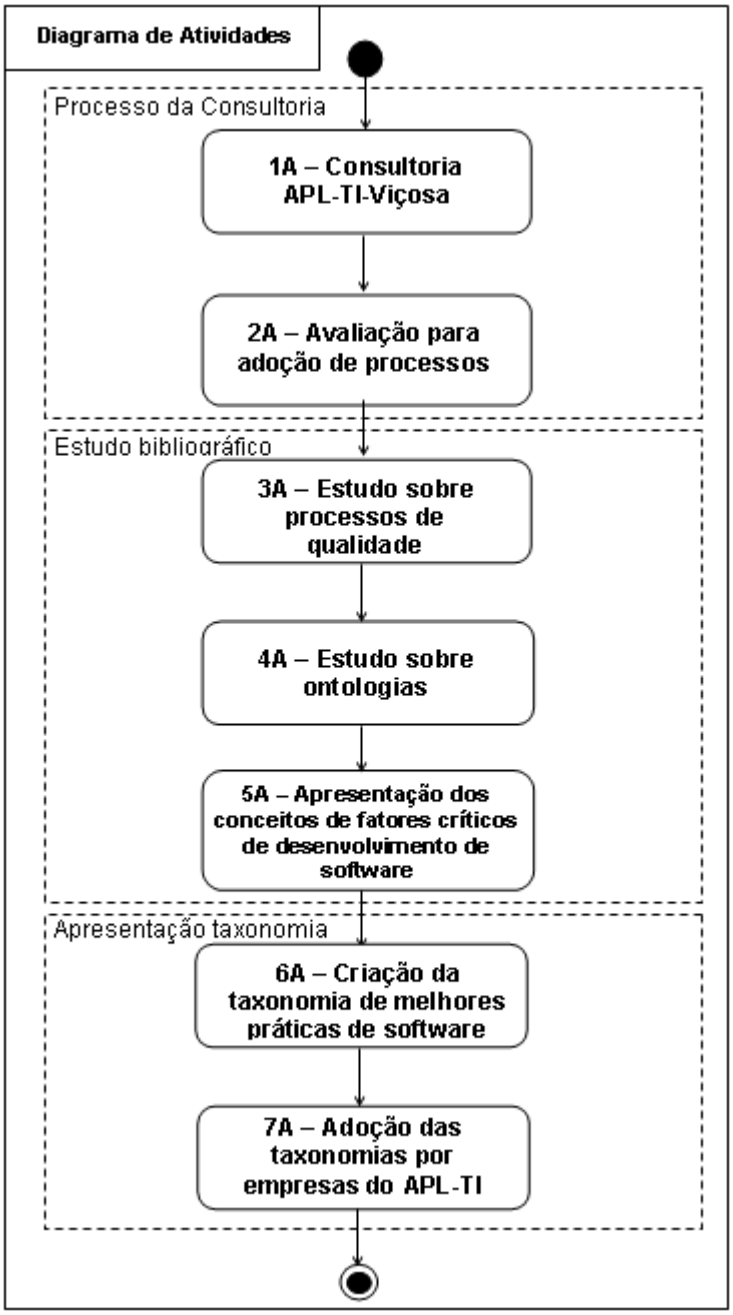

Figura 3: Apresentação de atividades do trabalho.

2 Modelos de Qualidade de Processos

A crescente demanda por sistemas computacionais no cotidiano de empresas e dos seres humanos, tem direcionado organizações na busca por desenvolver seus trabalhos amparados em processos definidos e cíclicos de desenvolvimento de softwares. A busca por profissionalização desse desenvolvimento se dá basicamente por demandas cada vez mais complexas devido principalmente ao dinamismo do mercado onde atuam essas organizações.

Portanto, cabem algumas perguntas: Como concorrer em um mercado que exige cada vez mais informações complexas e dinâmicas? Como conviver com softwares embutidos nos principais aparelhos de uso do cotidiano tornando-os de fácil uso, quase imperceptíveis? Como garantir que os softwares implantados tenham um ciclo de vida estável e longo com um mercado cada vez mais competitivo e dinâmico? Como conciliar o desenvolvimento do que está especificado, com prazo e evolução da informação e a necessidade de mudanças nos processos empresariais perante um mercado exigente? Como as MPEs se adaptam a um contexto tão adverso contando com recursos tão escassos?

São perguntas que necessitam ser respondidas. Algumas das respostas têm sido estudadas na ES, visando a promover uma verdadeira revolução na área de desenvolvimento de sistemas computacionais. Para isso, propõe técnicas e metodologias com o objetivo de aumentar a eficiência dos processos de software. Aliada a essas soluções garante-se a entrega de produtos em tempos hábeis, com custo atendendo a cronogramas físicos e financeiros pré-estabelecidos e principalmente atendendo a requisitos cada vez mais complexos e dinâmicos.

Segundo Sommerville (2003), a ES é uma disciplina que envolve boas práticas de software, cujo foco é o desenvolvimento de sistemas de software de alta qualidade com eficiência relativa ao custo, para solucionar problemas cuja análise demonstrou a necessidade do uso de um sistema de software.

Com a exigência do mercado e a demanda crescente por software mais complexos e projetos simultâneos nas empresas, surgiram os modelos com a

premissa de aumentar a qualidade dos processos organizacionais de desenvolvimento de software aplicáveis em vários níveis de competência:

CMM – *Capability Model Maturity e SW-CMM - Capability Maturity Model for Software* aplicável em nível organizacional para organizações voltadas ao desenvolvimento de software (SEI, 1997), além da evolução do modelo para o CMMI - *Capability Maturity Model Integration* (SEI, 2002);

MPS.BR – Melhoria de Processo de Software Brasileiro – aplicável em nível organizacional (Softex, 2005);

SPICE – *Software Process Improvement and Capability Determination* (norma ISO/IEC 15504: *Information Technology, Software Process Assessment*) – aplicável em nível organizacional (ISO/IEC 15504, 2003).

2.1 *Capability Maturity Model for Software* - SW-CMM e *Capability Model Maturity* - CMM

O modelo *Capability Maturity Model* (ou Modelo de Capacidade e Maturidade) (CMM) foi produzido pelo *Software Engineering Institute* (SEI) da Universidade Carnegie Mellon (CMU), em Pittsburgh, EUA, por um grupo de profissionais de software, sendo a 1ª versão lançada em ago/1991. Sendo sua primeira versão o *Capability Maturity Model for Software* (SW-CMM), focado apenas na disciplina da engenharia de software. Depois do SW-CMM, outros CMM's foram desenvolvidos, focados em outras disciplinas, como engenharia de sistemas, subcontratação, pessoas e desenvolvimento integrado de produtos (SEI, 1997).

Segundo discriminado no documento guia do SW-CMM (SEI, 1997) ele é definido como um conjunto de regras (ou ações) que visa maximizar a garantia da qualidade de software através da adoção de três princípios básicos (Paulk, 1993):

1. Elaboração de um questionário com o intuito de classificar o nível de maturidade da empresa produtora ante o processo de produção;
2. Avaliação do processo de software, visando determinar o nível atual de desenvolvimento do mesmo;
3. Avaliação da capacidade do software, identificando fornecedores e práticas adequadas para sua melhor confecção.

Além disso, o SW-CMM descreve os elementos-chave da evolução de um processo de software imaturo para um processo maduro e disciplinado. Abrange

práticas para planejamento, engenharia e gestão do desenvolvimento de software que, quando seguidas, melhoram a habilidade da organização em atender metas para custos, cronograma, funcionalidade e qualidade do produto.

No modelo, as definições básicas que permitem qualificar uma organização, conforme especificação do *Institute of Electrical and Electronics Engineers* (IEEE) (Paulk, 1993), podem ser vistas como:

1. Processo: É uma seqüência de passos cuja finalidade é atingir um objetivo próprio;
2. Processo de software: É o conjunto de atividades, métodos, ações e operações para produzir e manter os produtos de programação;
3. Capacidade do processo de software: Determina a gama de resultados esperados para o processo de software implantado;
4. Maturidade do processo de software: Caracteriza a definitiva aplicação dos processos necessários à capacitação plena.

Assim, pelas premissas mencionadas, o objetivo primordial do modelo é avaliar os processos de desenvolvimento software utilizados pelas empresas, visando estabelecer projeções para os aspectos de qualidade, custos e prazos, abrangendo tanto o desenvolvimento de programas quanto à manutenção dos mesmos, passando, também, pela sua aquisição (compra de pacotes prontos e respectivas manutenções) (SEI, 1997).

A estrutura do SW-CMM consiste dos cinco níveis de maturidade propostos inicialmente no modelo CMM e determina em qual dos níveis se encontram os processos de software das empresas, identificando pontos fortes e fracos e apontando caminhos para as melhorias necessárias. Como podem ser visualizados na **Figura 4**, os níveis propostos no CMM são: Inicial, Repetível, Definido, Gerenciado e Em otimização.

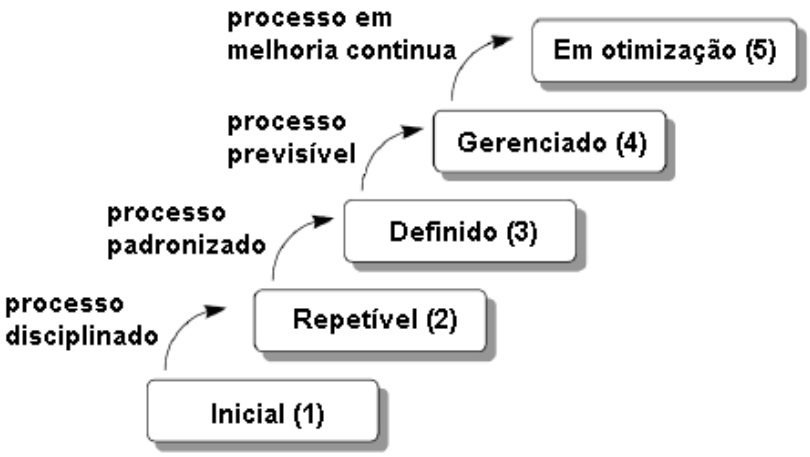

Figura 4: Níveis CMM (Paulk et al., 1993).

Cada nível de maturidade do modelo é composto por um conjunto de práticas que devem ser efetivas e executadas para que cada nível atinja a maturidade desejada. Esse conjunto de práticas é conhecido por *Key Process Area* (KPAs) (Paulk, 1993).

Após o primeiro nível, cada nível do modelo é decomposto em áreas-chave de processo, que indicam aquelas na qual uma organização deve focar seus esforços para a melhoria de seus processos de desenvolvimento de produto. Áreas-chaves são algumas áreas imprescindíveis para a maturidade do processo. Elas são identificadas em função de sua eficácia e eficiência na melhoria da capacidade do processo. Cada área-chave de processo identifica um grupo de atividades relacionadas, as quais, quando executadas coletivamente, atingem um conjunto de metas consideradas importantes para aumentar a capacidade do processo. Também cada área-chave é subdividida em um conjunto de práticas base que foram classificadas de acordo com cinco tipos de características comuns (Paulk, 1993):

a) Compromissos a realizar: as práticas agrupadas nesta classe referem-se a ações da alta gerência da organização. Exemplos: criação de estruturas e grupos e atribuição de responsabilidades;

b) Habilidades para realizar: as práticas agrupadas nesta classe referem-se à capacitação das pessoas e disponibilização de recursos humanos e materiais;

c) Atividades Realizadas: aqui estão as práticas básicas propriamente ditas, as atividades e tarefas necessárias para atingir os objetivos e metas da KPA;

d) Medição e Análise: feitas para monitorar quantitativamente a situação do processo e usadas para controle e melhoria;

e) Verificação da Implementação: atividades que permitem comparar o estabelecido nos procedimentos e planos com as práticas realmente realizadas. A verificação engloba tipicamente revisões e auditorias feitas pelo gerenciamento e pela garantia da qualidade de software.

Além do SW-CMM outras iniciativas de melhoria de processos organizacionais sugiram a partir do sucesso do modelo anterior (SEI, 2002).

- *Software Acquisition Capability Maturity Model* (SA-CMM): usado para avaliar a maturidade de uma organização em seus processos de seleção, compra e instalação de software desenvolvido por terceiros.

- *Systems Engineering Capability Maturity Model* (SE-CMM): avalia a maturidade da organização em seus processos de engenharia de sistemas, concebidos como algo maior que o software. Um sistema inclui o hardware, o software e quaisquer outros elementos que participam do produto completo.

- *Integrated Product Development Capability Maturity Model* (IPD-CMM): ainda mais abrangente que o SE-CMM, inclui também outros processos necessários à produção e suporte ao produto, tais como suporte ao usuário, processos de fabricação, entre outros.

- *People Capability Maturity Model* (P-CMM): avalia a maturidade da organização em seus processos de administração de recursos humanos no que se refere ao software; recrutamento e seleção de desenvolvedores, treinamento e desenvolvimento, remuneração, entre outros.

Com base nos problemas gerados pela diversidade de modelos que surgiram, como por exemplo, a falta de padronização de termos, diferentes números de níveis e formas diferentes para avaliar o progresso de maturidade da empresa, surgiu o modelo conhecido como *Capability Maturity Model Integration* (CMMI) (SEI, 2002).

2.2 *CMMI - Capability Maturity Model Integration*

A proposta do CMMI é a de um modelo integrado que pode ser utilizado em várias disciplinas, como disciplinas de engenharia de sistemas, engenharia de software, desenvolvimento de integração de processos e produtos e subcontratação (fornecedor/aquisição). Com isto, observa-se que não se fala mais em software, como o SW-CMM, mas em produtos e serviços, abrangendo disciplinas distintas (SEI, 2002).

2.2.1 Componentes CMMI

Com o objetivo de integrar os diversos CMM's numa estrutura única com mesma terminologia e processos de avaliação, aproveitar a experiência de vários anos no uso do CMM e também os compatibilizar com a ISO 15504/SPICE, o SEI criou o modelo CMMI (SEI, 2002). O CMMI é um guia desenvolvido pela comunidade de software. É um modelo único que integra os CMM's: SW-CMM, SE-CMM, SA-CMM e IPD-CMM.

O CMMI incorpora as necessidades de melhorias identificadas pelo uso do CMM no mundo. É compatível com a norma ISO/IEC 15504, é alinhado com o *Project Management Body of Knowledge* (PMBOK) (PMI, 2004) e possui um direcionamento claro e objetivo para a interpretação das práticas, apresentando subpráticas e produtos típicos de trabalho para cada prática (CMU, 2002).

O CMMI é estruturado através de um conjunto de componentes, agrupados em três categorias que refletem como eles devem ser interpretados (SEI, 2002):

- Requeridos: são os componentes essenciais para avaliar a satisfação de uma área de processo. São as metas específicas (SG's - *Specific Goals*) e as metas genéricas (GG's - *Generic Goals*);

- Esperados: servem como guias para quem implementa melhorias ou executa avaliações. Descrevem o que deve ser feito, mas não como fazer. São as práticas específicas (SP's - *Specific Practices*) e práticas genéricas (GP's - *Generic Practices*);

- Informativos: fornecem detalhes que ajudam os usuários do modelo a pensarem na forma como irão abordar as metas e as práticas. São as sub-práticas, os produtos típicos de trabalho, as amplificações relativas a uma disciplina, os textos descritivos relativos a práticas genéricas, os nomes das metas e das práticas, as notas relativas às metas e práticas e as referências.

2.2.2 Disciplinas e Áreas do CMMI

As disciplinas do CMMI podem ser vistas conforme discriminadas a seguir (SEI, 2002):

- Engenharia de sistemas (*Systems Engineering* - SE): abrange o desenvolvimento total de um sistema, o qual pode ou não incluir software. O seu foco é na transformação das necessidades do usuário, expectativas e restrições em soluções de produtos e nas atividades de suporte para toda a vida do produto.

- Engenharia de software (*Software Engineering* - SW): abrange o desenvolvimento de sistemas do software, com foco na aplicação sistemática, disciplinada e quantitativa de abordagem para o desenvolvimento, operação e manutenção de sistemas.

- Desenvolvimento integrado de produtos e processos (*Integrated Product and Process Development* - IPPD): abordagem sistemática para alcançar a colaboração dos *stakeholders* [1] relevantes em todo o ciclo de vida do produto, de forma a satisfazer as necessidades, expectativas e requisitos dos usuários. Os processos que suportam IPPD são integrados a outros processos da organização.

- Subcontratação (*Supplier Sourcing* - SS): abrange a aquisição de produtos de fornecedores considerando que, quando o trabalho se torna complexo, os projetos podem fazer uso de fornecedores para realizar algumas atividades ou para gerar acréscimo aos produtos.

Além das disciplinas, o CMMI sugere um agrupamento de práticas relacionadas que satisfazem a um conjunto de metas para a progressão da melhoria da área onde esta sendo aplicado o modelo. Elas podem ser vistas conforme discriminado na **Tabela 4** (SEI, 2002):

[1] O SEI define *stakeholder* como grupo ou indivíduo atingido ou envolvido de alguma forma no resultado de uma atividade.

Tabela 4: Agrupamento de práticas CMMI

Agrupamento de Práticas CMMI	Descrição Original	Sigla
Gerência de requisitos	*Requirement Management*	REQM
Planejamento de projeto	*Project Planning*	PP
Monitoramento e controle de projeto	*Project Monitoring and Control*	PMC
Gerência de subcontratação	*Supplier Agreement Management*	SAM
Medição e análise	*Measurement and Analysis*	MA
Garantia da qualidade do processo e produto	*Process and Product Quality Assurance*	PPQA
Gerência de configuração	*Configuration Management*	CM
Desenvolvimento de requisitos	*Requirements Development*	RD
Solução técnica	*Technical Solution*	TS
Integração do produto	*Product Integration*	PI
Verificação	*Verification*	VER
Validação	*Validation*	VAL
Foco no processo organizacional	*Organizational Process Focus*	OPF
Definição do processo organizacional	*Organizational Process Definition*	OPD
Treinamento organizacional	*Organizational Training*	OT
Gerência de riscos	*Risk Management*	RSKM
Análise e definição da decisão	*Decision Analysis and Resolution*	DAR
Gerência do projeto integrado	*Integrated Project Management*	IPM
Integração de equipe	Integrated Teaming	IT
Ambiente organizacional para integração	*Organizational Environment for Integration*	OEI
Gerência integrada de fornecedor	*Integrated Supplier Management*	ISM
Desempenho do processo organizacional	*Organizational Process Performance*	OPP
Gerência quantitativa de projeto	*Quantitative Project Management*	QPM
Inovação e distribuição organizacional	*Organizational Innovation Deployment*	OID
Análise e definição causal	*Causal Analysis and Resolution*	CAR

Fonte: (CMU, 2002).

2.3 Melhoria de Processo do Software Brasileiro - MPS.BR

2.3.1 Descrição Geral

O MPS.BR atende à necessidade de implantar os princípios de Engenharia de Software de forma adequada ao contexto das empresas brasileiras, estando em consonância com as principais abordagens internacionais para definição, avaliação e melhoria de processos de software.

O MPS.BR tem como objetivo definir um modelo de melhoria e avaliação de processo de software, preferencialmente para as micro, pequenas e médias empresas, de forma a atender às suas necessidades de negócio e a ser reconhecido nacional e internacionalmente como um modelo aplicável à empresa de software. Este é o motivo pelo qual ele está aderente a modelos e normas internacionais. O MPS.BR também define regras para sua implementação e avaliação, dando sustentação e garantia de que o MPS.BR está sendo empregado de forma coerente com as suas definições (MPS.BR, 2005).

O foco principal do MPS.BR está nas pequenas e médias empresas brasileiras de software. Geralmente empresas com poucos recursos e que necessitam melhorar radicalmente seus processos de software em 1 ou 2 anos. Essas empresas precisam saber como adaptar à sua realidade o correspondente aos níveis de maturidade 2 e 3 de modelos para melhoria de processos de software como preconizado em CMMI-SE/SWSM (SEI, 2002) e a ISO/IEC 15504-5 (ISO/IEC 15504, 2003). Com isso, pretende-se que o modelo seja adequado ao perfil de empresas com diferentes tamanhos e características, públicas ou privadas, embora com especial atenção às micro, pequenas e médias empresas; seja compatível com os padrões de qualidade aceitos internacionalmente e que tenha como pressuposto o aproveitamento de toda a competência existente nos padrões e modelos de melhoria de processo já disponíveis. Nesse contexto, ele tem como base os requisitos de processos definidos nos modelos de melhoria de processo.

A base técnica utilizada para a construção do MPS.BR é composta pelas normas NBR ISO/IEC 12207 (Lin, 2003) – Processo de Ciclo de Vida de Software e suas emendas 1 e 2 e a ISO/IEC 15504 – Avaliação de Processo (também conhecida por SPICE: *Software Process Improvement and Capability Determination*) e seu Modelo de Avaliação de Processo de Software ISO/IEC 15504-5, portanto o modelo está totalmente aderente a essas normas. O MPS.BR também cobre o conteúdo do CMMI-SE/SWSM, através da inclusão de processos e resultados de processos em relação aos processos da Norma NBR ISO/IEC 12207 (MPS.BR, 2005).

O MPS.BR baseia-se nos conceitos de maturidade e capacidade de processo para a avaliação e melhoria da qualidade e produtividade de produtos de software e serviços correlatos. Dentro desse contexto, o MPS.BR possui três componentes:

Modelo de Referência (MR-MPS);

Método de Avaliação (MA-MPS);

Modelo de Negócio (MN-MPS).

Cada componente do modelo foi descrito através de guias e dos Documentos do Projeto. A **Figura 5** apresenta graficamente os componentes e seus documentos.

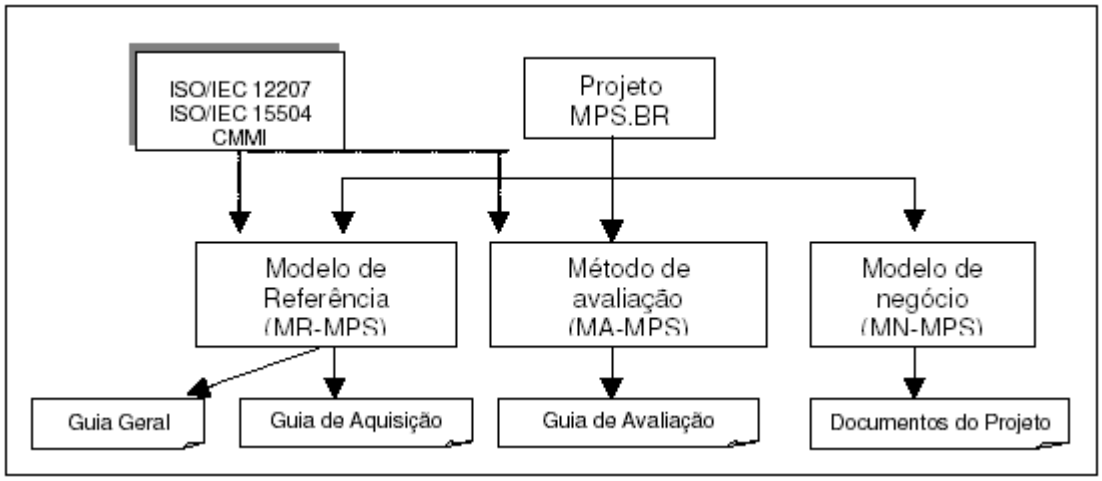

Figura 5: Componentes e documentos do MPS.BR (2005).

2.3.2 Níveis de Maturidade

O MR-MPS define sete níveis de maturidade:

- G - Parcialmente Gerenciado:

  O nível G é composto pelos processos de Gerência de Projeto e Gerência de Requisitos;

- F – Gerenciado:

  O nível F é composto pelo nível de maturidade anterior (G) acrescido dos processos de Gerência de Configuração, Garantia da Qualidade, Medição e Aquisição;

- E - Parcialmente Definido:

  É composto pelo nível de maturidade anterior (F), acrescido dos processos de Treinamento, Definição do Processo Organizacional, Avaliação e Melhoria do Processo Organizacional e Adaptação do Processo para Gerência de Projetos;

- D - Largamente Definido:

  É composto pelo nível de maturidade anterior (E), acrescido dos processos de Desenvolvimento de Requisitos, Solução Técnica, Validação, Verificação, Integração de Produto, Instalação de Produto e Liberação de Produto;

- C - Definido:

  É composto pelo nível de maturidade anterior (D), acrescido dos processos de Gerência de Riscos e Análise de Decisão e Resolução;

- B - Gerenciado Quantitativamente:

  É composto pelo nível de maturidade anterior (C), acrescido dos processos de Desempenho do Processo Organizacional e Gerência Quantitativa de Projeto;

- A - Em Otimização:

  É composto pelo nível de maturidade anterior (B), acrescido dos processos de Inovação e Implantação na Organização e Análise e Resolução de Causas.

Os níveis de maturidade estabelecem patamares de evolução de processos, caracterizando estágios de melhoria de implementação de processos na organização. O nível de maturidade em que se encontra uma organização permite prever seu desempenho futuro em uma ou mais disciplinas.

A escala de maturidade se inicia no nível G e progride até o nível A. Para cada um destes sete níveis de maturidade foi atribuído um perfil de capacidade de processos que indicam onde a organização tem que colocar esforço para melhoria de forma a atender os objetivos de negócio.

As **Tabelas 5, 6, 7, 8, 9, 10 e 11** a seguir apresentam os processos do MPS.BR de forma tabular, visando a uma melhor apresentação do objetivo de cada processo.

Tabela 5: Nível G - Parcialmente Gerenciado

Processo	Sigla	Sigla Nível MR-MPS	Propósito
Gerência de Projetos	GPR	G	O propósito do processo Gerência de projetos é identificar, estabelecer, coordenar e monitorar as atividades, tarefas e recursos que um projeto necessita para produzir um produto e/ou serviço, no contexto dos requisitos e restrições do projeto.
Gerência de Requisitos	GRE	G	O propósito do processo Gerência de Requisitos é gerenciar os requisitos dos produtos e componentes do produto do projeto e identificar inconsistências entre esses requisitos e os planos e produtos de trabalho do projeto.

Fonte: Guia de Referência MPS.BR (2005).

Tabela 6: Nível F – Gerenciado

Processo	Sigla	Sigla Nível MR-MPS	Propósito
Gerência de Configuração	GCO	F	O propósito do processo de Gerência de configuração é estabelecer e manter a integridade de todos os produtos de trabalho de um processo ou projeto e disponibilizá-los a todos os envolvidos.
Garantia da Qualidade	GQA	F	O propósito do processo Garantia da Qualidade é garantir que os produtos de trabalho e processos estão em conformidade com os planos e recursos predefinidos.
Medição	MED	F	O propósito do processo Medição é coletar e analisar os dados relativos aos produtos desenvolvidos e aos processos implementados na organização e em seus projetos, de forma a apoiar os objetivos organizacionais.
Aquisição	AQU	F	O propósito do processo de Aquisição é de se obter um produto e/ou serviço que satisfaça a necessidade expressa pelo cliente.

Fonte: Guia de Referência MPS.BR (2005).

Tabela 7: Nível E - Parcialmente Definido

Processo	Sigla	Sigla Nível MR-MPS	Propósito
Treinamento	TRE	E	O propósito do processo Treinamento é prover a organização e os projetos com profissionais que possuam os conhecimentos e as habilidades necessárias para executar suas funções de forma efetiva.
Definição do Processo Organizacional	DFP	E	O propósito do processo Definição do Processo Organizacional é estabelecer e manter um conjunto de ativos dos processos organizacionais usável e aplicável às necessidades de negócio da organização.
Avaliação e Melhoria do Processo Organizacional	AMP	E	O propósito do processo Avaliação e Melhoria do Processo Organizacional é determinar o quanto os processos-padrão da organização contribuem para a organização alcançar os objetivos de negócio e ajudam a organização a planejar e implementar melhorias contínuas nos processos com base no entendimento de seus pontos fortes e fracos.
Adaptação do Processo para Gerência de Projeto	APG	E	O propósito do processo Adaptação do Processo para Gerência de Projeto é estabelecer e gerenciar o projeto e envolver os interessados relevantes de acordo com o processo definido e integrado que é adaptado do conjunto de processos-padrão da organização.

Fonte: Guia de Referência MPS.BR (2005).

Tabela 8: Nível D - Largamente Definido

Processo	Sigla	Sigla Nível MR-MPS	Propósito
Desenvolvimento de Requisitos	DRE	D	O propósito do Desenvolvimento de Requisitos é estabelecer os requisitos dos componentes do produto, do produto em si e do cliente.
Solução Técnica	STE	D	O propósito da Solução Técnica é projetar, desenvolver e implementar soluções para atender aos requisitos.
Validação	VAL	D	O propósito do processo Validação é confirmar que um produto ou componente do produto atende a seu uso específico pretendido quando colocado no ambiente

			para o qual foi desenvolvido.
Verificação	VER	D	O propósito do processo Verificação é confirmar que cada serviço e/ou produto de trabalho do processo ou do projeto reflete apropriadamente os requisitos especificados.
Integração do Produto	ITP	D	O propósito do processo Integração do Produto é compor os componentes do produto, produzindo um produto integrado consistente com o projeto (*design*), e demonstrar que os requisitos funcionais e não-funcionais são satisfeitos para o ambiente alvo ou equivalente.
Instalação do Produto	ISP	D	O propósito do processo Instalação do Produto é instalar no ambiente alvo o produto ou componente do produto resultante da integração dos componentes do produto que atende aos requisitos acordados.
Liberação do Produto	LIP	D	O propósito do processo Liberação do Produto é entregar um produto ou serviço ao cliente que atenda aos requisitos acordados.

Fonte: Guia de Referência MPS.BR (2005).

Tabela 9: Nível C – Definido

Processo	Sigla	Sigla Nível MR-MPS	Propósito
Gerência de Riscos	GRI	C	O propósito do processo Gerência de Riscos é identificar, gerenciar e reduzir continuamente os riscos nos níveis organizacionais e de projeto.
Análise de Decisão e Resolução	ADR	C	O propósito do processo Análise de Decisão e Resolução é analisar possíveis decisões usando um processo formal de avaliação que avalie as alternativas identificadas em relação a critérios estabelecidos.

Fonte: Guia de Referência MPS.BR (2005).

Tabela 10: Nível B - Gerenciado Quantitativamente

Processo	Sigla	Sigla Nível MR-MPS	Propósito
Desempenho do Processo Organizacional	DEP	B	O propósito do processo Desempenho do Processo Organizacional é estabelecer e manter um entendimento quantitativo do desempenho do conjunto de processos padrão da organização para apoiar os objetivos de qualidade e de desempenho do processo. Também é propósito deste processo fornecer dados, linhas-básicas e modelos para gerenciar quantitativamente os projetos da organização.
Gerência Quantitativa do Projeto	GQP	B	O propósito do processo Gerência Quantitativa do Projeto é gerenciar quantitativamente o processo definido para o projeto de forma a alcançar os objetivos de qualidade e de desempenho do processo estabelecidos para o projeto.

Fonte: Guia de Referência MPS.BR (2005).

Tabela 11: Nível A - Em Otimização

Processo	Sigla	Sigla Nível MR-MPS	Propósito
Inovação e Implantação na Organização	IIO	A	O propósito da área de processo Inovação e Implantação na Organização é selecionar e implantar melhorias incrementais e inovadoras que, de forma mensurada, melhoram os processos e as tecnologias da organização. As melhorias implantadas apóiam os objetivos de qualidade e de desempenho de processo da organização, que são derivados de seus objetivos de negócio.
Análise e Resolução de Causas	ARC	A	O propósito do processo Análise e Resolução de Causas é identificar causas de defeitos e de outros problemas e tomar ações para prevenir suas ocorrências no futuro.

Fonte: Guia de Referência MPS.BR (2005).

2.4 MPEs frente aos modelos de qualidade

Conforme já citado na introdução deste trabalho, as MPEs passam por diversas dificuldades com relação ao desenvolvimento de software. Tais dificuldades podem ser vistas como:

- incompreensão das necessidades do usuário final;

- software difícil de manter e entender;

- baixo desempenho e baixa qualidade;

- recursos humanos atuando nas diversas fases do ciclo de vida do software sem um planejamento de atuação;

- inexistência de processos de rastreabilidade para melhor manutenção de códigos fontes;

- falta de conhecimento em métodos e técnicas de ES.

- investimentos focados em linguagens de programação e desenvolvimento e não em ES;

- inexistência de processos organizados e aplicados;

- recursos humanos e financeiros limitados;

- baixo nível de conhecimento no domínio gerencial;

- inexistência de políticas de curto ou longo prazo que auxiliem na melhoria dos processos de desenvolvimento Jones (1996) e Yourdon (1997).

Neste contexto, há uma dificuldade da criação e implantação de um processo sistemático, seqüencial e organizado que possa suprir as necessidades da MPEs no contexto de desenvolvimento de software.

Diante disso, os modelos apresentados com abordagem de qualidade organizacional baseados em níveis de certificação podem inviabilizar a implantação nessas MPEs. A falta de recursos e de conhecimentos em métodos de gerenciamento dificultam a implantação desses níveis sugeridos pelos modelos de qualidade. Além de demandarem maior tempo de implantação do que sugerem outras abordagens, como por exemplo, a adoção de melhores práticas de desenvolvimento de software.

A apresentação da taxonomia exposta nesse trabalho pretende preencher essa lacuna, esclarecendo para as empresas que sua melhoria de qualidade e viabilização de produção pode passar por processos ou passos mais curtos até às certificações oficiais disponíveis atualmente no mercado.

A escolha de elementos da taxonomia criada nesse trabalho, poderá auxiliar as MPEs em uma conduta no desenvolvimento de software, que poderá aumentar a qualidade da produção e criar uma cultura de uso de métodos e técnicas que serão úteis no momento da decisão de almejarem algum tipo de certificação.

3 Ontologias

3.1 Uma visão filosófica e conceitual

Há tempos o homem procura caminhos para explicar as coisas, os seres e as relações destes com o universo e com a vida, a partir do estudo lógico, das ciências naturais e da filosofia *metafísica* (Russel, 1995). No domínio da Filosofia, no século XVIII, na Grécia antiga, *Aristóteles* (384-322 a.C.) busca o entendimento a partir do estudo da essência das coisas e do estabelecimento de sistemas de categorias que podem servir como complemento para se explicar as coisas no mundo. Esses complementos seriam classificados por ele como: *substância, qualidade, quantidade, relação, ação, lugar e tempo*. Assim, poderia categorizar as coisas e também dar complemento para o seu entendimento. Ao dizermos que algum objeto está sobre uma mesa, estamos nos referenciando ao objeto classificando-o na categoria de *lugar*, enquanto ao dizermos que este objeto é cinza, ele é referenciado sendo classificado na categoria de *qualidade*.

Para Smith (2000), o termo *Ontologia* é um sistema particular de categorias que não depende de uma linguagem para uma definição das coisas presentes do mundo. Podendo ser visto como um sinônimo de *metafísica* e que tem como propósito classificar as entidades de uma porção da realidade, definindo seu vocabulário e as formulações canônicas de suas teorias.

Immanuel Kant (1724-1804) define que a essência das coisas é determinada não só pelas coisas em si próprias, mas também pela contribuição de quem as percebe e compreende (Calero et al., 2006). De acordo com Kant, uma questão-chave é "*quais estruturas nossa mente utiliza para captar a realidade?*". A resposta a esta pergunta leva a categorização proposta por Kant (Calero et al., 2006). A categorização de Kant propõe uma estrutura inicial de termos em que cada um possui uma estrutura de três categorias. Os termos que definem as coisas são: quantidade, qualidade, relação e modalidade. E cada termo apresenta sua categorização:

- quantidade: unidade, pluralidade e totalidade;
- qualidade: realidade, negação e limitação;

- relação: inerência, do nexo de causalidade e comunidade;

- modalidade: possibilidade, existência e necessidade (Calero et al., 2006).

Em (Calero et al., 2006) ontologia é definida como: *uma especificação formal e explícita de uma conceituação compartilhada*. Conceituação refere-se a um modelo de algum fenômeno no mundo identificado pelos conceitos desse fenômeno. Explícito significa que o tipo de conceito utilizado e as limitações à sua utilização, são explicitamente definidas. Formal se refere ao fato de que a ontologia deve ser legível por máquina. Compartilhada remete à noção de que uma ontologia captura o conhecimento consensual, isto é, não é privativa de alguns indivíduos, mas aceito por um grupo (Calero et al., 2006).

Tanto Gruber (1995) quanto Guarino (1998) descrevem que uma ontologia é uma especificação de uma conceituação, mas Guarino estende a definição dizendo que uma ontologia é na verdade uma especificação parcial e explícita que tenta, da melhor forma possível, aproximar a estrutura de mundo definida por uma conceituação. De maneira geral, uma ontologia define os conceitos e relações entre seus elementos a partir de determinada parte do domínio e não tenta esgotar todas as possibilidades a respeito daquele domínio.

Na comunidade da computação, em concordância com Guarino (1998), as ontologias são tratadas como um artefato computacional composto de um vocabulário de conceitos, suas definições e suas possíveis propriedades. Além de um modelo gráfico mostrando todas as possíveis relações entre os conceitos e um conjunto de axiomas formais que restringem a interpretação dos conceitos e relações, representando de maneira clara e não ambígua o conhecimento do domínio. Noy e Hafner (1997) sugerem o conceito como um artefato de engenharia, com um vocabulário de termos organizados em uma taxonomia, contendo suas definições, e um conjunto de axiomas formais usados para criar novas relações e para restringir as suas interpretações segundo um sentido pretendido.

Em (Falbo et al., 2004) considera-se que o modelo de domínio descrito por uma ontologia pode ser usado como uma estrutura unificadora para dar semântica e uma representação comum à informação, desta forma as ontologias oferecem meios para representar os recursos de informação.

De forma geral, a ontologia elimina confusões conceituais e terminológicas, fornecendo uma representação de um vocabulário especializado para o processo de software (Chandrasekaran et al., 1999). As ontologias permitem o compartilhamento

de conhecimento a partir da unificação de termos e concepções. O uso da ontologia para representar processos ou modelos de processos através de descrições formais para os mesmos, irá auxiliar o raciocínio lógico para a análise desses modelos (Chandrasekaran et al., 1999).

Os principais motivos para se desenvolver ontologias são:

- Compartilhar conhecimento e estruturas de informação entre pessoas e agentes de software;
- Reutilização do conhecimento de um domínio;
- Explicitar hipóteses;
- Separar conhecimento de um domínio do conhecimento operacional.

Os mais relevantes benefícios da utilização de ontologias serão encontrados em conceitos de reuso, que possibilita o compartilhamento de bases de ontologias para o desenvolvimento de sistemas; de produtos de ontologias prontos, disponibilidade de ontologias prontas, como em bibliotecas, que permitiriam a comunicação entre agentes; tradução e formalismo de representação, que permitiria a tradução da ontologia entre linguagens que podem ser utilizadas por diferentes agentes; utilização de editores de ontologias como o Protégé-2000 (Noy, 1997), onde pode-se gerar ontologias em Prolog, XML (*eXtensible Markup Language*), Jess 2, RDF (*Resource Description Framework*) e OIL (*Ontology Interface Layer*); benefícios de construção de bases para acesso *on-line* a servidores de ontologias que poderiam ser capazes de armazenar classes e instâncias, formando uma base uniforme e compartilhada entre empresas e grupos de pesquisa.

3.2 Ciclo de Vida das Ontologias

Um dos mais interessantes artigos investigados sobre ontologia nesse trabalho de pesquisa, foi o que sugere a publicação de título: *Ontologias: representando a pesquisa na área através de mapa conceitual* (Campos et al., 2007). Os autores objetivaram o mapeamento dos processos que envolvem a construção e definição de ontologias e o representaram através de um mapa conceitual. O mapa representa os aspectos ligados a cada processo do ciclo de vida das ontologias, a saber: criação, manutenção e uso além de outros aspectos relacionados à ontologia como tema geral (Campos et al., 2007).

A **Figura 6** apresenta a proposta dos autores com relação aos aspectos relacionados ao ciclo de vida de uma ontologia.

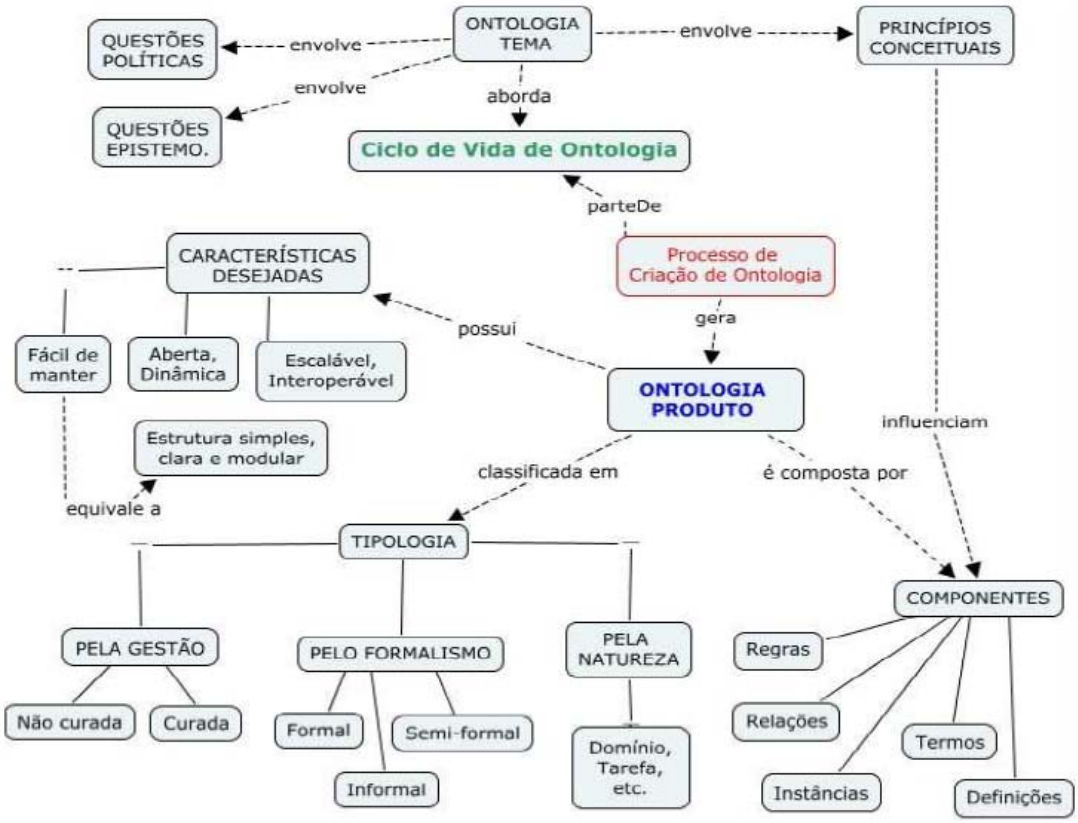

Figura 6: Aspectos gerais relacionados às ontologias (Campos et al., 2007).

A **Figura 7** apresenta o processo de criação de um ontologia, envolvendo os sub-processos de levantamento, as características de formas de criação e dos princípios de concepção (Campos et al., 2007).

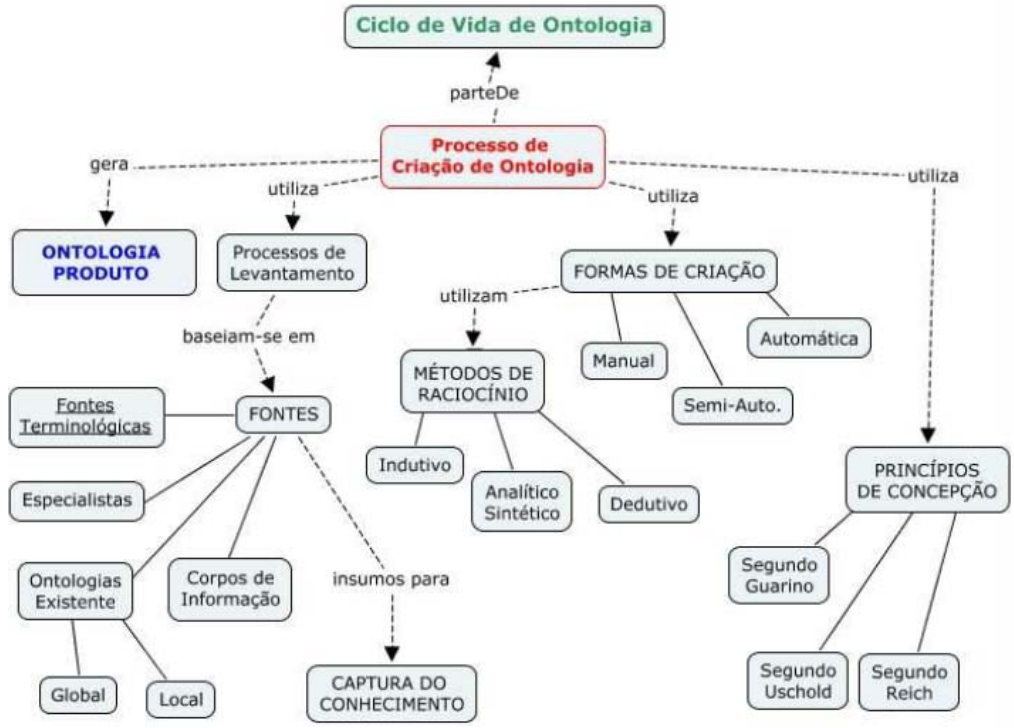

Figura 7: Processo de Criação de Ontologias (Campos et al., 2007).

A **Figura 8** apresenta o ciclo de manutenção de uma ontologia. Perpassam por esse ciclo sub-processos como versionamento, que envolve questões relacionadas à aquisição, alteração e exclusão de conceitos, bem como a gerência e controle das versões atualizadas de uma ontologia sobre as suas versões anteriores; avaliação, que envolve o estabelecimento e a aplicação de critérios e métricas de qualidade de modo a aferir e garantir a qualidade da ontologia gerada. Além dos dois conceitos apresentados, a figura ainda apresenta os diversos mecanismos de reuso de ontologias. Segundo os autores (Campos, et al. 2007), o processo de manutenção vai tratar das modificações da ontologia ao longo do seu ciclo de vida e para isso o processo está diretamente ligado a mecanismos de reuso tais como: mapeamento, alinhamento, junção e integração.

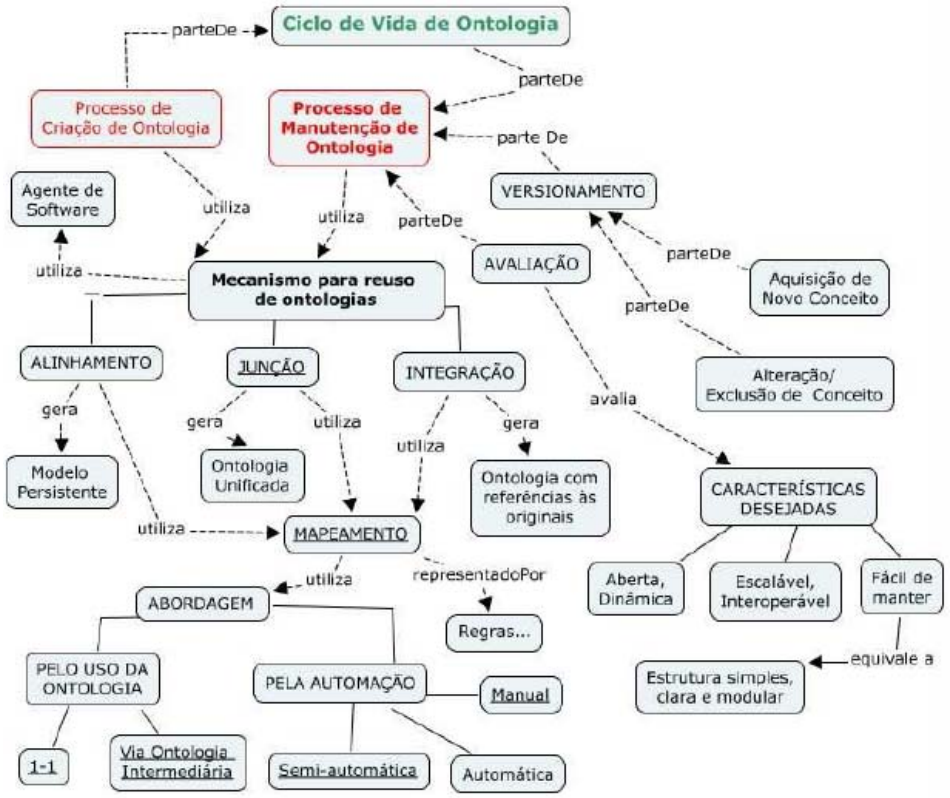

Figura 8: Processo de Manutenção de Ontologias (Campos et al., 2007).

A **Figura 9** apresenta o mapeamento do processo de uso de uma ontologia. Nele estão representados: o envolvimento de agentes de software, que são utilizados em uma ontologia para descrever serviços, recursos ou dados a serem integrados; a de integração de sistemas e de base de dados; e também a descrição e recuperação de serviços e recursos. Neste último caso, os autores citam como exemplo a descoberta de serviços web ou a anotação de seqüências “genômicas” (Campos et al., 2007).

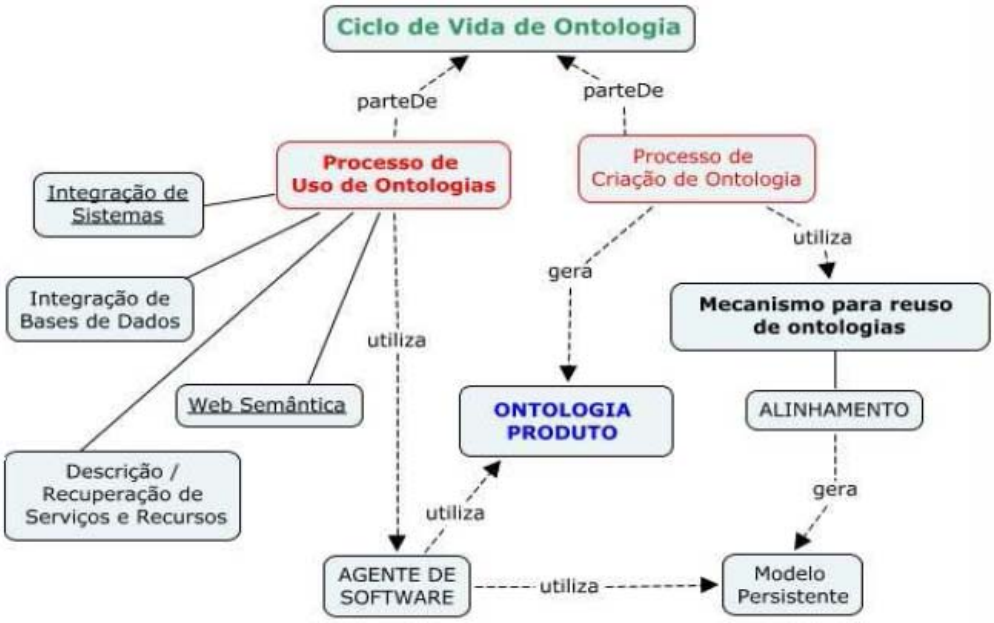

Figura 9: Processo de Uso de Ontologias (Campos et al., 2007).

3.3 Tipos de Ontologias

De acordo com Gomez-Perez (1999) existem diferentes tipos de ontologias:

• Ontologias de Representação: definem as primitivas de representação – frames, axiomas, atributos e outros – de forma declarativa;

• Ontologias Gerais: trazem definições abstratas necessárias para a compreensão de aspectos do mundo, como tempo, processos, papéis, espaço, seres, coisas, entre outros;

• Ontologias Centrais ou Genéricas: definem os ramos de estudos de uma área e/ou conceitos mais genéricos e abstratos desta área.

• Ontologias de Domínio: tratam de um domínio mais específico de uma área genérica de conhecimento, como ES, direito tributário, microbiologia, entre outros domínios;

• Ontologias de Aplicação: procuram solucionar um problema específico de um domínio, como identificar doenças do coração, a partir de uma ontologia de domínio de cardiologia.

Guarino defende um nível mais genérico para os tipos de ontologia (Calero et al., 2006), relaciona esses tipos da seguinte forma:

• Ontologias de Alto Nível: descrevem conceitos gerais, como espaço, tempo, material, objeto. Eles são independentes de um domínio específico ou problema. O seu objetivo é unificar critérios entre as grandes comunidades de usuários;

• Ontologias de Domínio: descrevem o vocabulário relacionado a um domínio genérico (por exemplo, sistemas de informação ou medicamentos), por meio da introdução de conceitos especializados às ontologias de alto nível;

• Ontologias de Tarefa: descrevem o vocabulário relacionado com uma tarefa ou atividade genérica (por exemplo, de desenvolvimento ou de vendas), por meio da introdução de conceitos especializados às ontologias de alto nível;

• Ontologias de Aplicação: descrevem conceitos pertencentes ao mesmo tempo a um domínio e uma tarefa, por meio da especialização dos conceitos de domínio e ontologias tarefa. Eles geralmente correspondem a papel desempenhado pelas entidades de domínio que executam uma atividade.

Algumas alternativas para a classificação de ontologias podem ser vistas também no que estabelece Fensel (2004):

• Ontologias Genéricas ou de Senso Comum: Permitem a captação de conhecimentos gerais do mundo. Elas fornecem noções básicas e conceitos de espaço, tempo, estado, eventos, e são válidas para uma grande variedade de domínios.

• Ontologias de Representação: não pertence a nenhum domínio particular. Oferecem entidades sem estabelecer o que eles podem representar. Por conseguinte, definem conceitos que expressam o conhecimento a partir da abordagem orientada por aproximação.

• Ontologias de Domínio: capturam o conhecimento válido para um determinado tipo de domínio (por exemplo, eletrônica, medicina, entre outros domínios).

• Ontologias de Método e Tarefa: O autor oferece uma terminologia específica para métodos de resolução dos problemas, enquanto a última provê termos para especificar tarefas. Ambas oferecem um ponto de vista razoável para o conhecimento de domínios.

As classificações de Guarino (1998) e Fensel (2004) são alinhadas de acordo com o modelo da **Figura 10** a seguir, sugerido por Calero et al.(2006):

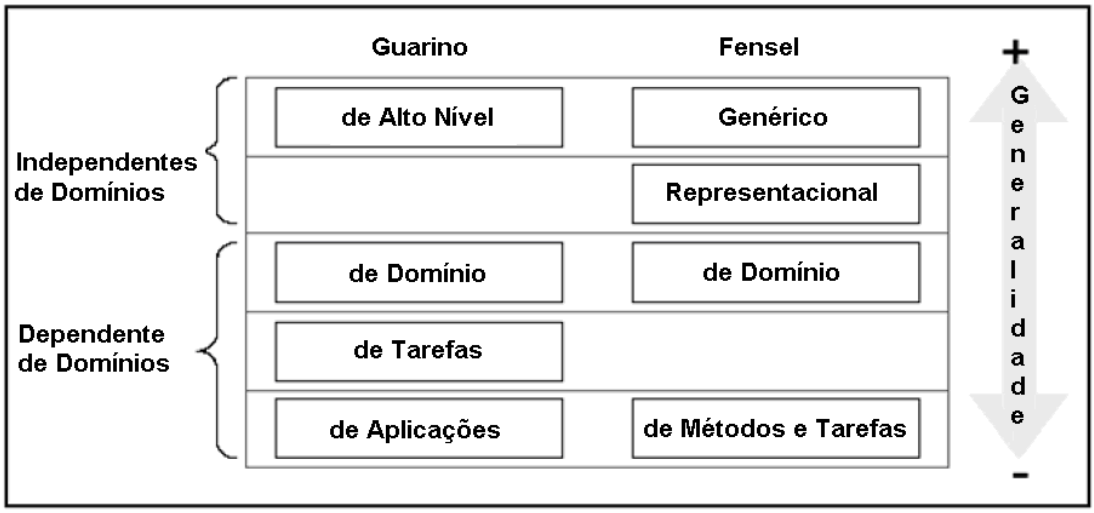

Figura 10: Alinhamento de conceitos domínios.
Adaptado de Guarino e Fensel et al. (2006).

Uma ontologia pode assumir uma variedade de formas, mas necessariamente irá incluir um vocabulário de termos, e alguma especificação de seu significado. Isto inclui definições e uma indicação de como conceitos estão interrelacionados que coletivamente impor uma estrutura sobre o domínio e restringe as possíveis interpretações dos termos (Calero et al., 2006).

Gomez-Perez (1999) conclui que o objetivo de ontologias é o conhecimento consensual de uma forma genérica e que este pode ser reutilizado e compartilhado entre aplicações de software e por grupos de pessoas.

3.4 Princípios para a Construção de Ontologias

Alguns princípios básicos são necessários para a construção de uma ontologia consistente. A seguir são relatados alguns critérios propostos por Gruber (1993) que devem ser seguidos no momento da construção de uma ontologia:

• Clareza: os termos utilizados na ontologia devem transmitir seu significado de forma objetiva, assim como suas definições. É importante que as definições sejam documentadas em linguagem natural. As definições devem declaradas como regra formal e uma definição completa. Gruber (1993) defende que uma definição deve ser formalizada por completo e não somente o necessário ou o que for suficiente.

• Nível Mínimo de Codificação: A conceituação deve ser centrada no nível do conhecimento, sem depender de um símbolo especial do nível de codificação. Codificação deve ser substituída pela documentação do conhecimento que poderá ser

compartilhada para diferentes agentes, que poderão implementar de diferentes formas.

• Evitar Revisões: é importante a definição de novos termos baseados no vocabulário existente, evitando a revisão de definições existentes.

• Coerência: uma ontologia deve ser coerente: que é, deve permitir que sejam feitas avaliações ou análises consistentes com as definições. Gruber (1993) cita um exemplo, [...] *Se uma frase que pode ser inferida a partir de axiomas contradiz uma definição ou exemplo dado informalmente, então a ontologia é incoerente.*

• Ontologia Confiável: a ontologia deve ser baseada no uso consistente de seu vocabulário. A confiança na ontologia é reduzida quando a sua teoria é insuficiente ou são definidos somente alguns termos para estabelecer a comunicação do conhecimento com a teoria.

3.5 Trabalhos correlatos: ontologias para ES

No domínio da ES a ontologia é utilizada para registrar o conhecimento e os dados a fim de se permitir o estabelecimento de relações entre diversos termos e assuntos. Desta forma, busca-se uma maneira mais eficiente de tomar decisões baseadas em contextos padronizados e conceituados.

Várias são as pesquisas e os trabalhos utilizando a ontologia como apoio no desenvolvimento de software no domínio de ES, como exemplo pode-se citar:

- *Uma Ontologia para o Domínio de Qualidade de Software com Foco em Produtos e Processos de Software* (Moro, 2008): estabelece uma Ontologia, a partir de linguagem gráfica e axiomas, para definição de conceitos e relações dos termos e assuntos que envolvem o domínio de qualidade de software.

- *Definição de Ontologia para Identificação de Riscos de Projetos de Software* (Costa, 2008): o artigo apresentada uma proposta de auxilio para o processo de gerenciamento de riscos através de um vocabulário de uma Ontologia de riscos em ambientes de desenvolvimento de software. A Ontologia é definida pelos autores como *mPRIME Ontology*.

- *Geração de Ontologias subsidiada pela Engenharia de Requisitos* (Felicíssimo et al., 2003): que discute sobre uma ferramenta amparada pela Engenharia de Requisitos para a definição de ontologias.

Outra importante publicação na área de ontologia é o livro *Ontologies for Software Engineering and Software Technology* dos autores Calero et al. (2006). O

livro trata de uma vasta investigação sobre o assunto de ontologias nos domínios de ES e de tecnologia de software. Nele estão reunidos conceitos e discussões a respeito de: princípios, métodos, ferramentas e linguagens na engenharia de ontologias, o uso de ontologias na ES e tecnologia, discute a cerca da engenharia de ontologias no *Software Engineering Body of Knowledge* (SWEBOK*)* (ABRAN et al., 2004):

- questões e técnicas, relata questões a respeito de ontologias para o desenvolvimento e manutenção de software;

- retrata discussões sobre uma ontologia para medição de software, discrimina as discussões investigadas sobre ontologia para SQL;

- relata sobre as pesquisas da OMG (*Object Management Group*) acerca de definições de metamodelos em ontologia;

- relata sobre o uso de ontologias em ambientes de desenvolvimento de software, entre outras discussões. Além disso, o livro traz consigo um registro da investigação sobre as principais publicações em diversos contextos, chamados pelos autores de categorias e subcategorias no domínio de ES e alguns de seus subdomínios. A **Tabela 12** apresenta as publicações relatadas no livro sobre propostas de ontologias de domínio.

Tabela 12: Propostas de ontologias de domínio**.**

Categoria / Subcategoria	Proposta	Referências
Genérica	Resultados no desenvolvimento de ontologias para a disciplina de engenharia.	Mendes e Abran (Mendes, 2005)
	A avaliação da representação ontológica do SWEBOK como uma ferramenta de revisão.	Sicilia et al. (Sicília et al., 2005)
	OntoGLOSE: ontologia para engenharia de software.	Hilera et al. (Hilera et al., 2005)
Específica / Requisitos de Software	Projeto conceitual baseado em modelo de análise de requisitos no *Win-Win Framework* para engenharia de requisitos concorrentes.	Bose (Bose, 1995)
	Uma ontologia genérica para a especificação de ontologias de domínio.	Girardi e Faria (Girardi, 2003)
	Uma ontologia sobre ontologias e modelos: uma discussão conceitual.	Sánchez et al. (Sánchez et al., 2005)
	OpenCyc.org: conhecimento comum formalizado.	Cyc (Cyc, 2009)
Específica / Projeto de Software	Uma ontologia sobre ontologias e modelos: uma discussão conceitual.	Sánchez et al. (Sánchez et al., 2005)
	OpenCyc.org: conhecimento comum formalizado.	Cyc (Cyc, 2009)
	XCM: uma ontologia de componentes.	Tansalarak e Claypool (Tansalarak, 2004)
Específica / Manutenção de Software	Uma abordagem conceitual orientada para o apoio às atividades de manutenção e reutilização de software.	Deridder (Deridder, 2002)
	Organizando o conhecimento usado em manutenção de software.	Dias et al. (Dias et al., 2003)
	Uma ontologia para a gestão de projetos de manutenção de software.	Ruiz et al. (Ruiz et al., 2004)
	Mesclando ontologias de manutenção de software: a nossa experiência	Vizcaino et al. (Vizcaino et al., 2005)
	Sobre uma ontologia para manutenção de software.	Kitchenham et al. (Kitchenham et al., 1999)
Específica / Qualidade de Software	Identificando conflitos de qualidade de requisitos.	Boehm (Boehm, 1996)
	Uma abordagem ontológica para engenharia de domínio.	Falbo et al. (Falbo et al., 2003)
Específica / Processos de ES	Usando ontologias para melhorar a integração do conhecimento em ambientes de ES.	Falbo et al. (Falbo et al., 1992)
	Direcionando a implementação de uma ferramenta para apoiar o processo de desenvolvimento de software.	Larburu et al. (Larburu et al., 2003)
	Uma ontologia para desenvolvimento de software.	González-Pérez e Henderson-Sellers (González-Pérez, 2006)
	Construindo uma base de conhecimento do IEEE/EAI 12207 e CMMI com ontologia.	Lin et al. (Lin et al., 2003)
Específica / Gerência em ES	REFSENO: um formalismo para a representação de ontologias em ES.	Tautz e Greese (Tautz, 1998)

Fonte: Elaborado pelo autor, adaptado de (Calero et al., 2006).

Em ES pode-se encontrar os chamados *Software Engineering Environments* (SEEs) que reúnem em um ambiente integrado diversas ferramentas para apoiar engenheiros no processo de ES. Apesar de eficientes esses ambientes só permitem a extração do conhecimento a partir de ferramentas e assistentes próprios que dificultam o seu compartilhamento ou o reuso (Calero et al., 2006). Por essa razão

existem algumas propostas de integrar esses SEEs a soluções baseadas em ontologias. Duas dessas propostas podem ser vistas a seguir:

MANTIS Environment: o Mantis leva o conceito de ambiente de ES estendido ou "*eXtended Software Engineering Environment*" no qual o uso da nomenclatura "*extended SEE*" dá ênfase à idéia de integração e aumento dos conceitos da metodologia de SEEs (Calero et al., 2006). O Mantis considera ferramentas para três categorias: conceitual, metodológico e técnico. Nos três níveis estão divididas sub-ontologias para delinear conceitos para amparar cada um dos processos propostos pelo ambiente. As sub-ontologias podem ser vistas da seguinte forma: inicialmente para tratar conceitualmente de questões dos produtos gerados, das atividades, dos processos relacionados e para descrever os diferentes agentes envolvidos no processo de manutenção de software; em outro patamar encontramos o que é chamado *Workflow Ontology* que possui três aspectos relevantes para tratar de definição e manutenção de processos: decomposição das atividades, a relação temporária entre as atividades e o controle da execução das atividades e projetos no momento da definição do processo; um outro nível chamado de *Measure Ontology* refere-se a aspectos estáticos e dinâmicos relacionados ao processo de medição de software (Calero et al., 2006).

TABA Workstation (Rocha et al., 1990): é capaz de gerar, por meio de instanciação, SEEs específicos adequados para particularidades de processos de software, aplicações de domínio ou de um projeto específico. Esse sistema permite o apoio da aquisição, organização, reutilização e compartilhamento de conhecimento de processos de software (Calero et al., 2006).

Outra contribuição para o meio científico e industrial de software é a construção e aplicação de ontologias considerando um grupo de boas práticas de desenvolvimento de software. As pesquisas mostram que o campo ainda é pouco explorado no que diz respeito ao relacionamento de um conjunto de práticas em uma ontologia que poderia auxiliar empresas e tomadores de decisão na aplicação correta de práticas de ES diante dos problemas do cotidiano da empresa. Talvez por ser um campo abrangente, os esforços de pesquisas estejam mais concentrados em pontos isolados das boas práticas, como podemos ver nos exemplos de publicações nacionais e internacionais citadas anteriormente.

4 TAXONOMIA DE BOAS PRÁTICAS EM DESENVOLVIMENTO DE SOFTWARE

Taxonomia é uma estrutura que possibilita classificar organismos vivos, produtos ou livros em grupos hierarquicamente organizados por série para facilitar a sua identificação, estudo e localização distribuindo-os em conjuntos ou classes. *Karl Von Linné* usou este conceito em 1735, portanto o termo não é novo. Ele criou um tipo de classificação que dividia em grupo os seres vivos, hierarquicamente, baseado em suas características em comum. Pode-se dizer que quase tudo pode ser classificado de acordo com algum esquema taxonômico (Terra e Michely, 2004).

Segundo Terra e Michely (2004), um dos objetivos é classificar a informação, de uma forma hierárquica, de maneira que seja facilitado o acesso a ela, melhorando a comunicação entre os principais usuários, quer entre especialistas, quer entre um público qualquer.

Apesar do foco deste trabalho de pesquisa estar concentrado em taxonomias, na seção 3 foi descriminado os conceitos a respeito de ontologias, essa detalhamento foi de extrema importância para que pudessem ser apresentadas e discutidas formas de relacionar elementos em uma visão ontológica e aproveitar o conhecimento adquirido para melhorar a construção da taxonomia proposta. Dessa forma, além de classificação dos elementos foram incorporadas à taxonomia de boas práticas de desenvolvimento de software questões de relação entre seus elementos baseados no estudo da ontologia apresentada.

4.1 Metodologia

A realização do trabalho de elaboração da taxonomia foi composta de atividades. A **Figura 11** apresenta os passos executados até a finalização do estudo de caso com a proposta de adoção da taxonomia.

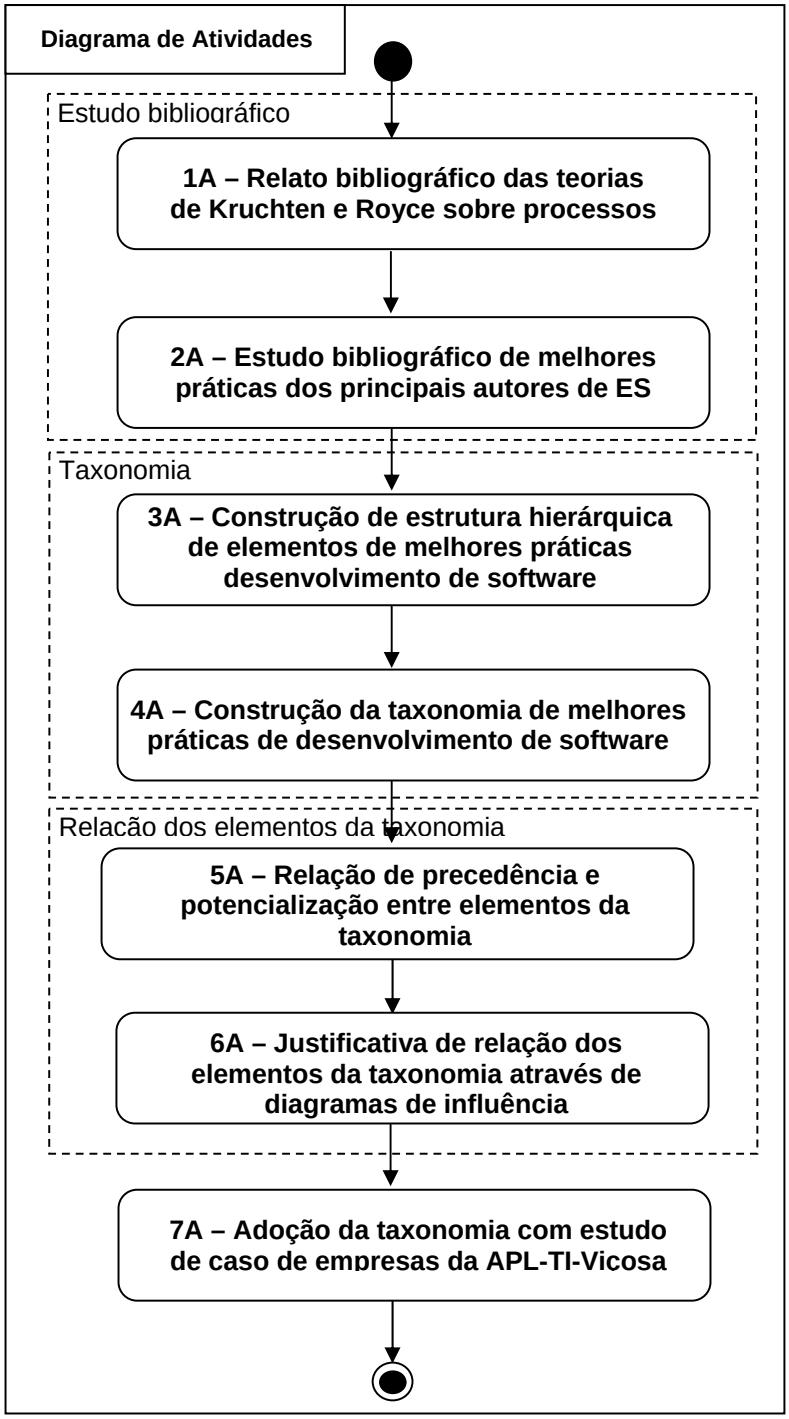

Figura 11: Diagrama de atividades dos passos da criação da taxonomia.

4.2 Elementos para composição da taxonomia

Esse trabalho de pesquisa está direcionado à investigação da construção e aplicação de uma taxonomia de boas práticas que esteja aderente à realidade de MPEs. A partir desse contexto, foi iniciado um trabalho de levantamento sobre boas práticas de

desenvolvimento de software em ES e verificou-se que organizações bem sucedidas adotam essas práticas com a intenção de atuar de forma eficaz na predição de problemas com a produção de software. Essas práticas, denominadas boas práticas, reduzem as chances de insucesso nos projetos, pois tratam as causas dos problemas. Kruchten (2001) propõe as seguintes práticas:

1) Desenvolvimento Iterativo: utilizado para esclarecer mais cedo os desentendimentos com relação a requisitos, encoraja o retorno do usuário com relação aos requisitos levantados, permite que a equipe trabalhe em um nível de abstração mais adequado, facilita a elaboração e a aplicação de testes contínuos e iterativos permitindo a avaliação objetiva de status do projeto, auxilia na correção das inconsistências entre requisitos, projetos e implementação, permite a distribuição da carga de tarefas da equipe nas devidas fases do projeto, há um aprendizado progressivo da equipe com uma carga menor e mais objetiva de tarefas executadas distribuídas em diferentes fases.
2) Gerência de Requisitos: é possível melhorar a comunicação entre as partes envolvidas, uma vez que ela é feita com base em requisitos definidos e registrados. Os requisitos podem ser priorizados, filtrados e rastreados, é possível a avaliação objetiva das funcionalidades do projeto e do desempenho desejado, as inconsistências são encontradas em fases preliminares e corrigidas mais cedo. Com suporte adequado de ferramentas, é possível ter um registro centralizado de requisitos, atributos e rastreabilidade do sistema, com o suporte de documentação adequada e disponível.
3) Arquitetura Baseada em Componentes: auxilia na organização do software, na seleção de elementos estruturais que compõem o sistema e suas interfaces, permite definir o comportamento dos elementos estruturais especificados nas suas colaborações, permite a reutilização dos componentes em estruturas maiores. Além disso, os componentes ajudam a obter arquiteturas mais estáveis, a modularidade facilita a focalização em elementos mais sujeitos a mudança, os componentes são uma base natural para gestão de configurações.
4) Modelagem Visual: permite com que casos de uso caracterizem os cenários e os modelos do projeto de software sem ambigüidades,

auxiliando na percepção de inconsistências de forma mais eficaz, permite que alguns detalhes fiquem escondidos quando necessário, permite garantir de forma preliminar a qualidade da aplicação, pois, facilita no desenvolvimento do projeto. E com ferramentas de modelagem visual pode-se melhor trabalhar os conceitos da *Unified Modeling Language* (UML).

5) Verificação Contínua de Qualidade: permite a avaliação de status do projeto uma vez que é feita objetivamente com base em resultados dos testes reais, e não documentos em papel, permite a correção de inconsistências em requisitos, projeto e implementação. Nessa atividade o foco dos testes se baseia em áreas de maior risco, minimizando as chances de ocorrências de problemas. Os defeitos são identificados mais cedo, trabalhando de forma eficaz na redução do custo de correção. A utilização de ferramentas automáticas de testes permite testar as funcionalidades, a confiabilidade e o desempenho do sistema produzido.
6) Gestão de Alterações: deverá ser pré-definida utilizando-se de mecanismos que descrevam o fluxo de trabalho em caso de mudança de requisitos. A requisição formalizada de mudanças facilita a comunicação e o entendimento do que se deve ser implementado. As equipes devem trabalhar com espaços bem definidos e isolados, reduzindo a interferência entre equipes trabalhando em paralelo na execução da linha base do projeto. Devem ser utilizadas de medições estatísticas para identificar a freqüência das solicitações de alterações com o objetivo de se identificar o status do projeto e o motivo das mudanças, para que seja possível controlar a essas solicitações.

As boas práticas descritas por Kruchten (2001), são baseadas na proposta do processo moderno sugerido por Royce (1998), que propõe cinco princípios para o processo moderno de desenvolvimento de software. A **Figura 12** apresenta esses princípios:

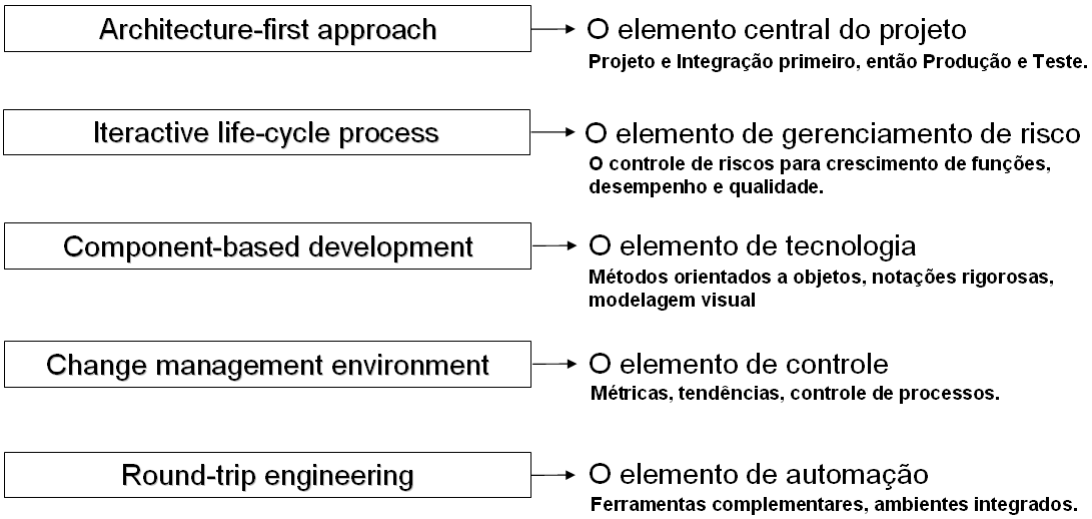

Figura 12: Princípios de um processo moderno de software (Royce, 1998).

1) *Architecture-first approach* (o elemento central do projeto): nesse princípio é importante o foco dos esforços no projeto de implementação e de integração do software e, só após isso, os esforços devem ser concentrados na produção e nos testes.
2) *Iterative life-cycle process* (o elemento de gerenciamento de risco): a abordagem nesse princípio deve ser concentrada no controle dos riscos a fim de suportar o crescimento das funcionalidades do software, suportar o crescimento e garantir o desempenho e a qualidade.
3) *Component-based development* (o elemento de tecnologia): este princípio sugere a adoção de modelos formais de documentação, de modelagem visual e métodos orientados a objetos para representar os requisitos que estão sendo especificados.
4) *Change management environment* (o elemento de controle): um processo de software moderno deve estar amparado por métricas, por indicadores de tendência que permita a tomada de decisão para a correção de desvios do planejado e de um controle organizado de todas as etapas do processo de desenvolvimento.
5) *Round-trip engineering* (o elemento da automação): no processo moderno, há a necessidade de se amparar o processo com ferramentas complementares de apoio ao desenvolvimento, o controle do projeto e a utilização de ambientes integrados.

Diante da apresentação das boas práticas propostas por Kruchten (2001) ou dos princípios de Royce (1998), foram investigados na literatura técnica quais seriam os elementos que poderiam compor inicialmente e de forma organizada uma estrutura hierárquica que representasse técnicas, artefatos, ferramentas e métodos que pudessem dar amparo a aplicação das boas práticas para o desenvolvimento de software em MPEs. Os elementos foram escolhidos com base na sua facilidade de implementação e aderência às características das MPES, uma vez identificados esses elementos não significa que toda a sua composição tenha que ser aplicado na empresa. Devem ser escolhidos os elementos ou o conjunto de sua relação conforme for mais adequada à realidade da empresa que estiver utilizando dessa hierarquia de componentes.

A estrutura sugerida de elementos pode ser vista conforme é apresentada na **Figura 13** e que foi montada a partir estudos de propostas de autores de ES como Pfleeger (1998), Pressman (2005), Sommerville (2003), Wazlawick (2004), Pádua Filho (2003), Yourdon (1997), Dennis & Wixom (2005) e também da literatura especializada em gerenciamento de projetos baseado no PMI (2004). A estrutura não pretende esgotar as possibilidades apresentadas entre as diversas ferramentas, técnicas e métodos de ES, mas serve como um direcionamento para o pensamento sistematizado a respeito do domínio aqui apresentado.

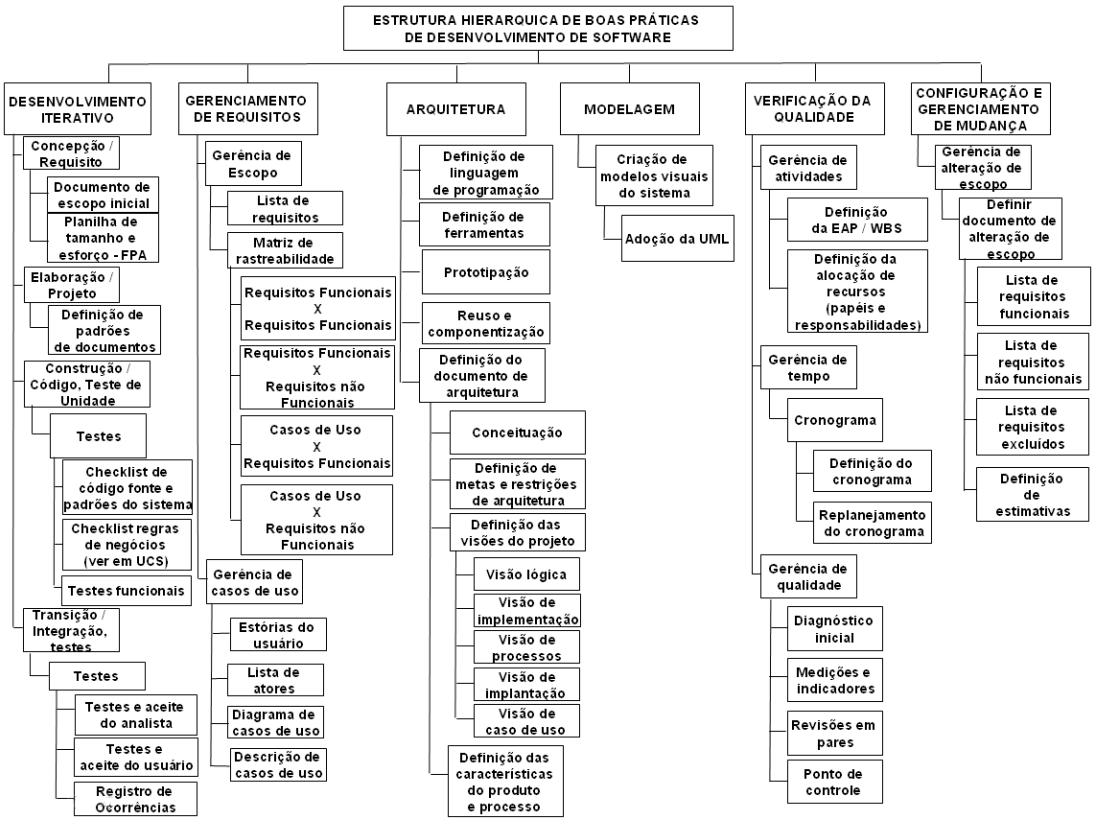

Figura 13: Elementos para apoio às boas práticas de desenvolvimento de software.
Fonte: do autor.

Após o levantamento dos elementos básicos de apoio a uma estrutura taxonômica, seria possível iniciar a construção da taxonomia de boas práticas da ES para desenvolvimento de software e compor a relação de seus elementos.

4.3 Taxonomia de boas práticas

A criação da taxonomia de boas práticas em ES para o "ciclo" de software teve como amparo a análise dos artigos de DING e FOO (2002a), (2002b), a análise do mapa conceitual proposto para o ciclo de vida da ontologia estabelecido por Campos et al. (2007) e a análise da metodologia proposta por Campos (2006).

Apoiado pela metodologia e pelos conhecimentos adquiridos pela leitura técnica especializada sobre o assunto, como por exemplo: (Guarino, 1998), (Gomez-Perez, 1999), (Gruber, 1993), (Cyc, 2009), (Ruiz, 2004), (Falbo et al., 2003), (Falbo et al., 1992), (González-Pérez, 2006), foi possível iniciar o processo de criação do modelo da taxonomia. O modelo teve por base os elementos organizados hierarquicamente conforme proposto em seções anteriores e a partir desses elementos foi conduzida a organização e a relação entre os mesmos.

Para o modelo da taxonomia foi utilizada a seguinte proposta de diagramação:

1) as relações do tipo gênero/espécie foram relacionadas através de linhas continuas sem qualquer tipo de identificação (Campos et al., 2007);

2) as linhas tracejadas foram utilizadas para compor a identificação das relações entre os elementos. E para enriquecer o entendimento das relações, as linhas foram rotuladas com expressões do tipo: *fazParteDe, representadoPor, gera*, entre outros (Campos et al., 2007).

As **Figuras 14, 15, 16, 17, 18, 19** e **20** a seguir, apresentam a taxonomia proposta. A **Figura 14** apresenta uma visão geral da taxonomia. No centro da figura é apresentado o assunto base da taxonomia, as boas práticas em ES. Fazem parte dessa base central os processos sugeridos por Kruchten (2001): processo de desenvolvimento iterativo, processo de gestão de requisitos, processo de modelagem, processo de arquitetura, processo de verificação de qualidade e processo de configuração e gestão de mudanças escolhidos não de forma a definir um modelo rígido ou direcionar a um modelo mercadológico, mas para servir como referência inicial de práticas utilizadas. A decomposição da taxonomia é feita a partir das outras figuras e foi objetivo seguir com a representação das práticas até chegar a um nível de artefato a ser gerado ou mesmo de elementos de um artefato. Por exemplo, na **Figura 20** o elemento da taxonomia que representa a boa prática de definição de *Documento de Alteração de Escopo*, é representado por seus elementos que já são itens que serão gerados em um artefato de alteração de escopo. A documentação poderá ser formada por uma lista de requisitos funcionais, uma lista de requisitos não funcionais, a lista de requisitos excluídos e a definição de estimativas de custo e prazo. As definições dos elementos da taxonomia podem ser consultadas e melhor entendidas em uma análise do **Apêndice A**.

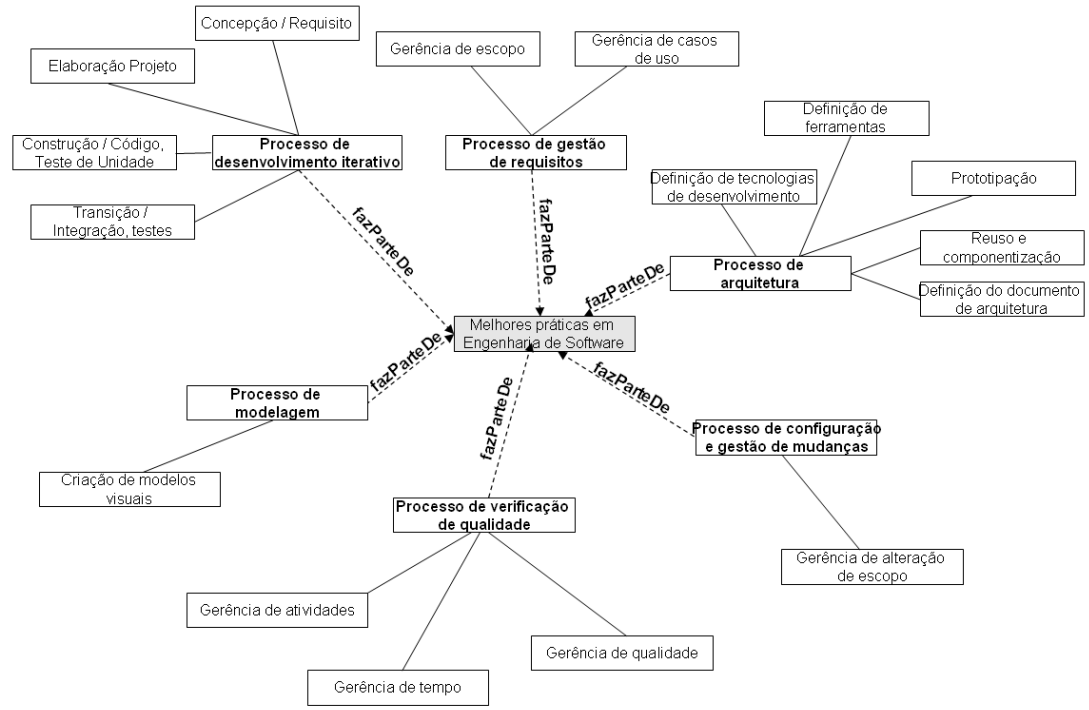

Figura 14: Taxonomia principal de boas práticas.
Fonte: do autor.

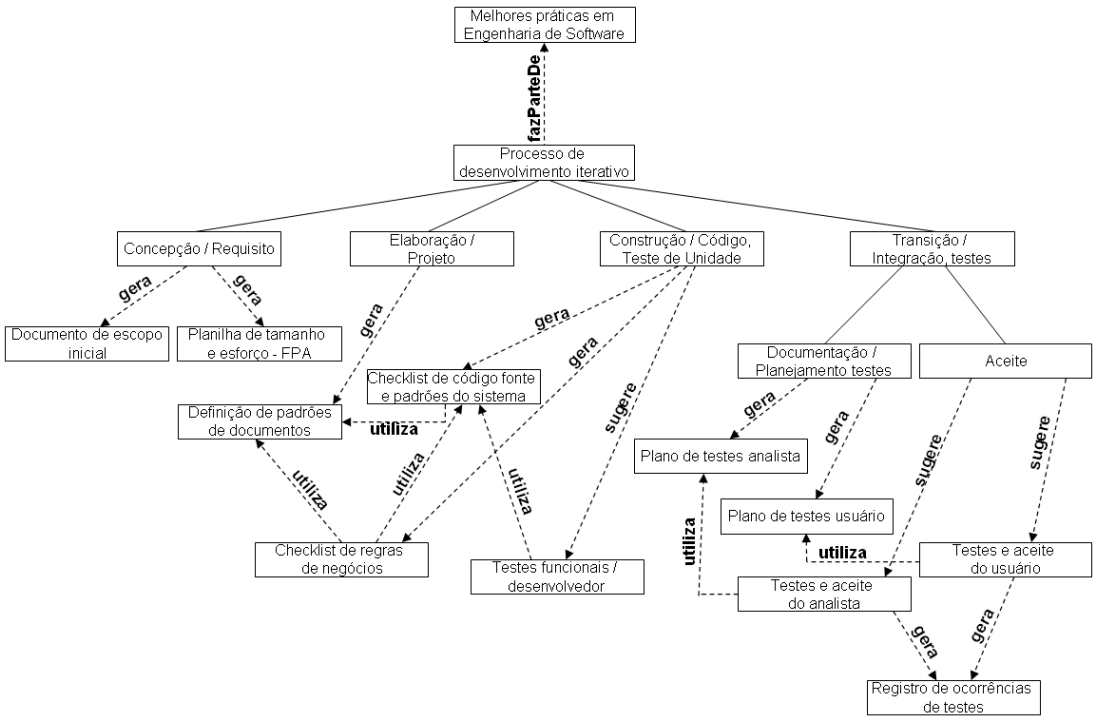

Figura 15: Taxonomia do processo de desenvolvimento iterativo.
Fonte: do autor.

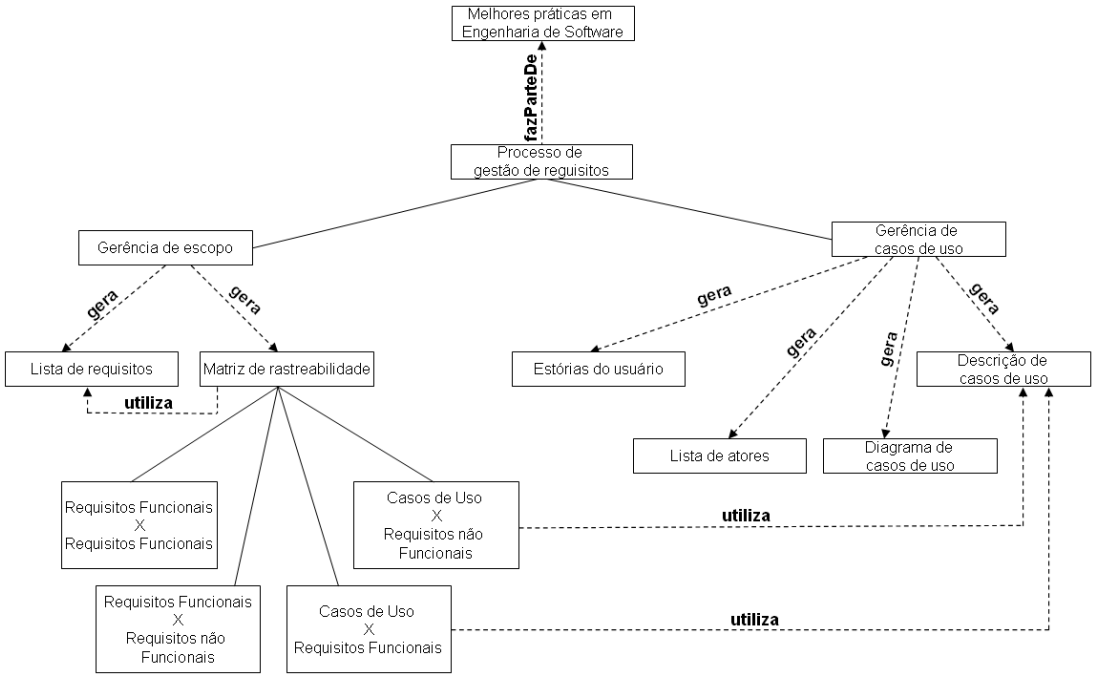

Figura 16: Taxonomia do processo de gestão de requisitos.
Fonte: do autor.

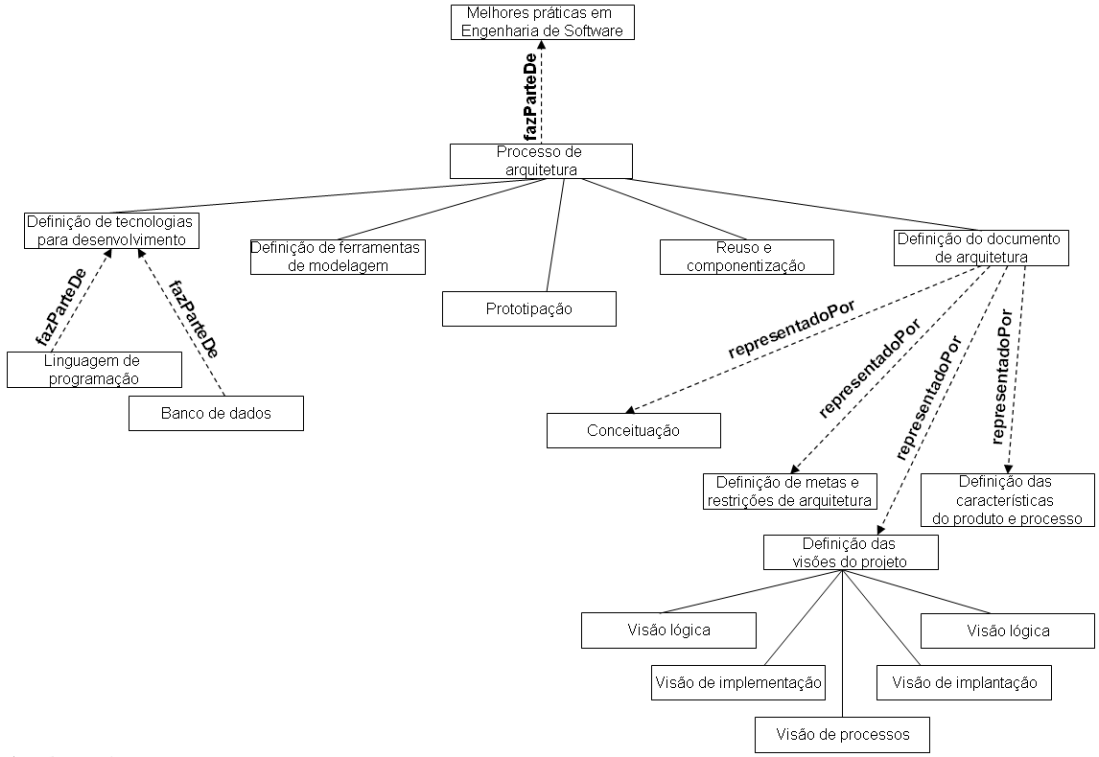

Figura 17: Taxonomia do processo de arquitetura.
Fonte: do autor.

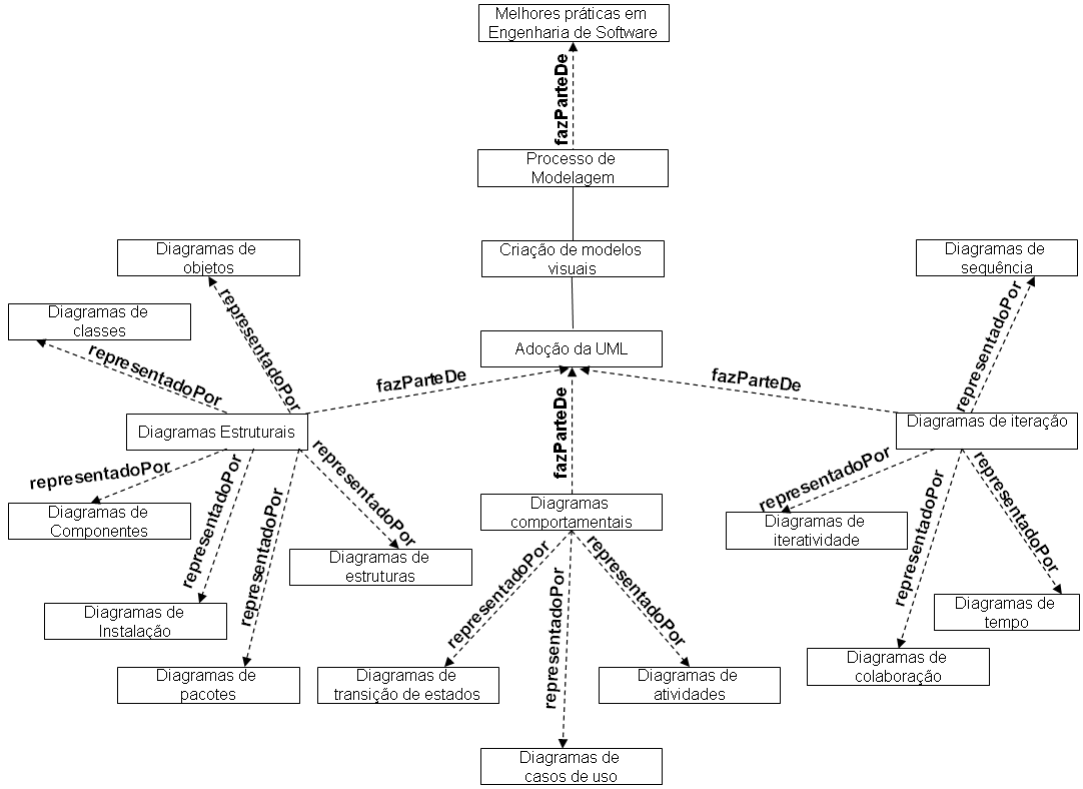

Figura 18: Taxonomia do processo de modelagem.
Fonte: do autor.

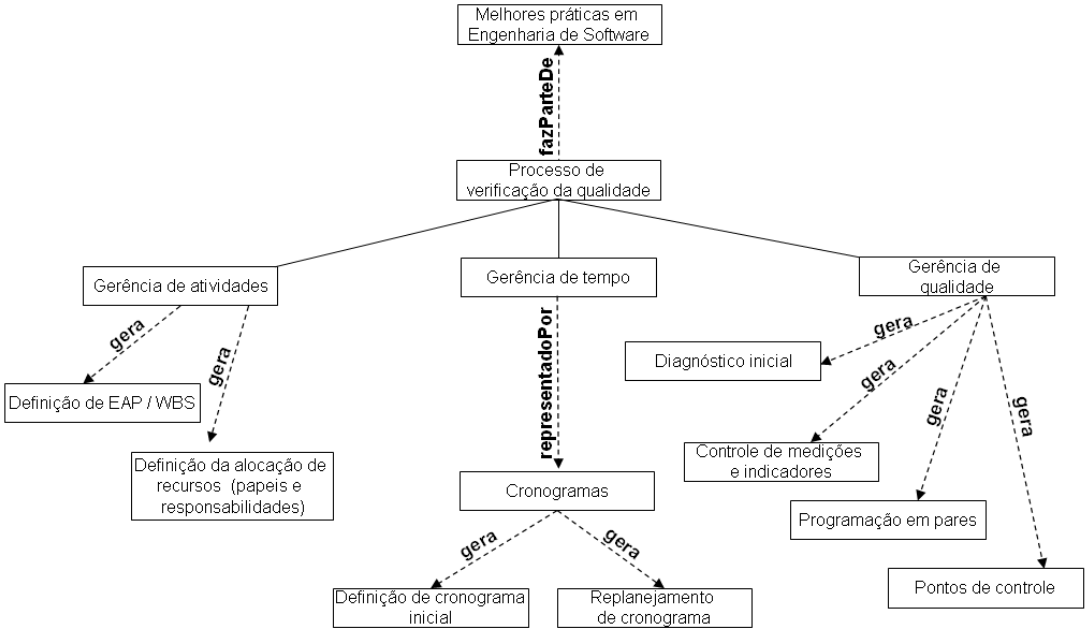

Figura 19: Taxonomia do processo de verificação de qualidade.
Fonte: do autor.

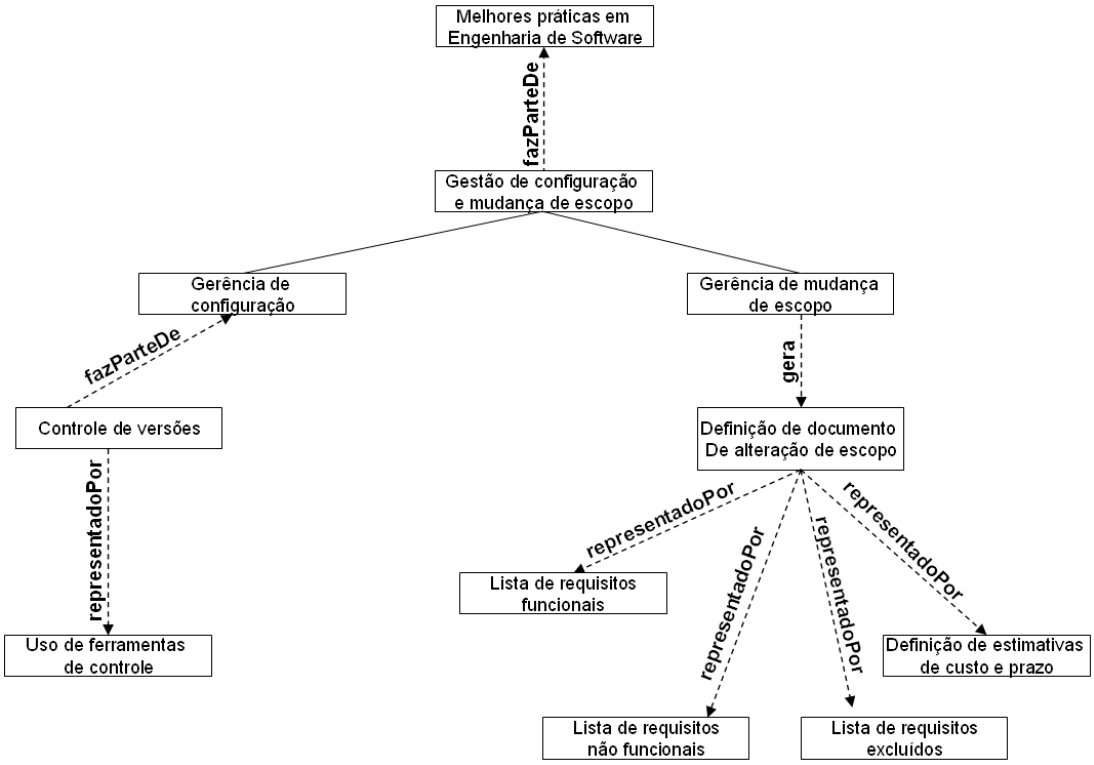

Figura 20: Taxonomia do processo de configuração e mudança de escopo.
Fonte: do autor.

4.4 Relação entre problemas, causas e boas práticas em desenvolvimento de software

Um fator importante a ser observado ao se tratar do assunto boas práticas é a sua relação entre os problemas e as principais causas com desenvolvimento de software. Foi avaliado o que os principais autores de ES apresentam sobre problema, suas causas e as possíveis boas práticas adotadas para a solução desses problemas.

Pfleeger (1998), Sommerville (2003) e Pressman (2005) citam que os principais sintomas de problemas com software podem ser vistos como:

- não entendimento das necessidades de usuário final;
- requisitos dinâmicos;
- módulos do sistema não se encaixam;
- manutenção difícil do código;
- descoberta tardia de erros;
- baixa qualidade;
- desempenho baixo do sistema;
- atrito de desenvolvedores e

- dificuldade em construir e liberar versões.

As principais causas relacionadas aos sintomas descritos são citadas pelos mesmos autores como:

- requisitos insuficientes;
- comunicação ambígua;
- arquitetura mal elaborada;
- alta complexidade;
- automação insuficiente;
- inconsistências não detectáveis;
- testes pobres;
- desenvolvimento em cascata, e;
- mudanças incontroláveis.

Com os elementos da estrutura analítica, seria possível estabelecer uma relação entre os principais sintomas dos problemas e suas causas conforme apresentado anteriormente e as possíveis soluções baseadas em boas práticas de desenvolvimento de software propostas a partir das conclusões de Kruchten (2001). As **Figuras 21, 22, 23, 24, 25** e **26** apresentam algumas relações possíveis e não se tem a intenção de esgotar todas as possibilidades dessas relações, mas apenas exemplificar algumas propostas Kruchten (2001).

4.4.1 Desenvolvimento Iterativo

Para Kruchten (2001), o desenvolvimento iterativo do software, como melhor prática, pode auxiliar nas soluções de problemas tais como: é possível reagir de forma eficaz encontrando os equívocos do software de forma antecipada; a abordagem incentiva o usuário final a concluir as etapas do ciclo iterativo tendo uma visão real dos requisitos do sistema; o teste iterativo e contínuo habilita uma avaliação objetiva do status do projeto e auxilia na detecção antecipada de erros antes do sistema prosseguir para fases mais adiantadas; a carga de trabalho da equipe é minimizada e distribuída de forma mais uniforme ao longo de todo o ciclo de vida do projeto.

A **Figura 21** apresenta uma relação dos sintomas, suas causas e o desenvolvimento iterativo do software como melhor prática.

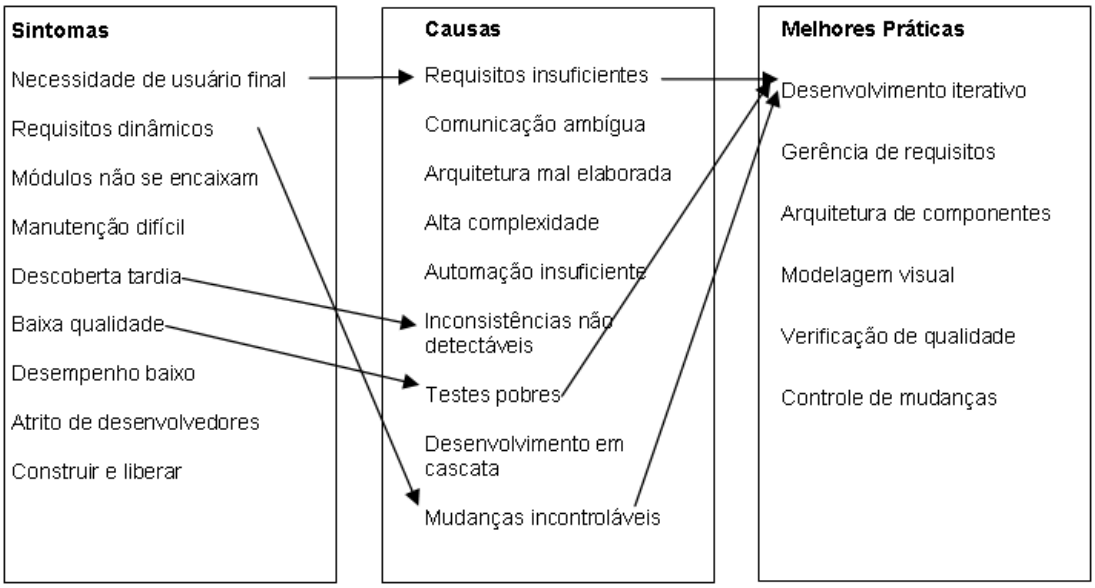

Figura 21: Relação de sintomas de problemas, suas causas e soluções – desenvolvimento iterativo.
Fonte: do autor.

4.4.2 Gerência de Requisitos

Com relação à Gerência de Requisitos como melhor prática, Kruchten (Kruchten, 2001) considera que a atividade auxilia no controle e identificação das mudanças dos requisitos; cria uma disciplina com relação ao gerenciamento dos requisitos; as discussões, negociações e comunicações são feitas com base nos requisitos definidos; auxilia na priorização dos requisitos a serem desenvolvidos e afeta diretamente na detecção antecipada de problemas.

A **Figura 22** apresenta uma relação dos sintomas, suas causas e a gerência de requisitos como melhor prática.

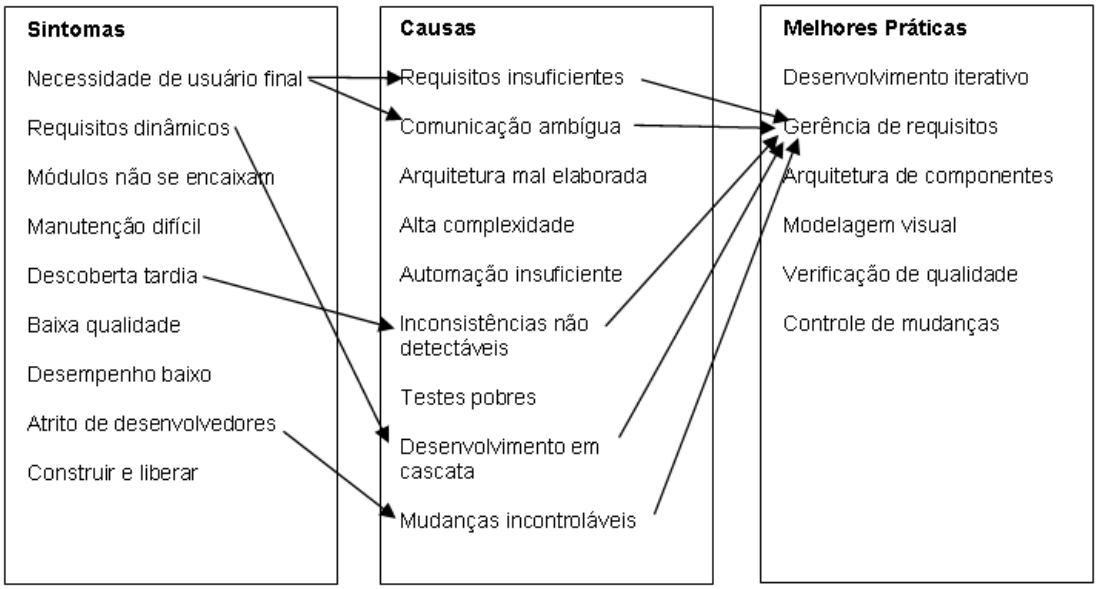

Figura 22: Relação de sintomas de problemas, suas causas e soluções – gerência de requisitos.
Fonte: do autor.

4.4.3 Arquitetura Baseada em Componentes

Kruchten (2001) sugere que seja utilizada a prática de arquitetura baseada em componentes, uma vez que ela auxilia no controle do desenvolvimento iterativo e incremental de um sistema. Com a prática é possível melhorar a gerência dos pontos de vistas distintos entre os diferentes profissionais envolvidos na equipe; permite uma melhor organização do software, uma melhor seleção dos elementos que compõem esse software; facilita o reuso dos componentes e a sua rápida recuperação; possibilita uma melhor separação das relações entre os elementos de um sistema suscetível a mudanças. Além disso, a prática não está relacionada somente às questões comportamentais e estruturais do sistema, está diretamente relacionada a fatores econômicos do software: a comercialização, estética, desempenho e compreensão.

A **Figura 23** apresenta uma relação dos sintomas, suas causas e a arquitetura baseada em componentes como melhor prática para solução dos problemas apresentados.

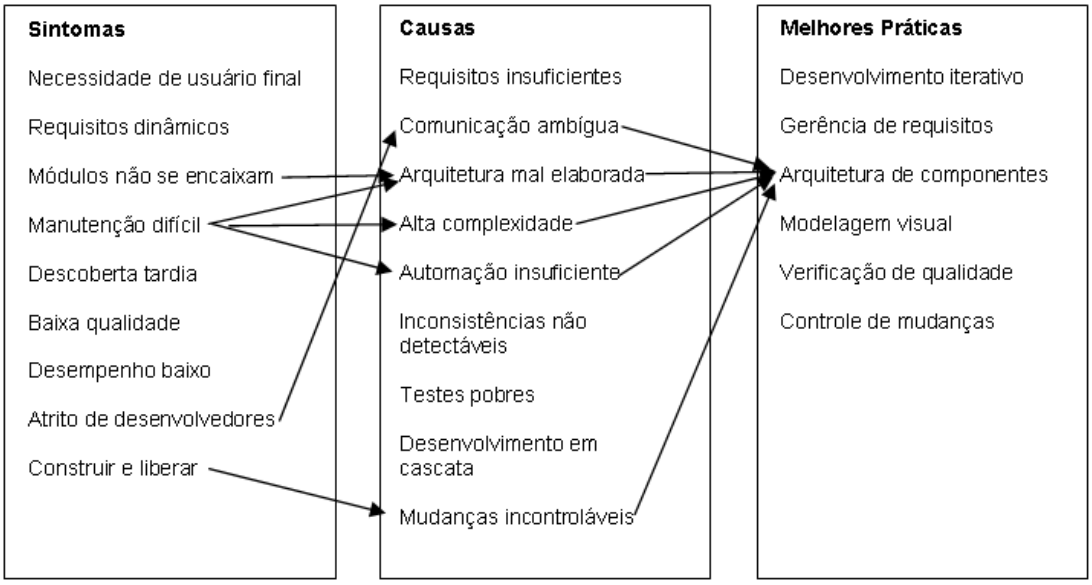

Figura 23: Relação de sintomas de problemas, suas causas e soluções – arquitetura de componentes.
Fonte: do autor.

4.4.4 Modelagem Visual

Com relação à prática de modelagem visual é importante o uso de diagramas, tais como diagramas de classe, de estado, de componentes, de cenário, de casos de uso e de distribuição, para modelar um sistema complexo. Assim, é possível melhorar o entendimento da complexidade de um sistema a partir dos modelos visuais de suas partes.

A modelagem visual ainda permite que sejam mantidas as consistências entre os artefatos de um sistema, seus requisitos, construções e implementações.

Para Kruchten (2001) a modelagem visual permite, quando unida à prática do desenvolvimento iterativo, expor e avaliar mudanças arquiteturais e comunicar essas mudanças para a equipe de desenvolvimento.

Kruchten (2001) sugere que a modelagem visual oferece várias soluções para os problemas de desenvolvimento de software tais como:

- Os modelos auxiliam na construção do software de forma a aproximar o projeto do sistema real;

- Arquiteturas não modulares e inflexíveis são expostas;

- A qualidade do sistema começa com um bom projeto;

- Ferramentas de modelagem visual fornecem suporte para modelagem UML.

- Obter os requisitos de forma mais precisa;

- Construir modelos estruturais para dar apoio à implementação.

A **Figura 24** apresenta uma relação dos sintomas, suas causas e a modelagem visual de sistemas como melhor prática para solução dos problemas apresentados.

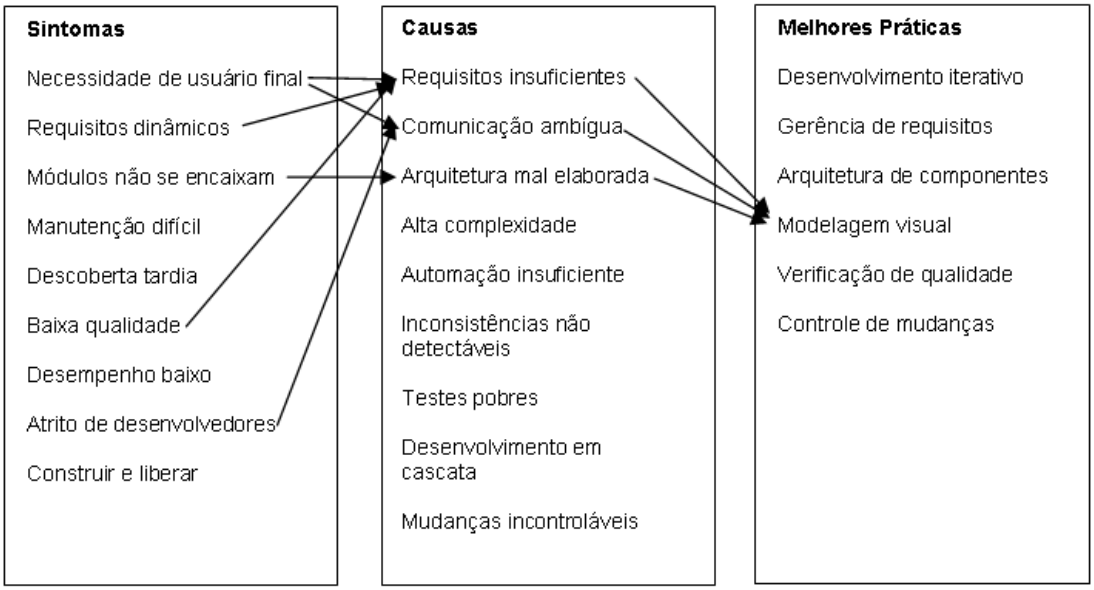

Figura 24: Relação de sintomas de problemas, suas causas e soluções – modelagem visual.
Fonte: do autor.

4.4.5 Verificação da Qualidade

A prática de verificação da qualidade está intimamente relacionada a processos que irão amparar a qualidade dos produtos e do processo. Verificar problemas no software de forma tardia é muito mais custoso do que em fases iniciais de projeto.

A melhoria da qualidade do produto deve ser garantida pela qualidade dos seus subprodutos, como por exemplo: componentes, subsistemas, arquiteturas, entre outros.

A qualidade do processo deve abranger atividades que garantam a qualidade de todo o ciclo de vida do desenvolvimento do software. Essas atividades estão diretamente relacionadas a planejamentos de iterações, de testes, a confecção de artefatos, modelos de sistemas ou de projeto, entre outros Kruchten (2001).

A **Figura 25** apresenta uma relação dos sintomas, suas causas e a verificação de qualidade de sistemas como melhor prática para solução dos problemas apresentados.

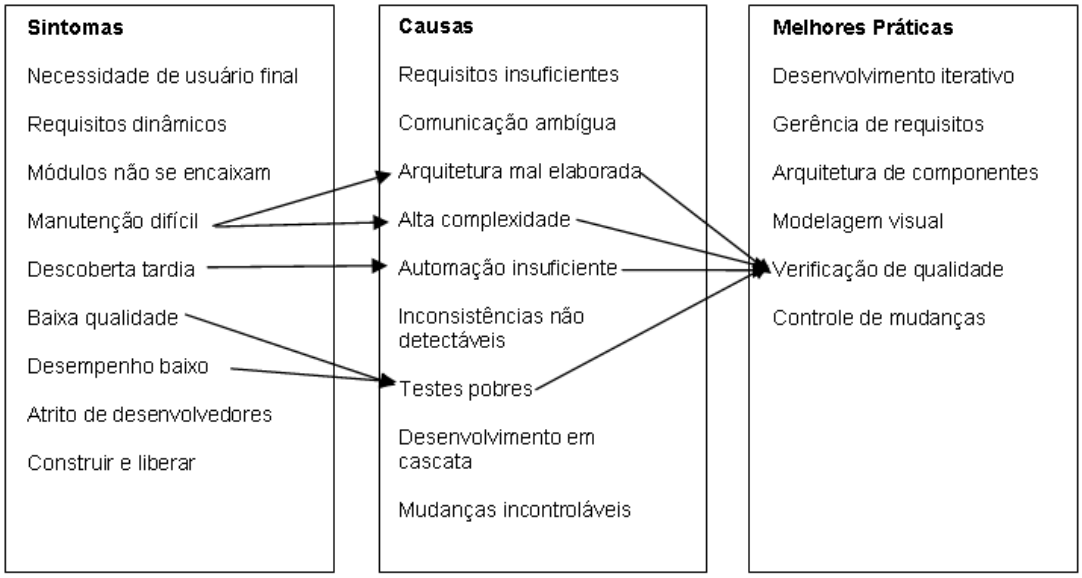

Figura 25: Relação de sintomas de problemas, suas causas e soluções – verificação de qualidade.
Fonte: do autor.

4.4.6 Controle de Mudanças

A prática de controlar mudanças é utilizada para que alterações solicitadas pelo usuário, seja por falta de entendimento dos requisitos estabelecidos ou por mudanças de escopo no software desenvolvido, sejam sincronizadas em todas as etapas do desenvolvimento. Artefatos, versões de código e projeto implementados devem estar sincronizados de acordo com essas alterações Kruchten (2001).

A **Figura 26** apresenta uma relação dos sintomas, suas causas e o controle de mudanças em sistemas como melhor prática para solução dos problemas apresentados.

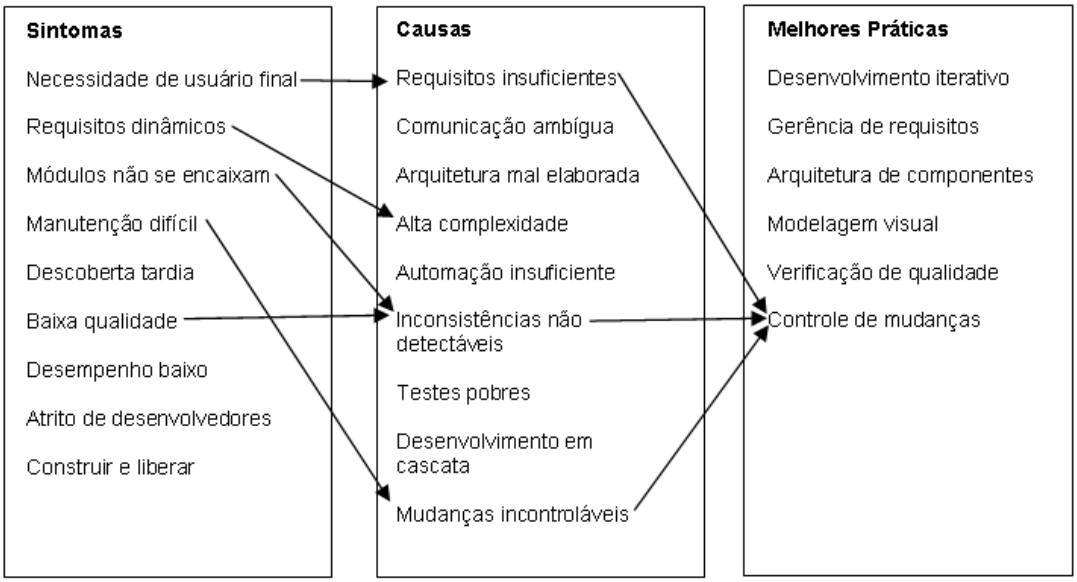

Figura 26: Relação de sintomas de problemas, suas causas e soluções – controle de mudanças.
Fonte: do autor.

5 USO DA TAXONOMIA EM RECOMENDAÇÃO DE ADOÇÃO DAS BOAS PRÁTICAS

5.1 Relações preliminares de precedência ou de potencialização de uso

As MPEs podem melhorar a qualidade dos processos de desenvolvimento de software com a utilização de boas práticas de ES. É importante, portanto, que as empresas entendam a utilização dos elementos propostos e o seu impacto na melhoria da qualidade a fim de utilizá-los para auxiliar na busca por soluções para os problemas que atingem os seus processos no ciclo de vida do software.

A relação de precedência ou potencialização entre elementos pode ser uma forma eficaz de apresentar às empresas como a utilização de como um elemento pode potencializar a utilização de outro ou como deve ser utilizado antes que outro seja aplicado. Dessa forma, a utilização conjunta de elementos da estrutura analítica será a base para que sejam atingidos os objetivos de cada melhor prática proposta.

As **Figuras 27** e **28** apresentam relações de precedência e potencialização entre alguns elementos da taxonomia e foram dispostas aqui de forma sucinta para demonstrar como alguns elementos têm relações de interdependência. As relações foram estabelecidas a partir da prática profissional do autor desse trabalho. As possibilidades de relacionamento dos elementos não se esgotam com os exemplos dispostos nas figuras a seguir, mas evidenciam as possibilidades de relação e servem para auxiliar a compreensão do que está sendo proposto no trabalho de pesquisa.

A **Figura 27** apresenta uma relação proposta de precedência entre alguns elementos da prática de *Gerenciamento de Requisitos* e *Qualidade de Processo e Produto*. Ou seja, a confecção de uma *Matriz de Rastreabilidade* contendo *Casos de Uso x Requisitos Funcionais* ou *Casos de Uso x Requisitos Não Funcionais* vai auxiliar a definição da *Estrutura Analítica do Projeto (EAP)* e a *Definição da*

Alocação de Recursos (papéis e responsabilidades), elementos que fazem parte da *Verificação da Qualidade*.

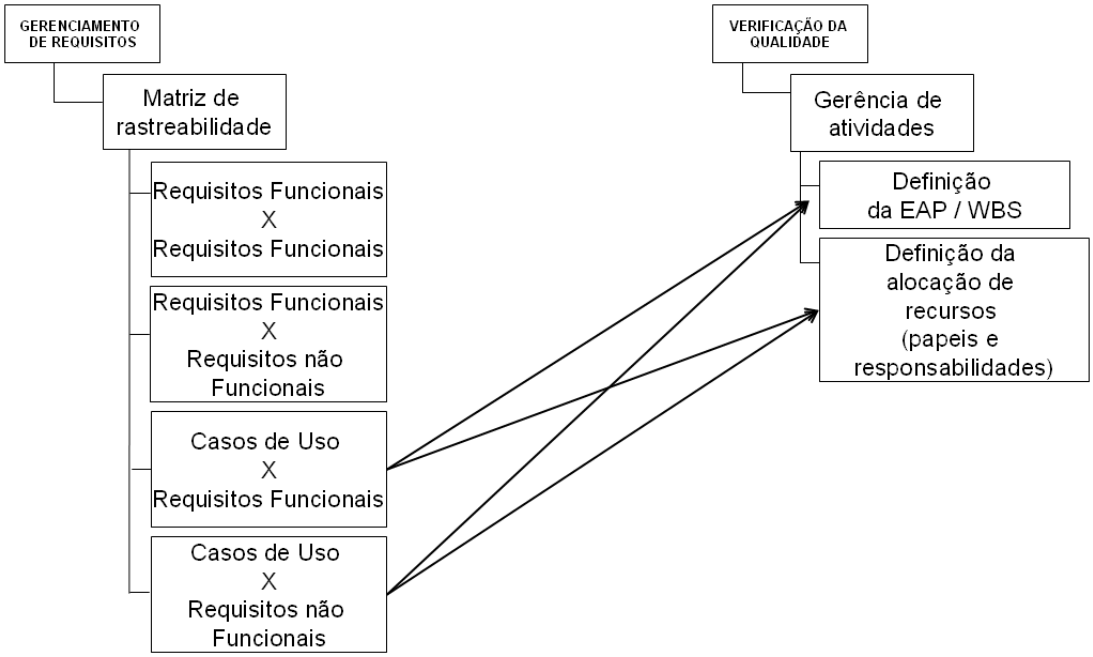

Figura 27: Relação de precedência entre elementos da estrutura analítica de boas práticas.
Fonte: do autor.

A **Figura 28** apresenta uma relação proposta de potencialização entre alguns elementos das práticas de *Verificação da Qualidade* e *Desenvolvimento Iterativo*. Na relação são apresentados os elementos de *Medições e Indicadores*, influenciando diretamente nos elementos de testes de aceite do analista, do usuário e no registro de ocorrências que pode ser coletado ao longo do processo de desenvolvimento.

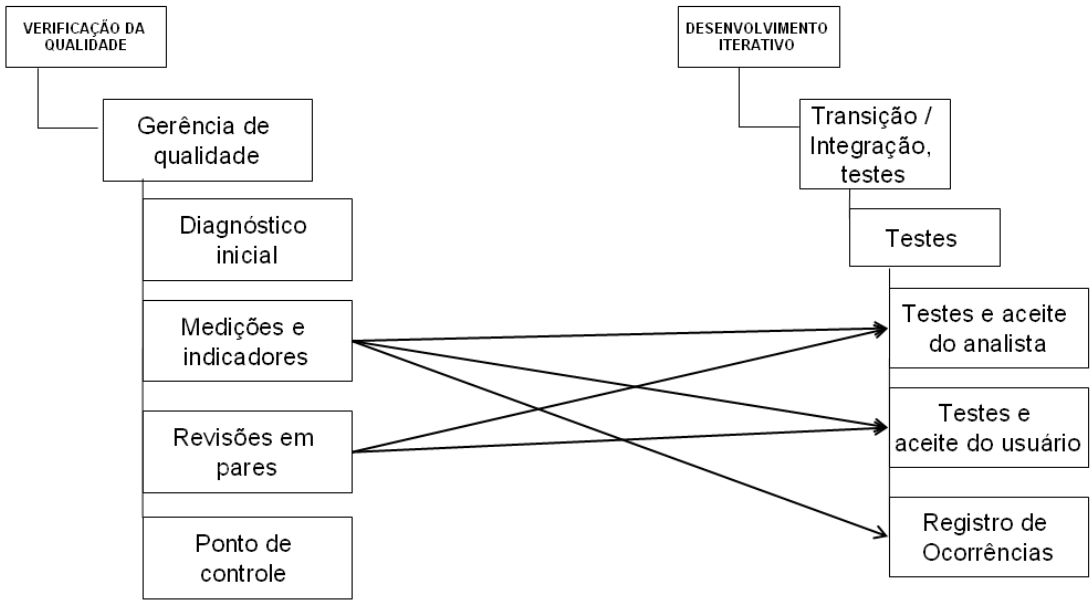

Figura 28: Relação de potencialização entre elementos da estrutura analítica de boas práticas.
Fonte: do autor.

5.2 Justificando a relação de elementos da taxonomia através de diagramas de influência

5.2.1 Visão Geral de Diagramas de Influência

Peter Senge (1990) propõe a utilização de diagramas de influência para efetuar a análise sistêmica. A análise do todo e das relações de suas partes poderão levar o tomador de decisões a entender melhor o problema que está sendo avaliado. Em software, pode-se afirmar que a falta de uma visão sistêmica do processo de desenvolvimento leva os gerentes a tomarem decisões reativas e pontuais, considerando apenas o problema presente, sem relacioná-lo com o seu ambiente, suas variáveis e demais problemas correlacionados (Ambrósio, 2008).

Apesar dos esforços das pesquisas em ES, os aspectos gerenciais do processo de software têm sido ineficientemente explorados. Essa deficiência tem trazido problemas de entendimento no processo de desenvolvimento, enfraquecendo o amparo para tomadas de decisão com relação ao projeto como um todo (Ambrósio, 2008).

Como exemplo de diagrama de influência na área de projetos de software pode ser apresentado na **Figura 29,** o que é proposto por Abel-Hamid e Madnick (1991).

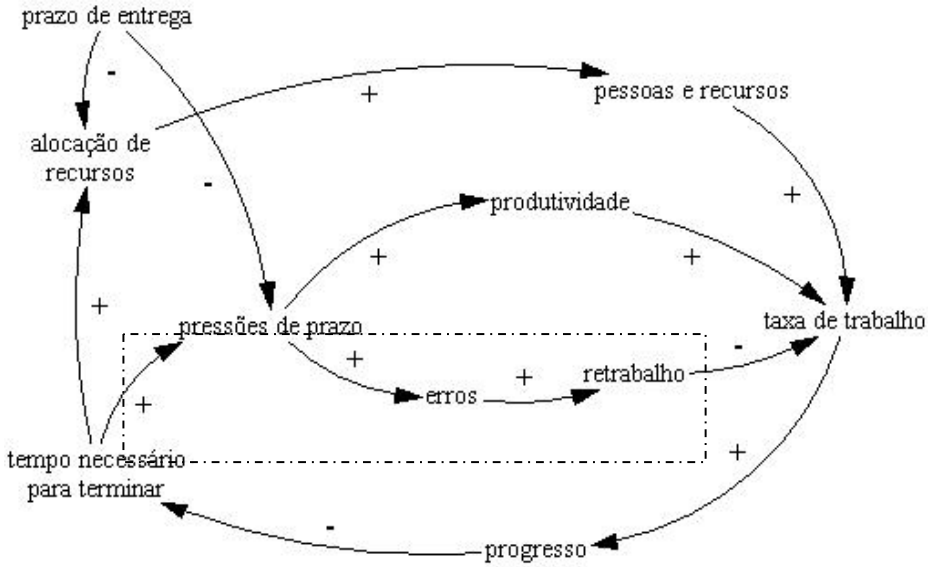

Figura 29: Relacionamentos entre variáveis envolvidas no gerenciamento de projetos de software (Ambrósio, 2008).

O exemplo está aderente a taxonomia proposta, uma vez que seu conteúdo está diretamente relacionado a questões do dia a dia de projetos de software. Dessa forma, pretende-se apenas apresentar de forma sucinta um contexto representado por diagrama de influência para facilitar a compreensão dos diagramas que serão apresentados em seguida.

O diagrama de influência apresenta a relação de pressões para entrega do projeto no prazo e o atraso de projetos. Diante da situação os tomadores de decisão tendem a impor um maior esforço no emprego de horas extras com o objetivo de aumentar a produtividade no projeto. No diagrama apresentado, pode-se observar o relacionamento de influência visível entre a variável *pressões de prazo* e a variável *produtividade*, e a conseqüente influência da relação das duas variáveis sobre uma terceira chamada *taxa de trabalho*.

Para Ambrosio (2008), a quantidade de erros devido ao cansaço e ao estresse é aumentado, influenciando diretamente nas taxas de retrabalho, o que na maioria das vezes não é considerado pelos tomadores de decisão. Após um tempo, a tendência é de atraso do projeto e diminuição de produtividade, uma vez que há o aumento do ciclo de erros e necessidade de aumento de horas, tendo ai um ciclo vicioso. Na figura apresentada pode-se notar essa relação de influência destacada em linhas tracejadas no diagrama.

O diagrama de influência pode ser utilizado para explicar os elementos e as relações que compõem a taxonomia e auxiliar nos processos decisórios e na escolhas de boas práticas diante das situações do cotidiano de projetos de software. Com ele é possível verificar graficamente e de forma sistêmica os elementos envolvidos em determinado momento de análise do problema e a influência de cada uma das partes na solução do que se está analisando.

5.2.2 Relação de Elementos da Taxonomia Suportada por Diagramas de Influência

A relação entre os elementos da taxonomia pode ser explicada a partir dos diagramas de influência. O desenho de um diagrama poderá influenciar nas decisões de quem o analisa e poderá ajudar na compreensão de quais práticas podem ser adotadas em determinadas situações com software.

Neste trabalho de pesquisa foram utilizados dois diagramas de influência para explicar a relação entre os elementos da taxonomia de boas práticas e para enriquecer a relação, foram adicionados alguns fatores importantes no contexto de software, como por exemplo: custo, prazo, ocorrências encontradas em testes e insatisfação de clientes. Tais fatores foram escolhidos aleatoriamente baseada na experiência do autor com a gerência de projetos de software e não se pretende com esses elementos representar todo o contexto de variáveis encontradas no processo de desenvolvimento de software.

No primeiro diagrama, foram incluídos elementos da taxonomia retirados de diversos processos a fim de se estabelecer e apresentar a relação entre esses elementos independentes do processo a que estão subordinados. Do *Processo de Gestão de Requisitos* foram selecionados a *Matriz de Rastreabilidade* com o elemento *Casos de Uso X Requisitos Funcionais*, do *Processo de Gestão de Qualidade* foram selecionados a *Definição da EAP*, *Definição de Alocação de Recursos*, a *Definição de Cronograma* e o *Replanejamento de Cronograma*. Além disso foram inseridos no diagrama o elemento *Mudança de Escopo* que pode ser visto no controle utilizado no *Processo de Configuração e Mudança de Escopo* na *Gerência de Mudança de Escopo.* Com isso pretende-se estabelecer as relações de influência positiva e negativa em cada relação estabelecida e avaliar o impacto da influência dessas relações com *Prazo* e *Custo*.

A **Figura 30** apresenta o diagrama de influência dos elementos supracitados.

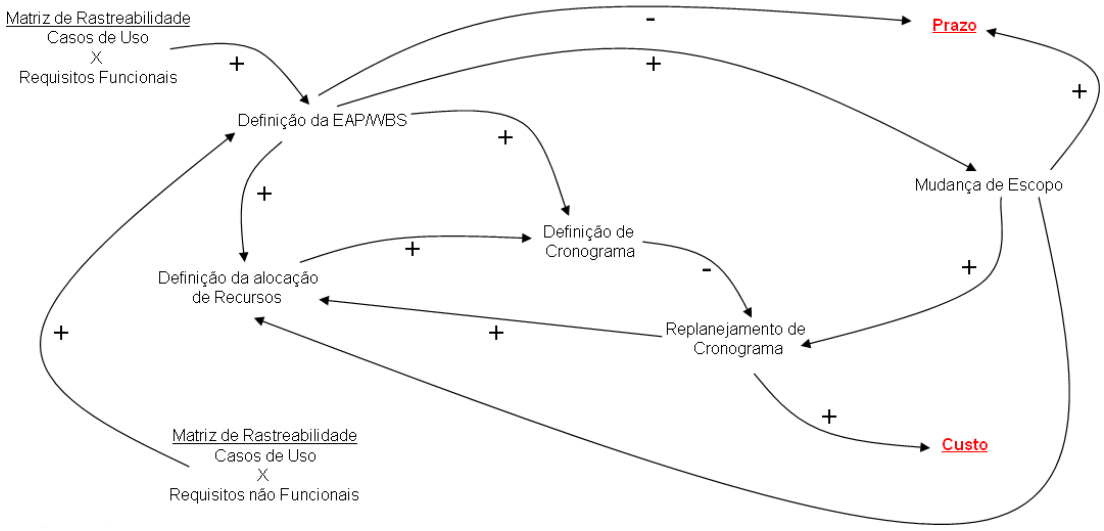

Figura 30: Relação de boas práticas por diagrama de influência. Diagrama 1.
Fonte: do autor.

A análise do diagrama apresentado pode ser feita da seguinte forma: em uma primeira análise, pode-se observar a influência positiva da elaboração de *Matrizes de Rastreabilidade* na construção e definição de uma *EAP*. Continuando a análise gráfica, é possível sugerir que uma boa construção de uma *EAP* poderá influenciar positivamente na definição de um *Cronograma de Atividades* e na *Alocação de Recursos*. A correta definição de *Alocação dos Recursos* irá influenciar positivamente na definição do cronograma inicial e consequentemente reduzir as alterações e replanejamentos de cronograma ao longo do processo de desenvolvimento do software.

Além disso, estabelecer inicialmente uma *EAP* bem estruturada, poderá influenciar positivamente o controle e registro das alterações de escopo do projeto. Pode-se verificar que o aumento de *Mudanças de Escopo* em um projeto de software irá influenciar no aumento de *Replanejamento de Cronograma*, na *Redefinição da Alocação de Recursos* e consequentemente no aumento de *Custos* e *Prazo*.

No segundo diagrama foram incluídos elementos de boas práticas de processos como: *Desenvolvimento Iterativo* e *Verificação da Qualidade*. Foram incluídos na análise questões como *Insatisfação* do usuário e *Ocorrências de Testes*, coletadas a partir de *Testes Funcionais do Desenvolvedor* e *Testes do Analista*.

A **Figura 31** apresenta o segundo diagrama com a relação de elementos de boas práticas e a sua influência no registro de ocorrências de testes e insatisfação do usuário.

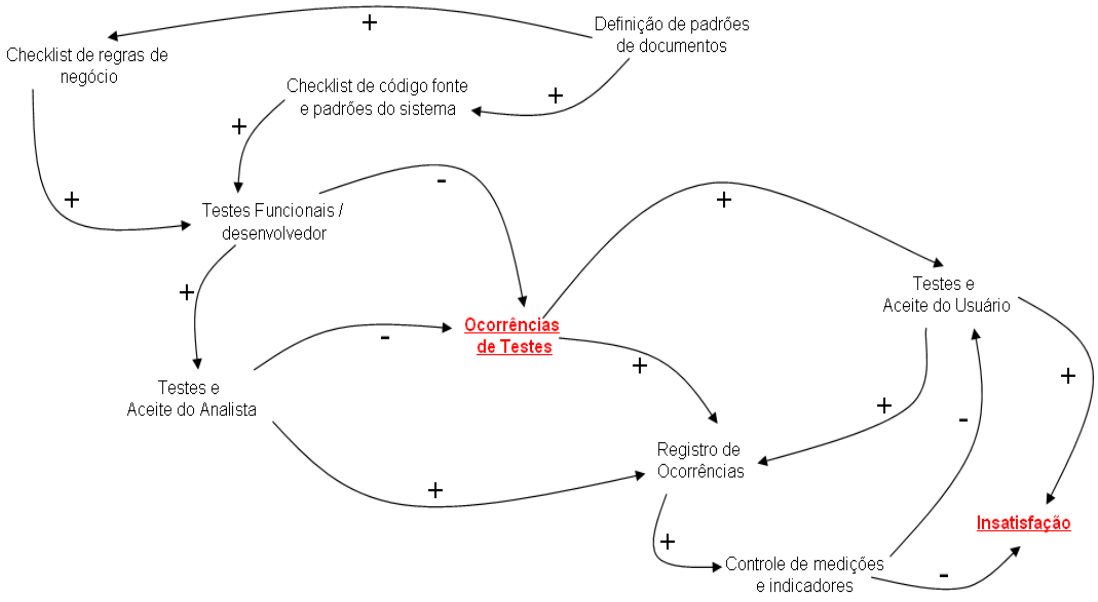

Figura 31: Relação de boas práticas por diagrama de influência. Diagrama 2.
Fonte: do autor

As relações estabelecidas convergem para auxiliar uma análise de como as boas práticas discriminadas no diagrama podem ajudar a minimizar problemas de insatisfação do usuário.

Primeiramente deve-se observar a influência positiva de uma definição inicial de padrões de documentação na construção de *Checklists* de código fonte e padrões do sistema, como também no *Checklist* de regras de negócio. Esses *Checklists* irão orientar os testes funcionais do desenvolvedor colaborando para que esses testes aumentem as chances de sucesso na entrega de um produto com maior qualidade. Os testes funcionais do desenvolvedor, de acordo com a influência apresentada no diagrama, irão afetar positivamente os *Testes e Aceite do Analista* funcional, que por sua vez irá registrar um menor número de *Ocorrências de Testes*. Por sua vez, caso o desenvolvedor efetue *Testes Funcionais e de Unidade* desordenadamente acarretará em um maior volume de *Registros de Ocorrências de Testes* e que deverá ter um maior controle na medição e no estabelecimento dos indicadores de testes.

O maior volume de controle de ocorrências de testes, de medição e indicadores ocasionará, segundo o diagrama, um menor índice de *Insatisfação* do usuário. Por outro lado, a insatisfação aumentará, caso o volume de testes do usuário seja maior, em que fará mais registros de ocorrências e necessitará de retestes de funcionalidades.

Apesar dos diagramas de influência terem sido construídos com base em recortes de partes do todo da taxonomia e agregados a eles fatores, inerentes ao desenvolvimento de software como custo, prazo, ocorrências de testes e insatisfação do usuário, pode-se verificar que os diagramas fazem parte de um grande sistema de interdependência entre todos os elementos da taxonomia, em que a elaboração ou a utilização de um ou outro elemento de forma ineficaz irá acarretar uma influência negativa e em cascata afetando todos os outros elementos e fatores de software envolvidos no projeto.

Pelas limitações de escopo desta dissertação, não foram elaboradas todas as relações entre os elementos da taxonomia. O que foi apresentado é apenas uma visão proposta e inicial de como as relações entre os elementos demonstram a sua interdependência e a influência negativa ou positiva entre as partes. Este recurso é uma importante ferramenta que permite ter visão sistêmica dos impactos de problemas e decisões no processo de desenvolvimento de software.

Portanto, cabe ressaltar que cada elemento deve ser utilizado conforme boas práticas de ES e que os diagramas de influência auxiliam fortemente na escolha do grupo de elementos da taxonomia de boas práticas que sejam mais aderentes à realidade do projeto de software, conduzindo os tomadores de decisão em uma escolha baseada em critérios técnicos, reduzindo o risco de adoção.

5.3 Aplicação da taxonomia para seleção e indicação de boas práticas

A partir da aplicação dos questionários, explicados na seção 2.5.1 Aspectos Metodológicos deste trabalho de pesquisa, foi possível a obtenção de um gráfico com indicação dos fatores de risco de cada empresa. Os cinco fatores de risco podem ser representados conforme a **Figura 32**, que apresenta a *plotagem* dos resultados de duas das empresas: a representação no gráfico com linha contínua é de uma empresa em que os riscos de adoção de MA são menores, e outra, representada no gráfico com linha tracejada, pode ser avaliada como de menor risco na adoção de MDP.

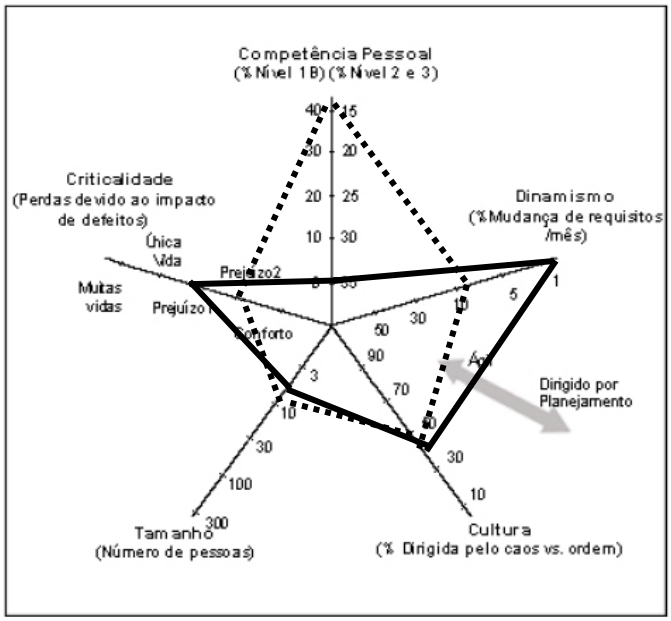

Figura 32: O perfil de duas empresas DPI-03 (▬) e DPI-04 (▪▪▪▪).

A análise deve ser feita levando-se em consideração a tendência dos fatores em direção ao centro do gráfico, caso que sugere inicialmente um menor risco na adoção de boas práticas relacionadas a MA. Quanto maior for o afastamento do centro do gráfico em direção às suas extremidades, maiores serão as chances de sucesso com adoção de boas práticas de MDP (Soares et al., 2008), e maiores os riscos de adoção de práticas dos MA. Alguns fatores têm predominância sobre os

demais, direcionando as decisões. Por exemplo, se o uso do software apresenta riscos à vida humana, o espaço de soluções fica muito restrito, exigindo métodos específicos e altos investimentos em qualidade, adoção de processos organizados, restringindo inclusive o porte e o tipo de empresas.

Os resultados tabulares que deram origem aos gráficos polares das empresas pesquisadas podem ser observados nas **Tabelas 13** e **14**. Os nomes das empresas foram substituídos por identificadores fictícios.

Tabela 13: Avaliação para a empresa DPI-03.

Identificador: DPI-03	
Tamanho da Equipe	Pequena
Dinamismo do Problema	Pouco Dinâmico
Cultura da Equipe	Cultura Tende a Métodos Dirigidos por Planejamento
Criticalidade dos Problemas	Risco de Prejuízo Financeiro
Competência Pessoal da Equipe	Equipe com perfil menos adequado à agilidade

Fonte: (Soares et al., 2008).

A empresa DPI-03 apresentou um gráfico simétrico onde os eixos *Dinamismo, Cultura* e *Criticalidade* sugerem uma empresa com características propícias à adoção de soluções e métodos dirigidos por planejamento. Um fator se destaca no eixo *Criticalidade*, onde o alto risco de prejuízo financeiro sugere uma abordagem de um método mais controlado e detalhado, com documentação mais extensiva, derivado de MDP. Além desse fator preponderante, o fator *Cultura* também sugere a abordagem planejada, pois entende-se que a empresa possua uma equipe ainda em formação, com pouca experiência, o que significa riscos maiores para adoção de práticas de organização do trabalho com maiores graus de liberdade.

Tabela 14: Avaliação para a empresa DPI-04.

Identificador: DPI-04	
Tamanho da Equipe	Média
Dinamismo do Problema	Muito Dinâmico
Cultura da Equipe	Cultura Tende a Métodos Dirigidos por Planejamento
Criticalidade dos Problemas	Risco de Prejuízo Financeiro
Competência Pessoal da Equipe	Equipe com perfil menos adequado à agilidade

Fonte: (Soares et al., 2008).

A empresa DPI-04 possui uma equipe de tamanho médio, que tem trabalhado com requisitos dinâmicos. Há uma concentração de profissionais do nível 1B que são considerados experientes, e que podem atingir habilidades mais sofisticadas. O eixo

de *Criticalidade* sugere uma empresa que vem trabalhando com projetos com um risco financeiro moderado. A *Cultura* da equipe aponta para métodos dirigidos por planejamento.

5.3.1 Justificando a adoção de boas práticas a partir de fatores críticos analisados no gráfico polar

A análise conjunta de fatores críticos pode ser observada no trabalho de pesquisa de Soares (2007). Tais observações concluem que alguns fatores avaliados em conjunto sugerem a adoção de métodos como MDP, MA ou híbridos, mas que também alguns fatores são determinantes na escolha de um método e por conseqüência das práticas a serem utilizadas no desenvolvimento do software.

A seguir será relatada a análise do trabalho de Soares (2007) com relação à interferência entre fatores:

Competência Pessoal e Cultura

Com relação à avaliação conjunta de fatores Soares (2007) conclui que, o fator crítico *Competência Pessoal* pode influenciar o resultado do eixo do fator *Cultura* no gráfico polar. Tal influência sugere que uma equipe constituída por um percentual alto de profissionais, com níveis de habilidade 2 ou 3, pode potencializar uma cultura orientada pelo caos. Profissionais com estes níveis de competência possuem mais experiência para lidar com situações adversas (com e sem precedentes) e possuem conhecimento tácito significativo podendo tomar decisões sobre o desenvolvimento de projetos de software sem a necessidade de ter como suporte uma vasta documentação (Soares, 2007). A **Figura 33** apresenta o gráfico da relação entre os dois fatores analisados.

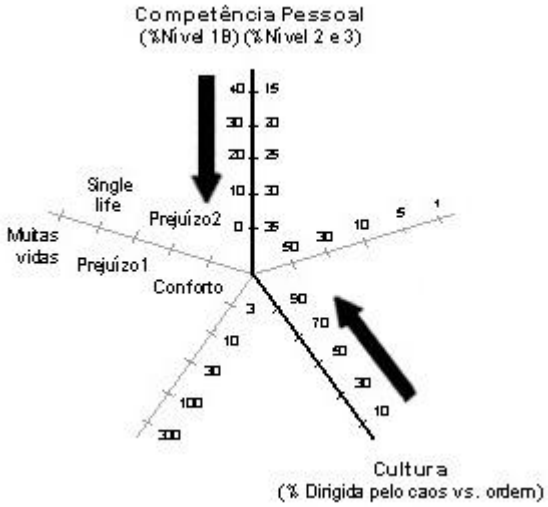

Figura 33: Representação da interferência entre o fator Competência Pessoal e o fator Cultura (Soares, 2007).

Dinamismo e Cultura

Outra avaliação de análise conjunta de fatores pode ser observada a partir da co-relação entre os fatores *Dinamismo* e *Cultura*.

Para Soares (2007), o fator crítico *Dinamismo* pode influenciar o fator crítico *Cultura* em contextos de desenvolvimento que apresentem um alto dinamismo. Uma cultura orientada pela ordem ou com alto grau de formalismo pode não ser apropriada quando se está trabalhando em contextos de softwares com requisitos muito dinâmicos. No ambiente de cultura formal e burocrática as atividades destinadas à organização do desenvolvimento implicam em trabalho adicional quando ocorrem mudanças nos requisitos para melhor gerenciá-los, desta forma o dinamismo dos requisitos pode inviabilizar o trabalho burocrático proposto em empresas que estão voltadas a uma cultura de formalismo. Já uma cultura orientada pelo caos impõe menos sobrecarga quando essas mudanças ocorrem, permitindo que a equipe esteja mais preparada para lidar com mudanças constantes nos requisitos. A **Figura 34** apresenta a relação entre os fatores *Dinamismo* e *Cultura*.

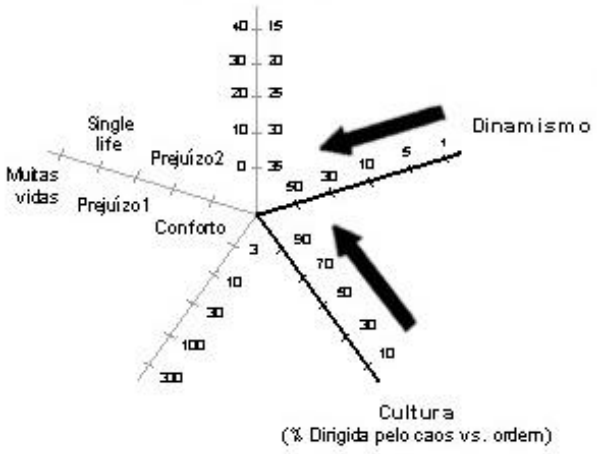

Figura 34: Representação da interferência entre o fator Dinamismo e Cultura (Soares, 2007).

Por outro lado, uma conclusão apresentada no trabalho de pesquisa de Soares (2007) indica que alguns fatores isoladamente são determinantes na escolha e na redução de riscos de adoção de métodos de desenvolvimento de software. Por exemplo, o fator *Criticalidade* sugere fortemente a adoção de MDP, uma vez que os softwares com essas características são elaborados para serem utilizados em ambientes com riscos de vida ou alto nível de prejuízo financeiro. Desta forma, esses fatores necessitam de maior controle e gerência sobre os processos de desenvolvimento.

O fator *Tamanho* da equipe por sua vez induz a utilização de MDP, uma vez que requer maior controle sobre as atividades quando se está utilizando de um número maior de recursos humanos nas tarefas do desenvolvimento de software.

Uma equipe formada na sua maior por profissionais com nível de habilidade - 1 e 1B irá determinar o fator crítico *Competência Pessoal* e consequentemente inviabilizar a aplicação de MA, visto que esses métodos demandam uma maior quantidade de profissionais com nível de habilidade mais alto (Boehm e Turner, 2003).

Diante do exposto, é possível justificar a partir da análise dos gráficos das empresas DPI-03 e DPI-04 a utilização de boas práticas da taxonomia proposta nesse trabalho de pesquisa. Apesar de ser uma abordagem superficial e inicial, a aplicação dos questionários, o desenho gráfico e sua análise, podem fazer parte da primeira melhor prática de desenvolvimento de software sugerido na taxonomia que é o *Diagnóstico Inicial* da empresa proposto no *Processo de Verificação da Qualidade* no item *Gerência da Qualidade*.

A avaliação preliminar do gráfico polar da empresa DPI-03 pode conduzir fortemente para uma análise correlacionada ao concluído por Soares et al. (2008) na relação dos fatores *Competência Pessoal* e *Cultura*, uma vez que no gráfico polar da empresa pode-se observar as linhas dos eixos *Competência Pessoal* e *Cultura* voltadas ao centro sugerindo a adoção de boas práticas voltadas a MA. Mas, conforme já discutido na análise da empresa os fatores *Dinamismo* e *Criticalidade* sugerem uma empresa com características propícias à adoção de soluções de MDP. Principalmente, o fator disposto no eixo *Criticalidade*, onde o alto risco de prejuízo financeiro sugere uma abordagem de um método mais controlado e detalhado. Ou seja, quanto maior a organização, documentação e o controle sobre os requisitos ao longo do processo de desenvolvimento do software, maiores serão as chances de sucesso de qualidade e produtividade para essa empresa.

Diante de um contexto voltado a MDP e do fator *Criticalidade* sendo preponderante na influência de outros fatores de risco, é importante que a empresa DPI-03 adote elementos da taxonomia que a auxiliem na organização de fatores preliminares no desenvolvimento de software, como é o caso dos elementos sugeridos na taxonomia nos processos de *Gerência de Escopo.* Por exemplo, a elaboração inicial, organização e atualização de uma lista de requisitos, e a adoção de *Matrizes de Rastreabilidade* irão influenciar fortemente na qualidade do entendimento dos requisitos gerados e consequentemente sua implementação.

Com a adoção de uma lista de requisitos, para esclarecer e dar uma visão ampla a respeito das funcionalidades do software; e também as matrizes de rastreabilidade, para indicar a interdependência entre os requisitos; é importante que a empresa DPI-03 adote elementos da *Gerência de Casos de Uso* com a documentação constando de diagramas e descrição de casos de uso para formalizar ações, regras e atores que irão compor as funcionalidades do software em questão. Outros elementos podem ser fortemente recomendados, como por exemplo, os elementos que compõem o *Processo de Desenvolvimento Iterativo na Transição / Integração e Testes*. Tais elementos são recomendados para maior planejamento e execução de testes, como também o controle de aceite do usuário e o registro de ocorrências de erros e melhorias.

Com isso, entende-se que a empresa DPI-03 deve iniciar um processo organizado de documentação dos requisitos envolvidos no software minimizando o risco de criticalidade com relação a prejuízos financeiros.

Cabe uma avaliação da taxonomia para cada caso, estudada pelo gestor de desenvolvimento, para que sejam aplicadas de forma correta após o diagnóstico inicial.

A empresa DPI-04 sendo avaliada isoladamente no seu fator de risco *Cultura*, estaria induzida à adoção de boas práticas de MA, o mesmo resultando se fosse avaliada apenas pela análise do eixo *Dinamismo*. Porém a análise conjunta dos eixos *Dinamismo* e *Cultura* sugerem a adoção de práticas de MDP, uma vez que o desenho do gráfico do eixo *Cultura* sugere uma equipe ainda em formação. Outro fator que pode ser acrescentado na análise conjunta de eixos é o fator *Competência Pessoal*, que pelo desenho gráfico pode-se observar uma equipe ainda imatura de conhecimentos que possa obter sucesso com a adoção de práticas originadas de MAs.

Assim como a empresa DPI-03, é fortemente recomendado à DPI-04 buscar na taxonomia de boas práticas elementos que a auxilie na condução do desenvolvimento de software com base em práticas organizadas e formais, ou seja, práticas de MDP. Por apresentar uma equipe com pouca experiência, as práticas iniciais de MDP, além de fomentar a organização da empresa com relação aos processos de software, poderá também auxiliar na formação e na cultura da equipe com relação às práticas organizadas de desenvolvimento de software.

As práticas sugeridas para a empresa DPI-04 perpassam pela adoção de elementos da taxonomia do *Processo de Desenvolvimento Iterativo*, onde a empresa poderá praticar a elaboração formal de documentos iniciais de escopo e avaliação de esforço, vistos na taxonomia como elementos gerados no item *Concepção/Requisito*. Passando por práticas de planejamento de cronograma inicial encontrados na taxonomia no *Processo de Verificação de Qualidade* no item *Gerência de Tempo*. Bem como, a adoção de elementos da *Gestão de Configuração e Mudança de Escopo*, com práticas de documentação das alterações de escopo solicitadas pelo usuário e que estão representados na taxonomia no item *Definição de Documentos de Alteração de Escopo*.

Com isso, a empresa DPI-04, de forma simples, poderá utilizar um conjunto pequeno de elementos da taxonomia que irá auxiliá-la na obtenção de um fluxo organizado de trabalho de forma a influenciar fortemente na prestação de serviços de software de qualidade.

A visão de dois mini processos ilustrativos contendo elementos da taxonomia sugeridos para representar as explicações acima para o caso da empresa DPI-04 podem ser vistos conforme o sugerido nas **Figura 35 e 36**.

As duas representações gráficas dos processos não foram baseadas ou inferidas com base no gráfico polar, mas foram desenhadas com base na experiência profissional do autor do trabalho e servem como um exemplo de possíveis processos que podem ser construídos levando-se em consideração ou mesmo, incluindo elementos da taxonomia proposta. As representações não pretendem esgotar todos os processos possíveis com relação ao desenvolvimento de software, mas servem como exemplo para ilustrar a construção de uma proposta de mini processo aderente a realidade de uma MPE.

A **Figura 35** apresenta um fluxo inicial da concepção de um projeto de software envolvendo elementos da taxonomia para a elaboração de documento inicial de escopo e esforço, além da geração do cronograma de atividades. Esse fluxo poderá auxiliar a empresa DPI-04 na organização inicial do projeto e na formalização das requisições do Cliente. Os elementos da taxonomia podem ser identificados no fluxo a partir das setas tracejadas.

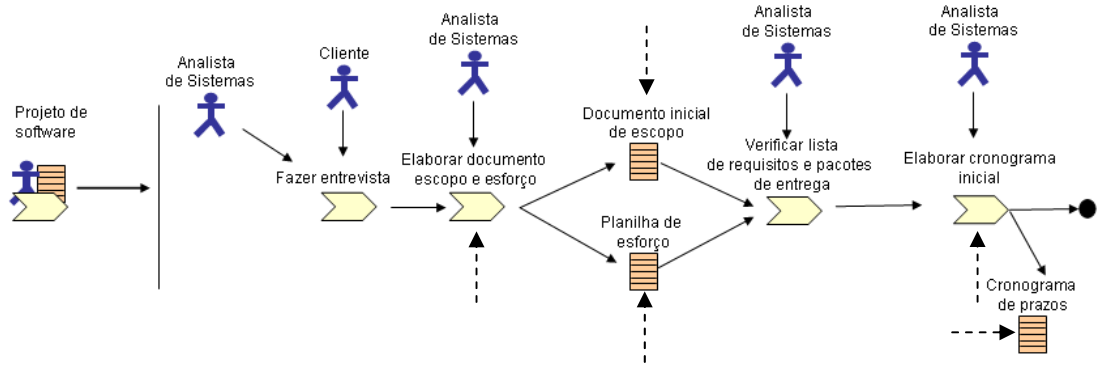

Figura 35: Fluxo de atividades iniciais de projeto baseado na notação SPEM (OMG, 2002).

A **Figura 36** apresenta um fluxo de solicitação de alterações de requisitos de software por parte do Cliente e um grupo de atividades do Analista de Sistemas para avaliar o impacto dessas solicitações e a sua formalização, documentação e replanejamento do cronograma de prazos de entrega das atividades diante das novas solicitações. Os elementos da taxonomia podem ser identificados no fluxo a partir das setas tracejadas.

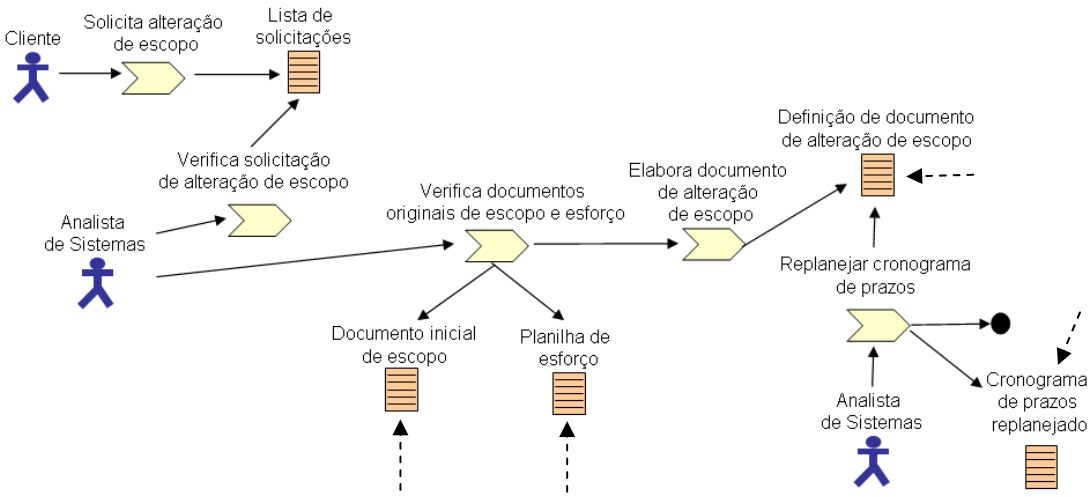

Figura 36: Fluxo de atividades de alteração de escopo baseado na notação SPEM (OMG, 2002).

Apesar dos dois processos desenhados para as empresas da pesquisa e da identificação de elementos da taxonomia de boas práticas que pudessem ser aplicados e utilizados pelas empresas na busca de melhoria de qualidade no desenvolvimento de software, os dois processos não foram implantados ou praticados pelas empresas.

6 CONCLUSÕES

Este trabalho apresentou uma proposta de taxonomia de boas práticas de ES para desenvolvimento de software e procurou justificar seu uso através de diagramas de influência, da literatura técnica especializada, da experiência profissional do autor e das pesquisas de campo efetuadas na consultoria do grupo GPES-DPI-UFV em duas empresas do APL TI-Viçosa, em 2007 e 2008.

A principal contribuição deste trabalho consiste em:

- formulação de uma taxonomia de elementos de boas práticas relacionados e analisados a partir de diagramas de influência sendo um produto de consulta proposto para micro-empresas de desenvolvimento de software;

- indicação de adoção de elementos da taxonomia de boas práticas de desenvolvimento de software para MPEs com base em perfil de risco.

Os principais objetivos propostos inicialmente foram alcançados, uma vez que a revisão bibliográfica auxiliou no suporte teórico a respeito de boas práticas utilizadas atualmente no mercado de software, além de ter permitido o entendimento sobre taxonomias. Essa fundamentação passou a dar o suporte básico para a construção da taxonomia proposta no trabalho de pesquisa. O principal objetivo da pesquisa também foi alcançado ao ser apresentada uma taxonomia com principais elementos de boas práticas em desenvolvimento de software e sua relação, tanto nas figuras da taxonomia quanto nos diagramas de influência que passaram a apresentar a iteração entre alguns elementos e problemas com software.

Foi também atingido o objetivo de apresentar um diagnóstico inicial de duas empresas e de forma ainda preliminar. O diagnóstico de perfil de risco serviu como base para a escolha de alguns elementos na taxonomia, de forma a auxiliar essas empresas na obtenção de melhoria de qualidade e organização do processo de desenvolvimento de software. Com isso, buscou-se dar amparo às empresas na escolha de elementos da taxonomia com base na análise técnica de características da empresa, da equipe, do projeto e de requisitos envolvidos.

O anteprojeto apresentado inicialmente sugeria a criação de um processo, seja adaptado ou customizado, com base na análise de características técnicas e medição de indicadores empresarias a fim de minimizar riscos de adoção de estratégias de desenvolvimento. Apesar disso, o estudo da consultoria do GPES-DPI-UFV verificou que as empresas do APL-TI-Viçosa estão inseridas no grupo de empresas consideradas como micro-empresas e com as mesmas dificuldades das micro empresas do cenário nacional, ou seja, investimentos focados em linguagens de programação e desenvolvimento e não em ES, inexistência de processos organizados e aplicados, recursos humanos e financeiros limitados, baixo nível de conhecimento no domínio gerencial e inexistência de políticas de curto ou longo prazo que auxiliem na melhoria dos processos de desenvolvimento. Diante do contexto encontrado, o trabalho foi reconduzido e direcionado para apresentação da justificativa do uso de boas práticas de software, mais facilmente adotáveis pelas micro-empresas dentro da sua realidade, preparando-as com passos mais curtos para uma futura adoção de processos organizacionais de desenvolvimento de software. Os itens do anteprojeto original foram transportados para o início deste trabalho de pesquisa nas seções *1.1 O problema e sua importância*, *1.2 Justificativa para desenvolvimento do tema* e *1.3 Objetivos da dissertação* na intenção de explicar e justificar o novo foco da pesquisa.

O trabalho não pretende esgotar o assunto e é um direcionamento inicial para a construção de uma taxonomia mais completa. Além disso, como perspectiva futura seria importante a avaliação de outros contextos empresariais, melhorar o diagnóstico atual das empresas com foco em um gráfico polar mais completo que pudesse levar também em consideração questões de projetos implementados pelas empresas e relacionar esse novo diagnóstico ao uso das taxonomias. Importante também seria um trabalho de pesquisa que defina os formatos para os artefatos gerados na taxonomia apresentada, para servir de objeto de consulta para as empresas utilizarem como padrão de documentação de software. Como por exemplo, a definição de formato para os documentos iniciais de escopo, de esforço, os planos de testes, os *checklists*, a planilha de registro de ocorrências, entre outros.

APÊNDICE A

A.1 Taxonomia com a definição dos elementos do Modelo de Processo de Desenvolvimento Iterativo

CONCEPÇÃO / REQUISITO	
Elemento	***Definição***
Documento inicial de escopo	Definir a declaração do escopo do projeto permite descrever, em detalhes, as entregas do projeto e o trabalho necessário para criar essas entregas. A declaração do escopo do projeto também fornece um entendimento comum do escopo do projeto a todas as partes interessadas descrevendo os principais objetivos do projeto. Além disso, permite que a equipe do projeto realize um planejamento mais detalhado, orienta o seu trabalho durante a execução e fornece a linha de base para avaliar solicitações de mudanças ou trabalho adicional e verificar se estão contidos dentro ou fora dos limites do projeto.
Planilha de tamanho e esforço – FPA	Definir a planilha contendo os pontos de função calculados a partir dos requisitos funcionais e não funcionais contidos no documento de escopo. Pode por variação informar o volume de horas que serão gastos na implementação de cada tarefa do projeto, como na elaboração de casos de uso, planos de testes, especificações técnicas, horas gastas com codificação, testes e implantação do sistema ou da funcionalidade desenvolvida.

ELABORAÇÃO / PROJETO	
Elemento	***Definição***
Definição de padrões de documentos	Definir o documento de padrões do projeto apresentando uma visão abrangente da arquitetura do sistema e as diferentes visões arquiteturais para ilustrar os diversos aspectos do sistema. Sua intenção é capturar e transmitir as decisões significativas do ponto de vista da arquitetura que foram tomadas em relação ao sistema. Além da definição de padrões de documentos e convenções de modelagem e projeto que irão ser utilizados ao longo do processo de desenvolvimento do software.

CONSTRUÇÃO / CÓDIGO, TESTE DE UNIDADE	
Elemento	***Definição***
Checklist de código fonte e padrões do sistema	Elaborar e executar um conjunto de procedimentos de verificação de padrões de código fonte, de padrões de tela, de sistema, e de regras simples como validações de campos.
Checklist de regras de negócio	Elaborar e executar um conjunto de procedimentos de verificação das principais regras de negócio envolvidas nas funcionalidades do sistema com base nos casos de uso desenvolvidos.
Testes funcionais / desenvolvedor	Executar testes de unidade tomando como base os checklists de código fonte, padrões do sistema e regras de negócio envolvidas no sistema.

TRANSIÇÃO / INTEGRAÇÃO	
Elemento	***Definição***
Documentação / Planejamento de testes	Item que concentra elementos relacionados às atividades de testes
Plano de testes analista	Elaborar e preencher o documento de planejamento de testes com os passos para testar as funcionalidades envolvidas e executar esse plano no momento do aceite nos testes do analista de sistemas.
Plano de testes do usuário	Elaborar e preencher o documento de planejamento de testes com os passos para testar as funcionalidades envolvidas e executar esse plano no momento do aceite nos testes do usuário.
Aceite	Item que concentra elementos relacionados a aceite
Testes e aceite do analista	Executar testes com acompanhamento de plano de testes do analista com o devido aceite formal registrado no plano de testes.
Testes e aceite do usuário	Executar testes com acompanhamento de plano de testes do usuário com o devido aceite formal registrado no plano de testes.
Registro de ocorrência de testes	Registrar e classificar as ocorrências encontradas nos testes de aceite de analista e de usuário.

A.2 Taxonomia com a definição dos elementos do Modelo de Processo de Gestão de Requisitos

GERÊNCIA DE ESCOPO	
Elemento	***Definição***
Lista de requisitos	Elaborar uma lista dos requisitos funcionais e não funcionais identificados e com uma descrição sucinta.
Matriz de rastreabilidade	Item que concentra elementos relacionados à rastreabilidade de funcionalidades ou características desejadas através de seu relacionamento.
Requisitos funcionais X Requisitos funcionais	Elaborar uma matriz relacionando requisitos funcionais com os mesmos requisitos funcionais, com o objetivo fim de estabelecer sua interdependência.
Requisitos funcionais X Requisitos não funcionais	Elaborar uma matriz relacionando requisitos funcionais com requisitos não funcionais, com o objetivo fim de estabelecer sua interdependência.
Casos de uso X Requisitos funcionais	Elaborar uma matriz relacionando casos de uso e requisitos funcionais, com o objetivo fim de estabelecer sua interdependência.
Casos de uso X Requisitos não funcionais	Elaborar uma matriz relacionando casos de uso e requisitos não funcionais, com o objetivo fim de estabelecer sua interdependência.

GERÊNCIA DE CASOS DE USO	
Elemento	***Definição***
Estórias do usuário	Elaborar e preencher documento para que o usuário possa discorrer a respeito das funcionalidades envolvidas no sistema, criando cenários com situações de utilização desejadas e que deveriam estar presentes no sistema a ser desenvolvido.
Lista de atores	Elaborar e preencher documento com lista de atores envolvidos nas funcionalidades do sistema, bem como uma descrição sucinta a respeito do papel, das características do ator, tais como: função, principais atividades, permissão de acesso e suas responsabilidades, entre outros.
Diagrama de casos de uso	Elaborar e preencher documento com a especificação dos relacionamentos entre casos de uso e os atores.
Descrição de casos de uso	Elaborar e preencher documento com a descrição dos casos de uso contendo os principais fluxos e subfluxos, detalhando passo a passo cada função do sistema. Além dos fluxos, é importante que sejam detalhadas as pré-condições de execução das funções.

A.3 Taxonomia com a definição dos elementos do Modelo de Processo de Arquitetura

DEFINIÇÃO DE TECNOLOGIAS PARA DESENVOLVIMENTO	
Elemento	***Definição***
Linguagem de programação	Elaborar documento com especificação da linguagem a ser utilizada no projeto de software, bem como os padrões que deverão ser utilizados na codificação, na estrutura de definições de variáveis, definição e retorno de funções, nomenclaturas, entre outros.
Banco de Dados	Elaborar documento com especificação do banco de dados a ser utilizado no projeto de software, bem como os padrões que deverão ser utilizados na concepção da estrutura física do banco, tais como, padrão de nomenclatura de campos e tabelas, padronização de tipos de campos, nomenclatura de chaves primárias e estrangeiras. Além da obrigatoriedade de uso de comentários para tabelas e campos.

DEFINIÇÃO DE FERRAMENTAS DE MODELAGEM	
Elemento	***Definição***
Definição de ferramentas de modelagem	Definir o uso de uma ferramenta de modelagem de banco de dados utilizada, seguindo as normas definidas no documento de padrão de banco de dados.

PROTOTIPAÇÃO	
Elemento	***Definição***
Prototipação	Elaborar documento para formatação de protótipos do sistema e para cada sistema elaborado, desenvolver o protótipos nos padrões estabelecidos e formalizar junto ao usuário o seu aceite.

REUSO E COMPONENTIZAÇÃO	
Elemento	***Definição***
Reuso e componentização	Estabelecer normas para reutilização de funções genéricas a serem utilizadas no sistema e criar repositório com finalidade de reutilização de objetos nos diversos projetos da empresa.

DEFINIÇÃO DO DOCUMENTO DE ARQUITETURA	
Elemento	***Definição***
Conceituação	Descrever uma definição da arquitetura levando em consideração os principais mecanismos e as convenções de modelagem para o sistema, além de verificar documentações de padrão pré-estabelecidas de forma organizacional e discorrer a respeito desses padrões no projeto para aumentar a visibilidade e compreensão de seus, sejam eles usuários ou técnicos.
Definição de metas e restrições de arquitetura	Estabelecer os impactos dos requisitos definidos para o software, tais como privacidade, segurança, portabilidade, distribuição e reutilização. Definir estratégias de design que poderão ser usado nos protótipos e nas telas finais do software, bem como, métodos de acesso a sistemas legado, métodos de conexão a hardware (*drivers* utilizados), métodos de acesso a bancos de dados (nome do banco, *driver* utilizado, permissões necessárias), sistemas operacionais, entre outros.
Definição de visões do projeto	Elaborar diferentes visões do software com o objetivo aumentar o prisma de observação para melhor entender os problemas envolvidos. As visões devem ser elaboradas de modo a se definir os diferentes papeis dos envolvidos no sistema e de como cada componente do software pode ser observado por perspectivas distintas, a fim de melhor esclarecer suas inter-relações.
Visão lógica	Utilizar dos diagramas de classes para definir claramente as questões de negócio e suas principais regras. Além dos principais atributos envolvidos e seus métodos. Está visão está fortemente relacionada aos requisitos funcionais que o software deve atender.

Visão de implementação	Discorrer sobre a organização dos módulos estáticos do software para facilitar a distribuição do trabalho de implementação e manutenção entre os membros da equipe de desenvolvimento, considerando aspectos de reuso, sub-contratação de desenvolvimento de software e aquisição de componentes terceiros.
Visão de processos	Discorrer sobre os principais pontos que o software irá relacionar no momento da sua execução, tais como o grau de confiabilidade exigido, os mecanismos de comunicação com outros software, desempenho e tolerância a falhas.
Visão de implantação	Discorrer a respeito das características necessárias para a disponibilização do software para uso. Esses aspectos devem levar em consideração questões de infra-estrutura, a alteração de ambientes de desenvolvimento para produção e principais configurações exigidas nos diferentes ambientes.
Visão de caso de uso	Discorrer a respeito de como os casos de uso críticos são executados pelo sistema dando ênfase ao seu relacionamento com tarefas, objetos e nós. Pode ser visto como o modelo de casos de uso do software.
Definição das características do produto e processo	Elaborar documento para demonstrar as principais características do produto de software que será produzido, dos seus componentes, subsistemas, arquitetura, bem como a lista das principais características de qualidade esperadas para o produto. Se faz importante elaborar juntamente com as características do produto, a lista e a definição das principais ações que irá fazer parte do processo de concepção, execução até o ponto de colocar esse produto disponível para uso, ou em produção.

A.4 Taxonomia com a definição dos elementos do Modelo de Processo de Modelagem / Criação de Modelos Visuais / Adoção de UML

DIAGRAMAS ESTRUTURAIS	
Elemento	***Definição***
Diagramas de objetos	Elaborar segundo os conceitos da UML, os diagramas de objetos para permitir a avaliação do perfil do software em determinado momento de sua execução.
Diagramas de classes	Elaborar segundo os conceitos da UML, os diagramas de classes para a representação das relações entre essas e para gerar um modelo dos objetos utilizados no software.
Diagramas de componentes	Elaborar segundo os conceitos da UML, os diagramas de componentes para a representação da organização das classes envolvidas.
Diagramas de instalação	Elaborar segundo os conceitos da UML, os diagramas de instalação a fim de se estabelecer a representação física de infra-estrutura de hardware e software envolvidos com o produto a ser desenvolvido.
Diagramas de pacotes	Elaborar segundo os conceitos da UML, os diagramas de pacotes para representar a inter-relação entre os módulos do software.
Diagramas de estruturas	Elaborar segundo os conceitos da UML, os diagramas de estrutura para descrever a relação interna entre os elementos, ou seja, a representação da colaboração entre elementos como classes, interfaces ou estruturas software

DIAGRAMAS COMPORTAMENTAIS	
Elemento	***Definição***
Diagramas de casos de uso	Elaborar segundo os conceitos da UML, os diagramas de casos de uso para compor uma estrutura diagramática das principais funcionalidades do sistema e os atores envolvidos.
Diagrama de transição de estados	Elaborar segundo os conceitos da UML, os diagramas de estados para representar os diversos estados em que o objeto poderá se encontrar no decorrer da execução do sofware.
Diagrama de	Elaborar segundo os conceitos da UML, os diagramas de atividades para

atividades	representar os fluxos de controle das atividades inerentes no processo de execução do sofware.

DIAGRAMAS DE ITERAÇÃO	
Elemento	***Definição***
Diagrama de iteratividade	Elaborar segundo os conceitos da UML, os diagramas de iteratividade para representar a seqüência de execução de atividades.
Diagrama de colaboração	Elaborar segundo os conceitos da UML, os diagramas de colaboração para representar os objetos envolvidos no sistema e os seus relacionamento, bem como as principais mensagens transmitidas entre eles
Diagrama de tempo	Elaborar segundo os conceitos da UML, os diagramas de tempo para identificar o comportamento dos objetos e sua interação em uma escala de tempo.
Diagrama de seqüência	Elaborar segundo os conceitos da UML, os diagramas de seqüência para representar a seqüência das mensagens transmitidas entre os objetos.

A.5 Taxonomia com a definição dos elementos do Modelo de Processo de Verificação e Qualidade

GERÊNCIA DE ATIVIDADES	
Elemento	***Definição***
Definição de EAP / WBS	Elaborar a Estrutura Analítica do Projeto (EAP) ou *Work Breakdown Structures* (WBS) com o objetivo de se estruturar hierarquicamente e com detalhes todas as atividades envolvidas no processo do software. A importância de uma EAP bem estruturada poderá auxiliar em processos como: definição de atividades, planejamento de recursos, estimativa de custos, elaboração do orçamento, planejamento e gerenciamento de riscos, verificação de escopo e controle de alterações de escopo.
Definição da alocação de recursos (papeis e responsabilidades)	Elaborar planilha baseada nos pacotes de entrega e na EAP com a alocação dos recursos e suas responsabilidades ao longo do tempo de execução do projeto de software.

GERÊNCIA DE TEMPO	
Elemento	***Definição***
Gerência de tempo	Item que concentra elementos relacionados às atividades de gerenciamento de cronogramas de prazo.
Cronogramas	Cronogramas a serem desenvolvidos.
Definição de cronograma inicial	Elaborar cronograma inicial relacionando as atividades e os pacotes de entrega, definindo principalmente atributos de atividades que possam auxiliar no controle do cronograma. Atributos de atividades tais como: identificados, nome da atividade, percentual realizado, data de início e fim de execução, atividades precedentes e recursos envolvidos.
Replanejamento de cronograma	Elaborar o replanejamento de cronograma com base no cronograma inicial, nas alterações de escopo, caso aconteçam, ou nos atrasos ou adiantamentos de atividades realizadas.

GERÊNCIA DE QUALIDADE	
Elemento	***Definição***
Diagnóstico inicial	Elaborar documento de diagnóstico inicial da equipe, do projeto a ser desenvolvido e da empresa a fim de se estabelecer uma análise de risco para identificar quais boas práticas serão importantes na aplicação do projeto a ser desenvolvido.
Controle de medições e indicadores	Utilizar o registro de ocorrências de testes para efetuar um controle de medição do processo de desenvolvimento e apresentar indicadores do andamento do projeto a fim de se obter uma ferramenta capaz de auxiliar na tomada de decisão das ações a serem efetuadas de acordo com os problemas encontrados.
Programação por pares	Promover, quando possível e segundo conceitos de métodos ágeis, a programação em pares a fim aumentar o poder de controle sobre o código que está sendo construído.
Pontos de controle	Efetuar periodicamente reuniões de ponto controle e registrar formalmente as ocorrências encontradas, as principais decisões, responsabilidade e pendências discutidas.

A.6 Taxonomia com a definição dos elementos do Modelo de Gestão de Configuração e Mudança de Escopo

GERÊNCIA DE CONFIGURAÇÃO	
Elemento	***Definição***
Controle de versões	Item que concentra elementos da gerência de configuração para auxiliar em controle de versões de artefatos, códigos fontes e toda documentação referente ao projeto.
Uso de ferramentas de controle	Utilizar ferramenta computacional para controle de versão de artefatos que compõem o projeto do software.

GERÊNCIA DE MUDANÇA DE ESCOPO	
Elemento	***Definição***
Definição de documento de alteração de escopo	Elaborar documento para formalizar a alteração de escopo solicitada e solicitar formalmente seu aceite.
Lista de requisitos funcionais	Elaborar uma lista dos novos requisitos funcionais representados pelos casos de uso identificados. A lista poderá ter informações do tipo: identificador do caso de uso, descrição, tipo (fluxo principal, subfluxo, fluxo alternativo ou relatório) e prioridade (essencial, desejável ou opcional).
Lista de requisitos não funcionais	Elaborar uma lista dos novos requisitos não funcionais representados. A lista poderá ter informações do tipo: identificador do requisito, descrição, tipo (segurança, desempenho, manutenabilidade, usabilidade e restrição de desenho), prioridade (essencial, desejável ou opcional) e complexidade (alta, média ou baixa).
Lista de requisitos excluídos	Elaborar lista com requisitos funcionais e não funcionais excluídos com a alteração de escopo.
Definição de estimativas de custo e prazo	Elaborar nova estimativa de custo e prazo e efetuar o replanejamento de cronograma.

REFERÊNCIAS BIBLIOGRÁFICAS

ABDEL-HAMID, T. K., MADNICK, S. E. (1991) Software Project Dynamics: an Integrated Approach. Englewood Cliffs: Prentice Hall, 1991. 264 p.

ABNT. (2001). Associação Brasileira de Normas Técnicas. “Série ISSO 9000:2000: Sistemas de Gestão da Qualidade”. ABNT, 2001.

ABRAN, A. and Moore, J.W. (2004). Swebok: Guide to the Software Engineering Body of Knowledge. IEEE Computer Society: Los Alamitos, California.

AGGARWAL, R. (1998) Small Firm Survival and Technological Activity. Small Business Economics 11, 1998.

AMBRÓSIO, B. G. M.Sc. (2008). Universidade Federal de Viçosa, abril de 2008. Modelagem da fase de requisitos em processos de desenvolvimento de software: uma abordagem utilizando dinâmica de sistemas.

BOEHM, B. (1996). In, H.: Identifying Quality Requirements Conflicts. IEEE Software, March: 25–35, 1996.

BOEHM, B. and TURNER, R. (2003). Using risk to balance agile and plan-driven methods. IEEE Computer, 36(6):57–66.

BOEHM, B. and TURNER, R. (2004). Balancing Agility and Discipline: A Guide for the Perplexed. Addison-Wesley, Boston.

BOSE, P. (1995): Conceptual design model based requirements analysis in the Win-Win framework for concurrrent requirements engineering. IEEE Workshop on Software Specification and Design (IWSSD), 1995.

BRASIL (1999). Lei n° 9.841, de 5 de outubro de 1999. Institui o Estatuto da Microempresa e da Empresa de Pequeno Porte, dispondo sobre o tratamento jurídico diferenciado, simplificado e favorecido previsto nos arts. 170 e 179 da Constituição (1988) e dá outras providências. Diário Oficial da União, Brasília, 06 out. 1999.

BRASIL (2005). Decreto n° 5.028, de 31 de março de 2005. Altera os valores dos limites fixados nos incisos I e II do art. 2º da Lei 9.841 de 5 de outubro de 1999, que instituiu o Estatuto da Microempresa e da Empresa de Pequeno Porte. Diário Oficial da União, Brasília, 01 abr. 2005.

CALERO C., RUIZ F., and PIATTINI M., Eds. (2006), Ontologies for Software Engineering and Software Technology, Springer, Berlin, Germany, 2006.

CAMPOS, M. L. A. (2006). Integração de ontologias: o domínio da bioinformática e a problemática da compatibilização terminológica. In: ENANCIB, 7., 2006, Marília. Anais... Marília: ANCIB, UNESP. Disponível em: <http://www.portalppgci.marilia.unesp.br/enancib/viewpaper.php?id=163>. Consultado em: 20/02/2009.

CAMPOS, M. L. A. , MARCONDES, C. H., OLIVEIRA, L. L., COSTA, L. C. da, CAMPOS, L. M., MALHEIROS, L. R. (2007). Ontologias: representando a pesquisa na área através de mapa conceitual. In: VIII ENANCIB, 2007, Salvador. VIIIENANCIB - En

CHANDRASEKARAN, B., JOSEPHSON, J.R., BENJAMINS, V.R. (1999): What Are Ontologies, and Why Do we Need Them?. IEEE Intelligent Systems Vol. 14, Issue 1. 1999. 20-26.

COCKBURN, A. (2000). Selecting a projects methodology. IEEE Software, 17(4):64–71.

COCKBURN, A. (2002). Agile Software Development. Addison-Wesley, Boston.

COSTA, T. S. P., GUSMÃO, C. M. G. (2008). Definição de Ontologia para Identificação de Riscos de Projetos de Software. In: Seminário de Pesquisa em Ontologia no Brasil, 2008, Niterói. Seminário de Pesquisa em Ontologia no Brasil, 2008.

CMU (2002). CARNEGIE MELLON UNIVERSITY. Capability Maturity Model Integration (CMMI), Version 1.1: CMMI for Software Engineering (CMMI-SW, V1.1) - Staged Representation. Pittsburg, 2002.

CYC. (2009) OpenCyc.org: Formalized Common Knowledge. Cycorp, USA. Disponível em: < http://www.opencyc.org68> Consultadodo em: 21/02/2009.

DENNIS, A., WIXOM, B. H. (2005) Análise e Projeto de Sistemas. Rio de Janeiro: LTC, 2005.

DERIDDER, D. (2002): A Concept-Oriented Approach to Support Software Maintenance and Reuse Activities. 5th Joint Conference on Knowledge-Based Software Engineering (JCKBSE), Maribor, Slovenia, September 2002.

DIAS, M.G., ANQUETIL, N., DE OLIVEIRA, K.M. (2003): Organizing the Knowledge Used in Software Maintenance. Journal of Universal Computer Science, 9(7): 641–658, 2003.

DING, Y., FOO, S. (2002a). Ontology research and development. Part 1 - a review of ontology mapping and evolving. Journal of Information Science, v. 28, n. 2, p. 123–136, 2002.

DING, Y., FOO, S. (2002b). Ontology research and development. Part 2 - a review of ontology generation. Journal of Information Science, v.28, n.5, p. 375-388, 2002.

FALBO, R.A., GUIZZARDI, G., DUARTE, K.C. (1992): An Ontological Approach to Domain Engineering. In Procedings of 14th International Conference on Software Engineering and Knowledge Engineering (SEKE), Ischia, Italy, July 1992, pp. 351–358.

FALBO, R.A., CRUZ, A.C., MIAN, P.G., BERTOLLO, G., BORGES, F. (2003): ODE: Ontology-based software Development Environment. IX Argentine Congress on Computer Science (CACIC), La Plata, Argentina, 6–7 October 2003.

FALBO, R., RUY, F., PEZZIN, J. and R., D. M. (2004). Ontologias e ambientes de desenvolvimento de software semântico, IV Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento, JIISIC'2004 1: 277–292.

FELICÍSSIMO, C. H., SILVA, L. F. da, BREITMAN, K. K., LEITE, J. C. S. do P. (2003). Geração de Ontologias Subsidiada pela Engenharia de Requisitos. In Proceedings of WER'2003. pp.255~269

FENSEL, D. (2004). Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce. Second Edition, Springer-Verlag, Berlin, Heidelberg.

FILHO, W. P. (2003). Engenharia de Software: Fundamentos, métodos e padrões. LCT. Segunda edição.

GIRARDI, R., FARIA, C. (2003): A Generic Ontology for the Specification of Domain Models. In Proceedings of 1st International Workshop on Component Engineering Methodology (WCEM'03) at Second International Conference on Generative Programming and Component Engineering, Erfurt, Germany, 2003.

GOMEZ-PEREZ, A. (1999). Tutorial on Ontological Engineering, Internacional Joint Conference on Artificial Intelligence - IJCAI´1999.

GONZÁLEZ-PÉREZ, C., Henderson-Sellers, B. (2006): An Ontology for Software Development Methodologies and Endeavours. In Ontologies for Software Engineering and Technology, Springer-Verlag, Berlin, chapter 4 (2006).

GRUBER T. R. (1993) Toward principles for the design of ontologies used for knowledge sharing. In: Guarino N, Poli R (eds) International Workshop on Formal Ontology in Conceptual Analysis and Knowledge Representation. Padova, Italy. (Formal Ontology in Conceptual Analysis and Knowledge Representation) Kluwer Academic Publishers, Deventer, The Netherlands.

GRUBER, T. R. (1995). Toward Principles for the Design of Ontologies used for Knowledge Sharing. Int. J. Human-Computer Studies, v. 43, n. 5/6, 1995.

GUARINO, N. (1998). Formal Ontologies and Information Systems. In: FIRST INTERNATIONAL CONFERENCE (FOIS), 1., 1998, Trento, Itália. Anais... Trento: IOS Press, 1998.

HILERA, J.R., SÁNCHEZ-ALONSO, S., GARCÍA, E., DEL MOLINO, C.J. (2005): OntoGLOSE: A Light-weight Software Engineering Ontology. 1st Workshop on

Ontology, Conceptualizations and Epistemology for Software and Systems Engineering (ONTOSE), Alcalá de Henares, Spain, 9–10 June 2005.

ISO/IEC 15504. (2003) International Organization for Standardization. “ISO/IEC 15504: Information Technology – Process Assessment, Part 1 to Part 5”. ISO/IEC Intermediate Report, 2003.

JONES, C. (1996). Patterns of Software Systems Failure and Sucess. Londres: International Thompson Computer Press. 1996.

KITCHENHAM, B.A., TRAVASSOS, G.H., MAYRHAUSER, A., NIESSINK, F., SCHNEIDEWIND, N.F., SINGER, J., TAKADA, S., VEHVILAINEN, R., YANG, H. (1999): Towards an Ontology of Software Maintenance. Journal of Software Maintenance: Research and Practice, 11(6): 365–389, 1999.

KRUCHTEN, P. (2001). The Rational Unified Software Process: an Introduction (2nd. edition). Upper Saddle River, NJ, Addison-Wesley, 2001.

LARBURU, I.U., PIKATZA, J.M., SOBRADO, F.J., GARCÍA, J.J., LÓPEZ, D. (2003): Hacia la implementación de una herramienta de soporte al proceso de desarrollo de software. Workshop in Artifificial Intelligence Applications to Engineering (AIAI), San Sebastián, Spain, 2003.

LIN, S., LIU, F., LOE, S. (2003): Building A Knowledge Base of IEEE/EAI 12207 and CMMI with Ontology. Sixth International Protégé Workshop, Manchester, England, 7–9 July 2003.

MENDES, O., ABRAN, A. (2005): Issues in the development of an ontology for an emerging engineering discipline. First Workshop on Ontology, Conceptualizations and Epistemology for Software and Systems Engineering (ONTOSE), Alcalá de Henares, Spain, 9–10 June 2005.

MORO, R. Dal, FALBO, R. A. (2008). Uma Ontologia para o Domínio de Qualidade de Software com Foco em Produtos e Processos de Software. In: III Workshop on Ontologies and Metamodeling in Software and Data Engineering, 2008, Campinas. Anais do III WOMSDE, 2008. p. 37-48. Encontro Nacional de Pesquisa em Ciência da Informação. Salvador, 2007.

MCT. (2001) Ministério da Ciência e Tecnologia. Qualidade e Produtividade no Setor de Software Brasileiro, Resultados da Pesquisa 2001. Disponível em: <http://www.mct.gov.br/Temas/info/Dsi/Quali2001/Public2001.htm>. Consultado em: 15/01/2009.

MCT. (2005) Ministério da Ciência e Tecnologia. Qualidade e Produtividade no Setor de Software Brasileiro, Resultados da Pesquisa 2005. Disponível em: < http://www.mct.gov.br/index.php/content/view/3253.html>. Consultado em: 15/01/2009.

MPS.BR (2005). Guia Geral. Associação para Promoção da Excelência do Software Brasileiro – Softex. MPS.BR – 2005.

NOY, N. F., HAFNER C.D. (1997). The State of Art in Ontology Design: A Survey and Comparative Review. AI Magazine, 1997.

OMG. (2002). Objeta Management Group, Software Process Engineering Metamodel Specification (SPEM), formal submission, OMG document number formal/02-11-14, november 2002.

PÁDUA, W. F. (2003) Engenharia de software: fundamentos, métodos e padrões. Rio de janeiro: LTC, 2003.

PAULK, M. C.; CURTIS, B.; CHRISSIS, M. B.; WEBBER, C. V. (1993) Capability maturity model sm for software, version 1.1. Technical report CMU/SEI-93-TR-024, SEI Joint Program Office Hanscom AFB, MA 01731-2116, 1993.

PFLEEGER, S.R. (1998) Software Engineering: Theory and Practice. Upper Saddle River: Prentice-Hall, 1998.

PMI (2004) Project Management Institute. Um guia do conjunto de conhecimentos em gerenciamento de projetos – Guia PMBOK. Terceira edição.

ROCHA, A.R.C., AGUIAR, T.C., SOUZA, J.M.. (1990). Taba: A Heuristic Workstation for Software development. Proceedings of COMPEURO 90, Tel Aviv, Israel, Maio de 1990.

ROCHA, A.R.C., MALDONADO, J.C., and WEBER, K.C. (2001). Qualidade de Software: Teoria e Prática. Prentice Hall, São Paulo, SP, BRASIL.

ROVERE L., RENATA L., ERBER, FABIO S., e HASENCLEVER L. (2000). Industrial and Technology Policies and Local Economic Development: enhancing and supporting clusters. Third Triple Helix International Conference, 26 a 29 de abril de 2000, Hotel Glória, Rio de Janeiro. Disponível em CDROM (PEP/COPPE/UFRJ)

ROYCE, W. (1998). Software Project Management - A Unified Framework. Addison-Wesley, 1998.

RUSSELL S., NORVIG P. (1995). Artificial Intelligence: A Modern Ap-proach, Prentice Hall, 1995.

RUIZ, F., VIZCAÍNO, A., PIATTINI, M., GARCÍA, F. (2004): An Ontology for the Management of Software Maintenance Projects. International Journal of Software Engineering and Knowledge Engineering, 14(3): 323–349, 2004.

SÁNCHEZ, D.M., CAVERO, J.M., MARCOS, E. (2005): An ontology about ontologies and models: a conceptual discussion. First Workshop on Ontology, Conceptualizations and Epistemology for Software and Systems Engineering (ONTOSE), Alcalá de Henares, Spain, 9–10 June 2005.

SBC (2006). Grandes Desafios da Pesquisa em Computação no Brasil. 2006 – 2016. SBC.

SEBRAE (2004). Serviço Brasileiro de Apoio às Micro e Pequenas Empresas. Fatores Condicionantes e Taxa de Mortalidade de Empresas no Brasil. Relatório de Pesquisa. Brasília, ago. 2004.

SEI. (2002) SOFTWARE ENGINEERING INSTITUTE. CMMI for Systems Engineering/Software Engineering (CMMI-SE/SW), Staged Representation. Version 1.1, Technical report CMU/SEI-2002-TR-02. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002.

SENGE, P. (1990) The Fifth Discipline: The Art and Practice of the Learning Organization. New York: Currency Doubleday, 1990. 371 p.

SICILIA, M.A.., CUADRADO, J.J., GARCÍA, E., RODRÍGUEZ, D., HILERA, J.R. (2005): The evaluation of ontological representation of the SWEBOK as a revision tool. In 29th Annual International Computer Software and Application Conference (COMPSAC), Edinburgh, UK, 26–28 July 2005.

SMITH, B. (2000). Ontology: Philosophical and Computational [online]. Disponível em: <http://wings.buffalo.edu/philosophy/faculty/smith/articles/ ontologies.htm>. Consultado em: 07/02/2009.

SOARES, L. (2007). Obtenção de requisitos para customização de processo de desenvolvimento de software. Master's thesis, Universidade Federal de Viçosa, CCE/DPI, dissertação de Mestrado.

SOARES, L. S., BRAGA, J.L., LEAL, A. L. C., SILVA, C. H. O. CAMPOS, J. P. (2008). Risk profile assessment for software development teams: a first step towards best practices adoption. In: CLEI - Conferência Latino-Americana de Informática, 2008, Santa Fé, Argentina. Anais do XXIV CLEI. Santa Fé, Argentina: SADIO - Sociedad Argentina de Informática, 2008. v. XXIV. p. 509-518.

SOFTEX (2005). MPS.BR - Melhoria de Processo do Software Brasileiro. Technical report, SOFTEX. Version 1.0.

SOMMERVILLE, I. (2003) Engenharia de Software / Ian Sommerville; tradução André Maurício de Andrade Ribeiro; revisão técnica Kechi Hirama. São Paulo: Addison Wesley, 2003.

TANSALARAK, N., CLAYPOOL, K.T. (2004): XCM: A Component Ontology. Workshop on Ontologies as Software Engineering Artifacts (OOPSLA), Vancouver, Canada, 24–28 October 2004.

TAUTZ, C., VON WANGENHEIM, C. (1998): REFSENO: A Representation Formalism for Software Engineering Ontologies. Fraunhofer IESE-Report No. 015.98/E, version 1.1, October 20, 1998.

TERRA, S.; MICHELY V. (2004) Taxonomia: Elemento Fundamental para a Gestão do Conhecimento, 2004.

VIZCAÍNO, A., ANQUETIL, N., OLIVEIRA, K., RUIZ, F., PIATTINI, M. (2005): Merging Software Maintenance Ontologies: Our Experience. First Workshop on

Ontology, Conceptualizations and Epistemology for Software and Systems Engineering (ONTOSE), Alcala de Henares, Spain, 9–10 June 2005.

VOS, JAN-PETER, KEIZER, JIMME, E HALMAN, JOOP M. (1998). Diagnosing Constraints in Knowledge of SMEs. Technological Forecasting and Social Change 58, 1998.

WAZLAWICK, R. S. (2004). Análise e projeto de sistemas de informação orientados a objetos. Rio de Janeiro: Elsevier, 2004.

YOURDON E. (1997). Death March: Managing Mission Impossible Projects. Upper Saddle River, NJ: Prentice-Hall, 1997.