Universidade Federal de Itajubá

Afonso Celso Soares

LPA: Um Processo Navegável para Desenvolvimento de *Software* Parcialmente Aderente ao SW-CMM Nível 2

Dissertação apresentada à Universidade Federal de Itajubá para obtenção do Título de Mestre em Engenharia Elétrica

Orientadora: Profa. Dra. Lúcia Regina Horta Rodrigues Franco

Co-orientador: Prof. Dr. Mauro de Mesquita Spinola

Itajubá, janeiro de 2004

Ficha catalográfica elaborada pela Biblioteca Mauá –
Bibliotecária Margareth Ribeiro- CRB_6/1700

S676l

Soares, Afonso Celso

LPA : um processo navegável para desenvolvimento de software parcialmente aderente ao modelo CMM nível 2 / por Afonso Celso Soares. -- Itajubá (MG) : [s.n.], 2003.

211 p. : il.

Orientadora : Profa. Dra. Lúcia Regina Horta Rodrigues Franco

Dissertação (Mestrado) – Universidade Federal de Itajubá - Departamento de Eletrônica.

1. CMM. 2. Processo unificado. 3. Qualidade de software. 4. UML. 5. RUP. 6. Processo de desenvolvimento de software. I. Franco, Lúcia Regina Horta Rodrigues, orient. II. Universidade Federal de Itajubá - Departamento de Eletrônica. III. Título.

CDU 004.41(043)

Para a minha querida esposa Bernadete e aos amados filhos Celso e Jamile, pela compreensão durante os eternos momentos de minha ausência.

Agradeço aos meus orientadores, Profa. Lúcia e Prof. Mauro, por acreditarem que este trabalho pudesse ser desenvolvido. Ao Caíque, pelo grande auxílio na identificação do escopo, ao time da Seção de Software Administrativo do Inatel, Clayton, Luciane e Suzana, Flávia e Luziana, por permitirem que ele pudesse ser aplicado e avaliado, e ao Inatel, por me fornecer a infra-estrutura necessária.

“O homem nasceu para aprender, aprender enquanto o tempo lhe permita”.

João Guimarães Rosa

SUMÁRIO

RESUMO

Este trabalho descreve um processo para desenvolvimento de *software* chamado LPA – Levantamento, Planejamento e Acompanhamento, aderente às áreas de processo-chave de requisitos, planejamento e acompanhamento de projeto de *software*, do modelo de qualidade de *software* SW-CMM nível 2, versão 1.1, para ser aplicado em organizações desenvolvedoras de *software* pertencentes ao nível 1 de maturidade deste modelo. Visando facilitar e motivar o usuário quanto à sua utilização, o processo foi desenvolvido segundo o estilo do Processo Unificado Rational, composto das disciplinas de Gestão de Requisitos e Gestão de Planejamento e Acompanhamento, sendo executado com sucesso na ferramenta Internet Explorer versão 5.0.

Palavras-chave: CMM; Processo Unificado; Qualidade de *Software*; UML; RUP; Processo de Desenvolvimento de *Software*.

ABSTRACT

This work describes a software development process called LPA. This acronym stands for Levantamento (Eliciting), Planejamento (Planning) and Acompanhamento (Follow-up). LPA is SW-CMM version 1.1 level 2 compliant to the following Key Process Area: Requirements Management, Software Project Planning and Software Project Tracking and Oversight and can be applied in immature organizations belonging to SW-CMM level 1. Intending to facilitate and motivate the use of the process by the users, the process was developed by using the Rational Unified Process – RUP style. LPA is composed of two disciplines called Requirements Management and Planning and Tracking Management and could be executed with success in Internet Explorer Version 5.0 navigator.

Key-words: CMM; Unified Process; Software Quality; UML; RUP; Software Development Process.

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE SÍMBOLOS, SIGLAS E ABREVIATURAS

ABNT – Associação Brasileira de Normas Técnicas

AM – *Agile Modeling*

CASE – *Computer Aid Software Engineering*

CASE – *Computer-Aid Software Engineering*

CMM – *Capability Maturity Model*

CMMI - *Capability Maturity Model Integration*

CMU – *Carnegie Mellon University*

DoD – *Department of Defense*

DSDM – *Dynamic Systems Development Method*

EUP – *Enterprise Unified Process*

GPA – Gestão de Planejamento e Acompanhamento

GR – Gestão de Requisitos

IEC – International Electrotechnical Commission

IEEE – *Institute of Electrical and Electronics Engineers*

ISO – *International Standard Organization*

ITU – *International Telecommunication Union*

KPA – *Key Process Area*

LPA – Levantamento, Planejamento e Acompanhamento

MCT – Ministério da Ciência e Tecnologia

MCU – Modelo de Casos de Uso

NBR – Norma Brasileira

OMG – *Object Management Group*

OML – *OPEN Modeling Language*

OMT – *Object Modeling Technique*

OOA – *Object-Oriented Analisys*

OOAD – *Object-Oriented Analisys and Design*

OOD – *Object-Oriented Design*

OOSA – *Object-Oriented Systems Analisys*

OOSE – *Object-Oriented Software Engineering*

OOSP – *Object-Oriented Software Process*

OPEN – *Object-Oriented Process, Environment and Notation*

PDA – Plano de Desenvolvimento e Acompanhamento

PGPA – Política da Gestão de Planejamento e Acompanhamento

PGR – Política da Gestão de Requisitos

PI – Pedido do Interessado

PM – Pedido de Mudança

RM – *Requirements Management*

RPW – *Rational Process Workbench*

RUP - *Rational Unified Process*

SDL – *Specification Description Language*

SEI – *Software Engineering Institute*

SEPIN – Secretaria de Política de Informática

SPP – *Software Project Planning*

SPTO – *Software Project Tracking and Oversight*

SW-CMM – *Capability Maturity Model for Software*

UML – *Unified Modeling Language*

XP – Extreme Programming

1 INTRODUÇÃO

Este capítulo faz uma abordagem de alguns processos, modelos e padrões existentes para a melhoria da qualidade em organizações de desenvolvimento de *software* e de alguns problemas que estas organizações enfrentam no seu dia-a-dia com a gerência de projetos. Em seguida, é apresentada a finalidade do presente trabalho, que se trata de um processo para desenvolvimento de *software* chamado LPA, e da forma como ele foi utilizado, para que suas características pudessem ser testadas.

1.1 Contribuições para a Melhoria da Qualidade de *Software*

Com o avanço tecnológico dos sistemas computacionais, as necessidades de *softwares* ficaram cada vez mais intensas. Além disso, estes *softwares* aumentaram em tamanho e complexidade. Todavia, desde que eles passaram a ocupar espaço comercial, os usuários sempre reclamavam da qualidade dos mesmos. Na medida em que se tornavam mais extensos e complexos, sua qualidade deveria se tornar cada vez maior. Para que estes *softwares* pudessem ser produzidos com melhor qualidade, a comunidade da Engenharia de *Software* sentiu a grande necessidade de produzir mecanismos que pudessem melhorar esta qualidade. Além da qualidade intrínseca dos *softwares*, perceberam-se que os projetos também apresentavam outros graves problemas: dificilmente os prazos para a entrega final do produto eram cumpridos conforme planejados, e os custos para o desenvolvimento quase sempre ultrapassavam os valores contratados. Para atender a estes tipos de problemas, uma série de iniciativas da comunidade da Engenharia de *Software* foram tomadas. Dentre tais iniciativas podemos citar a de 1986 quando o SEI - *Software Engineering Institute* iniciou, a pedido e financiamento do DoD - *Department of Defense* dos Estados Unidos da América, o desenvolvimento de um modelo de qualidade que pudesse ser seguido pelas organizações (PAULK et al., 1999o, p. 4). Esperava-se com isto que seus trabalhos em desenvolvimento de *software* pudessem ter maior qualidade. Este modelo foi chamado de SW-CMM - *Capability Maturity Model for Software*, sendo a versão 1.0 publicada oficialmente em 1991 (PAULK et al., 1999p, p. 6) e a versão 1.1 em 1995. O objetivo do DoD era contratar empresas desenvolvedoras de *software* que fossem certificadas por este modelo, fazendo com que os problemas anteriormente citados fossem minimizados.

Algum tempo após a liberação da versão 1.1 do SW-CMM, aconteceu um fato interessante. Outras organizações do mundo que não tinham nenhum vínculo com o *DoD* passaram

também a seguir este modelo. Percebia-se então que o mercado formado pelas organizações desenvolvedoras de *software* estava finalmente se preocupando com a qualidade de seus produtos. Um outro fato interessante é que o modelo SW-CMM não possui nenhuma padronização oficial, mas encontra-se amplamente utilizado nos dias de hoje, sendo inclusive uma excelente ferramenta de *marketing* das organizações. Pode-se então concluir que o modelo SW-CMM é um padrão de fato e não de direito, ou ainda um quase padrão.

Todavia, para PAULK (2001), o modelo SW-CMM possui uma estrutura muito complexa e foi desenvolvido para grandes organizações. Tem-se então uma grande solução para a melhoria da qualidade dos produtos de *software*, mas em contrapartida dois outros problemas conseqüentes. Primeiro: a grande dificuldade de sua utilização por se tratar de um modelo altamente complexo, mais voltado para grandes organizações, que requer alto capital de investimento para a sua implantação. Segundo: para a grande maioria das organizações que desenvolvem *software* sem nenhum processo de qualidade, torna-se um modelo extremamente difícil de ser institucionalizado devido ao problema da baixíssima maturidade das pessoas. Para BAMBERGER (1997), além do SW-CMM ter sido desenvolvido para atender a grandes contratos de companhias aeroespaciais, existe ainda o problema da linguagem do SW-CMM . Bamberger (1997) afirma que existe o idioma inglês e o idioma SW-CMM. Embora o modelo tenha sido construído com 5 níveis de maturidade, justamente para que as organizações fossem galgando um nível após o outro ao longo de sua utilização, mesmo assim, pensando-se no nível 2, torna-se muito difícil a sua utilização em organizações sem nenhuma experiência com a qualidade de *software*. Mesmo com todas estas dificuldades, PAULK (2001) afirma que o SW-CMM pode ser aplicado em projetos contendo desde 2 a 3 pessoas ou em projetos envolvendo até 500 pessoas.

Além do modelo SW-CMM, dezenas de outras contribuições foram desenvolvidas para a melhoria da qualidade de *software*, conforme ilustra a Figura 1, sob a forma de categorias.

Antes porém de relacionar estas contribuições, é interessante fazer algumas observações com relação aos termos processo, padrão e modelo. Devido a algumas sobreposições de seus significados eles muitas vezes geram confusão em algumas situações. Segundo PAULK et al. (1999k, p. 361), um processo é uma seqüência de passos realizados para uma finalidade específica, como por exemplo, o processo de desenvolvimento de *software*. Para PFLEEGER (2001, p.45), um processo constitui uma série de passos envolvendo atividades, restrições e recursos que levam a algum resultado pretendido. ZAHRAN (1998a, p. 139-140) faz a separação entre os termos **padrões** e **modelos**. Os padrões são normalmente desenvolvidos

sob o auspício de algum órgão internacional de padronização, como por exemplo a Organização Internacional de Padrões (ISO). Já os modelos, embora não sejam desenvolvidos por entidades de padronização, possuem alta penetração popular, sendo desenvolvidos por organizações especializadas ou por meio de consórcios de empresas. Um exemplo de organização especializada é o Instituto de Engenharia de Software (SEI) da Universidade Carnegie Mellon dos EUA.

Diante destas definições, chega-se à conclusão que um processo pode ser tanto um padrão, se for elaborado por algum órgão de certificação internacional, quanto um modelo, se for desenvolvido por um consórcio ou organização especializada, ou ainda apenas um processo se não se enquadrar em nenhuma destas características. Assim, embora alguns processos citados neste item pudessem ser classificados também como modelos, foram incluídos como tal apenas aqueles que trazem a palavra modelo em sua nomeação ou em sua definição.

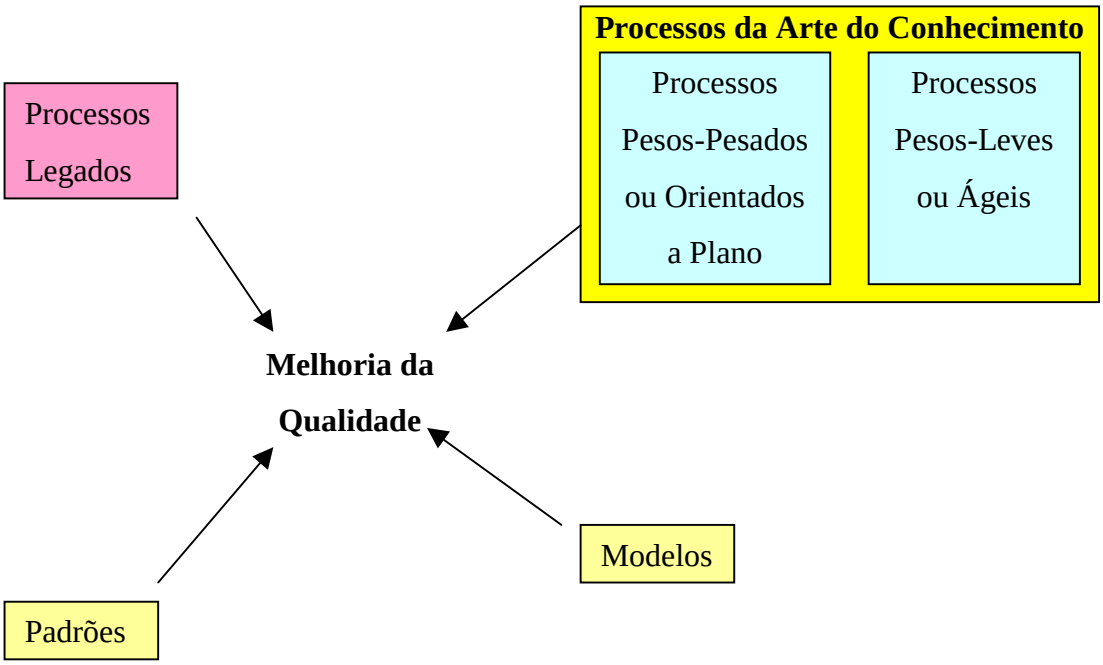

Figura 1 - Contribuições para a Melhoria da Qualidade de *Software*

Sem o intuito de citar todas as contribuições de todas estas categorias, segue uma relação reduzida.[1]

Processos Legados

Para este trabalho, foram considerados processos legados, aqueles atualmente considerados de pouca utilização atualmente pela comunidade de *software*.

- OMT (Técnica de Modelagem a Objeto) criado por James Rumbaugh em 1990 (RUMBAUGH, 1994).
- OOAD (Análise e Projeto Orientados a Objeto) criado por Grady Booch em 1991 (BOOCH, 1993).
- OOSE (Engenharia de *Software* Orientada a Objeto) criado por Ivar Jacobson em 1994 que evoluiu posteriormente para o Objectory (OSTEREICH, 1999).
- OOA/OOD (Análise e Projeto Orientados a Objeto) criado por Peter Coad e Ed Yourdon. Também conhecido como método Coad & Yourdon.
- OOSA (Análise de Sistemas Orientada a Objeto) criado por Sally Shlaer e Stephen Mellor por volta de 1989, também conhecido como método Shlaer-Mellor (SHLAIR; MELLOR, 1988).
- OODA (Análise e Projeto Orientados a Objeto) criado por Jim Odell e James Martin em 1994 (OSTEREICH, 1999, p. 5-7).

[1] Uma relação mais completa pode ser obtida em (CETUS LINKS, 2002).

Processos da Arte do Conhecimento

Processos "pesos-pesados" ou "Orientados a Planos".

Os processos pesos-pesados ou orientados a plano são aqueles baseados no modelo SW-CMM e que prometem previsibilidade, estabilidade e alta segurança (BOEHM, TURNER, 2003). Além disto, nestes processos os projetos começam com uma solicitação formal e uma documentação completa de todos os requisitos a serem desenvolvidos segundo um plano de desenvolvimento (WILLIAMS, COCKBURN, 2003). Segundo COHN e FORD (2003), alguns times possuem a sensação de maior liberdade devido ao fato de não existir um gráfico de Gantt dirigindo os trabalhos. Nesta categoria podemos relacionar:

- Fusion, desenvolvido pela *Hewlett Packard Laboratories* e publicado em 1993 (MALAN, R.; LETSINGER, R., 1993).
- Processo de Desenvolvimento de *Software* Unificado mais conhecido como Processo Unificado, criado pelos autores da UML, Ivar Jacobson, Grady Booch e James Rumbaugh em 1999 para ser um processo aberto (ARLOW; NEUSTADT, 2001, p. 22-26).
- RUP (Processo Unificado da Rational) pertencente à IBM Corporation, criado em 1998, e RUP 2001 em 2001 (KRUCHTEN, 2003). Este processo foi construído a partir do Processo Unificado.
- EUP (Processo Unificado de Empresa) (AMBLER, 2002). Este processo também foi construído a partir do Processo Unificado.
- Processo OPEN (Processo Orientado a Objeto, Notação e Ambiente) da Open Consortium, criado em 1994 (HENDERSON, 1997). Assim como o Processo Unificado, também foi construído para ser um processo aberto.
- OOSP (Processo de *Software* Orientado a Objeto), criado por Scott W. Ambler em 1998 (AMBLER, S. W, 1998).

Processos "Pesos-Leves" ou Ágeis

- XP (Programação Extrema) (BECK, 2000) criado por Kent Beck em 1996 (GOLDSTEIN, 2003).
- AM (Modelagem Ágil) de *Scott W. Ambler* criado em 2001 (AMBLER, 2003).
- Lean Development desenvolvido pelo casal Tom e Mary em 2003, a partir de conceitos de Manufatura Enxuta (POPPENDIECK, M; POPPENDIECK, T, 2003).
- Crystal Clear (COCKBURN, 2001). De acordo com COCKBURN (2002), trata-se de um processo que "...descreve papéis, times, valores, intenções, hábitos, atividades, políticas e

produtos de trabalho de projetos de pequenos times, onde o custo e a rápida entrega são os pontos mais importantes".

- SCRUM. Processo criado por Mike Cohn. Trata-se de um processo ágil para desenvolvimento de *software*, onde sua progressão é feita sob uma série iterações com a duração de 1 mês, chamadas de *Sprints*. (COHN, 2003).
- Desenvolvimento de *Software* Adaptável. (HIGHSMITH, 2000).
- Desenvolvimento Orientado a Característica de Peter Coad e Jeff De Luca. O foco do processo é o desenvolvimento de iterações curtas a cada 2 semanas e orientadas a modelos (COAD; LEFEBVRE; LUCA, 1999).
- DSDM (Método de Desenvolvimento de Sistema Dinâmico) pertencente ao Consórcio DSDM. O Consórcio foi criado em 1993 e a primeira versão do processo foi lançada em março de 1994 (DSDM Consortium, 2003a). Este processo trabalha de forma que os recursos e o tempo sejam mantidos fixos, enquanto que as funcionalidades desejadas para o sistema podem variar. Segundo este processo, esta abordagem é exatamente oposta à abordagem dos processos tradicionais onde a funcionalidade desejada é mantida constante e os recursos e o tempo podem variar ao longo do desenvolvimento [?] (DSDM Consortium, 2003b).

Modelos para Processos de Desenvolvimento de *Software*

- SW-CMM (Modelo de Maturidade da Capacidade para *Software*) versão 1.0, publicada oficialmente em 1991, e a versão 1.1 em 1995. Modelo desenvolvido pelo SEI (Instituto de Engenharia de *Software*) da Universidade Carnegie Mellon dos Estados Unidos da América.
- CMMI (Modelo de Maturidade da Capacidade para Integração), também produzidos pelo SEI e publicado em 2002, representando uma evolução do SW-CMM e incluindo integração com outros processos do SEI.
- BOOTSTRAP criado pelo Instituto BOOTSTRAP em 1996. Modelo utilizado para avaliações de processos de desenvolvimento para determinar a maturidade do processo utilizado por uma organização, identificar seus pontos fortes e fracos e oferecer diretrizes para a melhoria deste processo (ZAHRAN, 1998b, p. 339-355).

Linguagens de Modelagem

- UML (Linguagem de Modelagem Unificada) criado por Ivar Jacobson, Grady Booch e James Rumbaugh em 1996. Modelo composto de uma série de elementos que podem ser combinados para a construção de uma série de diagramas. Atualmente a UML encontra-se sob os cuidados da OMG (Grupo de Gerência de Objeto). Segundo a OMG, a UML é composta de diagramas estruturais, diagramas comportamentais e diagramas de gerência de modelos, utilizados para a modelagem de pequenos, médios e grandes projetos de *software* (OMG, 2003).
- OML (Linguagem de Modelagem Aberta) da Open Consortium, criado em 1994. Modelo que consiste de notações e metamodelo do processo OPEN (1999).
- SDL (Linguagem de Descrição de Especificação) criada pelo CCITT (***C****omité* ***C****onsultatif* ***I****nternational* ***T****éléphonique et* ***T****élégraphique*) atual ITU (União Internacional de Telecomunicações). Linguagem que provê uma especificações e descrição do comportamento de sistemas de telecomunicações sem ambigüidades (ITU-T Recomendation Z.100, 2002).

Padrões

- IEEE Std 1220-1998 – Padrão para Aplicação e Gerência de Processos de Engenharia de Sistemas. Este padrão define as tarefas interdisciplinares que são requeridas durante o ciclo de vida de um sistema, para transformar as necessidades do cliente, requisitos e restrições em uma solução de sistema (IEEE Std 1220-1998).
- ISO/IEC-12207. Padrão da Tecnologia da Informação que trata de Processos de Ciclos de Vida de *Software*. Este padrão estabelece um sistema para processos de ciclos de vida de *software* com uma terminologia bem definida. Contém processos, atividades e tarefas que devem ser aplicadas durante a aquisição de um sistema que contém *software*, um produto de *software stand-alone* e serviços de *software*. (ISO/IEC 12207:1995, 1995).
- ISO/IEC-15504 (Modelo SPICE) finalmente promulgada como padrão em 21 de junho de 2003. Este padrão é dividido em 5 partes. A única parte normativa é a parte 2 (*Performing an Assessment*). Esta parte descreve um processo para a aplicação e avaliação de processos de *software*, para se poder determinar a sua capacidade e identificar melhorias.
- ABNT NBR ISO9000-3 - Padrões de gestão da qualidade e garantia da qualidade - Parte 3: Diretrizes para a aplicação da NBR 19001 ao desenvolvimento, fornecimento e manutenção de *software*, publicada em novembro de 1993.

De todas estas contribuições para a sociedade de *software*, o que se percebe é que nenhuma delas pode ser considerada como sendo a solução definitiva para a melhoria da qualidade. A solução a ser adotada para uma organização pode ser totalmente diferente de uma outra organização, podendo inclusive acontecer situações de utilização de processos híbridos, possuindo tanto características ágeis quanto orientadas a plano (BOEHM, TURNER, 2003). Além disto, as organizações de *software* estão buscando cada vez mais a utilização efetiva de estratégias de desenvolvimento de *software* que possam trazer, além do benefício de melhor satisfação do cliente, uma redução do tempo de ciclo de vida de desenvolvimento, e por conseguinte, uma redução de custos.

1.2 Problemas e Necessidades de Organizações Imaturas

Antes de iniciar a identificação dos problemas e necessidades de organizações imaturas, vale a pena definir o conceito de organização. Assim como no SW-CMM, uma organização pode ser uma empresa matriz com todos os seus empregados, uma filial desta empresa matriz ou ainda parte ou partes de uma empresa global. O que se deve ressaltar é que, sendo parte ou não de uma empresa maior, uma organização é tratada como possuidora de independência suficiente para realizar suas atividades de desenvolvimento de *software*.

Considerando agora o termo “organizações imaturas” podemos classificar como sendo aquelas que desenvolvem *software* segundo um processo *ad-hoc*, ou imaturo, sem comprometimento algum com padrões de qualidade, quer seja por negligência ou pela simples falta de conhecimento. Segundo ZAHRAN (1998d, p. 17), processos não disciplinados e imaturos são aqueles que apresentam comportamentos inconsistentes. Para PAULK et al. (1999b, p.7), organizações imaturas são aquelas onde o processo de *software* é improvisado durante o desenvolvimento de um projeto. Se existe um processo especificado, ele não é seguido rigorosamente ou imposto. A organização é reacionária e os gerentes atuam na solução de crises atuando como bombeiros. Orçamentos e cronogramas sempre excedem as expectativas devido a estimativas não-realistas. Não existem bases objetivas para se determinar a qualidade de um produto. O cliente tem pouca participação no desenvolvimento, além de outras características. Em tais organizações percebe-se ainda que grande parte é composta de poucos funcionários, o que leva a uma grande dificuldade na implantação de processos grandes e complexos, devido à necessidade de grandes investimentos. Segundo HIGHSMITH (2002), 60% dos projetos de *software* do mundo possuem no máximo 10 pessoas envolvidas.

No contexto destas organizações pode-se relacionar alguns problemas e necessidades muitas vezes comuns a todas elas, tais como:

- Rotatividade de recursos humanos. Segundo HUMPHREY (2001, p. 13-15), 1 a cada 5 programadores mudam de emprego a cada ano, o que produz uma perda de tempo e de dinheiro de proporções extraordinárias.
- Ausência de um processo de desenvolvimento de *software*, ou mal definido e não disciplinado. Para ZAHRAN (1998c, p. 14-15), a ausência de um processo leva às ações de bombeiros ou super-heróis para a resolução de conflitos, e processos não disciplinados levam ao estresse interno da organização.
- Ausência de um proprietário do processo de desenvolvimento. Conforme descreve ZAHRAN (1998g, p. 99), um processo sem propriedade pode facilmente cair em desuso, tornando-se apenas um objeto de prateleira "*shelfware*".
- Falta de recursos e/ou falta de apoio da alta gestão para a implantação de programas da qualidade. Para ZAHRAN (1998f, p. 39), um processo para ser efetivo deve possuir propriedade e ser apoiado pela alta gestão da organização, além de outras características. Pesquisas recentes revelaram que, de 446 organizações de *software*, apenas 28,2% conhecem e usam sistematicamente um dos modelos de qualidade: ISO/IEC-12207 (3.9%), ISO-9000 (19,4%), SW-CMM (3,9%) ou ISO/IEC-15504 (1,0%) (MCT, SEPIN, 2001, p. 86-87).
- Utilização inadequada das ferramentas de *software* existentes na organização.
- Grande dificuldade no cumprimento de prazos. Segundo PAULK et al.(1999q, p. 3), a capacidade de desenvolver e entregar um *produto* de *software* confiável, usável, dentro do orçamento e compromissos previsto continua até hoje a iludir muitas organizações de *software*.
- Ausência de envolvimento do cliente durante o desenvolvimento. Segundo PAULK et al. (1999r, p. 7), enquanto o produto de *software* não é entregue, o cliente tem pouca visão do que está acontecendo com o desenvolvimento.
- Requisitos mal definidos ou não-documentados. Segundo PAULK (1998), tanto em pequenas quanto em grandes organizações, existem problemas relacionados a requisitos não-documentados.

- Dificuldade de se acompanhar a evolução tecnológica dos *softwares* de suporte.
- Dificuldade de se estimar custos, prazos e recursos necessários. Para PAULK et al. (1999r, p. 7), é muito comum ocorrerem estouros de cronogramas e orçamentos em organizações imaturas, devido ao fato de que seus cálculos não são baseados em estimativas realísticas.
- Dependência de profissionais altamente especializados e dedicados. ZAHRAM (1998e, p. 28), em sua descrição de características de processos imaturos, diz que uma dessas características é onde reina o caos, e a ação de super-heróis ou bombeiros torna-se algo normal.
- Dificuldade de aceitar as mudanças devido à implantação de um programa de qualidade. ZAHRAM (1998h, p. 99) diz que introduzir as mudanças necessárias para a implementação de um processo não é uma tarefa fácil, pois muitos consideram as atividades chatas, sacrificantes e pouco excitantes. WIEGERS (1999) também comenta que as pessoas naturalmente resistem às mudanças da forma como elas trabalham porque estão familiarizadas com as práticas atuais e também pelo medo do desconhecido.

Com todas estas dificuldades, aliadas ainda ao "...reconhecimento do desenvolvimento de *software* como sendo um processo empírico (não-linear)" (WILLIAMS, COCKBURN, 2003), o que dificulta a sua gerência, uma organização precisa inicialmente definir um processo de desenvolvimento ajustado às suas necessidades e não apenas às necessidades do processo. A alta gestão deve apoiar irrestritamente a aplicação deste processo e deve manter a equipe motivada para que o processo possa ser seguido, acompanhado e melhorado de forma sistemática.

1.3 Proposta de pesquisa e motivação

Tendo em vista as dificuldades citadas anteriormente na utilização de processos que oferecem grande dificuldade de entendimento e implementação por organizações imaturas, o presente trabalho busca atender aos seguintes objetivos:

- Apresentar, aplicar e mostrar os resultados de um processo aderente parcialmente ao SW-CMM nível 2 em uma organização pertencente ao nível 1 do modelo SW-CMM.
- Ser de fácil utilização, para que as pessoas de organizações imaturas possam utilizá-lo com baixos índices de rejeição às mudanças necessárias.

Para atender a estes objetivos, este trabalho apresenta o desenvolvimento de um processo chamado LPA (**L**evantamento, **P**lanejamento e **A**companhamento), orientado a plano, e 50% aderente ao modelo SW-CMM nível 2 de maturidade, uma vez que ele atende a apenas 3 KPAs das 6 existentes neste nível, cobrindo os processos gerenciais de requisitos e de projeto (planejamento e acompanhamento). Estes processos são fundamentais para que um *software* possa ser desenvolvido. Ou seja, o desenvolvimento de qualquer projeto de *software* sempre irá envolver o **L**evantamento dos requisitos apresentados pelo cliente, o **P**lanejamento das atividades a serem desempenhadas e o **A**companhamento destas atividades até a entrega do produto final.

Visando tornar um processo agradável de ser utilizado, ele foi construído para ser executado em um navegador Internet (testado com sucesso no *Internet Explorer da Microsoft*), com uma forte similaridade visual com o RUP.

Como validação do trabalho proposto, foi aplicado um trabalho de pesquisa-ação[2] em 1 projeto na SSA (Seção de *Software* Administrativo) do Inatel (Instituto Nacional de Telecomunicações).

Esta organização tinha as seguintes características:

---

[2] O leitor não deve confundir este termo com o termo “estudo de caso”. Pesquisa-ação é um dos meios de pesquisa que envolve a intervenção participativa do pesquisador em alguma situação real de algum problema coletivo. Já o estudo de caso é um dentre outros meios de pesquisa, para um determinado estudo de alguma unidade ou organização de forma profunda e intensa, com a finalidade principal de detalhar, descrever e aprender sob o objeto pesquisado. (http://www.fsanet.com.br/canais/posgraduacao/pesquisa/pesquisa.html).

- Equipe de 5 pessoas. Um gerente de projeto que também assumia os papéis de líder de projeto e desenvolvedor. Uma líder de projeto que também assumia o papel de desenvolvedora e 3 outras desenvolvedoras.
- Responsável pelo desenvolvimento de *softwares* administrativos internos do Inatel.
- Período de trabalho de 20h00 semanais aplicadas em atividades de desenvolvimento e 20h00 semanais dedicadas à manutenção de produtos por eles desenvolvidos. O LPA foi aplicado dentro das horas dedicadas a desenvolvimento.
- Plataforma Windows
- Ferramental de desenvolvimento da Borland com linguagem de programação Delphi.

Os resultados, conclusões e trabalhos futuros podem ser observados nos itens 4 e 5.2.

1.3.1 Ferramental Utilizado e Metodologia de Desenvolvimento

Para o desenvolvimento do processo LPA foram utilizadas as seguintes ferramentas de *software* e outros materiais.

1. Modelo SW-CMM, especificamente as KPAs (Área de Processo-Chave) da Gestão de Requisitos, de Planejamento de Projeto de *Software* e de Supervisão e Acompanhamento de Projeto de *Software* do nível 2. Este modelo possibilitou que o processo atendesse às especificações destas KPAs.
2. Ferramenta CASE *Rational ROSE Enterprise Edition*, versão 2002.05.00 da *Rational Software Corporation*[3], com o *plug-in* RPW (*Rational Process Workbench*) versão 2002.05.00.25, também do mesmo fabricante. Nesta ferramenta é que foi construído o processo LPA. A ferramenta CASE Rational ROSE e o respectivo *plug-in* RPW foram utilizados para a construção do processo em si, com seus elementos e diagramas nas visões lógica e de componentes. A visão lógica continha todo o repositório do processo em termos de elementos e diagramas, e a visão de componentes continha os elementos que realmente seriam utilizados para a publicação deste processo.

[3] Atualmente, a *Rational Software Corporation* foi adquirida pela IBM Corporation.

3. Ferramenta RUP - *Rational Unified Process* versão 2002.05.00.25 da *Rational Software Corporation*. Esta ferramenta foi utilizada como apoio para a construção das disciplinas, papéis, atividades, artefatos e diagramas do LPA.
4. Ferramenta Dreamweaver 2 da Macromedia. Esta ferramenta foi utilizada para a construção de todas as páginas de extensão html do processo LPA, tornando-o navegável.
5. Ferramenta Microsoft WORD 97 SR-2 da Microsoft Corporation. Esta ferramenta foi utilizada para a construção dos gabaritos do LPA. Por meio destes gabaritos é que o usuário do LPA pode produzir os artefatos requeridos durante o desenvolvimento de um projeto de *software.*
6. Ferramenta Microsoft Project Standard 2002 da Microsoft Corporation. Esta ferramenta foi utilizada para a construção do gabarito Cronograma do LPA. Por meio deste gabarito, o usuário do processo LPA pode construir o cronograma do projeto de *software*.

A partir das características identificadas no modelo SW-CMM, o processo foi sendo construído utilizando-se as ferramentas citadas anteriormente. O processo navegável era gerado a cada mudança significativa e testado quanto à sua navegabilidade.

1.4 Principais Resultados Esperados

Diante do cenário caótico de desenvolvimento de *software* no mundo, e face às inúmeras metodologias de desenvolvimento, metodologias estas que possuem uma abrangência muito além do que a maioria das organizações podem almejar, espera-se que este trabalho possa ajudá-las a possuírem um processo reduzido de gerência de requisitos e de projeto, com uma qualidade satisfatória, servindo de base para um amadurecimento gradativo e conseqüente melhoria contínua da qualidade dos produtos de *software* produzidos e do processo utilizado.

As informações contidas neste trabalho se propõem a servir de auxílio para a construção de processos mais adequados ao perfil de cada organização.

Caso não exista interesse das organizações em se enquadrarem aos requisitos do SW-CMM nível 2, elas podem fazer um ajuste ainda mais fino ao elaborarem seu próprio modelo de desenvolvimento, pois não estarão obrigadas a seguirem certas atividades consideradas desnecessárias, ou então simplificá-las. Para simplificar esta adaptação, o Apêndice A - Mapeamento LPA x SW-CMM Nível 2 apresenta um mapeamento entre o SW-CMM nível 2 e o LPA.

1.5 Estrutura do Trabalho

Esta dissertação encontra-se organizada em capítulos.

Descreve-se no capítulo 1 os problemas gerais e contribuições para a melhoria da qualidade no desenvolvimento de *software*, as motivações para o trabalho, a metodologia utilizada e os resultados esperados.

No capítulo 2, faz-se uma revisão bibliográfica do Processo Unificado Rational e do modelo SW-CMM, que serviram de base para o desenvolvimento.

O capítulo 3 é o coração da dissertação, e contém a descrição completa do processo LPA.

Apresenta-se no capítulo 4 uma análise dos resultados obtidos durante a aplicação do processo na SSA do Inatel.

Por último, apresenta-se no capítulo 5 uma conclusão e descrevem-se algumas sugestões para trabalhos futuros.

Dentre os apêndices inclusos neste material, encontra-se um CD contendo: esta dissertação em formato eletrônico, todo o processo que pode ser executado em uma ferramenta de navegação da Internet e todos os gabaritos especificados pelo LPA.

2 REVISÃO BIBLIOGRÁFICA

Este capítulo aborda uma visão geral do Modelo SW-CMM e do RUP, por terem sido as principais referências utilizadas no desenvolvimento do processo LPA. O modelo SW-CMM foi o modelo de qualidade de *software* escolhido para que o processo LPA fosse concordante, ressalvando o nível 2 e as KPAs conforme descritas no item 1.3. Já o RUP serviu de elo entre o SW-CMM e o LPA, para facilitar a identificação dos elementos e conteúdos do LPA.

2.1 Visão Geral do Processo Unificado Rational - RUP

O RUP é um processo iterativo, incremental, fortemente centrado na arquitetura e orientado por casos de uso e componentes.

O RUP pode ser interpretado segundo uma visão bidimensional. No eixo vertical tem-se a visão de processo em termos de disciplinas, ao todo 9, a serem utilizadas durante o ciclo de vida de Desenvolvimento de *Software*, representando de forma estática o processo em um determinado momento do desenvolvimento. No eixo horizontal, tem-se a visão do ciclo de vida evoluindo ao longo do tempo de desenvolvimento, representando os aspectos dinâmicos do processo e expresso em termos de fases, iterações e marcos. A visão do ciclo de vida é composta de 4 fases distintas: Concepção, Elaboração, Construção e Transição que são finalizadas mediante o alcance de um marco específico. Em cada fase pode-se ter uma ou mais iterações. Cada iteração se comporta sob a forma de um modelo minicascata, onde a intensidade (esforço) de cada disciplina irá variar dependendo do momento. Por exemplo, na fase de requisitos, tem-se uma grande intensidade de requisitos e muito pouco de teste.

A Figura 2 ilustra estas duas visões do RUP, que costumam receber o nome de Gráfico das Baleias ou Gráfico das Baleias Corcundas.

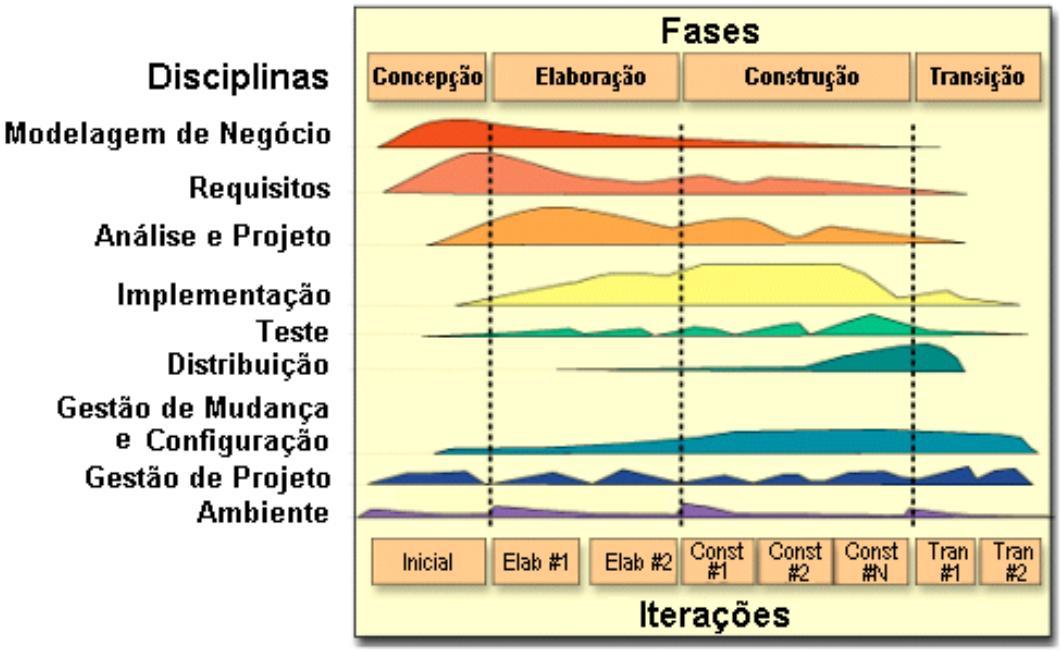

Figura 2 - Gráfico das baleias do RUP

FONTE: Tradução do original RUP on-line - 2001A.04.00.13.

O gráfico das baleias do RUP não define um formato único para as intensidades de cada disciplina, devendo ser ajustado para cada tipo de organização e projeto de *software*.

2.1.1 Objetivos do RUP

O RUP é um processo de engenharia de *software* que tem os seguintes objetivos:

- Melhoria da produtividade e das entregas realizadas pela equipe de desenvolvimento.
- Desenvolvimento com alta qualidade.
- Atendimento das necessidades dos usuários finais do *software* desenvolvido.
- Atuação segundo cronograma e orçamento previsíveis.

2.1.2 A Visão do Ciclo de Vida

O ciclo de vida é basicamente iterativo e incremental. Ele é dividido em 4 fases que vão se sucedendo ao longo do tempo. Estas fases são chamadas de Concepção, Elaboração, Construção e Transição. Em cada fase pode existir uma ou mais iterações. Estas iterações se comportam como sendo ciclos de vida em minicascatas como pode ser visto na Figura 3.

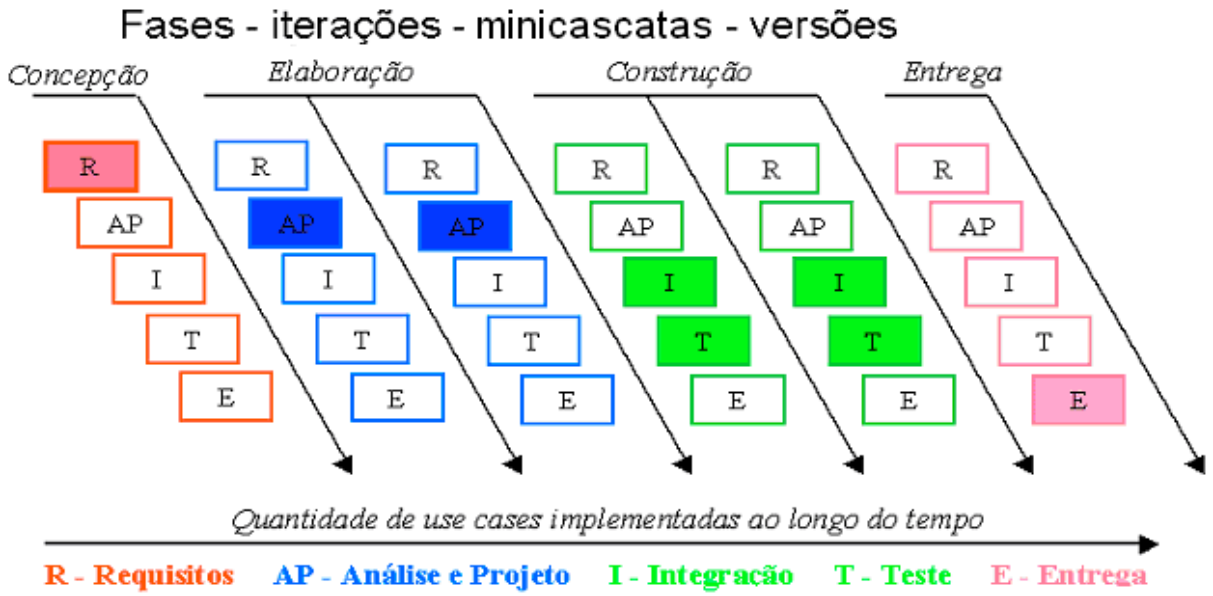

Figura 3 - Conjunto de Minicascatas de um Desenvolvimento Iterativo

Em cada minicascata tem-se que passar por cada disciplina da visão de processo, umas com mais outras com menos intensidade. Ao final de cada iteração, tem-se a liberação de uma *release* do sistema. Esta *release* terá uma determinada capacidade funcional, capacidade esta que irá aumentando nas iterações ou *releases* seguintes, até se ter o sistema totalmente implementado e implantado no cliente. A Figura 4 ilustra melhor o conceito de desenvolvimento iterativo.

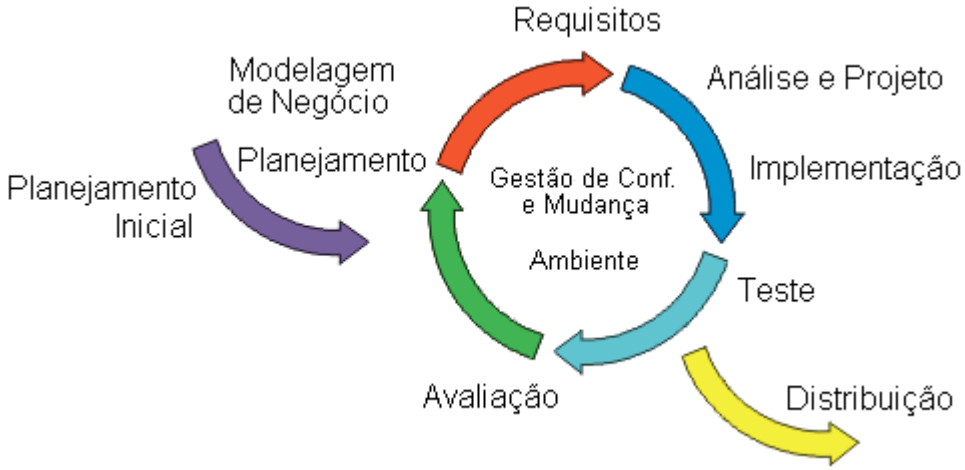

Figura 4 - Desenvolvimento Iterativo

FONTE: Tradução do original RUP on-line - 2001A.04.00.13.

A abordagem de desenvolvimento iterativo advém do modelo espiral definido por BOEHM B. (1988). A idéia principal do desenvolvimento iterativo é a produção sistemática de *releases* ao final de cada iteração, onde o sistema irá crescendo gradativamente em termos de capacidade funcional até o produto final.

Para cada fase do processo existe um objetivo claro a ser atingido (marco), conforme descrito nos itens seguintes.

Fase de Concepção – O marco é o Escopo

Durante esta fase é estabelecido o caso de negócio para o sistema e o **escopo** do projeto é determinado. O caso de negócios inclui critérios de sucesso, avaliação de riscos, estimativa de recursos necessários e um plano para a iteração da fase, incluindo os principais marcos. É muito aconselhável nesta fase a construção de um ou mais protótipos executáveis, que servirão de teste para a validação desta fase. Estes protótipos devem ser descartados ao final da fase ou devidamente ajustados para que possam fazer parte do sistema.

Nesta fase a preocupação deve residir em descobrir o que o cliente deseja e identificar o que será necessário desenvolver para se atender aos objetivos do mesmo.

A partir da pergunta clássica “O que se deseja que seja desenvolvido?”, são iniciadas reuniões com os interessados (*stakeholders*) e faz-se a compilação de todas as informações em um artefato do RUP chamado *Vision* (Visão).

A entrada desta fase pode ser caracterizada com a pergunta “O que se deseja que o sistema faça?” e a saída como sendo a resposta a essa pergunta.

Diversas hipóteses devem ser avaliadas, algumas aceitas e outras rejeitadas.

As atividades principais desta fase compreendem: ***Brainstorming*** com os clientes, pesquisas com especialistas no domínio do problema, análises de Custo/Benefício com Gerentes e Líderes de Projetos, análise de **Casos de Uso** com especialistas e o desenvolvimento de protótipos executáveis. Os protótipos irão produzir conceitos importantes acerca do sistema alvo.

Neste marco tem-se:

Um exame dos resultados obtidos para o ciclo de vida.

Uma decisão pela continuidade ou não do projeto - *go-no-go*.

Fase de Elaboração – O marco é a Arquitetura

A entrada desta fase pode ser caracterizada com a pergunta “Como o sistema será feito?”.

Para que esta fase seja estabelecida, é necessário que a maioria dos requisitos do sistema estejam especificados.

As metas desta fase compreendem a análise do domínio do problema, o estabelecimento de uma arquitetura sólida, o desenvolvimento do plano de projeto, a eliminação dos elementos de maior risco de todo o sistema, a descrição da maioria dos casos de uso e suas restrições, a implementação dos casos de uso mais importantes e uma verificação.

Neste marco tem-se:

Os objetivos, o escopo, a arquitetura e a resolução dos riscos principais.

Uma decisão pela continuidade ou não do sistema.

Fase de Construção – O marco é a Capacidade Operacional do *software*

A entrada desta fase pode ser caracterizada pela pergunta “Como será o código completo do sistema?”.

Nesta fase, tem-se o desenvolvimento de um produto de *software* completo, de forma iterativa e incremental, e que estará pronto para seus usuários após a fase de Transição. Para que isto possa ser atingido, é necessário que todos os requisitos do sistema estejam especificados.

O projeto passará a possuir mais corpo devido à lógica adicional inserida e serão realizados testes unitários, de integração e de sistema.

Ao final desta fase, deve-se verificar se tanto o *software* quanto usuários e o ambiente encontram-se prontos para se tornarem operacionais.

As metas desta fase compreendem a implementação completa do produto, a formatação dos casos de uso restantes e suas implementações em iterações, incluindo a aplicação de testes unitários, de integração e de sistema.

Neste marco tem-se:

O *software*, o ambiente e os usuários prontos para entrarem em operação.

A versão alfa do *software*.

Fase de Transição – O marco é o Produto Final

A entrada desta fase pode ser caracterizada pela pergunta “Como liberar o sistema para o cliente?”.

As metas desta fase compreendem treinamentos ao cliente, ajustes da versão *Beta* devido a algumas necessidades identificadas pelo cliente, correções de *bugs* identificados pelo mesmo, implementação de características que ficaram postergadas e entrega oficial do produto de *software* por meio de uma *release* externa.

Ao final desta fase, deve-se verificar se os objetivos do ciclo de vida foram alcançados e se será necessário iniciar um novo. As lições aprendidas até o presente momento deverão ser comunicadas a toda a equipe de desenvolvimento, de forma a aumentar a maturidade tanto do produto quanto do processo de desenvolvimento.

Neste marco tem-se:

Geração de uma *release* externa.

Avaliação dos objetivos do ciclo de vida.

Decisão pelo início de um novo ciclo de vida.

Comunicação das lições aprendidas à equipe de desenvolvimento.

2.1.3 A Visão de Processo

A visão de processo é dividida em diversas disciplinas. Uma disciplina é constituída de um fluxo de trabalho (*Workflow*), composto por uma série de macroatividades interrelacionadas sob o foco de uma determinada área do processo de desenvolvimento ou gerencial e que deverão produzir um ou mais artefatos. Para a produção destes artefatos, cada macroatividade do fluxo de trabalho de uma disciplina irá descrever os papéis das pessoas envolvidas na produção destes artefatos, as atividades que estas pessoas deverão realizar e os artefatos que elas deverão utilizar como entrada.

A Figura 5 ilustra as 9 disciplinas definidas pelo RUP.

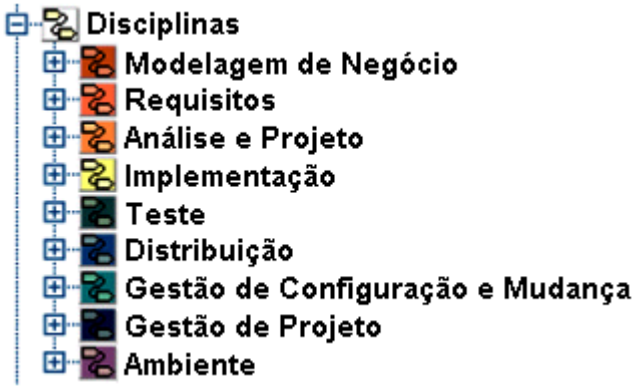

Figura 5 - Disciplinas do RUP

FONTE: Tradução do original RUP on-line - 2001A.04.00.13.

As macroatividades de uma disciplina são representadas utilizando-se o diagrama de atividades da UML. A Figura 6 ilustra as macroatividades da disciplina de Requisitos (*Requirements*) do RUP.

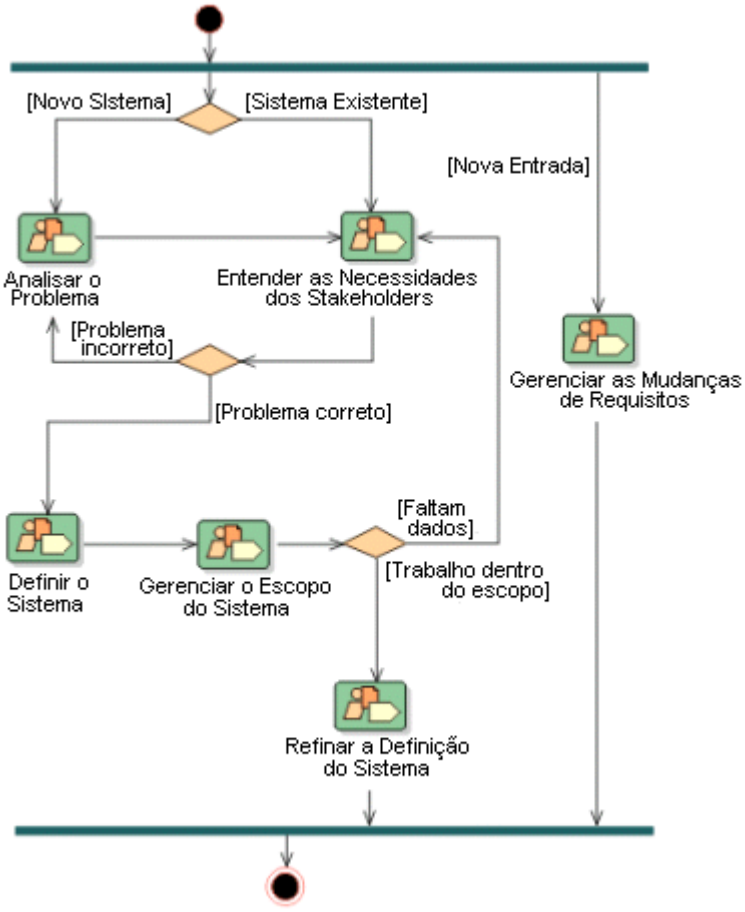

Figura 6 - As macroatividades da disciplina de Requisitos do RUP

FONTE: Tradução do original RUP on-line - 2001A.04.00.13.

Cada macroatividade é detalhada (*Workflow Detail*). Uma macroatividade detalhada é um agrupamento de atividades que devem ser realizadas em conjunto, sendo que estas atividades recebem e produzem artefatos. A título de exemplo, a Figura 7 ilustra o detalhamento da macroatividade “Definir o Sistema” da disciplina de Requisitos.

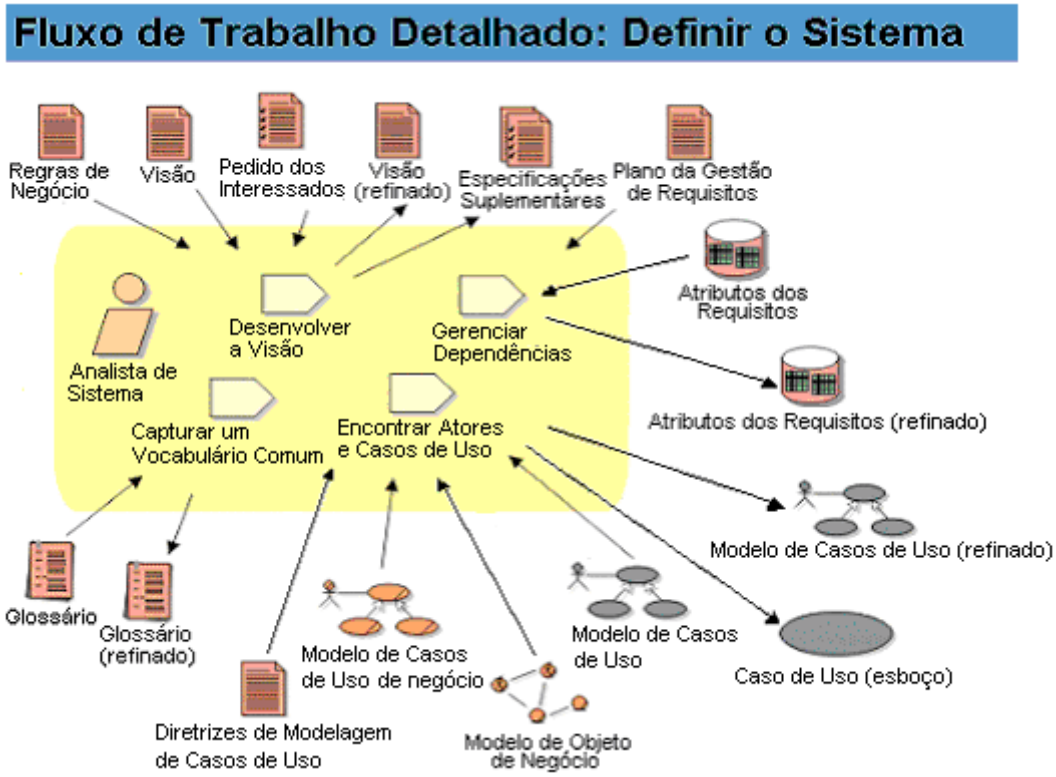

Figura 7 - Detalhamento de uma macroatividade

FONTE: Tradução do original RUP on-line - 2001A.04.00.13.

Os ícones internos indicam o papel e as atividades a serem realizadas. O papel deve ser realizado por uma pessoa que assumirá a responsabilidade pela realização das atividades. Os ícones externos direcionados pelas setas indicam os artefatos a serem utilizados como entrada para a realização das atividades e os artefatos que deverão ser produzidos como resultados. Observa-se que alguns artefatos podem ser considerados como sendo de entrada e de saída (Glossário, Visão, Atributos dos Requisitos e Modelo de Casos de Uso). Isto significa que a atividade irá receber o artefato com uma certa quantidade de informações e ao final da atividade estas informações estarão atualizadas ou completadas.

Além disto, como o RUP pode ser executado em uma ferramenta de navegação Internet, todos os ícones apresentados no detalhamento de uma macroatividade podem ser clicados a qualquer momento e irão apresentar um detalhamento completo de um papel, de uma atividade ou de um artefato.

As disciplinas dos processos da engenharia de *software* são:

Modelagem de Negócio
Permite entender a estrutura e o dinamismo da empresa na qual o sistema deverá ser desenvolvido, propondo melhorias significativas e permitindo que tanto a organização quanto a equipe de desenvolvimento compactuem deste entendimento.

Requisitos
Identificar o que o sistema deverá fazer, especificar os requisitos tornando-os claros aos desenvolvedores, identificar as fronteiras do sistema (atores), prover uma base para o planejamento das iterações de desenvolvimento, prover uma base para as estimativas de custo e tempo de desenvolvimento e definir as interfaces de usuário do sistema.

Análise & Projeto
Permite transformar os requisitos em um modelo de projeto para o sistema, criando ao mesmo tempo uma arquitetura robusta.

Implementação
Produção do código do sistema. Este código deverá estar organizado em termos de subsistemas e camadas de *software*, implementando as classes e objetos do sistema em componentes, tais como: arquivos fonte, arquivos binários, arquivos executáveis etc, realizar o teste unitário destes componentes e integrar os resultados para produzir o executável do sistema.

Teste
Verificar a interação entre os objetos, a integração entre os componentes do *software*, se os requisitos foram todos implementados corretamente e garantir que sejam encontrados e corrigidos a maior quantidade possível de defeitos antes que o sistema seja entregue para o cliente.

Entrega
Permite o empacotamento do sistema para entrega, sua instalação no ambiente do cliente e uma forma de acesso *on-line* via Internet.

As disciplinas dos processos gerenciais são:

Gestão de Configuração & Mudança
Deve-se identificar os itens de configuração, restringir as mudanças a estes itens, realizar auditoria nas mudanças realizadas e definir a gestão de configuração a ser seguida.

Gestão de Projeto
Deve-se prover diretrizes práticas para se planejar as atividades, a alocação da equipe de trabalho, a execução do projeto e seu monitoramento. Em adição, deve prover um arcabouço (*framework*) para a gestão de riscos.

Ambiente
Deve-se configurar tanto o processo quanto o projeto a ser desenvolvido, fornecendo ao time de desenvolvimento um ambiente composto pelo processo e pelas ferramentas necessárias a este time.

2.1.4 Outros Recursos do RUP

Além do que foi apresentado, o RUP possui ainda uma série de outros documentos que permitem auxiliar a equipe de desenvolvimento durante seus trabalhos. São eles:

- Diretrizes de Trabalho (*Work Guidelines*) – Textos que apresentam técnicas e conselhos práticos e úteis para a realização de uma determinada atividade.
- Dicas de Ferramentas (*Tool Mentors*) – As dicas são textos que descrevem como executar uma determinada atividade de uma determinada disciplina, utilizando para isto uma determinada ferramenta de *software* da Rational, como por exemplo: Rational Rose, RequisitePro, Rational ClearCase, Rational ClearQuest etc.
- Conceitos (*Concepts*) – Descrições de conceitos importantes referentes a todo o processo ou a cada disciplina (ex.: iteração, fase, risco, teste de desempenho, requisitos etc).
- Listas de Verificação (*Checkpoints*) – Listas contendo uma série de itens a serem seguidos, para que a qualidade de um artefato possa ser avaliada.
- Relatórios (*Reports*) – Documentos produzidos por uma ferramenta de engenharia de *software* referente a um elemento de um modelo ou de um modelo do sistema em desenvolvimento.

- Diretrizes para Artefatos (*Artifact Guidelines*) – Textos que apresentam informações sobre como um artefato deve ser desenvolvido, avaliado e utilizado, fornecendo assim informações essenciais para que o artefato possa ser produzido.
- Gabaritos (*Templates*) – Os gabaritos são documentos (nos formatos *MSWord*, HTML, *Framemaker* ou *MSPRoject*) parcialmente preenchidos e que contêm instruções para auxiliar o autor nas partes que necessitam de preenchimento adicional. Os gabaritos constituem assim um protótipo para a construção de um determinado artefato.

A Figura 8 ilustra os diversos conceitos do RUP tratados até o momento.

Figura 8 - Conceitos Básicos do RUP

FONTE: Tradução do original RUP on-line - 2001A.04.00.13.

2.2 Visão Geral do CMM para *Software* Nível 2

O Modelo de Maturidade da Capacidade para *Software* Nível 2 - CMM foi desenvolvido devido à "Crise do *Software*" que vem se mantendo há 2 ou 3 décadas. Tem-se então neste cenário os problemas clássicos que comprometem a qualidade de todo o desenvolvimento: qualidade, cronograma e custos não cumpridos.

O modelo SW-CMM foi desenvolvido pelo SEI, uma instituição federal de pesquisa e desenvolvimento dos Estados Unidos, mantida pelo DoD e operada pela CMU – *Carnegie*

Mellon University. Este modelo foi solicitado pelo DoD após a constatação do fracasso de uma série de projetos solicitados por ele (PAULK et al., 1999q, p. 3).

Ao longo dos trabalhos para a elaboração do modelo, identificou-se que os problemas enfrentados não eram técnicos e sim organizacionais. “Depois de 2 décadas de promessas vazias com respeito aos ganhos de qualidade e de produtividade devido a aplicação de novas metodologias e tecnologias de *software*, somente agora as organizações estão percebendo que seus problemas fundamentais encontram-se na incapacidade de gerenciar os **processos** de *software*.” (PAULK et al., 1999q, p. 3). “... os maiores problemas de hoje para com o desenvolvimento de *software* militar não são os problemas técnicos, mas os problemas gerenciais.” (PAULK et al., 1999o, p. 4).

O que agrava este cenário é que os avanços tecnológicos em termos de capacidade de processamento e complexidade dos *softwares* são muito maiores que os avanços de nossa maturidade para acompanharmos esta evolução dentro dos padrões de qualidade. Cada vez mais, as empresas são obrigadas a investirem em conhecimentos de novas tecnologias, treinamento de recursos humanos, novas metodologias de desenvolvimento e assim por diante, o que contribui para sufocá-las e dificultar a aplicação, de forma contínua e eficaz, de um processo de qualidade.

Desta forma, este modelo aborda muito mais o processo gerencial do desenvolvimento de *software*, deixando fracamente detalhadas as questões técnicas do desenvolvimento.

Pode-se considerar nos dias de hoje, como sendo o modelo de qualidade de *software* mais utilizado mundialmente, o que lhe garante ser um modelo de referência para qualquer empresa de desenvolvimento de *software*, mesmo que esta venha a utilizar um outro modelo qualquer.

O SW-CMM é um *framework* que descreve uma série de elementos-chave para um processo de *software* eficiente, permitindo a transformação da cultura organizacional de processos *ad hoc* ou imaturos, para processos disciplinados. Para que uma organização passe para um processo disciplinado, é necessário que atinja diversos níveis de maturidade, o que constitui sua estrutura.

2.2.1 Níveis de Maturidade

O SW-CMM é estruturado sob 5 níveis de maturidade, onde o nível 1 indica o nível de menor maturidade e o 5 o de maior maturidade, conforme ilustra a Figura 9.

Figura 9 - Níveis de Maturidade do SW-CMM

FONTE: Traduzida do site - http://www.sei.cmu.edu/cmm.

- **O nível 1** é considerado Inicial. Este nível é pertencente às organizações com processos caóticos ou *ad hoc*, onde não existe um ambiente estável para desenvolvimento e manutenção de *software*. A grande característica deste nível é que o sucesso dos projetos depende diretamente de pessoas altamente talentosas da equipe de desenvolvimento, muitas vezes chamados de super-heróis, bombeiros ou salvadores da pátria.
- **O nível 2** é considerado Repetível. Neste nível as organizações possuem um processo gerencial que é capaz de rastrear o custo, o cronograma e as funcionalidades de um sistema. O processo gerencial passa a ser disciplinado, e por conseguinte, as histórias de sucesso tornam-se repetitivas para desenvolvimentos de outros projetos similares.
- **O nível 3** é considerado Definido. Neste nível a organização passa a ter algum controle interno do desenvolvimento, ou seja, não apenas da parte gerencial, mas também da parte técnica ou de engenharia. Os processos de desenvolvimento e de manutenção são então definidos, padronizados e consistentes, onde as atividades de engenharia de *software* e de manutenção tornam-se estáveis e repetíveis.

- **O nível 4** é considerado Gerenciado. Neste nível a organização consegue realizar medições de algumas atividades internas de qualidade e produtividade no processo de desenvolvimento e de manutenção em todos os sistemas. Estas medidas vão sendo analisadas e guardadas e com isto, tanto o processo quanto o produto, podem ser entendidos e controlados de forma quantitativa. Com isto a organização consegue prever quais são as tendências com relação à qualidade do processo corrente e de produtos futuros.
- **O nível 5** é considerado Em Otimização. Neste nível existe o conceito da melhoria contínua do processo. São feitas análises dos pontos fortes e pontos fracos no processo, com tomadas de ações de forma pró-ativa, procurando ao máximo prevenir a ocorrência de defeitos. Com isto estas ações provocam realimentações quantitativas e qualitativas no processo. A eficiência do processo de *software* também é avaliado para a inserção de novas idéias e tecnologias, de acordo com a relação de custo e benefício.

A Figura 10 ilustra a visibilidade que a organização possui no processo de desenvolvimento de *software* em cada nível de maturidade. Observa-se que no nível 1 não se tem conhecimento do processo. Sabe-se apenas que são fornecidas algumas entradas para o processo, e que de alguma forma, tem-se o produto na saída. No nível 2 ainda não se sabe o que acontece internamente no processo de desenvolvimento, mas este passa a ser dividido em estágios. Nos intervalos entre os estágios, pode-se realizar medidas para verificar se os resultados dos produtos de trabalho do estágio anterior estão adequados para serem entregues ao próximo estágio. Já no nível 3 tem-se conhecimento do processo que ocorre internamente nos estágios de desenvolvimento e com isto permite que sejam controlados. No nível 4 o processo passa a ser medido, e com isto pode ser controlado de forma quantitativa. Finalmente no nível 5, a organização consegue identificar partes do processo que podem ser substituídas por outras melhores em virtude da análise de dados coletados.

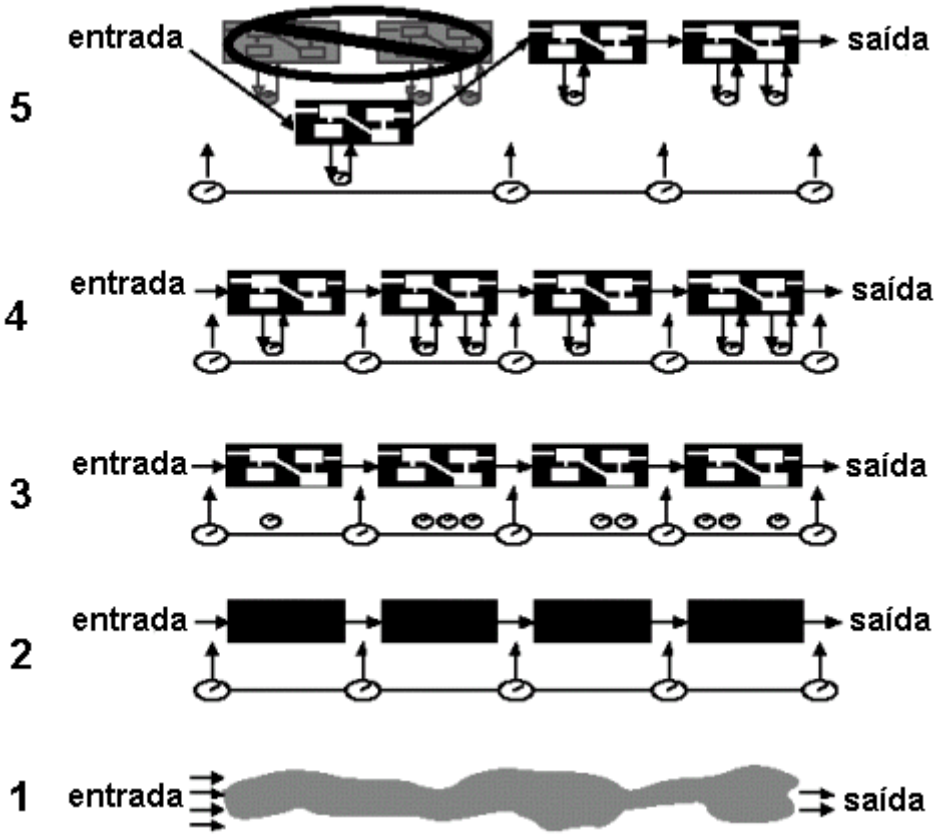

Figura 10 - Visibilidades em Cada Nível do SW-CMM (PAULK et al., 1999s, p. 23).

2.2.2 A Estrutura do SW-CMM

O entendimento de todos os níveis do SW-CMM e seu detalhamento são simples de serem entendidos, a partir do momento que se entende a sua estrutura.

Com exceção do nível 1 que não possui nenhum detalhamento, os demais níveis podem ser decompostos em partes estruturais idênticas, porém com conteúdos distintos. Estas partes podem ser visualizadas conforme a Figura 11.

Figura 11 – A Estrutura do SW-CMM (GONÇALVES, BOAS, 2001).

A hierarquia mais alta da estrutura do SW-CMM compreende os **níveis de maturidade**. Como foi visto, estes níveis variam de 1 a 5, onde cada um compreende uma determinada maturidade da organização para com o processo de *software* em busca constante pela melhoria da qualidade. Cada nível de maturidade à exceção do nível 1, contém uma série de **Áreas de Processo-Chave –** KPA, que compreende o segundo nível hierárquico da estrutura do SW-CMM. Os agrupamentos de KPAs de cada nível de maturidade definem uma série de metas que devem ser alcançadas para que o nível de maturidade respectivo possa ser considerado alcançado. Cada KPA se refere a uma área de processo-chave de um determinado nível, onde cada nível possui 2 ou mais áreas de processo-chave. Estas áreas de processo-chave definem

os aspectos que devem ser atacados para que a maturidade do nível corrente possa ser alcançado. Para os níveis 3, 4 e 5, tanto suas KPAs quanto todas abaixo do nível corrente devem ser atendidas. As metas de cada KPA definem escopo, limites e intenção para que ações possam ser tomadas e que com isto as metas possam ser alcançadas. Ao todo o SW-CMM define 18 KPAs, sendo 6 para o nível 2, 7 para o nível 3, 2 para o nível 4 e 3 para o nível 5. Os nomes de todas as KPAs são:

1. Para o Nível 2
    1.1. Gestão de Requisitos
    1.2. Planejamento de Projeto de *Software*
    1.3. Supervisão e Acompanhamento de Projeto de *Software*
    1.4. Gestão de Subcontratação de *Software*
    1.5. Garantia da Qualidade de *Software*
    1.6. Gestão de Configuração de *Software*
2. Para o Nível 3
    2.1. Foco no Processo da Organização
    2.2. Definição do Processo da Organização
    2.3. Programa de Treinamento
    2.4. Gestão de *Software* Integrada
    2.5. Engenharia do Produto de *Software*
    2.6. Coordenação Intergrupos
    2.7. Revisão por Pares
3. Para o Nível 4
    3.1. Gestão Quantitativa de Processo
    3.2. Gestão de Qualidade de *Software*

4. Para o Nível 5

   4.1. Prevenção de Defeitos

   4.2. Gestão de Mudança Tecnológica

   4.3. Gestão de Mudança de Processo

As KPAs são organizadas em **Características Comuns**, o que constitui o terceiro nível hierárquico da estrutura do SW-CMM. As Características Comuns são práticas que, em conjunto, irão descrever o que deve ser feito para se alcançar as metas de uma determinada KPA. Estas Características Comuns podem ser:

1. Compromissos – Comprometimentos da alta gestão da organização e definição de ações claras para que os objetivos da KPA possam ser alcançados.
2. Habilidades – Precondições necessárias para que o processo possa ser definido de forma competente pelas pessoas da organização, e irão envolver: recursos, estruturas organizacionais e treinamento.
3. Atividades – Atividades que deverão ser realizadas envolvendo papéis assumidos por pessoas e seguindo planos e/ou procedimentos documentados ou não, para que uma KPA possa ser implementada.
4. Medidas e Análise – Realização de medidas no processo e análises destas medidas para verificar o *status* do processo e definir ações no sentido de melhorá-lo.
5. Verificação da Implementação – Garantir que as atividades definidas pelo processo sejam realizadas de forma correta. Estas verificações incluem revisões e auditorias que devem ser feitas tanto pela gerência quanto pelo grupo de garantia da qualidade da organização.

Das 5 Características Comuns, a de Atividades é a que define o que deve ser feito e implementado com relação à KPA corrente. As demais servem como ferramentas para que estas atividades possam ser institucionalizadas na organização.

Cada Característica Comum contém uma série de **Práticas-Chave**, o que constitui o quarto e último nível hierárquico da estrutura do SW-CMM. Cada Prática-Chave pode ser entendida como sendo aquilo que deve ser feito para se implementar ou institucionalizar uma Característica Comum de uma KPA. Todavia, seu conteúdo não deve ser entendido como sendo a única forma de se fazer. Cada Prática-Chave é descrita em uma sentença e pode vir seguida de exemplos.

A tabela 1 apresenta as quantidades de metas e práticas-chave da estrutura do nível 2 de maturidade - Repetível:

KPA	Metas	Características Comuns				
Gestão de Requisitos	2	Compromissos	Habilidades	Atividades	Medição e Análise	Verificação de Implementação
		1 prática	4 práticas	3 práticas	1 prática	3 práticas
KPA	Metas	Características Comuns				
Planejamento de Projeto de *Software*	3	Compromissos	Habilidades	Atividades	Medição e Análise	Verificação de Implementação
		2 práticas	4 práticas	15 práticas	1 prática	3 práticas
KPA	Metas	Características Comuns				
Supervisão e Acompanhamento de Projeto de *Software*	3	Compromissos	Habilidades	Atividades	Medição e Análise	Verificação de Implementação
		2 práticas	5 práticas	13 práticas	1 prática	3 práticas
KPA	Metas	Características Comuns				
Gestão de Subcontratação de *Software*	4	Compromissos	Habilidades	Atividades	Medição e Análise	Verificação de Implementação
		2 práticas	3 práticas	13 práticas	1 prática	3 práticas
KPA	Metas	Características Comuns				
Garantia da Qualidade de *Software*	4	Compromissos	Habilidades	Atividades	Medição e Análise	Verificação de Implementação
		1 prática	4 práticas	8 práticas	1 prática	3 práticas
KPA	Metas	Características Comuns				
Gestão de Configuração de *Software*	4	Compromissos	Habilidades	Atividades	Medição e Análise	Verificação de Implementação
		1 prática	5 práticas	10 práticas	1 prática	3 práticas

Tabela 1 – Quantidade de Metas e Práticas-Chave do Nível 2 do SW-CMM

A título de exemplo, é dado a seguir uma descrição de toda a estrutura do nível 2 para a KPA de Gestão de Requisitos, à exceção dos detalhamentos das Práticas-Chave.

Nível de Maturidade 2 – Repetível

KPAs:

Gestão de Requisitos

Meta 1 – Os requisitos alocados ao *software* devem ser controlados para que seja estabelecida uma *baseline* para uso na manutenção e engenharia de *software.*

Meta 2 – Os planos, produtos e atividades de *software* são mantidos consistentes com os requisitos alocados aos *softwares*.

Características Comuns:

Compromissos

Práticas-Chave:

Compromisso 1 – O projeto segue uma política organizacional escrita para se gerenciar os requisitos do sistema alocados ao *software*.

Habilidades

Práticas-Chave:

Habilidade 1 – Para cada projeto é estabelecida uma responsabilidade para a análise dos requisitos do sistema e para alocá-los tanto ao *software* quanto ao *hardware* e outros componentes.

Habilidade 2 – Os requisitos alocados são documentados.

Habilidade 3 – Fundos e recursos adequados são providos para que os requisitos alocados possam ser gerenciados.

Habilidade 4 – Membros do grupo de engenharia de *software* e outros grupos relacionados a *software* são treinados para que possam realizar suas atividades de gestão de requisitos.

Atividades

Práticas-Chave:

Atividade 1 – O grupo de engenharia de *software* revisa os requisitos alocados antes que eles sejam incorporados ao projeto de *software*.

Atividade 2 – O grupo de engenharia de *software* utiliza os requisitos alocados como base para a

elaboração de planos de *software*, produtos de trabalho e atividades.

Atividade 3 – As mudanças nos requisitos alocados são revisados e incorporados ao projeto de *software*.

Medidas e Análise

Práticas-Chave:

Medida 1 – São realizadas medidas e elas são usadas para se determinar o *status* das atividades de gestão de requisitos alocados.

Verificação da Implementação

Práticas-Chave:

Verificação 1 – As atividades da gestão de requisitos alocados são revisadas com um gerente sênior sob uma base periódica.

Verificação 2 - As atividades da gestão de requisitos alocados são revisadas com um gerente de projeto sob uma base periódica ou na ocorrência de eventos.

Verificação 3 – O grupo de garantia da qualidade do *software* revisa e/ou audita as atividades e produtos de trabalho da gestão de requisitos alocados e relata os resultados.

A tabela 2 apresenta o total de componentes em toda a estrutura do SW-CMM para todos os níveis.

	Níveis	KPAs	Metas	Práticas-Chave
	1	-	-	-
	2	6	20	121
	3	7	17	108
	4	3	9	49
	5	2	6	38
Totais	5	18	52	316

Tabela 2 – Quantidade de Componentes do SW-CMM

2.2.3 Gestão de Requisitos

Nesta Área de Processo-Chave, procura-se manter sincronismo entre os requisitos fornecidos pelo cliente durante todo o desenvolvimento. Para isto, os requisitos do sistema devem ser devidamente alocados ao *software*, ao *hardware* ou a outros componentes, como por exemplo a recursos humanos. Esta alocação dos requisitos do sistema são tratados a partir de um acordo entre a organização que irá colher os requisitos e o cliente. A gestão de requisitos é a base para a previsão de estimativas, elaboração de planos, execução e acompanhamento das atividades de desenvolvimento durante todo o ciclo de vida do projeto. Nos casos de mudanças em um ou mais requisitos alocados, todos os documentos produzidos, produtos e atividades realizadas devem ser atualizados, mantendo-se assim o sincronismo entre o que o cliente deseja e o que está sendo desenvolvido. Esta KPA possui 2 metas e 12 Práticas-Chave distribuídas em: 1 compromisso, 4 habilidades, 3 atividades, 1 medição e 3 verificações.

2.2.4 Planejamento de Projeto de *Software*

Nesta Área de Processo-Chave, são tratadas as questões de como deverá ser planejado o desenvolvimento do *software*, sendo que este planejamento tem como entrada principal os resultados obtidos da KPA de Gestão de Requisitos. O planejamento deve abordar as questões de estimativas para o que deverá ser feito, tais como a quantidade de recursos e tamanhos do que deverá ser desenvolvido, firmar compromissos, tais como a elaboração de cronogramas,

onde os riscos deverão ser identificados, avaliados e negociados, e definição de um plano para o desenvolvimento do *software*, se for necessário. Caso este plano exista, ele deverá servir de base para a gestão de projeto e deverá possuir compromissos claros tanto da parte do cliente quanto do fornecedor, e que estarão fundamentados nos recursos a serem alocados. Além disto, este plano deverá incluir as restrições de projeto, bem como qual deverá ser sua capacidade final. Esta KPA possui 3 metas e 25 Práticas-Chave distribuídas em: 2 compromissos, 4 habilidades, 15 atividades, 1 medição e 3 verificações.

2.2.5 Acompanhamento e Supervisão de Projeto de *Software*

Nesta Área de Processo-Chave, utiliza-se como base o plano de desenvolvimento de *software* que possa ter sido criado na KPA de Planejamento de Projeto de *Software*. Desta forma, deve ser feito um acompanhamento das atividades realizadas durante o desenvolvimento e confrontadas com o que havia sido planejado. São feitas comparações entre o que foi planejado e o realizado com relação a tamanho, esforço, custo e cronograma. Caso sejam percebidos desvios significativos entre o que havia sido planejado e o realizado, devem ser tomadas ações corretivas e, se necessário, o plano de desenvolvimento de *software* deverá ser devidamente ajustado (replanejamento) para contemplar as mudanças ocorridas, devendo ser revisadas as demais atividades a serem realizadas. Esta KPA possui 3 metas e 24 Práticas-Chave distribuídas em: 2 compromissos, 5 habilidades, 13 atividades, 1 medição e 3 verificações.

2.2.6 Gestão de Subcontratação de *Software*

Nesta Área de Processo-Chave, prevê-se que parte ou todo o desenvolvimento de *software* deverá ser subcontratado, cabendo ao contratante o planejamento e acompanhamento do *software*, procurando gerenciar as interfaces de produto e processo entre as duas partes. O subcontratado deverá ser selecionado de acordo com o processo de subcontratação da organização contratante. A subcontratação é realizada com a definição de um acordo documentado entre as partes. Este acordo deverá contemplar requisitos técnicos e não técnicos (como por exemplo, data de entrega de produtos de trabalho), requisitos estes que servirão de base para que o contratante possa gerenciar as atividades do subcontratado. A aceitação dos produtos resultantes do acordo firmado entre as partes significa que o contratante está

concordando com os seus critérios de aceitação. Esta KPA possui 4 metas e 22 Práticas-Chave distribuídas em: 2 compromissos, 3 habilidades, 13 atividades, 1 medição e 3 verificações.

2.2.7 Garantia da Qualidade

Nesta Área de Processo-Chave, deve ser garantido que os projetos que estão usando o processo definido pela organização seja seguido e que os produtos de trabalho estejam atendendo a seus requisitos. Para isto, um grupo (chamado de grupo de garantia da qualidade) formado por uma ou mais pessoas realizam, de forma periódica ou por evento, revisões e auditorias nas atividades seguidas pela equipe de desenvolvimento, líderes de projeto e gerentes. Tem-se como objetivo garantir que as atividades definidas pelo processo e padrões sejam seguidos e que os produtos de trabalho satisfaçam aos requisitos definidos para o projeto. As atividades do grupo de qualidade devem ser relatadas às gerências da organização. Além disto, este grupo deve participar da elaboração de planos, padrões e procedimentos referentes a cada novo projeto que surgir, colaborando para a agregação de valores ao projeto, tais como restrições de projeto e obediência ao processo de desenvolvimento definido pela organização. Os problemas encontrados pelo grupo de garantia da qualidade durante as revisões e auditorias são solucionados pela equipe de projeto ou pelas gerências quando for o caso. Esta KPA possui 4 metas e 17 Práticas-Chave distribuídas em: 1 compromisso, 4 habilidades, 8 atividades, 1 medição e 3 verificações.

2.2.8 Gestão de Configuração

Nesta Área de Processo-Chave, deve ser mantida a integridade de todos os produtos de trabalho produzidos durante o desenvolvimento. Isto é válido para documentos, arquivos-fonte, objetos ou executáveis e ferramentas utilizadas durante o desenvolvimento, tais como os compiladores. Para isto, todos os produtos de trabalho são identificados e controlados quanto às mudanças sofridas entre uma versão e outra, de forma que possa ser feito um rastreamento de qualquer um deles, a qualquer momento, desde a sua criação inicial. Uma *baseline* contendo os produtos de trabalho iniciais deve ser criada e sujeita a alterações durante todo o desenvolvimento. Auditorias periódicas devem ou não ser realizadas nesta *baseline* para garantir sua consistência. Esta KPA possui 4 metas e 21 Práticas-Chave distribuídas em: 1 compromisso, 5 habilidades, 10 atividades, 1 medição e 4 verificações.

2.3 Conclusão

Atualmente, tanto o modelo SW-CMM quanto o processo RUP encontram-se em grande utilização pela comunidade de *software* e foram desenvolvidos por organizações altamente conceituadas como o SEI e a IBM Corporation. Segundo o relatório “Perfil de Maturidade de Processo”, que o SEI publica periodicamente, o mais recente relatório de abril de 2003 aponta que foram realizadas, até o momento, 2616 avaliações em 1978 organizações, envolvendo 10867 projetos de *software* espalhados em 51 países (CARNEGIE MELLON UNIVERSITY, 2003). Já a IBM Corporation, que adquiriu no ano passado a Rational Software Corporation, com experiência de 20 anos (IBM, 2003), construiu o RUP com a experiência de 3 consagrados especialistas em processos de desenvolvimento de *software* (Ivar Jacobson, Grady Booch e James Rumbaugh). Conclui-se que o SW-CMM e o RUP servem de bases sólidas para o desenvolvimento do LPA, processo proposto neste trabalho.

3 O LPA

Neste capítulo é apresentado o processo LPA, acrônimo para **L**evantamento, **P**lanejamento e **A**companhamento de projetos de *software* e que pode ser executado em um navegador da Internet. O processo navegável encontra-se disponível no Apêndice C – Conteúdo em CD , e pretende-se que este processo se torne disponível também na Internet para *download* e utilizado sem restrições. A Figura 12 apresenta uma visão sinóptica do LPA, visando facilitar ao leitor o entendimento do processo durante a leitura deste capítulo. Para a sua descrição, deve-se supor que a figura encontra-se dividida em 4 seções horizontais a partir do topo para a base. Na primeira seção horizontal superior tem-se os papéis assumidos pelas pessoas da organização durante o desenvolvimento de um projeto de *software*. Na segunda seção horizontal tem-se as disciplinas da gestão de requisitos e da gestão de planejamento e acompanhamento. Cada disciplina é composta de macroatividades, onde cada uma é novamente decomposta em atividades que são realizadas por um ou mais papéis assumidos durante o desenvolvimento. Na terceira seção horizontal tem-se todos os artefatos, na maioria deles documentos que são criados, consultados e/ou atualizados por um ou mais papéis, de acordo com as recomendações descritas nas atividades. Na quarta e última seção horizontal tem-se uma legenda da figura, mostrando os significados dos ícones e dos acrônimos utilizados.

Ao longo da descrição do processo foram inseridas figuras que são produzidas durante a execução do processo a partir de um navegador Internet. Algumas figuras irão mostrar diagramas de atividades formados por um conjunto de macroatividades de cada disciplina (ex. Figura 13), definindo uma seqüência de utilização destas macroatividades. Muitas figuras irão mostrar diagramas de atividades formados por um conjunto de atividades, chamado de fluxo de trabalho detalhado (ex. Figura 15). Estes diagramas explodem os conteúdos das macroatividades, definindo uma ordem para a execução de cada uma delas.

Com o intuito de destaque na descrição textual de todo o processo, a primeira letra dos papéis e artefatos foram mantidas em caixa-alta.

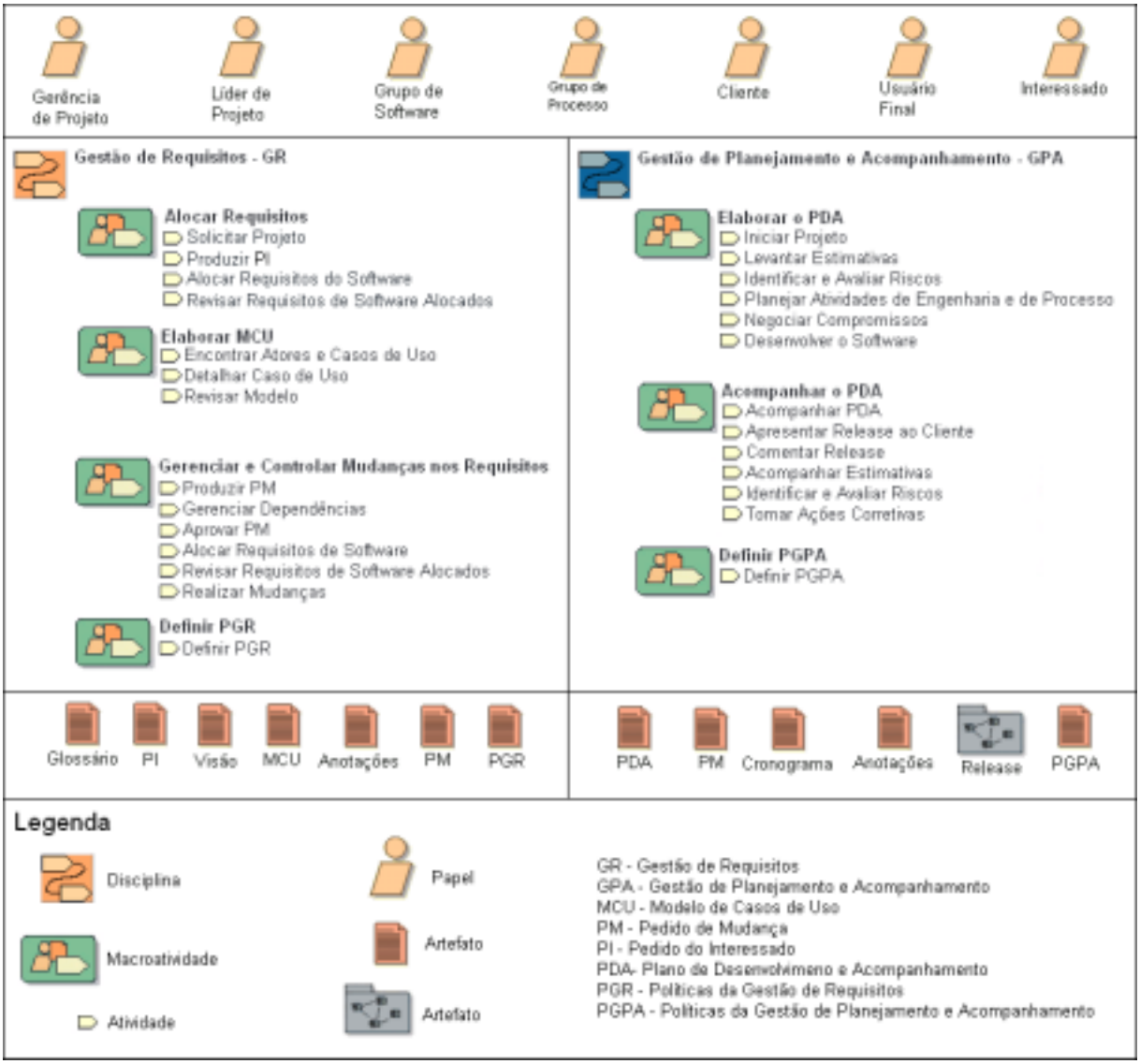

Figura 12 - Figura sinóptica do LPA

3.1 Definição de Políticas

De acordo com o SW-CMM, para cada KPA deve existir uma política documentada contendo as regras gerais que deverão ser seguidas. Como o LPA atende às KPAs da Gestão de Requisitos, Planejamento de Projeto de *Software* e Acompanhamento e Supervisão de Projeto de *Software*, são necessárias 3 políticas respectivamente. Todavia, como as KPAs de planejamento e de acompanhamento e supervisão são muito interrelacionadas, o LPA define apenas uma política para ambas.

3.1.1 PGR - Políticas para a Gestão de Requisitos

As políticas desta gestão cobrem a KPA de Gestão de Requisitos do SW-CMM e devem descrever as regras necessárias para que os requisitos alocados ao *software* possam ser adequadamente gerenciados, bem como estabelecer um entendimento entre ambas as partes Cliente/Organização, no qual os requisitos do Cliente serão utilizados como base para todo o desenvolvimento. Os requisitos alocados correspondem às entradas principais para o planejamento do desenvolvimento do *software*. A análise dos requisitos de *software* elabora e refina os requisitos alocados cujo resultado são os requisitos de *software* que devem ser documentados e implementados.

A Gestão de Requisitos deve cobrir:

- Estabelecimento e manutenção de um acordo comum para com os requisitos a serem desenvolvidos.
- Documentação dos requisitos funcionais ou técnicos.
- Documentação dos requisitos não-funcionais.
- Documentação dos requisitos não-técnicos.
- Documentação das estimativas atreladas aos requisitos.
- Documentação do planejamento para desenvolvimento dos requisitos.
- Implementação dos requisitos.
- Documentação de rastreamento dos requisitos ao longo do desenvolvimento de acordo com as atividades a eles atreladas.
- Controle das mudanças relacionadas com os requisitos.
- Replanejamento do desenvolvimento nas ocorrências de mudanças de requisitos.

As políticas adotadas pelo LPA são as seguintes:

1. Todo o projeto de *software* deverá seguir esta política.
2. Devem ser alocados recursos humanos e financeiros para que os requisitos possam ser identificados, desenvolvidos e acompanhados quanto a mudanças e cumprimento das necessidades do cliente.
3. Todos os participantes do time de desenvolvimento devem ser devidamente treinados para desempenharem suas atividades em um projeto de *software*.

4. Todos os requisitos alocados ao *software* deverão ser documentados inicialmente no documento Visão e posteriormente no Modelo de Caso de Uso.
5. Todos os requisitos alocados devem ser revisados pela Gerência de Projeto e pelo Líder de Projeto.
6. Os requisitos alocados devem servir de base para o planejamento do projeto e todos os planos, produtos de trabalho e atividades devem ser mantidos de forma consistente com as mudanças que ocorrerem nos requisitos alocados.

Esta política deve estar definida antes que um projeto se inicie, podendo ser mantida para todos os projetos da organização ou devidamente ajustada a cada novo projeto.

O Grupo de Processo é responsável pela elaboração deste artefato em conjunto com a Gerência de Projeto e o Líder de Projeto, se necessário.

3.1.2 PGPA - Política para a Gestão de Planejamento e Acompanhamento

Planejar e acompanhar as atividades a serem desempenhadas no desenvolvimento do *software*, fazendo-se uso intensivo de análise de riscos e geração de *releases*, para que o produto final a ser entregue atenda às necessidades do Cliente e Usuários Finais.

A Gestão de Planejamento e Acompanhamento deve cobrir:

- Gerência de riscos.
- Planejamento de cada fase e de cada iteração do projeto.
- Acompanhamento do progresso do projeto, realizando medidas adequadas para análise posterior.

As políticas adotadas pelo LPA são as seguintes:

Para a Gestão de Planejamento

1. O Líder de Projeto deve ser o responsável pela negociação de todos os compromissos do projeto.
2. Os requisitos de *software* alocados para desenvolvimento devem ser utilizados para o planejamento do projeto. Vide item 3.1.1.
3. Todos os compromissos do projeto identificados no PDA (Plano de Desenvolvimento e Acompanhamento) e Cronograma devem ser devidamente

negociados com todos os envolvidos, tanto internamente (compromissos internos) quanto externamente (compromissos externos).

4. Todos os envolvidos no projeto devem estar devidamente documentados no PDA.
5. As estimativas do projeto (tamanho, esforço, custo e cronograma) devem ser revisadas pela Gerência de Projeto.
6. A Gerência de Projeto deve revisar todos os compromissos externos do projeto, fazendo uma aprovação formal ou não.
7. O PDA deve ser gerenciado (mantido sob controle de versão) e controlado (com relação a mudanças da *baseline*).[4]

Para a Gestão de Acompanhamento

1. O Líder de Projeto é o responsável pelas atividades e resultados do projeto. Os requisitos de *software* alocados para desenvolvimento devem ser utilizados para o planejamento do projeto. Vide item 3.1.1.
2. Deve ser criado um plano de desenvolvimento (PDA) de forma documentada, para que seja utilizado no acompanhamento do projeto.
3. O PDA deve incluir um cronograma para distribuir as atividades, recursos e prazos para a realização das atividades de projeto.
4. A Gerência de Projeto deve ser mantida informada do andamento do projeto nos finais de fase ou a qualquer momento, se necessário.
5. O Líder de Projeto deve realizar reuniões periódicas para acompanhar o andamento do projeto. Devem participar desta reunião o Grupo de *Software* e outras pessoas e grupos, se necessário.
6. O Líder de Projeto deve realizar reuniões nos finais de cada fase de desenvolvimento (reuniões eventuais) para acompanhamento do projeto e apresentação de *release* para o Cliente. Devem participar desta reunião o Grupo de *Software* (parcial ou totalmente), o Cliente, a Gerência de Projeto e outras pessoas ou grupos, se necessário.

[4] Como o LPA não cobre a KPA de Gestão de Configuração e Mudança, cabe à organização implantar um controle de configuração e mudança, caso queira atender a esta política.

7. Os resultados das reuniões periódicas ou de finais de fase devem ser registrados de alguma forma e distribuídos para todos os interessados.
8. O Líder de Projeto deve tomar ações corretivas para reduzir riscos do projeto ou aplicar planos de contingência sempre que o projeto sair do rumo. Em todos os casos, o PDA e outros artefatos, se necessário, devem ser atualizados.
9. Todas as mudanças de compromissos, tanto interna quanto externamente, só podem ser formalizadas após devida negociação com os envolvidos.
10. A Gerência de Projeto deve revisar todas as mudanças de compromissos externos do projeto, fazendo uma aprovação formal ou não.

3.2 Definição das Macroatividades

As macroatividades irão oferecer uma visão de alto nível daquilo que deve ser feito para que a gestão específica possa ser executada. Para que isto seja facilitado, devem estar organizadas sob a forma de um diagrama de atividades da UML.

3.2.1 Macroatividades da Gestão de Requisitos

A Figura 13 ilustra o diagrama de atividades para as macroatividades desta gestão.

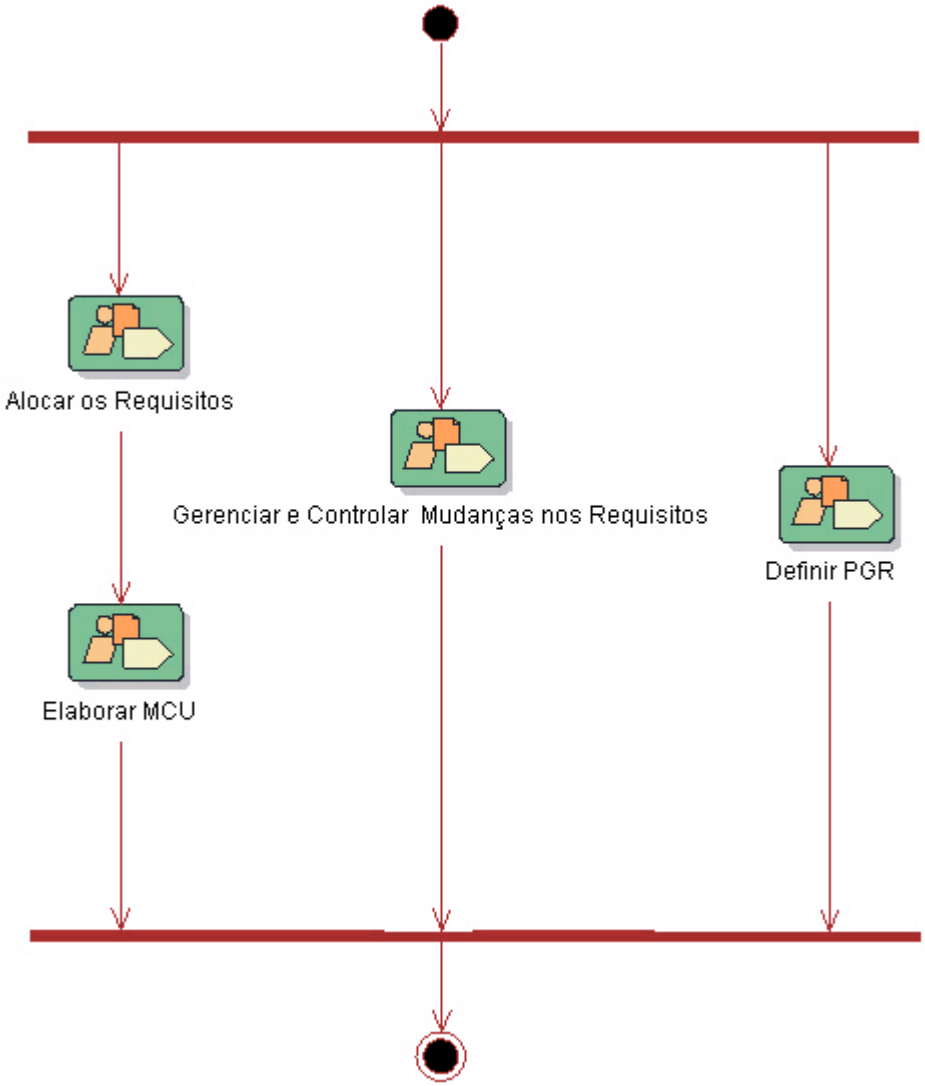

Figura 13 - Diagrama de Atividades da Gestão de Requisitos

Macroatividade Alocar Requisitos

Esta macroatividade consiste em obter os dados iniciais com o Cliente para o desenvolvimento ou manutenção de um projeto, identificar os requisitos que deverão ser desenvolvidos (requisitos alocados) e permitir uma revisão destes requisitos.

Macroatividade Elaborar MCU (Modelo de Casos de Uso)

Esta macroatividade consiste em fazer um detalhamento dos requisitos alocados, construir o MCU, detalhar um caso de uso e permitir uma revisão do modelo construído.

Macroatividade Gerenciar e controlar Mudanças nos Requisitos

Esta macroatividade deve gerenciar os requisitos alocados anteriormente e consiste na identificação das mudanças solicitadas, uma análise de impacto destas mudanças, uma decisão pela implementação ou não destas mudanças e a realização das mudanças em caso de aprovação.

Macroatividade Definir PGR

Esta macroatividade consiste na criação das Políticas da Gestão de Requisitos, contendo as regras gerais a serem seguidas nesta gestão para um projeto específico.

3.2.2 Macroatividades da Gestão de Planejamento e Acompanhamento

A Figura 14 ilustra o diagrama de atividades para as macroatividades desta gestão.

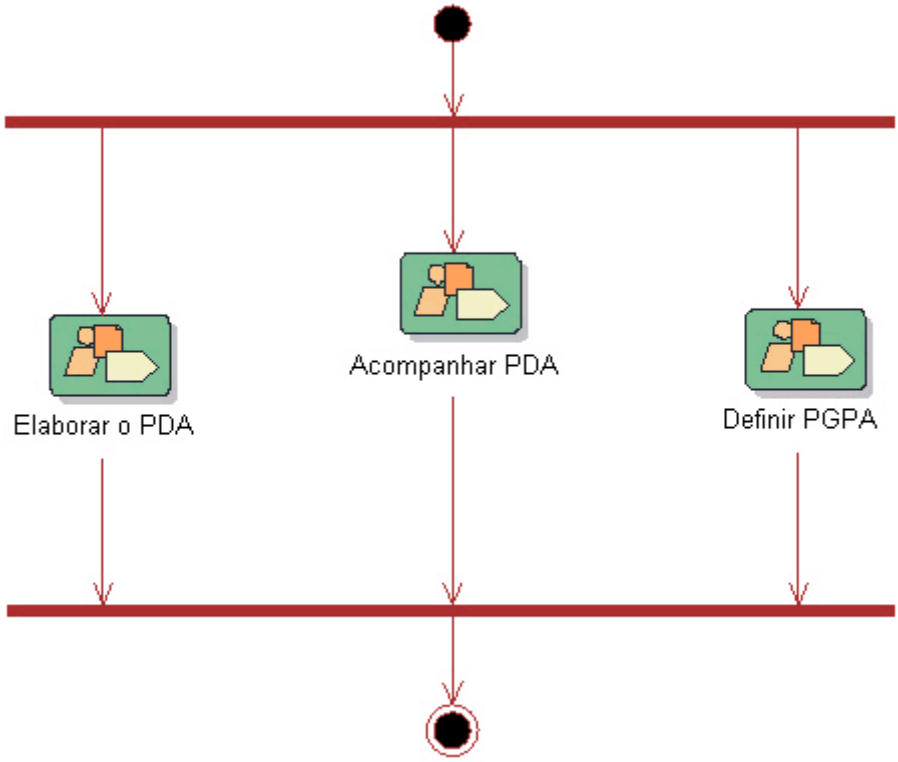

Figura 14 - Diagrama de Atividades da Gestão de Planejamento e Acompanhamento

Macroatividade Elaborar o PDA

A partir dos dados fornecidos pelo Cliente, realização de entrevistas e identificação dos casos de uso a serem desenvolvidos, pode ser iniciado o planejamento do sistema. O planejamento deve ser iniciado com a identificação da estrutura organizacional, as pessoas envolvidas no desenvolvimento, as interfaces externas ao sistema, levantamento de estimativas iniciais (tamanho, esforço, custo e prazo), elaboração de um cronograma inicial, previsão das *releases* a serem entregues para o Cliente e uma análise dos riscos envolvidos.

Macroatividade Acompanhar PDA

Esta macroatividade consiste em acompanhar de forma periódica e também eventual o andamento do desenvolvimento do sistema. Nos momentos de acompanhamento, devem ser feitas medidas, bem como a produção de estimativas para as etapas seguintes. Os valores estimados e reais devem ser registrados e analisados posteriormente pelo Grupo de Processo para que sejam analisados e aplicadas ações efetivas para a melhoria do processo. Estas ações deverão resultar em estimativas mais precisas nos novos projetos. O Cronograma também

deve ser acompanhado, e se existir alguma *release* a ser entregue, o Cliente deve estar presente ou seu representante. Caso ele não possa estar presente, pode-se optar pelo envio da *release* para avaliação. Ao final do acompanhamento, deve-se identificar se o projeto saiu ou não do rumo previsto e, neste caso, ações corretivas devem ser tomadas, envolvendo com isto um replanejamento das atividades. O PDA e o Cronograma são os artefatos principais desta macroatividade.

Macroatividade Definir PGPA

Esta macroatividade consiste na criação das Políticas da Gestão de Planejamento e Acompanhamento de projeto, contendo as regras gerais a serem seguidas nesta gestão para um projeto específico.

3.3 Descrição dos Fluxos de Trabalho Detalhados

Cada macroatividade é composta por uma ou mais atividades, executadas por um ou mais papéis que irão utilizar um ou mais artefatos de entrada e produzir ou atualizar um ou mais artefatos de saída. Esta composição é chamada de Fluxo de Trabalho Detalhado. Da mesma forma que foram criados diagramas de atividades da UML para as macroatividades, as atividades de cada macroatividade também encontram-se organizadas sob a forma de um diagrama de atividades da UML. Estes diagramas irão representar de forma mais detalhada a ordem de execução de cada uma das atividades, os artefatos a serem gerados e os papéis dos participantes.

3.3.1 Fluxo de Trabalho Detalhado das Macroatividades da Gestão de Requisitos

Para a Macroatividade Alocar os Requisitos:

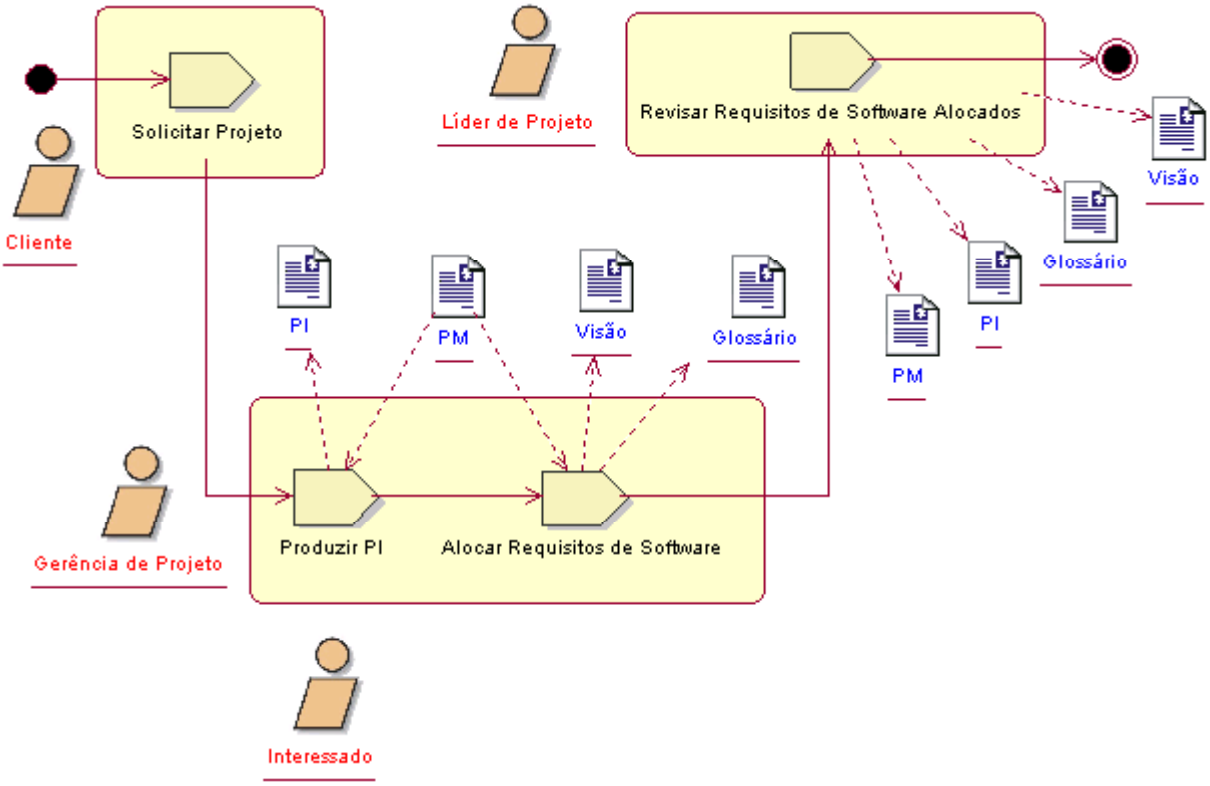

Figura 15 – Fluxo de Trabalho Detalhado da Macroatividade Alocar os Requisitos

Este fluxo de trabalho detalhado consiste em:

- Obter do Cliente uma proposta para desenvolvimento do sistema.
- Colher com o Cliente as informações do problema a ser solucionado, utilizando para isto o documento PI - Pedido do Interessado.
- Fazer uma análise destas informações coletadas e com isto identificar os requisitos que deverão ser alocados ao *software* utilizando o documento Visão.
- Criar um glossário com todos os termos técnicos usados durante o desenvolvimento, utilizando para isto o documento Glossário.
- Revisar os requisitos alocados ao *software*.

A responsabilidade de alocação dos requisitos é da Gerência de Projeto, embora o Cliente e os demais participantes do desenvolvimento devam contribuir.

Para a Macroatividade Elaborar MCU

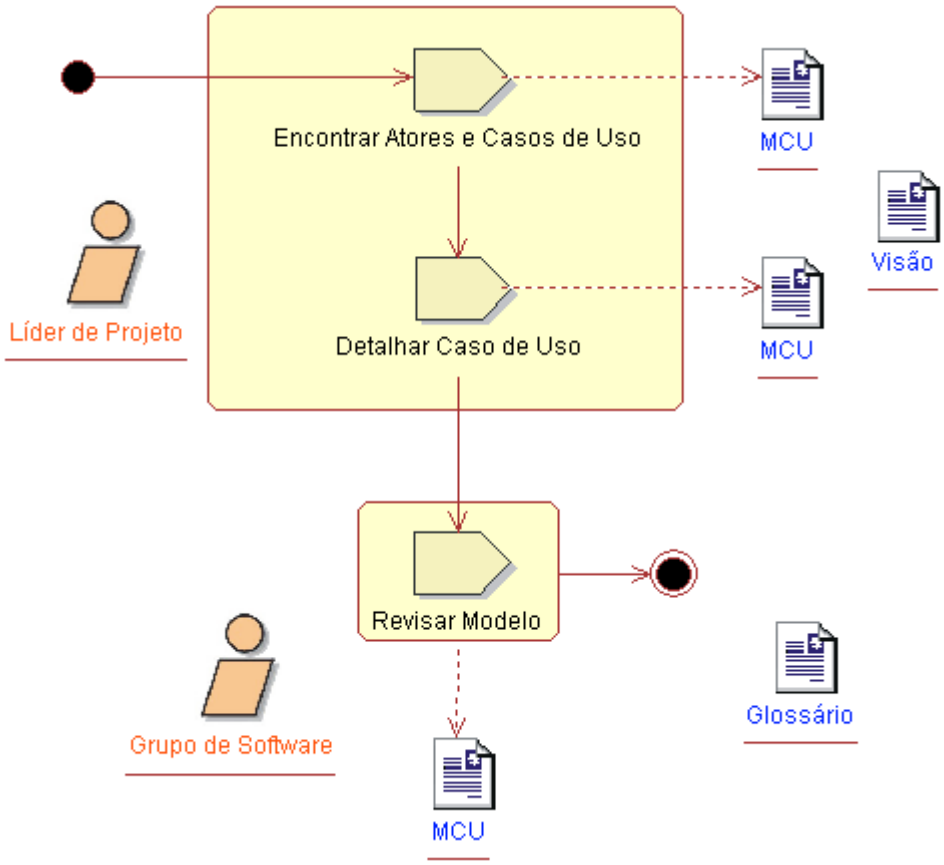

Figura 16 – Fluxo de Trabalho Detalhado da Macroatividade Elaborar MCU

Este fluxo de trabalho detalhado consiste em:

- Detalhar os requisitos de *software* alocados.
- Encontrar os atores e os casos de uso.
- Detalhar os casos de uso.
- Revisar o modelo.

Em todas as atividades deste fluxo de trabalho detalhado o documento a ser produzido deverá ser o MCU, servindo de base os documentos Visão e Glossário, que podem ser alterados para atualizações e melhorias.

A elaboração do MCU é algo que irá sendo evoluído ao longo do desenvolvimento. Primeiramente deve-se verificar se as características do produto, identificadas no documento Visão, encontram-se detalhadas em nível satisfatório para se iniciar a identificação dos casos de uso do sistema. Em seguida deve-se identificar os atores e casos de uso que farão parte do sistema. Deve-se detalhar os casos de uso de forma iterativa, partindo-se do mais prioritário e

complexo, devendo-se inicialmente desenvolver os caminhos ou cenários principais de todos eles até completar os demais cenários, gradativamente, durante as demais iterações. Ao final de cada iteração, ou sempre que necessário, deve-se proceder a uma revisão do MCU em desenvolvimento, para garantir completude e corretude dos requisitos que encontram-se em implantação.

A responsabilidade das atividades deste fluxo é do Líder de Projeto, tendo o Grupo de *Software* a incumbência de realizar uma revisão no MCU produzido durante o desenvolvimento.

Para a Macroatividade Gerenciar e Controlar Mudanças nos Requisitos

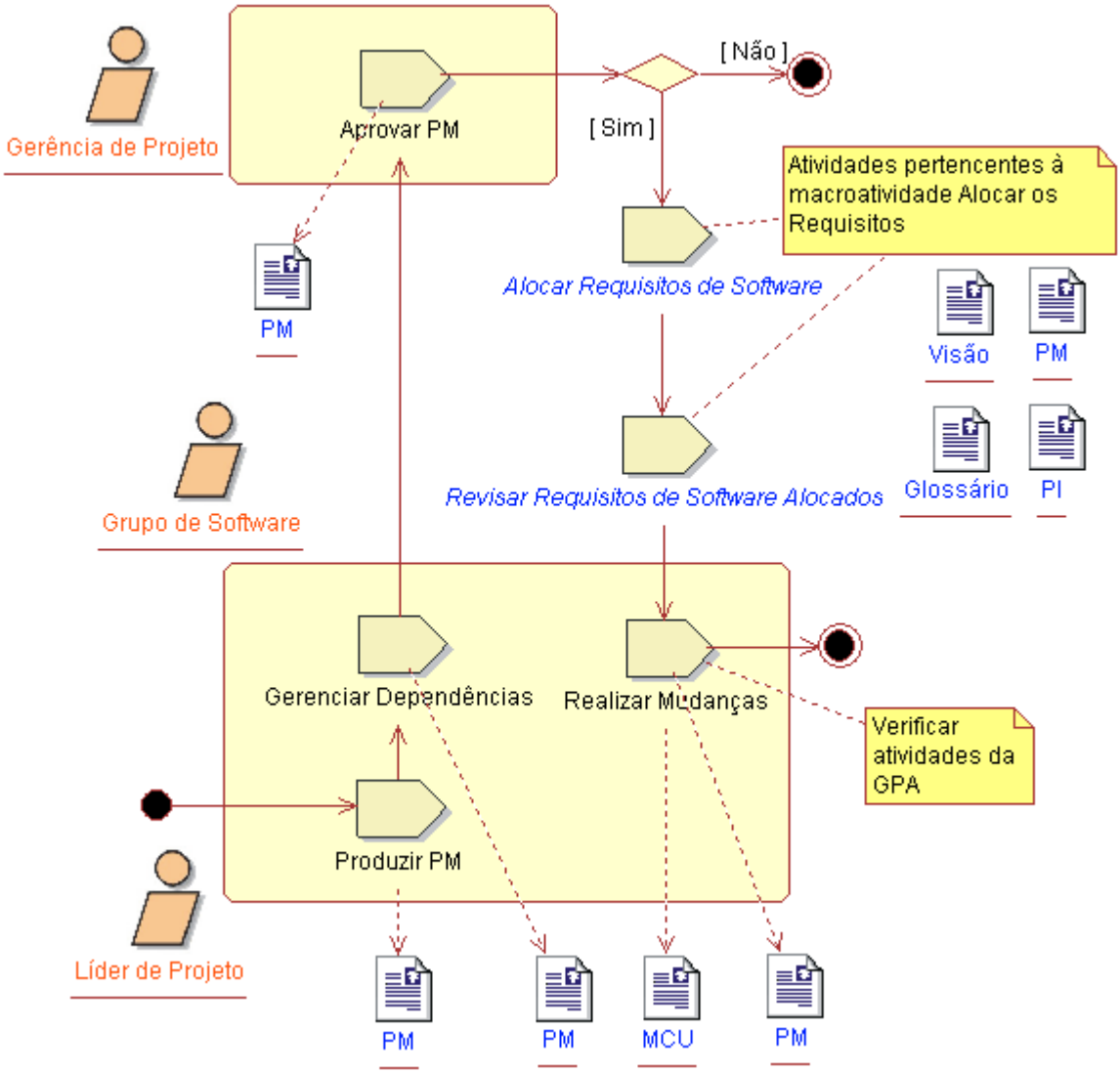

Figura 17 – Fluxo de Trabalho Detalhado da Macroatividade Gerenciar e Controlar Mudanças nos Requisitos

Este fluxo de trabalho detalhado consiste em:

- Avaliar os pedidos de mudança submetidos quanto ao seu impacto no conjunto de requisitos existentes do sistema.
- Identificar ou alterar o MCU.
- Atualizar o rastreamento dos requisitos.
- Verificar se os resultados da Gestão de Requisitos estão atendendo à visão do cliente para com o produto.

Por melhor que se faça a identificação dos requisitos do produto de forma clara, completa, com rastreamento e bem desenvolvidos, muito provavelmente eles irão mudar. A questão não é tentar evitar que eles mudem ou impedir que o Cliente realize mudanças. Isto simplesmente faria com que as necessidades do Cliente não fossem atendidas, inviabilizando o projeto. A questão deve então ser tratada no âmbito da gerência das mudanças destes requisitos.

As mudanças nos requisitos impactam naturalmente em algumas ou em todas as atividades e artefatos produzidos durante o desenvolvimento. Os relacionamentos das rastreabilidades identificadas na atividade Gerenciar Dependências identifica os relacionamentos existentes entre os requisitos e outros artefatos. Estes relacionamentos são a chave para o entendimento do impacto nas mudanças dos requisitos. Um outro conceito importante é o acompanhamento do histórico dos requisitos. Pela captura da natureza e razão das mudanças nos requisitos, pode-se coletar as informações necessárias para se responder a estas mudanças de forma apropriada.

O Cliente é a principal pessoa que realiza solicitações de mudanças de requisitos, sendo possível também que isto possa ser feito por qualquer pessoa do time de desenvolvimento, com o consentimento e aprovação do Cliente. Todavia, a pessoa responsável pela criação do Pedido de Mudança é o Líder de Projeto.

Para a Macroatividade Definir PGR – Política da Gestão de Requisitos

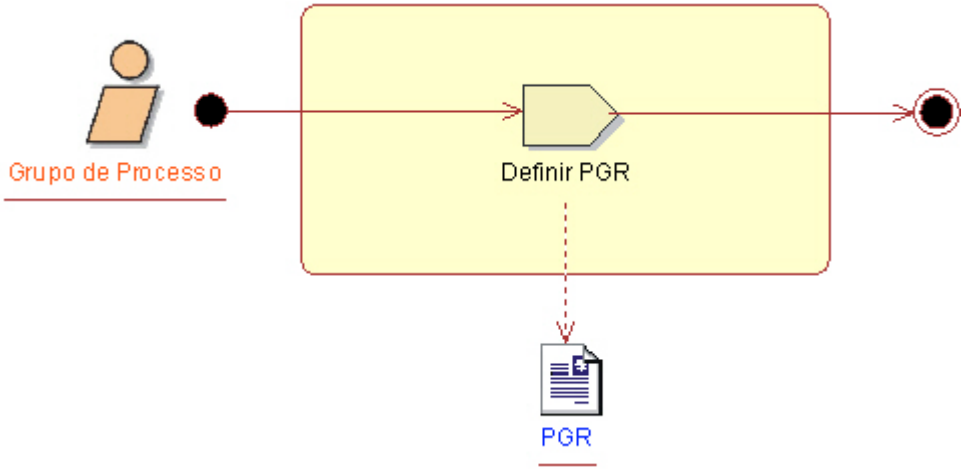

Figura 18 – Fluxo de Trabalho Detalhado da Macroatividade Definir PGR

Este fluxo de trabalho detalhado consiste em:

- Definir as regras a serem seguidas para a gerência dos requisitos alocados ao *software*.

Orientações:

- Definir um responsável pela documentação dos requisitos alocados ao *software*.
- Os requisitos têm que ser documentados.
- Os requisitos têm que ser revisados pela Gerência de Projeto, pelo Líder de Projeto e por outras pessoas ou grupos quando necessário.
- As mudanças de requisitos devem ser avaliadas, documentadas e o sistema deve ser replanejado.
- Devem ser produzidas estimativas referentes aos requisitos alocados ao *software*.
- O desenvolvimento deve ser mantido consistente com as mudanças de requisitos.

A PGR deve ser aplicada e seguida por toda a gerência e time de desenvolvimento. A responsabilidade pela criação das políticas é do Grupo de Processo, tendo-se conhecimento e aprovação da Gerência de Projeto.

3.3.2 Fluxo de Trabalho Detalhado das Macroatividades da Gestão de Planejamento e Acompanhamento

Para a Macroatividade Elaborar o PDA - Plano de Desenvolvimento e Acompanhamento

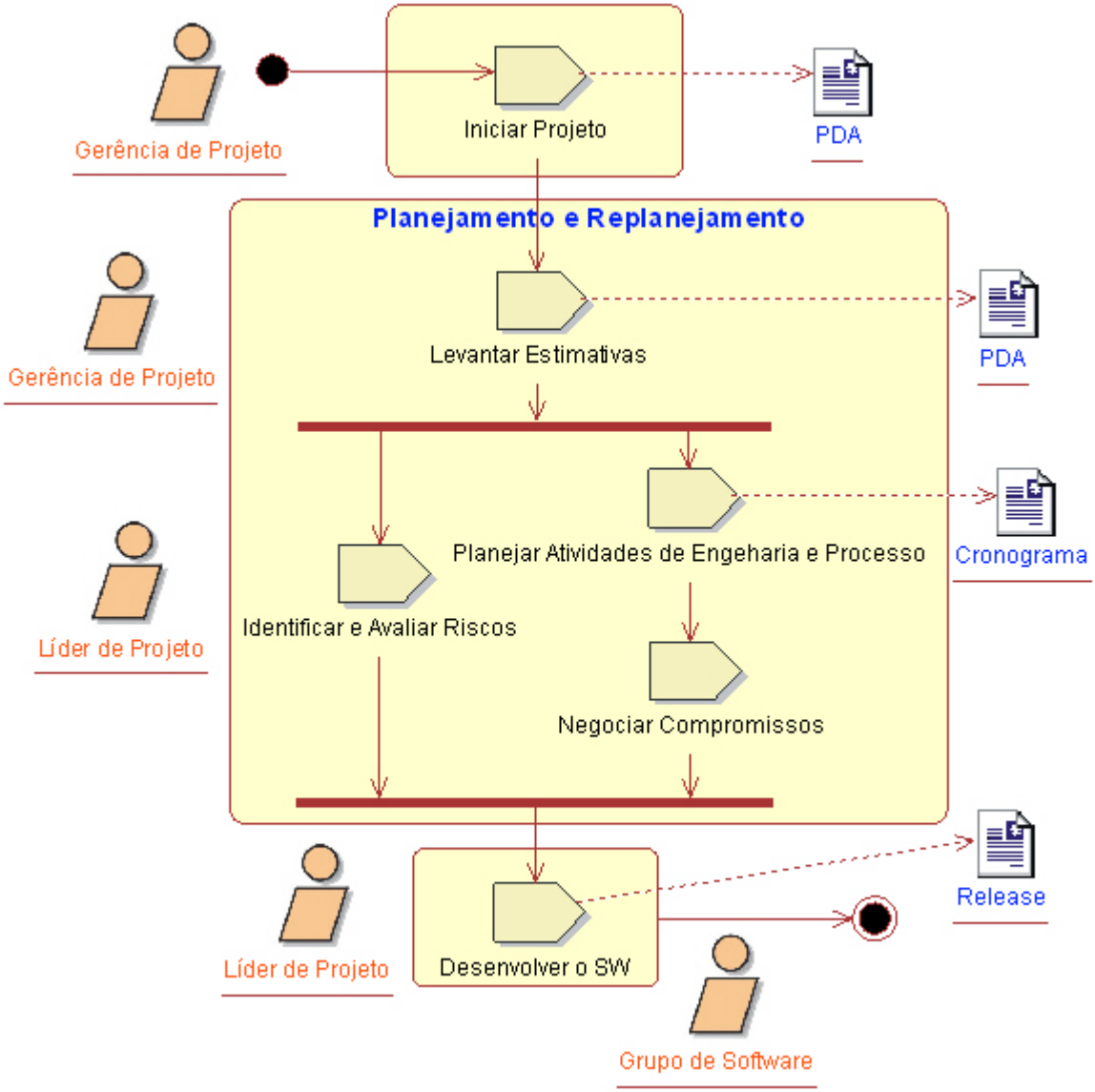

Figura 19 – Fluxo de Trabalho Detalhado da Macroatividade Elaborar o PDA

Este fluxo de trabalho detalhado consiste em:

- A partir dos dados fornecidos pelo Cliente, da realização de entrevistas e identificação dos casos de uso a serem desenvolvidos, deve-se iniciar o planejamento do sistema.
- O planejamento deve ser iniciado com a identificação da estrutura organizacional, as pessoas envolvidas no desenvolvimento, as interfaces externas ao sistema, levantamento de estimativas iniciais (tamanho, esforço, custo e prazo), elaboração de um Cronograma

inicial, previsão das *releases* a serem entregues para o Cliente e uma análise dos riscos envolvidos.

A responsabilidade pelo projeto deve ser do Líder de Projeto. O PDA deve ser iniciado pela Gerência de Projeto e refinado ao longo de todo o desenvolvimento pelo Líder de Projeto.

Para a Macroatividade Acompanhar PDA

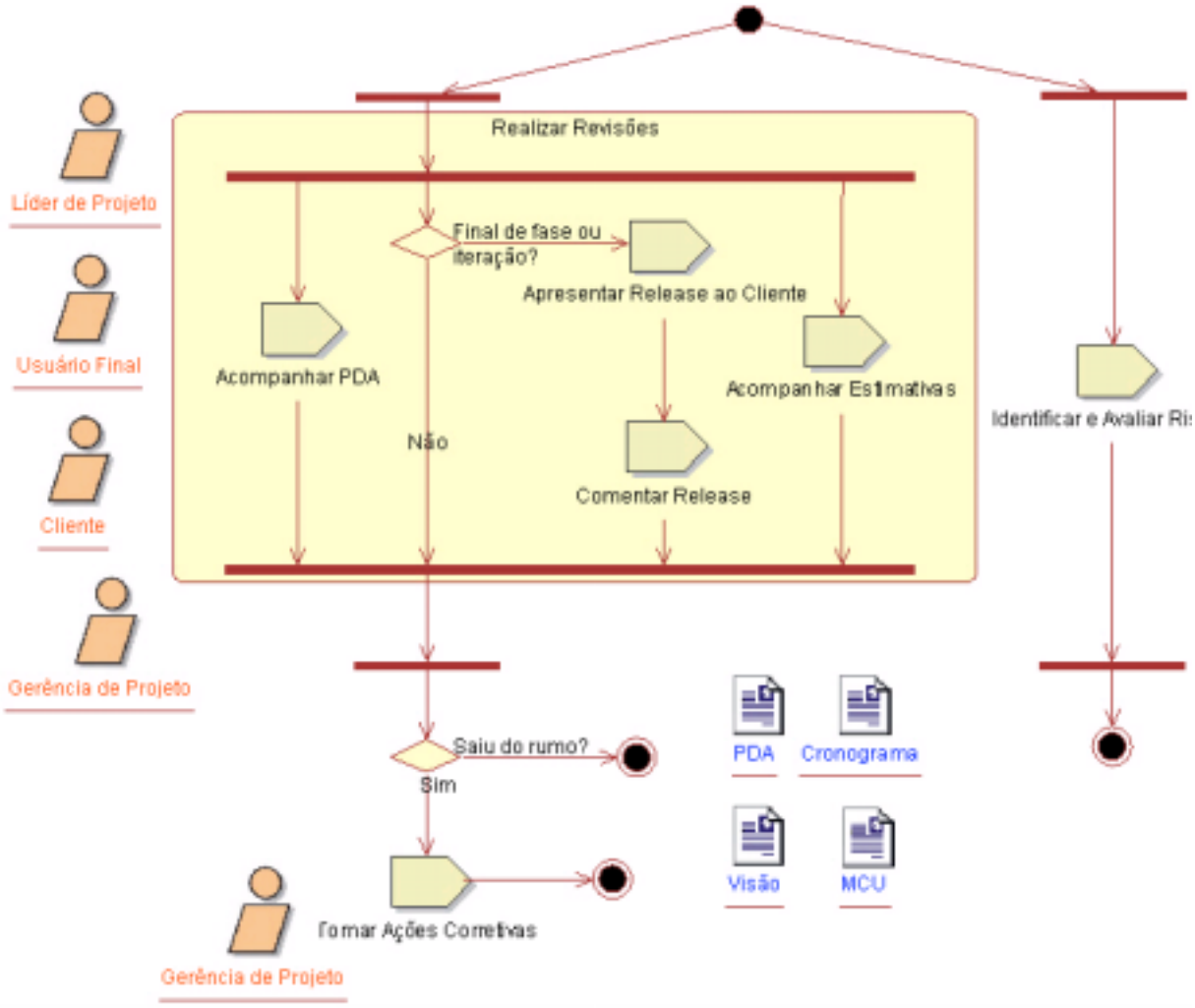

Figura 20 – Fluxo de Trabalho Detalhado da Macroatividade Acompanhar PDA

Este fluxo de trabalho detalhado consiste em:

- Acompanhar de forma periódica e também eventual o andamento do desenvolvimento do projeto.
- Nos momentos de acompanhamento, deve-se se fazer medidas e realizar novas estimativas para as etapas seguintes.

- Os valores estimados e reais devem ser registrados e analisados posteriormente pelo Grupo de Processo, para que sejam analisadas e aplicadas ações efetivas para a melhoria do processo.
- As ações tomadas com as análises dos valores reais medidos em comparação com os estimados devem resultar em estimativas mais precisas para novos projetos.
- O Cronograma também deve ser acompanhado, e se existir alguma *release* a ser entregue, o Cliente, ou seu representante deve estar presente, ou deve a mesma ser enviada para que ele possa fazer uma avaliação dos resultados.
- Ao final do acompanhamento, deve-se identificar se o projeto saiu ou não do rumo previsto e, neste caso, ações corretivas devem ser tomadas, envolvendo com isto um replanejamento das atividades futuras.
- O PDA e o Cronograma são os artefatos principais desta macroatividade.

Dos acompanhamentos periódicos devem participar o Líder de Projeto e o Grupo de *Software*. Dos acompanhamentos eventuais devem participar a Gerência de Projeto, o Líder de Projeto e o Grupo de *Software* (em parte ou no todo). Em ambos acompanhamentos o Cliente também deverá estar presente caso exista alguma *release* a ser entregue e demonstrada.

É de responsabilidade do Líder de Projeto definir a periodicidade das reuniões, tanto periódicas quanto não periódicas.

Para a Macroatividade Definir PGPA - Definir Políticas da Gestão de Planejamento e Acompanhamento

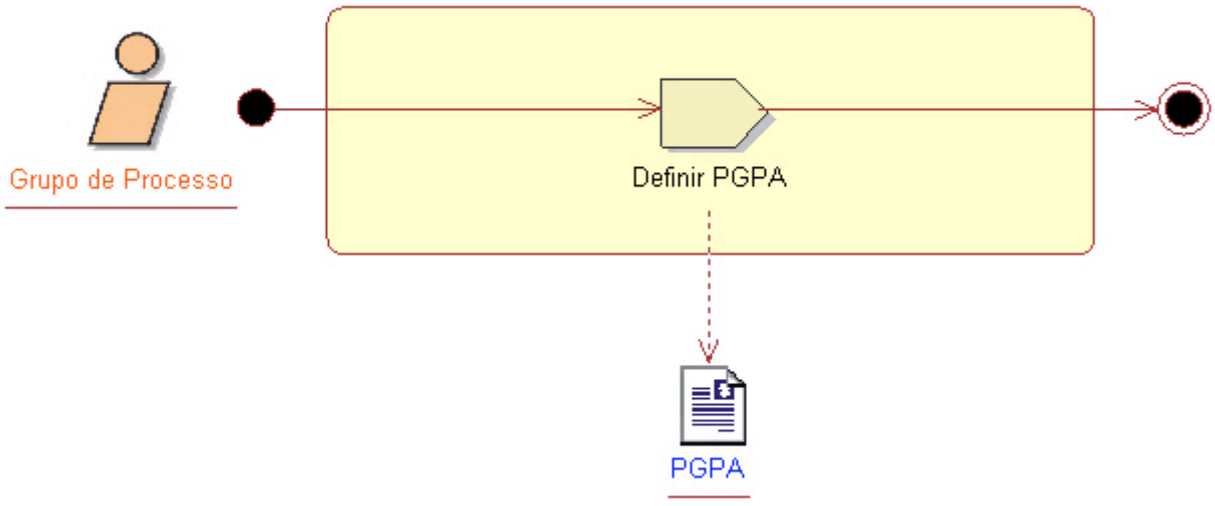

Figura 21 – Fluxo de Trabalho Detalhado da Macroatividade Definir PGPA

Este fluxo de trabalho detalhado consiste em:

- Definir regras claras para que sejam seguidas durante a gestão de planejamento e acompanhamento.

Orientações:

- Definir um responsável pela negociação dos compromissos e desenvolvimento do projeto de *software*.
- Definir um responsável pelos resultados do projeto.
- Os requisitos de *software* alocados para desenvolvimento devem ser utilizados para o planejamento e acompanhamento do projeto.
- Documentar o plano de desenvolvimento do projeto, utilizando para isto o documento PDA, e acompanhar seu desenvolvimento.
- No mínimo, o plano de desenvolvimento deve conter a relação de integrantes do projeto, organograma, as estimativas de projeto, cronograma, riscos e *releases* a serem entregues.
- As estimativas devem ser revisadas, bem como os compromissos externos.
- As informações do andamento do projeto devem ser repassadas para a Gerência do Projeto.
- Devem acontecer reuniões regulares do time de desenvolvimento para se acompanhar o andamento das atividades de desenvolvimento.
- O Cliente deve sempre ser envolvido nas reuniões de apresentação de *releases*.
- Todos os resultados das reuniões de acompanhamento devem ser divulgadas de forma impressa ou eletrônica.
- Devem ser conduzidas ações corretivas para reduzir a probabilidade de ocorrência de riscos.
- Devem ser conduzidas ações corretivas para recolocar o projeto em seu rumo natural de desenvolvimento sempre que um risco se fizer presente, providenciando replanejamento do projeto.

A Política da Gestão de Planejamento e Acompanhamento deve ser aplicada e seguida por toda a gerência e time de desenvolvimento. A responsabilidade pela sua criação é do Grupo de Processo, tendo-se conhecimento e aprovação da Gerência de Projeto.

3.4 Definição das Atividades da Gestão de Requisitos

As atividades em conjunto com os artefatos e papéis compõem a menor divisão do LPA e para cada uma tem-se uma descrição da finalidade da atividade e os passos necessários para a sua realização. Além disto, também é indicada a pessoa responsável pela atividade, os artefatos de entrada para a sua realização e os artefatos alterados ou produzidos como resultados, podendo conter ainda outras informações menos relevantes.

Os itens seguintes apresentam as atividades de cada fluxo de trabalho detalhado das macroatividades da gestão de requisitos.

3.4.1 Fluxo de Trabalho Detalhado da Macroatividade Alocar os Requisitos

Para a atividade Solicitar Projeto:

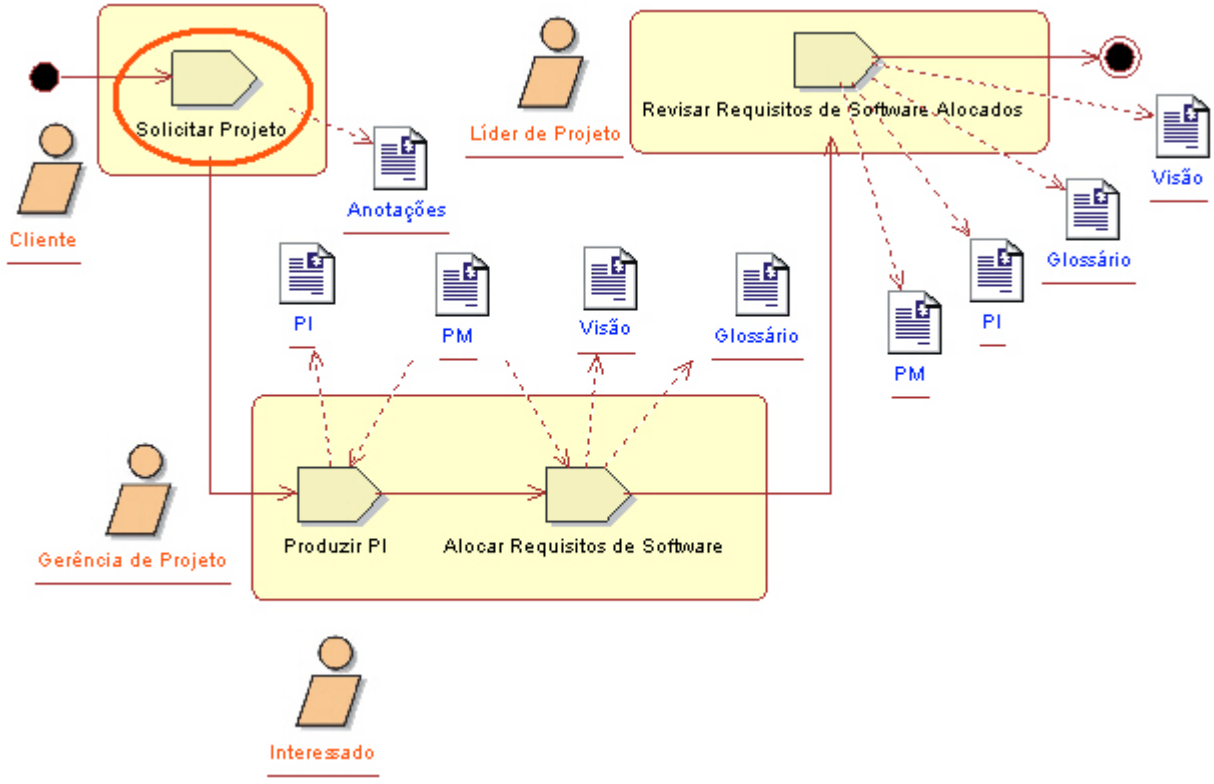

Figura 22 – Atividade: Solicitar Projeto

Artefatos de Entrada: Não definido pelo processo

Artefatos de Saída: Anotações

Papel: Cliente

Finalidade:

Apresentar para a organização que irá desenvolver o projeto um ou mais documentos que contenham informações suficientes para que o projeto possa ser estimado e iniciado. Esta atividade deve ser feita em conjunto com uma ou mais entrevistas a serem realizadas entre os interessados.

Passos:

1. **Fazer uma descrição textual do problema a ser solucionado**

   O interessado deve fazer uma descrição do problema que ele está enfrentando, justificando a necessidade de se ter uma solução em *software*.

2. **Fazer uma descrição textual daquilo que se deseja que o *software* realize**

   O Interessado deve fazer uma descrição de tudo aquilo que ele deseja que o *software* realize (funcionalidades), procurando detalhar as informações. Além das funcionalidades, o interessado deve incluir todas as restrições desejadas, tais como tipo de linguagem de programação, tempo de resposta a eventos, aspectos de segurança, portabilidade e outras informações que achar necessário.

3. **Informar as prioridades para cada funcionalidade desejada**

   O Interessado deve descrever as prioridades que deseja para cada funcionalidade e, se possível, uma justificativa para cada uma delas. Sugere-se que isto seja feito sob a forma de uma tabela.

4. **Identificar as pessoas de interface**

   O Interessado deve citar outros interessados pelo *software* que possam servir de interface para com a equipe de desenvolvimento, descrevendo seu papel no contexto de trabalho.

5. **Definir as estimativas de entrega**

   O Interessado deve descrever, de preferência sob a forma de uma tabela, os prazos que ele estima para a entrega do *software*. Além disto, deve incluir algum material que seja de sua

propriedade, sem o qual o time de desenvolvimento não terá condições de desenvolver o trabalho.

Para a atividade Produzir PI - Pedido do Interessado:[5]

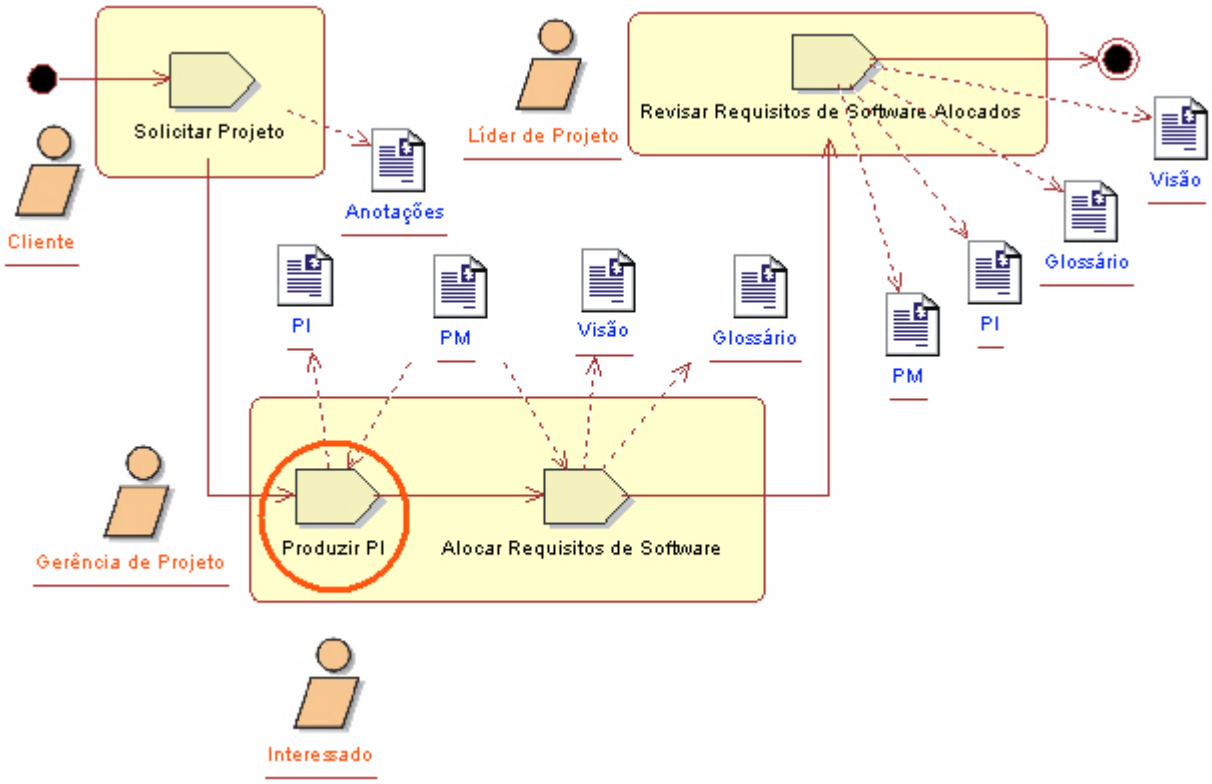

Figura 23 – Atividade: Produzir PI

Artefato de Entrada: PM (se existir)

Artefato Resultante: PI

Papel: Gerência de Projeto

Finalidade:

Entender quais são os Interessados no projeto, coletar os requisitos para preencher tudo aquilo que o *software* deverá atender e priorizar as solicitações dos Interessados.

[5] Esta atividade foi traduzida e adaptada da atividade "*Elicit Stakeholder Request*" do RUP 2002.05.00.25

Passos:

1. **Determinar as Fontes de Requisitos**

   Caso seja um sistema existente, esta atividade deverá ser feita utilizando-se também um PM - Pedido de Mudança formal, seguindo as descrições da atividade Produzir PM, descrita no item 3.4.3. Isto irá prover um bom começo para a coleta de dados e refinamento do PI. Deve-se procurar por parceiros, usuários, clientes, especialistas no domínio do problema apresentado e qualquer outra pessoa que possa representar o papel de Interessado. Deve-se determinar as pessoas que irão trabalhar na coleta das informações considerando o conhecimento, habilidades de comunicação, disponibilidade e importância. Estas pessoas irão atuar como interessadas pelo projeto, como se fosse um time de projeto estendido. Em geral, 2 a 5 pessoas interessadas é a quantidade adequada para se trabalhar durante todo o projeto. Manter grupos muito grandes dificulta a gerência das atividades e a alocação do tempo de forma eficiente. Estas pessoas não irão trabalhar em tempo integral no projeto. Tipicamente irão participar durante a coleta dos requisitos, durante as fases de Concepção e Elaboração e nas sessões de revisão. Encontre uma forma de aprender como os outros fariam aquilo que se está tentando fazer. Isto significa coletar informações competitivas, outros sistemas, como eles os utilizam e o que pode ser melhorado. Uma fonte importante na obtenção das informações é qualquer descrição existente sobre a organização no qual o sistema será utilizado.

2. **Coletar as Informações**[6]

   Entrevistas - Trata-se de um dos métodos mais úteis de coleta de informações com os Interessados. Durante as entrevistas, não se deve deixar de utilizar o PI. Não espere respostas simples, não pressione o entrevistado para que ele responda a todas as suas perguntas, procure fazer perguntas curtas e principalmente aprenda a ouvir, ouvir e ouvir.

   Questionários:

   Esta é uma técnica muito utilizada. Após a condução de diversas entrevistas, pode-se perceber que algumas informações irão surgir de forma repetitiva. Este tipo de informação pode ser coletado para formar um conjunto de perguntas com respostas típicas que poderá

[6] O leitor poderá utilizar outras formas para a coleta de informações além das apresentadas neste item, tais como a leitura de documentos.

ser enviado para os Interessados. Este método permite reunir estatísticas das respostas dadas por eles. A chave, contudo, é ser capaz de formular perguntas que irão apresentar respostas cujas estatísticas correspondam àquilo de que eles realmente precisam.

Os Interessados podem receber as perguntas e enviar as respostas pela Internet. Isto permitirá coletar informações de uma maior quantidade de pessoas do que diretamente por meio de entrevistas, tendo-se porém menor controle dos resultados. Devido ao contato apenas eletrônico, sua eficiência é menor, o que pode levar a entendimentos incorretos. Os questionários podem ser uma ferramenta muito poderosa, mas não substituem as entrevistas. As entrevistas permitem que se formulem frases em conjunto com o Interessado e com o entendimento dele, tornando-as mais consistentes.

3. Avaliar os Resultados

Certifique-se de que:

- foi atribuída uma prioridade para cada requisito.
- existem informações das fontes de requisitos, tanto por meio de documentos quanto por meio da identificação de pessoas.
- foram corrigidas possíveis inconsistências entre os requisitos.

Para a atividade Alocar Requisitos de *Software*:[7]

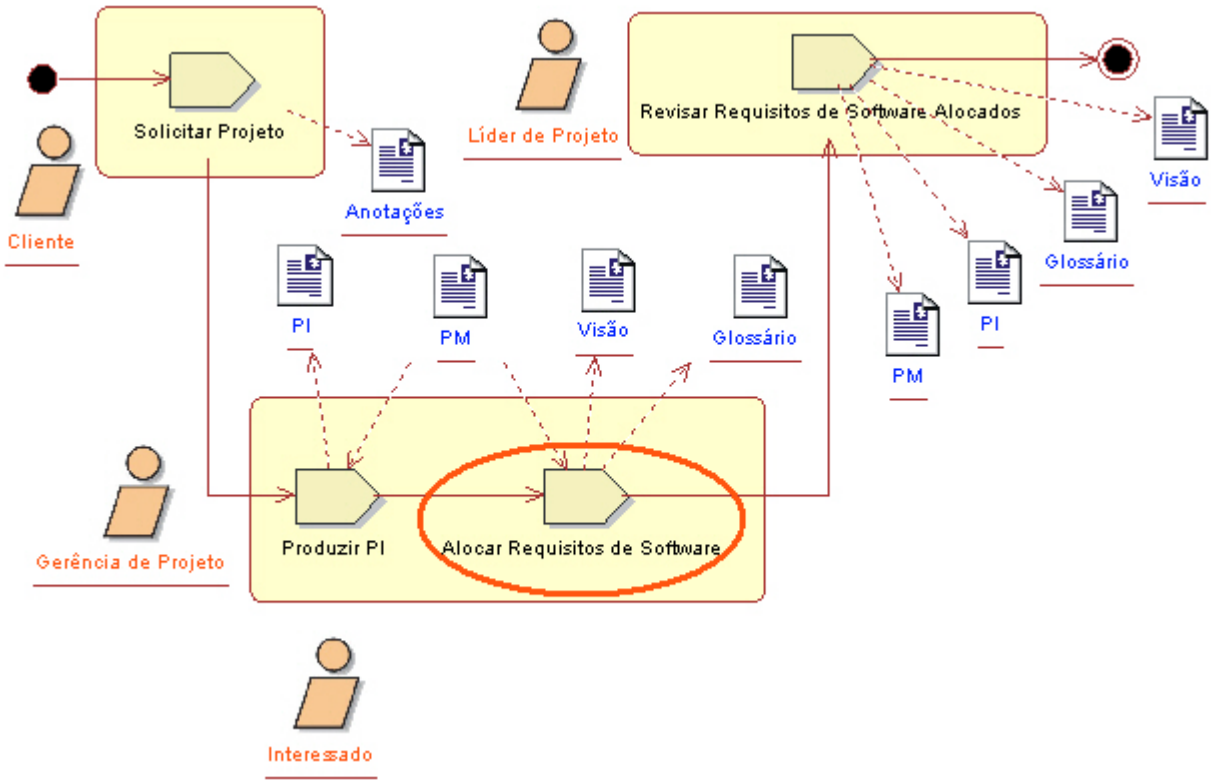

Figura 24 – Atividade: Alocar Requisitos de *Software*

Artefatos de Entrada: PI e PM (se existir)

Artefatos de Saída: Glossário e Visão

Papel: Gerência de Projeto

Finalidade:

Chegar a um acordo dos problemas que precisam ser solucionados, identificar os Interessados para o sistema, definir as fronteiras do sistema e descrever suas principais características sob a forma de requisitos.

Passos:

1. Chegar a um Acordo a Respeito do Problema que Deverá ser Solucionado.

Uma das formas mais simples de se obter um acordo da definição do problema é escrevê-lo e ver se todos concordam.

---

[7] Esta atividade foi traduzida e adaptada da atividade "*Develop Vision*" do RUP 2002.05.00.25.

Pergunte ao grupo: Qual é o problema?

É muito comum que as pessoas fiquem divagando na solução do problema em vez de consumir o tempo no entendimento do mesmo. Deve-se escrever o problema e verificar se é possível obter o acordo de todos com a definição produzida.

Em seguida pergunte ao grupo: Qual é o problema real?

Busque pelas causas dos problemas, ou "o problema por detrás do problema". O problema real freqüentemente fica escondido por detrás daquilo que percebemos ser o problema.

Não aceite a primeira declaração do problema. Continue perguntando "Por quê?", para encontrar aquilo que seja realmente o problema.

Algumas vezes o grupo pode ficar envolvido com uma solução visionária, difícil de alcançar para formular aquilo que seja realmente o problema. Em tais casos, pode ser benéfico explorar os benefícios da solução e então tentar encontrar os problemas que serão solucionados por estes benefícios. Pode-se explorar se os problemas identificados são ou não realmente os problemas da organização. Técnicas comuns que são usadas para encontrar o problema que existe por detrás do problema são *brainstorming*, diagramas espinha de peixe e diagrama de Pareto.

2. Identificar os Interessados

Dependendo da experiência do time de desenvolvimento, a identificação dos Interessados torna-se uma tarefa trivial ou não. Freqüentemente, isto é obtido realizando-se entrevistas com pessoas que tomam decisões, usuários em potencial e outras pessoas interessadas. Uma boa estratégia é fazer as seguintes perguntas para as pessoas:

- Quem são os usuários do sistema?
- Quem são os compradores econômicos do sistema?
- Quem mais seria afetado pelas saídas que o sistema irá produzir?
- Quem irá avaliar e qualificar o sistema quando ele for liberado e implantado?
- Existe algum outro usuário interno ou externo do sistema que precisará ser envolvido?
- Quem irá manter o novo sistema?
- Existe mais alguém?
- Ok, existe mais alguém?

Inicie desenvolvendo perfis de usuários potenciais ou reais do sistema. Isto irá mapear os atores do sistema que será desenvolvido.

3. Definir as Fronteiras do Sistema

As fronteiras do sistema definem os limites entre a solução e o mundo real que circunda a solução. Em outras palavras, as fronteiras do sistema descrevem um envelope no qual está contida a solução do problema. As informações de entrada e saída são passadas dos usuários para o sistema e vice-versa, constituindo uma comunicação entre os atores e o sistema. Todas as interações com o sistema irão ocorrer via interfaces das fronteiras entre o sistema e o mundo externo.

As interfaces entre os usuários e a aplicação da maiorias dos sistemas aplicativos consistem das janelas de diálogo que eles utilizam para acessá-lo e quaisquer relatórios e caminhos de comunicação que o sistema usa para documentar ou transmitir as informações desejadas.

É altamente eficiente utilizar os atores para se definir e descrever as fronteiras do sistema. Veja a atividade Encontrar Atores e Casos de Uso descrita no item 3.4.2.

4. Identificar as Restrições a Serem Impostas ao Sistema

Existe uma série de fontes de restrição a serem consideradas. Muitas destas restrições podem ser documentadas no documento MCU. Segue uma lista de perguntas que correspondem a fontes de restrições em potencial a serem investigadas:

- Política: Existem questões políticas internas ou externas que afetam as soluções em potencial?
- Econômica: Quais são as restrições financeiras e orçamentárias aplicáveis? Existem considerações com relação aos custos e preço do produto? Existem questões de licenciamento?
- Ambiental: Existem restrições reguladoras ou ambientais? São legais? Existem outros padrões a que estamos restritos?
- Técnico: Existem restrições quanto à escolha das tecnologias a serem utilizadas? Existem restrições quanto às plataformas de desenvolvimento? Pode-se usar alguma tecnologia nova?

- Viabilidade: O cronograma está definido? O time irá ficar restrito aos recursos existentes atualmente na organização? Pode-se utilizar trabalhos de terceiros? Pode-se expandir os recursos? De forma temporária? De forma permanente?
- Sistema: A solução a ser construída já existe em algum sistema desenvolvido pelo time anteriormente? Deve-se manter compatibilidade com as soluções existentes? Qual é o sistema operacional e o ambiente necessário ao desenvolvimento?

As informações coletadas com estas perguntas servirão de entrada para as restrições de projeto a serem incluídas nas especificações suplementares do documento MCU.

5. Formular uma Declaração do Problema

Reúna todo o time e, de posse de algum quadro de anotações, preencha o gabarito Visão para cada problema que for identificado:

O problema de <descreva o problema>
afeta <os interessados envolvidos pelo problema>.
O impacto é <qual é o impacto do problema>.
Uma solução de sucesso poderia <relacione alguns benefícios-chave para uma solução de sucesso>.

A finalidade desta descrição é ajudar a distinguir soluções/respostas de problemas/perguntas.

Exemplo:

O problema de : Questões mal resolvidas ou inadequadas para com o nosso serviço de atendimento ao cliente
afeta: nossos clientes, os responsáveis pelo suporte ao cliente e os técnicos de serviço.
O impacto é: a insatisfação do cliente, percepção de falta de qualidade, empregados infelizes e perda de receita.
Uma solução de sucesso poderia: prover um acesso em tempo real a um banco de dados de pesquisa de defeitos para os responsáveis pelo suporte ao cliente, bem como facilidades de fornecer os serviços técnicos de forma conveniente, para aqueles que precisarem de assistência.

6. Definir as Características do Sistema sob a Forma de Requisitos

Baseado nos benefícios listados nas declarações do problema, desenvolva uma lista de características que o cliente deseja para o sistema e descreva isto sob a forma de requisitos. Faça esta descrição de forma sucinta e inclua para cada requisito um ou mais atributos, para ajudar a definir seu estado geral e sua prioridade no projeto (para maiores detalhes sobre atributos, veja a atividade Gerenciar Dependências descrita no item 3.4.3.

7. Negociar Compromissos

A partir dos requisitos alocados e descritos no documento Visão, deve-se identificar as pessoas que ficarão responsáveis por cada um deles, verificando as habilidades de cada um, e negociar as estimativas iniciais de prazo para a conclusão de cada um deles.

8. Avaliar os Resultados

Faça uma revisão no documento Visão.

Para a atividade Revisar Requisitos de *Software* Alocados:

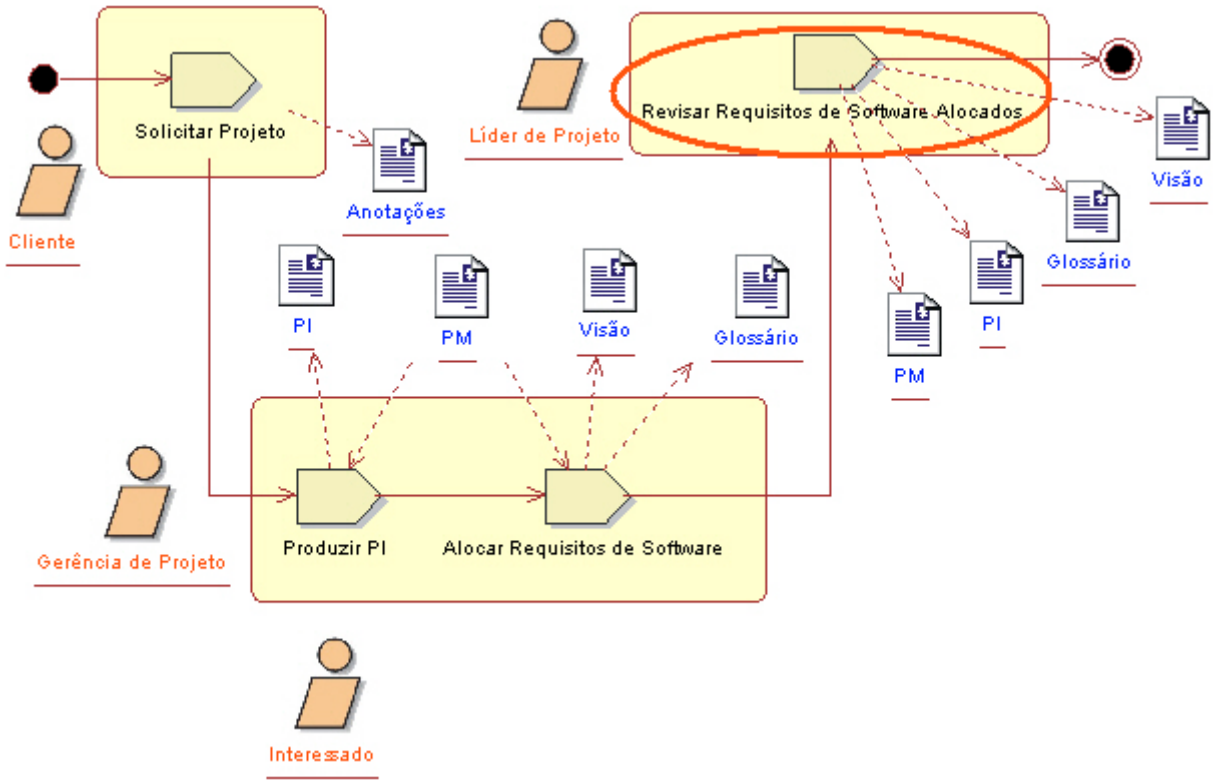

Figura 25 – Atividade: Revisar Requisitos de *Software* Alocados

Artefatos de Entrada: PI e PM (se existir)

Artefatos de Saída: Glossário e Visão

Papel: Gerência de Projeto

Finalidade:

Verificar se os requisitos alocados encontram-se claros, completos, consistentes, sem ambigüidades, factíveis e testáveis, antes que sejam incorporados para desenvolvimento. Verificar formalmente se os requisitos estão de acordo com a visão do sistema pelo Cliente.

As seguintes diretrizes devem ser levadas em consideração durante o processo de revisão:

- Deve-se procurar sempre fazer as revisões sob a forma de uma reunião. É vantajoso que cada um dos participantes da reunião tenham feito uma revisão individual nos documentos a serem revistos. Esta reunião deverá possuir no mínimo duas pessoas, sendo obrigatória a presença da Gerência de Projeto para solucionar os problemas encontrados e providenciar as alterações necessárias. Procure, sempre que possível, incluir o Cliente na reunião de revisão, ou alguém que o represente. Caso o Grupo de *Software* não participe desta reunião de revisão, este grupo deve ser comunicado dos requisitos alocados para conhecimento e tomada de opiniões.
- Verificar, de forma contínua, o que está sendo produzido, para certificar-se de que a qualidade do produto está sendo alcançada.

Passos:

1. **Fazer verificações**

   O processo de revisão deverá verificar os seguintes aspectos em cada requisito identificado:

   - Se é possível de ser implementado.
   - Se está apropriado ao contexto do projeto.
   - Se está descrito de forma clara e sem ambigüidades.
   - Se está declarado de forma adequada, de acordo com algum padrão da organização, podendo ser utilizado o especificado no documento PI.
   - Se possui consistência individual e conjunta com outros requisitos.
   - Se é possível de ser testado.

3.4.2 Fluxo de Trabalho Detalhado da Macroatividade Elaborar MCU

Para a atividade Encontrar Atores e Casos de Uso: [8]

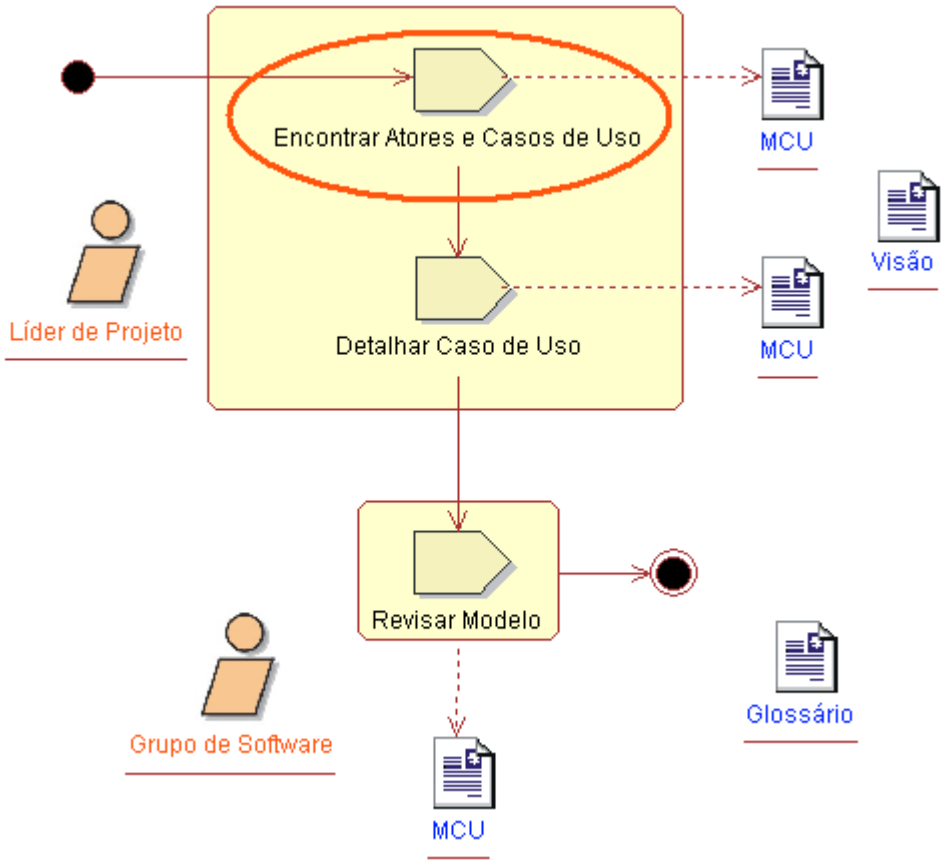

Figura 26 – Atividade: Encontrar Atores e Casos de Uso

Artefatos de Entrada: Visão, Glossário e PI

Artefato de Saída: MCU

Papel: Gerência de Projeto

Finalidade:

Modelar as funcionalidades do sistema, definir o que ele deverá manipular e o que deverá ser manipulado fora dele, definir o que e quem irá interagir com o sistema e criar diagramas de casos de uso.

[8] Esta atividade foi traduzida e adaptada da atividade "*Find Actors and Use Cases*" do RUP - 2002.05.00.25.

Passos:

1. **Detalhar os Requisitos do *Software***

   Certifique-se de que todos os requisitos encontram-se especificados, de tal maneira que estejam em condições de serem refinados no documento MCU. Na medida em que os requisitos não-funcionais forem sendo descobertos, deve-se acrescentá-los em local adequado indicado neste mesmo documento.

   Como estes requisitos serão rastreados ao longo do desenvolvimento, deve-se tentar manter as nomeações recomendadas neste documento, facilitando o rastreamento.

2. **Lista de Verificação**[9]

   Esta lista de verificação tem por finalidade fazer com que o profissional possa obter maiores detalhes de como capturar a maioria dos requisitos não-funcionais, requisitos estes que nem sempre poderão ser descritos por meio dos casos de uso.

   - *Funcionalidade*: O que é que o *software* se propõe a fazer? Isto deve incluir:
     - Verificações válidas das entradas.
     - Respostas gerais para as situações anormais incluindo: *overflow*, facilidades de comunicação, manipulação e recuperação de erro.
     - Relacionamentos das entradas com as saídas, incluindo seqüências de entrada/saída e fórmulas de conversão de entradas em saídas.
   - *Interface Externa*: Como o *software* interage com as pessoas, com o *hardware* do sistema, com outros dispositivos de *hardware* e com outros *softwares*?
   - *Desempenho*: Qual é a velocidade, a disponibilidade, os tempos de resposta e de recuperação das várias funcionalidades que o *software* deverá implementar?
   - *Requisitos Lógicos de Banco de Dados*: Foram especificados todos os registros lógicos para todas as informações que deverão ser colocadas no banco de dados?

[9] Esta lista de verificação foi traduzida e adaptada do "*Checkpoint - Supplementary Specification*" do RUP - 2002.05.00.25.

Isto pode incluir:

- Tipos das informações usadas pelas várias funcionalidades.
- Freqüência de uso.
- Capacidades de acesso.
- Entidades de dados e seus relacionamentos.
- Restrições de integridade.
- Requisitos de retenção de dados.

- *Concordância a Padrões*: Os requisitos foram derivados de algum padrão? Quais? Como que isto será acompanhado?
- *Atributos*: Quais são as considerações de confiabilidade, disponibilidade, portabilidade, corretude, manutenibilidade, segurança etc.?
- *Restrições de Projeto*: Foi requerido que o projeto seguisse um determinado padrão, uma linguagem de implementação, políticas para a integridade do banco de dados, limites de recursos, ambientes operacionais etc.?

3. Encontrar Atores

Encontrar atores é um dos primeiros passos na definição do uso do sistema. Cada tipo de fenômeno externo com o qual o sistema deverá interagir deve ser representado por um ator. Para facilitar a descoberta dos atores, faça as seguintes perguntas:

- Quais usuários ou grupos de usuários precisam da ajuda do sistema para realizarem suas tarefas?
- Quais usuários ou grupos de usuários são necessários para executarem as funções principais e mais óbvias do sistema?
- Quais usuários ou grupos de usuários são solicitados a executar as demais funções secundárias do sistema, tais como sua administração e manutenção?
- O sistema irá interagir com algum *hardware* ou *software* externo?

Qualquer pessoa, grupo de pessoas ou fenômenos que atendam a uma ou mais destas perguntas são consideradas fortes candidatas a serem atores do sistema.

Não é necessário que sejam capturados todos os atores de uma só vez. Na medida em que se descobrir novos casos de uso, estes irão contribuir para a descoberta de outros atores.

Nomeie e Faça uma Breve Descrição de Cada Ator Encontrado

Os atores devem ser nomeados de forma que seus papéis fiquem claramente definidos. Certifique-se de que existe pouco risco, de no futuro, o nome (papel) de um ator ser confundido com outro.

Faça uma breve descrição do ator, incluindo a sua área de responsabilidade, e para quê o ator precisa do sistema.

4. Encontrar Casos de Uso

Assim que os primeiros atores do sistema forem encontrados, o próximo passo é procurar pelos casos de uso do sistema. Os primeiros casos de uso poderão sofrer alterações na medida em que se entender melhor o comportamento do sistema e a maneira como os atores deverão interagir com eles. Deve-se perder um pouco de tempo até que se possa considerá-los estáveis. Em caso de dificuldades na identificação dos casos de uso, pode ser que os requisitos estejam deficientes ou que a análise esteja vaga, fazendo com que as funcionalidades do sistema não fiquem muito claras. Durante este trabalho de identificação dos casos de uso, deve-se estar preparado para adicionar, remover, combinar ou dividir casos de uso até que se chegue a uma versão final. Espera-se que se tenha um bom entendimento dos casos de uso após se proceder a uma descrição detalhada do comportamento de cada um, por meio dos fluxos de eventos.

A melhor forma de se descobrir casos de uso é avaliar o que cada ator deseja do sistema. Deve-se lembrar que o sistema existe apenas para seus usuários, e deve desta forma estar baseado em suas necessidades. Estas necessidades podem ser descobertas investigando-se os requisitos funcionais do sistema. Para cada ator, seja ele humano ou não, deve-se fazer para si mesmo as seguintes perguntas:

- Quais são as principais tarefas que ele deseja que o sistema realize?
- Ele precisa criar, guardar, mudar, remover ou ler dados do sistema?
- Ele precisa informar ao sistema sobre mudanças externas?
- Ele precisa ser informado sobre certas ocorrências no sistema?
- Ele precisa realizar operações de inicializações ou desligamentos?

As respostas a estas perguntas representam os casos de uso candidatos. Nem tudo poderá ser considerado como caso de uso candidato, podendo-se ter uma variante de um caso de uso (inclusão, extensão ou generalização). Estas variantes não devem ser utilizadas, caso não se tenha o domínio de seus entendimentos, procurando manter o diagrama de casos de uso o mais simples possível. Estas variantes poderão ser identificadas de forma mais clara durante a criação dos fluxos de eventos de cada caso de uso.

Um sistema pode possuir diversos modelos de casos de uso e todos eles podem estar corretos. A melhor forma de saber qual deles é o melhor é construí-los e optar pelo mais simples e mais claro.

Quando terminar de elaborar a primeira versão do modelo de casos de uso, deve-se verificar se ele atende a todos os requisitos funcionais que o sistema deverá implementar.

Nomeie e Faça uma Breve Descrição dos Casos de Uso Encontrados

Cada caso de uso deverá possuir um nome que indique aquilo que os atores desejam de cada um. Este nome deve possuir diversas palavras e formar uma expressão verbal de forma que possa ser claramente entendido. Dois casos de uso não podem possuir os mesmos nomes.

Cada caso de uso deve possuir uma breve descrição. Durante esta descrição, pode ser necessário atualizar o documento Glossário para definir novos termos.

Identifique Requisitos Adicionais

Durante a construção do modelo de casos de uso, devem ser identificados os requisitos que não poderão ser alocados a nenhum caso de uso por se tratarem de requisitos não-funcionais. Estes requisitos devem ser registrados em um item específico do documento MCU.

5. Construir Diagramas de Casos de Uso

Os atores e casos de uso descobertos devem ser identificados em um ou mais diagramas de casos de uso, onde serão mostrados também os relacionamentos entre atores e casos de uso e possíveis dependências entre casos de uso. Caso o sistema apresente muitos casos de uso, pode-se construir um diagrama principal e outros secundários.

6. **Rastrear Requisitos em Casos de Uso (mapeamento)**

   Deve-se preencher a tabela de Requisitos X Casos de Uso, dos casos de uso do sistema identificados no passo anterior. Esta tabela permitirá uma visualização rápida dos requisitos funcionais e não-funcionais (especificações suplementares) que se encontram atreladas aos casos de uso.

7. **Revisar**

   Deve-se fazer uma revisão dos passos realizados anteriormente, garantindo sua completude e corretude. Atenção especial deve ser dada na revisão da tabela de mapeamento do passo anterior. Todo os mapeamentos devem ser validados. Em seguida, deve-se identificar a existência de linhas vazias (requisitos não implementados) e colunas vazias (casos de uso desnecessários). Garante-se com isto que aquilo que o Cliente solicitou encontra-se devidamente mapeado, e que não foi acrescentado nada desnecessariamente.

8. **Avalie os Resultados**

   Verifique se o modelo de casos de uso construído nesta atividade encontra-se satisfatório.

   É importante que o Cliente aprove o modelo de casos de uso criado até o momento. Desta forma, deve-se envolver os usuários Interessados antes de finalizar esta atividade.

   Os Interessados devem determinar se:

   - foram identificados todos os casos de uso.
   - existem casos de uso desnecessários.
   - o comportamento de cada caso de uso está descrito de forma correta.

Para a atividade Detalhar Caso de Uso: [10]

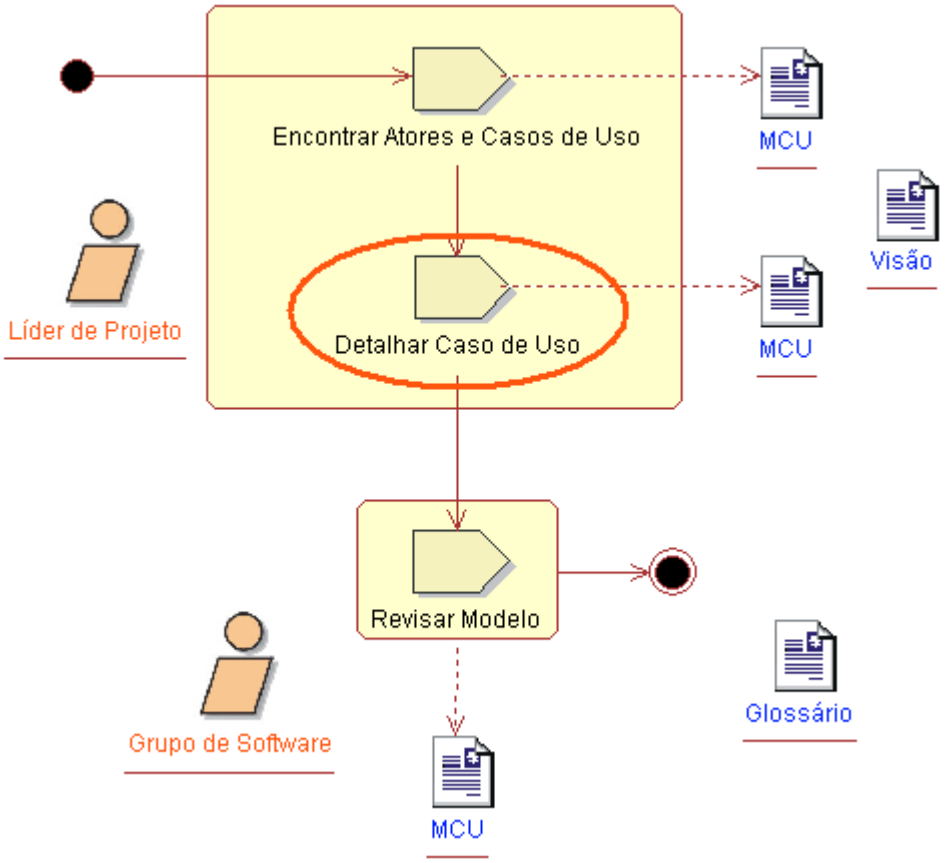

Figura 27 – Atividade: Detalhar Caso de Uso

Artefatos de Entrada: Visão, Glossário, PI e MCU

Artefato de Saída: MCU

Papel: Gerência de Projeto

Finalidade:

Detalhar os fluxos de eventos de um caso de uso de forma que o cliente e os usuários possam entendê-los.

[10] Esta atividade foi traduzida e adaptada da atividade *"Detail a Use Cases"* do RUP - 2002.05.00.25.

Passos:

1. **Detalhar o Fluxo de Eventos de um Caso de Uso**[11]

   Deve-se iniciar fazendo-se perguntas para o caso de uso. Analisam-se então as informações e transportam-se os resultados para o MCU, sob a forma de fluxos de eventos. Algumas perguntas a serem feitas são:

   - Como o caso de uso se inicia? O início de um caso de uso deve claramente descrever o evento que irá ativá-lo o caso de uso. Escreva por exemplo, "Este caso de uso se inicia quando ..... acontece" ou "Este caso de uso se inicia quando o ator ....".
   - Como o caso de uso termina? Deve-se declarar claramente tudo o que pode acontecer durante o fluxo para que o caso de uso seja finalizado. Pode-se escrever por exemplo, "Quando ... acontece, o caso de uso é finalizado" ou "Este caso de uso é finalizado quando .....".
   - Como o caso de uso interage com os atores? Para minimizar qualquer risco de má interpretação, deve-se dizer exatamente aquilo que irá existir dentro do sistema e o que irá existir fora dele. Deve-se estruturar a descrição com uma série de parágrafos, onde cada parágrafo irá expressar uma ação no formato: "Quando o ator ....., o sistema ....." As interações entre o caso de uso e o ator podem ser descritas por meio do envio de sinais entre eles, como por exemplo: "O caso de uso se inicia quando ele recebe o sinal de *start* do Operador".
   - Como ocorre o intercâmbio de informações com o ator? Caso seja conveniente, podem-se definir argumentos de sinais ou descrever da forma: "O caso de uso se inicia quando o Usuário se loga no sistema entrando com seu nome e sua senha".
   - Como o caso de uso repete algum comportamento? Pode-se fazer isto por meio da linguagem natural. Pode-se optar ainda pela construção de estruturas algorítmicas do tipo "Enquanto faça" "Repita enquanto" "Se então senão" e "loops", contanto que seja fácil de ser lida e mantida.
   - Existem situações opcionais no fluxo de eventos do caso de uso? Algumas vezes o ator apresenta uma série de opções. Uma forma de fazer esta descrição é:

[11] Para maiores detalhes sobre descrição de fluxos de eventos, consulte SOARES (2002).

"O ator escolhe uma das opções abaixo, uma ou mais vezes:

a) . . .

b) . . .

c) . . ."

- Como deve ser descrito o caso de uso de forma que o Cliente e o usuários possam entendê-lo? Uma boa prática é numerar as ações enviadas dos atores para o caso de uso e também as ações realizadas pelo caso de uso.

Deve-se concentrar em descrever o que o caso de uso faz e não a forma como os problemas deverão ser resolvidos. Isto deverá ser feito durante a etapa de identificação dos modelos de objetos.[12]

Caso se verifique que o fluxo de um determinado caso de uso está ficando complexo, ou se surgem partes que são independentes umas das outras, deve-se dividir o caso de uso em 2 ou mais.

Caso durante a descrição de um caso de uso surjam novos termos, deve-se incluí-los imediatamente no Glossário. Nenhum termo do Glossário deve ser alterado sem uma comunicação prévia ao time de desenvolvimento. Caso termos a serem alterados se refiram ao Cliente, certifique-se com ele de que a mudança fará sentido.

Descrição do Conteúdo de um Fluxo de Eventos

A descrição de um fluxo de eventos explora:

- Como e quando o caso de uso se inicia.

  Exemplo:

  "Este caso de uso se inicia quando a função "Administrar Pedido" é ativada pelo usuário".

[12] Estes modelos não são cobertos pelo LPA. Para maiores detalhes vide o artefato "*Design Model*" do RUP 2002.05.00.25

- Quando o caso de uso interage com os atores e que informações eles trocam.

  Exemplo:

  "Para criar um novo pedido, o usuário ativa a função "Novo" e inclui os dados no que diz respeito ao pedido: nome e quantidade. O usuário então ativa a função "OK" e um novo pedido será criado no sistema". Deve-se descrever explicitamente as informações que são trocadas entre os atores e o caso de uso. Caso contrário, provavelmente o Cliente e os Usuários Finais não irão entender a descrição do caso de uso.

- Como e quando o caso de uso utiliza os dados armazenados no sistema ou quando e como o caso de uso guarda os dados no sistema.

  Exemplo:

  "O usuário ativa a função "Modificar" para modificar um pedido existente e especifica o número do pedido (número inteiro). O sistema inicia um formulário de pedido com os dados do pedido. Estes dados são recuperados de um dispositivo de armazenamento secundário".

- Como e quando o caso de uso é finalizado.

  Exemplo:

  "O caso de uso finaliza quando a função "Sair" é ativada pelo solicitante do pedido.

Deve-se descrever também os acontecimentos excepcionais do fluxo de eventos. Um fluxo excepcional é um subfluxo do caso de uso que não atua segundo o seu comportamento normal ou básico. Se o caso de uso receber alguma informação inesperada, ele irá terminar.

Outros aspectos que devem ser levados em consideração ao descrever um caso de uso:

- Descrever o fluxo de eventos e não apenas a finalidade ou a funcionalidade do caso de uso.
- Descrever apenas os fluxos que pertençam ao caso de uso e não aquilo que irá acontecer em paralelo com a execução do mesmo.
- Não mencionar atores que não se comunicam com o caso de uso em descrição.
- Não fornecer muitos detalhes ao descrever as interações do caso de uso com qualquer ator.

- Se não existir uma ordem para os acontecimentos dos subfluxos, não tentar descrevê-los em uma determinada ordem.
- Utilizar os termos descritos no documento Glossário e incluir novos termos se surgirem.
- Utilizar um vocabulário simples. Evitar utilizar termos complexos quando termos mais simples puderem descrever da mesma forma.
- Escrever sentenças curtas e concisas.
- Evitar advérbios, tais como muito, mais, bastante e provavelmente.
- Utilizar a pontuação correta.
- Evitar sentenças compostas.

2. Descreva um Protocolo de Comunicação

Caso o ator associado a um caso de uso seja um outro sistema de *software* ou *hardware* externo, descreva um protocolo de comunicação. A descrição do caso de uso deve declarar se existe algum protocolo (talvez algum padronizado) a ser utilizado. Caso o protocolo seja novo, a sua descrição completa deverá ser criada durante a construção do modelo de objetos.

3. Descreva as Precondições do Caso de Uso

Uma precondição de um caso de uso especifica o estado que o sistema deverá possuir antes que o caso de uso possa iniciar.

Exemplo:

Para que um Caixa Eletrônico possa ser capaz de fornecer dinheiro, as seguintes precondições precisam ser satisfeitas:

- A rede de dados deve estar acessível.
- O Caixa Eletrônico precisa estar no estado "Pronto" para que possa aceitar transações.
- O Caixa Eletrônico deve ter algum valor em espécie internamente para que possa oferecer ao correntista.

- O Caixa Eletrônico deve possuir papel suficiente na impressora para poder emitir o recibo de, no mínimo, uma transação.

Todas estas precondições precisam ser atendidas para que o Caixa Eletrônico seja capaz de fornecer dinheiro.

Deve-se ter o cuidado com a descrição do estado do sistema, evitando-se descrever os detalhes de outras atividades acidentais que possam ocorrer antes que o caso de uso inicie sua execução.

Precondições não são usadas para criar uma seqüência de ações de um caso de uso. Nunca deverá existir uma situação onde se tenha que executar um caso de uso primeiro, depois o outro, de forma a se ter um fluxo de eventos significativo. Caso exista a necessidade de se fazer isto, pode significar que o modelo de casos de uso foi decomposto de forma excessiva. Deve-se corrigir o problema fazendo-se uma combinação de casos de uso que devem atuar em seqüência, em apenas um caso de uso. Caso isto leve a um caso de uso muito complexo, avalia-se a possibilidade de estruturar o modelo, criando subfluxos ou cenários secundários.

4. Descreva as Pós-condições do Caso de Uso

Uma pós-condição de um caso de uso lista todos os estados possíveis que o sistema pode assumir após a finalização do caso de uso. O sistema deverá então assumir um dos estados apresentados assim que o caso de uso finalizar. É comum também fazer uma declaração de ações que devem ser executadas ao final do caso de uso, sem levar em consideração o que possa ter ocorrido durante a execução do mesmo.

Exemplo:

Se o Caixa Eletrônico sempre apresenta a mensagem "Seja Bem-vindo" ao final de um caso de uso, isto poderia ser documentado na pós-condição do caso de uso.

Da mesma forma, se o Caixa Eletrônico sempre encerra a transação do cliente ao final do caso de uso, como por exemplo na transação "Sacar", independente do que possa ter ocorrido durante a transação, isto deveria ser registrado sob a forma de pós-condição do caso de uso.

Pós-condições são usadas para reduzir a complexidade e melhorar a confiabilidade do fluxo de eventos do caso de uso.

5. Avalie os Resultados

Revisar e discutir os casos de uso com os Interessados, de forma que eles possam entendê-los de maneira clara e concordem com suas descrições.

A descrição do caso de uso somente estará completa quando descrever tudo o que o caso de uso realiza, implementa. Uma revisão mais profunda será feita na execução da atividade Revisar Modelo descrita na atividade seguinte.

Para a atividade Revisar Modelo:[13]

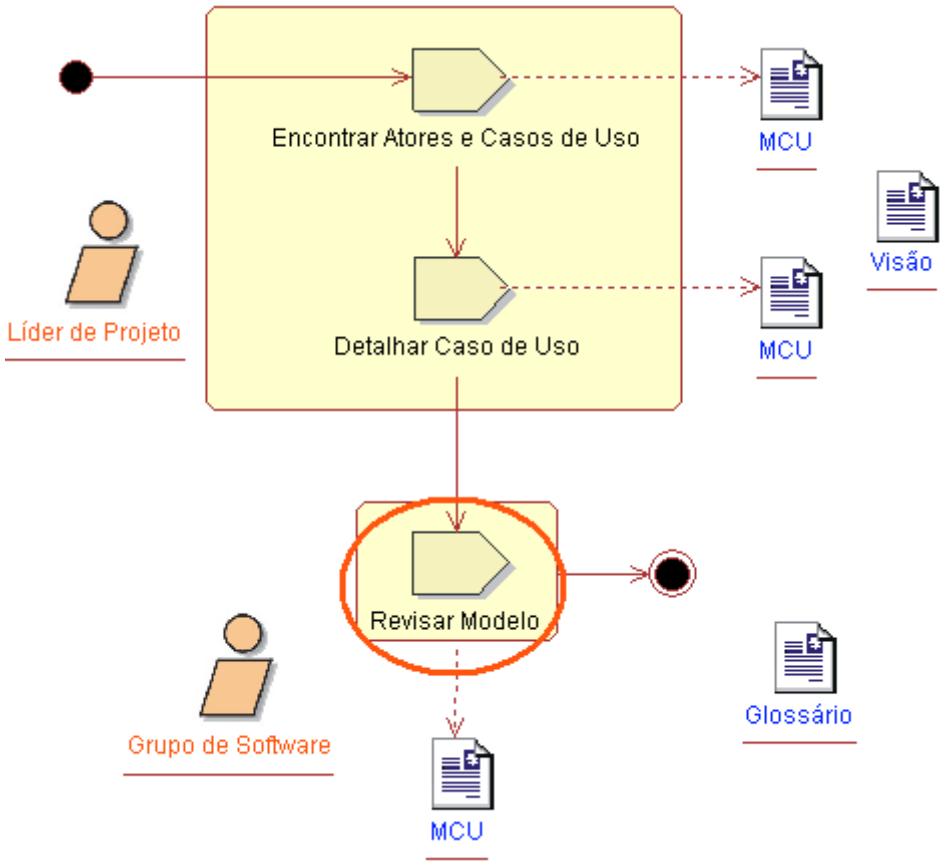

Figura 28 – Atividade: Revisar Modelo

Artefatos de Entrada: Visão, Glossário, PI e MCU

Artefato de Saída: MCU

Papel: Grupo de *Software*

[13] Esta atividade foi traduzida e adaptada da atividade *"Review Requirements"* do RUP - 2002.05.00.25.

Finalidade:

Verificar formalmente se os requisitos estão sendo corretamente implementados e de acordo com a necessidade do Cliente.

As seguintes diretrizes são úteis para a realização de uma revisão nos requisitos:

- Conduzir sempre as revisões sob a forma de reuniões, embora os participantes necessitem fazer uma revisão antecipada individualmente.
- Verificar de forma continuada o que será produzido, para garantir que a qualidade do produto seja a mais alta possível. Pode-se utilizar reuniões de revisões informais no dia-a-dia do trabalho.

Dependendo do tipo de revisão a ser realizada, deve-se procurar alocar alguns papéis estratégicos dependendo da fase de desenvolvimento. Uma boa regra é alocar o Cliente, ou alguém representando os interesses do cliente, nas revisões de início de desenvolvimento, e sempre ter alguém da fase anterior e alguém da fase posterior ao ciclo de vida utilizado.

É importante balancear a quantidade de participantes de uma reunião de revisão, para que ela possa ser gerenciada e ao mesmo tempo produtiva.

Passos:

1. **Conduzir Reuniões de Revisão**

   Normalmente pode-se dividir a revisão dentro das seguintes reuniões:

   - Uma revisão do Pedido de Mudança, devido ao fato de que ele pode vir a impactar o conjunto de requisitos existentes.
   - Uma revisão de todo o Modelo de Casos de Uso.

   Deve-se estar preparado para realizar novas revisões sempre que for produzida uma nova versão do Modelo de Casos de Uso.

   Recomenda-se que seja organizada uma revisão do Modelo de Casos de Uso para cada iteração nas fases de Concepção e de Elaboração, onde terá que ser feito o acompanhamento do desenvolvimento ainda no início dos trabalhos, obtendo maior garantia de que o modelo esteja correto. Deve-se ter em mente que, ao final da fase de Elaboração, 80% do Modelo de Casos de Uso deverão estar concluídos. Deve-se tentar organizar também uma revisão para cada iteração das fases de Construção e Transição,

caso o Modelo de Casos de Uso venha a ser refinado. Esta revisão deve se concentrar apenas na parte do modelo que estiver fazendo parte da iteração.

2. Retrabalhar o MCU

O retrabalho consiste na correção dos erros identificados pelo Grupo de *Software* durante as revisões. Este retrabalho deverá ser feito pelo autor do documento MCU, no caso o Líder de Projeto. O Líder deverá então realizar as devidas correções e gerar um novo documento com as devidas atualizações. Cabe ao Grupo de *Software* decidir se após o retrabalho deverá haver uma nova revisão ou não. Esta decisão pode ser tomada baseada na quantidade de erros encontrados e também na sua severidade. Por meio de delegação do Líder de Projeto, o Grupo de *Software* pode proceder às correções do MCU.

3.4.3 Fluxo de Trabalho Detalhado da Macroatividade Gerenciar e Controlar Mudanças nos Requisitos[14]

Para a atividade Produzir PM:[15]

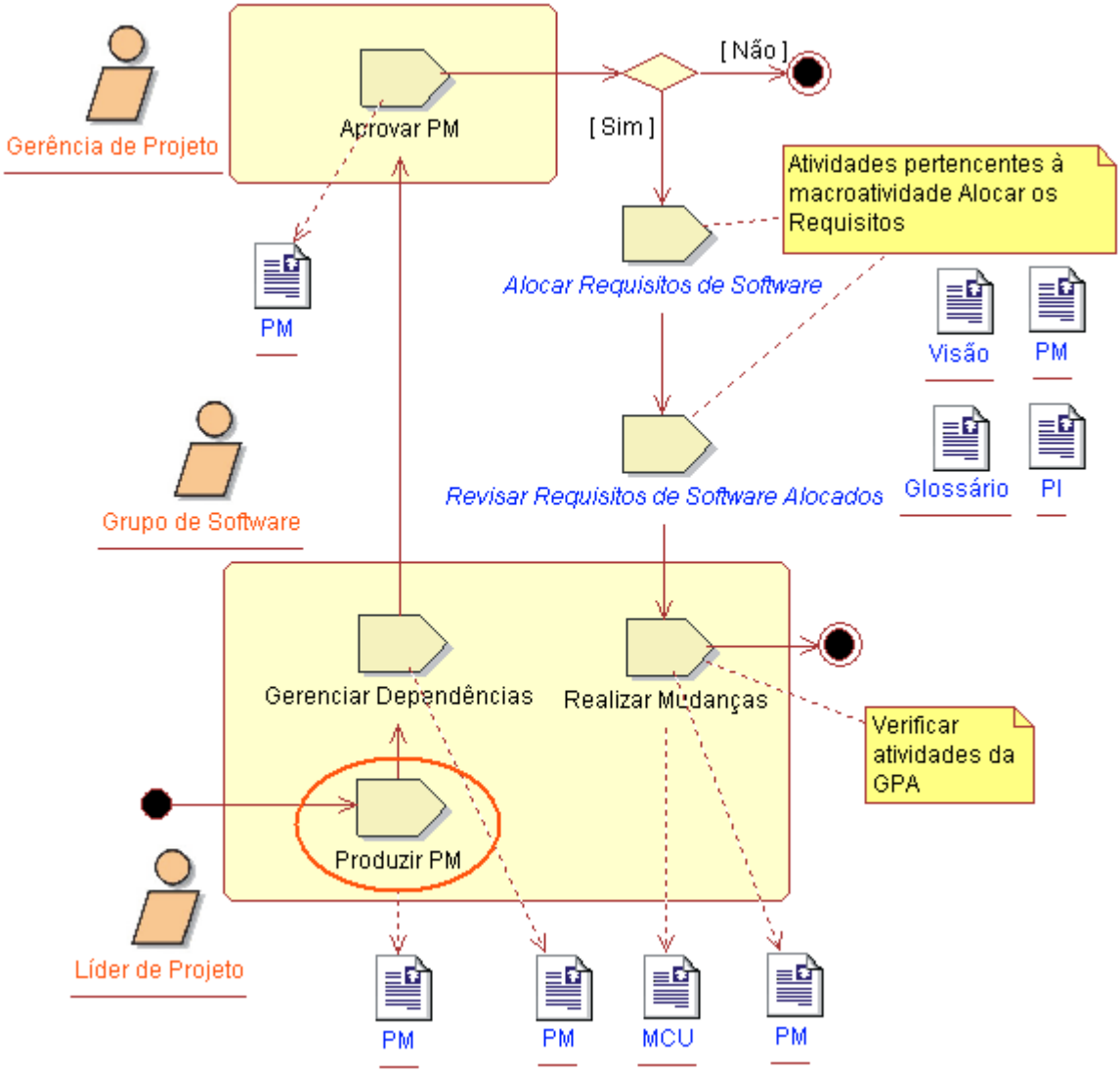

Figura 29 – Atividade: Produzir PM

Artefatos de Entrada: inexistente

Artefato de Saída: PM

Papel: Líder de Projeto

---

[14] As atividades “Alocar Requisitos de *Software*” e “Revisar Requisitos de *Software* Alocados” já foram descritas na macroatividade “Alocar Requisitos” (item 3.4.1)

[15] Esta atividade foi traduzida e adaptada da atividade *"Submit Change Request"* do RUP - 2002.05.00.25.

Finalidade:

Registrar um pedido de mudança. Um pedido de mudança pode incluir solicitações de novos requisitos para o sistema, melhorias, correções, mudanças de requisitos e assim por diante. Qualquer pessoa pode solicitar um pedido de mudança durante todo o ciclo de vida do sistema.

Passos:

1. **Completar o Formulário de PM - Pedido de Mudança**

   O formulário PM é utilizado para se acompanhar a solicitação de mudança, podendo envolver novas características (requisitos), melhorias, defeitos, mudanças de requisitos etc. Todo o histórico de uma mudança deverá ser mantido no PM, incluindo os estados da mudança durante sua evolução, com datas e motivos para a mudança. Estas informações devem ser utilizadas para revisões e finalização da mudança.

2. **Submeter o PM**

   Uma vez preenchido, deverá ser submetido a aprovação. Antes disso, deve-se verificar o impacto da mudança. Isto será contemplado na atividade Gerenciar Dependências, descrita na atividade seguinte. Uma vez aprovado, deverá ser realizada a atividade Realizar Mudanças. Qualquer pessoa pode solicitar um pedido de mudança. O PM deve ser mantido sob controle de forma manual ou de forma automática para rastreamento.

   Os possíveis estados que um PM pode ter são:

   - Submetido -------- Preenchido.
   - Duplicado -------- Existência de um pedido igual ou similar.
   - Prorrogado ------- Não é possível a realização da mudança solicitada no momento.
   - Rejeitado -------- Solicitação inválida ou impacto elevado e não aceito pelo cliente.
   - Mais Informações - Os dados não são suficientes para se proceder à análise.
   - Aberto ---------- Análise de impacto iniciada.
   - Alocado ---------- Início das mudanças.
   - Resolvido -------- Mudanças concluídas.
   - Falha de Teste --- Ocorrência de falha durante o teste.
   - Verificado ------- Mudanças concluídas, testadas e aprovadas.

Para a Atividade Gerenciar Dependências:[16]

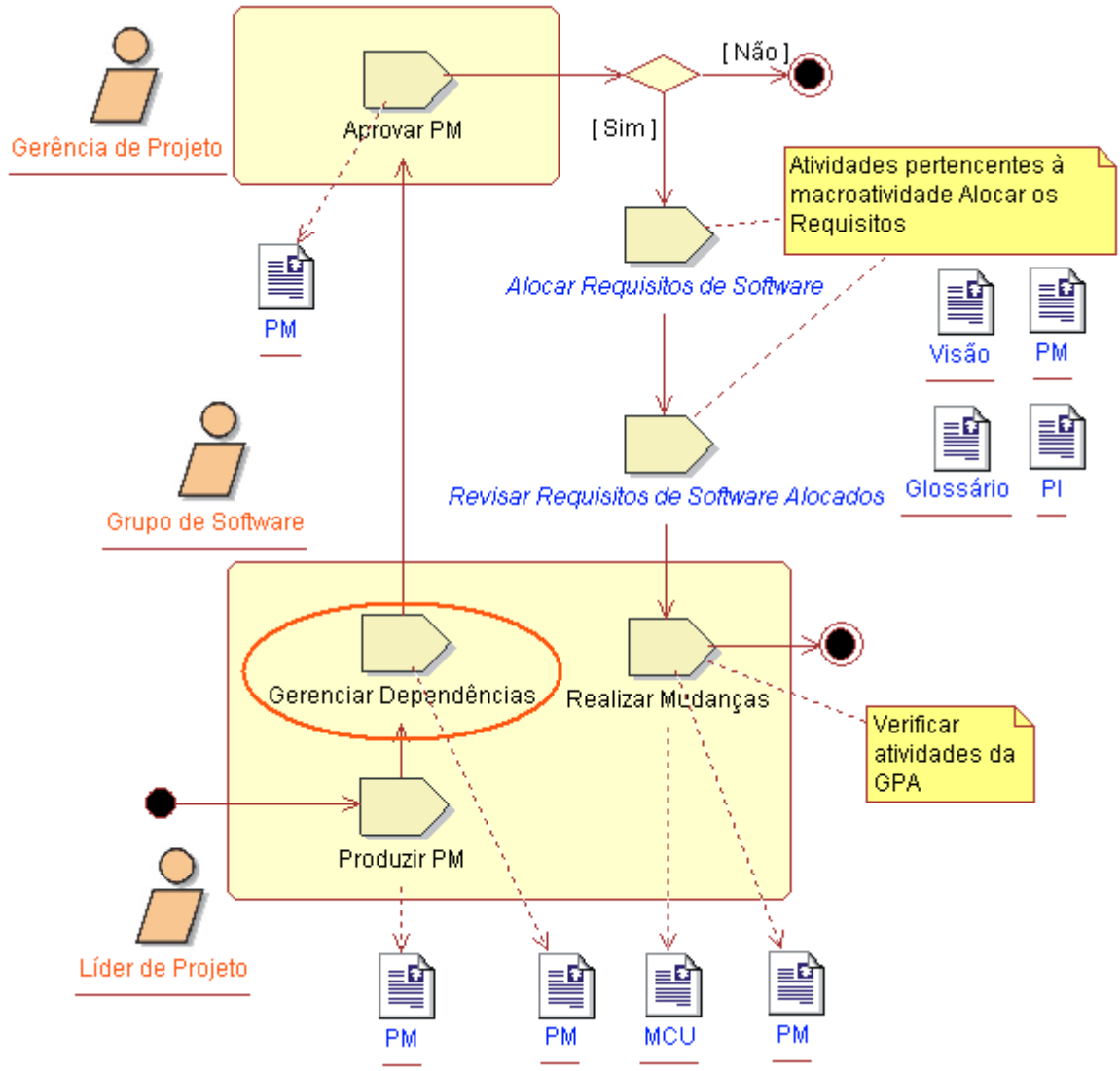

Figura 30 – Atividade: Gerenciar Dependências

Artefatos de Entrada: PI, Visão, Glossário, MCU, PM

Artefato de Saída: PM

Papel: Líder de Projeto

[16] Esta atividade foi traduzida e adaptada da atividade *"Manage Dependencies"* do RUP - 2002.05.00.25.

Finalidade:

Usar atributos e rastreabilidade dos requisitos do projeto para ajudar no controle do escopo do projeto e na gerência de mudança de requisitos.

Passos:

1. **Pontuar os Atributos dos Requisitos**

   Os atributos mais importantes que devem ser acompanhados para cada requisito são:

   - Benefício (do ponto de vista dos Interessados).
   - Esforço (para implementação).
   - Risco (para o desenvolvimento do esforço).
   - Estabilidade (probabilidade de não sofrer mudanças pelo cliente).
   - Impacto na Arquitetura (mudanças significativas na arquitetura).

   O Benefício e a Estabilidade devem ser levantados pelo Líder de Projeto e em conjunto com os Interessados. O Esforço e o Risco devem ser levantados pelo Líder de Projeto em consulta com a Gerência de Projeto. O Impacto na Arquitetura deve ser levantado pela Gerência de Projeto.

   Requisitos instáveis com alto risco, alto esforço ou alto benefício devem ser analisados com mais detalhes. Requisitos de baixo benefício, alto esforço, risco ou instabilidade devem ser fortes candidatos a serem rejeitados.

   A tabela abaixo apresenta um exemplo de um conjunto de requisitos obtidos do documento Visão, de uma aplicação de Gestão de Requisitos, mostrando os atributos associados a cada um deles. O atributo Benefício é uma opinião do Cliente e o Esforço é uma opinião do Grupo de *Software*.

Requisitos	Benefício	Esforço	Risco	Impacto na Arq.	Estabilidade
R1: Critérios de salvamento, recuperação, classificação e filtro	Médio Alto	Baixo	Baixo	Baixo	Alto
R2: Capacidade de visualizar a exclusão de um requisito da janela	Médio	Médio Alto	Médio	Baixo	Médio
R3: Suporte ao acesso a tipos de dados concorrentes	Médio	Médio	Médio Baixo	Baixo	Médio
R4: Mostrar os atributos de um requisito a partir de um documento texto	Médio	Médio	Médio	Baixo	Médio Alto
R5: *Wizard* para a criação de um novo projeto	Médio Alto	Alto	Médio Alto	Alto	Médio
R6: Criação rápida de um requisito	Médio Alto	Médio Baixo	Médio Baixo	Baixo	Alto
R7: Permitir auto-salvamento do projeto em arquivo	Médio	Médio Baixo	Médio	Baixo	Médio
R8: Permitir mudança de um ou mais atributos de um conjunto de requisitos selecionados	Médio	Médio Alto	Médio	Baixo	Médio
R9: Habilidade de colocar estruturas de um projeto para facilitar a criação de novos projetos a partir de outros mais antigos	Alto	Médio	Médio	Baixo	Baixo
R10: Permitir impressão com alto desempenho dos requisitos selecionados e seus atributos	Médio Baixo	Médio Alto	Médio	Baixo	Médio Alto

Baseado no conteúdo desta tabela, pode-se determinar que apenas parte dela será implementada na primeira iteração (por exemplo, 2/3). Esta análise deve ser feita também com base nos recursos existentes para a implementação destes requisitos. Deve-se verificar quais requisitos exercitam completamente a arquitetura desejada e implementá-los primeiro (por exemplo, o requisito 5). Todavia, este requisito possui Média Estabilidade. Desta forma, deve-se trabalhar mais com os Interessados para elevar este nível para Alto, o mais rápido possível.

O requisito 10 possui Benefício Médio Baixo e requer Esforço Médio Alto, sendo desta forma um forte candidato a ser rejeitado.

Para fazer com que as entregas ao Cliente sejam feitas dentro do prazo esperado, deve-se tentar evitar requisitos de Esforço Alto, especialmente se combinados com Instabilidade. Assim, seria bom que os requisitos 2, 8 e 9 fossem excluídos.

2. **Estabelecer e Verificar Rastreabilidade**

Deve-se verificar, neste momento, qual será o impacto destes requisitos no mapeamento dos casos de uso, tentando identificar quais os casos de uso que terão que ser alterados e em que extensão.

3. **Controlar as Mudanças de Requisitos**

Algumas diretrizes para que isto possa ser feito são:

Reavaliar os Atributos dos Requisitos e sua Rastreabilidade.

Mesmo que um requisito não tenha mudado, os atributos e a rastreabilidade associados a ele podem ter mudado. Desta forma, isto precisa ser verificado.

Controlar Mudança de Forma Hierárquica

Uma mudança de um requisito pode ter um efeito "cascata", podendo vir a afetar outros requisitos relacionados, bem como o projeto ou outros artefatos gerados até aquele momento do desenvolvimento. Para gerenciar este efeito, devem-se mudar os requisitos de cima para baixo (*top down*). Revise o impacto no documento Visão, depois no Modelo de Casos de Uso e por último no material de suporte ao usuário final (por exemplo, manual do usuário, tutoriais, guia de manutenção, sistema de *help online* etc). Para gerenciar o impacto da mudança dos requisitos no esforço de teste, deve-se procurar revisar as informações contidas nos procedimentos de testes e nos mapeamentos dos testes dos casos de uso.

Para a Atividade Aprovar PM:

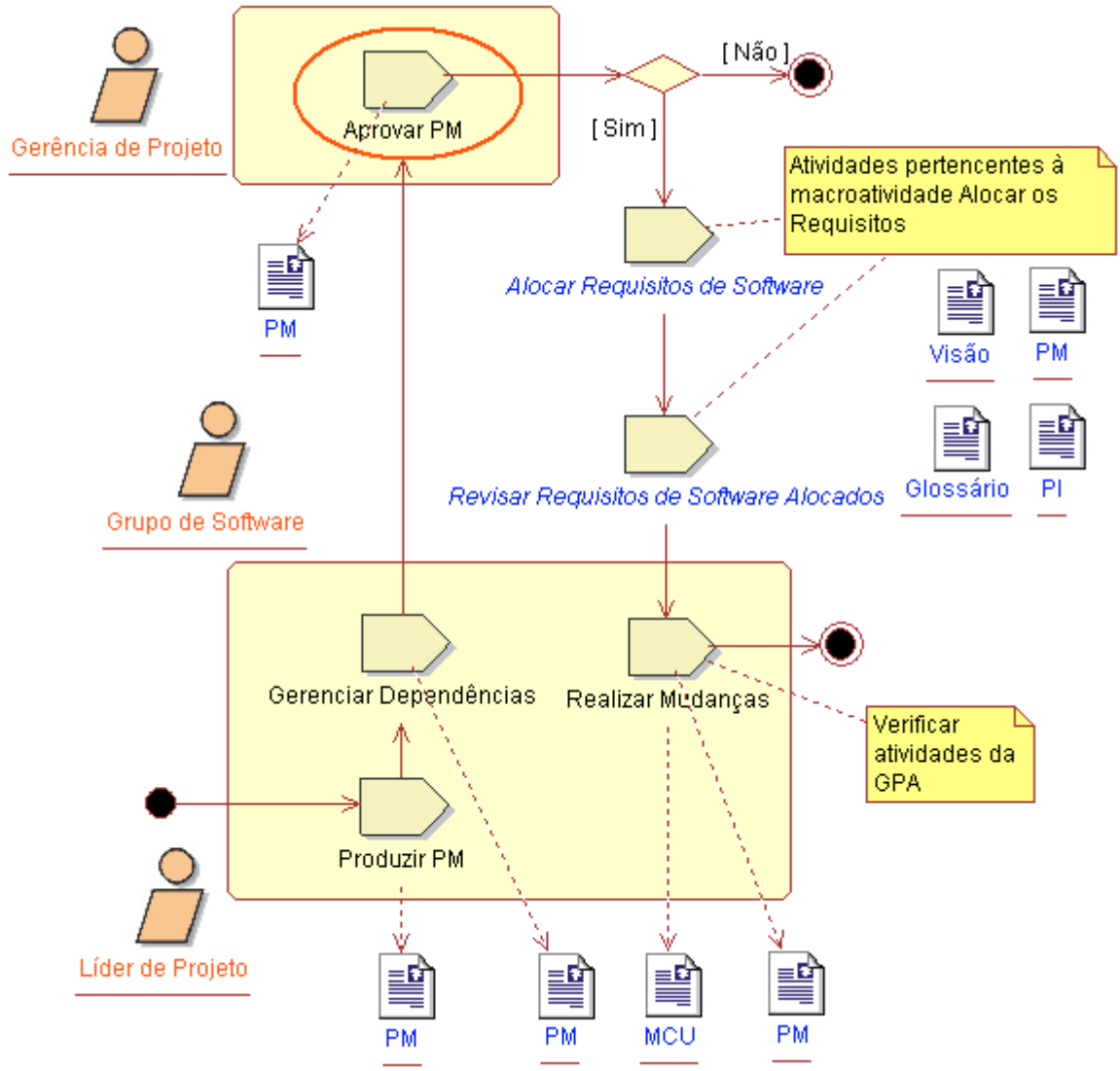

Figura 31 – Atividade: Aprovar PM

Artefatos de Entrada: PI, Visão, Glossário, MCU, PM

Artefato de Saída: PM

Papel: Gerência de Projeto

Finalidade:

Avaliar as informações passadas pelo Líder de Projeto com relação à mudança solicitada, decidindo pela sua implementação ou não.

Passos:

1. **Avaliar Impacto da Mudança Solicitada**

   A Gerência de Projeto, de posse das informações fornecidas pelo Líder de Projeto no decorrer da atividade anterior Gerenciar Dependências, deverá analisar todas as informações, para decidir se a mudança será aprovada. As informações a serem avaliadas devem estar direcionadas para:

   - Os benefícios da mudança para o Cliente.
   - O esforço de implementação e conseqüentemente do custo.
   - Os riscos associados a esta mudança.
   - A estabilidade da mudança pelo Cliente, ou seja, qual deve ser a probalidade de o Cliente solicitar nova mudança referente a esta.
   - O Impacto na arquitetura atual do sistema.

2. **Decidir pela Implementação**

   De posse das informações do passo anterior, a Gerência de Projeto deve decidir pela implementação ou não da mudança solicitada. Caso esta mudança envolva alterações nos custos e prazos estabelecidos entre ambas as partes no início do desenvolvimento, a Gerência de Projeto deve obter aprovação formal do Cliente para que possa iniciar a implementação.

Para a Atividade Realizar Mudanças:

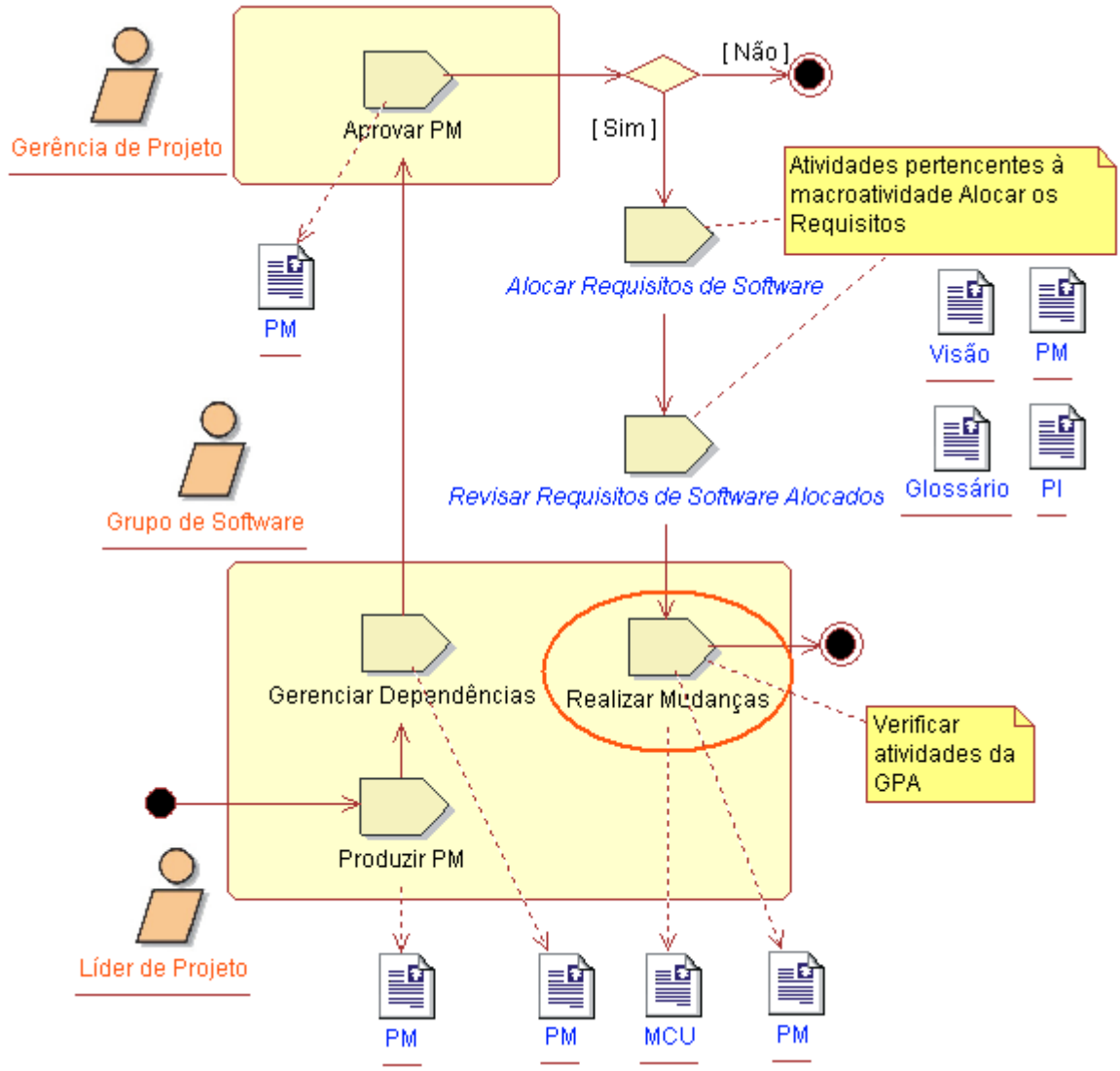

Figura 32 – Atividade: Realizar Mudanças

Artefatos de Entrada: Visão, Glossário, PI, PM

Artefatos de Saída: PM, MCU

Papel: Líder de Projeto

Finalidade:

Realizar alterações no Modelo de Casos de Uso para comportar as mudanças a serem implementadas e desenvolver uma *release* para esta mudança.

Passos:

1. **Atualizar MCU**

   Proceder às mudanças necessárias no MCU. Para maiores detalhes, vide as atividades: Encontrar Atores e Casos de Uso e Detalhar Caso de Uso, descritas no item 3.4.2.

2. **Implementar as Mudanças**

   Implementar as mudanças após as alterações do MCU. Isto provavelmente irá envolver a geração de uma *release* a ser apresentada para o Cliente. Os detalhes desta etapa deverão estar previstos no PDA e no Cronograma de desenvolvimento. Para maiores detalhes, vide atividades da GPA, descritas no item 3.5.

3. **Avaliar os Resultados**

   Revisar e discutir as mudanças realizadas, procurando envolver tanto o Grupo de *Software* quanto outros Interessados, de forma que eles possam entendê-las de forma clara e concordem com suas descrições.

3.4.4 Fluxo de Trabalho Detalhado da Macroatividade Definir PGR – Política da Gestão de Requisitos

Para a atividade Definir PGR:

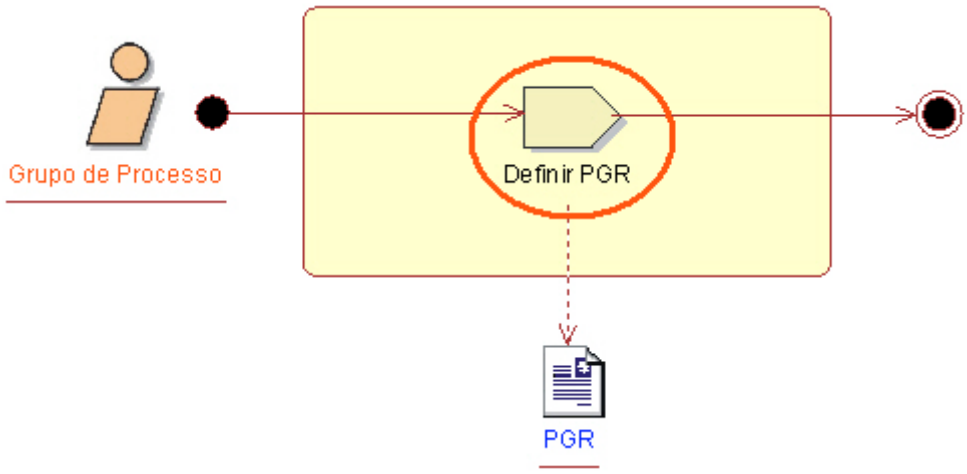

Figura 33 – Atividade: Definir PGR

Artefatos de Entrada: Inexistente

Artefato de Saída: PGR

Papel: Grupo de Processo

Finalidade:

Definir de forma clara, objetiva e escrita as políticas a serem seguidas durante a Gestão de Requisitos do LPA.

Passos:

1. **Definir e Documentar as Políticas da Gestão de Requisitos**

   As políticas a serem definidas podem ser utilizadas por diversos projetos ou sofrer ligeiras ou muitas alterações para cada tipo de projeto a ser desenvolvido.

   Estas políticas devem ser definidas pelo Grupo de Processo e em conjunto com as demais pessoas da organização.

   As políticas devem ser especificadas de forma documentada e devem tipicamente, descrever que:

   - Deve existir um responsável pela alocação dos requisitos ao *software*.
   - Os requisitos alocados ao *software* devem ser documentados.
   - Os requisitos alocados ao *software* devem ser revisados pelo Líder de Projeto em conjunto com a Gerência de Projeto, podendo envolver ainda o Grupo de *Software*.
   - Os requisitos devem ser utilizados como base para o planejamento do projeto, bem como os planos produzidos. Os produtos de trabalho e as atividades devem ser mantidos consistentes com as mudanças que possam vir a ter nos requisitos alocados ao sistema.

2. **Avaliar os Resultados**

   O documento PGR, que define as políticas da gestão de requisitos, deverá ser revisado e aprovado pela Gerência de projeto.

3.5 Definição das Atividades da Gestão de Planejamento e Acompanhamento

Os itens seguintes apresentam as atividades de cada fluxo de trabalho detalhado das macroatividades da gestão de planejamento e acompanhamento.

3.5.1 Fluxo de Trabalho Detalhado da Macroatividade Elaborar o PDA

Para a atividade Iniciar Projeto:

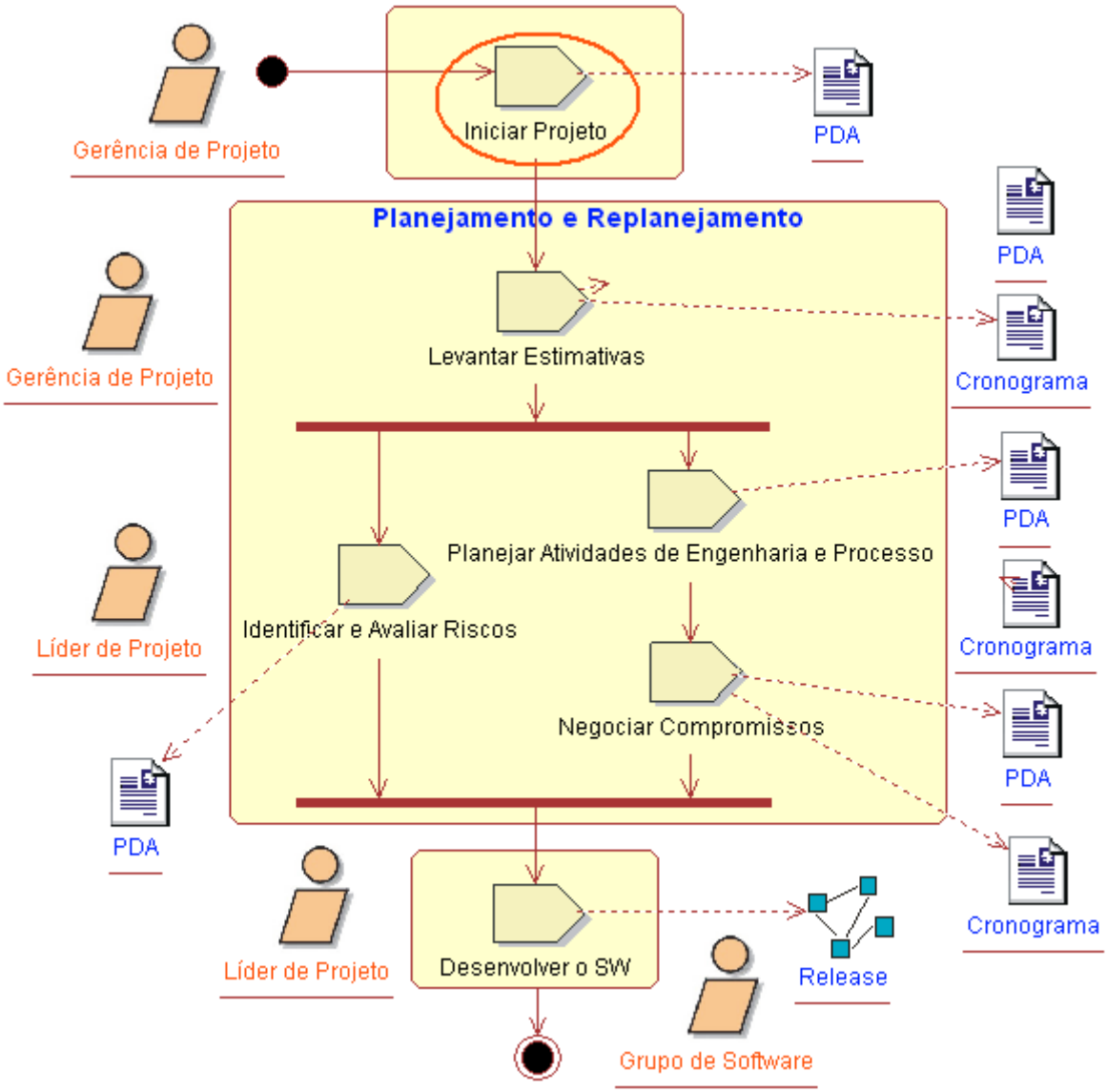

Figura 34 – Atividade: Iniciar Projeto

Artefatos de Entrada: Glossário, PI e Visão

Artefato de Saída: PDA

Papel: Grupo de Processo

Finalidade:

Alocar oficialmente a equipe de trabalho para o projeto e definir os produtos de trabalho principais e atividades. Os requisitos definidos servirão de base para o desenvolvimento do PDA e para todo o desenvolvimento do sistema.

Esta atividade deve ser iniciada após a liberação do PI. Ela deve definir quem assumirá os papéis de Gerência de Projeto, Líder de Projeto e os demais integrantes do time de desenvolvimento, ou seja, o Grupo de *Software*. Além disto, deve-se determinar quais serão os critérios de aceitação do sistema para que o projeto possa ser considerado finalizado com sucesso. A partir da estimativa de tempo para o projeto, deve-se definir os marcos principais de cada fase e as principais atividades.

Passos:

1. **Treinamento**

   A Gerência de Projeto e outras pessoas envolvidas deverão receber treinamento com relação aos itens previstos nesta atividade, para que o planejamento do sistema possa iniciar e ser documentado utilizando-se o PDA e Cronograma. Este treinamento poderá ser realizado sob a forma de um curso contratado, tutoriais on-line, repasse de conhecimento de colegas de trabalho, participação em *workshops*, seminários e conferências, ou ainda uma combinação destas formas.

2. **Definir uma Gerência para o Projeto**

   Alguém da organização de desenvolvimento de *software* deve assumir o papel da Gerência de Projeto. Esta nomeação deverá vir de alguém que venha a estar acima da hierarquia interna desta organização ou pelos próprios integrantes desta organização. Esta pessoa deverá possuir não apenas habilidades técnicas em diversos projetos, mas também habilidades em gerenciar pessoas. Esta pessoa pode fazer parte do Grupo de *Software*.

3. **Definir o Time de Desenvolvimento**

   Uma vez definida a Gerência de Projeto, esta deverá ser capaz de identificar as pessoas que farão parte do time de desenvolvimento, time este denominado de Grupo de *Software*, que fará parte do respectivo projeto. Além disto, um dos integrantes do Grupo de *Software* deverá assumir o papel de Líder de Projeto.

4. Definir Marcos e Atividades Principais

Utilizando-se o gabarito do Cronograma do projeto, a Gerência de Projeto deverá definir, em conjunto com o Líder de Projeto, os marcos principais a serem alcançados durante o desenvolvimento, tais como início e final de projeto, liberação de *releases*, finais de fase, desenvolvimento de casos de uso e outras que considerarem relevantes. Vale ressaltar que o Ciclo de Vida a ser seguido no projeto deverá ser o Iterativo e Incremental, composto de minicascatas em cada iteração. A Figura 35 ilustra este ciclo de vida, sendo uma adaptação do ciclo de vida Iterativo e Incremental do RUP – 2002.05.00.25.

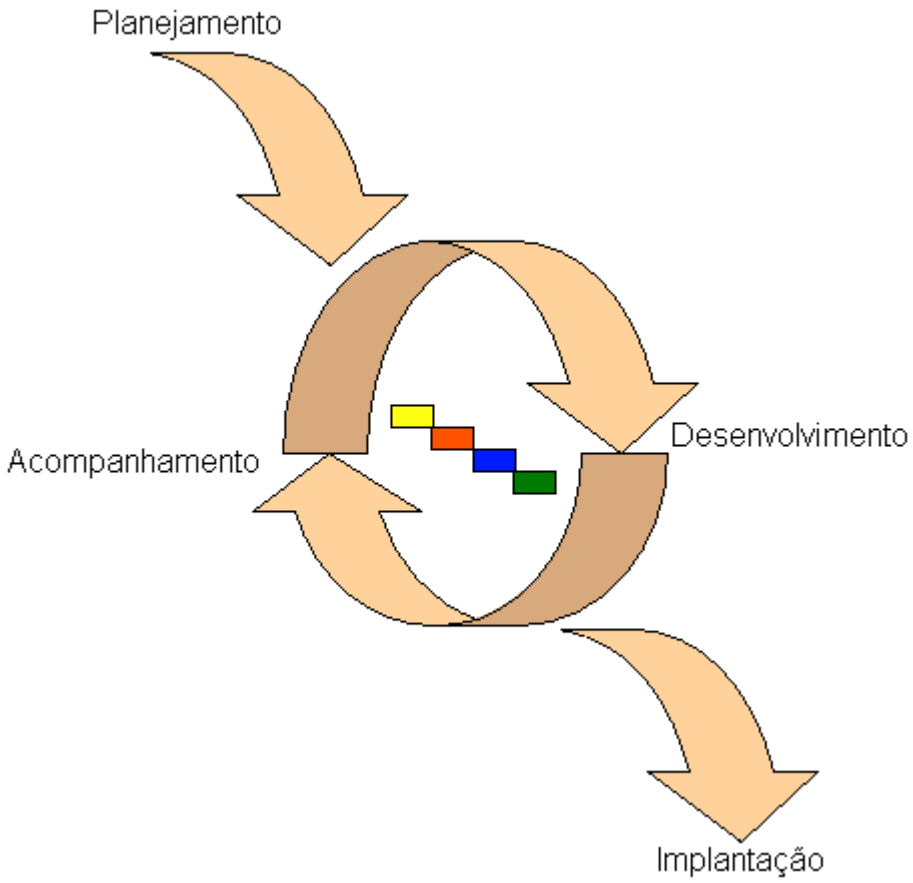

Figura 35 - Ciclo de Vida Iterativo e Incremental do LPA

O círculo central corresponde a uma iteração que deverá seguir um modelo minicascata. Antes de iniciar uma iteração, deve ser feito um planejamento das atividades, e ao término de diversas iterações deverá ser gerada uma última *release* a ser implantada no ambiente de trabalho do Cliente para uso pelos Usuários Finais.

O PDA deve ser criado já no início do projeto (fase de concepção) e deve definir, dentre outras informações, algumas estimativas iniciais, tais como os tempos estimados de duração de cada fase do desenvolvimento.

Para planejar as fases de um projeto para o primeiro ciclo evolutivo, deve-se possuir algum conhecimento sobre os marcos do projeto, tendo-se por base as seguintes informações:

- Experiência com projetos similares tanto com relação à sua natureza quanto ao domínio técnico.
- Grau de novidades.
- Restrições específicas de ambiente, tais como tempo de resposta, distribuição e segurança.
- Maturidade da organização.

Utilizando estimativas baseadas em sua própria experiência com outros projetos de natureza similar, pode-se criar um orçamento inicial do projeto, fazendo uma distribuição do esforço e custo necessários ao longo de cada fase do mesmo.

5. Identificar os Recursos Computacionais Críticos

Os recursos computacionais críticos se referem àqueles que podem exceder às disponibilidades normais existentes pela organização para o desenvolvimento do projeto, podendo provocar grande risco durante o desenvolvimento. Alguns exemplos incluem capacidade de memória de um ou mais computadores, capacidade em disco, capacidade de um canal de comunicação de dados etc. Os recursos críticos computacionais não podem ser confundidos com outras necessidades naturais que o time de desenvolvimento irá necessitar para o desenvolvimento do projeto, tais como compra de computadores, mais licenças de ferramentas de *software*, contratação de pessoas, aquisição de um banco de dados etc. Uma vez identificados quais serão estes recursos, os mesmos deverão ser acompanhados durante o desenvolvimento do projeto. Não necessariamente todos os recursos críticos computacionais terão que ser identificados de imediato, podendo ocorrer novas descobertas ao longo dos trabalhos.

6. Definir e Aprovar os Critérios de Aceitação para o Projeto

O último passo desta atividade é definir alguns critérios objetivos que serão usados pelo Cliente para determinar quando o produto final entregue e implantado será considerado aceito. Estes critérios deverão ser definidos pelo Cliente e em conjunto com o Líder de Projeto ou a Gerência de Projeto, podendo incluir:

- Entrega de todos os artefatos (documentos e *releases*) previstos.
- Lista de todos os participantes requeridos para o teste de aceitação.
- Local dos testes.
- Completar com sucesso os testes nos artefatos avaliados.
- Completar com sucesso o treinamento ao Cliente.
- Completar com sucesso a instalação no ambiente do Cliente.
- Medidas que identifiquem aquilo que excedeu às especificações originais do projeto.

7. Revisar o PDA

A Gerência de Projeto deve passar os itens preenchidos do PDA e do Cronograma para uma revisão pelo Líder de Projeto. Ao término desta revisão, pode-se considerar que ambos encontram-se aprovados.

Para a atividade Levantar Estimativas:[17]

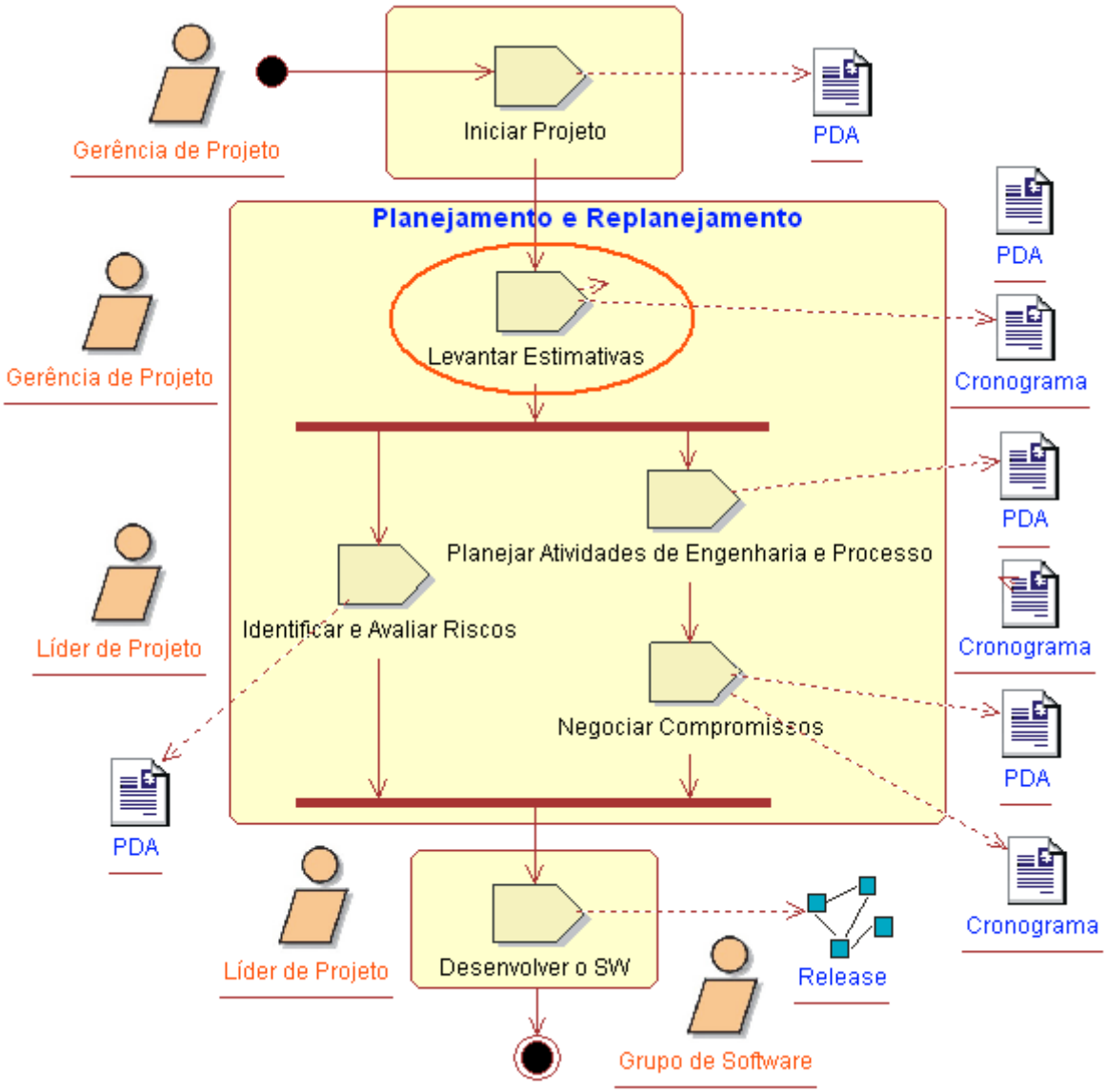

Figura 36 – Atividade: Levantar Estimativas

Artefato de Entrada: MCU

Artefato de Saída: PDA

Papel: Gerência de Projeto

[17] Esta atividade foi traduzida e adaptada da atividade *"Plan Phases and Iterations"* do RUP - 2002.05.00.25.

Finalidade:

Fazer diversas estimativas de projeto e acompanhá-las durante o desenvolvimento. Este acompanhamento servirá para a melhoria do próprio processo de estimativas para que, em projetos posteriores, estas mesmas estimativas possam ser identificadas com maior precisão.

- Estimar o escopo do projeto, esforço e custo para seu desenvolvimento.
- Desenvolver um esboço do plano de projeto, enfatizando os marcos principais e as entregas principais a serem feitas para o Cliente durante o ciclo de vida.
- Definir um conjunto de iterações dentro de cada fase do projeto e identificar os objetivos e orçamentos para o projeto.
- Desenvolver um Cronograma e orçamento para o projeto.
- Identificar os recursos necessários para o projeto.
- Definir as atividades necessárias para completar o projeto.

Passos:

1. **Treinamento**

   Antes que as estimativas do projeto possam ser realizadas, as pessoas envolvidas deverão receber treinamento adequado. Este treinamento poderá ser realizado sob a forma de um curso contratado, tutoriais on-line, repasse de conhecimento de colegas de trabalho, participação em *workshops*, seminários e conferências, ou ainda uma combinação destas formas. Este treinamento deverá envolver, principalmente, estimativas de tamanho de produtos de trabalho, esforço nas atividades de desenvolvimento do projeto, custos e prazos.

2. **Estimativas de Projeto**

   Durante a fase de concepção, deve-se preparar estimativas para o trabalho proposto no projeto. As estimativas devem seguir o processo descrito nos 4 passos seguintes:

   1. Estimar o tamanho do produto.
   2. Estimar o esforço e custo total do projeto.
   3. Aplicar as restrições e prioridades (por exemplo, tamanho da equipe de trabalho, datas de entregas e orçamento).
   4. Selecionar um Cronograma adequado, esforço e estimativa de custo.

Estimativa de Tamanho de Produto

Esta é a estimativa de entrada do processo de estimativas. Se não for possível estimar a magnitude do trabalho a ser feito, qualquer Cronograma de projeto criado estará muito distante da realizada. Existem duas abordagens para se estimar o tamanho do produto de *software* e que pode ser usado na fase inicial do projeto. São elas: Dimensionando pela Analogia e Dimensionando pela Análise.

Dimensionamento pela Analogia

Quando se estima o escopo do produto usando a abordagem do Dimensionamento pela Analogia, compara-se o novo produto que será desenvolvido com outros produtos (de mesmo tamanho) desenvolvidos anteriormente. Deve-se analisar as diversas características dos produtos que estão sendo comparados, tais como o número de casos de uso, número de atores, complexidade e tamanho das bases de dados e número de similaridades de programas *on-line* e *batch*.

Pela comparação destas características, pode-se estimar o tamanho relativo do novo produto comparado com outros antigos, devendo então usar os tamanhos conhecidos dos produtos antigos para calcular o novo tamanho estimado. Deve-se ter em mente que é importante comparar produtos de complexidades similares. O desenvolvimento, usando abordagens similares, mas com variações tanto nestas similaridades quanto no nível de detalhamento das descrições de casos de uso, pode invalidar as comparações realizadas.

Dimensionando pela Análise

Após a fase de concepção, é bem provável que se tenha reunido informações suficientes com respeito ao novo produto, para que possam ser utilizadas técnicas analíticas na estimativa de tamanho do mesmo. Estas técnicas residem na descrição funcional do produto de *software* que será disponibilizado e na aplicação de regras padronizadas de contagem para determinar uma medida de tamanho para estas descrições. Provavelmente a técnica mais conhecida é a de Contagem de Pontos de Função, embora existam outras técnicas, tais como Pontos Característicos (uma modificação da técnica de Pontos de Função para sistemas de tempo real) e Pontos de Objetos Previstos (uma medida para sistemas baseados em orientação a objetos sob uma análise de complexidade de classes e de hierarquias). Um outro método que pode ser utilizado é o de estimativas de tamanho baseadas em Casos de Uso. Para este último método, deve-se ter em mente que se deve

calibrar as expressões de acordo com o estilo de modelagem de casos de uso da organização, devido ao fato de que os casos de uso variam muito em termos de nível de abstração e da maneira como eles são descritos. Uma vez calibrado, é importante manter o estilo selecionado para as escritas dos casos de uso, caso contrário as estimativas de tamanho poderão se tornar incorretas.

Estimativa do Esforço e Custo Total do Projeto

O esforço total da equipe e do cronograma para um projeto pode ser calculado a partir de estimativas de tamanho obtidas de modelos científicos. Os 2 modelos mais utilizados atualmente são: o COCOMO (COnstructive COst MOdel), desenvolvido por Barry Boehm, e a metodologia Putnam, de Larry Putnam. Ambos os modelos foram validados em confronto com dados da indústria.

Além da estimativa de tamanho, uma outra entrada é a medida da produtividade do time de desenvolvimento. Este valor determina o esforço geral do projeto. O Cronograma total do projeto está relacionado de forma não-linear com o esforço total. Infelizmente os modelos existentes são matematicamente complexos, de forma que é melhor fazer uso de um *software* para ajudar nos cálculos.

Aplicando as Restrições e Prioridades

Todo o projeto está sujeito a restrições (por exemplo, deve ser entregue numa certa data, ou o custo não pode exceder a um determinado valor) ou em termos de prioridade (por exemplo, a necessidade de se ter o produto tão logo quanto possível, ou de determinadas funcionalidades do produto o mais rápido possível). Dado um tamanho fixo do produto, as restrições e prioridades serão influenciadas de acordo com o tamanho do time de desenvolvimento. Vale salientar que o relacionamento entre tamanho e cronograma não é linear, de forma que é necessário o uso de modelos científicos para gerar um número de cenários baseados nas variações de tamanhos dos times. *Softwares* que automatizam as estimativas são muito úteis para esta finalidade.

Seleção de um Cronograma Adequado, Esforço e Estimativa de Custo

De posse dos dados obtidos, deve-se determinar a quantidade de pessoas necessárias para se atender a um determinado prazo e qual o custo envolvido.

3. Definir as Metas dos Marcos

Cada marco está focado em uma entrega específica, onde cada uma irá prover um ponto de transição bem definido para a próxima fase.

Fase	Marco	Finalidade
Concepção	Objetivo do Ciclo de Vida	Comprometimento com os recursos do projeto
Elaboração	Arquitetura do Ciclo de Vida	Estabelecer a arquitetura do produto
Construção	Capacidade Operacional Inicial	Desenvolvimento completo do produto
Transição	*Release* de Produto	Produto entregue de forma satisfatória

Cada marco representa uma barreira, a qual o projeto deve ultrapassar. Além disto, cada marco do projeto é o momento de decidir pela sua continuidade ou não (*go-no-go*).

4. Definir Número, Tamanho e Objetivos das Iterações dentro das Fases

Uma vez que o comprimento das fases esteja determinado, o número de iterações e seus comprimentos também precisarão ser determinados.

Cada iteração produz a entrega de um executável do produto (*release*), que será utilizada para avaliar seu progresso e sua qualidade. Devido ao fato de que cada iteração possui um foco diferente, a funcionalidade e a completude das *releases* de cada iteração irão variar. As metas de cada iteração devem ser específicas o suficiente para poderem ser avaliadas. Nas iterações recentes, as metas normalmente são expressas em termos de **redução de riscos**. Nas iterações posteriores as metas são expressas em medidas de funcionalidades completas e de qualidades.

5. Refinar o Escopo e as Datas do Marco

Ao final da fase de concepção, as demais fases podem ser planejadas de forma mais precisa, levando-se em consideração os seguintes aspectos:

- Número de casos de uso identificados.
- Complexidade dos casos de uso já estudados.
- Riscos identificados, técnicos e de negócios.
- Pontos de função, métricas de casos de uso ou qualquer outra medida de tamanho.
- Resultados de protótipos.

O PDA, que era muito superficial e rudimentar, é então atualizado durante a fase de elaboração. Ele servirá de base para a construção do restante do plano de projeto.

6. Determine os Recursos Requeridos para o Projeto

Baseado nas estimativas de esforço e no Cronograma de desenvolvimento derivados destas estimativas, pode-se agora definir os recursos requeridos para lidar com o projeto. Para cada fase e iteração, identifique os papéis necessários e quantos em cada iteração.

7. Desenvolver um Plano de Finalização

Descrever uma série de atividades que irão orientar como o projeto pode ser finalizado de forma ordenada.

Estas atividades podem ser:

1. Conferir se todas as entregas previstas para o Cliente foram realizadas.
2. Concluir com sucesso o teste de aceitação do sistema.
3. Colher e arquivar um registro de aceitação do produto final feito pelo Cliente.
4. Avaliar as estimativas e medidas realizadas durante o acompanhamento e produzir um relatório gerencial de resultados das análises feitas, incluindo os motivos das discrepâncias identificadas e as lições aprendidas.
5. Manter todos os dados estimados e medidos em uma base de dados para utilização em próximos projetos.
6. Arquivar em CDROM todos os produtos de trabalho produzidos, tanto *software* quanto não-*software*.
7. Distribuir a equipe para outras atividades em outros sistemas.

8. Revisar o PDA

A Gerência de Projeto deve passar os itens preenchidos do PDA para uma revisão pelo Líder de Projeto. Ao término desta revisão, pode-se considerar que este artefato esteja aprovado.

Estas atividades devem estar previstas no Cronograma de desenvolvimento.

Para a atividade Identificar e Avaliar Riscos:[18]

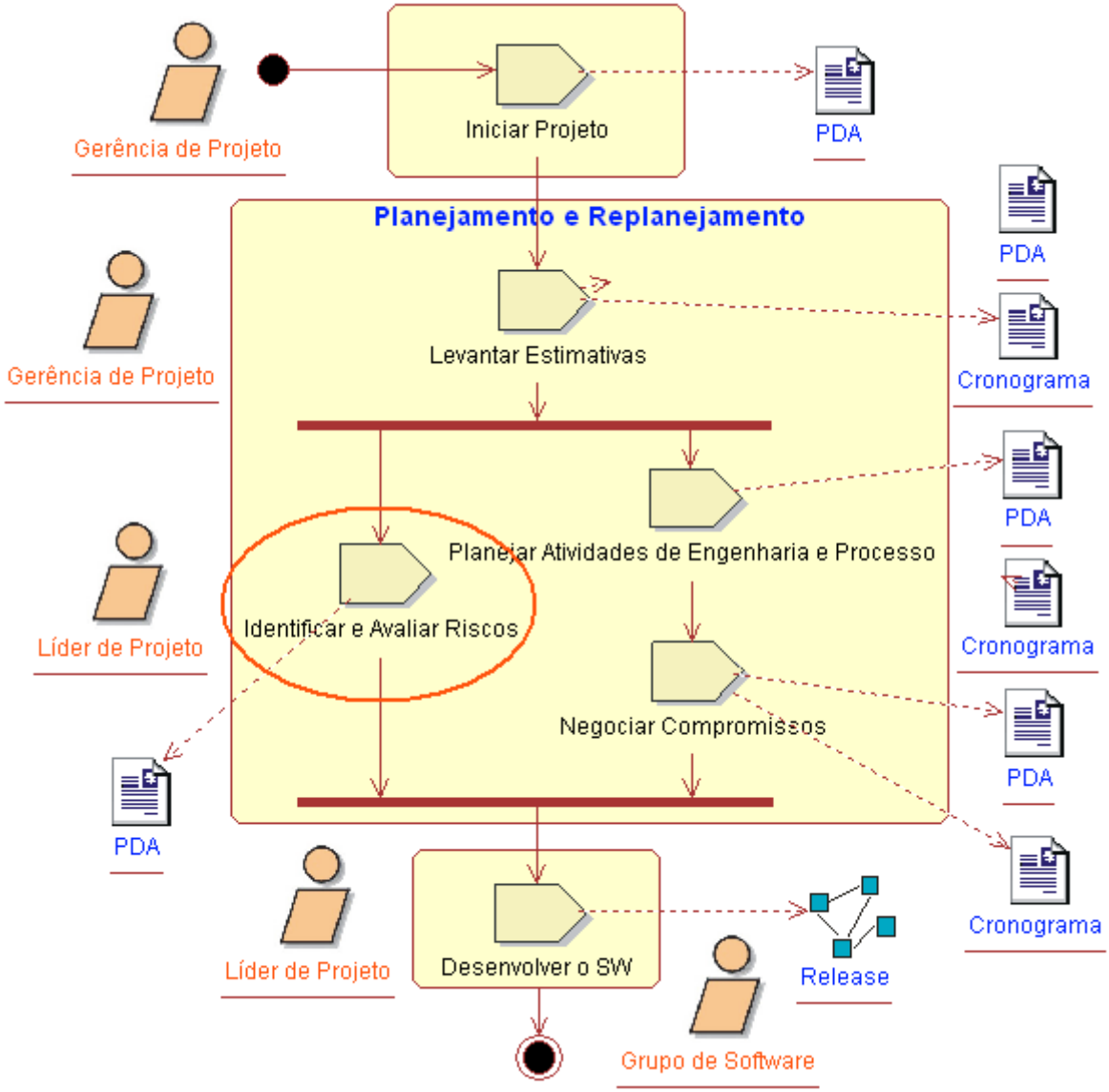

Figura 37 – Atividade: Identificar e Avaliar Riscos

Artefatos de Entrada: MCU e Cronograma

Artefato de Saída: PDA

Papel: Líder de Projeto

Finalidade:

Identificar, analisar e priorizar os riscos identificados no projeto. Atualizar a relação de riscos.

[18] Esta atividade foi traduzida e adaptada da atividade *"Identify and Assess Risks"* do RUP - 2002.05.00.25.

Finalidade:

Identificar, analisar e priorizar os riscos identificados no projeto. Atualizar a relação de riscos.

Passos:

1. **Identificar os Riscos**

   Em conjunto com o Grupo de *Software* e possivelmente com o Cliente, deve-se identificar, por meio de uma sessão de *brainstorming,* quais os riscos (diretos – sobre os quais se tem algum controle, ou indiretos – sobre os quais não se tem controle) que podem surgir durante o desenvolvimento, comprometendo o sucesso do projeto. Deve-se tentar identificar as situações que podem fazer com que o projeto saia do controle. Estes riscos devem ser acompanhados periodicamente em cada iteração, verificando se foram reduzidos ou não, e se surgiram outros novos.

   Os riscos podem estar relacionados com a organização, com recursos financeiros, com as pessoas envolvidas no projeto, tanto da parte do Cliente quanto do desenvolvimento, com os negócios do Cliente, com o escopo do projeto, as tecnologias envolvidas, dependências externas e Cronograma. Em todos os riscos identificados, devem ser analisadas as questões da probabilidade de comprometer as entregas propostas dentro do prazo previsto, nos limites de orçamento e com qualidade.

   Para facilitar a identificação dos riscos, deve-se levar para a sessão de *brainstorming* uma relação de riscos que a literatura aborda ou que tenham acontecido em outros projetos. Isto irá facilitar a condução inicial da sessão.

2. **Analisar e Priorizar os Riscos**

   Verifique na relação de riscos identificados se alguns podem ser agrupados, tais como riscos que representam sintomas de riscos maiores, o que irá contribuir para a redução da quantidade de riscos finais.

   Deve-se fazer em seguida uma pontuação para cada risco, identificando qual o seu valor e a probabilidade (em termos percentuais) de que possa vir a ocorrer. A multiplicação destes dois valores dará uma idéia mais clara do grau de exposição do projeto com relação a cada um deles.

   Faça em seguida uma análise do impacto que cada risco poderá provocar no projeto, caso eles venham a ocorrer. Esta análise deverá ficar em torno de desvios do Cronograma, de

esforço ou custos envolvidos. Concluída a análise de impacto, deve-se descrever um plano para redução ou eliminação de cada risco e um plano de contingência caso o risco venha a acontecer (plano B).

Uma vez identificados os valores de exposição do projeto com relação aos riscos, deve-se fazer uma classificação em ordem decrescente de riscos e discriminar de alguma forma os 10 riscos principais (*top 10 risks*).

Finalizado este levantamento, deve-se fazer com que todos os integrantes do desenvolvimento tenham conhecimento destes riscos.

É natural que a quantidade de riscos aumente no decorrer do projeto até meados da fase de elaboração, devido a poucas informações que se tinha do projeto. Na medida em que a fase de elaboração for avançando, deverá ocorrer um decréscimo gradual destes riscos.

3. **Identificar Planos para Redução dos Riscos**

   Deve-se identificar ações que possam ser tomadas para que os riscos sejam reduzidos. Para isto, deve-se coletar outras informações referentes a cada risco para que as incertezas existentes ao redor do risco possam ser eliminadas. Muitas destas ações podem ser feitas com a geração de protótipos durante a fase de concepção.

4. **Descrever Estratégias de Contingência**

   Caso algum risco venha a acontecer, deve-se seguir uma estratégia de contingência para cada um deles (plano B). A materialização dos riscos acontece mais em riscos do tipo indiretos, uma vez que para os riscos diretos os planos de redução de riscos podem ser aplicados sem maiores problemas.

5. **Acompanhe os Riscos**

   Os riscos identificados devem ser acompanhados de forma periódica em conjunto com todo o time de desenvolvimento, e também ao final de cada iteração ou de uma fase de desenvolvimento. Tipicamente, deve-se verificar se ocorreu alguma mudança nos riscos, se os planos de redução dos riscos estão sendo considerados e se os riscos estão realmente declinando ou se foram eliminados. Caso venham a surgir novos riscos, deve-se recalcular e conferir os valores de todos eles.

Para a atividade Planejar Atividades de Engenharia e Processo:

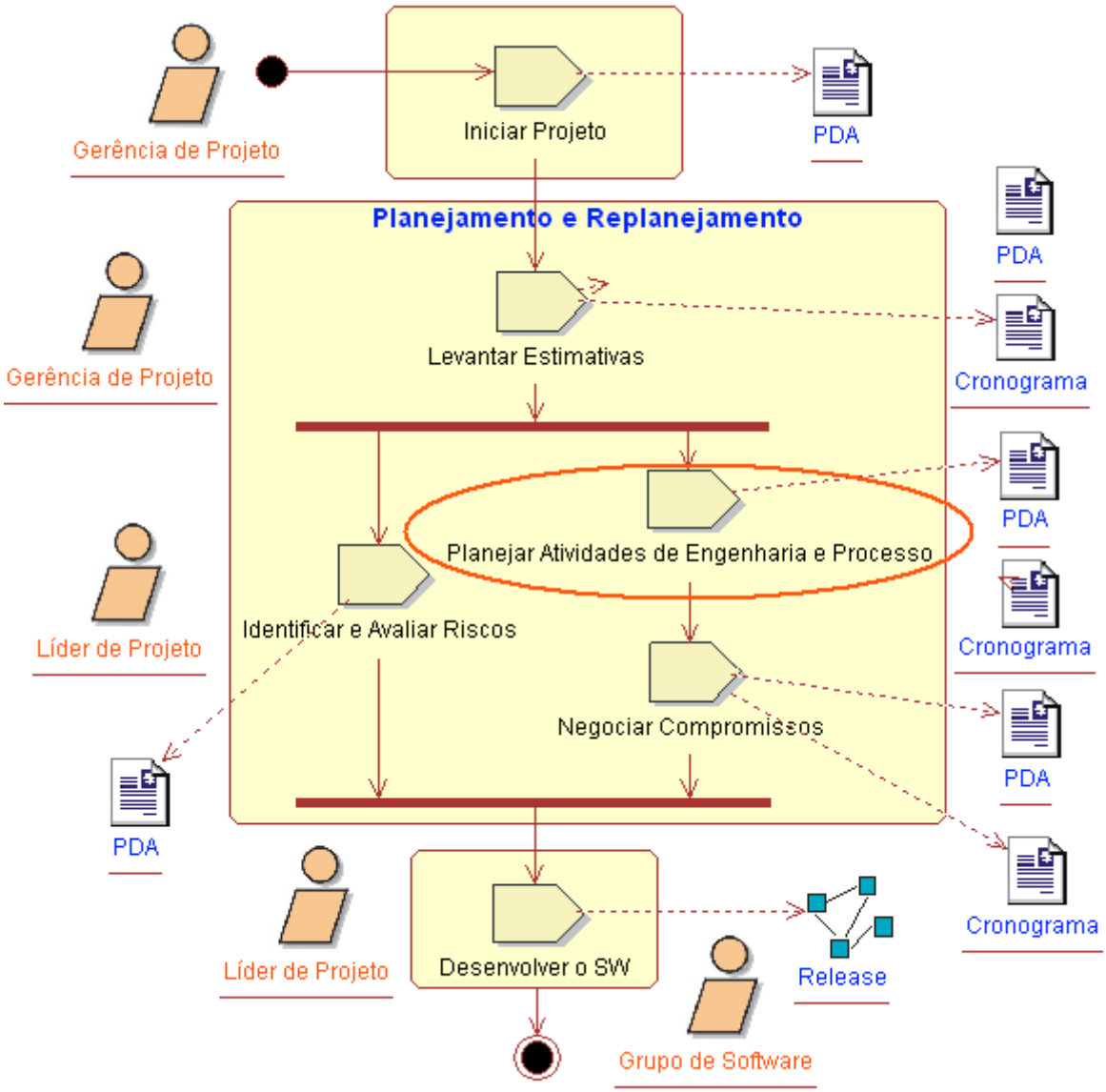

Figura 38 – Atividade: Planejar Atividades de Engenharia e Processo

Artefatos de Entrada: MCU, PDA e Cronograma

Artefatos de Saída: PDA e Cronograma

Papel: Líder de Projeto

Finalidade:

Definir as atividades a serem realizadas durante o desenvolvimento do sistema em um Cronograma.

Passos:

1. **Treinamento**

   O Líder de Projeto e outras pessoas envolvidas deverão receber treinamento com relação aos itens previstos nesta atividade, para que as atividades, tanto relacionadas com o processo quanto as de engenharia de *software*, possam ser planejadas de forma gradativa no PDA e Cronograma. Este treinamento poderá ser realizado sob a forma de um curso contratado, tutoriais on-line, repasse de conhecimento de colegas de trabalho, participação em *workshops*, seminários e conferências, ou ainda uma combinação destas formas.

2. **Identificar o tempo de duração de cada fase de desenvolvimento**

   Cabe ao Líder de Projeto fazer uma alocação do tempo que cada fase do ciclo de vida irá levar. Esta alocação deverá ser feita na distribuição do tempo de duração do projeto, obtida segundo uma metodologia definida pela organização e que deve ter sido iniciada na execução da atividade Levantar Estimativas, descrita anteriormente neste item.

   Estes tempos de duração devem ser medidos ao final de cada fase. As fases seguintes deverão ter seus prazos reestimados.

3. **Identificar as atividades que serão desenvolvidas em cada fase**

   O Líder de Projeto, sendo apoiado pelo Grupo de *Software*, deve utilizar o Cronograma para auxiliar na identificação das atividades a serem seguidas pelo time ao longo do desenvolvimento. Este gabarito pode ser alterado para se adequar melhor às necessidades do time e do projeto.

   Nem todas as atividades precisam ser definidas de uma vez. Elas podem ir surgindo ao longo das iterações de desenvolvimento. Uma boa recomendação é detalhar na fase de concepção, apenas as atividades desta fase e algumas da fase de elaboração, se necessário. Na fase de elaboração procede-se da mesma forma, e assim por diante até a fase de transição.

4. **Distribuir as atividades durante os dias de desenvolvimento**

   Uma vez identificadas as atividades a serem seguidas durante o desenvolvimento, tanto com relação aos grupos internos quanto aos grupos externos à organização, cabe ao Líder de Projeto distribuir todas elas ao longo de uma ou mais fases de desenvolvimento, definindo prazos adequados para que elas possam ser cumpridas segundo apenas a sua

estimativa. Além disto, para cada atividade deve ser alocada, no mínimo, uma pessoa para o seu desenvolvimento. Na atividade seguinte, Negociar Compromissos, todas estas atividades precisarão ser adequadamente negociadas com todos os envolvidos, interna ou externamente ao projeto.

5. **Revisar o cronograma**

   Cabe à Gerência de Projeto e ao Grupo de *Software* revisar o Cronograma produzido pelo Líder de Projeto, para que se tenha maior garantia de que as atividades, os prazos e os recurso alocados foram definidos de forma adequada.

6. **Revisar o PDA**

   O Líder de Projeto deve encaminhar os itens preenchidos do PDA para uma revisão pela Gerência de Projeto. Ao término desta revisão, pode-se considerar que este artefato esteja aprovado.

Para a atividade Negociar Compromissos:

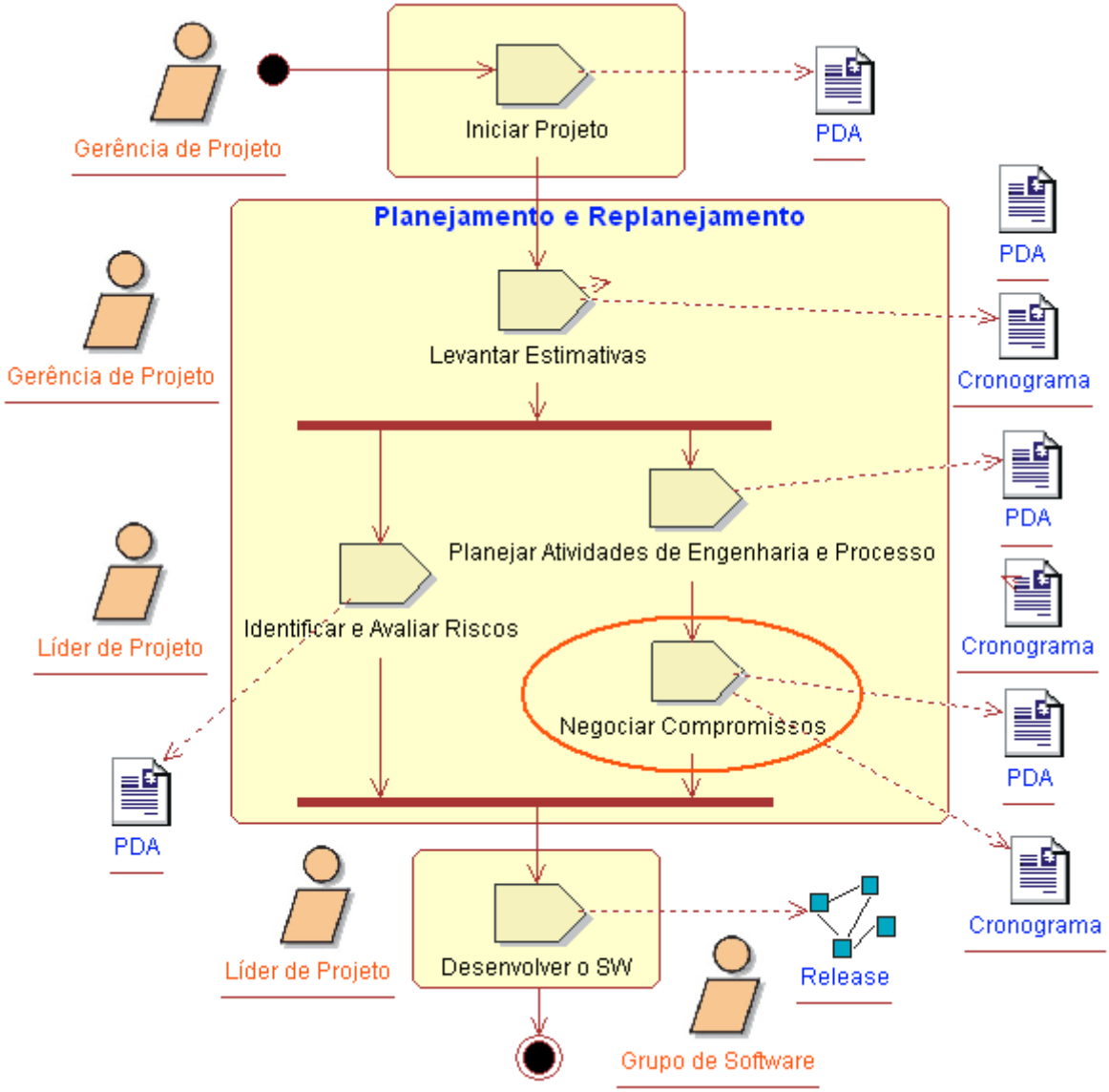

Figura 39 – Atividade: Negociar Compromissos

Artefatos de Entrada: PDA e Cronograma

Artefatos de Saída: PDA e Cronograma

Papel: Líder de Projeto

Finalidade:

Negociar os compromissos derivados dos requisitos alocados ao *software* ou devido a mudanças nestes requisitos, bem como no planejamento e replanejamento das atividades do projeto, para que o desenvolvimento possa ser realizado dentro de prazos adequados.

Passos:

1. **Negociar compromissos internos**

   Cabe ao Líder de Projeto negociar junto aos integrantes do projeto os prazos para que os produtos de trabalho derivados das atividades definidas no Cronograma possam ser entregues de forma adequada.

   Sempre que o projeto sofrer replanejamento, os novos compromissos derivados das mudanças produzidas deverão ser renegociados.

2. **Negociar compromissos externos**

   Cabe ao Líder de Projeto negociar junto ao Cliente e outros Interessados externos à organização os prazos para fornecer as necessidades, para que o time de projeto possa iniciar ou dar continuidade ao desenvolvimento. Estas necessidades podem ser equipamentos, *softwares*, móveis, autorizações, contratos, padrões, manuais, recomendações, termos técnicos ou outros documentos.

   Sempre que o projeto sofrer replanejamento, os novos compromissos derivados das mudanças produzidas deverão ser também renegociadas.

3. **Revisar Compromissos Externos e Mudanças**

   Cabe à Gerência de Projeto ser comunicada para poder revisar os compromissos acordados com os grupos externos à organização e também com relação às mudanças nestes compromissos. Estes compromissos envolvem as datas combinadas para entregas previstas do time de desenvolvimento para com o Cliente.

Para a atividade Desenvolver o *Software*:

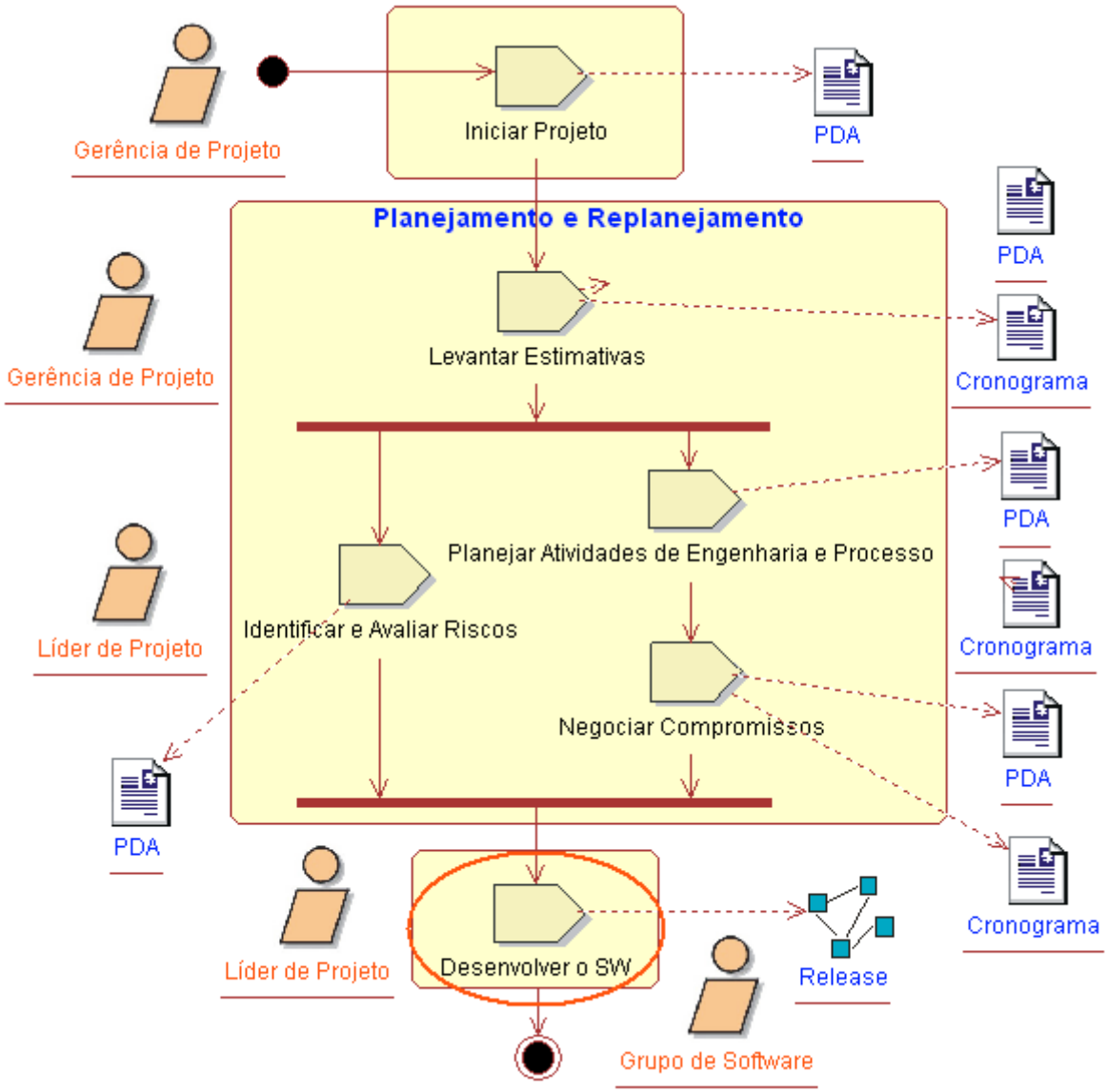

Figura 40 – Atividade: Desenvolver o *Software*

Artefatos de Entrada: PDA, Cronograma e MCU

Artefato de Saída: *Release*

Papel: Grupo de *Software*

Finalidade:

Desenvolver *releases* do sistema, seguindo o modelo de engenharia de *software* da organização.

Passos:

1. **Seguir o modelo de engenharia de *software* da organização**

   O processo LPA não contempla as etapas de desenvolvimento da organização, ficando a critério dela definir a forma mais adequada de desenvolvimento. A única restrição é que o modelo seguido deverá ser aderente ao modelo de desenvolvimento segundo o ciclo de vida iterativo e incremental, com minicascatas em cada iteração.

2. **Testar**

   O processo LPA não define os critérios e tipos de testes a serem seguidos durante o desenvolvimento, cabendo à organização definir aqueles que mais lhe convierem. Devido ao fato de o LPA ser orientado ao ciclo de vida iterativo e incremental, recomenda-se fortemente que os testes sejam planejados desde o início do desenvolvimento. Nesta etapa, podem ser identificados diversos casos de teste para cada caso de uso existente.

3. **Revisar**

   Nenhuma *release* pode ser entregue ao Cliente sem ter passado antes por todos os testes definidos pela organização, e após a correção de todos os defeitos identificados. Além disto, a *release* somente poderá ser liberada após contemplar todos os testes com sucesso. A partir deste momento, a *release* pode ser considerada aprovada pelo Líder de Projeto e verificada pela Gerência de Projeto.

3.5.2 Fluxo de Trabalho Detalhado da Macroatividade Acompanhar PDA[19]

Para a atividade Acompanhar PDA:

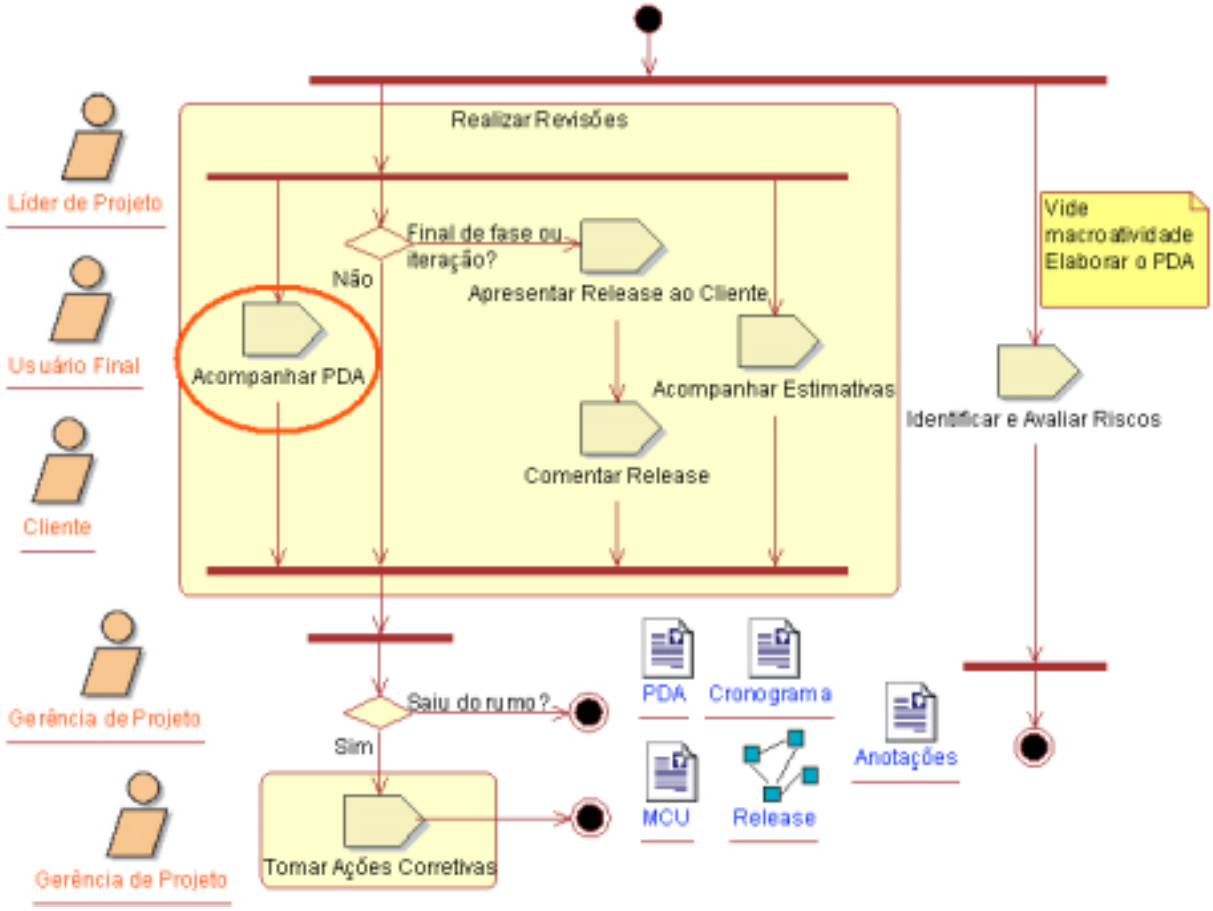

Figura 41 – Atividade: Acompanhar PDA

Artefatos de Entrada: Cronograma, PDA e MCU

Artefatos de Saída: Cronograma e PDA

Papel: Líder de Projeto

Finalidade:

Capturar o *status* corrente do projeto e avaliar este *status* com o planejado.

[19] A atividade “Identificar e Avaliar Riscos” já foi descrita na macroatividade “Elaborar O PDA” (item 3.5.1).

Passos:

1. **Treinamento**

   A Gerência de Projeto e o Líder de Projeto deverão ser treinados com relação aos aspectos técnicos do projeto e também com relação ao relacionamento entre pessoas e liderança, caso não possua experiência anterior, para poderem conduzir as atividades de acompanhamento de forma adequada. Este treinamento poderá ser realizado sob a forma de um curso contratado, tutoriais on-line, repasse de conhecimento de colegas de trabalho, participação em *workshops*, seminários e conferências, ou ainda uma combinação destas formas.

2. **Capturar o *status* do trabalho periodicamente**

   O Líder de Projeto deverá definir, no início do projeto, a periodicidade a ser seguida para os acompanhamentos do andamento do mesmo. Estes acompanhamentos deverão ser feitos sob a forma de reuniões e deverá ter a presença do Grupo de *Software.* A Gerência de Projeto poderá participar apenas se necessário. As ocorrências destas reuniões devem estar previstas no Cronograma do projeto e devem ter sido negociadas com os envolvidos de acordo com a atividade Negociar Compromissos, descrita no item 3.5.1.

   Nesta reunião, o Líder de Projeto deve acompanhar a evolução das atividades realizadas até a presente data, se elas foram cumpridas com sucesso ou não. Em caso de realização com sucesso, isto deverá ser comprovado mediante a apresentação dos resultados obtidos, por meio de documentos ou *releases* produzidas, onde todos irão avaliar se os resultados encontram-se satisfatórios com os objetivos das atividades. Caso a realização não tenha sido alcançada com sucesso, o responsável pela atividade deverá levantar os problemas encontrados, as hipóteses possíveis e se surgiram novos riscos.

   Além do acompanhamento das atividades definidas no projeto, o Líder de Projeto deverá acompanhar também as demais informações contidas no PDA, realizando medições de recursos financeiros, recursos de *hardware* e *software*, esforço dos recursos humanos para com o projeto e para com as atividades do processo com a Gestão de Requisitos e a Gestão de Planejamento e Acompanhamento, recursos computacionais críticos, estimativas de risco, tamanho dos casos de uso, tempo com o desenvolvimento dos casos de uso e outras que considerar necessárias, além das que foram estimadas no início do projeto.

3. Capturar o *status* do trabalho eventualmente

O Líder de Projeto deverá definir, no início do projeto, os momentos em que irão acontecer os acompanhamentos eventuais para avaliação do *status* do projeto. Sugere-se que estas reuniões sejam feitas nos finais de iterações ou sempre que acontecer uma liberação de *release*. Estes acompanhamentos deverão ser feitos sob a forma de reuniões e ter como foco principal a apresentação de uma *release* do sistema para o Cliente, conforme descrito na atividade seguinte Apresentar *Release* ao Cliente. Desta forma, é fundamental que estejam presentes o Cliente ou seu representante e outros Interessados externos. Em último caso, se nem o Cliente nem seu representante puderem estar presentes, a *release* poderá ser enviada formalmente para sua apreciação. Além destes integrantes, devem comparecer a Gerência de Projeto, o Líder de Projeto e o Grupo de *Software* (no todo ou em parte). As ocorrências destas reuniões devem estar previstas no Cronograma do projeto e devem ter sido negociadas com os envolvidos de acordo com a atividade Negociar Compromissos, descrita no item 3.5.1.

Nesta reunião, o Líder de Projeto deverá apresentar a *release* corrente do sistema, conforme definido na atividade seguinte Apresentar *Release* ao Cliente, para saber se as necessidades do Cliente estão sendo atendidas.

4. Fazer Verificações Periódicas e Eventuais

A Gerência de Projeto deverá fazer verificações com o Líder de Projeto com relação às atividades da Gestão de Requisitos e da Gestão de Planejamento e Acompanhamento de forma periódica e/ou nos finais de iteração e/ou fase.

O que verificar?

- O *status* de cada requisito do projeto.
- Se ocorreram mudanças e se estão sob controle.
- Se o Cliente encontra-se satisfeito com os requisitos implementados até o momento.
- Se o plano de desenvolvimento está sendo seguido.
- Se as estimativas iniciais estão sendo medidas e avaliadas quanto às diferenças encontradas.
- Se o desempenho técnico do projeto está sendo satisfatório conforme previsto.

- Se os gastos estão conforme o planejado.
- Se o time está coeso.
- Se o cronograma está sendo atualizado e seguido pelo time.
- Se existem questões não resolvidas entre o Líder de Projeto e o Grupo de *Software.*
- Se os riscos estão sendo reduzidos, se surgiram novos, se algum se fez presente e que ações estão sendo tomadas.
- Se os recursos computacionais críticos estão disponíveis e operando adequadamente.

5. Avaliar os resultados e iniciar replanejamento

Para as reuniões periódicas:

- Ao término da reunião, o Líder de Projeto, em consenso com todos, deve chegar a um resultado da reunião, ou seja, se foi satisfatória ou não. Os sucessos obtidos devem dar origem a novas atividades, e os fracassos devem ser anotados e avaliados quanto aos impactos do não cumprimento dos compromissos futuros, culminando com uma tomada de decisão para ações corretivas. Para isto, consulte a atividade Tomar Ações Corretivas, descrita neste item.
- Os resultados da reunião devem ser registrados sob a forma de uma ata de reunião, um comunicado por email ou ainda no próprio PDA na sessão de acompanhamento.

Para as reuniões eventuais:

- Ao término da reunião, o Líder de Projeto deve chegar a um consenso do resultado da reunião, tendo por objetivo principal as avaliações referentes à *release* apresentada aos interessados externos da organização. Os sucessos obtidos devem dar origem a novas atividades e os fracassos devem ser anotados para a tomada de decisão a ser realizada na atividade Tomar Ações Corretivas, descrita neste item.
- Os resultados da reunião devem ser registrados sob a forma de uma ata de reunião, um comunicado por email ou ainda no próprio PDA na sessão de acompanhamento

Em todas as situações de mudanças definidas no PDA, bem como no surgimento devido a realização de ações corretivas, este documento deverá ser alterado para se adequar ao novo cenário de desenvolvimento (replanejamento). Para isto, deve-se seguir os passos definidos nas atividades Levantar Estimativas, Identificar e Avaliar Riscos, Planejar as Atividades de Engenharia e de Processo e Negociar Compromissos, descritas no item 3.5.1.

Para a atividade Apresentar *Release* ao Cliente:

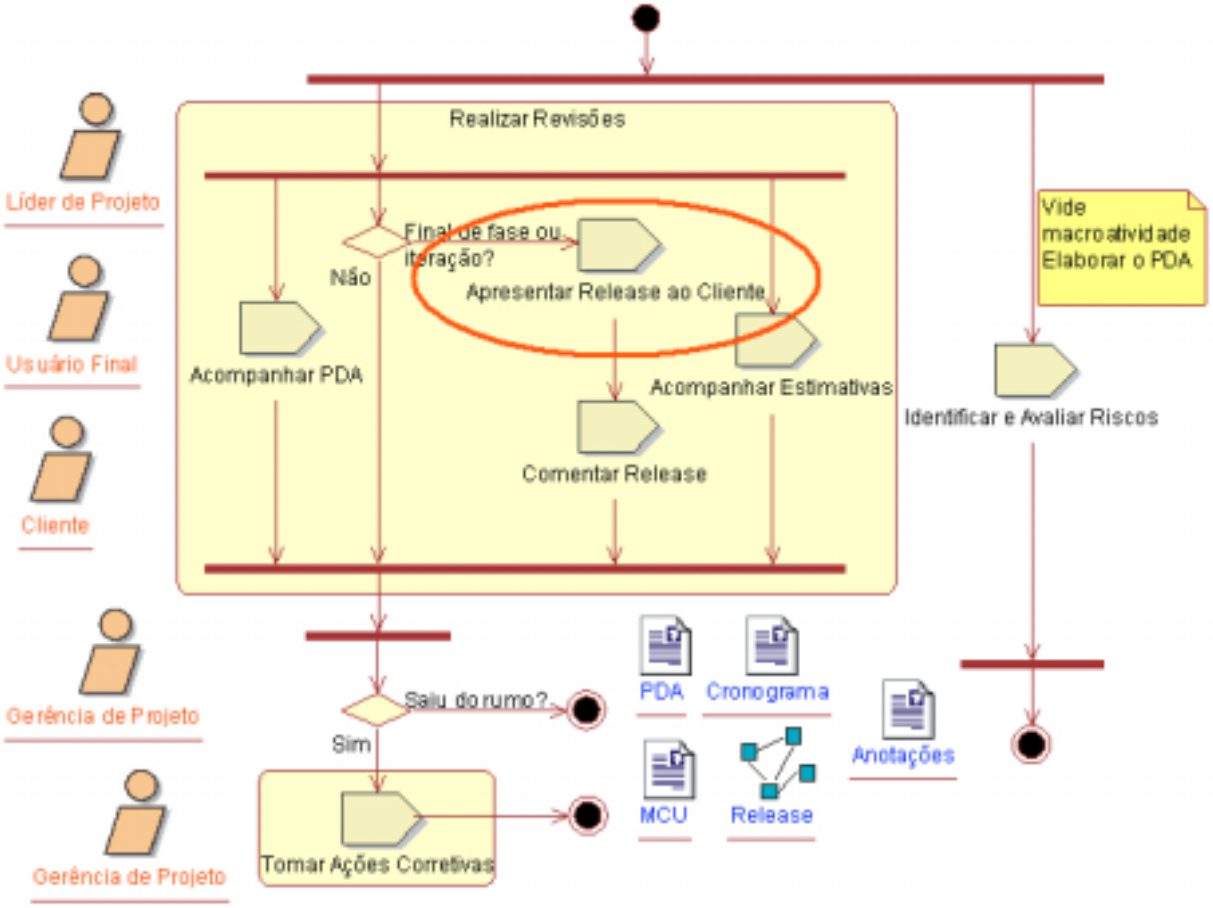

Figura 42 – Atividade: Apresentar *Release* ao Cliente

Artefatos de Entrada: PDA, Cronograma e MCU, *Release*

Artefato de Saída: Anotações

Papel: Líder de Projeto

Finalidade:

Obter uma avaliação do Cliente sobre o desenvolvimento, ou seja, se está de acordo com suas necessidades.

Passos:

1. **Apresentar *release* ao Cliente e a outros Interessados**

   Quanto antes o Cliente puder receber os resultados alcançados com os trabalhos de desenvolvimento, melhor. Vale lembrar que o LPA segue o modelo iterativo e incremental e que possui, dentre outros aspectos importantes, o de ir apresentando *releases* do sistema na medida em que estas vão sendo concluídas. Desta forma, as funcionalidades do sistema vão sendo apresentadas aos poucos e de forma crescente. O crescimento das funcionalidades define a capacidade operacional do sistema num dado momento.

   A partir de uma reunião eventual para captura do *status* do trabalho, descrita na atividade anterior, o Líder de Projeto deverá apresentar a *release* produzida para o Cliente ou seu representante. Para evitar superexpectativas por parte do Cliente, o Líder de Projeto deverá antecipar os objetivos da reunião, as capacidades, limitações e restrições funcionais da *release* apresentada.

   Caso seja necessário, esta *release* poderá ser enviada antecipadamente para apreciação pelo Cliente, para que a reunião possa se desenrolar de forma mais eficiente.

2. **Obter parecer do Cliente e de outros Interessados**

   As responsabilidades do Cliente e de outros Interessados externos à organização encontram-se descritas na atividade Comentar *Release*, descrita ainda neste item.

   Durante a reunião, uma pessoa que tenha assumido o papel de redator deve ir fazendo Anotações de todas as observações identificadas, não apenas do Cliente, mas de todos os participantes que possam agregar valor para a melhoria da *release* apresentada.

   O Líder de Projeto deve ficar atento neste momento para que as sugestões apresentadas não sejam, na verdade, sugestões de melhorias futuras ou novas funcionalidades. Em qualquer um destes casos, deve ser iniciada a macroatividade Gerenciar e Controlar Mudanças nos Requisitos, descrita no item 3.4.3. Deve ficar claro para o Cliente que esta ação se torna necessária para que o projeto não saia do rumo, não significando dizer que as sugestões apresentadas irão sempre envolver mudanças de custos e de prazos de Cronograma.

   Ao término da reunião, o Líder de Projeto deve perguntar ao Cliente e demais Interessados externos se os objetivos da *release* apresentada corresponderam às suas necessidades.

3. **Avaliar os resultados**

   Concluída a reunião, o Líder de Projeto, em conjunto com o Grupo de *Software*, deve fazer uma avaliação da *release* apresentada, procurando identificar os pontos fortes e os pontos fracos, bem como as sugestões apresentadas.

4. **Tomar ações corretivas**

   As ações corretivas visam eliminar os defeitos identificados na *release* apresentada ao Cliente, bem como aplicar planos para a redução ou eliminação dos riscos anteriormente estimados, de novos riscos que possam ter surgido, ou mesmo aplicar planos de contingência caso algum risco tenha se tornado real. Em todos os casos de risco execute a atividade Identificar e Avaliar Riscos, descrita no item 3.5.1.

Para a atividade Comentar *Release*:

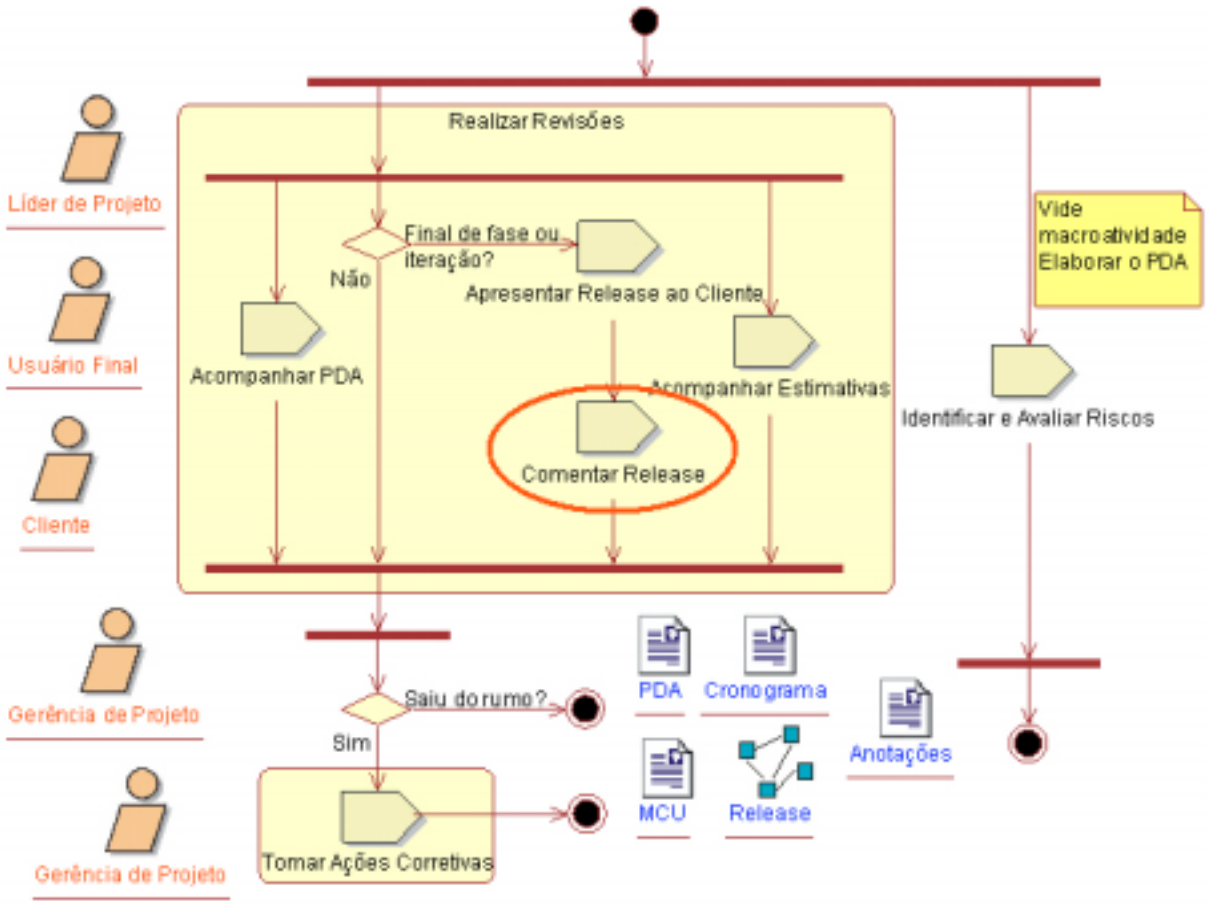

Figura 43 – Atividade: Comentar *Release*

Artefatos de Entrada: *Release*

Artefato de Saída: Anotações

Papel: Líder de Projeto

Finalidade:

Colher informações do Cliente e de outros Interessados externos à organização sobre a *release*, para atender da melhor forma possível às necessidades do mesmo.

Passos:

1. **Avaliar a *release* apresentada**

   O Cliente e outros Interessados externos da organização, como por exemplo Usuários Finais, devem fazer uma avaliação da capacidade contida na *release* apresentada pelo Líder de Projeto em nome do time de desenvolvimento. Esta avaliação deve ser feita durante a reunião eventual para acompanhamento do *status* do trabalho descrita neste item na atividade Acompanhar PDA, e a pessoa no papel de redator deve fazer as anotações, conforme descrito no documento Anotações.

   O Cliente deve estar ciente de que a *release* apresentada, a menos que seja a última, não irá conter todas as funcionalidades previstas, o que irá definir o escopo da avaliação.

   Outro aspecto importante é que ele deve concentrar os esforços na identificação dos possíveis defeitos existentes na *release*. Uma boa estratégia é receber a *release* antecipadamente do Líder de Projeto e aplicar os testes que ele considerar convenientes. As sugestões e comentários serão bem-vindos pelo Líder de Projeto. Até mesmo alterações dos requisitos ou a solicitação de novas funcionalidades devem ser aceitas, já que o desenvolvimento deve ser feito de forma que o Cliente fique satisfeito com os resultados e que realmente venha a utilizar o produto que irá receber. Todavia, deve ficar claro que qualquer alteração que afete o escopo de desenvolvimento, definido no documento Visão, irá disparar todas as atividades da macroatividade Gerenciar e Controlar Mudanças nos Requisitos, descrita no item 3.4.3, para que isto possa ser feito de forma controlada. Estas alterações podem, mas não necessariamente, provocar alterações dos prazos de conclusão do projeto e de seus custos, conforme definido no documento PDA.

Para a atividade Acompanhar Estimativas:

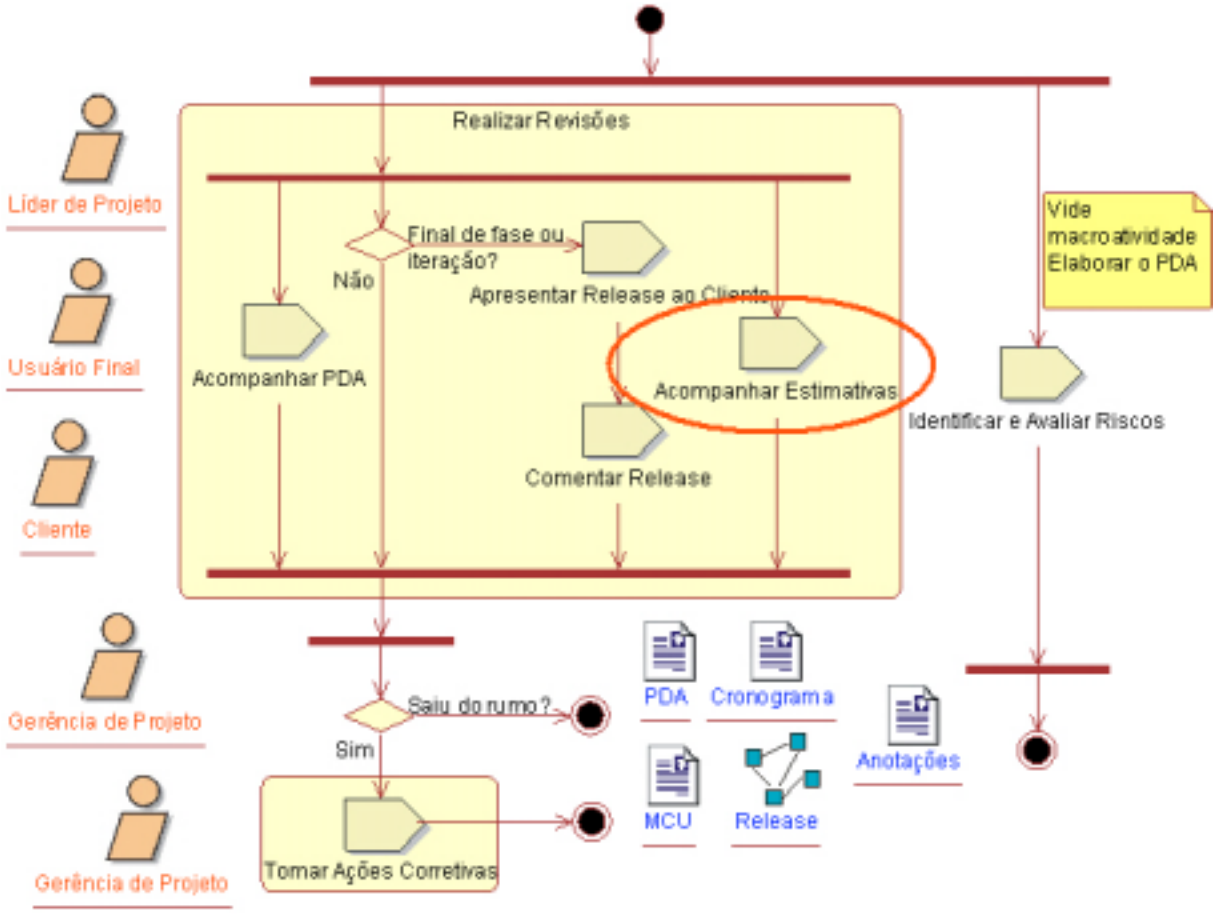

Figura 44 – Atividade: Acompanhar Estimativas

Artefatos de Entrada: MCU, Cronograma e PDA

Artefato de Saída: PDA

Papel: Líder de Projeto

Finalidade:

Acompanhar as estimativas de tamanho, esforço, prazo e custo que foram determinadas no início do projeto e atualizadas nos marcos do desenvolvimento, para que possa ser feita uma análise dos resultados e aplicar melhorias e ajustes, permitindo com isto estimativas mais precisas nos futuros projetos.

Passos:

1. **Fazer medidas em marcos do desenvolvimento**

   Nos marcos principais do projeto, tais como nos finais das fases de concepção, elaboração, construção e transição, ou ainda ao final de cada iteração, o Líder de Projeto deverá coletar medidas de tamanho do projeto (casos de uso), tamanho de documentos, esforço em atividades de projeto (tempo consumido com as atividades de engenharia, tais como análise de requisitos, arquitetura, projeto, codificação e teste) e de processo (tempo consumido para a realização de atividades referentes ao processo LPA nas atividades de requisitos, planejamento e acompanhamento), custo do projeto e andamento do Cronograma, e registrar todos estes valores no PDA.

2. **Confrontar os valores medidos com os estimados**

   O Líder de Projeto deverá sempre comparar os valores medidos com os estimados anteriormente. Nesta comparação, deverá identificar se as diferenças são para mais ou para menos do valor estimado, e se estas diferenças ocorreram devido a falhas das estimativas (sub ou superestimadas) ou devido a grandes facilidades ou grandes dificuldades encontradas na realização das atividades do projeto. Este exercício irá permitir que o time vá adquirindo maior experiência com as estimativas, podendo produzir estimativas mais precisas em outras oportunidades.

3. **Aplicar os ajustes necessários**

   A situação mais crítica e que deve ser analisada com cuidado pelo Líder de Projeto e eventualmente pela Gerência de Projeto são as diferenças identificadas para mais nas medidas que ultrapassarem os valores estimados (por exemplo: uma atividade prevista para levar 10 dias levou 15 dias, provocando uma diferença, para mais, de 5 dias), o que pode provocar comprometimento nas entregas definidas no PDA. O Líder de Projeto deve ficar atento para identificar os motivos de tal ocorrência, avaliar as causas e tomar medidas corretivas ou de contingência para que o projeto volte ao controle.

   Além disto, caso as diferenças identificadas, para mais ou para menos, tenham acontecido devido a falhas de estimativas, o Líder de Projeto, em conjunto com a Gerência de Projeto e o Grupo de *Software*, devem ajustar a metodologia utilizada nas estimavas adotadas, tais como alteração de expressões de cálculo, maior treinamento, diminuição de subjetividade etc, para que elas possam ser aplicadas de forma mais precisa em novos projetos.

Para a atividade Tomar Ações Corretivas:

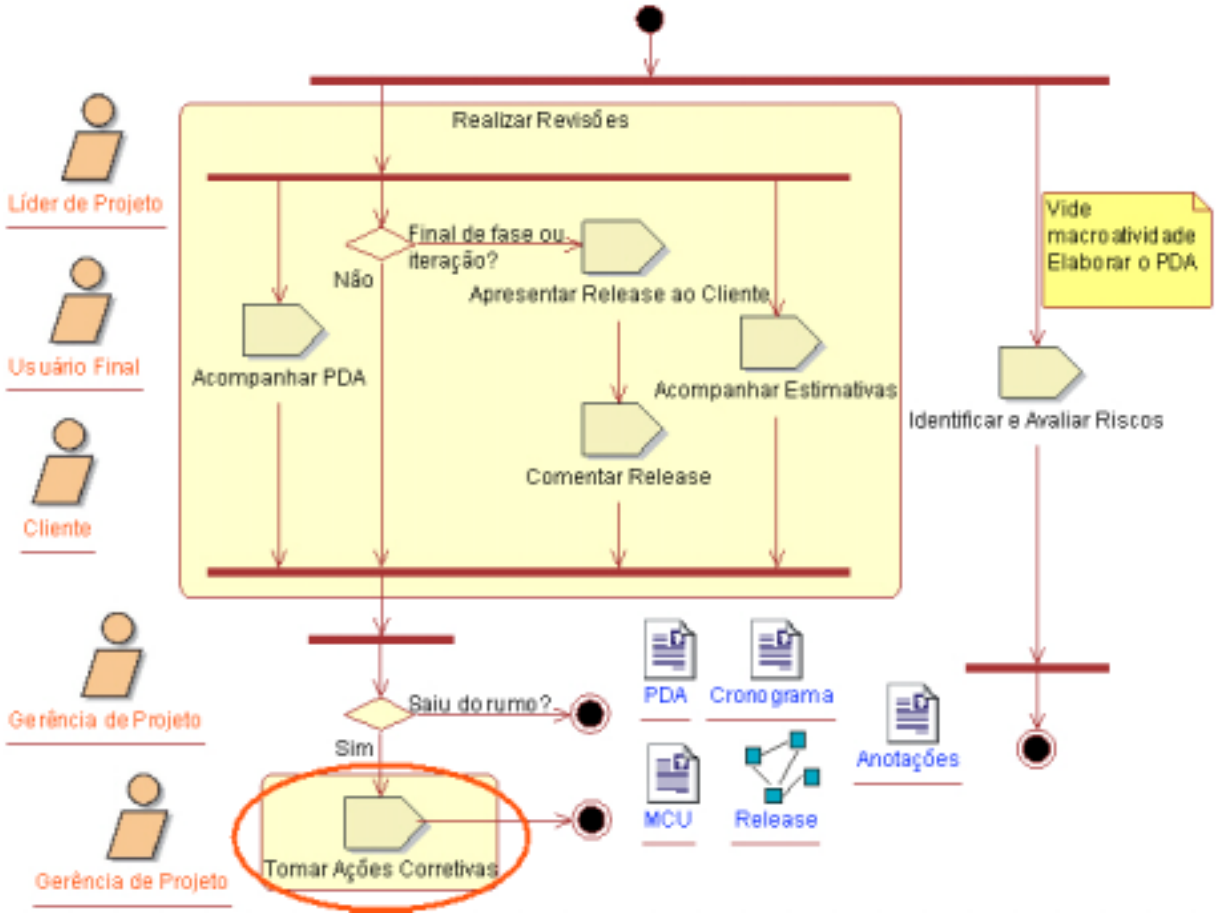

Figura 45 – Atividade: Tomar Ações Corretivas

Artefatos de Entrada:

Artefato de Saída:

Papel: Líder de Projeto

Finalidade:

Fazer com que o projeto retorne ao rumo correto, aplicando-se ações corretivas ou planos de contingência e renegociação de compromissos internos e externos.

Passos:

1. **Atualizar e identificar novos riscos**

   O Líder de Projeto deve ficar atento para que os riscos existentes no projeto e devidamente documentados no PDA sejam acompanhados para a sua redução gradativa. O fato de o projeto ter saído do rumo pode significar que um risco não tenha sido reduzido suficientemente ou que possam ter surgido outros.

2. **Aplicar planos de contingência**

   Caso algum risco venha a acontecer, seja ele previsto ou não, será o momento de serem aplicados planos de contingência (plano B) para fazerem com que o projeto retorne ao seu rumo. Em virtude da aplicação de planos de contingência, é bem provável que o Cliente venha a sofrer conseqüências, o que deverá levar a uma renegociação de compromissos.

3. **Renegociar compromissos**

   A aplicação de ações corretivas e/ou planos de contingência irá certamente provocar alterações no Cronograma de desenvolvimento e possivelmente nas entregas previstas para o Cliente. Todos os novos compromissos identificados deverão ser devidamente renegociados com os envolvidos, tanto internos quanto externos, para que as responsabilidades possam ser atribuídas.

   Dependendo do tipo de compromisso a ser alterado devido a um plano de contingência, pode ser que até o escopo do projeto seja modificado, o que deverá provocar também alterações contratuais entre Cliente e a organização desenvolvedora.

   Todas as mudanças de compromissos externos deverão ser revisadas, aprovadas, formalmente ou não, pela Gerência de Projeto e comunicadas a todos os envolvidos no projeto.

3.5.3 Fluxo de Trabalho Detalhado da Macroatividade Definir PGPA

Para a atividade Definir PGPA:

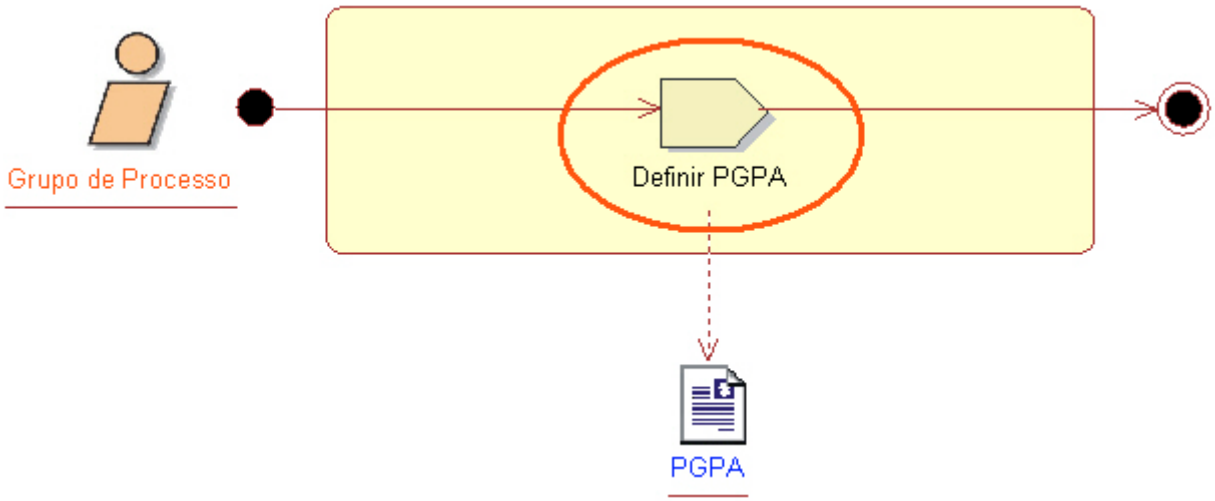

Figura 46 – Atividade: Definir PGPA

Artefatos de Entrada: Inexistente

Artefatos de Saída: PGPA

Papel: Grupo de Processo

Finalidade:

Definir regras claras a serem seguidas durante a gestão de planejamento e acompanhamento.

1. **Definir e Documentar as Políticas da Gestão de Planejamento e Acompanhamento**

   As políticas a serem definidas podem ser utilizadas por diversos projetos ou sofrer ligeiras ou muitas alterações para cada tipo de projeto a ser desenvolvido.

   Estas políticas devem ser definidas pelo Grupo de Processo e em conjunto com as demais pessoas da organização.

   As políticas devem ser especificadas de forma documentada e tipicamente devem descrever:

- **Para a Gestão de Planejamento:**
    - Devem ser alocados recursos e verbas para suportar as atividades de acompanhamento do projeto.
    - O Líder de Projeto deve ser o responsável pela negociação de todos os compromissos do projeto.
    - Os requisitos de *software* alocados para desenvolvimento devem ser utilizados para o planejamento do projeto. Vide PGR.
    - Todos os compromissos do projeto identificados no PDA e Cronograma devem ser devidamente negociados com todos os envolvidos, tanto interna (compromissos internos) quanto externamente (compromissos externos).
    - Todos os envolvidos no projeto devem estar devidamente documentados no PDA.
    - As estimativas do projeto (tamanho, esforço, custo e Cronograma) devem ser revisadas pela Gerência de Projeto.
    - A Gerência de Projeto deve revisar todos os compromissos externos do projeto, fazendo uma aprovação formal ou não.
    - O PDA deve ser gerenciado (mantido sob controle de versão) e controlado[20] (com relação a mudanças da *baseline*).
- **Para a Gestão de Acompanhamento:**
    - Devem ser alocados recursos e verbas para suportar as atividades de acompanhamento do projeto.
    - O Líder de Projeto é o responsável pelas atividades e resultados do projeto.
    - Deve ser criado um plano de desenvolvimento - PDA de forma documentada, para que seja utilizado para acompanhamento do projeto.
    - O PDA deve incluir um Cronograma para distribuir as atividades, recursos e prazos para a realização das atividades de projeto.

[20] Como o LPA não cobre a KPA de Gestão de Configuração e Mudança, cabe à organização implantar um controle de configuração e mudança, caso queira atender a esta política.

- A Gerência de Projeto deve ser mantida informada do andamento do projeto nos finais de fase ou a qualquer momento, se necessário.

- O Líder de Projeto deve realizar reuniões periódicas para acompanhar o andamento do projeto. Devem participar desta reunião o Grupo de *Software* e outras pessoas e grupos, se necessário.
- O Líder de Projeto deve realizar reuniões nos finais de cada fase de desenvolvimento para acompanhamento do projeto e apresentação de *releases* para o Cliente. Devem participar desta reunião o Grupo de *Software* (parcial ou total), o Cliente ou seu representante, a Gerência de Projeto e outras pessoas ou grupos se necessário.
- Os resultados das reuniões periódicas ou de finais de fase devem ser registrados de alguma forma e distribuídos para todos os envolvidos.
- O Líder de Projeto deve tomar ações corretivas para reduzir riscos do projeto ou aplicar planos de contingência sempre que o projeto sair do rumo. Em todos os casos, o PDA e outros artefatos, se necessário, devem ser atualizados.
- Todas as mudanças de compromissos, tanto interna quanto externamente, só poderão ser formalizadas após devida negociação com os envolvidos.
- A Gerência de Projeto deve revisar todas as mudanças de compromissos externos do projeto, fazendo uma aprovação formal ou não.

2. Avaliar os Resultados

Estando o PGPA devidamente elaborado, deverá ser revisado e aprovado pela Gerência de projeto.

3.6 Papéis do LPA

3.6.1 Cliente

O papel de Cliente é uma especialização do papel Interessado. Segundo PAULK et al. (1999a, p. 355), corresponde a uma pessoa ou o responsável de uma organização que aceita o produto produzido pela empresa desenvolvedora e autoriza o pagamento a ela.

Responsabilidades:

- Fornecer prazo estimado para execução do projeto.
- Arcar com as cláusulas contratuais firmadas entre a organização do Cliente e a organização desenvolvedora.
- Fornecer informações para preenchimento do PI pela Gerência de Projeto.
- Indicar outras pessoas que possam contribuir com o PI.
- Fornecer ou indicar as pessoas que farão as entregas de documentos, equipamentos etc, que possam ser necessários por parte da organização desenvolvedora, para trabalhar no desenvolvimento do projeto.

A Figura 47 ilustra as atividades alocadas a este papel e os artefatos utilizados e/ou produzidos.

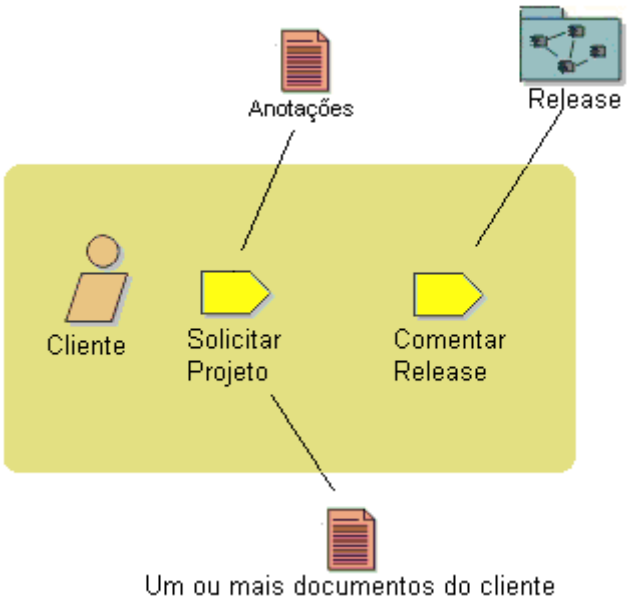

Figura 47 – Atividades exercidas pelo papel Cliente

3.6.2 Gerência de Projeto

Segundo PAULK et al. (1999c, p. 363), a Gerência de Projeto é responsável por todos os negócios de um projeto específico. É a pessoa que dirige, controla, administra e regula a construção de um projeto de *software* ou de um sistema de *hardware* e *software*. Esta gerência é que assume todas as responsabilidades para com o Cliente.

Habilidades e Experiência

As habilidades e experiências necessárias para que alguém assuma o papel da Gerência de Projeto irá depender do tamanho e complexidade técnica e gerencial do projeto a ser gerenciado. Todavia, para projetos pequenos com duração de até 6 meses[21], este papel deverá atender aos seguintes quesitos:

- Ser experiente no domínio da aplicação e no desenvolvimento do *software*.
- Possuir habilidades para analisar e tomar decisões quanto ao planejamento, estimativas e na gerência e análise de riscos.
- Habilidades na negociação, comunicação e apresentação.
- Demonstrar liderança e capacidade de construir um time de trabalho.
- Saber gerenciar seu tempo e tomar decisões rapidamente e sob forte pressão de trabalho.
- Possuir habilidades interpessoais e ser capaz de selecionar adequadamente sua equipe de trabalho.
- Ser objetivo na configuração e distribuição do trabalho, garantindo envolvimento do time.
- Compartilhar sua visão arquitetural do projeto, sendo porém pragmático na definição de escopo e na implementação dos planos de trabalho, e altamente honesto na avaliação dos resultados.
- Focalizar no valor daquilo que será entregue para o Cliente, de forma que o *software* atenda ou mesmo exceda às expectativas do cliente

[21] Maiores detalhes sobre a classificação de pequenos projetos, consulte PAULK M. C. (1999s).

Responsabilidades na Gestão de Requisitos:

- Gerenciar e documentar os requisitos do sistema e suas alocações ao *software* no documento Visão durante todo o ciclo de vida do projeto (PAULK et al., 1999d, p.128).
- Revisar os requisitos alocados ao *software* (PAULK et al., 1999e, p. 127).
- Efetivar as mudanças que ocorrerem nos requisitos do sistema e nas suas alocações ao *software* (PAULK et al., 1999f, p. 128).
- Os requisitos documentados devem incluir os requisitos não-técnicos que afetam e determinam as atividades do projeto de *software*. Alguns exemplos de requisitos não-técnicos são: acordos, condições e termos contratuais. Alguns exemplos de acordos, condições e termos contratuais incluem: produtos a serem entregues, datas de entrega e marcos do ciclo de vida (PAULK et al., 1999g, p. 128).
- Os requisitos documentados devem incluir os requisitos técnicos do *software*. Alguns exemplos de requisitos técnicos incluem: usuário final, operador, suporte ou funções de integração, desempenho, restrições de projeto, linguagem de programação, interface, portabilidade, usabilidade e todas as funcionalidades (PAULK et al., 1999h, p. 128).
- Os requisitos documentados devem incluir os critérios de aceitação que serão usados pelo Cliente para verificar que o produto a ser entregue irá atender aos requisitos alocados, validando com isto o produto por ele recebido ao final do desenvolvimento (PAULK et al., 1999i, p. 129).

Responsabilidades na Gestão de Planejamento e Acompanhamento

Planejamento de um projeto**:**

- Estabelecer uma declaração de trabalho, por meio do documento Visão, para que o planejamento do sistema possa iniciar (PAULK et al., 1999j, p. 136).
- Negociar os compromissos externos do projeto.
- Fazer o levantamento das estimativas de projeto, tais como tamanho, esforço, custo, restrições e cronograma.
- Alocar a equipe de trabalho.
- Possuir habilidades, não apenas técnicas, mas também com a gerência de pessoal.
- Definir os marcos de um projeto e os prazos de início e finalização.
- Definir os recursos necessários.
- Fazer o planejamento inicial.
- Identificar os recursos computacionais críticos.
- Definir e aprovar os critérios de aceitação.

Acompanhamento de um projeto:

- Responsabilizar-se pelo desenvolvimento do projeto perante o Cliente.
- Participar de reuniões periódicas de acompanhamento.
- Participar de reuniões eventuais de acompanhamento.
- Acompanhar os riscos, aplicando planos de contingência se necessário.
- Renegociar e aprovar mudanças de compromissos externos.
- Avaliar Pedidos de Mudança.
- Acompanhar o PDA, realizando medidas diversas.

A Figura 48 ilustra as atividades alocadas a este papel e os artefatos utilizados e/ou produzidos.

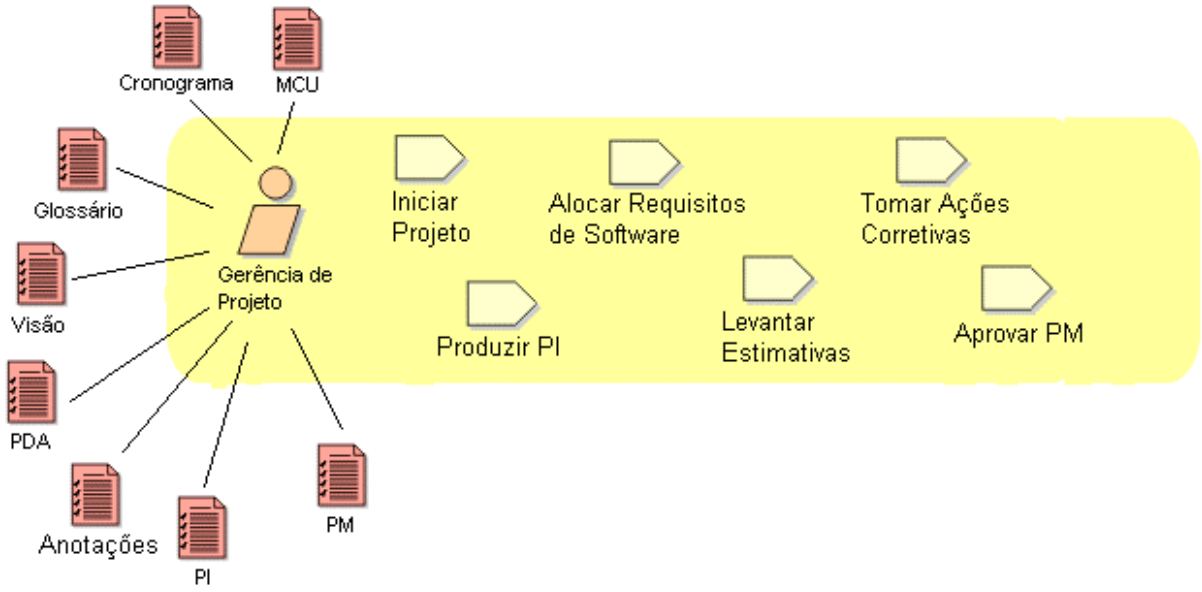

Figura 48 – Atividades exercidas pelo papel Gerência de Projeto

3.6.3 Grupo de Processo

O Grupo de Processo é responsável pelo desenvolvimento do processo de desenvolvimento de *software* da organização. Este processo, uma vez definido, deve ser devidamente ajustado ou configurado para um determinado projeto e sofrer melhorias regulares que vão sendo identificadas durante seu uso.

Na linguagem do SW-CMM, este grupo é conhecido como Grupo de Processo de Engenharia de *Software*, que segundo PAULK et al. (1999l, p. 364) é formado por especialistas que facilitam a definição, manutenção e melhoria do processo de *software* usado pela organização.

Habilidades e Experiência

As pessoas que atuam neste grupo devem possuir amplo conhecimento de Engenharia de *Software*, bem como fortes habilidades de comunicação.

Responsabilidades do Grupo de Processo:

- Elaborar, manter e melhorar o processo definido da organização.
- Institucionalizar o processo definido da organização.

A Figura 49 ilustra as atividades alocadas a este papel e os artefatos utilizados e/ou produzidos durante o desenvolvimento do projeto de *software*.

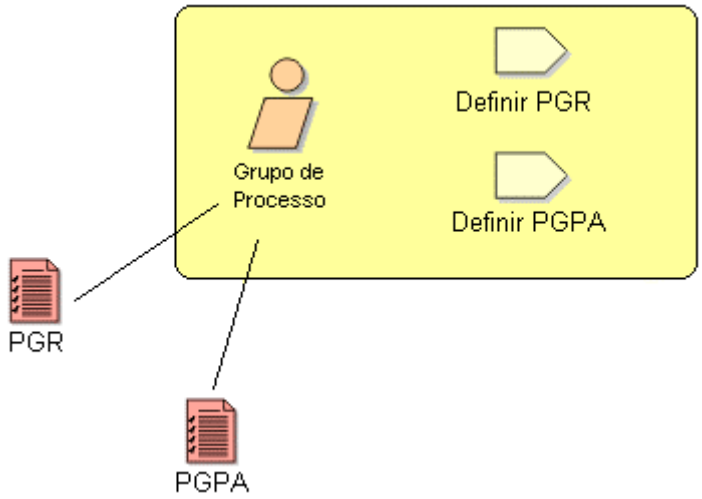

Figura 49 – Atividades exercidas pelo papel Grupo de Processo

3.6.4 Grupo de *Software*

O Grupo de *Software* é responsável pelo desenvolvimento e teste dos produtos de *software* (sistema, subsistema e componentes), de acordo com padrões adotados para o projeto (se existirem).

Habilidades e Experiência

- Conhecimento do sistema ou da aplicação sob teste.
- Familiaridade com testes e ferramentas de automação de testes.
- Habilidades com programação.

Responsabilidades do Grupo de *Software*:

- Participação do grupo, no todo ou em parte, das reuniões eventuais e periódicas de acompanhamento do projeto.
- Desenvolver o sistema de acordo com o PDA.
- Liberar *releases* para avaliação do Cliente.
- Revisar os requisitos para verificar se estão de acordo com as necessidades do Cliente.

A Figura 50 ilustra as atividades alocadas a este papel e os artefatos utilizados e/ou produzidos.

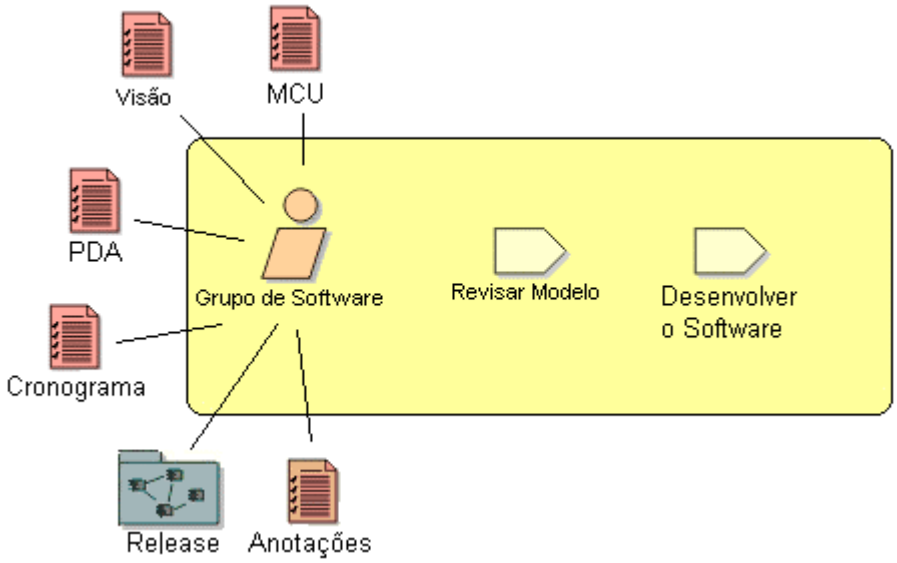

Figura 50 – Atividades exercidas pelo papel Grupo de *Software*

3.6.5 Interessado[22]

O Interessado é qualquer pessoa que direta ou indiretamente colabora para com o desenvolvimento do sistema, incluindo as pessoas do projeto, de outros departamentos da organização, do Cliente ou de outros fornecedores.

Este papel pode ser assumido por qualquer pessoa que esteja de alguma forma envolvida com os resultados do projeto.

Para que os problemas mais complexos de um projeto possam ser solucionados, é necessário que se entenda as necessidades dos Interessados. Tipicamente, os Interessados possuem diferentes visões do problema e também diferentes necessidades que devem ser encaminhadas para uma solução. Muitos dos Interessados são usuários do sistema. Outros Interessados são apenas usuários indiretos do sistema ou são envolvidos com os resultados dos negócios atrelados ao sistema. Um bom entendimento de quais são os Interessados de um sistema e suas necessidades formam elementos-chave para o desenvolvimento eficiente de uma solução em *software*.

[22] Este papel foi traduzido e adaptado do papel *"Role: Stakeholder"* do RUP - 2002.05.00.25.

- **Exemplos de Interessados:**
    - Cliente ou seu representante.
    - Usuário ou seu representante.
    - Investidor.
    - Parceiro financeiro.
    - Gerente de produção.
    - Comprador.
    - Projetista .
    - Testador.
    - Documentalista.
    - Outros.

Responsabilidades do Interessado:

- Contribuir para o fornecimento de informações de suas necessidades para o desenvolvimento do projeto.
- Participar das reuniões de entrega de *releases* do sistema
- Fazer comentários com relação às *releases* de desenvolvimento.

A Figura 51 ilustra as atividades alocadas a este papel e os artefatos utilizados e/ou produzidos.

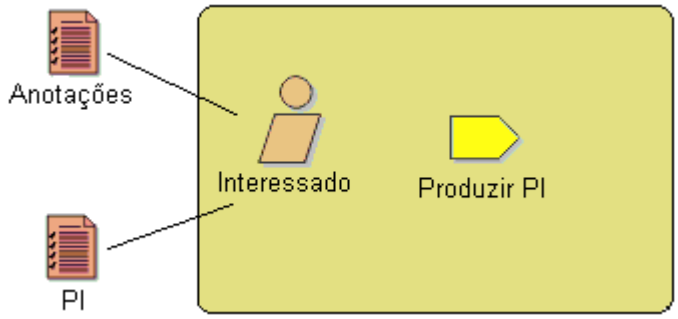

Figura 51 – Atividades exercidas pelo papel Interessado

3.6.6 Líder de Projeto

Para PAULK et al. (1999m, p. 368), o Líder de Projeto é aquele que assume tarefas técnicas para com uma equipe de desenvolvimento com relação a um projeto de *software*, devendo desta forma assumir responsabilidades técnicas e prover as direções para que a equipe do projeto possa desenvolver suas atividades.

O Líder de Projeto também é conhecido com Arquiteto de *Software* (RATIONAL SOFTWARE CORPORATION, 2002). Este assume papel de liderança e coordenação técnica das atividades e artefatos a serem produzidos durante o projeto. Ele estabelece a estrutura geral para cada visão arquitetural: a decomposição da visão, o agrupamento dos elementos e a interface entre os agrupamentos principais. Em contraste com outros papéis que possuem uma visão em profundidade, o Arquiteto possui sempre uma visão em "largura".

Responsabilidades na Gestão de Requisitos:

- Revisar os requisitos antes e após serem alocados ao *software*.
- Fazer o detalhamento dos requisitos de *software* alocados por meio do MCU.

Responsabilidades na Gestão de Planejamento e Acompanhamento

Planejamento de um projeto:

- Colher pedidos de mudança do sistema em desenvolvimento e proceder a análises de viabilidade.
- Identificar e avaliar riscos.
- Fazer o planejamento das atividades de processo e de engenharia do projeto.
- Negociar compromissos alocados ao projeto.

Acompanhamento de um projeto:

- Responsabilizar-se pelo andamento do projeto perante a Gerência de Projeto.
- Proceder ao acompanhamento eventual e periódico do PDA.
- Fazer apresentações de *releases* do projeto ao Cliente.
- Coordenar as atividades de projeto desempenhadas por ele e pelo Grupo de *Software*.

A Figura 52 ilustra as atividades alocadas a este papel e os artefatos utilizados e/ou produzidos.

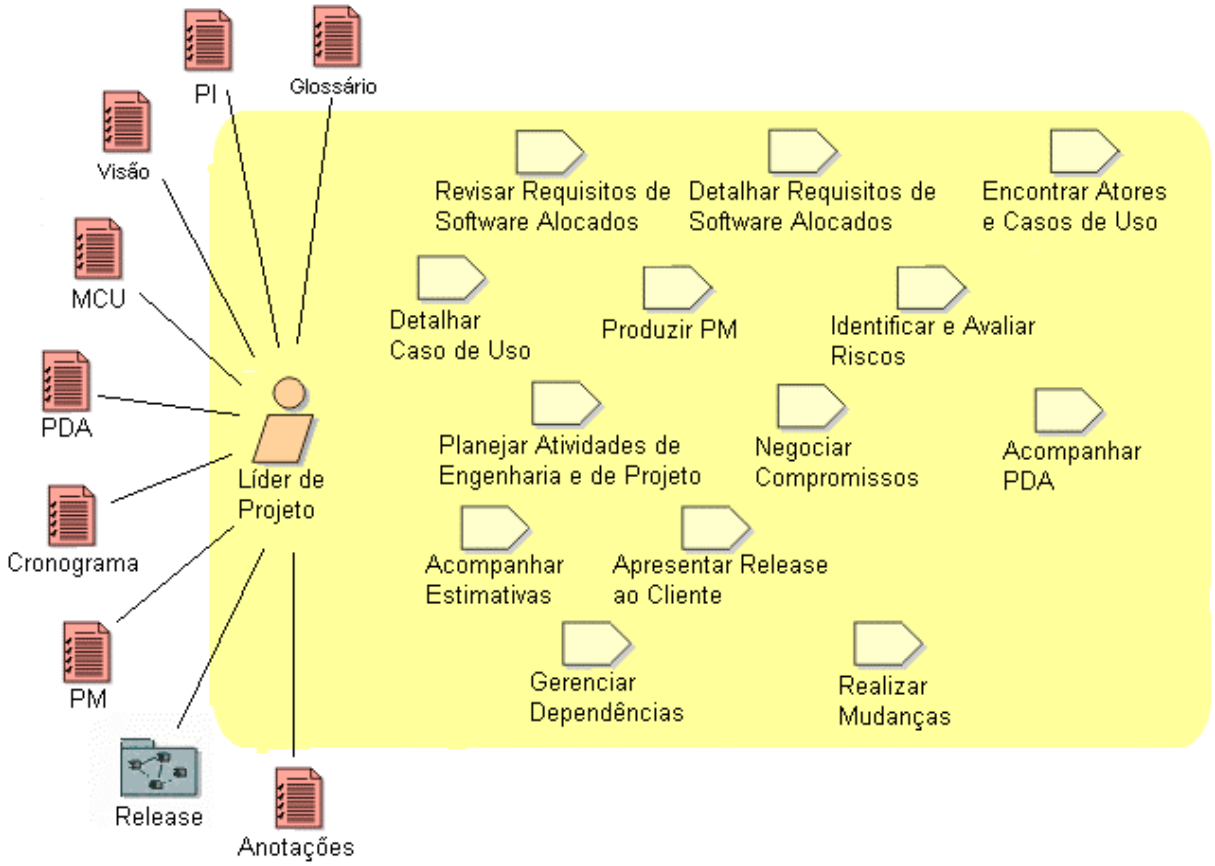

Figura 52 – Atividades exercidas pelo papel Líder de Projeto

3.6.7 Usuário Final

Para PAULK et al. (1999n, p. 356), o Usuário Final é uma pessoa ou um grupo de pessoas que irão utilizar o sistema de acordo com a sua finalidade operacional pretendida, assim que o sistema for entregue em seu ambiente de trabalho.

Responsabilidades do Usuário Final:

- Fornecer informações para que a *release* final ou de produto possa ser instalada em seu ambiente de trabalho.
- Realizar testes para que a *release* possa ser considerada aceita.

A Figura 53 ilustra as atividades alocadas a este papel e os artefatos utilizados e/ou produzidos.

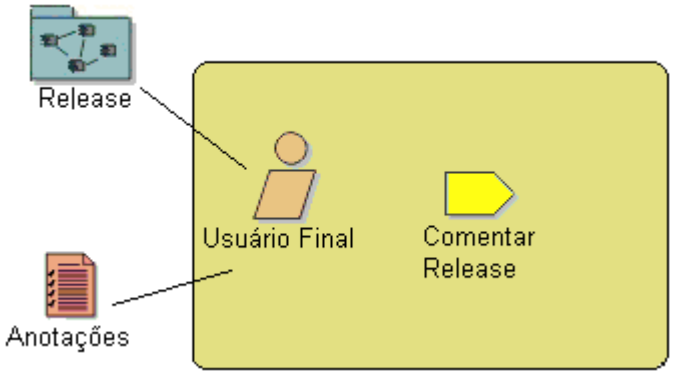

Figura 53 – Atividades exercidas pelo papel Usuário Final

3.7 Artefatos

Artefatos são produtos finalizados ou em evolução que são utilizados e produzidos durante o projeto e servem para capturar as informações do projeto. Um artefato pode ser:

- Um documento, tal como um plano de desenvolvimento de projeto.
- Um modelo, tal como um modelo de casos de uso.
- Um elemento pertencente a um modelo, tal como um caso de uso, um ator, uma classe etc.

Os artefatos encontram-se agrupados por disciplinas e para cada artefato, à exceção dos artefatos *Release* e Anotações, presente no LPA, existe um gabarito para auxiliar na sua construção. Os gabaritos do LPA podem ser consultados a partir do processo navegável.

A Figura 54 ilustra os artefatos existentes no LPA e o fluxo de informação existente entre eles.

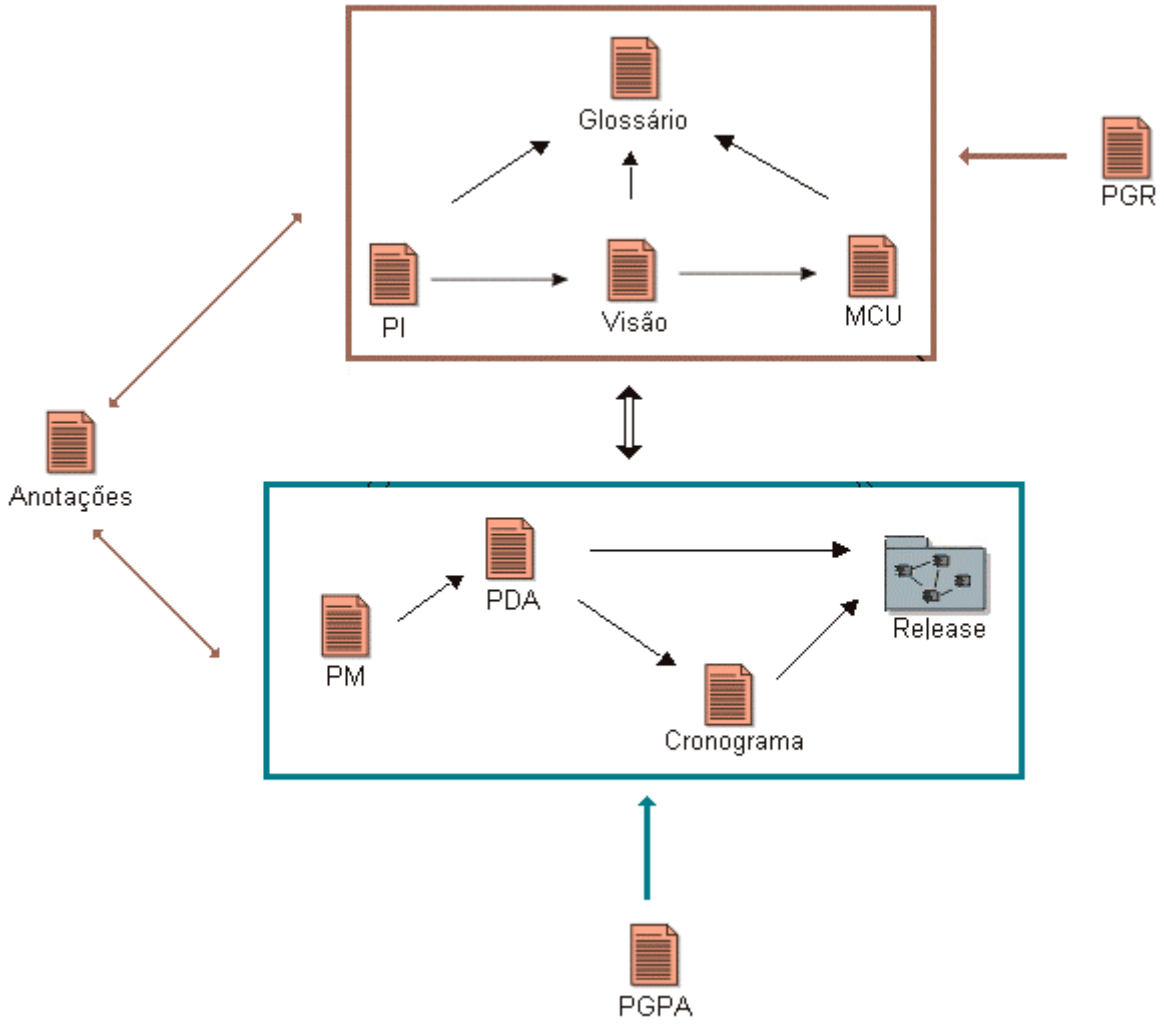

Figura 54 – Artefatos do LPA e fluxo de informações

3.7.1 Artefatos da Gestão de Requisitos

A FIGURA 55 ilustra os artefatos desta gestão, bem como os papéis envolvidos na sua utilização e/ou construção. Estes artefatos encontram-se detalhados nos demais itens.

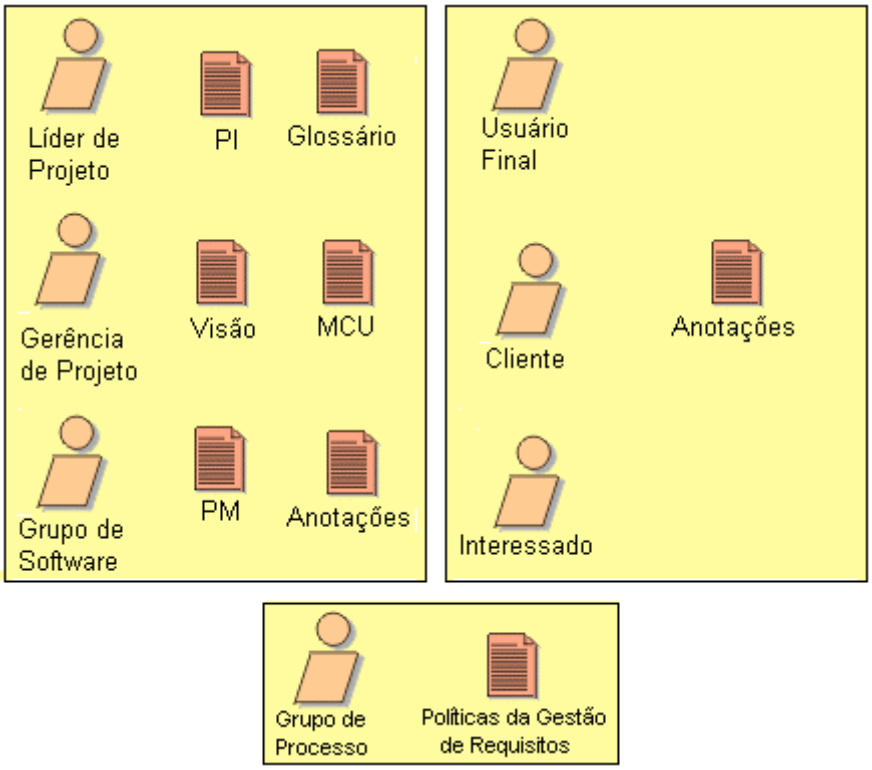

Figura 55 – Artefatos da Gestão de Requisitos

3.7.2 Políticas da Gestão de Requisitos – PGR

Finalidade

Descrever as regras necessárias para que os requisitos alocados ao *software* possam ser adequadamente gerenciados.[23]

Ocorrências

Esta política deve estar definida antes que um projeto se inicie, podendo ser mantida para todos os projetos da organização ou ajustada.

---

[23]Requisitos alocados ao *software* correspondem a um subconjunto dos requisitos do sistema, compreendendo apenas aqueles que podem ser implementados por *software*. Os requisitos alocados correspondem às entradas principais para o planejamento do desenvolvimento do *software*. A análise dos requisitos de *software* elabora e refina os requisitos alocados, cujo resultado são os requisitos de *software* que devem ser documentados e implementados.

Responsabilidade

O Grupo de Processo é responsável pela elaboração deste artefato, podendo envolver a Gerência de Projeto e o Líder de Projeto, se necessário.

3.7.3 Pedido do Interessado – PI[24]

Finalidade

A finalidade deste artefato é capturar todos os requisitos de um projeto e entender como a organização tem lidado com os problemas devido a estas necessidades. Embora a Gerência de Projeto seja responsável por este artefato, o Grupo de *Software*, ou apenas o Líder de Projeto e também outros Interessados pelo sistema podem contribuir para a análise e fornecimento de informações. As informações podem ser coletadas manualmente, usando-se o gabarito deste artefato, ou de forma automatizada. Os requisitos coletados devem ser acompanhados para verificar o andamento de sua implementação.

Alguns exemplos de fontes de informação para ajudar na coleta dos dados são:

- Resultados de entrevistas com Interessados.
- Resultados das sessões de levantamento de requisitos e de *workshops*.
- Pedidos de mudança.
- Declaração de trabalho.
- Pedido de proposta.
- Declaração de missão.
- Declaração do problema.
- Regras de negócios.
- Regulamentos e leis.
- Sistemas legados.
- Modelos de negócios.

[24] Artefato traduzido e adaptado do artefato "*Stakeholder Requests*" do RUP 2002.05.00.25.

Ocorrências

Este artefato deve ser coletado principalmente durante as fases de concepção e de elaboração. Todavia, ele deve continuar evoluindo nas demais fases do ciclo de vida para contribuir para o planejamento e atualizações do produto. Uma ferramenta de rastreamento de pedidos de mudança pode ser muito útil na coleta e priorização destes pedidos.

Responsabilidade

A Gerência de Projeto (como analista de sistema) é responsável pela integridade deste artefato, garantindo que:

- Foi dada a oportunidade para que todos os Interessados pudessem adicionar requisitos ao projeto.
- Todos os itens deste artefato sejam levados em consideração quando os requisitos forem alocados e detalhados por meio dos documentos Visão e MCU.

3.7.4 Visão[25]

Finalidade

Este artefato deve prover uma **base contratual** para que, a partir dos requisitos identificados, possam ser mais detalhados posteriormente. Ele captura os requisitos de alto nível e restrições de projeto para que o usuário deste documento possa ter um entendimento do sistema que deverá ser desenvolvido. Ele provê uma entrada para o processo de aprovação do projeto. Suas informações devem ficar concentradas nos "por quês e o quês" relacionados ao projeto e deve ser sempre levado em consideração para validar as decisões futuras durante o desenvolvimento.

Ele deverá ser consultado pelos Interessados, Líder de Projeto e Grupo de *Software*.

Ocorrências

Deve ser criado no início da fase de concepção e usado com base para se identificar os riscos do projeto e para a construção do MCU.

[25] Artefato traduzido e adaptado do artefato "*Vision*" do RUP - 2002.05.00.25.

Responsabilidade

A Gerência de Projeto é responsável pela integridade deste artefato, garantindo que seja sempre atualizado e distribuído. Ele pode ainda ser atualizado pelo Líder de Projeto.

Informações Adicionais

Este documento pode ser alterado na medida em que os requisitos, arquitetura, planos e tecnologia evoluem com o ciclo de vida. Todavia, ele deve ser alterado lentamente e normalmente no início de cada fase do ciclo de vida.

3.7.5 Glossário[26]

Finalidade

Este artefato deve conter todos os termos técnicos a serem utilizados no projeto, com a finalidade de evitar que aconteçam interpretações incorretas a respeito dos seus significados, bem como ambigüidades. Assim como os demais artefatos do LPA, este também deve ser construído gradativamente.

Ocorrências

Sua construção deverá ser feita nas fases iniciais do projeto (concepção e elaboração).

Responsabilidade

Embora outras pessoas possam contribuir para a construção deste artefato, o responsável pela sua construção e integridade é a Gerência de Projeto.

3.7.6 Modelo de Casos de Uso – MCU

Finalidade

Este artefato pode ser considerado o elo entre o Cliente e o time de desenvolvimento. Ele transforma os requisitos do sistema que foram obtidos no artefato Visão em atores, casos de uso, diagrama de casos de uso e especificações suplementares que correspondem aos requisitos não-funcionais do sistema. É por meio da descrição dos fluxos de eventos dos casos de uso que o Grupo de *Software* inicia a codificação do sistema.

---

[26] Artefato traduzido e adaptado do artefato *"Glossary"* do RUP - 2002.05.00.25.

Além disto, para que os requisitos do sistema possam ser rastreadas para verificação e validação, este artefato possui uma tabela de mapeamento dos requisitos funcionais e não-funcionais em casos de uso.

Assim que o diagrama de casos de uso estiver pronto, deverá ser apresentado ao cliente para validação de suas necessidades. Antes disto, o mesmo deverá receber um treinamento rápido do significado dos elementos e função deste diagrama.

Este artefato deve se manter íntegro a todas as solicitações de melhorias e mudanças que vierem a ocorrer nos requisitos do sistema. Para isto, sua atualização deve ocorrer tendo-se como entrada o artefato Visão.

Uma grande característica deste artefato, no que se refere aos casos de uso, é que ele deve ser utilizado pela equipe de teste para a criação dos casos de teste. Estes casos de teste deverão ser aplicados nas atividades de teste de sistema.

Este artefato poderá ser utilizado pelo:

- Líder de Projeto - para a transformação dos requisitos do sistema em um modelo de casos de uso e em especificações suplementares.
- Grupo de *Software* - para a implementação do sistema.
- Cliente - para validar suas necessidades funcionais.
- Testadores - para criação dos casos de teste e aplicação nos testes de sistema.
- Gerência de Projeto - para verificação de conteúdo.

Ocorrências

Este artefato deve ser criado inicialmente durante a fase de concepção e refinado durante as fases de elaboração e construção.

Responsabilidade

O Líder de Projeto é a pessoa responsável pela criação e manutenção deste artefato, embora outros possam proceder à sua alteração sob seu conhecimento e validação.

3.7.7 Pedido de Mudança – PM[27]

Finalidade

Manter os dados referentes a mudanças de forma documentada e para que sua evolução possa ser acompanhada durante a análise e implementação, devendo ser mantido sob controle, quer de forma manual ou automática, por meio de uma ferramenta de rastreamento.

Quando o pedido se referir a uma melhoria do sistema, deverá ser usado pelo Líder de Projeto para determinar os requisitos futuros a serem incluídos no produto. Neste caso, deve ser usado como entrada para as alterações a serem feitas no PI, para que as novas necessidades possam ser entendidas.

Quando o pedido se referir a um defeito, significará uma anomalia ou falha em algum produto de trabalho entregue. Defeitos incluem coisas do tipo: omissões ou imperfeições encontradas durante as fases iniciais do ciclo de vida e falhas contidas no *software* durante sua operação ou teste. Os defeitos também podem incluir desvios de expectativas ou outros problemas que precisam ser rastreados e resolvidos.

O PM deve ser utilizado para comunicar detalhes de um problema, disparar ações corretivas, resolver o problema e acompanhar a evolução de sua análise e implementação das mudanças.

Análise de um PM

Os seguintes atributos devem ser levados em consideração na análise e tomada de decisão de um PM pendente:

- Quantidade de trabalho a ser modificado.
- Quantidade de trabalho a ser adicionado.
- Existem outras alternativas sem envolver necessariamente a mudança?
- A mudança proposta será fácil de ser implementada?
- Quais são as conseqüências que poderão surgir após a implementação da mudança?
- Qual é o impacto ou conseqüência se a mudança não for implementada?
- A mudança é na verdade um pedido de melhoria?

---

[27] Artefato traduzido e adaptado do artefato: "*Change Request*" do RUP - 2002.05.00.25.

- Quando a mudança deverá estar finalizada?
- Ela é possível de ser implementada?
- Qual é o custo ou economia de se fazer a mudança?
- Existem outras mudanças que poderão substituir ou invalidar esta mudança?
- Existem requisitos especiais para que a mudança possa ser testada?

Ocorrências

Um PM pode ser submetido a qualquer momento durante o ciclo de vida.

A principal fonte de identificação de defeitos são os resultados de testes de integração de sistema e de desempenho. Todavia, um defeito pode surgir a qualquer momento do ciclo de vida de desenvolvimento, como por exemplo: casos de uso, casos de teste ou documentações incompletas ou ausentes.

Responsabilidade

Qualquer pessoa pode solicitar a abertura de um PM, sendo que a responsabilidade pela coleta e preenchimento é do Líder de Projeto. Este, juntamente com o Grupo de *Software*, faz a avaliação da solicitação e encaminha para aprovação pela Gerência de Projeto que, em caso de aprovação, irá providenciar recursos financeiros para que a mudança possa ser implementada.

3.7.8 Anotações

Finalidade

Este artefato pode ser considerado qualquer registro que seja feito durante o desenvolvimento, podendo ter o caráter informal ou formal, dependendo de seu conteúdo.

Uma anotação pode ser considerada uma ata de reunião, um comunicado por email, um arquivo-texto, um questionário de uma entrevista, memorandos, descrição de uma *release*, descrição de ambiente ou de um processo de instalação, uma carta, um formulário preenchido via Internet etc.

Ocorrências

Sempre, durante as ocorrências de reuniões eventuais ou periódicas de acompanhamento do projeto, nas descrições das *releases* internas ou externas, nas formalizações de compromissos externos e sempre que a necessidade momentânea do projeto exigir.

Responsabilidade

O Líder de Projeto é o papel responsável pela criação das anotações por ser ele o que se envolve mais com as atividades do projeto. Todavia, qualquer integrante do projeto pode fazer uso deste artefato, inclusive os Usuários Finais, Cliente e outros Interessados. Sempre que uma anotação for realizada por outro papel que não seja o Líder de Projeto, este deverá ter ciência de seu conteúdo.

3.8 Artefatos da Gestão de Planejamento e Acompanhamento

A FIGURA 56 ilustra os artefatos desta gestão, bem como os papéis envolvidos na sua utilização e/ou construção. Estes artefatos encontram-se detalhados nos demais itens.

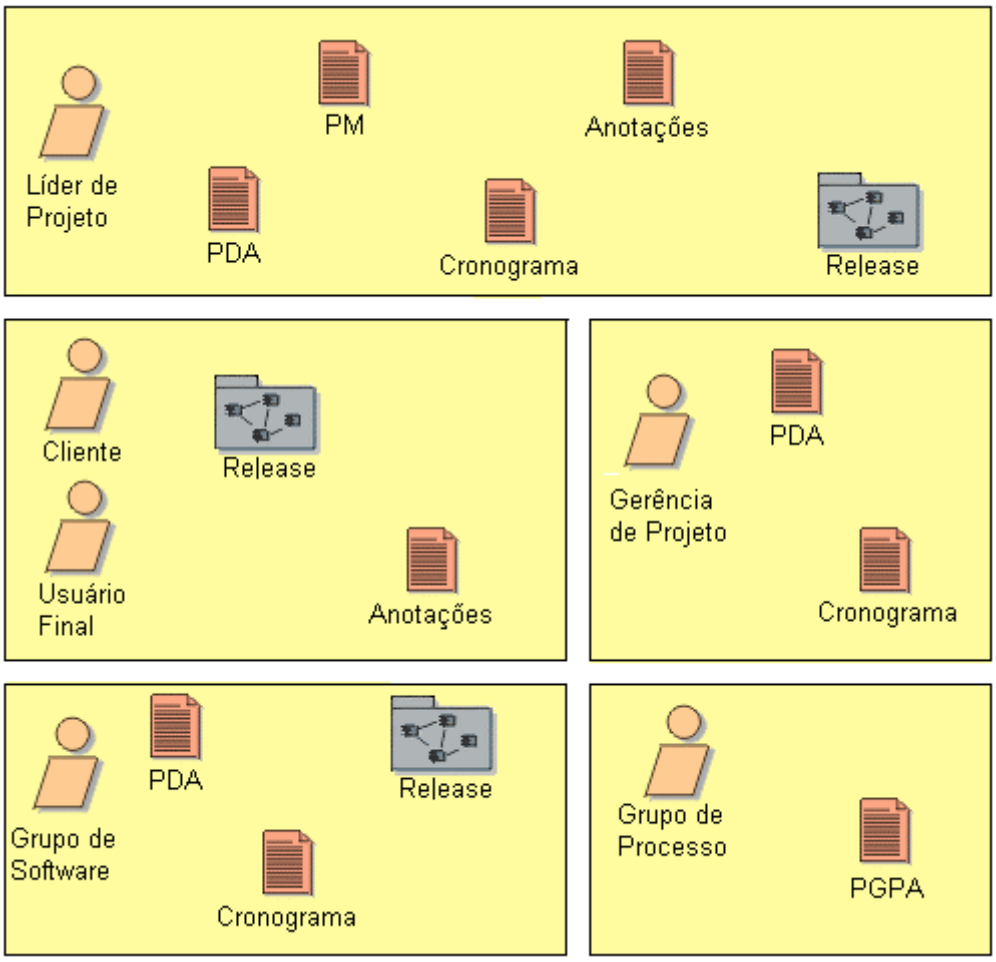

Figura 56 – Artefatos da Gestão de Requisitos

3.8.1 Políticas da Gestão de Planejamento e Acompanhamento – PGPA

Finalidade

Descrever as regras necessárias para que os requisitos alocados ao *software* possam ser adequadamente gerenciados.

Ocorrências

Esta política deve estar definida antes que um projeto se inicie, podendo ser mantida para todos os projetos da organização ou ajustada.

Responsabilidade

O Grupo de Processo é responsável pela elaboração deste artefato.

3.8.2 Plano de Desenvolvimento e Acompanhamento – PDA

Finalidade

A finalidade do PDA é reunir todas as informações necessárias para o controle do projeto. Ele descreve a abordagem a ser seguida durante o desenvolvimento.

Ocorrências

Desenvolvido durante a fase de concepção, e completado e atualizado nas demais fases.

Responsabilidade

O responsável por manter este artefato é o Líder de Projeto, embora a Gerência de Projeto inicie a sua construção. O Grupo de *Software* utiliza este artefato para identificar as responsabilidades assumidas no projeto.

Informações Adicionais

Um bom desenvolvimento de *software* envolve um PDA que seja útil e atualizado periodicamente, e entendido e compromissado entre todos os integrantes do desenvolvimento.

Ele deve conter, dentre outras informações:

- Estrutura organizacional para o projeto.
- Interfaces com pessoas e grupos externos ao projeto.
- Papéis e responsabilidades assumidas durante o projeto.

- Estimativas de tamanho do projeto, recursos humanos, esforço, custo e Cronograma para desenvolvimento e medidas para acompanhamento.
- Marcos principais e datas de alcance.
- *Releases* a serem produzidas.
- Gestão de riscos.
- Acompanhamento de *status* do desenvolvimento.

3.8.3 Cronograma

Finalidade

Este artefato é uma parte do PDA, sendo que, juntos, permitirão a gerência do projeto.

Por meio do Cronograma, poderão ser definidas tarefas e os recursos necessários para o desenvolvimento do projeto dentro de limitações de tempo e custo.

Ocorrências

O Cronograma deverá ser criado na fase de concepção, onde deverão ser identificados os marcos de cada fase, datas de início e fim estimadas, recursos e custos envolvidos, além do detalhamento da fase de concepção. No início das demais fases, as atividades da fase corrente deverão ser detalhadas e as informações anteriores deverão ser atualizadas para acompanhamento do projeto. Além disto, este artefato deverá ser consultado e poderá ser atualizado a qualquer momento.

Responsabilidade

O Líder de Projeto é o responsável pela manutenção deste artefato. Todavia, sua criação encontra-se ligada diretamente à Gerência de Projeto, que definirá os marcos principais, recursos e custos iniciais do projeto.

O Grupo de *Software* deve ter ciência do conteúdo deste artefato e consultá-lo de forma sistemática.

Outras Informações

O Cronograma pode ser construído de uma forma bem simplificada, usando-se folhas de papel ou um editor de textos, com identificação das tarefas a serem realizadas, quem irá realizá-las e

as datas de início e fim de cada uma. Uma forma mais completa seria a utilização de ferramentas de *software* específicas.

Este artefato e o PDA formam os principais artefatos para se gerenciar o projeto, devendo ser consultado, acompanhado e atualizado de forma consistente e compromissada.

3.8.4 *Release*

Finalidade

Consiste de uma versão executável, estável e testada do sistema, devendo ser entregue com uma determinada capacidade operacional do mesmo, ou seja, uma *release* não precisa necessariamente conter toda a capacidade operacional prevista no produto final. Caso a *release* contenha toda a capacidade operacional desejada pelo Cliente para o produto final, deverá ser chamada de *release* de produto ou ainda *release* final.

Ocorrências

Releases internas podem ser geradas a qualquer tempo do desenvolvimento.

Releases externas normalmente são geradas nos finais das iterações ou das fases do ciclo de vida.

Na fase de concepção, normalmente são gerados protótipos para a redução de riscos de projeto e uma *release* externa ao término desta fase contendo uma cobertura parcial do escopo do projeto.

Na fase de elaboração devem ser geradas uma ou mais *releases* externas que deverão consolidar a arquitetura prevista para o projeto.

Na fase de construção deverão ser geradas uma ou mais *releases* externas que conterão certas capacidades operacionais previstas, sendo que a última *release* desta fase deve ser a Versão Beta, a ser entregue para teste formal pelos Usuários Finais. A Versão Beta deverá conter praticamente todas as funcionalidades previstas.

Na fase de transição costuma-se fornecer apenas uma *release* externa com os devidos ajustes e correções identificadas nos testes da Versão Beta realizados pelos Usuários Finais.

Responsabilidade

O Grupo de *Software* é o responsável pela geração das *releases* sob a coordenação do Líder de Projeto.

Outras Informações

Uma *release* pode ser interna ou externa. A diferença entre as duas é que a *release* externa deverá ser formalmente entregue para um ou mais Usuários Finais do sistema. Estes usuários deverão testar a *release* validando sua capacidade operacional de acordo com um ou mais requisitos do projeto. Após os testes, os resultados deverão ser reportados para o Cliente, que deverá avaliar os resultados, dando o aceite ou não. Desta forma, uma *release* externa pode fazer parte de marcos do Cronograma, e de acordo com cláusulas contratuais, podendo ainda estar vinculada a pagamentos para a organização desenvolvedora.

Tanto uma *release* interna quanto uma externa devem vir acompanhadas de um arquivo de Anotações contendo: a descrição da *release*, suas limitações, abrangência e descrição de ambiente e de instalação.

Tanto uma *release* interna quanto externa, devem ser previstas durante o planejamento do projeto no PDA e agendadas no Cronograma. As *releases* externas costumam ser liberadas em marcos do ciclo de vida do projeto, normalmente nos finais de fase.

3.8.5 Anotações

Este artefato foi descrito no item 3.7.8.

3.9 Conclusão

A gerência de um processo pode ser realizada com a utilização do processo LPA segundo as descrições da gerência de requisitos, gerência de planejamento e de acompanhamento, que cobrem 50% do nível 2 do modelo SW-CMM em termos de quantidades de KPAs. Estas descrições, além de textuais, possuem um forte apelo visual por meio da utilização de ícones e digramas de atividades que procuram facilitar a utilização do processo. Como o processo pode ser executado em uma ferramenta de navegação da Internet, sua execução se torna altamente amigável, motivadora e dinâmica, devido aos inúmeros caminhos que podem ser tomados de acordo com seus links de navegação.

4 APLICAÇÃO E ANÁLISE DOS RESULTADOS

Neste capítulo são apresentados os critérios adotados na aplicação do processo na Sessão de Software Administrativo – SSA do Inatel, a aplicação do processo, a coleta dos dados durante as atividades e uma análise dos resultados produzidos.

4.1 Critérios Adotados

A SSA é composta de 5 funcionários do Inatel, responsáveis pela produção dos *softwares* administrativos utilizados pela instituição. O LPA foi aplicado na sua íntegra em um projeto com a duração estimada de 6 meses, com uma carga horária semanal de 20h00 de cada integrante da equipe. O projeto era fortemente orientado a dados e desenvolvido em plataforma Delphi. A equipe foi dividida de acordo com os papéis determinados pelo LPA.

Os trabalhos foram aplicados sob a forma de consultoria, com a realização de reuniões semanais de 2h00 cada uma, ou de 2 reuniões semanais de 1h00 cada. A cada reunião de consultoria, eram coletadas as horas de dedicação do pessoal com a aplicação do processo ao projeto, e ao final eram determinadas as atividades que as pessoas deveriam desempenhar durante a semana.

O processo foi aplicado com autorização e apoio da alta gestão da instituição.

4.2 Aplicação do LPA

No dia 18 de março de 2003, realizou-se a primeira reunião de consultoria, quando foi feito um levantamento inicial do perfil da organização e apresentados os objetivos da consultoria. Em seguida e nas próximas reuniões, iniciou-se a aplicação de uma forte carga de treinamento para a organização. Esta forte carga não se refere especificamente ao tempo consumido com os treinamentos, mas sim à grande quantidade de conceitos importantes administrados da engenharia de software.

Foram abordados os seguintes tópicos:

- As necessidades de um processo de desenvolvimento de *software*. Foram apresentadas as importâncias de se ter um processo, as vantagens e benefícios resultantes de seu uso.
- Visão geral do SW-CMM. Como o LPA é parcialmente aderente ao modelo SW-CMM nível 2, fez-se uma rápida apresentação do modelo e sua importância.
- O modelo de ciclo de vida espiral. Como um dos quesitos do SW-CMM nível 2 é a definição do ciclo de vida de desenvolvimento, foi apresentado então o modelo do ciclo de vida espiral, servindo de ponte para a apresentação do ciclo de vida iterativo e incremental utilizado pelo LPA.
- Visão geral sobre o Processo Unificado Rational – RUP. Como o LPA foi construído baseado no RUP, foram apresentadas as características principais deste processo, tendo-se como alvo a sua estrutura bidimensional entre o tempo (fases) e as atividades (disciplinas) a serem seguidas durante o desenvolvimento de um projeto de *software*.
- Visão geral do processo LPA. Foi apresentado o LPA, seu ciclo de vida, suas disciplinas gerenciais de requisitos, de planejamento e acompanhamento, os diagramas das macroatividades destas disciplinas e os diagramas contidos em cada macroatividade. Foram mostrados os gabaritos PGR e PGPA que cobrem as políticas a serem seguidas pela organização para as duas disciplinas do LPA.
- Classificação de requisitos. Como base para o entendimento da disciplina da gestão de requisitos, foi apresentada uma abordagem de requisitos, bem como as formas de classificação entre estruturada e orientada a objetos. Durante a abordagem deste tópico, frisou-se a importância de se gerenciar as mudanças de requisitos que normalmente acontecem durante um projeto. Para apoiar as mudanças de requisitos, foi apresentado o artefato PM que agrupa as mudanças de requisitos vindas de algum solicitante.
- Modelo de Casos de Uso. Como o LPA segue a notação UML para o desenvolvimento dos requisitos do projeto, foram abordados os conceitos de atores, casos de uso, relacionamentos entre atores e casos de uso e a construção de diagramas de casos de uso. Completou-se este tópico com a descrição dos fluxos de eventos contidos nos cenários existentes de um caso de uso. Procurou-se frisar o cuidado necessário para não confundir os conceitos de casos de uso com uma prática muito comum, porém incorreta, de se construir um modelo de casos de uso segundo uma decomposição funcional.

- Como o LPA pode ajudar na coleta e desenvolvimento dos requisitos. Foram apresentados as facilidades do LPA para a gestão de requisitos de um projeto. Foram descritas as atividades cobertas por esta gestão, bem como as finalidades dos gabaritos do LPA Pedido do Interessado, Visão e Modelo de Casos de Uso.
- Noções de planejamento de projeto. Visando atender à disciplina da gestão de planejamento e acompanhamento de projeto do LPA, foram abordadas as necessidades de se estimar, planejar e realizar medições do projeto, para que se possa ir acumulando informações históricas, para análises e melhorias de estimativas e planejamentos em projetos futuros. Abordaram-se questões de identificação de riscos do projeto, planos de redução e de contingência para os riscos, uma metodologia de estimativa de tamanho do projeto e, por conseguinte, a sua derivação em estimativas de esforço (homem-horas), cronograma (dias) e custo do mesmo (R$). Foram apresentados ao final os gabaritos PDA e o Cronograma que cobrem a disciplina da gestão de planejamento e acompanhamento de projeto.
- Controle de versão e de mudança. Embora o LPA não dê suporte à gerência de configuração de um projeto, abordou-se o conceito de itens de configuração e de *baseline*, em virtude de a organização utilizar controles de versão de seus artefatos produzidos durante o desenvolvimento de um projeto. Para o controle de mudanças, voltou-se a comentar que o LPA poderia contribuir com o artefato PM, que poderia capturar qualquer necessidade de mudança solicitada por algum interessado do projeto.

Com o término das atividades de treinamento, iniciou-se a utilização do LPA em um processo típico da organização.

Na reunião do dia 25 de abril de 2003, definiu-se o projeto que seria utilizado no LPA, e na reunião seguinte, realizada no dia 02 de maio, iniciaram-se os trabalhos de coleta de informações junto ao cliente para o entendimento de suas necessidades. Em paralelo, a gerência de projeto fez um planejamento inicial do mesmo, incluindo os recursos a serem utilizados e as estimativas iniciais. Este projeto recebeu uma estimativa de esforço de 1620 homem-horas e 91 dias úteis, envolvendo um time de 4 a 5 pessoas.

Nas reuniões que se sucederam, definiram-se as atividades de desenvolvimento do projeto e fez-se o acompanhamento de sua execução. Eventualmente, novos treinamentos acabavam

acontecendo, em virtude de alguma dificuldade na aplicação dos conceitos utilizados anteriormente.

De posse das informações do cliente para o desenvolvimento do projeto, iniciou-se a elaboração dos artefatos PI, em seguida o Visão e por último o MCU. Em paralelo, o projeto continuou com seu planejamento, onde foi possível decompô-lo nas fases de concepção, elaboração, construção e transição, definição de prazos para cada uma dessas fases e as pessoas envolvidas. Fez-se também um detalhamento das atividades da fase de concepção e um pouco da fase de elaboração.

Na medida em que a equipe ia construindo o modelo de casos de uso (atores, casos de uso, diagramas de casos de uso e fluxos de eventos), faziam-se revisões dos trabalhos e novos treinamentos para dirimir dúvidas existentes.

Durante a aplicação do processo, solicitou-se que as pessoas fizessem comentários sobre o processo e seus gabaritos, para que o mesmo pudesse ser melhor ajustado às características da organização. Dentro desta linha, o LPA ia sofrendo atualizações constantes para atender às necessidades da organização.

Em um dado momento do desenvolvimento dos fluxos de eventos de alguns casos de uso, perceberam-se dificuldades da equipe em desenvolver estes fluxos de eventos sob a forma de tabelas e descrições similares a um algoritmo de codificação, conforme encontra-se previsto no gabarito MCU. Partiu-se então para a adoção de uma estratégia simplificada de descrição, feita sob a forma de uma descrição puramente textual, o que proporcionou maior eficiência à equipe.

As atividades de desenvolvimento da gestão de requisitos eram sempre realizadas de forma conjunta, o que facilitava o processo de revisão previsto pelo LPA.

Na reunião do dia 12 de junho de 2003, surgiu a necessidade de uma mudança de requisito que poderia comprometer a entrega final prevista para o produto. Foi possível então avaliar o artefato PM do LPA e das atividades da macroatividade Gerenciar e Controlar Mudanças nos Requisitos.

No dia 08 de agosto de 2003, foi apresentada internamente a primeira *release* do projeto. Esta *release* continha algumas funcionalidades de alguns casos de uso do sistema; em seguida foram definidas as novas funcionalidades a serem incluídas na próxima *release*.

No dia 12 de agosto de 2003, finalizou-se a fase de elaboração do desenvolvimento, com a apresentação de uma *release* para o cliente. Esta apresentação foi importante para o time de desenvolvimento, pois pôde-se perceber o envolvimento do cliente com o projeto. Durante a apresentação, o cliente teceu alguns comentários, sendo que um deles foi caracterizado como sendo um requisito adicional, que precisou ser tratado adequadamente conforme previsto no LPA.[28] Além disto, fez-se uma reunião de final de fase, onde o Líder de Projeto apresentou a situação atual do projeto, e a Gerência de Projeto pôde fazer uma verificação do andamento do mesmo.

Devido ao fato de as consultorias inicialmente programadas terem sido consumidas, não foi possível para a equipe concluir o projeto, restando as fases de construção e de transição. Apesar disto, considera-se que o LPA passou por várias situações que permitiram sua avaliação, de acordo com os objetivos inicialmente traçados para este trabalho.

4.3 Coleta de Informações

A coleta das informações para a análise e avaliação do LPA foi feita de forma interativa: durante as reuniões periódicas de aplicação do processo, pelas atas das reuniões de trabalho e por meio de questionários de avaliação do processo. Dos questionários aplicados, um foi subjetivo, composto de 14 perguntas, e outro objetivo, composto de 15 perguntas. Estes questionários podem ser observados no Apêndice B – Questionário de Avaliação do LPA.

Todas estas informações coletadas aconteceram no período de 18 de março de 2003 a 12 de setembro de 2003, perfazendo 6 meses aproximadamente. Durante esse período, realizaram-se 20 reuniões de consultoria, perfazendo um total de 32h15 consumidas com as reuniões, 377h00 consumidas com as atividades de desenvolvimento (para as fases de concepção e elaboração) e produziram-se 20 atas.

[28] Para maiores detalhes, consulte a macroatividade Gerenciar e Controlar Mudanças nos Requisitos, descrita no item 3.4.3

4.4 Resultados

Das reuniões de consultorias e pelos registros produzidos nas atas destas reuniões, identificaram-se as dificuldades e facilidades encontradas durante os trabalhos conforme descritas.

Dificuldades encontradas:

- Baixo grau de maturidade da organização para com processos de desenvolvimento de *software*. Embora isto tivesse sido previsto no início, foi interessante perceber esta realidade durante o andamento das reuniões. Percebeu-se que a organização possuía grandes habilidades com o desenvolvimento de *software*, mas muito pouco com relação à gerência de requisitos e de projeto.
- Inércia inicial. Observaram-se dificuldades na absorção dos conhecimentos que eram transmitidos durante as reuniões de consultoria devido ao número elevado de novidades que eram apresentadas. Conceitos como: UML, RUP, diagramas de casos de uso e fluxos de eventos dos casos de uso, volatilidade de requisitos, rastreamento de requisitos, ciclos de vida de desenvolvimento, estimativas de tamanho, de esforço, de custo e de riscos, acompanhamento de projeto e *releases* gradativas, eram tidos como novos ou muito pouco conhecidos pela organização. Isto veio a comprometer o tempo de entendimento do processo pela organização e por conseguinte a sua utilização, uma vez que ele utiliza estes conceitos.
- Tendência estruturada. Observaram-se dificuldades da equipe em produzir casos de uso conceitualmente corretos e também realizar a descrição dos fluxos de eventos. Percebeu-se uma forte tendência de se fazer uma decomposição funcional dos casos de uso.
- *Release* única. Perceberam-se dificuldades de se pôr em prática a produção gradativa de *releases*, embora a equipe tenha percebido a validade e a importância desta estratégia. Por meio de *releases* gradativas, as funcionalidades vão surgindo aos poucos e o cliente interage com a equipe de forma mais constante, o que permite que sejam feitos tratamentos adequados das necessidades do cliente e das mudanças sugeridas de forma gradativa.
- Elevado índice de interrupções. Devido ao fato de a organização possuir uma carga horária dividida entre atividades de manutenção (20h00 semanais) e de desenvolvimento (20h00

semanais), a equipe era constantemente solicitada a utilizar, em parte ou no todo, a carga horária dedicada ao desenvolvimento em atividades de manutenção. Isto também provocou dificuldades na implantação do processo. Vale ressaltar que o LPA foi aplicado apenas nas atividades de desenvolvimento desta organização.

Facilidades encontradas:

- Grande interesse no aprendizado e utilização do LPA. Percebeu-se desde o início o grande interesse de toda a organização, o que de certa forma permitiu superar em parte as dificuldades citadas anteriormente.
- Apoio e dedicação da equipe. Percebeu-se também alto grau de dedicação da organização na participação ativa às reuniões e nos questionamentos feitos durante a aplicação do processo.

Ao final das atividades de consultoria, aplicou-se um questionário a todos os membros da organização. Este questionário foi dividido em uma parte subjetiva e outra objetiva. A parte subjetiva foi composta de 14 perguntas abertas. A parte objetiva foi composta de 16 perguntas de múltipla escolha, com as possíveis respostas: “não”, “+/-“, “sim” ou “não foi possível avaliar”.

Do questionário subjetivo aplicado aos 5 integrantes da organização, fez-se uma compilação das respostas, sendo elas transcritas conforme a síntese descrita a seguir. Em algumas respostas foram descritas as ações tomadas para contornar as dificuldades encontradas.

Perguntas subjetivas e síntese das respostas:

1. O Líder de Projeto e a Gerência de Projeto tiveram maior controle do projeto?

A maioria das pessoas respondeu que sim. Toda a equipe ficou envolvida com o andamento do projeto. A gerência não ficou perdida com o andamento do projeto, pois o cronograma facilitou o acompanhamento. Os problemas e as falhas foram tratados de forma mais controlada. O fato de as informações não ficarem na cabeça de apenas uma pessoa permitiu maior controle do mesmo. Uma minoria respondeu que teve a impressão de que o projeto estava sendo melhor controlado, mas que faltaram informações para poder fazer uma avaliação melhor.

2. Os cronogramas estão sendo atendidos adequadamente?

De forma geral o cronograma não estava sendo atendido. A equipe sofreu durante o período de consultoria uma forte demanda por atividades de manutenção, o que provocava constantes replanejamentos do projeto.

3. O tempo gasto com o uso do processo (*overhead*) comprometeu o desenvolvimento do projeto?

Houve um peso maior no início de aplicação do processo devido ao desconhecimento do mesmo. Poucas pessoas consideraram que o processo provoca um *overhead* no tempo de desenvolvimento devido à documentação do projeto, mas possuíam plena consciência da necessidade disto, visando colaborar para minimizar os impactos provocados durante a fase de manutenção do projeto. Com uma documentação adequada, a manutenção fica facilitada e outras pessoas, que não participaram do desenvolvimento do projeto, puderam prestar a manutenção com maior eficiência. Uma minoria declarou não ter informações suficientes para responder a esta pergunta.

4. O processo foi seguido pela organização?

Todos responderam que sim. Além de ser seguido, eles possuíam o aval da alta direção da instituição.

5. Os projetos estão sendo melhor planejados?

Como o LPA foi aplicado em apenas um projeto, não se pôde avaliar a sua eficiência em outros projetos. Todavia, todos declararam que o projeto em que foi aplicado o LPA teve um planejamento melhor que nos projetos do passado sem a utilização do LPA.

6. Qual foi o grau de facilidade no entendimento do processo?

De forma geral as pessoas consideraram que o LPA teve um grau de facilidade médio para o entendimento do mesmo. As nomenclaturas utilizadas no LPA e os novos conceitos introduzidos contribuíram na dificuldade de entendimento do mesmo.

7. **O cliente percebeu maior qualidade do serviço prestado e ficou mais satisfeito com os resultados?**

A maioria percebeu que, não apenas a organização assumiu responsabilidades para com o desenvolvimento, mas também o cliente, e ele percebeu esta característica, o que lhe transmitiu uma imagem positiva quanto à qualidade do trabalho.

Quanto à satisfação do cliente para com os resultados do projeto, ainda não foi possível avaliar, tendo-se em vista que foi entregue apenas uma *release* do projeto, e que não contemplava muitas funcionalidades.

8. **Qual foi a quantidade de replanejamentos produzidos?**

Aconteceram replanejamentos em virtude de mudanças de requisitos e da alocação da equipe em outras atividades. Algumas pessoas declararam não terem informações suficientes para avaliar esta questão.

Isto se deveu ao fato de que nem todas pessoas participaram de perto das atividades de planejamento do projeto.

9. **Ocorreram muitas mudanças nos requisitos do sistema em desenvolvimento?**

A maioria das pessoas afirmou terem ocorrido algumas ou muitas mudanças nos requisitos, em virtude de um novo levantamento das necessidades do usuário e das alocações de tempo das pessoas nas atividades de manutenção de outros projetos. Uma minoria declarou não ter informações suficientes para responder a esta questão e outra declarou que não aconteceram mudanças.

10. **Houve muita resistência das pessoas durante o processo de implantação do LPA? Quais?**

A maioria das pessoas declarou que houve uma certa resistência na utilização do processo, principalmente no início de sua utilização, o que foi diminuindo com o tempo. O consumo de tempo com preenchimento de documentos foi um dos fatores de resistência devido ao costume de se sair codificando. Todavia, têm-se a consciência de que este tempo consumido se torna compensador nos momentos de codificação e de manutenção, devido à melhoria de eficiência. Outro fator de resistência foi a dificuldade de identificação dos casos de uso e da descrição dos fluxos de eventos. Neste caso, teve-se que adotar uma estratégia diferente para a descrição dos fluxos de eventos, fazendo-se de forma puramente textual, o que trouxe maior conforto ao

desenvolvedor e melhoria da eficiência. Existiu resistência, mas ela se transformou em força motivadora dada a consciência da necessidade de um processo definido para a organização. Uma minoria respondeu que não houve resistências.

11. O processo é muito burocrático?

A maioria das pessoas respondeu que o LPA é um pouco burocrático, pois as informações parecem se repetir em vários lugares. Uma minoria respondeu que no início parecia ser burocrático, mas que ao longo de sua utilização cada documento mostrou a sua razão de ser, e uma outra minoria respondeu que não é burocrático.

12. Você acha que deve continuar utilizando o processo?

Todos responderam que sim. Estavam fazendo conforme aprenderam na escola. O processo tinha sido de grande valor. Os ganhos na época de manutenção provavelmente serão enormes. Facilita o repasse de informações para outras pessoas. Trata-se de um caminho sem volta.

13. Você acha que o processo precisa ser alterado? Onde?

A maioria das pessoas respondeu que as mudanças necessárias foram sugeridas durante a aplicação do mesmo, e o processo foi alterado sem contudo afetar as exigências do modelo SW-CMM. Uma minoria respondeu que não percebe até o momento qualquer necessidade de mudança no LPA.

14. O que mais lhe agradou no processo?

Todas as pessoas opinaram de alguma forma. A documentação possui a sua importância. A forma como as informações são organizadas. O glossário facilita os trabalhos, pois elimina ambigüidades e evita interpretações incorretas no futuro. O fato de o diagrama de casos de uso oferecer uma visão ampla do projeto como um todo. O envolvimento do cliente no processo. A apresentação das *releases* de forma iterativa e incremental. Facilidade e rapidez de se transmitir as informações do projeto para outras pessoas. A utilização dos gabaritos do LPA.

Perguntas objetivas e respostas:

1. O Líder de Projeto e a Gerência de Projeto estão com maior controle do projeto?
2. Os compromissos definidos no cronograma do projeto estão sendo atendidos no prazo?
3. Você pode afirmar que o tempo gasto com o uso do processo (*overhead*) não veio a comprometer o desenvolvimento do projeto?
4. O processo é seguido pela organização?
5. Os projetos são melhor planejados?
6. O processo é de fácil entendimento?
7. O cliente percebe maior qualidade do serviço prestado?
8. O cliente demonstra maior satisfação com os resultados apresentados?
9. Você pode afirmar que não ofereceu muita resistência durante a implantação do processo, devido às mudanças nos hábitos que possuía antes de utilizá-lo?
10. Você pode afirmar que o processo não é muito burocrático?
11. Você continuaria utilizando o processo?
12. Você acha que o processo, por ser navegável, facilita a sua utilização?
13. Você acha que os gabaritos do LPA facilitam e motivam as pessoas a usarem o processo?
14. Você percebeu que tem mais controle dos requisitos do projeto?
15. Você acha que as diretrizes de planejamento definidas pelo LPA atendem às necessidades do projeto?

As respostas estão apresentadas na Tabela 3, com a indicação da quantidade de pessoas que responderam em cada opção.

Pergunta	Não	+/-	Sim	Não foi possível avaliar
1			4	1
2	1	4		
3	1	2	1	1
4			5	
5			5	
6		4	1	
7		1		4
8				5
9	1	2	2	
10		4	1	
11			5	
12			5	
13			5	
14			5	
15			4	1
Totais	3	17	43	12

Tabela 3 - Respostas das Perguntas Objetivas

Os valores apresentados na tabela podem ser representados em valores percentuais conforme ilustra a Figura 57.

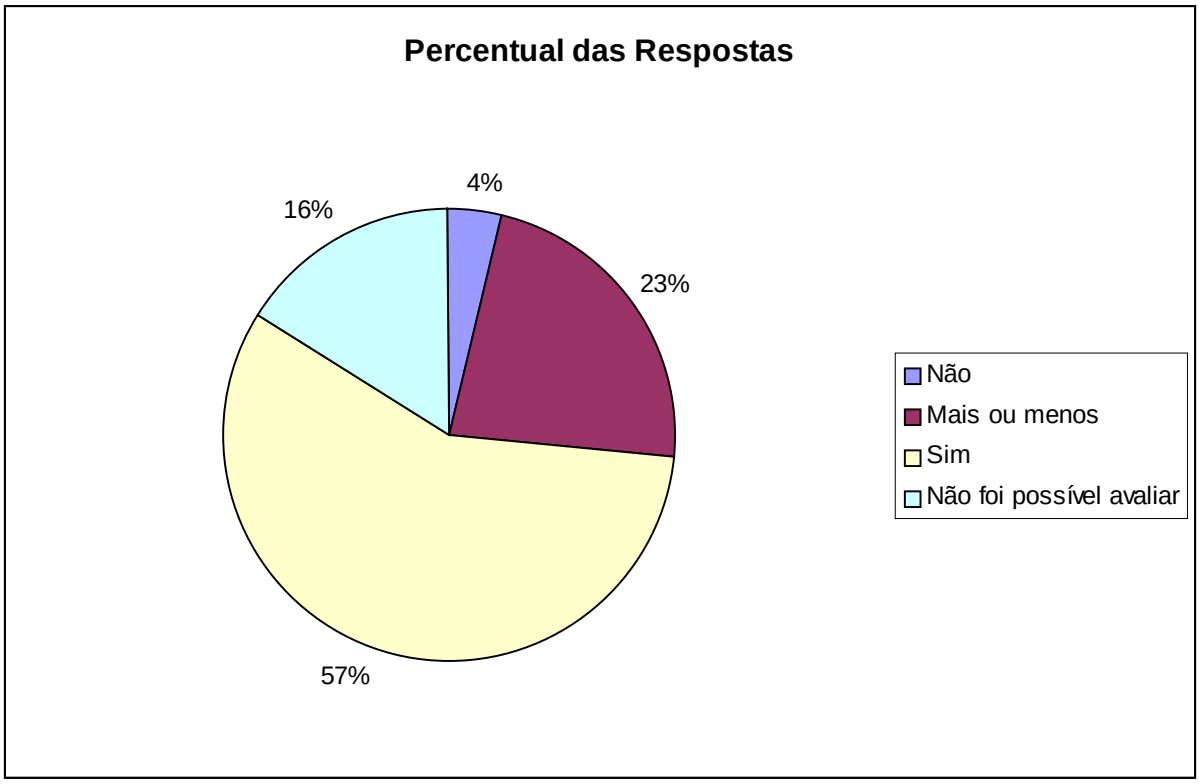

Figura 57 - Valores Percentuais das Respostas às Perguntas Objetivas

Observa-se que o LPA demonstrou ser 57% satisfatório às necessidades da organização e que este percentual poderia ser melhorado caso houvesse mais tempo para avaliá-lo, para poder reduzir os 27% de respostas indicadas como a não possibilidade de avaliação.

5 CONCLUSÃO

Neste capítulo são abordadas as conclusões obtidas com a elaboração, aplicação e resultados alcançados com o processo LPA e dos trabalhos futuros que possam melhorar e estender as capacidades do mesmo.

5.1 Contribuições da Pesquisa

Este trabalho é o resultado da experiência acumulada com a implantação do modelo de qualidade de SW-CMM nível 2 na organização Inatel Competence Center – ICC do Inatel, no período de março de 2001 a fevereiro de 2003. Com os conhecimentos que foram sendo adquiridos, iniciou-se a pesquisa, construção e aplicação do processo LPA que consumiram 660h00 de dedicação, incluindo-se a aplicação do mesmo na SSA (Sessão de *Software* Administrativo) do Inatel.

O LPA foi desenvolvido para ser parcialmente aderente ao SW-CMM nível 2 e permitir a sua execução ou navegação por meio de uma ferramenta de navegação da Internet, onde pôde-se comprovar estes resultados mediante o mapeamento do LPA para com os quesitos do SW-CMM nível 2, disponíveis no Apêndice A - Mapeamento LPA x SW-CMM Nível 2, e sua execução pôde ser feita utilizando-se a ferramenta Internet Explorer 6.0, utilizada durante a aplicação do processo na SSA do Inatel.

A aplicação do processo exigiu um alto esforço com treinamentos das pessoas integrantes da organização. Uma parte deste treinamento se referiu exclusivamente à estrutura do LPA composta de papéis, artefatos e atividades, além de uma visão geral sobre o RUP e o SW-CMM. A outra parte do treinamento esteve relacionada com carências conceituais da Engenharia de *Software*, tais como: ciclos de vida, UML, modelagem de casos de uso, planejamento e estimativas de projeto e controle de versão e mudanças de produtos, produtos estes resultantes das atividades realizadas durante o uso do processo no projeto. Observou-se grande atenção e interesse das pessoas durante esta fase de aplicação do processo e que sem estes fatores provavelmente os trabalhos seguintes seriam prejudicados. Pôde-se aprender nesta fase de aplicação do LPA que uma organização imatura para com processos de desenvolvimento de *software* possui grande carência de informações com relação à Engenharia de *Software*, requerendo grande esforço para repasse dos conceitos fundamentais,

para que seja alcançado o nivelamento dos conhecimentos a ponto de o processo poder ser aplicado.

Após os treinamentos iniciais e com a identificação do projeto a ser utilizado com o LPA, perceberam-se dificuldades de se manter os cronogramas do projeto devido ao envolvimento do time com outras atividades relacionadas com a manutenção de outros projetos que haviam sido desenvolvidos por eles. Além disto, percebeu-se que o time tinha forte tendência para o início da codificação, deixando-se de lado as questões como planejamento do projeto. Para não desmotivar ou intimidar demais o time, deixou-se que fosse dada maior prioridade com relação à especificação dos requisitos do projeto e construção do modelo de casos de uso (atores, casos de uso, diagramas de casos de uso e fluxos de eventos), para que as pessoas pudessem praticar com o LPA. Após algumas semanas, teve-se que mostrar a necessidade e os benefícios de um bom planejamento para o projeto, quando então uma parte do time passou a ficar mais envolvida com as questões do planejamento e o acompanhamento do projeto.

Percebeu-se, de forma extremamente importante, a motivação das pessoas para com a utilização do processo e procurou-se adequar as situações do LPA às necessidades reais do time. Um exemplo foi a mudança na descrição dos fluxos de eventos dos casos de uso, onde percebeu-se grande dificuldade na descrição, sob a forma tabular algorítmica (conforme previsto no gabarito MCU), permitindo-se uma mudança para o formato puramente descritivo. Pôde-se perceber que a imposição de determinados aspectos do processo (como o citado anteriormente) poderia levar o time à desmotivação e ao estresse, comprometendo a efetividade de sua utilização. Segundo ZAHRAN (1998i, p. 38), a falta de imposição na utilização de um processo pode levar a um ambiente não efetivo de sua utilização, ou seja, pode tornar o processo não institucionalizado. É claro que deve-se entender o termo “imposição” não na obrigação a todo o custo da utilização do processo, mas sim no entendimento das necessidades dos usuários do processo e nas flexibilidades que o processo deve possuir, para não se tornar algo extremamente amarrado, constituindo-se uma camisa-de-força para quem vier a utilizá-lo.

Pelo perfil da organização onde o processo foi aplicado, pôde-se perceber que as questões de custo do projeto simplesmente não eram tratadas, uma vez que os desenvolvimentos eram feitos apenas para outras organizações do próprio Inatel, ou seja, os projetos desenvolvidos pela organização eram todos internos. Embora esta necessidade não seja eminente, salientou-se que, se a organização passasse a fornecer estas informações para a alta gestão da

organização, ela poderia ter maior controle dos investimentos a serem feitos e ficaria mais satisfeita por ter uma ferramenta a mais para o gerenciamento das atividades de seus subordinados, podendo-se estimar melhor os investimentos futuros.

Percebeu-se também, de forma importante, a declaração de que o projeto passou a ser melhor controlado, o que pode-se considerar como sendo um grande ponto forte do LPA. Isto foi devido ao desenvolvimento de um planejamento e acompanhamento das atividades planejadas.

Embora tenha sido entregue apenas uma *release* para o Cliente, a declaração de que ele ficou mais envolvido no projeto, percebendo que também possui responsabilidades para com o desenvolvimento, torna-se um fator positivo para o LPA.

Com relação à resistência das pessoas para com a utilização do processo, embora não tenha sido algo crítico para a aplicação do LPA, pode-se perceber que esta resistência sempre irá existir devido ao fator novidade que o processo traz para a organização e as mudanças nos costumes de trabalho da mesma.

Apesar de o processo ter sido considerado um pouco burocrático pela maioria das pessoas, percebeu-se que elas dão grande importância à documentação do projeto, visando facilitar os trabalhos de manutenções futuras, bem como o repasse das informações para outras pessoas que não participaram do desenvolvimento. Trabalhos futuros podem identificar os pontos em que o processo demonstra ser burocrático e proceder a ajustes e novas aplicações para teste.

Quanto às mudanças necessárias no LPA, percebeu-se como sendo um fator positivo o fato de não terem surgido muitas necessidades de mudança, embora algumas adaptações tenham sido feitas durante a sua aplicação. Estas adaptações visaram à correção de defeitos identificados durante a aplicação e mudanças de conteúdo para melhor atendimento das necessidades da organização.

Deve-se fazer uma consideração com relação à presença do autor durante a aplicação do LPA na organização. Devido ao fato da minha participado nos treinamentos iniciais, e eventuais nas reuniões de consultoria e em algumas reuniões de acompanhamento, pode-se questionar qual seria a efetividade e eficiência da utilização do mesmo sem a presença do autor. Trabalhos futuros podem avaliar esta questão, ressalvando que os treinamentos iniciais deverão ser considerados, caso a organização apresente alguma carência para com os pré-requisitos

iniciais de utilização do processo, tais como: ciclo de vida interativo e incremental e modelagem de casos de uso.

Vale a pena, neste momento, rever os objetivos inicialmente traçados para o LPA e verificar sua completude. Os objetivos inicialmente traçados para este trabalho, conforme descritos no item 1.3, foram:

- Apresentar, aplicar e mostrar os resultados de um processo aderente parcialmente ao SW-CMM nível 2 em uma organização pertencente ao nível 1 do modelo SW-CMM.
- Ser de fácil utilização, para que as pessoas de organizações imaturas possam utilizá-lo com baixos índices de rejeição às mudanças necessárias.

De acordo com estes objetivos, verificou-se que:

- O LPA pôde ser apresentado à SSA do Inatel durante um período aproximado de 6 meses e que trouxe os resultados apresentados no item 4.4. Verificou-se que o LPA permite que projetos de desenvolvimento de *software* possam ser feitos de forma mais planejada e controlada, gerenciando-se adequadamente os requisitos ao longo de sua implementação, tendo-se como base as premissas especificadas no SW-CMM nível 2.
- Também de acordo com os resultados apresentados no item 4.4, verificou-se que o processo foi de fácil utilização, principalmente devido à sua característica de ser navegável em uma ferramenta de navegação na Internet.

O mais importante de todo este trabalho foi poder perceber que, independentemente das qualificações ou desqualificações do LPA, da intensidade ou não de como foi utilizado, do envolvimento das pessoas com outras atividades e também do processo não ter sido acompanhado até o final do projeto, foi identificar mediante a análise dos questionários aplicados e pelas opiniões das pessoas que houve uma melhoria na qualidade dos trabalhos que eles passaram a desempenhar e que todos pretendem continuar utilizando o processo. Uma vez que o processo está trazendo maior qualidade no desenvolvimento de *software*, justifica a sua utilização e análises para a sua constante melhoria.

5.2 Trabalhos Futuros

Assim como qualquer trabalho de pesquisa, desenvolvimento e aplicação sempre resulta em possibilidades de melhoria, com o LPA a situação não poderia ser diferente. Durante seu desenvolvimento e aplicação, pôde-se perceber as seguintes possibilidades de trabalhos futuros que poderiam provocar a sua melhoria contínua:

- Aplicação do processo em outros projetos para permitir melhor avaliação dos resultados.
- Aplicação do processo sem a intervenção do autor, para verificar se ele é claro o suficiente para que possa ser utilizado com a mesma efetividade e eficiência.
- Identificar os pontos em que o processo demonstrou ser burocrático, proceder a ajustes de conteúdo e tornar a avaliá-lo, aplicando-se em um novo projeto.
- Desenvolvimento de 1 livro didático para ser aplicado em disciplinas de Engenharia de *Software* em cursos de Sistemas de Informação. Este livro ofereceria subsídios para cobrir boa parte da ementa, tais como:
  - Ciclo de vida – Incluindo os ciclos de vida cascata, espiral, iterativo e incremental.
  - Requisitos – Incluindo a captura dos requisitos a partir de entrevistas com o cliente, especificação dos requisitos com a identificação de atores, casos de uso e a construção de um diagrama de casos de uso, bem como o desenvolvimento destes requisitos com a identificação dos fluxos de eventos dos casos de uso identificados, concluindo-se, com isto, a modelagem de casos de uso.
  - Planejamento e Acompanhamento - Incluindo o levantamento das estimativas iniciais de tamanho do projeto e derivando as estimativas de esforço, prazo e custo. Planejamento da equipe de desenvolvimento, os recursos de *hardware* e *software* necessários, as entregas previstas, os riscos identificados e um acompanhamento geral de todas estas atividades.
- Introdução de práticas de processos de desenvolvimento ágeis, tais como: XP, SCRUM, Crystal Clear e outros.
- Expansão do LPA com a introdução das KPAs de Garantia de Qualidade de *Software* e Gerência de Configuração e Mudança, oferecendo maior cobertura para com o nível 2 do SW-CMM ou migrá-lo para o modelo CMMI.

- Aplicar o processo em outros tipos de organização, com a finalidade de verificar as diferenças de ajustes necessários, criando-se assim perfis de organização e instâncias adequadas do LPA para cada perfil organizacional.
- Avaliar o LPA por alguma organização oficial de avaliação do modelo SW-CMM para verificar se ele realmente atende a 50% das KPAs do nível 2.

REFERÊNCIAS

AMBLER, S. W. Introduction to the Enterprise Unified Process (EUP). **A Ronin International, Inc. White Paper**. 26 dez. 2002. Disponível em: <http://www.enterpriseunifiedprocess.info/essays/introduction.html> Acesso em: 10 jul. 2003.

AMBLER, S. W. **Process patters**: Building large-scale systems using object technology, UK: Cambridge University Press, 1998, 549 p.

AMBLER, S. **Why the name change?** The Oficial Agile Modeling (AM) site. Disponível em: <http://www.agilemodeling.com/essays/nameChange.htm> Acesso em: 21 jun. 2003.

ARLOW, Jim; NEUSTADT, I. **UML and the unified process:** practical object-oriented analysis and design. Massachusetts, USA: Addison-Weslay, 2001. 416 p. p. 22-26.

BAMBERGER J. Essence of the capability maturity model**. IEEE Computer Magazine**, v. 30, n. 6, p. 112-114, June 1997**.** Também disponível sob restrição em: <http://dlib2.computer.org/co/books/co1997/pdf/r6112.pdf> Acesso em: 18 jul. 2003.

BECK, K. **Extreme programming explained**: Embrace change. Massachusetts, USA: Addison Wesley, 2000. 224 p.

BOEHM, B. W. A spiral model of software development and enhancement. **IEEE Computer,** v. 21, n. 5, p. 61-72, May 1988. Também disponível sob restrição em: <http://csdl.computer.org/comp/mags/co/1988/05/r5toc.htm> Acesso em: 10 abr. 2003.

BOEHM. B.; TURNER R. Using risk to balance agile and plan-driven methods. **IEEE Computer Magazine**, v. 36, n. 6, p. 57-66, June 2003. Também disponível sob restrição em: <http://csdl2.computer.org/dll/mags/co/2003/06/r6057.pdf> Acesso em: 03 jul. 2003.

BOOCH, G. **Object-oriented analysis and design with applications.** Massachusetts, USA: Addison Wesley, 1993. 608 p.

CARNEGIE MELLON UNIVERSITY. Software Engineering Institute. **Process Maturity Profile**, abril 2003. Disponível em: <http://www.sei.cmu.edu/sema/pdf/SW-CMM/2003apr.pdf> Acesso em: 07 set. 2003.

CETUS LINKS – OBJECT ORIENTATION, Aug. 2002. Disponível em: <http://www.cetus-links.org/oo_ooa_ood_methods.html> Acesso em: 07 set. 2003.

COAD, P; LEFEBVRE E.; LUCA J. de. **Feature-Driven development**. Disponível em: <http://www.coad.com/peter/download/bookpdfs/jmcuch06.pdf> Acesso em: 16 ago. 2003.

COCKBURN, A. **Agile software development**. Massachusetts, USA: Addison-Wesley, 2001. 256 p.

COCKBURN, A. **The laissez-faire of programming**: 'Crystal Clear' as a human-powered methodology for small development teams. Humans and Technology Jan 29, 2002. Disponível em: <http://alistair.cockburn.us/crystal/books/cc/crystalclear.doc> Acesso em: 06 jul. 2003.

COHN M. **The scrum development process**. Disponível em: www.mountaingoatsoftware.com/scrum Acesso em: 13 mai. 2003.

COHN M.; FORD D. Introducing an Agile Process to an Organization. **IEEE Computer**, New York, USA, v. 36, n. 6, p. 74-78, jun. 2003. Também disponível sob restrição em: <http://csdl.computer.org/dl/mags/co/2003/06/r6074.pdf> Acesso em: 02 jul. 2003.

DSDM CONSOTIUM. **The history of DSDM**. Disponível em: <http://www.dsdm.org/en/about/history.asp> Acesso em: 16 ago. 2003(a).

DSDM CONSOTIUM. **Why is DSDM different?** Disponível em: <http://www.dsdm.org/en/about/overview.asp> Acesso em: 16 ago. 2003(b).

GOLDSTEIN, A. **Extreme programming takes hold in software collaboration**. Sabre Holdings, 23 fev. 2003. Disponível em: <http://www.sabre-holdings.com/newsroom/promo/02_23_03.html> Acesso em: 21 jun. 2003.

GONÇALVES, J. M; BOAS, A. V. **CMM Nível 2**. CPQD, Campinas, SP: 2001**.** Disponível em: <http://www.mct.gov.br/sepin/Dsi/PBQP/Reuniao%20BSB/CMM-TR24_.V1.2.pdf > Acesso em: 20 set. 2002.

HENDERSON, B. **OPEN – Object-oriented process, environment and notation**: The first full lifecycle third generation OO method. Centre for Object Technology Applications and Research. School of Information Technology. Swinburn University of Technology. Disponível em: <http://www.open.org.au/Publications/Documents/crcchap.pdf> Acesso em: 21 jun. 2003.

HIGHSMITH J. **Agile software development ecosystem**. Massachusetts, USA: Addison-Weslay, 2002.

HIGHSMITH, J. A.; ORR, Ken. **Adaptative software development**: A Collaborative Approach to Managing Complex Systems. New York, USA: Dorset House, 2000, 392 p.

HUMPHREY J. W.; Engineers will tolerate a lot of abuse**. IEEE Software Magazine.** v. 18 n. 5, p. 13-15, Sept. 2001. Também disponível sob restrição em: <http://dlib2.computer.org/so/books/so2001/pdf/s5013.pdf> Acesso em: 18 jul. 2003.

IBM. **Industry-Proven Best Practices**, 2003. Disponível em: <http://www.rational.com/corpinfo/practices.jsp> Acesso em: 07 set. 2003

IEEE. **Std 1220-1998**: IEEE Standard for Application and Management of the Systems Engineering Process. 1998. Disponível sob restrição em: <http://ieeexplore.ieee.org/xpl/tocresult.jsp?isNumber=16015> Acesso em: 02 jul. 2003.

International Conference on Software Quality, 1998, Portland, OR, USA. Disponível em: <http://www.sei.cmu.edu/activities/cmm/papers/cmm-small.pdf> Acesso em: 23 maio 2002.

INTERNATIONAL TELECOMMUNICATIONS UNION. **ITU-T Recommendation Z.100**: specification and description language (SDL), 2002. Disponível de forma restrita em: <http://www.itu.int/rec/recommendation.asp?type=items&lang=e&parent=T-REC-Z.100-200208-I> Acesso em: 16 ago. 2003.

ISO/IEC **12207:1995**. Information Technology: software life cycle processes. Disponível de forma restrita em: <http://www.iso.ch/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=21208&ICS1=35&ICS2=80&ICS3= > Acesso em: 16 ago. 2003.

KRUCHTEN, P. What is the Rational Unified Process? **Rational Edge**, fev. 2003. Disponível em: <http://www.therationaledge.com/content/feb_03/PDF/WhatisRUP_TheRationalEdge_Feb2003.pdf> Acesso em: 10 jul. 2003.

MALAN, R.; LETSINGER, R. **Object-oriented development at work.** Upper Saddle River, NJ, USA: Prentice Hall, 1996. 389 p.

OMG. **Introduction to OMG's unified modeling language (UML)**. Disponível em: <http://www.omg.org/gettingstarted/what_is_uml.htm> Acesso em: 21 jun. 2003.

OPEN. **OML: Open modeling language.** 1999. Disponível em: <http://www.open.org.au/Publications/omlbooklink.html> Acesso em: 21 de jun. 2003.

OSTEREICH, B. **Developing software with UML**: Object-Oriented Analysis and Design in Practice. Massachusetts, USA: Addison Wesley, 1999, 321 p., p. 5-7.

OSTEREICH, B. **Developing software with UML**: Object-Oriented Analysis and Design in Practice. Massachusetts, USA: Addison Wesley, 1999, 321 p., p. 5-7.

PAULK, M. C. et al.; **The capability maturity model**: Guidelines for Improving the Software Process. Massachusetts, USA: Addison Wesley, 1999(a), 441 p., p. 355.

PAULK, M. C. et al.; **The capability maturity model**: Guidelines for Improving the Software Process. Massachusetts, USA: Addison Wesley, 1999(b), 441 p., p. 7.

PAULK, M. C. et al.; **The capability maturity model**: Guidelines for Improving the Software Process. Massachusetts, USA: Addison Wesley, 1999(c), 441 p., p. 363.

PAULK, M. C. et al.; **The capability maturity model**: Guidelines for Improving the Software Process. Massachusetts, USA: Addison Wesley, 1999(d), 441 p., p. 128.

PAULK, M. C. et al.; **The capability maturity model**: Guidelines for Improving the Software Process. Massachusetts, USA: Addison Wesley, 1999(e), 441 p., p. 127.

PAULK, M. C. et al.; **The capability maturity model**: Guidelines for Improving the Software Process. Massachusetts, USA: Addison Wesley, 1999(f), 441 p., p. 128.

PAULK, M. C. et al.; **The capability maturity model**: Guidelines for Improving the Software Process. Massachusetts, USA: Addison Wesley, 1999(g), 441 p., p. 128.

PAULK, M. C. et al.; **The capability maturity model**: Guidelines for Improving the Software Process. Massachusetts, USA: Addison Wesley, 1999(h), 441 p., p. 128.

PAULK, M. C. et al.; **The capability maturity model**: Guidelines for Improving the Software Process. Massachusetts, USA: Addison Wesley, 1999(i), 441 p., p. 129.

PAULK, M. C. et al.; **The capability maturity model**: Guidelines for Improving the Software Process. Massachusetts, USA: Addison Wesley, 1999(j), 441 p., p. 136.

PAULK, M. C. et al.; **The capability maturity model**: Guidelines for Improving the Software Process. Massachusetts, USA: Addison Wesley, 1999(k), 441 p., p. 361.

PAULK, M. C. et al.; **The capability maturity model**: Guidelines for Improving the Software Process. Massachusetts, USA: Addison Wesley, 1999(l), 441 p., p. 364.

PAULK, M. C. et al.; **The capability maturity model**: Guidelines for Improving the Software Process. Massachusetts, USA: Addison Wesley, 1999(m), 441 p., p. 368.

PAULK, M. C. et al.; **The capability maturity model**: Guidelines for Improving the Software Process. Massachusetts, USA: Addison Wesley, 1999(n), 441 p., p. 356.

PAULK, M. C. et al.; **The capability maturity model**: Guidelines for Improving the Software Process. Massachusetts, USA: Addison Wesley, 1999(o), 441 p., p. 4.

PAULK, M. C. et al.; **The capability maturity model**: Guidelines for Improving the Software Process. Massachusetts, USA: Addison Wesley, 1999(p), 441 p., p. 6.

PAULK, M. C. et al.; **The capability maturity model**: Guidelines for Improving the Software Process. Massachusetts, USA: Addison Wesley, 1999(q), 441 p., p. 3.

PAULK, M. C. et al.; **The capability maturity model**: Guidelines for Improving the Software Process. Massachusetts, USA: Addison Wesley, 1999(r), 441 p., p. 7.

PAULK, M. C. et al.; **The capability maturity model**: Guidelines for Improving the Software Process. Massachusetts, USA: Addison Wesley, 1999(s), 441 p., p. 237.

PAULK, M. C. Extreme programming from a CMM perspective. **IEEE Software**, v.18 , n.6 , p. 19-26, Nov. 2001. Também disponível sob restrição em: <http://dlib2.computer.org/so/books/so2001/pdf/s6019.pdf> Acesso em: 06 jan. 2003.

PAULK, M. C. **Using the software CMM in small organizations**. The Joint 1998 Proceedings of the Pacific Northwest Software Quality Conference and the Eighth

PAULK, M. C. Using the software CMM with good judgment**. ASQ Software Quality Professional**, v. 1, n. 3, p. 19-29, June 1999s. Também disponível em: <http://www.sei.cmu.edu/activities/cmm/papers/judgment.pdf >**.** Acesso em 10 ago. 2003.

PFLEEGER, S. L. **Software engineering**: Theory and practice. 2. ed. USA: Prentice-Hall, Inc., 2001. 659 p., p. 45.

POPPENDIECK, M.; POPPENDIECK, T. **Lean software development**: an agile toolkit for software development managers. Massachusetts, USA: Addison-Wesley, 2003. 240 p.

QUALIDADE E PRODUTIVIDADE NO SETOR DE SOFTWARE BRASILEIRO 2001. Brasília: MCT-SEPIN: n. 4, 2001. 260 p., p. 86-87.

RATIONAL SOFTWARE CORPORATION . RUP - Rational Unified Process 2002.05.00.25. Rational USA, 2002.

RUMBAUGH, J. **OMT Papers**. 1994. Disponível em: <http://www.rational.com/media/whitepapers/omtpapers.pdf> Acesso em: 21 jun. 2003.

SHLAIR, S; MELLOR, S. J. **Object-oriented systems analysis**: Modeling the world in data. New York: Pearson Education, 1988. 144 p.

SOARES A. C. Trabalhando requisitos usando a modelagem de casos de uso. **Revista Científica da FAI**, Santa Rita do Sapucaí, MG, v. 2, n. 1, p. 46-56, 2002.

WIEGERS K.; **Process improvement that works. Software Development**, Oct. 1999. Disponível em: <http://www.sdmagazine.com/documents/s=753/sdm9910a/9910a.htm> Acesso em: 01 abr. 2002

WILLIAMS, L.; COCKBURN, A. Agile software development: it's about feedback and change. **IEEE Computer Magazine**, v. 36, n. 6, p. 39-43, June 2003. Disponível sob restrição em: <http://csdl2.computer.org/dll/mags/co/2003/06/r6039.pdf> Acesso em: 03 jul. 2003.

ZAHRAM, S. **Software process improvement**: practical guidelines for business success. Massachusetts, USA: Addison-Wesley, 1998(a), 447 p., p. 139-140.

ZAHRAM, S. **Software process improvement**: practical guidelines for business success. Massachusetts, USA: Addison-Wesley, 1998(b), 447 p., p. 339-355.

ZAHRAN, S. **Software process improvement:** practical guidelines for business success. Inglaterra: Massachusetts, USA: Addison-Wesley, 1998(c), 447 p., p. 14-15.

ZAHRAN, S. **Software process improvement:** practical guidelines for business success. Inglaterra: Massachusetts, USA: Addison-Wesley, 1998(d), 447 p., p. 17.

ZAHRAN, S. **Software process improvement:** practical guidelines for business success. Inglaterra: Massachusetts, USA: Addison-Wesley, 1998(e), 447 p., p. 28.

ZAHRAN, S. **Software process improvement:** practical guidelines for business success. Inglaterra: Massachusetts, USA: Addison-Wesley, 1998(f), 447 p., p. 39.

ZAHRAN, S. **Software process improvement:** practical guidelines for business success. Inglaterra: Massachusetts, USA: Addison-Wesley, 1998(g), 447 p., p. 99.

ZAHRAN, S. **Software process improvement:** practical guidelines for business success. Inglaterra: Massachusetts, USA: Addison-Wesley, 1998(h), 447 p., p. 208.

ZAHRAN, S. **Software process improvement:** practical guidelines for business success. Inglaterra: Massachusetts, USA: Addison-Wesley, 1998(i), 447 p., p. 38.

OBRAS CONSULTADAS

ADOLPH, S. **But we´re CMM level 5!**. Disponível em <http://www.sdmagazine.com/print/documentID-20210> Acesso em 04 jan. 2002.

ALBUQUERQUE, S. F. Ciclo de vida no desenvolvimento de sistemas. **GuidelineBRISA**, out. 2002. Disponível sob restrição em: http://www.brisa.org.br Acesso em: 13 maio 2003.

AMARATUNGA, D.; SARSHAR, M.; BALDRY, D.. Process improvement in facilities management: the SPICE approach, **Bussiness Process Management Journal**, v. 8, n. 4, p. 318-337, 2002

AMBER, S. W. **The principes of Agile Modeling (AM)**. Disponível em <http://agilemodeling.com/principles.htm> Acesso em 13 nov. 2001.

AMBLER, S. **Enhancing the unified process**.Disponível em < http://www.aigsi.com/unifiedProcess.pdf> Acesso em 06/10/2003.

AMBLER, S. W. **Agile modeling and the unified process**. Disponível em: <http://www.agilemodeling.com/essays/agilemodelingRUP.htm> Acesso em: 4 jan. 2002.

AMBLER, S. W. Introduction to the Enterprise Unified Process (EUP). **A Ronin International, Inc. White Paper**. 26 dez. 2002. Disponível em: <http://www.enterpriseunifiedprocess.info/essays/introduction.html> Acesso em: 10 jul. 2003.

AMBLER, S. W. Lessons in agility from internet-based development. **IEEE Software Magazine**. p. 66-73, Mar./Apr. 2002. Também disponível sob restrição em: <http://dlib2.computer.org/so/books/so2003/pdf/s2066.pdf> Acesso em: 10 jul. 2003.

AMBLER, S. W. **The practices of Agile modeling**. Disponível em: <http://www.agilemodeling.com/practices.htm> Acesso em: 13 nov. 2001.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 10520**: Informação e documentação – Citações em documentos – Apresentação. Rio de Janeiro, ago. 2002.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 6023**: Informação e documentação – Referências – Elaboração. Rio de Janeiro, ago. 2003.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **Projeto 21:101.01-009**: Engenharia de software – Qualidade de produto – Parte 1: Modelo de Qualidade. Rio de Janeiro, jun. 2002.

Beck, K. et Al. **Manifesto for Agile Software Development**. Disponível em: <http://www.agilemanifesto.org> Acesso em: 03 jul. 2003.

BECK, K.; BOEHM, W. B. Agility through discipline: A debate. **IEEE Computer Society**, v. 6, n. 36, p. 44-46, jun. 2003.

BELL, D. **UML basics:** an introduction to the unified modeling language**.** Disponível em: <http://www.therationaledge.com/content/jun_03/pdf/f_umlintro_db.pdf > Acesso em: 18 jun. 2003.

BITTNER, K. Why use case are not “functions”. **Rational Edge**, Dec. 2000. Disponível em: <http://www.therationaledge.com/content/dec_00/t_ucnotfunctions.html> Acesso em: 03 jul. 2001.

BOEHM, B. **CMMI and the balance of discipline and agility**. Disponível em <http://www.dtic.mil/ndia/2002emmi/boehm.pdf> Acesso em: 03 jul. 2003.

BOEHM, B. **Get ready for agile methods, with care**. Disponível em: <http://dlib2.computer.org/co/books/co2002/pdf/r1064.pdf> Acesso em: 03 set. 2003.

BONA, C. **Avaliação de processos de software**: um estudo de caso em XP e ICONIX. Florianópolis, 2002. 122 f. Dissertação (Mestrado em Engenharia de Produção), Universidade Federal de Santa Catarina.

CANTOR, M. L. **Thoughts on functional decomposition**. Disponível em: <http://therationaledge.com/content/apr_03/f_functionalDecomp_me.jsp> Acesso em: 18 jul. 2003.

CARDOZO, E. L. Project planning best practices. **Rational Edge**, Aug. 2003. Disponível em: <http://www.therationaledge.com/content/aug_03/m_projectplanning_dd_ec.jsp> Acesso em: 15 ago. 2003.

CARNEGIE MELLON UNIVERSITY. Software Engineering Institute. **Cleanroom Software Engineering**, Oct, 1997. Disponível em: <http://www.sei.cmu.edu/str/descriptions/cleanroom.html> Acesso em: 08 jul. 2003.

CAUWENBERGHE, P. V. **Agile fixed price projects part 1:**’the price is right’. Disponível em <http://agilealliance.org/articles/index> Acesso em: 29 ago. 2003.

CAUWENBERGHE, P. V. **Agile fixed price projects part 2:**’do you want agility with that?**’** Disponível em <http://agilealliance.org/articles/index> Acesso em: 29 ago. 2003.

COOPER, K. **CARE Assisant tool project – proposal**. Disponível em <http://mrpeabody.utdallas.edu/~rshah/documents/Proposal_FINAL.doc> Acesso em: 20 maio 2003.

ECKEL, B. Create winning projects in any language. **Software Development**. Disponível em: <http://www.sdmagazine.com/print/documentID=11241> Acesso em: 01 abr. 2002.

EVANS, G. K. **A simplified approach to RUP**. Disponível em: <http://therationaledge.com/content/jan_01/t_rup_ge.html> Acesso em: 18 jun. 2003.

EVANS, G. K., NAZZARO, W. F. **Killing your project with Use Cases!**. Disponível em <http://www.williamnazzaro.com/presentations.htm> Acesso em: 17 abr. 2003.

FALBO, R. A.; MENEZES, C. S.; ROCHA, A. R. C. **Assist-Pró: Um assistente baseado em conhecimento para apoiar a definição de processos de software**. Disponível em: <http://www.inf.ufsc.br/sbes99/anais/sbes-completo/14.pdf> Acesso em: 04 jun. 2002.

FERM, F. The want, why, and how of a subsystem. **Rational Edge**, Jun. 2003. Disponível em: <http://www.therationaledge.com/content/jun_03/t_subsystem_ff.jsp> Acesso em: 10 ago. 2003.

FOWLER, M. **Is design dead?**. Disponível em: <http://www.martinfowler.com/articles/designDead.html> Acesso em: 17 abr. 2003.

FOWLER, M. **The new methodology**, Disponível em: <http://www.thoughtworks.com/library/newMethodology.pdf> Acesso em 26 mar. 2002

HENDERSON-SELLERS, B. Object-oriented methods and processes. In: INTERNATIONAL CONFERENCE ON SOFTWARE AND TOLLS, nov. 2000, Wollongong, Austrália. p. 7-12.

HENDERSON-SELLERS, B; DUÉ, R.; GRAHAM, I. Third Generation OO processes: A critique of RUP and OPEN from a project managemente perspective. In: SEVENTH ASIA-PACIFIC SOFTWARE ENGINEERING CONFERENCE (APSEC'00), Singapura, Dez. 2000. p. 428-435

HEUMANN, J. **Introduction to business modeling using the unified modeling language (UML)**. Disponível em: <http://therationaledge.com/content/mar_01/m_uml_jh.html> Acesso em: 18 jul. 2003.

JACOBSON, I.; **Use Cases - Yesterday, today, and tomorrow**. Disponível em <http://www.therationaledge.com/contente/mar-03/f_useCases_ij.jsp> Acesso em 01 jul. 2003.

JAKOBSSON, M. **Predicting software quality with ISO/IEC TR 15504**: capability determination of the rational unified process**.** Boras, Suécia: 2000. Dissertação de mestrado do Departamento de Informática do Curso de Administração em Negócios e Ciência da Computação da Universidade de Högskolan i Borås.

JOHNSON, D. L.; BRODMAN, J. G. Applying CMM project planning practices to diverse enviroments. **IEEE Software,** vol. 17 , n. 4, p. 40-47, jul./ago. 2000.

JURIC, R.; KULJIS, J. Building na evaluation instrument for OO CASE tool assessment for unified modelling language support. In: PROCEEDINGS OF THE 32ND HAWAII INTENTIONAL CONFERENCE ON SYSTEM SCIENCES, jan. 1999, Maui, Hawaii, USA Institute of Electrical and Electronics Engineers, Inc. (IEEE).

KENT, B. Embracing change wih extreme programming. **IEEE Computing**, v.32, n.10, p.70-77, out. 1999

KRUCHTEN, P. **From waterfall to iterative development –A challenging transition for project managers**. Disponível em: <http://therationaledge.com/content/dec_00/m_iterative.html> Acesso em: 22 fev. 2003.

LARMAN, C.; BASILI, V. R. Iterative and incremental development: A brief history. **IEEE Computer Society**, v. 36, n. 9, p. 47-56, jun. 2003.

LINGER, R. C. Cleanroom process model. **IEEE Software**, n.11 v. 2, p. 50-58, Mar. 1994.

MANZONI, L. V; PRICE, R. T. Identifying Extensions Required by RUP (Rational Unified Process) to Comply with CMM (Capability Maturity Model) Levels 2 and 3. **IEEE Transactions on Software Enginering**, v. 29, n. 2, p. 181-192, fev. 2003.

MARK, C. P.;CURTIS, B.;CHRISSIS, M. B. Capability maturyty model, version 1.1. **IEEE Software**, vol. 10, n. 4, p. 18-27, Jul. 1993

MERZ, U. **Small teams, big problems**. Disponível em <http://www.sdmagazine.com/print/documentID=11080> Acesso em 4 jan. 2002.

Mills, H.D., Dyer, M., Linger, R.C., *Cleanroom Software Engineering*. **IEEE Software,** n. 4, p. 19-24, Set. 1987

MINI-RECIPES, DEFINE, DESIGN, DEVELOP, TEST. Disponível em: <hhtp://www.agilealiance.com/articles/loadrecipes18.pdf> Acesso em 03/07/2003.

NAGESWARAN, S. **Test effort estimation using use case points**. Disponível em: <http://www.cognizant.com/cogcommunity/presentations/Test_Effort_Estimation.pdf> Acesso em: 12 abr. 2002.

NAYIMA, P. V. C. **Agile fixed price projects part 1:'The Price is Right'**. Disponível em <http://agilealliance.org/articles/index> Acesso em: 29 ago. 2003.

NAYIMA, P. V. C. **Agile fixed price projects part 2:'Do you want agility with that?'** . Disponível em <http://agilealliance.org/articles/index> Acesso em: 29 ago. 2003.

NG, P. W. **Adopting use cases Part I: Understanding types of use cases and artifacts**. Disponível em <http://www.therationaledge.com/content/jun-03/m_ng.jsp> Acesso em 01 jul. 2003.

NG, P. W. **Adopting use cases Part II: Puting learning into practice**. Disponível em <http://www.therationaledge.com/contente/jun-03/m_parttwo_ng.jsp> Acesso em 01 jul. 2003.

NG, P. W. **Effective business modeling with UML: describing business Use Cases and realizations**. Disponível em <http://www.therationaledge.com/content/nov_02/t_businessUML_pn.jsp> Acesso em 01 jul. 2003.

OSTERWEIL, L. J.;KRUCHTEN, P.; FOWLER, M. Lightweight vs. Heavyweight processes: is this even the right question? In. PROCEEDINGS OF THE 24TH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, Orlando, Flórida, Estados Unidos, 2002, p. 649-652.

PAULK, M. C. **Extreme programming from a CMM perspective**. Disponível em < http://www.sei.cmu.edu/cmm/papers/xp-cmm-paper.pdf> Acesso em 10 mai. 2003.

PAULK, M. **Questions and Answers on the CMM, Issue 2**, 1994. Disponível em: <http://stfc.comp.polyu.edu.hk/library.html/q and a.2.html> Acesso em: 10 jul. 2003.

PAULK, M. **Questions and Answers on the CMM, Issue 3**, 1994. Disponível em: <http://stfc.comp.polyu.edu.hk/library.html/q and a.3.html> Acesso em: 10 jul. 2003.

PEVILLIERS, DJ. **Introducing the RUP into na organization**. Disponível em: <http://therationaledge.com/content/jan_02/m_introducingTheRUP_dd.html> Acesso em: 22 fev. 2002.

POLLICE, G. **RUP and xp, part I:** finding common ground, Apr. 2001. Disponível em: <http://www.therationaledge.com/content/mar_01/f_xp_gp.html> Acesso em: 10 jul. 2003.

POLLICE, G. **RUP and xp, part II:** valuing differences, Apr. 2001. Disponível em: <http://www.therationaledge.com/content/mar_01/f_xp_gp.html> Acesso em: 10 jul. 2003.

PRIESTLEY, M.; UTT, M. H. A Unified process for Saftware and documentation development. In: PROCEEDINGS OF IEEE PROFESSIONAL COMMUNICATION SOCIETY INTERNATIONAL / PROFESSIONAL COMMUNICATION CONFERENCE AND PROCEEDINGS OF THE 18TH ANNUAL ACM INTERNATIONAL CONFERENCE ON COMPUTER DOCUMENTATION: TECHNOLOGY & TEAMWORK, Cambridge, Massachusetts, Set. 2000. p. 24-27.

PROBASCO, L. **Dear Dr. Use Case: What About "shall" wording in a use case?** Disponível em: <http://therationaledge.com/content/dec_01/t_drUseCase_Ip.html> Acesso em: 22 fev. 2002.

PROBASCO, L. **Dear Dr. Use Case: What About function points and use cases?**. Disponível em: <http://therationaledge.com/content/aug_02/t_drUseCase_Ip.jsp> Acesso em: 18 jul. 2003.

PROBASCO, L. **The Tem Essentials of RUP**. Disponível em: <http://therationaledge.com/content/dec_00/f_drup.html> Acesso em: 22 fev. 2002.

PROBASCO, L.; LEFFINGWELL, D. **Combining software requirements specifications with Use-Case modeling**. Disponível em <http://www.spc.ca/downloads/srs_usecase.doc> Acesso em: 15 jul. 2003.

SMITH, J. **A comparison of RUP and XP**. Disponível em <http://rational.com/media/whitepapers/rp167.pdf> Acesso em: 08 abr. 2002.

SMITH, J. **The estimation of effort based on use cases**. Disponível em: <http://www.rational.com/media/whitepapers/finalTP171.PDF?SMSESSION=NO > Acesso em: 30 jun. 2003.

SMITH, R. Defining the uml kernel. **Software Development**. Disponível em: <http://www.sdmagazine.com/print/documentID=11139> Acesso em: 04 set. 2002.

SOFTWARE ARCHITECTURE DOCUMENTATION IN PRACTICE: DOCUMENTING ARCHITECTURAL LAYERS. Disponível em <http://www.sei.cmu.edu/publications/documents/00.reports/00sr004/00sr004chap01.html> Acesso em 06/09/2003.

THE OFFICIAL AGILE MODELLING (AM) SITE. Disponível em:<http://www.agilemodeling.com/> Acesso em 13/11/2001

WILLIAMS, L et Al. Strengthening the case for pair programming. **IEEE software**, n. 4, p. 19-25.jul./ago. 2000.

WILLIAMS, L. A et Al. **The Effects of "pair-pressure" and "pair-learning" on software engineering education**. Disponível em: <http://www.pairprogramming.com/cseet.pdf> Acesso em 08 abr. 2002.

WILLIAMS, L. A.; KESSLER, R. R. Experimenting with Industry's 'Pair-Programming' Model in the Computer Science Classroom, **Journal on SW Engineering Education**, Dec. 2000.

WILLIAMS, L. et al. Strengthening the case for pair-programming. **IEEE Software**, v. 17, n. 4, p. 19-25, July-Aug. 2000. Também disponível sob restrição em: <http://ieeexplore.ieee.org/xpls/VSearch.jsp> Acesso em: 08 ago. 2002.

WYDER, T. **Capturing Requirements with Use Cases**. Disponível em: <http://www.sdmagazine.com/print/documentID=11489> Acesso em 6 mar. 2002

APÊNDICE A - MAPEAMENTO LPA X SW-CMM NÍVEL 2[29]

1. KPA da Gestão de Requisitos - RM

Compromissos	
SW-CMM-2	LPA
1 – O projeto segue uma política organizacional escrita para gerenciar os requisitos do sistema alocados ao *software*.	Artefato PGR (item 3.7.2) da atividade "Definir PGR" da macroatividade "Definir PGR" (item 3.4.4).

Habilidades	
SW-CMM-2	LPA
1 – Para cada projeto são estabelecidas responsabilidades para analisar os requisitos do sistema e alocá-los ao *hardware*, ao *software* e a outros componentes do sistema.	Atividade "Alocar Requisitos de *Software*" da macroatividade "Alocar Requisitos" da GR (item 3.4.1), que deve ser realizada pela Gerência de Projeto.
2 – Os requisitos alocados são documentados.	Os requisitos são documentados inicialmente no artefato Visão (item 3.7.4) e posteriormente detalhados em casos de uso no artefato MCU (item 3.7.6). A pessoa responsável pela documentação dos requisitos é a Gerência de Projeto (item 3.6.2).
3 – São providos recursos e verbas para se gerenciar os requisitos alocados.	É uma das políticas do LPA definida no artefato PGR (item 3.4.4). Além disto, os requisitos são acompanhados durante a atividade "Acompanhar PDA" da macroatividade de mesmo nome da GPA (item 3.5.2).
4 – Para realizar as atividades da gestão de requisitos, os membros do grupo de desenvolvimento de *software* são treinados.	É uma política do LPA definida no artefato PGR (item 3.4.4).

[29] As práticas e subpráticas do SW-CMM nível 2 foram traduzidas, adaptadas e utilizadas parcialmente de (GONÇALVES, BOAS, 2001).

Atividades	
SW-CMM-2	LPA
1 – O grupo de desenvolvimento de *software* revisa os requisitos alocados antes que eles sejam incorporados ao projeto de *software*.	Atividade "Revisar Requisitos de *Software* Alocados" da macroatividade "Alocar os Requisitos" da GR (item 3.4.1). Esta atividade é de responsabilidade do Líder de Projeto e é feita sob a forma de uma reunião com integrantes do projeto e sempre que possível com o Cliente.
2 – O grupo de desenvolvimento de *software* utiliza os requisitos alocados como base para os planos de *software*, produtos de trabalho e atividades.	Além de fazer parte das políticas descritas nos artefatos PGR (item 3.7.2) e PGPA (item 3.8.1), a atividade "Iniciar Projeto" da macroatividade "Elaborar o PDA" da GPA (item 3.5.1) descreve esta necessidade.
3 – As mudanças nos requisitos alocados são revisadas e incorporadas ao projeto de *software*.	As mudanças nos requisitos são tratadas na macroatividade "Gerenciar e Controlar Mudanças nos Requisitos" (item 3.4.3) onde é prevista a revisão e incorporação ao projeto dos requisitos alterados na atividade "Revisar Requisitos de *Software* Alocados" da macroatividade "Alocar os Requisitos" da GR (item 3.4.1).

Medições	
SW-CMM-2	LPA
1 – São realizadas medições para se gerenciar o *status* das atividades da gestão de requisitos alocados.	Atividade "Acompanhar PDA" da macroatividade de mesmo nome da PGPA (item 3.5.2). As medições são registradas no artefato PDA (item 3.8.2).

Verificações	
SW-CMM-2	LPA
1 – As atividades de gerência da alocação dos requisitos são revisadas periodicamente com a gerência sênior.	A Gerência Sênior do LPA é desempenhada pela Gerência de Projeto. Esta verificação é feita na atividade "Acompanhar PDA" da macroatividade de mesmo nome da GPA (item 3.5.2).
2 – As atividades de gerência da alocação dos requisitos são revisadas periodicamente e, eventualmente, com a gerência de projeto.	Esta verificação é feita na atividade "Acompanhar PDA" da macroatividade de mesmo nome da GPA (item 3.5.2).
3 – As atividades e produtos de trabalho da gestão de requisitos alocados são revisadas e/ou auditadas pelo grupo de garantia da qualidade e, ao final, os resultados são divulgados.	O LPA não cobre a KPA de Garantia da Qualidade de *Software*.

2. KPA da Gestão de Planejamento - SPP

Compromissos	
SW-CMM-2	LPA
1 – É designado um gerente do projeto de *software* para ser responsável pela negociação dos compromissos e desenvolvimento do plano de *software* do projeto.	Vide as responsabilidades da Gerência de Projeto (item 3.6.2) e do Líder de Projeto (item 3.6.6). A Gerência de Projeto é responsável por negociar os compromissos externos e fazer o planejamento inicial. O Líder de Projeto, pelos compromissos internos e continuidade com o planejamento do projeto.
2 – O projeto segue uma política organizacional escrita para o planejamento do projeto de *software*.	Vide artefato PGPA – Políticas da Gestão de Planejamento e Acompanhamento (item 3.8.1).

Habilidades	
SW-CMM-2	LPA
1 – Existe um documento de contrato de trabalho aprovado para o projeto de *software*.	Vide artefato Visão (item 3.7.4) e responsabilidade da Gerência de Projeto (item 3.6.2), onde define o contrato de trabalho como sendo o Visão (item 3.7.4).
2 – São atribuídas responsabilidades para o desenvolvimento do plano de desenvolvimento de *software*.	Vide artefato Cronograma (item 3.8.3), cuja responsabilidade de construção e manutenção pertence à Gerência de Projeto e ao Líder de Projeto.
3 – São providos recursos e verbas para o planejamento do projeto de *software*.	Vide responsabilidades da Gerência de Projeto (item 3.6.2).
4 – As gerências de *software*, engenheiros de *software* e outras pessoas envolvidas no planejamento do projeto de *software* são treinados em estimativas de *software* e procedimentos de planejamento aplicáveis às suas áreas de responsabilidades.	Atividades “Iniciar Projeto”, “Levantar Estimativas” e “Planejar Atividades de Engenharia e de Processo” da macroatividade “Elaborar o PDA” da GPA (item 3.5.1).

Atividades	
SW-CMM-2	LPA
1 – O grupo de desenvolvimento de *software* participa da proposta de projeto do time.	Vide responsabilidade da Gerência de Projeto (item 3.6.2). Esta gerência deve elaborar a proposta de trabalho documentada no artefato Visão (item 3.4.1) na atividade “Alocar Requisitos de *Software*” da macroatividade “Alocar os Requisitos” da GR (item 3.4.1). Este artefato é revisado pelo Líder de Projeto na atividade “Revisar Requisitos de *Software* Alocados” pertencente à mesma macroatividade e gestão.
2 – O planejamento do projeto de *software* é	O planejamento do projeto é iniciado tão logo

iniciado nos primeiros estágios e em paralelo com o planejamento geral do projeto.	se tenha os dados principais para o projeto, conforme descreve a atividade “Iniciar Projeto” da macroatividade “Elaborar PDA” da GPA (item 3.5.1).
3 – O grupo de desenvolvimento de *software*, juntamente com outros grupos envolvidos, participam do planejamento geral durante toda a vida do projeto.	O plano de desenvolvimento do projeto é elaborado pela Gerência de Projeto e pelo Líder de Projeto, com o apoio do Grupo de *Software*. Vide atividades “Iniciar Projeto”, “Planejar Atividades de Engenharia e de Processo” da macroatividade “Elaborar PDA” da GPA (item 3.5.1).
4 – Os compromissos do projeto de *software* assumidos com pessoas e grupos externos à organização são revisados pela gerência sênior de acordo com um procedimento documentado.	Vide atividade “Negociar Compromissos” da macroatividade “Elaborar PDA” da PGA (item 3.5.1). Esta revisão é feita pela Gerência de Projeto.
5 – É identificado ou definido um ciclo de vida de *software* que possua estágios predefinidos e de tamanhos gerenciáveis.	Vide atividade “Iniciar Projeto” da macroatividade “Elaborar PDA” da PGA (item 3.5.1). O ciclo de vida adotado pelo LPA é o Iterativo e Incremental.
6 – O plano de desenvolvimento do *software* é desenvolvido de acordo com um procedimento documentado.	Este procedimento encontra-se descrito tanto no próprio gabarito PDA (item 3.8.2), como nas atividades da macroatividade “Elaborar PDA” da GPA, descritas no item 3.5.1.
7 – O plano para o projeto de *software* é documentado.	Vide artefato PDA (item 3.8.2).
8 – São identificados os produtos de trabalho de *software* necessários para estabelecer e manter o controle do projeto de *software*.	Não coberto. Requer uma adaptação do LPA para a KPA de Gerência de Configuração de *Software*.
9 – As estimativas de tamanho dos produtos de *software* (ou de mudanças no tamanho destes produtos) são obtidas a partir de um procedimento documentado.	As estimativas a serem feitas são descritas na atividade “Levantar Estimativas” da macroatividade “Elaborar PDA” da PGA (item 3.5.1) e são documentadas no artefato PDA (item 3.8.2).
10 – As estimativas de esforço e custo do projeto de *software* são obtidas a partir de um procedimento documentado.	As estimativas a serem feitas são descritas na atividade “Levantar Estimativas” da macroatividade “Elaborar PDA” da PGA (item 3.5.1) e são documentadas no artefato PDA (item 3.8.2).
11 – As estimativas dos recursos críticos computacionais do projeto são obtidas a partir de um procedimento documentado.	Vide atividades da macroatividade “Elaborar PDA” da GPA (item 3.5.1), onde se descreve o que vêm a ser estes recursos, que devem ser documentados no PDA.
12 – O cronograma do *software* do projeto é obtido a partir de um procedimento documentado.	O LPA cita a necessidade de utilização de um cronograma de desenvolvimento do projeto em diversas atividades das macroatividades da PGA.
13 – São identificados, avaliados e documentados os riscos do *software*	Os riscos são identificados, avaliados e acompanhados na atividade “Identificar e

associados ao custo, recurso, cronograma e aspectos técnicos do projeto.	Avaliar Riscos" da macroatividade "Elaborar PDA" da GPA (item 3.5.1).
14 – São preparados planos para as facilidades de desenvolvimento de *software* do projeto, bem como ferramentas de suporte.	O plano de desenvolvimento é documentado no artefato PDA (item 3.8.2), que também descreve as ferramentas de *hardware* e *software* que serão necessárias.
15 – São registrados os dados de planejamento de *software*.	Os dados de planejamento são registrados nos artefatos Cronograma (item 3.8.3) e PDA (item 3.8.2), incluindo estimativas e valores reais de tamanho, recursos, esforço, prazos e custos.

Medições	
SW-CMM-2	LPA
1 – São realizadas medições para se gerenciar o *status* das atividades da gestão de planejamento de *software*.	Vide atividade "Acompanhar PDA" da macroatividade de mesmo nome da PGPA (item 3.5.2). As medições são registradas no artefato PDA (item 3.8.2).

Verificações	
SW-CMM-2	LPA
1 – As atividades da gestão de planejamento do projeto de *software* são revisadas periodicamente com a gerência sênior.	A Gerência Sênior do LPA é desempenhada pela Gerência de Projeto. Esta verificação é feita durante reuniões periódicas descritas na atividade "Acompanhar PDA" da macroatividade de mesmo nome da GPA (item 3.5.2).
2 – As atividades da gestão de planejamento do projeto de *software* são revisadas periodicamente e, eventualmente, com a gerência de projeto.	Esta verificação é feita durante reuniões periódicas e eventuais descritas na atividade "Acompanhar PDA" da macroatividade de mesmo nome da GPA (item 3.5.2).
3 – As atividades e produtos de trabalho da gestão de planejamento do projeto de *software* são revisadas e/ou auditadas pelo grupo de garantia da qualidade, e ao final os resultados são divulgados.	O LPA não cobre a KPA de Garantia da Qualidade de *Software*.

3. KPA da Gestão de Acompanhamento – SPTO

Compromissos	
SW-CMM-2	LPA
1 – É designado um gerente do projeto de *software* para ser o responsável pelas atividades de *software* do projeto e pelos resultados obtidos.	Vide responsabilidades da Gerência de Projeto (item 3.6.2) e do Líder de Projeto (item 3.6.6). A Gerência responde pelo projeto ao Cliente e o Líder de Projeto responde pelo projeto à Gerência de Projeto.
2 – O projeto segue uma política organizacional escrita para a gestão de projeto de *software*.	Vide artefato PGPA (item 3.8.1) onde são descritas as políticas da GPA.

Habilidades	
SW-CMM-2	LPA
1 – O plano de desenvolvimento de *software* é documentado e aprovado.	O plano é documentado nos artefatos PDA (item 3.8.2) e Cronograma (item 3.8.3). Ambos são revisados pelo Líder de Projeto ou pela Gerência de Projeto, dependendo do momento. Ao término desta revisão, estes artefatos podem ser considerados aprovados.
2 – A gerência de *software* do projeto atribui de forma explícita as responsabilidades para as atividades e produtos de trabalho de *software*.	Vide atividade “Iniciar Projeto” da macroatividade “Elaborar PDA” da GPA (item 3.5.1). Além disto, o Cronograma (item 3.8.3) do projeto descreve as responsabilidades pelas atividades a serem realizadas.
3 – São providos recursos e verbas para o acompanhamento do projeto de *software*.	Vide artefato PGPA (item 3.8.1) que define as políticas a serem seguidas pela GPA.
4 – As gerências de *software* são treinadas para saberem lidar com os aspectos técnicos e pessoais do projeto de *software*.	Vide atividade “Acompanhar PDA” da macroatividade de mesmo nome da GPA (item 3.5.2). São treinados a Gerência de Projeto e o Líder de Projeto.
5 – As gerências de primeira linha recebem orientações quanto aos aspectos técnicos do projeto de *software*.	Esta gerência é acumulada pela Gerência de Projeto, sendo esta que inicia os contatos com o Cliente, estabelece o contrato para o projeto pelo artefato Visão (item 3.7.4), aloca os requisitos de *software* (item 3.4.1) e inicia e acompanha o planejamento do mesmo (item 3.5.2). Vide atividades “Iniciar Projeto” (item 3.5.1), “Produzir PI” (item 3.4.1), “Alocar Requisitos de *Software*” (item 3.4.1), “”Levantar Estimativas” e “Tomar Ações Corretivas” (item 3.5.2).

Atividades	
SW-CMM-2	LPA
1 – É utilizado um plano documentado de desenvolvimento de *software* para	O acompanhamento é feito com o artefato PDA (item 3.8.2) e Cronograma (item 3.8.3).

acompanhar as atividades de *software* e comunicar o seu *status*.	
2 – O plano de desenvolvimento de *software* é revisado (atualizado) de acordo com um procedimento documentado.	A atividade “Acompanhar PDA” da macroatividade de mesmo nome e da GPA descreve este procedimento (item 3.5.2).
3 – Os compromissos do projeto de *software* e as alterações de compromissos assumidos com pessoas e grupos externos à organização são revisados em conjunto com a gerência sênior, de acordo com um procedimento documentado.	A Gerência de Projeto acumula o papel da Gerência Sênior. Vide atividade “Negociar Compromissos” (item 3.5.1). A Gerência de Projeto deve ser comunicada e avaliar as mudanças de compromissos externos do projeto. Vide também responsabilidades da Gerência de Projeto (item 3.6.2) para acompanhamento de projeto.
4 – As alterações em compromissos aprovados que afetem o projeto de *software* são comunicadas aos membros do grupo de desenvolvimento e a outros grupos de *software* relacionados.	As mudanças de compromissos devem ser negociadas e revisadas pela Gerência de Projeto. É uma política definida no artefato PGPA (3.8.1). Isto também está previsto na atividade “Tomar Ações Corretivas” da macroatividade “Acompanhar PDA” da GPA (item 3.5.2).
5 – São acompanhados os tamanhos dos produtos de trabalho (ou tamanho das mudanças nestes produtos de trabalho) e tomadas ações corretivas quando necessário.	Vide atividade “Acompanhar Estimativas” da macroatividade “Acompanhar PDA” da GPA (item 3.5.2). Devem ser acompanhados tamanho, esforço, custo e prazos. A atividade “Tomar Ações Corretivas” da mesma macroatividade trata sobre o que deve ser feito caso este acompanhamento identifique mudanças de rumo do projeto.
6 – São acompanhados os custos e esforços de *software* do projeto e tomadas ações corretivas quando necessário.	Vide atividade “Acompanhar PDA” da macroatividade de mesmo nome do PGA (item 3.5.2). Este acompanhamento é feito de forma periódica ou eventual. Caso o projeto saia do rumo, a atividade “Tomar Ações Corretivas” da mesma macroatividade descreve o que deve ser feito.
7 – São acompanhados os recursos críticos computacionais do projeto e tomadas ações corretivas quando necessário.	Vide atividade “Acompanhar PDA” da macroatividade de mesmo nome do PGA (item 3.5.2). Este acompanhamento é feito de forma periódica ou eventual. Caso o projeto saia do rumo, a atividade “Tomar Ações Corretivas” da mesma macroatividade descreve o que deve ser feito.
8 – É acompanhado o cronograma de *software* do projeto e tomadas ações corretivas quando necessário.	Vide atividade “Acompanhar PDA” da macroatividade de mesmo nome do PGA (item 3.5.2). Este acompanhamento é feito de forma periódica ou eventual.
9 – São acompanhadas as atividades técnicas de desenvolvimento e tomadas ações corretivas quando necessário.	Vide atividade “Acompanhar PDA” da macroatividade de mesmo nome do PGA. Este acompanhamento é feito de forma

	periódica ou eventual (item 3.5.2).
10 – São acompanhados os riscos de *software* do projeto, associados com o custo, recursos, cronograma e aspectos técnicos.	Vide atividade “Acompanhar o PDA” da macroatividade “Acompanhar PDA” da GPA onde os riscos estimados são acompanhados e verifica-se o surgimento de novos (item 3.5.2).
11 – São registrados os dados de replanejamento e os dados reais das medições do projeto de *software*.	Estes dados são registrados no artefato PDA conforme descreve a atividade, contida no Planejamento e Replanejamento da macroatividade “Elaborar PDA” (item 3.5.1) e na atividade “Acompanhar PDA” da macroatividade “Acompanhar PDA” (item 3.5.2), ambos pertencentes à GPA.
12 – O grupo de desenvolvimento de *software* conduz revisões internas periódicas para acompanhar o progresso técnico, os planos, o desempenho e outras questões, de acordo com o plano de desenvolvimento de *software*.	As revisões internas são feitas segundo periodicidade definida pelo Líder de Projeto, conforme descreve a atividade “Acompanhar PDA” da macroatividade de mesmo nome da GPA (item 3.5.2).
13 – São conduzidas revisões formais em marcos selecionados do projeto, para verificar as realizações e tratar os resultados do projeto de *software* de acordo com um procedimento documentado.	As revisões formais são feitas nos finais de iteração ou fase, segundo critérios do Líder de Projeto, conforme descreve a atividade “Acompanhar PDA” da macroatividade de mesmo nome da GPA (item 3.5.2).

Medições	
SW-CMM-2	LPA
1 – São realizadas medições para se determinar o *status* das atividades da gestão de acompanhamento e supervisão de *software*.	Vide atividade “Acompanhar PDA” da macroatividade de mesmo nome da GPA (item 3.5.2). As medições são registradas no artefato PDA (3.8.2).

Verificações	
SW-CMM-2	LPA
1 – As atividades da gestão de acompanhamento e supervisão de *software* são revisadas periodicamente com a gerência sênior.	A Gerência Sênior do LPA é desempenhada pela Gerência de Projeto. Esta verificação é feita durante reuniões periódicas descritas na atividade “Acompanhar PDA” da macroatividade de mesmo nome da GPA (item 3.5.2).
2 – As atividades da gestão de acompanhamento e supervisão de *software* são revisadas periodicamente e eventualmente com a gerência de projeto.	Esta verificação é feita durante reuniões periódicas e eventuais descritas na atividade “Acompanhar PDA” da macroatividade de mesmo nome da GPA (item 3.5.2).
3 – As atividades e produtos de trabalho da gestão de acompanhamento e supervisão de *software* são revisadas e/ou auditadas pelo grupo de garantia da qualidade, e ao final, os resultados são divulgados.	O LPA não cobre a KPA de Garantia da Qualidade de *Software*.

APÊNDICE B – QUESTIONÁRIO DE AVALIAÇÃO DO LPA

Questionário de avaliação do processo LPA

Parte 1

1. O Líder de Projeto e a Gerência de Projeto tiveram maior controle do projeto?
2. Os cronogramas estão sendo atendidos adequadamente?
3. O tempo gasto com o uso do processo (*overhead*) comprometeu o desenvolvimento do projeto?
4. O processo foi seguido pela organização?
5. Os projetos estão sendo melhor planejados?
6. Qual foi o grau de facilidade no entendimento do processo?
7. O cliente percebeu maior qualidade do serviço prestado e ficou mais satisfeito com os resultados?
8. Qual foi a quantidade de replanejamentos produzidos?
9. Ocorreram muitas mudanças nos requisitos do sistema em desenvolvimento?
10. Houve muita resistência das pessoas durante o processo de implantação do LPA? Quais?
11. O processo é muito burocrático?
12. Você acha que deve continuar utilizando o processo?
13. Você acha que o processo precisa ser alterado? Onde?
14. O que mais lhe agradou no processo?

Parte 2

1. O Líder de Projeto e a Gerência de Projeto estão com maior controle do projeto?
2. Os compromissos definidos no cronograma do projeto estão sendo atendidos no prazo?
3. O tempo gasto com o uso do processo (*overhead*) comprometeu o desenvolvimento do projeto?
4. O processo é seguido pela organização?
5. Os projetos são melhor planejados?
6. O processo é de fácil entendimento?
7. O cliente percebe maior qualidade do serviço prestado?
8. O cliente demonstra maior satisfação com os resultados apresentados?
9. Você sentiu que ofereceu muita resistência durante a implantação do processo devido às mudanças nos hábitos que você possuía antes de utilizá-lo?
10. O processo é muito burocrático?
11. Você continuaria utilizando o processo?
12. Você acha que o processo, por ser navegável, facilita a sua utilização?
13. Você acha que os gabaritos do LPA facilitam e motivam as pessoas a usarem o processo?
14. Você percebeu que tem mais controle dos requisitos do projeto?
15. Você acha que as diretrizes de planejamento definidas pelo LPA atendem às necessidades do projeto?

Nome do respondente: _____

Para cada pergunta, marque um “x” na coluna que melhor corresponda ao seu pensamento.

Pergunta	Não	+/-	Sim	Não foi possível avaliar
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				

APÊNDICE C – CONTEÚDO EM CD

Conteúdo deste CD:

- Dissertação do mestrado em:
  - \LPA_CD_Client\Dissertacao\LPA.pdf
- Gabaritos do LPA (formatos MSProject, MSWord e Acrobat) em:
  - \LPA_CD_Client\LPA_Gabaritos
- Processo LPA navegável em:
  - \LPA_CD_Client\LPA_Navegavel\index.htm – Impossibilidades de execução do processo devem ser reportadas para o correio eletrônico do autor afonso_soares@hotmail.com