3
Processo Proposto

3.1.
Características Desejadas

Por ser um modelo de processo extremamente aderente ao ambiente de pequenas equipes [BECK00], escolhemos como ponto de partida o XP, e adotamos alguns conceitos estabelecidos pelo RUP [BOOCH99], como a definição de múltiplos processos baseados em um fluxo de atividades que recebem artefatos como entrada ou geram artefatos em sua saída, de forma a conseguir um modelo de processo ágil mais prescritivo e controlável.

Analisamos a seguir algumas questões e lacunas verificadas no XP – modelo no qual o processo aqui proposto foi originalmente inspirado - e desejamos chegar a um processo mais completo, prescritivo e controlável, sem incorrer em um nível de complexidade desnecessário, minimizando ao máximo possível o custo de processo.

Um conceito útil aqui é o de *custo de processo*. Definimos como custo de processo o total de recursos alocados para realização de atividades que não têm como saída direta artefatos que representam o objetivo final do projeto.

Com o objetivo de caracterizar formalmente o *custo de processo*, consideraremos que o *custo total* do projeto é uma realização de uma variável aleatória, e que só pode ser medido de fato ao final do projeto. Sabemos porém que nem tudo sempre ocorre como planejado, e que os riscos de projeto devem ser considerados. Definiremos portanto o *custo ideal* e o *custo do dano* como variáveis auxiliares para nos ajudar a definir o custo total em termos estatísticos. O *custo ideal* representa a componente não aleatória do *custo total*. O custo ideal pode ser visto portanto como o custo total do projeto “se tudo correr da melhor forma possível”; ou seja, se nenhum risco de projeto se consilidar. O *custo do dano* representa a média estatística dos custos incorridos na realização dos riscos do projeto, e serve como um estimador da componente aleatória do custo total *a priori*.

Temos portanto:

$$\tilde{c}_{total} = c_{ideal} + \tilde{c}_{dano} = c_{ideal} + \sum c^{i}_{risco}\, p^{i}_{risco} \; ;$$

$\forall i$

onde:

$\tilde{c}_{total}$: custo total estimado

c_{ideal} : custo ideal

$\tilde{c}_{dano}$: custo do dano estimado (a média estatística é usada como estimador)

c^{i}_{risco} : custo decorrente da realização do i-ésimo risco

p^{i}_{risco} : probabilidade de ocorrência do i-ésimo risco

Assumiremos aqui que o objetivo final de qualquer projeto que utilize o processo aqui proposto é o desenvolvimento de um sistema de software satisfatório aos requisitos reais com um *custo total estimado* mínimo. O "objetivo final do projeto" considerado aqui é portanto o objetivo do cliente e dos *stakeholders* do projeto. Baseamos este conceito na própria definição de qualidade. O IEEE define qualidade como "o grau em que um sistema, componente ou processo satisfaz as necessidades e expectativas de seu cliente ou usuário". Existem diversas definições, mas qualquer que seja a definição escolhida, estão todas intimamente relacionadas ao conceito de adequação ao uso proposta por Juran [JURAN88]. Esta conceituação de "objetivo final do projeto" é bastante eficaz e preciso em estabelecer quais artefatos constituem um custo colateral necessário e quais constituem parte do objeto final, que é diferente para cada projeto.

Temos normalmente uma atividade diretamente relacionada a este objetivo, que é a codificação do sistema em si. Em alguns casos particulares, por exemplo, é um requisito exigido pelo cliente a criação de uma documentação técnica detalhada que permita a manutenção futura por terceiros, ou um conjunto de testes

que garanta o funcionamento correto do sistema em qualquer ambiente de execução. Nestes casos específicos, a criação de documentação ou a codificação de testes automatizados para os fins específicos requisitados pelo cliente podem ser consideradas atividades primárias, e não contabilizam como custo processual. Qualquer outra atividade que não seja requisito direto do cliente será daqui em diante classificada como uma atividade secundária. O somatório dos recursos consumidos por todas as atividades secundárias é definido como *custo do processo*. Desejamos um processo minimalista, que apresente baixo custo de projeto segundo esta definição.

Entendemos que a codificação de testes automatizados é importante para assegurar a qualidade de engenharia de um sistema, e que tal qualidade deva constar sempre como um requisito exigido pelo cliente, mas assumiremos a codificação de testes automatizados como uma atividade geradora de custo processual caso o conjunto de artefatos de testes em si não conste na especificação de requisitos. Mesmo admitindo que na maioria das vezes a codificação do sistema possa ser a única atividade que não constitui custo processual, e considerando que esta atividade constitui algo em torno de 12% [BOEHM81] do custo total de projetos comuns, manteremos esta definição. Deve-se esclarecer, porém, que o objetivo ao instanciar o processo não é meramente minimizar o custo processual. O minimalismo a que nos referimos aqui diz respeito a encontrar o ponto onde a relação entre a qualidade do produto final e o custo processual é ótima.

Apresentamos a seguir uma análise crítica do XP que servirá como justificativa para a escolha dos aspectos herdados deste processo pelo processo proposto.

3.1.1. Análise Crítica do XP

Em alguns ambientes, a adoção de métodos ágeis pode trazer resultados positivos significativos, mas estes não são aplicáveis a qualquer ambiente ou organização. Ambientes especialmente favoráveis à adoção deste tipo de metodologia são aqueles onde a comunicação é facilitada. Equipes

excessivamente grandes são problemáticas, pois a comunicação "360º" torna-se difícil. O tamanho e disposição do espaço físico também influem na comunicação.

Cockburn [COCKBURN02] afirma que organizações que desejem adotar métodos ágeis devem primar pela apresentação de resultados de qualidade, muito mais do que primar pelos planos estabelecidos. Embora pareça uma afirmação um tanto presunçosa, ele procura com ela expor as diferenças profundas entre a adoção de métodos ágeis e de outros modelos que têm ênfase em normalizar o processo de desenvolvimento como meio para garantir a qualidade.

Métodos ágeis em geral são especialmente indicados em projetos onde a volatilidade e dificuldade de definição dos requisitos na fase inicial do projeto [LINDVALL01] apresentem um desafio.

Mas mesmo para aqueles que se encaixam ou que desejam se adaptar para a adoção de um método ágil, estes não representam uma "bala de prata", como apontada por Brooks [BROOKS87]**.** As metodologias podem e devem ser adaptadas à realidade de cada ambiente e de cada projeto específico.

São apresentadas a seguir algumas das questões que podem representar riscos e com os quais devemos nos preocupar ao optar pela adoção de um método ágil. Tais questões são conclusões retiradas da experiência prática apresentada como estudo de caso no capítulo quatro deste trabalho, e representam os pontos problemáticos que nos motivaram a criar o modelo de processo mais prescritivo apresentado aqui.

3.1.1.1. Ênfase na Modelagem a Partir dos Requisitos

Métodos ágeis tendem a colocar grande parte das atividades de alto nível sob responsabilidade dos desenvolvedores, como, por exemplo, o levantamento de requisitos e a modelagem. Diferente de analistas, desenvolvedores tendem a ter uma visão do sistema voltada para as questões meramente tecnológicas. É necessário primar pelas expectativas do cliente, muito mais do que pelos méritos técnicos no desenvolvimento do sistema.

3.1.1.2. Inexistência de um Modelo Documentado

Quando for usado XP, por exemplo, não existe um modelo documentado para referência de toda a equipe. A manutenção da consistência do modelo usado é um resultado natural de um conjunto de práticas aplicadas conjuntamente, como refatoramento, testes de unidade e testes de integração freqüentes. O problema desta abordagem é que os modelos tendem a ser concebidos levando em conta estritamente questões locais (considerando somente poucas historietas e/ou poucas unidades), sem considerar aspectos de alto nível do modelo.

Ainda assim, pode ser interessante ter um modelo inicial, sem que isso impeça sua evolução e alterações ao longo da implementação. Partir de um modelo concebido antes do início da implementação ajuda a equipe a visualizar o que tem que ser feito. Garantir a evolutibilidade deste modelo e mantê-lo devidamente documentado também ajuda a equipe a compreender o que está sendo feito sob um ponto de vista mais abrangente, e quais as abordagens sendo utilizadas para fazê-lo.

3.1.1.3. Questões Contratuais

Diagramas de WBS ou GANTT são difíceis de obter quando se estiver utilizando uma metodologia como XP. Embora existam modelos propostos para contratos de escopo aberto [BECK99], poucas são as organizações que podem se dar ao luxo de contratar um projeto sem saber de antemão o total de recursos que deve ser reservado para desenvolver o sistema, sem uma especificação clara do que será obtido no final, contratando somente uma quantidade determinada de recursos, em horas-desenvolvimento, ficando à mercê da eficiência e produtividade da equipe de desenvolvimento contratada.

Além da questão do escopo, outras exigências contratuais comuns podem ser difíceis de se cumprir em um projeto ágil, por exemplo, a geração de documentação técnica.

3.1.1.4.
Cliente Presencial

Dois fatores aqui podem caracterizar problemas, dependendo da natureza do projeto:

- Assumir que um único cliente possa representar os interesses de todos os *stakeholders* do projeto
- Conseguir que o cliente disponibilize uma pessoa de sua força de trabalho, preferencialmente um *expert* no universo de discurso em tempo integral para a realização do projeto.

3.1.1.5.
Falta de Rigor na Definição dos Processos

Manter uma boa comunicação requer mais do que uma sala pequena cheia de programadores sentados aos pares. O minimalismo exagerado do XP pode levar à confusão, ambigüidade e inconsistência. XP parece depender de uma equipe brilhante e disciplinada, completamente comprometida com os interesses do cliente. Faltam processos bem definidos que possam ser seguidos mesmo por uma equipe mediana, e que possam ser controlados e avaliados.

Verificamos que abordagens como aquelas propostas por metodologias ágeis são bastante aderentes a projetos com pequenas equipes, mas encontramos dificuldades quando tentamos sistematizar a implantação do processo e a institucionalização das práticas contínuas nele descritas. Isso ocorre porque *eXtreme Programming* é quase totalmente baseado em práticas, que são atividades que devem ser conduzidas continuamente ou em momentos específicos. Por esta característica dizemos que XP depende fortemente do talento, disciplina e comprometimento individual dos membros da equipe de desenvolvimento, e oferece pouca estrutura processual formal para garantir tal comprometimento. Mas nem sempre dispomos facilmente de grandes talentos para os projetos em um ambiente realista.

Práticas não são atividades. Pela definição do modelo PEP adotada aqui, atividades são realizadas por papéis, tais como as práticas. Mas atividades também estão relacionadas a um contexto e a artefatos de entrada e de saída bem definidos. Tomemos como exemplo a prática “refatorar constantemente”, adotada pelo XP.

Ela regula a sistemática de trabalho que os desenvolvedores devem seguir, e relaciona-se com outras práticas através de uma relação de dependência – para que determinada prática seja implementada, outras práticas também devem ser implementadas concomitantemente – mas não é representada de forma concreta dentro de um fluxo de atividades. A conseqüência disso é uma falta de formalismo do processo de trabalho em si. Fica muito difícil para um gerente de equipe garantir de forma precisa se uma prática está efetivamente implementada dentro do processo.

Porém, tais práticas são justificáveis no ambiente de uma equipe pequena. Uma das características de metodologias ágeis vista na seção 2 diz respeito ao enfoque nas pessoas em contrapartida ao enfoque em processos. Tom Mochal [W_MOCHAL03] defende a idéia de que processos para projetos pequenos devem ser baseados em práticas relacionadas com a manutenção da qualidade e menos focados em processos. Isto ocorre porque processos baseados em práticas apresentam um custo de processo menor que processos que apresentam um fluxo seqüencial de atividades.

Temos portanto um conflito: desejamos um processo ágil, de pouco custo, mas que possa ser avaliado e controlado da mesma forma que um processo definido de maneira mais formal. A forma com que abordaremos esta questão no processo proposto a seguir é muito simples: continuaremos fazendo uso de práticas, mas teremos processos formalmente definidos para verificação da efetividade e eficácia da adoção destas práticas.

3.1.1.6. Forte Dependência do Conhecimento Tácito da Equipe

A dinâmica de trabalho de um projeto XP e a falta de documentação causam uma forte dependência do sucesso do projeto ao conhecimento tácito da equipe de desenvolvimento. Toda a documentação gerada se limita ao código gerado. O entendimento do macro modelo se dá através de práticas como a propriedade coletiva e programação em duplas. A consistência é garantida através da adoção de padrões, da integração contínua e dos testes automatizados. A comunicação é muito valorizada e isso garante que um conhecimento profundo sobre todas as questões do projeto seja compartilhado de forma equânime por toda a equipe.

Mas projetos eventualmente chegam ao fim e existe uma inevitável renovação da equipe de desenvolvimento, ainda que seja possível manter essa renovação em um ritmo lento. Projetos são realizados o tempo todo, e a tendência da equipe é voltar o intelecto totalmente à realização do projeto em mãos. Como se daria então a manutenção ou extensão de funcionalidades de um projeto realizado anteriormente? Não havendo documentação, e com uma equipe possivelmente diferente, isso caracteriza um sério risco para o cliente. Boehm [BOHEM00] estabelece que a ausência de documentação técnica implica um aumento da ordem de 50% no custo total do projeto.

A solução proposta é um processo no qual os artefatos são definidos formalmente, e é estabelecido um modelo para eles no início de cada projeto. Dentre estes artefatos figuram modelos do sistema. Veremos que o nível de detalhamento destes modelos é diferente para cada projeto executando este processo – ou seja, para cada instância do processo.

3.2. Visão Geral do Processo Proposto

O processo apresentado nesta seção faz uso extenso dos valores e práticas do XP, e é fortemente baseado neste modelo de processo. É composto de valores, e práticas (como o XP), mas também define um repositório de artefatos, papéis e processos.

Para representação do processo proposto, utilizaremos os diagramas definidos no repósitório da notação definida pelo *framework* PEP, descrito na seção 2.

Definiremos um conjunto de processos formalmente descritos que ofereçam suporte concreto às práticas propostas pelo XP. Veremos que os processos têm por objetivo a supervisão, inspeção e validação das práticas. Isso torna o processo mais controlável e transfere responsabilidade que no XP está espalhada verticalmente por toda a equipe de desenvolvimento para processos bem definidos.

Nem todas as práticas definidas pelo XP serão adotadas, sendo que algumas delas foram substituídas por unidades de processos (ver seção 2.4.1.2). Da mesma forma, alguns papéis e valores foram substituídos. Não podemos portanto afirmar

que o processo aqui descrito caracterize-se como uma extensão do XP. Tomamos o XP como base para criação do processo híbrido aqui proposta. A razão para isso reside no fato de que a metodologia proposta surgiu em um ambiente real de uma pequena equipe de desenvolvimento, que em um primeiro momento adotou o XP como alternativa para o processo de desenvolvimento. Este processo adotado inicialmente evoluiu e foi alterado, resultando na metodologia que descrevemos nesta seção.

Uma questão central na estruturação do processo proposto diz respeito a escolher um conjunto de práticas e atividades consistentes e complementares. Alguns resultados [MACCORMACK03] sugerem que certos conjuntos de práticas, quando adotadas em conjunto, colaboram de forma efetiva para a melhoria da qualidade e aumento da produtividade ao longo do desenvolvimento. Em outras palavras, certas práticas são complementares e só representam uma vantagem quando adotadas conjuntamente. Outras práticas são mutuamente excludentes, não representando vantagem real sua adoção conjunta. Escolher um conjunto de práticas e atividades consistente representa portanto um desafio na elaboração de um processo híbrido. Embora o bom senso seja geralmente uma ferramenta útil nesta escolha, devem ser considerados também resultados obtidos por estudos e confirmados por boas práticas consolidadas na indústria. O estudo de caso desenvolvido na seção 4 busca validar, ainda que sem formalismo estatístico, o conjunto de práticas e a estrutura de atividades do processo proposto.

Embora prescritivo, o processo proposto pode ser adaptado a projetos de qualquer natureza, desde que estes projetos não envolvam equipes com mais do que algo em torno de 10 pessoas. Para adaptar o processo, deve-se somente alterar o nível de detalhamento dos artefatos gerados.

3.2.1. Valores

3.2.1.1. Minimalismo

O *design* do software deve ser continuamente simplificado. O processo em si também é adaptado, a cada dia, se alguém observar como torná-lo mais simples. Nunca adicionar um artefato ou tarefa desnecessária. Ao realizar a implementação

muitas vezes somos tentados a adotar uma solução mais complexa por puro perfeccionismo, ainda que o resultado final seja o mesmo, que a manutenibilidade e reusabilidade do código não seja aumentada de forma realmente impactante na prática e que isso implique em mais horas de trabalho. Esse tipo de escolha deve ser evitada.

3.2.1.2. Comunicação

Métodos simples de comunicação são desejáveis. Deve-se preferir chat a eMail, telefonemas a chat, conversar pessoalmente a telefonemas, trabalhar na mesma sala a ter salas isoladas, trabalhar em conjunto a revisar o resultado final. Evitar ao máximo a comunicação através de artefatos. A comunicação através de artefatos é útil somente para equipes grandes, onde a comunicação informal é dificultada.

3.2.1.3. Feedback

Ao longo do desenvolvimento, uma lista de pendências é criada. Todas as pendências devem ser atendidas antes do fim da iteração. Este ciclo está descrito nos fluxogramas das Unidade de Processos, mas ter feedback como um valor implica mais que simplesmente seguir o fluxograma: onde possível, as pendências devem ser atendidas com uma frequencia e granularidade ainda maior. Todo problema deve ser evidenciado o mais cedo possível para que possa ser corrigido o mais cedo possível. Toda oportunidade deve ser descoberta o mais cedo possível para que possa ser aproveitada o mais cedo possível.

3.2.2. Práticas Contínuas

Práticas contínuas são atividades que não constam no fluxograma das Unidades de Processo por serem realizadas com uma freqüência indepentente do andamento do fluxograma. Algumas são realizadas diariamente, outras com uma freqüência ainda maior. Todas elas são executadas somente durante a execução da

tarefa de implementação, realizada dentro da Unidade de Processo “Executar Iteração”.

3.2.2.1. Reuniões em Pé

Esta prática consiste em uma reunião diária de toda a equipe de projeto, na qual é discutido o que foi desenvolvido no dia anterior e o que será desenvolvido a seguir. O nome reunião “em pé” (“*stand-up meeting*”) sugere que a reunião seja efetuada com os participantes em pé, em torno de uma mesa ou quadro branco. A razão para isso é motivar que esta seja uma reunião rápida e objetiva. Esta é uma prática herdada de outras metodologias ágéis, como Extreme Programming [BECK00] e SCRUM [POWER02].

3.2.2.2. Propriedade Coletiva do Código

A equipe como um todo é responsável por cada arquivo de código. Não é preciso pedir autorização para alterar qualquer arquivo. Em uma equipe pequena, onde a comunicação é facilitada, qualquer alteração que se faça necessária em um artefato pode ser feito pelo próprio programador. Isso torna o processo menos oneroso, na medida que torna desnecessário o custoso vai-e-vem de requisições de alterações comuns em processos adotados por equipes mais numerosas.

A Gerência de Configuração, ou *Software Configuration Management (SCM)* está incorporada nesta prática, porém não está totalmente relegada a ela. Exatamente como o XP [EKMAN04]**,** o processo proposto não apresenta um processo específico de gerência de configuração, mas esta é realizada ao longo das Unidades de Processo descritas e suportada também pelas práticas contínuas, principalmente por esta.

3.2.2.3. Propriedade Coletiva da Documentação

Equivalentemente à propriedade coletiva de código, os modelos também podem ser alterados por qualquer pessoa quando necessário. Porém, todas as alterações são supervisionadas no processo de gerência de documentação. Esta

prática visa diminuir a necessidade de atividades de modelagem em baixo nível, pois na prática, a manutenção de modelos de baixo nível, como o diagrama de classes, tem forte interação com a própria atividade de implementação.

3.2.2.4. Refatoramento Constante

Esta pratica se refere principalmente ao refatoramento em ponto pequeno, mas mesmo necessidade de refatoramento estrutural pode ser verificada e executada pelos desenvolvedores a qualquer momento da implementação.

3.2.2.5. Uso de Padrões

Para tornar viável a propriedade coletiva do código, é necessário que seja definido e adotado um padrão de codificação. Isso contribui para a inteligibilidade do código, uma vez que a propriedade deste é coletiva.

Também devem ser adotados padrões de arquitetura, com o objetivo de garantir uma abordagem tecnológica razoavelmente padronizada entre os projetos e entre os componentes dentro de um mesmo projeto.

3.2.3. Definição Formal do Processo Proposto

Nesta seção será apresentada uma definição formal do processo. O framework PEP [DAFLON03] será usado como notação para tal. Serão definidos os containers de Papéis, Artefatos, Atividades e de Unidades de Processo. As unidades de processo serão descritas em detalhe através de diagramas.

3.2.3.1. O Container de Papéis

Estão descritos abaixo os papéis neste container, com a descrição e responsabilidade de cada um deles. É importante ressaltar que não se faz necessário ter uma pessoa inteiramente dedicada a cada um destes papéis. Alguns dos papéis podem ser desenvolvidos por mais de uma pessoa, bem como uma pessoa pode desempenhar mais de um papel, dependendo do volume de trabalho e

importância deste papel em cada instante do projeto. O importante é que os responsáveis pelos papéis sejam formalmente definidos.

Quando neste trabalho se define a responsabilidade de um papel sobre uma atividade ou artefato, isso significa que a pessoa que exerce o papel em questão é que deve garantir a realização da atividade ou a existência do artefato. Nada impede, porém, que a responsabilidade seja delegada. Por exemplo, o gerente de testes pode eventualmente delegar testes de sistema aos desenvolvedores. Porém, estes devem se reportar ao gerente de testes e é ele que deve verificar e garantir que os testes tenham sido realizados da forma adequada e que seus resultados sejam considerados. Esta é uma peculiaridade do processo proposto e não encontra espaço para definição dentro da notação utilizada.

Cabe ressaltar que, tendo em vista o tamanho da equipe do projeto, o número de papéis formalmente definidos pelo processo é bastante reduzido em relação a processos como o Unified Process. Em comparação a este último, o processo aqui descrito agrega as responsabilidades co-relacionadas ou competências semelhantes de diferentes papéis de forma a obter um número menor de papéis sem abrir mão de responsabilidades e aspectos essenciais do processo. O Gerente de Implementação, por exemplo, tem responsabilidades de arquiteto, de responsável pela implementação e pela manutenção de toda a documentação técnica. Veremos adiante, na descrição das unidades de processo que as atividades são dispostas de forma a evitar a superposição – ou seja, a execução em paralelo – de atividades. Prevemos desta forma que o Gerente de Implementação pode trabalhar como desenvolvedor ao longo da codificação do sistema.

3.2.3.1.1. Gerente de Projeto

Diferente do XP, onde o planejamento é feito pela própria equipe de desenvolvimento, neste processo o gerente de projeto é o responsável pelo planejamento de cada iteração e etapa. Porém o plano de trabalho não é ditado e imposto à equipe, mas um trabalho colaborativo, onde o gerente de projeto deve coletar estimativas dadas pela própria equipe de desenvolvimento e aplicar as correções adequadas, de acordo com medidas baseadas na assertude das

estimativas dadas pela equipe em relação ao tempo real de execução de todas as tarefas executadas anteriormente no projeto.

Como responsável por garantir a execução do processo, ele supervisiona o cumprimento de todas as atividades e práticas definidas, além da extração de indicadores e controle do andamento do projeto

Responsabilidades
• Alocar e gerenciar os recursos • Gerenciar o cronograma • Gerenciar riscos do projeto • Garantir a correta execução das Unidades de Processo definidas • Registrar lições aprendidas • Promover a adoção concreta dos valores definidas pelo processo • Garantir o cumprimento das práticas contínuas definidas pelo processo
Requisitos
• Capacidade de Liderança • Organização • Conhecimento dos Aspectos Gerais de Processos de Desenvolvimento de Software • Domínio do Processo Proposto

Quadro 1 – Responsabilidades e requisitos do Gerente de Projeto

3.2.3.1.2. Analista

O analista desempenha um papel importantíssimo no processo de desenvolvimento: ele é o responsável pela comunicação entre o cliente e a equipe de desenvolvimento. Ele deve garantir a cada iteração que o que está sendo desenvolvido está de acordo com as expectativas do cliente, e, da mesma forma, que a equipe tenha um bom entendimento do que é desejado.

O analista é responsável pela elicitação dos requisitos, a elaboração de um documento de requisitos na forma de *wish list* ou outra tecnologia mais elaborada e o desdobramento destes em casos de uso, descrições de cenários e protótipos ou descrições da interface do sistema. É responsável por elaborar e garantir e

homologação destes artefatos de especificação junto ao cliente ao início do projeto e refinar a especificação ao início do desenvolvimento de cada release.

Ao fim do desenvolvimento de cada release, o analista é responsável por submetê-lo à aprovação do cliente, revisando e possivelmente alterando ou incluindo novos requisitos e casos de uso.

Responsabilidades
• Definir o escopo do sistema • Elicitar, analisar e documentar os requisitos do sistema de forma incremental • Manter um registro da evolução dos requisitos • Especificar funcionalidades do sistema através de protótipos, descrições de cenários ou modelos de interface • Especificar características do sistema derivadas de requisitos não funcionais • Garantir contínua consistência entre a especificação e a implementação
Requisitos
• Conhecimento de técnicas de elicitação e modelagem de requisitos • Grande capacidade de comunicação • Dinamismo e capacidade de compreender tanto aspectos do domínio do sistema quanto limitações e dificuldades técnicas para implementação do sistema.

Quadro 2 – Responsabilidades e requisitos do Analista

3.2.3.1.3. Arquiteto

O arquiteto lidera e coordena atividades e artefatos técnicos ao longo do projeto. Ele estabelece a estrutura em ponto grande do sistema: os componentes, a abordagem e os recursos tecnológicos adotados. O arquiteto é o responsável pela qualidade do código e dos componentes implementados. Ele deve, ao fim de cada iteração, verificar oportunidades de refatoramento no código e nos modelos técnicos documentados.

O arquiteto tem o papel de líder dentro da equipe de desenvolvedores. Ele tem a palavra final quanto a todas as questões técnicas relacionadas à tarefa de codificação.

Responsabilidades
• Definir abordagem tecnológica • Elaborar modelos abstratos • Avaliar modelos técnicos e identificar oportunidades de melhoria dos modelos e refatoramento do sistema em ponto grande a cada iteração • Garantir refatoramento constante do sistema em ponto pequeno • Garantir a gerência de configuração eficaz no escopo do projeto
Requisitos
• Modelagem de sistemas e linguagens de representação • Conhecimento técnico amplo

Quadro 3 – Responsabilidades e requisitos do Arquiteto

3.2.3.1.4. Cliente

Este papel é exercido pelo contratante, usuários e stakeholders do sistema. Exatamente como para os outros papéis, as pessoas que os exercem devem ser definidas e alocadas no início do projeto.

Em alguns processos de desenvolvimento, como o RUP, os clientes do sistema não são definidos como papéis da equipe, mas aqui optamos por fazê-lo, de forma a caracterizar bem suas responsabilidades e principalmente, definir formalmente as pessoas que integram este papel desde o início do projeto.

Diferente do papel homônimo definido pelo XP, o cliente aqui não tem uma atuação presencial em todo o processo de desenvolvimento. Porém, os usuários e stakeholders devem estar bem definidos e disponíveis para realizar as tarefas pelas quais são responsáveis: homologação de artefatos de especificação, homologação dos releases e definição dos requisitos do sistema.

Este papel normalmente é exercido por mais de uma pessoa, tendo em vista que é interessante agregar aqui os futuros usuários do sistema, o contratante – que paga pelo sistema, e até mesmo beneficiários indiretos. As responsabilidades e

requisitos, bem como as atividades exercidas por este papel são portanto características do coletivo.

Responsabilidades
• Estabelecer objetivos do sistema • Homologar especificação • Homologar produto final e releases intermediários
Requisitos
• Ter profundo conhecimento do domínio do sistema • Ter autoridade para homologar artefatos e releases

Quadro 4 – Responsabilidades e requisitos do papel “Cliente”

3.2.3.1.5. Desenvolvedor

O desenvolvedor é responsável pela codificação dos componentes e integração constante do sistema. No processo proposto, o desenvolvedor também tem permissão para alterar o design e a documentação técnica. Isso porque é freqüente, ao longo do desenvolvimento, surgirem questões não consideradas no momento da elaboração destes modelos. O desenvolvedor também é responsável por escrever testes automatizados e gerar documentação técnica de acordo com as diretrizes estabelecidas.

Responsabilidades
• Implementar e integrar componentes do sistema • Escrever e executar testes automatizados • Escrever documentação técnica do sistema • Identificar e executar oportunidades de refatoramento • Registrar e compartilhar conhecimento adquirido • Estimar custos de desenvolvimento do sistema
Requisitos
• Conhecimento das tecnologias e ferramentas utilizadas no processo • Conhecimento das linguagens de representação utilizadas nos artefatos de documentação do sistema.

Quadro 5 – Responsabilidades e requisitos do Desenvolvedor

3.2.3.1.6. Gerente de Documentação

O gerente de documentação é responsável pela elaboração e manutenção dos modelos do sistema e toda a documentação adicional, como manuais de usuário ou manuais de referência da equipe de desenvolvimento.

Na verdade, a manutenção dos modelos é feita pelo Analista e pelos Desenvolvedores ao longo de cada iteração. Ao Gerente de Documentação cabe inspecionar e supervisionar o trabalho de manutenção destes modelos, de forma a garantir que esta atividade esteja sendo efetivamente e corretamente realizada ao longo de cada iteração.

Também é responsabilidade do Gerente de Documentação estabelecer no início do projeto quais são os artefatos de documentação que devem ser desenvolvidos e a natureza destes documentos, junto com diretrizes gerais quanto à abrangência, nível de detalhamento e formato destes documentos.

Responsabilidades
• Definir artefatos de documentação gerados ao longo do projeto • Definir diretrizes para escrita dos documentos gerados • Inspecionar toda a documentação gerada e garantir que está de acordo com as diretrizes estabelecidas.
Requisitos
• Conhecimento de linguagens de representação • Conhecimento técnico dos recursos tecnológicos empregados no projeto

Quadro 6 – Responsabilidades e requisitos do Gerente de Documentação

3.2.3.1.7. Gerente de Testes

O gerente de testes é responsável por todos os testes, automatizados ou não. Os desenvolvedores são responsáveis por escrever os testes de unidade, mas estes devem fazê-lo de acordo com as diretrizes formalmente definidas pelo gerente de testes. Estas diretrizes podem ser casos de teste bem definidos em alguns casos, ou simplesmente requisitos do tipo “deve-se testar todos os métodos do artefato A,

tanto com dados válidos como com entradas espúrias” ou “para o artefato B, deve-se testar somente sua interface”.

O gerente de testes também é responsável por inspecionar os testes de unidade escritos pelos desenvolvedores e garantir que estes são abrangentes o suficiente. Caso os testes escritos não satisfaçam os critérios pré-estabelecidos ou não sejam satisfatórios, este deverá gerar relatórios simples onde conste o artefato e o caso de teste que apresenta problema, e o que deve ser alterado.

O gerente de testes não é responsável pelos testes de aceitação, que consistem numa forma de homologação dos releases do projeto, e são de responsabilidade do Cliente e realizados em conjunto com o Analista.

Responsabilidades
• Estabelecer diretrizes para implementação de testes autmatizados • Conceber casos de teste a cada iteração • Coordenar realização de *alfa-testes* • Registrar resultados de testes • Inspecionar implementação de testes automatizados a cada iteração, garantindo que estes estão corretos e de acordo com as diretrizes estabelecidas
Requisitos
• Experiência com concepção de casos de testes • Conhecimento tecnológico suficiente para analisar o projeto do sistema e definir escopo e metodologia de testes • Conhecimento de metodologias, recursos e ferramentas para testes

Quadro 7 – Responsabilidades e requisitos do Gerente de Testes

3.2.3.2. O Container de Artefatos

3.2.3.2.1. Visão-Escopo

O objetivo deste documento é alinhar as partes (cliente e equipe de desenvolvimento) quanto ao que será desenvolvido. Consiste numa descrição inicial do universo de discurso do sistema a ser desenvolvido, bem como uma descrição em alto nível de abstração das funcionalidades do sistema. Define a

visão dos *stakeholders* a respeito do sistema a ser implementado (necessidades e funcionalidades). Serve como contrato entre desenvolvedores e interessados.

3.2.3.2.2. Lista de Requisitos

Descreve uma lista dos requisitos funcionais e não funcionais do sistema. Os riscos técnicos e ações mitigatórias relacionados a cada requisito devem ser contemplados no Plano de Gestão de Riscos (ver ítem 3.2.3.2.5).

3.2.3.2.3. Modelo Abstrato

O modelo abstrato consiste numa visão geral da arquitetura e componentes do sistema a ser desenvolvido. Ele tem um papel importante nas etapas iniciais do projeto, servindo como instrumento para alinhamento da equipe do projeto em torno da solução a ser implementada quando os modelos refinados ainda não existem.

3.2.3.2.4. Plano de Alocação de Recursos

Este artefato consiste numa simples lista de toda equipe do projeto, indicando o(s) papel (ou papéis) exercidos por cada integrante.

3.2.3.2.5. Plano de Gestão de Riscos

Contém uma lista dos riscos técnicos e de processo do projeto. Cada ítem listado deve conter uma indicação de como atuar para evitar que ocorra, a probabilidade de ocorrência e qual seria o impacto caso o risco se consolide. Contém também as ações mitigatórias a serem realizadas para o caso da cosolidação de cada risco.

3.2.3.2.6.
Plano de Documentação

Este artefato descreve diretrizes para documentação do sistema. Ele define quais artefatos de código devem ser documentados e o nível de detalhamento e abstração a ser usado para cada um destes artefatos.

3.2.3.2.7.
Plano de Testes

O plano de testes define dois pontos básicos: quais artefatos e funcionalidades devem ser testados, com diretrizes gerais para teste de cada artefato, e um cronograma de execução dos testes. Podem ser incluidos aqui testes automatizados, testes manuais e testes de homologação. De forma geral, este documento consiste num conjunto de diretrizes para a concepção dos casos de testes.

3.2.3.2.8.
Padrões de Codificação

Estabelece padrões de codificação que devem ser obedecidos pelos desenvolvedores. Como todos os artefatos de código são de propriedade coletiva, estes padrões garantem a manutenibilidade dos artefatos por toda a equipe.

3.2.3.2.9.
Especificação do Ambiente de Projeto

Uma especificação das ferramentas e metodologias a serem usadas, tanto para o desenvolvimento quanto para a gestão do processo.

3.2.3.2.10.
Interface Abstrata

Consiste num modelo da interface final do sistema, com o objetivo de homologar a interface antes de iniciar a etapa de implementação. Pode ser um protótipo interativo ou um modelo de interface abstrata, que consiste numa visualização não-iterativa para cada caso de uso ou cenário especificado.

3.2.3.2.11. Cronograma

Este artefato consiste numa descrição das atividades a serem realizadas e o intervalo de tempo em que pretende-se realizá-las. Contém um plano de *releases* que define a parte do escopo a ser atacado em cada release. Este documento é criado de forma iterativa, de modo que a iteração corrente deve sempre ter um nível de detalhamento maior que as iterações subseqüentes. Um novo plano de iterações é criado na primeira iteração de cada release.

3.2.3.2.12. Modelos de Implementação

Um único documento contendo modelos de implementação do sistema, voltado a desenvolvedores responsáveis pela manutenção evolutiva do código em *releases* subseqüentes. Pode conter diagramas UML (de classe, de seqüência, etc) não abstratos (no nível do código implementado), diagramas de arquitetura e documentação descritiva para cada artefato de código gerado.

3.2.3.2.13. K-Log

Este artefato consiste num corpo de conhecimento utilizado e criado pela própria equipe de desenvolvimento evidenciando eventuais dificuldades técnicas encontradas ao longo da codificação. O objetivo é ter uma referência para soluções de problemas recorrentes ao longo da implementação. Podem ser documentadas questões de implementação, configuração ou mesmo sobre o ambiente de desenvolvimento e de produção.

3.2.3.2.14. Componentes

Estes artefatos compõem o código do sistema propriamente dito.

3.2.3.2.15. Scripts de Teste

São testes automatizados. Pode ser usada a própria linguagem de programação do sistema, linguagens de script ou frameworks específicos para tal.

Os scripts de teste originam-se de casos de teste, mas nem todos os casos de testes são implementados como scripts de teste.

3.2.3.2.16. Resultados de Testes

Consiste num relatório de execução dos testes (automatizados e manuais). Um *release* só estará pronto quando não houver falhas evidenciadas pelos testes neste relatório.

3.2.3.2.17. Laudo de Inspeção

Laudos de Inspeção contêm uma lista dos pontos reprovados por uma inspeção, que deverão ser corrigidos.

3.2.3.2.18. Plano de Medição

O plano de medição define as métricas a serem usadas ao longo de todo o projeto. Define também os momentos onde as métricas serão aferidas, estabelecendo limites aceitáveis e, quando aplicável, endereçando ações mitigatórias caso as medidas ultrapassem estes limites. As métricas a serem usadas são estabelecidas baseadas no plano de gerência de riscos, e as ações mitigatórias podem referenciar sempre este documento.

3.2.3.2.19. Medidas

Base histórica contendo as realizações das métricas aferidas ao longo do projeto, de acordo com o plano de medição.

3.2.3.3. O Container de Atividades

Atividade	Elaborar Documento de Visão	
Descrição	O propósito desta atividade é obter uma visão geral do sistema a ser desenvolvido, definindo as fronteiras do sistema, isto é, o que o sistema irá ou não atender. É elaborada uma descrição do problema e é estabelecido um acordo entre os desenvolvedores e interessados no sistema	
Papel	Analista	
Artefatos de Entrada		Artefatos de Saída
(1) Documentação do Domínio da Aplicação (opcional), (2) Sistemas Similares (opcional)		(1) Documento de Visão-Escopo
Tarefas	(1) Formular Descrição do Problema, (2) Identificar Interessados no Sistema, (3) Definir Domínio da Aplicação, (4) Nivelar Expectativas com o Cliente	

Atividade	Elicitar Requisitos	
Descrição	O propósito desta atividade é realizar um levantamento dos requisitos do sistema, baseado em reuniões com o cliente e no estudo do *universo de discurso* (ou "domínio do sistema").	
Papel	Analista	
Artefatos de Entrada		Artefatos de Saída
(1) Documento de Visão-Escopo, (2) Documentação do Domínio da Aplicação (opcional), (3) Sistemas Similares (opcional)		(1) Lista de Requisitos
Tarefas	(1) Efetuar Reunião com Cliente, (2) Registrar Pedidos dos Interessados, (3) Estudar Sistema Similares, (4) Estudar Domínio do Sistema	

Atividade	Homologar Requisitos	
Descrição	Esta atividade consiste na validação da lista de requisitos por parte do cliente.	
Papel	Cliente	
Artefatos de Entrada		Artefatos de Saída
(1) Lista de Requisitos		(1) Lista de Requisitos (homologada)
Tarefas	(1) Verificar Requisitos Inadequados, (2) Comunicar Pontos de Discordância	

Atividade	Elaborar Modelo Abstrato	
Descrição	O propósito desta atividade é a elaboração de um documento contendo um modelo técnico de alto grau de abstração do sistema a ser desenvolvido. Devem ser abordados as questões e recursos tecnológicos necessários.	
Papel	Arquiteto	
Artefatos de Entrada		Artefatos de Saída
(1) Documento de Visão-Escopo, (2) Lista de Requisitos		(1) Modelo Abstrato
Tarefas	(1) Analisar Requisitos do Sistema, (2) Efetuar Reunião com Provável Equipe de Desenvolvimento, (3) Elaborar Abordagem Tecnológica, (4) Levantar Requisitos Tecnológicos, (5) Elaborar Modelo Abstrato de Componentes do Sistema	

Atividade	Alocar Recursos	
Descrição	Esta atividade tem por objetivo definir a equipe de desenvolvimento, contratar pessoas com as competências necessárias para o desenvolvimento do projeto e definir as responsabilidades de cada membro da equipe na realização do projeto.	
Papel	Gerente de Projeto	
Artefatos de Entrada		Artefatos de Saída
Não Existe		(1) Plano de Alocação de Recursos
Tarefas	(1) Analisar Levantamento Inicial, (2) Definir Habilidades Necessárias, (3) Contratar Recursos Indisponíveis, (4) Alocar Papéis, (5) Treinar Equipe	

Atividade	Gerenciar Riscos	
Descrição	O objetivo desta atividade é definir os riscos do projeto, classificá-los de acordo com sua probabilidade de ocorrência e impacto no projeto, planejar formas de identificar a iminência de consolidação de cada ponto de risco e mapear ações mitigatórias para cada um destes pontos.	
Papel	Gerente de Projeto	
Artefatos de Entrada		Artefatos de Saída
Não Existe		(1) Plano de Gestão de Riscos, (2) Plano de Medição
Tarefas	(1) Realizar Reunião com Equipe, (2) Identificar Potenciais Riscos, (3) Classificar Potenciais Riscos, (4) Elaborar Plano de Gestão de Riscos, (5) Analisar Riscos do Projeto, (6) Definir Métricas, (7) Registrar Métricas num Documento de Referência	

Atividade	Planejar Documentação	
Descrição	O propósito desta atividade é estabelecer em linhas gerais os requisitos quanto à documentação do sistema. Deve-se levantar as expectativas e necessidades do cliente, bem como as necessidades de documentação por parte da equipe de desenvolvimento suficientes para garantir a manutenibilidade do sistema no futuro. Um documento com diretrizes gerais sobre estas necessidades de documentação deve ser criado, de forma a garantir que a documentação seja suficiente mas que o esforço para criá-la e mantê-la nunca seja maior que o necessário.	
Papel	Gerente de Documentação	
Artefatos de Entrada		Artefatos de Saída
(1) Modelo Abstrato, (2) Lista de Requisitos, (3) Documento de Visão-Escopo		(1) Plano de Documentação, (2) Padrões de Codificação
Tarefas	(1) Identificar Requisitos do Cliente quanto à Documentação, (2) Identificar Documentação Suficiente e Necessária para Garantir Manutenibilidade do Sistema, (3) Definir Padrões de Codificação, (4) Elaborar Documento de Referência com Diretrizes de Documentação	

<table>
<tr><td>Atividade</td><td colspan="2">Planejar Testes</td></tr>
<tr><td>Descrição</td><td colspan="2">Esta atividade tem por objetivo definir diretrizes para criação dos testes (automatizados ou não) do sistema. Tais diretrizes devem citar quais artefatos devem ser testados com maior profundidade, bem como a forma e natureza destes testes em linhas gerais. Deve-se considerar os pontos críticos do sistema, onde faltas teriam grande impacto negativo. Deve-se considerar também aspectos do sistema cuja consistência deva ser garantida a todo instante por testes automatizados, ou seja, aspectos e artefatos do sistema a ser desenvolvido onde uma alteração inadequada feita por qualquer desenvolvedor da equipe deva ser prontamente identificada através da execução de testes automatizados.</td></tr>
<tr><td>Papel</td><td colspan="2">Gerente de Testes</td></tr>
<tr><td colspan="2">Artefatos de Entrada</td><td>Artefatos de Saída</td></tr>
<tr><td colspan="2">(1) Modelo Arquitetural, (2) Lista de Requisitos, (3) Modelos de Implementação (quando disponível)</td><td>(1) Plano de Testes</td></tr>
<tr><td>Tarefas</td><td colspan="2">(1) Analisar Modelos, (2) Elaborar Documento com Diretrizes prar Escrita de Testes</td></tr>
</table>

Atividade	Especificar Ambiente	
Descrição	Definir as ferramentas tecnológicas de suporte ao processo de desenvolvimento, desde a gestão de processo até o ambiente de desenvolvimento e produção, incluindo ferramental de suporte a testes e documentação.	
Papel	Gerente de Projeto	
Artefatos de Entrada		Artefatos de Saída
(1) Modelo Abstrato		(1) Especificação do Ambiente de Desenvolvimento
Tarefas	(1) Efetuar Reunião com Equipe de Desenvolvimento, (2) Definir Ferramental para Suporte ao Processo, (3) Definir Ambiente de Desenvolvimento, (3) Definir Ambiente de Testes, (4) Definir Ambiente de Produção, (5) Definir Ferramental para Manutenção da Documentação, (6) Comunicar Ambiente a ser Implantado	

Atividade	Instanciar Ambiente	
Descrição	Esta atividade consiste em preparar o ambiente definido para desenvolvimento do projeto, implantar e configurar as ferramentas necessárias.	
Papel	Desenvolvedor	
Artefatos de Entrada		Artefatos de Saída
(1) Especificação do Ambiente de Desenvolvimento		(1) Ambiente de Desenvolvimento Implantado e Configurado
Tarefas	(1) Implantar e Configurar Ferramental Tecnológico de Suporte ao Processo	

Atividade	Desenvolver Interface Abstrata
Descrição	Esta atividade visa o desenvolvimento de uma interface abstrata, por intermédio da qual o cliente final possa ter uma idéia concreta da interface do sistema a ser desenvolvido. O modelo de interface abstrata pode consistir em *storyboards* (onde os cenários são ilustrados de forma detalhada), protótipos semi-funcionais ou qualquer forma de representação que garanta o entendimento geral do que será o produto final do desenvolvimento. É importante lembrar que esta atividade é exercida pelo analista pois o processo aqui proposto refere-se a um ambiente onde nem sempre é possível dispor de especialistas em interação entre o usuário e o sistema. O Analista, conforme descrito, deve ser capacitado para realização desta tarefa.
Papel	Analista
Artefatos de Entrada	Artefatos de Saída
(1) Especificação de Requisitos	(1) Interface Abstrata
Tarefas	(1) Efetuar Reunião com Gerente de Requisitos, (2) Desenvolver Protótipo para Homologação

Atividade	Implementar Interface Abstrata
Descrição	Implementação de um protótipo como parte da interface abstrata desenvolvida.
Papel	Desenvolvedor
Artefatos de Entrada	Artefatos de Saída
(1) Interface Abstrata (especificação do protótipo), (2) Especificação de Requisitos.	(1) Interface Abstrata (implementação do protótipo)
Tarefas	(1) Efetuar Reunião com Gerente de Requisitos, (2) Desenvolver Protótipo para Homologação

Atividade	Homologar Interface Abstrata	
Descrição	Aprovação pelo Cliente do protótipo (Interface Abstrata) desenvolvido.	
Papel	Cliente	
Artefatos de Entrada		Artefatos de Saída
(1) Interface Abstrata		(1) Interface Abstrata (homologada)
Tarefas	(1) Analisar Interface Abstrata, (2) Comunicar Pontos de Discordância (se houver)	

Atividade	Refinar Modelo Abstrato	
Descrição	O propósito desta atividade é a revisão e detalhamento do modelo arquitetural tendo em vista a especificação detalhada do próximo release.	
Papel	Arquiteto	
Artefatos de Entrada		Artefatos de Saída
(1) Modelo Abstrato		(1) Modelo Abstrato (revisto e detalhado)
Tarefas	(1) Analisar Interface Abstrata, (2) Analisar Cenários ou Casos de Uso, (3) Levantar Possíveis Necessidades de Alteração, (4) Efetuar Reunião com Equipe de Desenvolvimento, (5) Alterar Modelo Arquitetural Existente	

Atividade	Evoluir Especificação de Requisitos	
Descrição	O propósito desta atividade é rever e corrigir possíveis erros de elicitação e incorporar novos requisitos à especificação.	
Papel	Analista	
Artefatos de Entrada		Artefatos de Saída
Não Existe		(1) Modelo Arquitetural (revisto e detalhado)
Tarefas	(1) Analisar Interface Abstrata, (2) Analisar Cenários ou Casos de Uso, (3) Levantar Possíveis Necessidades de Alteração, (4) Efetuar Reunião com Equipe de Desenvolvimento, (5) Alterar Modelo Arquitetural Existente	

Atividade	Planejar Cronograma	
Descrição	O objetivo desta atividade é o desenvolvimento de um descrição das atividades a serem realizadas e o intervalo de tempo em que pretende-se relalizá-las contendo um plano de releases que define a parte do escopo a ser atacado em cada release.	
Papel	Gerente de Implementação	
Artefatos de Entrada		Artefatos de Saída
(1) Especificação de Requisitos, (2) Interface Abstrata		(1) Modelos de Implementação
Tarefas	(1) Analisar Interface Abstrata, (2) Analisar Requisitos, (3) Analisar Modelo Abstrato, (4) Reunir Equipe de Desenvolvimento e Avaliar Custo de Implementação das Especificações, (5) Reunir Cliente e Propor Escopo de Releases, (6) Identificar Atividades de Desenvolvimento, (7) Criar Documento com Cronograma	

Atividade	Homologar Plano de Releases	
Descrição	Aprovação pelo Cliente do escopo e prazo de entrega de cada release.	
Papel	Cliente	
Artefatos de Entrada		Artefatos de Saída
(1) Cronograma		(1) Cronograma com Plano de Releases Homologado
Tarefas	(1) Analisar Escopo dos Releases e Prazos de Desenvolvimento, (2) Comunicar Pontos de Discordância (se houver)	

Atividade	Elaborar Modelos de Implementação	
Descrição	O objetivo desta atividade é o desenvolvimento de um modelo de implementação, com baixo nível de abstração, do release a ser desenvolvido. É aconselhado o uso da UML para tal.	
Papel	Arquiteto	
Artefatos de Entrada		Artefatos de Saída
(1) Interface Abstrata, (2) Especificação de Requisitos, (3) Modelo Abstrato		(1) Modelos de Implementação
Tarefas	(1) Analisar Interface Abstrata, (2) Analisar Requisitos, (3) Analisar Modelo Arquitetural, (4) Desenvolver Modelos UML	

Atividade	Refinar Modelos de Implementação	
Descrição	O objetivo desta atividade é o refinamento dos modelos de implementação dos artefatos a serem desenvolvidos na iteração a ser iniciada.	
Papel	Arquiteto	
Artefatos de Entrada		Artefatos de Saída
(1) Interface Abstrata, (2) Especificação de Requisitos, (3) Modelo Abstrato, (4) Modelos de Implementação		(1) Modelos de Implementação (refinados)
Tarefas	(1) Analisar Interface Abstrata, (2) Analisar Requisitos, (3) Analisar Modelo Arquitetural, (4) Desenvolver Modelos UML	

Atividade	Refinar e Rever Cornograma	
Descrição	O propósito desta atividade é a revisão do cronograma baseado nas estimativas de custo de implementação	
Papel	Gerente de Projeto	
Artefatos de Entrada		Artefatos de Saída
(1) Plano de Releases, (2) Plano de Iterações, (3) Modelo de Implementação (quando houver)		(1) Plano de Releases (revisado)
Tarefas	(1) Rever Coronograma, (2) Alterar Plano de Releases	

Atividade	Elaborar Casos de Teste	
Descrição	O propósito desta atividade é elaborar cenários de teste no escopo na iteração/ release corrente. Os casos de teste devem ser o mais simples possíveis, não sendo necessária uma notação formal. Deve-se considerar testes automatizados bem como testes não automatizados.	
Papel	Gerente de Testes	
Artefatos de Entrada		Artefatos de Saída
(1) Plano de Teste		(1) Casos de Teste
Tarefas	(1) Analisar Requisitos, (2) Analisar Modelo de Implementação, (3) Descrever e Documentar Casos de Teste	

Atividade	Desenvolver Componentes	
Descrição	Esta é a atividade mais complexa do processo. Ela consiste não somente na codificação do sistema em sí. Dentro desta atividade está também a manutenção da documentação técnica, a escrita e execução de testes automatizados de unidade, a integração e a escrita e execução dos testes de integração. Todas estas tarefas devem ser executadas com a maior freqüência possível.	
Papel	Desenvolvedor	
Artefatos de Entrada		Artefatos de Saída
(1) Padrões de Codificação, (2) Casos de Testes, (3) Modelos de Implementação, (4) Documento de Questões Técnicas		(1) Componentes, (2) Modelos de Implementação, (3) Documento de Questões Técnicas, (4) Scripts de Teste, (5) Resultados de Testes
Tarefas	(1) Implementar Código dos Componentes, (2) Refatorar, (3) Registrar Alterações nos Modelos, (4) Escrever Testes Automatizados, (5) Executar Testes de Unidade, (6) Integrar, (7) Executar Testes Automáticos de Integração, (8) Registrar Questões e Soluções Encontradas	

Atividade	Inspecionar Scripts de Teste	
Descrição	Esta atividade existe para garantir o cumprimento adequado da atividade "Implementação" no que diz respeito à escrita de testes automatizados. Através desta atividade, ao fim de cada iteração, o Gerente de Testes deve homologar os testes escritos pelos Desenvolvedores ao longo da Iteração. Ele deve verificar se todos os Casos de Teste foram implementados adequadamente e seguindo as diretrizes para escrita de testes criadas anteriormente. Caso haja inconsistências, o Gerente de Testes pode fazer as alterações por conta própria ou comunicar as necessidades de alterações à equipe de Desenvolvimento, que deve implementá-las. Isso continua até que todos os testes estejam de acordo com os padrões estabelecidos.	
Papel	Gerente de Testes	
Artefatos de Entrada		Artefatos de Saída
(1) Plano de Testes, (2) Scripts de Teste		(1) Laudo de Inspeção
Tarefas	(1) Analisar Scripts de Teste, (2) Gerar Laudo de Inspeção	

Atividade	Inspecionar Documentação	
Descrição	Esta atividade existe para garantir o cumprimento adequado da atividade "Implementação" no que diz respeito à manutenção da documentação técnica do sistema pelos desenvolvedores. O Gerente de Documentação deve verificar se os modelos refletem o código implementado e se estão de acordo com as Diretrizes para Documentação estabelecidas. O Modelo Abstrato também deve ser revisado.	
Papel	Gerente de Documentação	
Artefatos de Entrada		Artefatos de Saída
(1) Plano de Documentação, (2) Documentação		(1) Laudo de Inspeção

Tarefas	(1) Analisar Documentação, (2) Comunicar Inconsistências

Atividade	Inspecionar Código	
Descrição	Esta atividade existe para garantir o cumprimento adequado da atividade "Implementação" de forma a garantir o respeito aos padrões de codificação estabelecidos e às boas práticas de programação, bem como verificar possiveis oportunidades de refatoramento em ponto pequeno.	
Papel	Desenvolvedor	
Artefatos de Entrada		Artefatos de Saída
(1) Componentes, (2) Padrões de Codificação		(1) Laudo de Inspeção
Tarefas	(1) Analisar Componentes, (2) Comunicar Inconsistências, (3) Realizar Alterações	

Atividade	Executar Testes Automatizados	
Descrição	Execução de Todos os Casos de Testes no escopo das funcionalidades já implementadas.	
Papel	Gerente de Testes	
Artefatos de Entrada		Artefatos de Saída
(1) Componentes, (2) Scripts de Teste, (3) Casos de Teste		(1) Resultados de Testes
Tarefas	(1) Executar Testes	

Atividade	Realizar Alpha-Testes	
Descrição	Esta atividade consiste num esforço conjunto de toda a equipe do projeto no sentido de encontrar possíveis falhas que não tenham sido detectadas pelos scripts e casos de teste desenvolvidos. Para cada falta exercitada, os casos de teste devem ser alterados de forma a garantir que a nova falta passe a ser verificada pelos Casos de Teste existentes.	
Papel	Gerente de Testes	
Artefatos de Entrada		Artefatos de Saída
(1) Componentes, (2) Scripts de Teste, (3) Casos de Teste		(1) Componentes (com faltas detectadas corrigidas), (2) Script de Teste (revisados)
Tarefas	(1) Procurar Faltas Não Exercitadas, (2) Comunicar Faltas Encontradas, (3) Estender Casos de Teste, (4) Implementar Testes Automatizados para Faltas Não Exercitadas	

Atividade	Avaliar Métricas	
Descrição	O objetivo desta atividade é constatar a iminência de concretização de riscos definidos no Plano de Gestão de Riscos. As medidas definidas pelo Plano de Medição devem ser extraídas e avaliadas, e ações mitigatórias devem ser tomadas caso necessário.	
Papel	Gerente de Projeto	
Artefatos de Entrada		Artefatos de Saída
(1) Plano de Medição, (2) Plano de Gestão de Riscos		(1) Medidas
Tarefas	(1) Extrair Medidas, (2) Analisar Métricas, (3) Tomar Ações Mitigatórias	

Atividade	Extrair Lições Aprendidas	
Descrição	Esta atividade consiste em uma ou mais reuniões entre toda a equipe do projeto. O objetivo desta reunião é levantar as dificuldades e questões encontradas ao longo da última iteração e registrar estas dificuldades e as soluções encontradas.	
Papel	Gerente de Projeto	
Artefatos de Entrada		Artefatos de Saída
Não Existe		(1) K-Log (estendido)
Tarefas	(1) Efetuar Reunião com Equipe do Projeto, (2) Levantar Questões, Problemas e Dificuldades, (3) Discutir Soluções, (4) Registrar Questões e Soluções	

3.2.3.4. O Container de Unidades de Processo

O diagrama a seguir representa a visão macro do processo. Para defini-la é utilizado um diagrama de unidade de processo (UP) abstrato, representando a seqüência de execução das unidades de processo detalhadas adiante. Para cada unidade de processo são apresentados os contextos iniciais e resultados (antes e após a execução das Unidades de Processo), e a solução, representada através de um diagrama de atividades de acordo com a notação definida pelo framework PEP [DAFLON03].

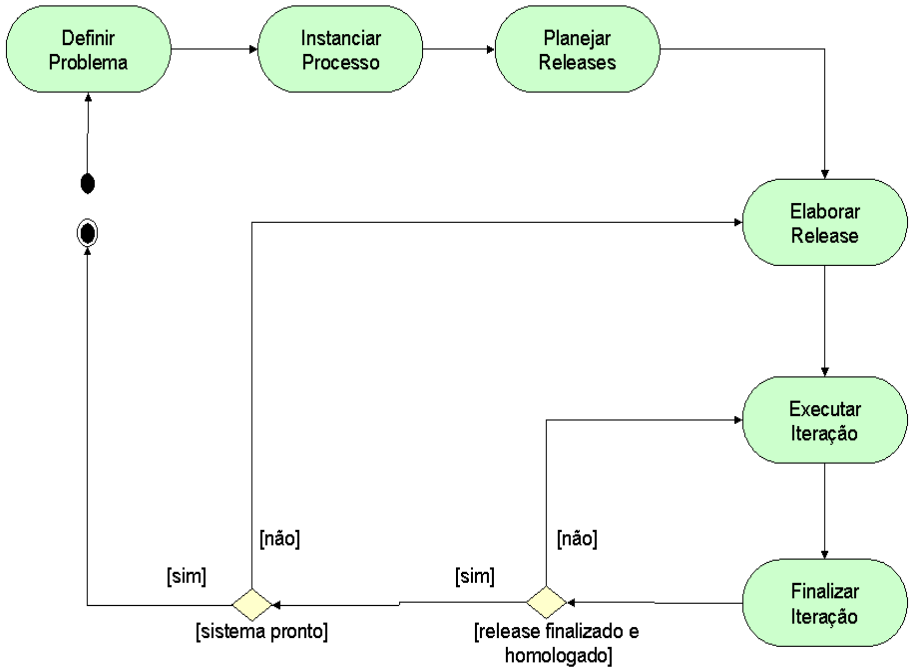

Figura 4 – Diagrama de Processo Abstrato para o Processo Proposto.

3.2.3.4.1. UP Definir Problema

Esta unidade de processo tem como objetivo estabelecer uma visão geral do problema a ser resolvido, definir as fronteiras do sistema e estabelecer um contrato entre a equipe de desenvolvimento e os interessados no sistema. Já nesta etapa deve ser definida, de forma abstrata, a solução a ser implementada. Esta visão abstrata da solução será detalhada iterativamente ao longo da execução do projeto.

Contexto Inicial

Necessário

- É necessário o envolvimento e comprometimento ativo dos interessados no projeto.
- Embora no momento da execução dessa UP o processo ainda não esteja instanciado e a equipe ainda não esteja formalmente alocada, os membros do projeto envolvidos na análise do problema devem ser facilitadores

eficientes e ter as habilidades necessárias para exercer os papéis de Analista, Cliente e Arquiteto (a lista destas habilidades consta na seção 3.2.3.1)

Recomendado

- É recomendado que sejam utilizados como artefatos de entrada documentos do domínio da aplicação.
- É recomendada a análise de sistema similares como referencial para as funcionalidades a serem desenvolvidas.
- É recomendado o uso das boas práticas na disciplina de Engenharia de Requisitos, como análise de documentos, análise de protocolo, entrevistas estruturadas e metodologias de modelagem de requisitos, na medida em que estas práticas não venham a onerar demasiadamente o processo de elicitação e modelagem dos requisitos.

Contexto Resultado

Necessário

- Após a execução desta UP deverá estar determinado o escopo inicial do projeto. As fronteiras do sistema devem estar acordadas entre os interessados no sistema.
- Deverá estar determinado também uma visão abstrata da solução a ser desenvolvida, bem como os requisitos técnicos para implementação do sistema.
- Os artefatos produzidos após a execução desta unidade de processo são: (1) Documento de Visão, (2) Lista de Requisitos, (3) Modelo Abstrato do sistema a ser desenvolvido.

Recomendado

- Esta UP pode ser re-executada várias vezes ao longo do projeto, de acordo com a evolução dos requisitos, antes do início do planejamento de cada release.

Solução

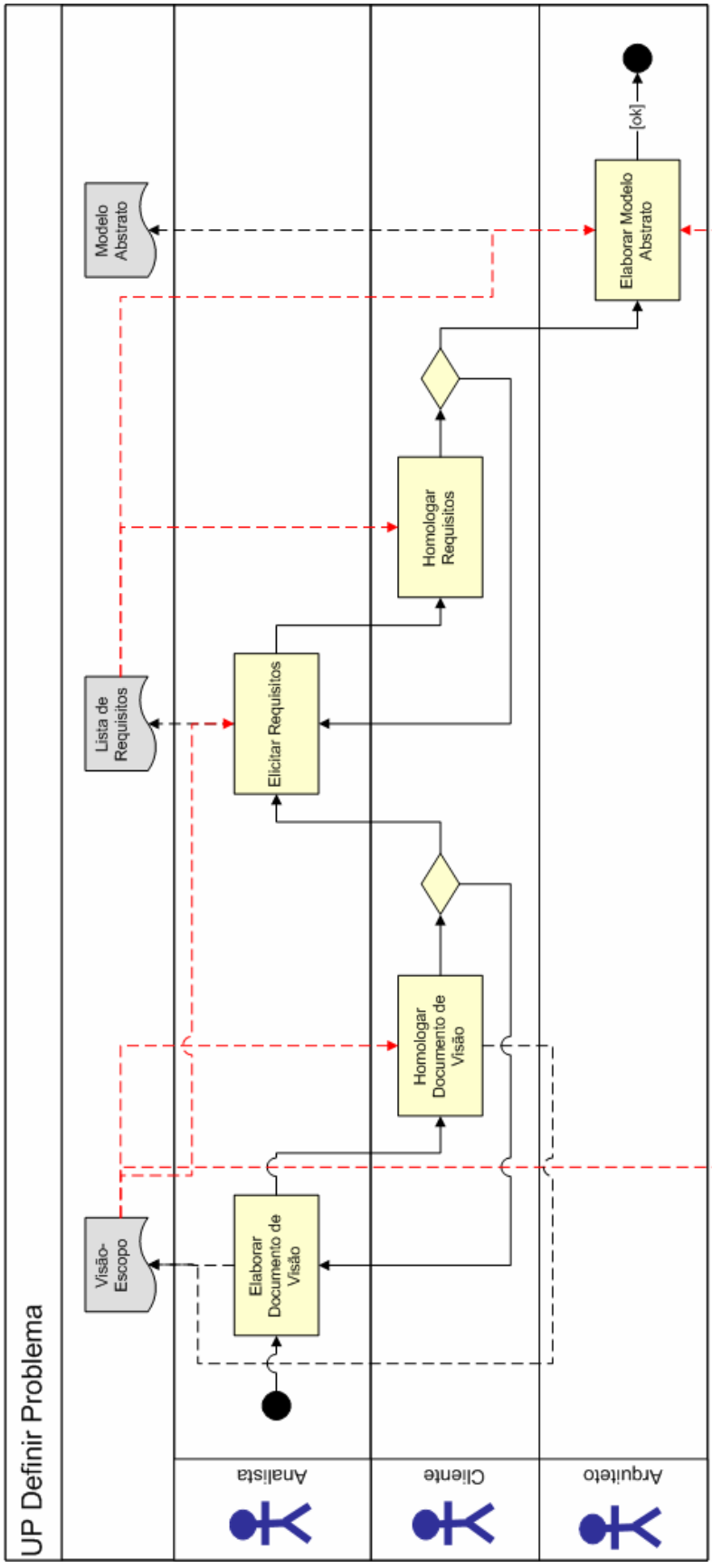

Figura 5 – Solução para a UP “Definir Problema”

3.2.3.4.2. UP Instanciar Processo

O objetivo desta unidade de processo é a instanciação do processo para diferentes projetos. O *framework PEP* prevê o controle de versão de processos, e usaremos este conceito de forma que, para cada processo, teremos uma versão do processo.

A instanciação é feita através da uniade de processo descrita a seguir. Estes processos têm como saída artefatos que definem o formato de todos os outros artefatos produzidos ao longo do projeto. Veremos portanto que a instanciação do processo consiste basicamente na definição do nível de abstração e detalhamento de cada artefato produzido.

Por exemplo, um projeto extremamente pequeno não necessita um Plano de Gerência de Riscos mais detalhado que uma breve lista dos principais riscos a serem observados. O projeto de um sistema de missão crítica, ainda que pequeno, deve ter um documento de requisitos mais detalhado que um sistema onde pequenas falhas são aceitáveis.

A escolha do formato e nível de detalhamento dos artefatos é feita pelo gerente do projeto antes do início deste. O processo a ser adotado levará em conta o tamanho e proporções do projeto e da equipe envolvida. Procuraremos tratar somente as atividades onde concentra-se o risco do projeto, minimizando o *overhead* de tarefas relacionadas à garantia de qualidade ao máximo possível. O resultado deve ser um processo minimalista, suficiente para garantir a qualidade do produto final, mas que não acrescente custo desnecessário ao projeto. Este critério é claramente exposto e justificado nos valores adotados, herdados do XP.

Contexto Inicial

Necessário

- É necessário que a abordagem para o problema, bem como as tecnologias a serem utilizadas estejam definidas, ainda que de forma superficial.
- É necessário que os objetivos e prioridades dos envolvidos com o sistema estejam claros, tanto quanto aos riscos críticos, quanto às expectativas quanto à qualidade do projeto e à documentação necessária.

- É necessária a boa comunicação entre os papéis definidos de forma a garantir a consistência do processo.
- Os artefatos necessários para esta unidade de processo são: (1) Documento de Visão, (2) Lista de Requisitos, (3) Modelo Abstrato.

Recomendado

- É recomendado que o Cliente acompanhe esta etapa do processo. Neste caso, pode ser necessário um esforço de comunicação por parte do Gerente de Projeto para garantir o entendimento, ainda que superficial, das diretrizes estabelecidas para o projeto, principalmente no que diz respeito ao Plano de Gestão de Riscos e às Diretrizes para Documentação.

Contexto Resultado

Necessário

- Após a execução desta unidade de processo deve estar claro a natureza do sistema a ser desenvolvido, o seu escopo, as pessoas envolvidas no projeto e suas responsabilidades, as ferramentas a serem utilizadas.
- Esta unidade de processo deve gerar um diagnóstico eficaz dos riscos do processo.
- Os artefatos gerados por esta unidade de processo servem como base para criação de todos os outros artefatos ao longo do processo.
- Os artefatos gerados por esta unidade de processo são: (1) Plano de Documentação, (2) Plano de Testes, (3) Plano de Gestão de Riscos, (4) Plano de Medição, (5) Padrões de Codificação, (6) Ambiente de Projeto Instanciado.

Recomendado

- É recomendado uma ou mais sessões de treinamento da equipe nas quais devem ser comunicadas os riscos do projeto, bem como as diretrizes estabelecidas para documentação, testes e codificação.

Solução

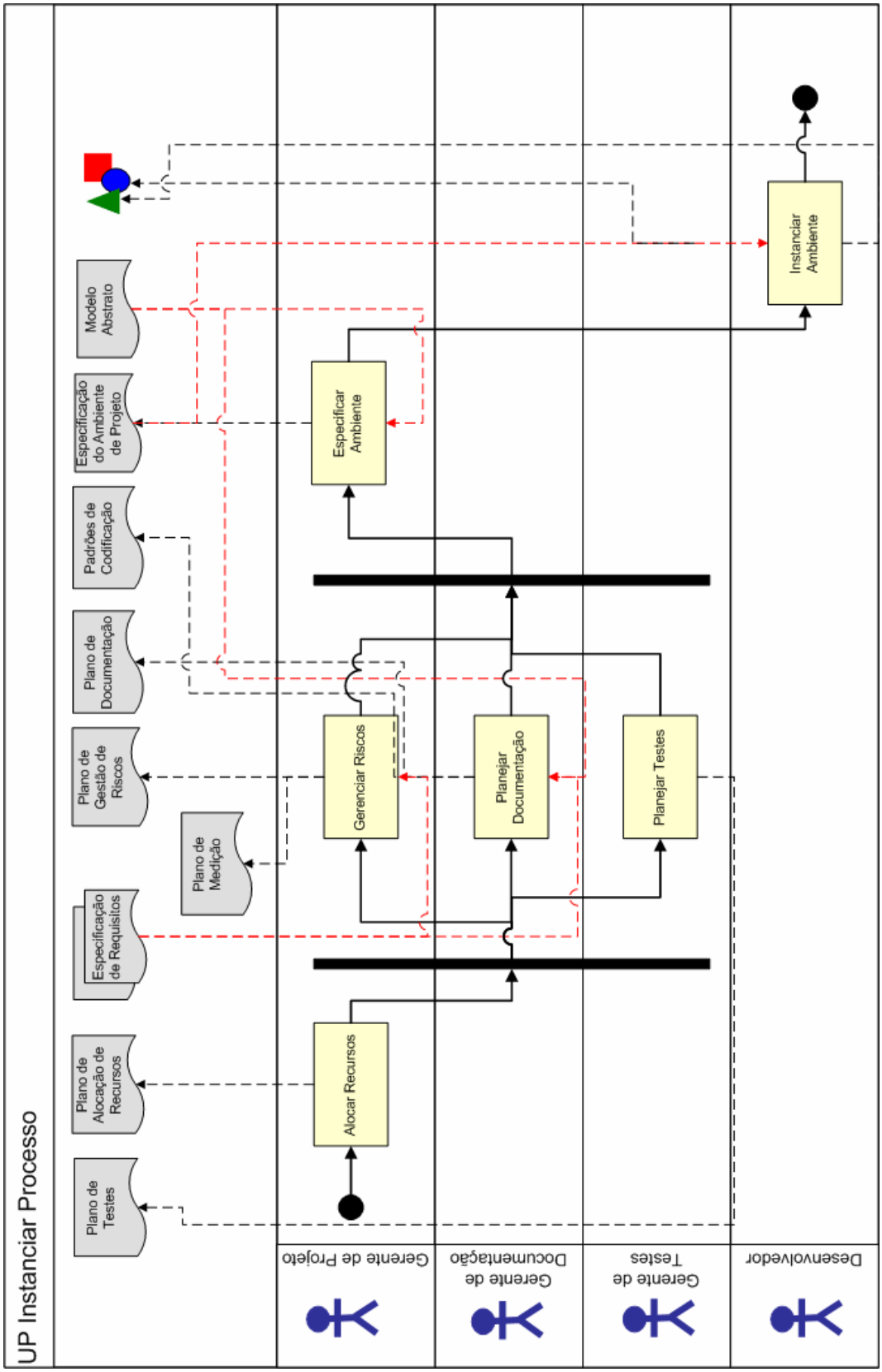

Figura 6 – Solução para a UP “Instanciar Processo”

3.2.3.4.3. UP Planejar Releases

Esta Unidade de Processo tem por objetivo a concepção de um cronograma de desenvolvimento do sistema, com marcos de entregas parciais. Cada conjunto de funcionalidades entregue constitui em uma parte funcional do sistema que pode ser homologado e criticado pelo Cliente, minimizando riscos em relação ao modelo de processo onde existe somente um ponto de entrega e homologação, ao final do projeto.

Contexto Inicial

Necessário

- É necessário que a equipe de desenvolvimento tenha experiência suficiente com desenvolvimento de sistemas para dar estimativas realistas do custo de implementação.
- Os Artefatos necessários para execução desta unidade de processo são: (1) Lista de Requisitos e (2) Documento de Visão-Escopo.

Recomendado

- Recomenda-se que o cliente participe como observador presencial em todas as atividades nesta unidade de processo.

Contexto Resultado

Necessário

- Os artefatos gerados após a execução desta unidade de processo são: (1) Interface Abstrata e (2) Cronograma.

Recomendado

- Para representar a especificação de requisitos, pode-se utilizar estorietas do tipo das utilizadas por XP, cenário ou qualquer outra representação que seja eficaz na medida do necessário para descrever as funcionalidades do sistema.

Solução

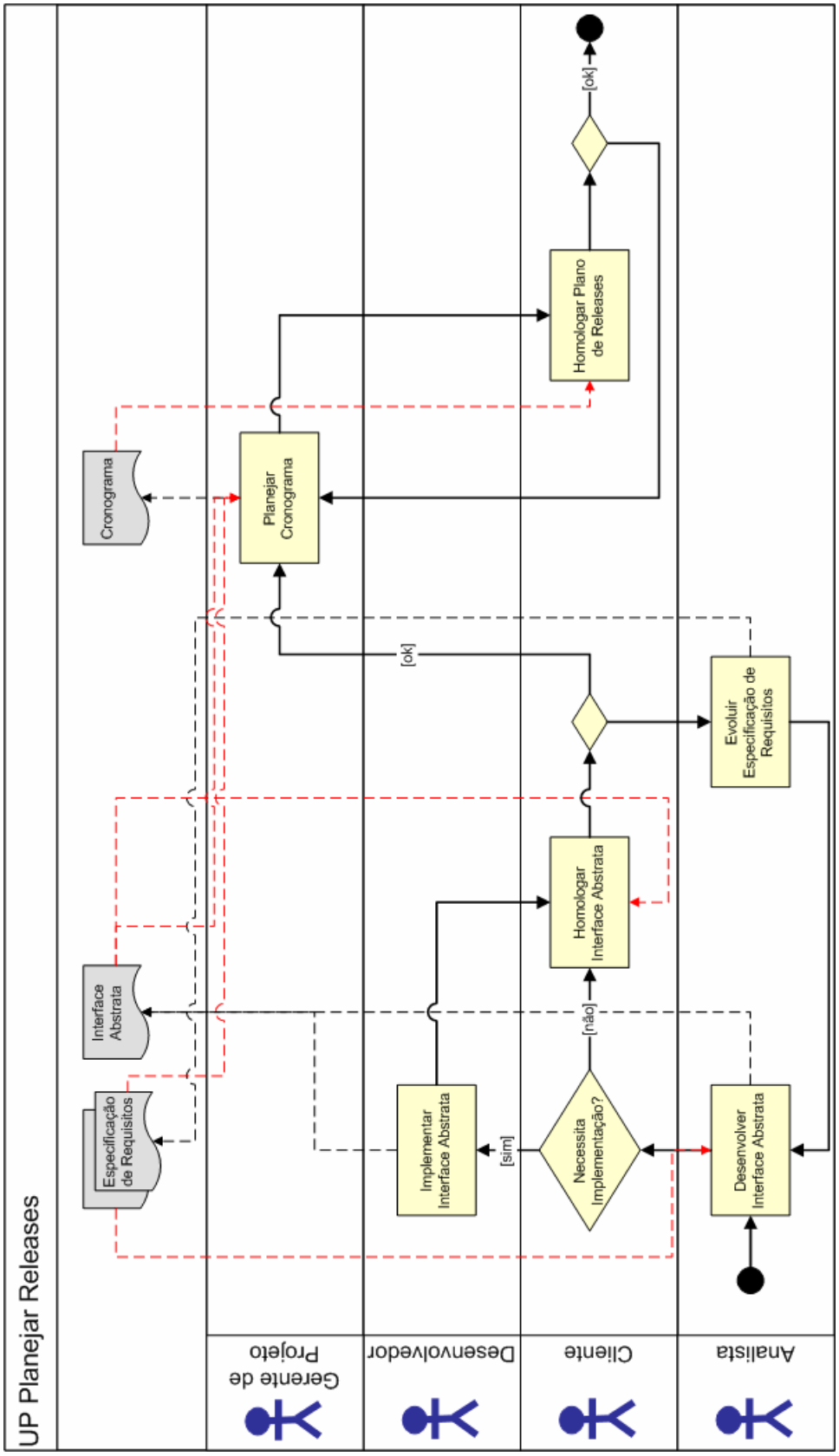

Figura 7 – Solução para a UP “Planejar Releases”

3.2.3.4.4. UP Elaborar Release

Esta unidade de processo é executada uma vez antes da primeira iteração de cada entrega (ou "*release*") prevista. O propósito central desta UP é obter com maior profundidade os requisitos e criar modelos mais completos e detalhados do sistema. Tal refinamento é feito de forma incremental, e a cada execução desta unidade de processo são refinados somente os requisitos e modelos no escopo da próxima entrega.

Contexto Inicial

Necessário

- É essencial a comunicação eficiente entre o Analista e o Gerente de Implementação para garantir que o Modelo Abstrato e os Modelos de Implementação contemplem todos os aspectos que constam nos Requisitos e Interface Abstrata.
- É necessária a comunicação eficiente entre o Gerente de Requisitos e o Cliente no refinamento dos Requisitos e no Desenvolvimento da Interface Abstrata.
- Os Artefatos necessários para execução desta unidade de processo são: (1) Especificação de Requisitos e (2) Modelo Abstrato

Recomendado

- É recomendado que toda a equipe de implementação participe ativamente da elaboração dos Modelos de Implementação.
- É recomendado o uso de modelos UML para representação dos Modelos de Implementação.

Contexto Resultado

Necessário

- Os artefatos gerados após a execução desta unidade de processo são: (1) Interface Abstrata, (2) Modelo Arquiterural refinado e (3) Modelos de Implementação.
- O modelo de Interface Abstrata deve mostrar de forma clara o resultado final do release sendo elaborado.

Recomendado

- Não Existe

Solução

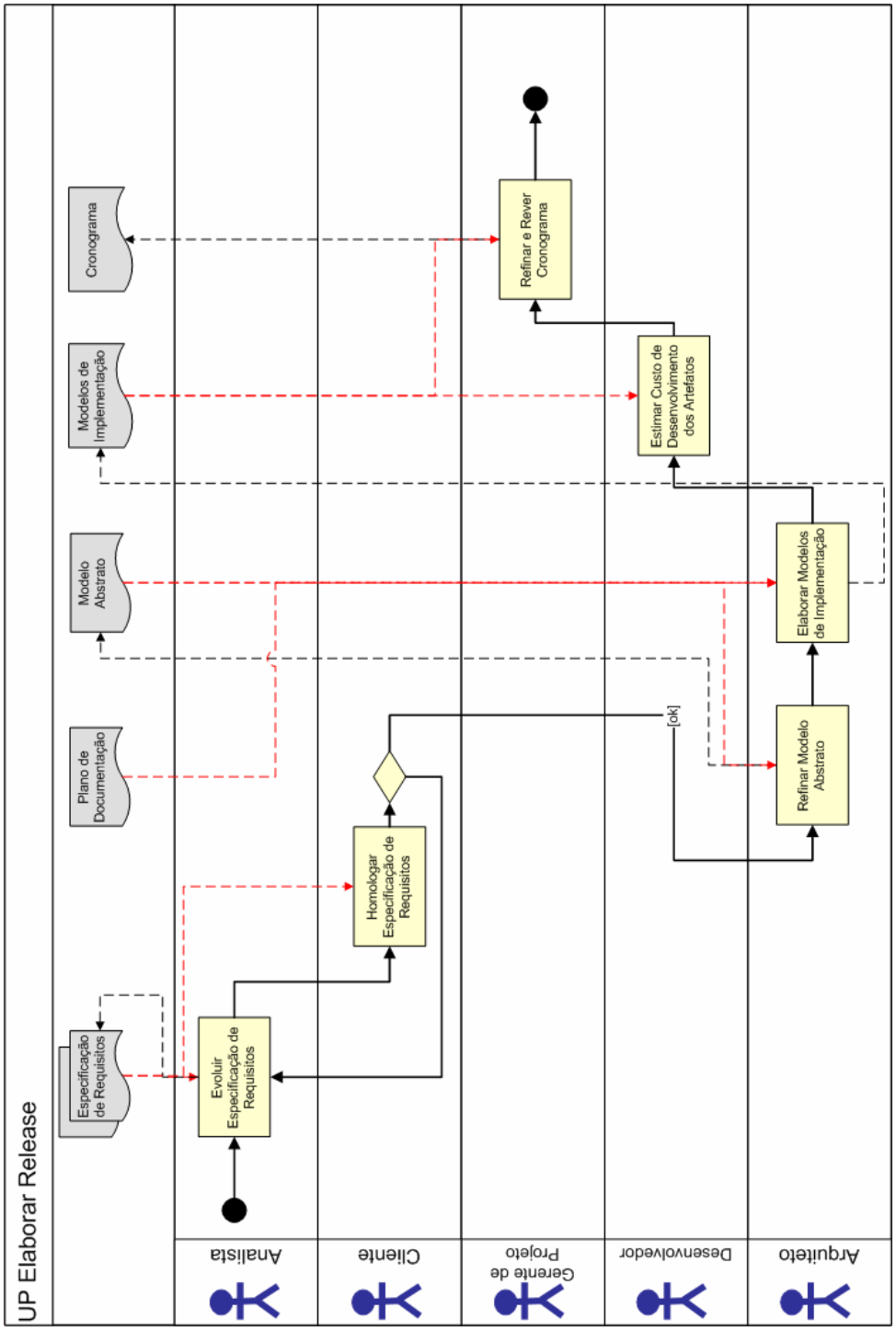

Figura 8 – Solução para a UP “Elaborar Release”

3.2.3.4.5. UP Executar Iteração

Esta unidade de processo é muito provavelmente a mais complexa do processo, e também a mais dependente dos valores e práticas apresentadas anteriormente. Todo o trabalho de implementação está concentrado em uma única atividade, denominada “Desenvolver Componentes” (descrita na seção 3.2.3.3). Esta porém é uma atividade complexa, que envolve a escrita e execução de testes automatizados, a integração e refatoramento (assim como em Extreme Programming), além da manutenção da documentação técnica do sistema. Tudo isso com a maior freqüência e granularidade possível.

Contexto Inicial

Necessário

- É necessário que a equipe de desenvolvimento tenha comprometimento com a execução das práticas contínuas descritas na seção 3.2.2.

Recomendado

- Não Existe

Contexto Resultado

Necessário

- Ao fim da execução desta unidade de processo os artefatos de código no escopo da iteração corrente devem estar todos implementados, documentados e testados de acordo com as diretrizes estabelecidas.
- Ao fim da execução desta unidade de processo os artefatos gerados são: (1) Componentes codificados, (2) Questões Técnicas, (3) Scripts de Teste, (4) Resultados de Testes, (5) Medidas.

Recomendado

- Não existe.

Solução

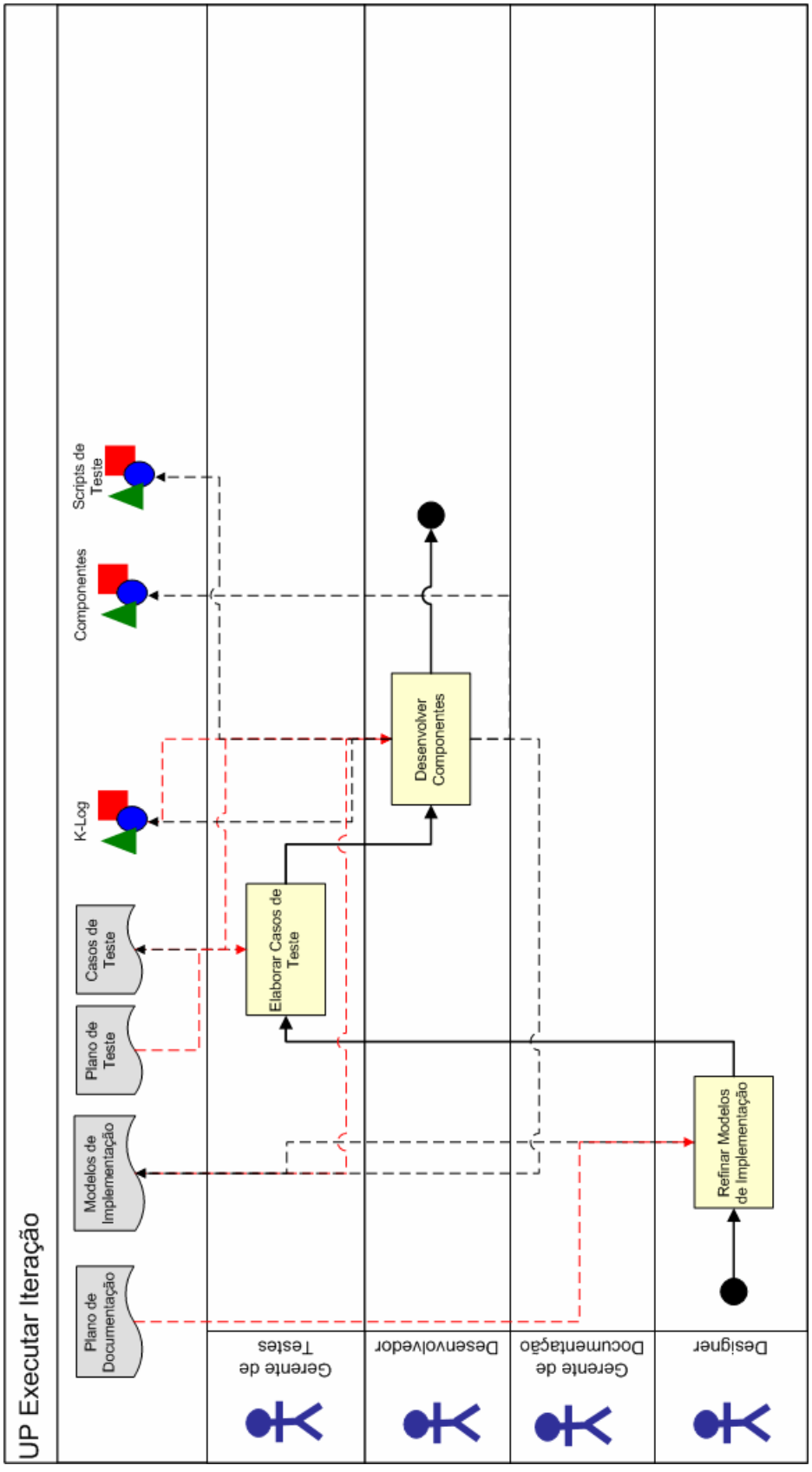

Figura 9 – Solução para a UP “Executar Iteração”

3.2.3.4.6. UP Finalizar Iteração

Esta unidade de processo tem três objetivos centrais: garantir a correta execução da tarefa de desenvolvimento, seguindo as práticas contínuas, padrões diretrizes estabelecidas para o projeto; gerar documentação e executar testes.

Contexto Inicial

Necessário

- É necessário ter todo o escopo da iteração implementado.

Recomendado

- Opcionalmente o Gerente de Testes por relegar a tarefa de "Revisar Testes", mas o resultado final e a qualidade do resultado final é de sua exclusiva responsabilidade. O mesmo se aplica à tarefa "Revisar Código" para o gerente de implementação e "Revisar Documentação" para o gerente de documentação.

Contexto Resultado

Necessário

- Ao fim da execução desta unidade de processo os artefatos de código no escopo da iteração corrente devem estar todos implementados, documentados e testados de acordo com as diretrizes estabelecidas.
- Ao fim da execução desta unidade de processo os artefatos gerados são: (1) Componentes codificados, (2) Questões Técnicas, (3) Scripts de Teste, (4) Resultados de Testes, (5) Medidas.
- Os modelos de implementação e a documentação devem estar de acordo com o código implementado.
- Os scripts de teste devem respeitar as diretrizes estabelecidas.
- O código deve estar de acordo com os padrões de codificação estabelecidos.

- Os riscos de projeto devem ser avaliados através das métricas estabelecidas e ações mitigatórias devem ser tomadas.

Recomendado

- Recomenda-se que os laudos de inspeção sejam o mais simples e objetivos possíveis, e não precisam ser registrados.

Solução

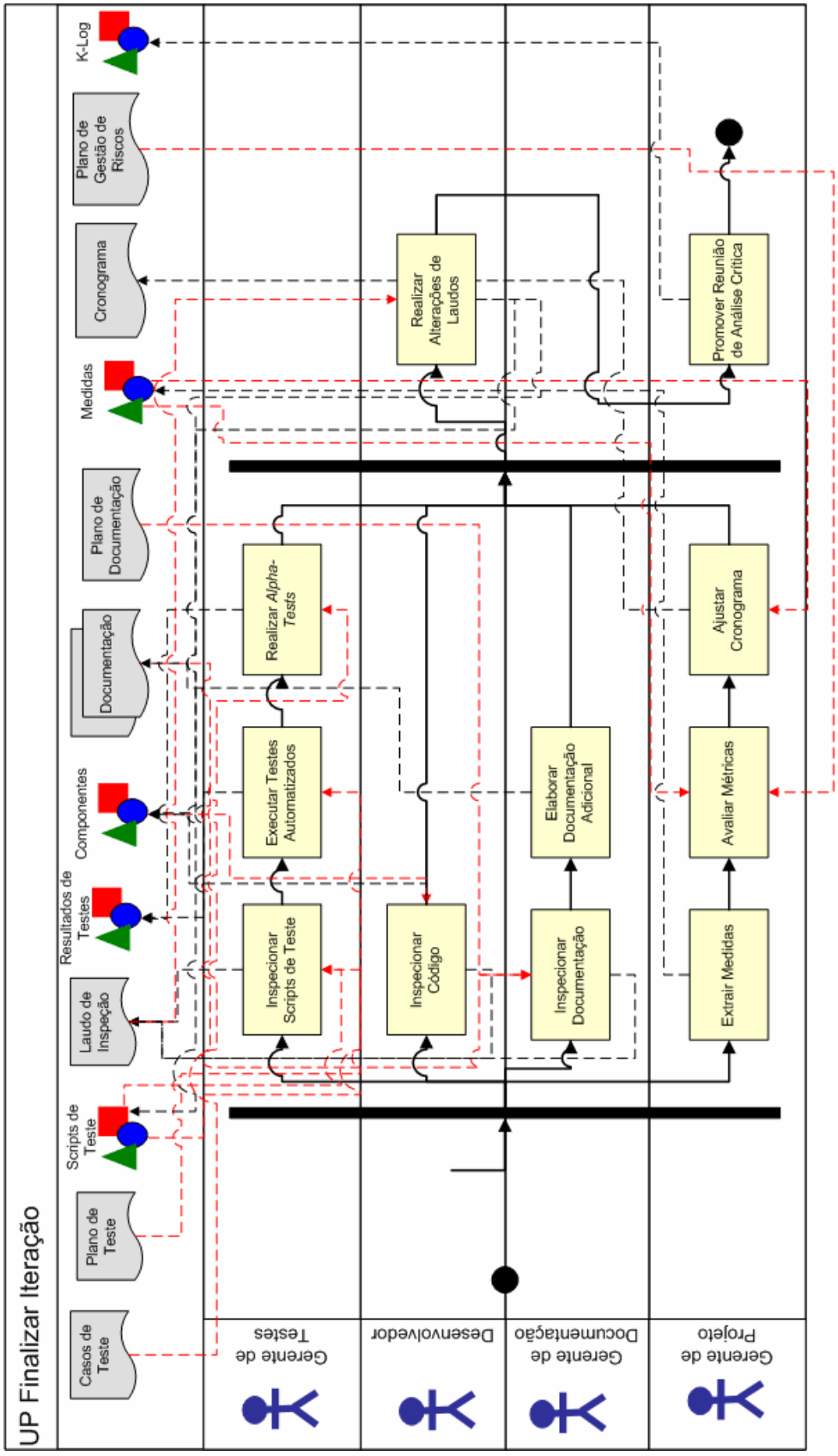

Figura 10 – Solução para a UP “Finalizar Iteração”