4
Estudo de Caso

Na primeira parte desta seção apresentaremos um estudo de caso contendo o registro de evolução do processo proposto conduzido em um ambiente que atende às caracteristicas para a qual o processo descrito se propõe: um ambiente de comunicação facilitada (todos os membros da equipe trabalham diariamente num mesmo ambiente físico), com equipes extremamente pequenas em um ambiente em rápida mudança e evolução. O objetivo deste primeiro estudo analítico é documentar o processo de evolução gradual do processo proposto e justificar seu modelo com base nas características e restrições comuns ao ambiente típico para aplicação deste processo. As conclusões e observações obtidas desta experiência são parte importante desta dissertação.

Na segunda subseção apresentaremos um estudo de casos baseado em um projeto real de desenvolvimento uma ferramenta corporativa. O objetivo desta segunda análise é validar o processo proposto neste trabalho e ilustrar de forma prática uma instância do processo em seu formato final, conforme apresentado na seção anterior.

Ambos os estudos de caso foram desenvolvidos em uma empresa de pequeno porte na incubadora de empresas da PUC-Rio, sob coordenação dos próprios proponentes do processo aqui apresentado.

4.1.
Estudo de Caso 1: Implantação do Processo

Inicialmente, a equipe de desenvolvimento era composta de dois integrantes, no caso dois dos sócios-fundadores da empresa. Neste cenário, o processo de desenvolvimento era ad-hoc, e comportava muito bem as demandas e nescecidades de uma equipe deste tamanho, principalmente considerando que, dado as áreas de conhecimento distintas, os dois desenvolvedores trabalhavam a maioria do tempo em frentes de desenvolvimento distintas. havendo pouco esforço para integração das soluções. Cada desenvolvedor realizava, de forma

iterativa e por conta própria, curtos ciclos de especificação, projeto, codificação e testes dos componentes pelos quais eram responsáveis.

A situação era ainda facilitada pela baixa maturidade do sistema desenvolvido: o empreendimento que compõe o cenário deste estudo caracteriza-se como um empreendimento focado na pesquisa e deenvolvimento de tecnologias de ponta, com planos de trazer tecnologias de processamento de linguagem natural ainda em desenvolvimento no meio acadêmico para o mercado, tentando compor um portfólio de produtos já determinado de antemão por um plano de negócios bem elaborado. Vale ressaltar que, embora estas características não se encaixem no perfil histórico mais comumente verificado no mercado real, é o procediemnto ideal. Planejar o negócio principal da empresa, avaliar mercado e estabelecer um portfólio de produtos e serviços criteriosamente elaborados são passos quase sempre cruciais para um novo empreendimento, e como verificamos neste estudo, é o cenário ideal mesmo para a subsequente implantação eficaz de um processo adequado.

Os primeiros projetos desenvolvidos neste cenário inicial tinham por objetivo demonstrar a viabilidade técnica e por conseqüência validar o pontencial comercial do projeto em questão. Por esta característica, os projetos técnicos tinham parcos requisitos de qualidade e características não muito distantes daquelas dos protótipos desenvolvidos em meio acadêmico. O objetivo era partir de tais protótipos desenvolvidos em pesquisas – parcamente capazes de funcionar de forma minimamente estável e frequentemente carentes de supervisão e interferência humana - para sistemas mais estáveis, mas ainda sem porte ou complexidade que justificassem um rigor primoroso na concepção da arquitetura ou em testes, inspeções e documentação com objetivo de garantir a ausência de falhas. O esforço para integração dos componentes gerados pelas duas frentes de trabalho era facilitado por estas características do projeto, bem como pelo fato de que cada frente era composta por um único desenvolvedor, o que era suficiente para garantir eficiencia quase absoluta na comunicação e excluía a necessidade de dispositivos formais de gerência de comunicação e gestão de conhecimento.

Verificamos que na etapa extremamente inicial de um novo negócio, o conhecimento do domínio e a proficiência técnica dos profissionais envolvidos era o ponto mais importante para o sucesso dos projetos em curso, e o uso de um processo ad-hoc não apresentava desvantagens concretas. Muito pelo contrário,

verificamos que a definição de um processo mais rigoroso e repetível neste estágio da empresa representaria um overhead no minímo desnecessário, se não fatal, para o sucesso dos projetos.

O estágio seguinte de existência do negócio caracterizou-se pela realização dos primeiros projetos comerciais para clientes externos. No estágio inicial, o empreendimento tinha baixíssimos custos operacionais, sendo fincanciada por investidores externos e pelos próprios sócios fundadores e a realização de projetos externos tornou-se necessária antes mesmo da obtenção de um primeiro release estável da plataforma sendo desenvolvida. Fez-se necessário a entrada de novos desenvolvedores, e um processo de gerência mínimo para garantir o andamento de frentes de trabalho paralelas. Os primeiros desenvolvedores a entrarem na equipe não dispunham do conhecimento de domínio nem tao pouco de proficiência avançada. Era necessária uma estrutura de trabalho que permitisse o subsídio técnico ao trabalho dos novos (e menos experientes) desenvolvedores da equipe. O processo começa a nascer com uma divisão de papéis. De início, foram formalizados os papéis de gerente de projeto, analista, arquiteto, desenvolvedor e cliente. Neste estágio inicial do projeto, o gerente de projeto era responsável estritamente por estabelecer junto com os desenvolvedores um plano de trabalho com um detalhamento das atividades a um cronograma. Esta atividade era realizada ao início do projeto e revisada com periodicidade indefinida. O analista era responsável pela análise de requisitos e homologação do sistema. O arquiteto por sua vez era responsável pela concepção do sistema, e os comunicava como diretrizes de implementação aos desenvolvedores responsáveis pela implementação da solução nos moldes concebidos pelo arquiteto. Estes pequenos ciclos de elicitação de requisitos, modelagem e concepção da solução, implementação, homologação e replanejamento das atividades e do cronograma compunham uma primeira versão do que seria uma “iteração” mais formalmente estabelecida. Tais ciclos não foram uma evolução natural, mas resultado do conhecimento dos membros mais experientes e antigos das boas práticas de engenharia de software adotadas pela indústria, fortemente inspirados pelos preceitos adotados por Extreme Programming. Embora os artefatos, entradas e saídas de cada uma destas atividades ainda não fosse formalmente estabelecidos para todas as classes de projeto, os resultados obtidos com esta abordagem iterativa eram visivelmente positivos. A falta de proficiência dos novos

desenvolvedores gerava a necessidade de um modelo detalhado, desenvolvidos analista. Tal nível de detalhamento era viável somente para pequenos incrementos do sistema de cada vez.

Da mesma forma, do ponto de vista da engenharia de requisitos, uma visão geral do sistema era estabelecida de início, e detalhada iterativamente. O cliente percebia o avanço do projeto a cada iteração. Pequenas inconsistências ou alterações no sistema podiam ser prontamente atendidas, e o detalhamento dos requisitos tornarva-se cada vez mais fácil com o avanço do projeto. Esta é inclusive uma contatação interessante neste estudo: nos projetos realizados, o cliente conseguia mais facilmente conceber e comunicar os requisitos relacionados aos incrementos seguintes quando os resultados parciais gerados pelas iterações anteriores estavam disponíveis. A concepção inicial do sistema parece ser mais complexa. Uma hipótese para isso é que a visão do sistema ainda é abstrata nas fases iniciais do projeto, e embora o cliente possua uma visão bem formulada dos objetivos do sistema, os detalhes essenciais para o desenvolvimento - como regras de negócio e de relacionamento entre objetos do modelo - não são observadas. Evidenciá-las e tratá-las com a ajuda de um analista capaz de entender as nuances por tras de tais detalhes pode não ser tão eficaz quanto ver concretamente o resultado final das decisões tomadas durante a análise de requisito dos incrementos anteriores. Esta constatação consumou-se como motivação decisiva para adotar um processo onde a elicitação inicial fosse o menor possível, deixando o detalhamento da elicitação o máximo possível dentro do escopo de cada iteração. Foi também o amadurecimento do processo de gestão de requisitos nesta etapa de existência do negócio que gerou a necessidade de uma definição formal do papel de cliente: foi evidenciado na prática a notória necessidade de estabelecer de forma completa os diferentes aspectos cobertos pelo papel cliente definido no processo apresentado: do patrocinador (que paga o projeto), passando pelos usuários e futuros responsáveis pela manutenção do sistema no cliente. Tais pessoas eram mapeadas como “clientes” do projeto, e definia-se de início quem seria responsável pela homologação do sistema e quem atuaria somente na elicitação dos requisitos (lembrar que pela definição do processo proposto, um papel pode ser exercido por mais de uma pessoa, e no caso do cliente, isso deve acontecer a maioria das vezes).

Foi também neste estágio em que o trabalho realizado desenvolvedores com pouca experiência tinha que ser constantemente supervisionado e validado por desenvolvedores, analistas ou arquitetos geralmente mais experientes que houve a adoção dos primeiros padrões. Mesmo no estágio inicial do empreendimento, com comunicação extremamente facilitada entre os poucos desenvolvedores, existiam discrepâncias nos padrões de programação, mas cada desenvolvedor tinha conhecimento dos padrões adotados por outros, e não se dispendia muito tempo para resolver as diferenças. Mas desenvolvedores inexperientes tendiam a adotar padrões pouco ortodoxos e inadequados, que por vezes causavam o caos no código gerado. O primeiro padrão a ser adotado portanto foram padrões de codificação, pelo simples motivo de que esta foi a primeira atividade a ser efetivamente distribuída por um número maior de membros da equipe. Em um segundo momento, com a entrada de profissionais que passaram a dividir atividades relacionadas ao design e arquitetura do sistema, padrões de arquitetura também foram adotados. Sem o controle do código gerado pelos desenvolvedores, de forma a constatar o uso dos padrões, estes logo caíram em desuso. Foi criada então uma atividade que deveria realizar-se antes da reunião de fechamento de uma iteração, onde o desenvolvedor senior (o arquiteto na prática) do projeto fazia a verificação do uso de todos os padrões adotados. Esta prática mostrou-se eficaz sob o aspecto de forçar o uso de padrões. Artefatos que estivessem em desacordo com os padrões ou diretrizes estabelecidas teriam de ser refeitos. Inconsistências encontradas durante estas inspeções, que eram muito comuns no início, passaram com o tempo a ser extremamente raras. A prática foi essencial portanto para a eficácia na implantação do uso de padrões.

Com o tempo ficou evindenciado que o incremento produzido por uma iteração nem sempre era adequado para a realização de homologações do cliente. Com uma equipe bastante pequena, muitas vezes o escopo de uma iteração tinha como saída alterações que não poderia ser efetivamente homologadas pelo cliente, como a implementação de uma camada de comunicação com a base de dados ou o refatoramento de um componente. Era interessante que, nos moldes propostos pelo XP, que serviu como principal inspiração inicialmente para o processo, fossem avaliados em testes de aceitação aspectos funcionais do sistema, que compõem um objeto de fato passível de aceitação. Não interessa ao cliente homologar a estrutura e implementação da base de dados ou da camada de

negócios do sistema. Testes de aceitação baseados em critérios funcionais provém feedback muito mais útil e preciso para a equipe de desenvolvimento, por isso o processo proposto, seguindo as práticas de XP, desencoraja, a não ser quando inevitável, a homologação por parte do cliente da especificação e dos modelos, e encoraja a homologação de releases funcionais. Mas nem todo incremento resultante de uma iteração é passível de homologação nestes moldes. Seguindo este preceito, foi criado um ciclo externo de homologação, que pode conter várias iterações. O resultado de cada ciclo é um release, ou uma parte do sistema com funcionalidades fechadas e passíveis de homologação

A realização de projetos comerciais traz consigo outra necessidade importante: a gestão de escopo do projeto. No processo proposto, a gestão de escopo está diretamente relacionada à gestão de requisitos. Em um projeto particular - um dos primeiros realizados pelo grupo – a falta de formalização de uma visão inicial do escopo, a falta de meios de controle de alteração e evolução dos requisitos iniciais, bem como uma postura excessivamente permissiva em relação a pequenas alterações nos requisitos que implicavam no aumento do esforço total, levaram a um resultado final problemático tanto na relação com o cliente quanto em relação aos atributos financeiros e de prazo, já que o custo total do projeto acabou ultrapassando a receita gerada.

Vimos portanto a evolução do processo ao longo dos estágios iniciais da empresa. No primeiro estágio temos um número bastante pequeno de desenvolvedores trabalhando em artefatos diferentes, ainda que em um mesmo projeto. Cada desenvolvedor tem pleno conhecimento do domínio dos artefatos em que trabalha e o esforço de integração é mínimo. Vimos que neste contexto, um processo ad-hoc é suficientemente adequado. Num segundo momento, tivemos dois fatores que motivaram a adoção de um processo mais estruturado: a entrada de desenvolvedores sem pleno domínio do sistema trabalhando conjuntamente em um mesmo artefato ou projeto e necessidade de realização de projetos externos, que gerou necessidade de divisão de responsabilidades, padronização e controle das atividades e atividades relacionadas à gestão de conhecimento, bem como a adoção de atividades relacionadas à gestão de requisitos.

Um terceiro momento na evolução do negócio abrange necessidades relacionadas à repetibilidade, ao aumento natural da maturidade bem como à gestão mais eficiente dos aspectos de qualidade de engenharia e de processo. Nesta etapa do ciclo de vida, os projetos realizados eram de maior porte e houve a entrada de profissionais mais qualificados, ainda que persistisse a necessidade de uso massivo de recursos com baixa ou média qualificação. O amadurecimento do processo acabou por distanciá-lo da abordagem inicial - fortemente baseada em XP, para um modelo mais formal e controlável, principalmente em função das características enumeradas na seção 3.1.1 deste documento (Análise Crítica do XP). Foi adotado um processo iterativo mais próximo àquele proposto pelo SCRUM [POWER02]. Esta evolução nasceu da própria maturidade de uso no processo e ao uso de sistemas de acompanhamento de issues (issue trackers) para acompanhamento das atividades de uma iteração, uso que vai muito além do simples acompanhamento de resolução de falhas, que é a proposta inicial deste tipo de ferramenta. Cada iteração era dividida em atividades, geralmente relacionadas a um produto final. Cada iteração tinha como saída um incremento do sistema, e leva o nome deste incremento (por exemplo “camada de persistência” ou “interface do administrador do sistema”). Essa abordagem difere da adotada pelo XP, onde o resultado de cada iteração ou mesmo das atividade dentro de cada iteração equivale a uma estorieta no nível funcional do sistema. Também seguindo práticas semelhantes ao SCRUM, foram institucionalizados dois tipos de reuniões: reuniões em pé (3.2.2.1), realizadas diariamente, onde é feita uma avaliação dia-a-dia do andamento da iteração e reuniões de fechamento ou de análise crítica, onde é formalizado o fechamento de cada iteração (3.2.3.4.6).

As atividades relacionadas à gestão de requisitos também foram melhor formalizadas, englobando uma descrição das atividades e artefatos por ela geradas. Foram institucionalizadas atividades e formalizados artefatos relacionados às diretrizes estabelecidas, em cada projeto, para análise de requisitos, modelagem arquitetural do sistema, desenvolvimento e manutenção da documentação do sistema, bem como desenvolvimento e realização de testes. Tais diretrizes tinham de ser desenvolvidas para cada projeto específico antes mesmo do início da execução do projeto. Na prática, verificamos diretrizes bastante parecidas para projetos de mesma natureza, com pequenas alterações entre elas.

Verificamos que a maturidade de um processo somente é alcançada com a repetição e ajuste do processo: ao longo do tempo, o grupo passou a dispor de uma biblioteca de diretrizes para diferentes projetos, e a instanciação de um novo projeto tende a tornar-se com o tempo a mera herança e adaptação dos pontos de instanciação de projetos passados. Em outras palavras, projetos passados são usados como modelos para novos projetos, não somente em relação aos pontos de instanciação do processo, como também para aspectos mais práticos, como a definição do custo (em horas) para execução de determinadas atividades do projeto.

Há uma consideração importante a ser feita em relação aos tipos de projeto empreendidos pelo grupo estudado: devido ao tipo de solução desenvolvida, que envolve tecnologia de ponta proprietária, dois grandes grupos de projetos podem ser facilmente identificados. O primeiro destes grupos engloba os projetos de pesquisa e desenvolvimento de novas soluções. Tais projetos têm características extremamentes distintas entre sí e de difícil controle. Os padrões e diretrizes estabelecidos são extremamente voláteis e devem ser reavaliados a todo instante. O segundo grupo engloba projetos para o mercado de implantação ou adaptação dos sistemas desenvolvidos no primeiro grupo. Tais projetos têm características geralmente mais repetíveis e padronizadas, e envolvem menos controle, embora tenham normalmente requisitos de padrão de qualidade de engenharia mais elevados. Projetos no segundo grupo obedecem um ciclo de vida mais bem definido de desenvolvimento, testes, produção e fechamento. Projetos no primeiro grupo muito frequentemente são problemáticos por não terem um produto final bem definido (por exemplo, a melhoria na performance de um determinado algorítmo já existente), e deve-se tomar cuidado para garantir que não tornem-se processos contínuos de desenvolvimento, ao invés de projetos com tempo limitado de execução. Observamos claramente que a realização repetida de projetos do primeiro grupo foi essencial para estabelecimento de um processo inicial, e foi subsequentemente aplicado a projetos do primeiro grupo.

4.2.
Estudo de Caso 2: Execução do Processo

4.2.1.
Descrição do Projeto

O projeto desenvolvido neste estudo de caso consiste no refatoramento de uma ferramenta para suporte ao processo corporativo de Inteligência Competitiva[2]. A ferramenta começou a ser desenvolvida dois anos antes por uma dupla de estudantes que tinham como projeto o desenvolvimento da primeira ferramenta do gênero no mercado brasileiro. Como não se tratava então de um projeto encomendado por um cliente, mas de uma proposta inovadora e sem paralelos, os requisitos no início eram baseados meramente na visão dos seus idealizadores e em sistema similares existentes no mercado exterior. Com o tempo a ferramenta foi implantada em ambiente real e passou por muitas mudanças funcionais. Os requisitos evoluíram muito e muito rapidamente, de tal forma que a estrutura do projeto inicial já não era mais capaz de dar vazão à esta evolução, e um refatoramento profundo fazia-se necessário.

Um projeto de refatoramento configura-se como um estudo de caso interessante. Veremos que os requisitos são poucos, já que a própria definição de refatoramento implica que não haja alterações funcionais no sistema, mas somente em sua estrutura interna. O objetivo deste refatoramento foi prioritariamente tornar o sistema mais manutenível, estável e escalável, e estes podem ser considerados os requisitos sobre o qual todo o projeto foi desenvolvido. Não obstante, a equipe envolvida no projeto já gozava de um bom conhecimento do domínio do sistema e das questões tecnológicas, o que a tornava proficiente e capaz de desenvolver um modelo muito mais consistente e concreto já nas primeiras etapas do projeto.

[2] Inteligência Competitiva é o processo sistemático de coleta, tratamento, análise e disseminaão da informação sobre atividades dos concorrentes, tecnologias e tendências gerais do negócio, visando subsidiar a tomada de decisão e atingir as metas estratégicas da empresa [COELHO99] .

4.2.2. Artefatos Gerados

Apresentaremos nesta seção alguns dos artefatos gerados pelas unidades de processo. Não serão descritos todos os artefatos, mas daremos ênfase àqueles que dizem respeito à instanciação do processo, com o objetivo de caracterizar a instância usada neste projeto.

Modelo Abstrato

A primeira versão do modelo abstrato concebido durante a UP "Definir Problema" contemplava os componentes que seriam desenvolvidos, suas responsabilidades, interfaces e características e a tecnologia a ser utilizada na integração destes componentes.

Por se tratar de um domínio já conhecido, a primeira versão do documento era bastante precisa e as versões posteriores não evoluíram muito. Já na versão original o documento definia diretrizes técnicas para o desenvolvimento das camadas de persistência, controle e apresentação. Definia por exemplo que a camada de persistência deveria ser desenvolvida utilizando recursos do *framework* "Hibernate" [W_HIBERNATE], que a camada de comando seria desenvolvida seguindo o padrão MVC implementado pelo *framework* "Struts" [W_STRUTS], e que na camada de apresentação seriam utilizados componentes visuais desenvolvidos especificamente para esta aplicação. Não especificava porém ainda a arquitetura utilizada na implementação destas camadas.

Como definido na descrição do processo, o Modelo Abstrato constitui o último artefato gerado pela Unidade de Processos "Definir Problema". Com a definição de **o quê** será desenvolvido definida pelo Documento de Visão e pela Especificação de Requisitos, o objetivo deste artefato é definir **como** o problema será abordado, em linhas gerais.

Na prática, este artefato surgiu de uma extensa discussão da equipe de desenvolvedores, arquiteto e gerente de projeto sobre as principais questões e limitações da versão anterior do sistema, e como seria a abordagem adotada no refactoring. Este processo de discussão, formalizado no modelo abstrato, serviu para alinhar a equipe de desenvolvimento em torno da abordagem a ser adotada.

Plano de Documentação

O Plano de Documentação deste projeto estabeleceu que os seguintes artefatos de documentação deveriam ser gerados:

Diagrama de Componentes e diagrama de seqüência contendo cenários de iteração entre os componentes, para mostrar a lógica utilizada para intergação.

Modelo de Classes detalhado da camada de negócios, descrevendo padrões todos os utilizados através de notas.

Modelo de classes e diagrama de sequência dos artefatos na camada de comando. Foi estabelecido que estes modelos não deveriam conter todas as classes implementadas, mas somente classes abstratas e interface, com o objetivo de documentar os padrões utilizados.

Javadoc [W_JAVADOC] de todas as classes e métodos. Foram definidos com precisão os artefatos que deveriam conter descrições e padrões para estas descrições. O Plano de Documentação desenvolvido definiu, por exemplo, que todas as classes no primeiro nível de hierarquia deveriam conter uma descrição de todas as propriedades e métodos, excetuando-se métodos de acesso às propriedades *(getters e setters)*.

Manuais de implementação, contendo diagramas e explicações sobre tarefas comuns na atividade de implementação, como a instanciação de uma nova interface de cadastro, configuração da persistência de uma nova classe, etc. A primeira versão deste documento não definia explicitamente quais tarefas deveriam conter manuais, mas somente a diretriz de documentar “tarefas realizadas com grande frequência por desenvolvedores ao extender de alguma forma a camada de negócios ou a interface do usuário”. Ao longo das iterações surgiu a necessidade de evolução do Plano de Documentação, que passou a definir mais explicitamente quais tarefas deveriam ser suportadas por manuais técnicos.

Plano de Testes

O Plano de Testes define que serão implementados testes de unidade para a camada de negócio e persistência. Não seriam realizados testes de integração, mas deveriam ser implementados testes automatizados para todos os cenários

especificados. Foi definido também que todas as falhas exercitadas ao longo do alfa-teste em cada iteração deveriam ser cobertas posteriormente por testes automatizados.

Plano de Medição

O plano de medição mapeia pelo menos uma métrica para cada um dois fatores de riscos identificados no Plano de Gestão de Riscos.

Alguns dos riscos enumerados, como o aumento da complexidade do sistema são facilmente identificáveis através de métricas de complexidade. Foram utilizadas medidas relativas neste caso, buscando identificar a relação entre a evolução do projeto de engenharia e a complexidade do código. Foi utilizada por exemplo a proporção entre o número de classes implementadas e o número de iterações. O objetivo era identificar a **evolução da complexidade** do sistema ao longo do projeto. Os patamares aceitáveis para valores destas medidas foram estabelecidas logo no início da primeira iteração, durante a elaboração do plano de métricas, mas tiveram que ser reavaliados ao longo do projeto. A razão para isto é que não se sabia de início estimar um valor razoável, e os valores estabelecidos em um primeiro instante como aceitáveis mostraram-se demasiadamente rígidos, e, embora tivessem sido ultrapassados logo no início do projeto, diminuíram e estabilizaram ao longo das iterações. A avaliação dos níveis aceitáveis para estas medidas foi feita em conjunto com a equipe de desenvolvimento. Tais limites, muito voláteis ao longo do projeto piloto, foram estabelecidos como razoáveis somente ao final do projeto e registrados para uso em projetos futuros. Concluímos com isso que a composição de métricas também é um trabalho evolutivo, e a correlação dos valores medidos com a concretização ou não de um determinado risco de projeto atrelado a estes valores é um assunto complexo, que deve ser amadurecido ao longo do projeto e até mesmo entre projetos semelhantes.

Outro problema encontrado, naturalmente, foi o de estabelecer métricas para alguns dos riscos identificados. Alguns aspectos de risco conceitualmente mais abstratos, como a adequação da abordagem tecnológica escolhida são dificilmente diagnosticável através de métricas delgadas, deixando a responsabilidade deste diagnóstico para as reuniões de análise crítica realizadas ao fim de cada iteração.

Este diagnóstico, embora não tenha o mesmo formalismo daqueles feitos com o uso de métricas bem estabelecidas, era o único possível, e mostrou-se bastante eficaz e apropriado dentro do contexto de um processo que propõe-se leve e minimalista. Como dito anteriormente, estas reuniões serviram inclusive para avaliação do próprio conjunto de métricas utilizados.

Por ser um projeto piloto, foram definidas métricas para análise do próprio processo de desenvolvimento. Algumas métricas desenvolvidas buscavam estimar o *overhead processual*, conforme definido na seção anterior. As realizações destas métricas foram registradas com o objetivo de serem comparadas com outras instâncias do processo, em projetos diferentes. O objetivo era avaliar o impacto de possiveis alterações futuras no processo aqui proposto; as diferenças práticas entre projetos de características diferentes utilizando o mesmo processo e os pontos de melhoria no processo. Foi avaliado por exemplo o tempo de execução de cada uma das atividades do processo. Estas avaliações culminaram na re-engenharia do processo ao longo do próprio projeto. As alterações realizadas já constam na descrição do processo apresentada na seção anterior.

Lista de Requisitos

Por tratar-se de um projeto de refatoramento, a lista de requisitos permaneceu quase essencialmente composto de requisitos não-funcionais, mas não unicamente. A razão para que existissem requisitos funcionais na lista de requisitos é que um refatoramento da camada de apresentação do sistema permitiria a implementação de melhorias na interface com o usuário que seriam impossíveis ou demasiadamente complexas com a arquitetura anterior. Estas alterações não mudaram essencialmente a funcionalidade do sistema, levando em conta que os casos de uso permaneceram inalterados, mas aspectos funcionais da interface mudaram profundamente.

Conforme especificado no container de artefatos, a lista de requisitos é simplesmente uma lista enumerando, em linguagem natural, coisas que o sistema deve ou não fazer.