UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE CIÊNCIAS DA COMPUTAÇÃO

UMA FERRAMENTA WEB PARA SUPORTE À DEFINIÇÃO DE PROCESSOS ÁGEIS

Luiz Paulo de Farias Júnior

Trabalho de conclusão de curso
apresentado como parte dos
requisitos para obtenção
do grau de Bacharel em
Ciências da Computação.

Orientadora:

Profª. Drª. Patrícia Vilain

Florianópolis,
2013/1

Luiz Paulo de Farias Júnior

UMA FERRAMENTA WEB PARA SUPORTE À DEFINIÇÃO DE PROCESSOS ÁGEIS

Trabalho de Conclusão de Curso apresentado como parte dos requisitos para obtenção do grau de bacharel em Ciências da Computação

---

Profª. Drª. Patrícia Vilain
Universidade Federal de Santa Catarina
Orientadora

Banca Examinadora

Prof. Dr. Vitório Bruno Mazzola
Universidade Federal de Santa Catarina

Prof. Dr. Leandro José Komosinski
Universidade Federal de Santa Catarina

AGRADECIMENTOS

Aos meus pais, por me apoiarem e me propiciarem todas as condições para realização deste trabalho, à minha orientadora, pelas inúmeras reuniões, além de todo auxílio prestado e também a todo o corpo discente desta universidade pela direta ou indireta participação neste trabalho.

SUMÁRIO

“Parte da ausência de humanidade do computador deve-se a que,
competentemente programado e trabalhando bem,
é completamente honesto.”

Isaac Asimov

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Atualmente, os métodos ágeis tem obtido grande atenção por parte da comunidade de engenharia de software por proporcionarem uma alternativa para o desenvolvimento de sistemas de forma mais rápida e eficiente atendendo à real necessidade e mudanças que ocorrem durante um projeto de sistema computacional. No mercado existem diversos métodos ágeis propostos, sendo estes métodos compostos por diversas práticas ditas ágeis. Com base nisso foi desenvolvido um framework para auxiliar a montagem de processos ágeis customizados a partir da seleção de um conjunto de práticas ágeis, conforme as reais necessidades do projeto. Como existe a possibilidade de fazer diversas combinações entre estas práticas ágeis, faz-se necessário o auxílio de um sistema computacional. Este trabalho descreve o planejamento e desenvolvimento de uma ferramenta web para auxiliar neste processo, além de demonstrar dois casos de utilização da ferramenta desenvolvida.

Palavras-chave: engenharia de software, programação web, métodos ágeis, projeto de sistemas.

ABSTRACT

At present, agile methods have been achieving great attention from the software engineering community by offering an efficient alternative to conservative methods in software development, enabling accelerated value generation in the view of the real - constant changing - necessities. Several different agile methods were proposed, all of them consisting of various agile practices. On this basis, a framework was established in order to assist the modeling of custom agile processes accordingly to each project characteristics [FAGUNDES 2005]. As these agile practices and activities can be combined in different ways, the aid of a computer system to facilitate the task of defining agile processes is also made necessary. This work aims to develop a web system that provides such assistance.

Keywords: software engineering, web programming, agile methods, system projects.

1 INTRODUÇÃO

Nos últimos anos, diversos métodos tem surgido com diferentes propostas para agilizar o processo de desenvolvimento de software. Tais métodos possuem em comum o fato de serem aplicados utilizando ciclos iterativos curtos (Figura 1), planejamento guiado por funcionalidades, retroalimentação constante, tolerância a mudanças, proximidade da equipe, intimidade com o cliente e um foco no ambiente geral de trabalho da equipe [FAGUNDES 2005].

Os diversos métodos que sugiram oferecem às organizações uma grande variedade de práticas e enfoques para o desenvolvimento de software, propiciando diferentes opções sobre o método que melhor supra suas necessidades.

Para a escolha do melhor método a ser utilizado, as organizações precisam estudar e analisar os métodos existentes, o que demanda tempo e nem sempre há a possibilidade da realização de um estudo muito abrangente. Além do mais, dificilmente um único método vá suprir todas as necessidades de uma organização, o que despende mais tempo para a adaptação do método escolhido.

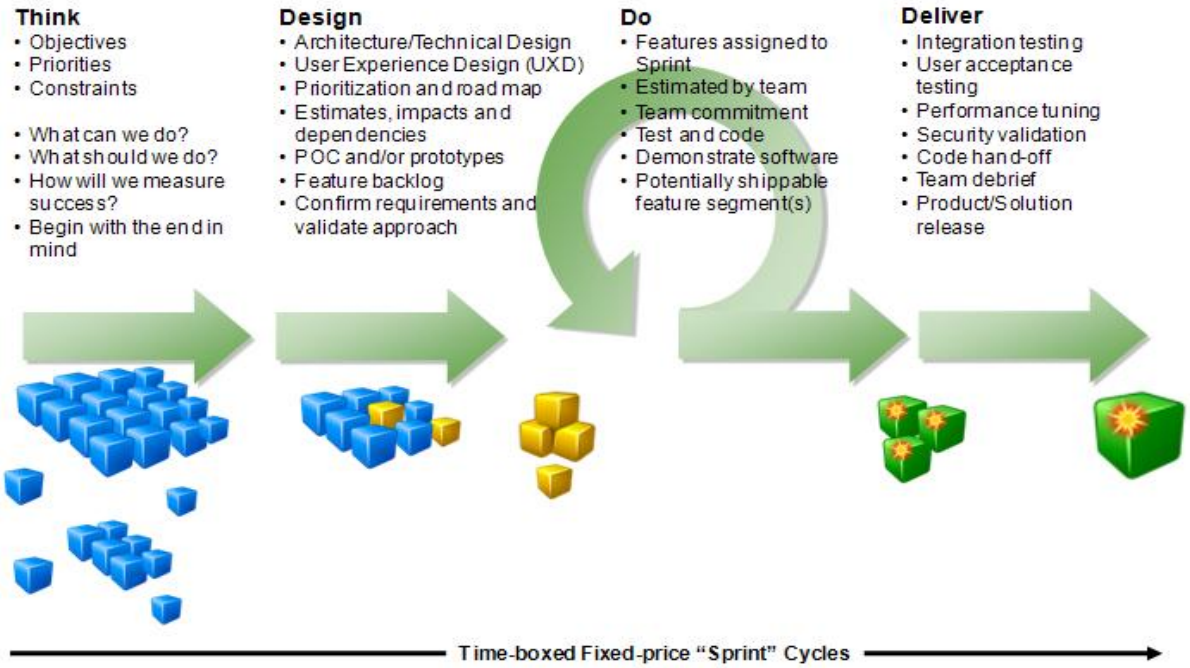

Figura 1: Um ciclo básico de um processo ágil[1]

A fim de auxiliar nesta difícil missão de adequar um método a uma realidade específica de uma organização, em [FAGUNDES 2005] é proposto um framework, *Framework for Comparing and Analyzing Agile Methods*, que agrupa práticas de diferentes métodos ágeis para que sejam selecionadas de acordo com as necessidades próprias do interessado. Contudo, existem algumas incompatibilidades entre as diversas práticas que devem ser observadas, sendo adequado o uso de um software para a definição de novos processos de software a partir do framework.

[1] http://www.progyan.com/agile.html

1.1 OBJETIVO GERAL

Este trabalho propõe o desenvolvimento de uma solução web que suporte de maneira intuitiva e fácil a definição de processos ágeis específicos à realidade do usuário. Esta ferramenta deve auxiliar na seleção das práticas ágeis que irão compor o novo processo ágil utilizado durante o processo de desenvolvimento do software, pela organização. Esta ferramenta deve também auxiliar no cálculo da agilidade do processo criado, através da aplicação de diretrizes definidas em [FAGUNDES 2005].

1.2 OBJETIVOS ESPECÍFICOS

A seguir, são apresentados os objetivos específicos do trabalho.

- Identificar os requisitos da ferramenta web que será implementada.
- Projetar a ferramenta web que será implementada.
- Desenvolver e testar a ferramenta web especificada acima.

1.3 JUSTIFICATIVA DO TRABALHO

O Framework proposto em [FAGUNDES 2005] é um importante para a definição de processos ágeis customizados, contudo, o processo envolvido nessa montagem é custoso e complexo para ser realizado manualmente, sendo dessa forma uma adoção complicada e onerosa aos interessados.

Em resposta à crescente necessidade das empresas em definir seus próprios processos ágeis, se torna evidente a demanda por uma ferramenta que torne rápida e o mais fácil possível este processo.

Em [MACHADO 2005], foi definida uma primeira ferramenta para dar suporte ao framework, em versão desktop. Porém, como naquele momento a necessidade de

flexibilidade para dar suporte à evolução do framework ainda não havia sido identificada, esta ferramenta não possibilitava alterações no framework, impossibilitando, assim, a adição de práticas de novos métodos ágeis criados, bem como mudanças na estrutura do framework.

A popularização do acesso à internet, bem como a possibilidade de colaboração entre pessoas em diferentes locais para a definição de um processo torna o desenvolvimento de uma aplicação web para auxiliar neste processo uma alternativa interessante.

1.4 ESTRUTURA DO TRABALHO

Este trabalho está organizado da seguinte maneira.

No capítulo 2, são apresentados os principais conceitos com relação aos métodos ágeis e apresentado o Framework de Processos Ágeis [FAGUNDES 2005].

No capítulo 3, são mostrados aspectos relacionados a efetividade de um processo ágil [SCHOEPPING 2012].

No capitulo 4, é mostrado o projeto e desenvolvimento da ferramenta.

No capitulo 5 é mostrado um pouco do funcionamento da aplicação e suas configurações.

Por fim no capitulo 6 é feita a conclusão do trabalho e projetos futuros são propostos.

2 FRAMEWORK E MÉTRICAS DE AGILIDADE

Em meados da década de 90, começaram a surgir as primeiras definições de desenvolvimento ágil de software, em resposta aos métodos comumente utilizados naquela época, vistos como burocráticos, lentos e contraditórios com a maneira que os engenheiros de software realmente trabalhavam [SCHOEPPING 2011].

Em 2001, alguns membros desta nova comunidade criada definiram formalmente um conjunto de valores e princípios que norteariam esta pratica de desenvolvimento, o "Manifesto Ágil [BECK 2001]". Posteriormente algumas das pessoas que assinaram inicialmente o manifesto fundaram a *Agile Alliance*[2], uma organização sem fins lucrativos com objetivo de difundir as metodologias ágeis de desenvolvimento de software, organização esta que perdura até os dias de hoje como uma das principais referências sobre o assunto.

Inicialmente surgiram diversos modelos de métodos ágeis podendo citar como exemplo o *Scrum*, *Extreme Programming* (XP), *Feature Driven Development* (FDD), *Adaptative Software Development* (ASD) e *Agile Modeling* (AM), dentro muitos outros. Tais métodos possuem em comum o fato de serem aplicados em projetos não muito complexos, utilizando ciclos iterativos curtos, planejamento guiado por funcionalidades, retroalimentação constante, tolerância a mudanças, proximidade da equipe, intimidade com o cliente e um foco no ambiente geral de trabalho da equipe [HIGHSMITH 2002].

Para chegar a estes objetivos, os diversos métodos se apoiam em um agrupamento de práticas muitas vezes comuns entre si, cabendo ao interessado discernir sobre qual melhor se encaixa às suas necessidades. Porém, em grande parte das vezes as práticas

[2] www.agilealiance.org

destes métodos não suprem totalmente as necessidades do utilizador, além de despender um tempo considerável para a sua customização.

2.1 FRAMEWORK PARA COMPARAÇÃO E ANÁLISE DE METODOLOGIAS ÁGEIS

Com intuito de facilitar o processo descrito acima, [FAGUNDES 2005**]** realizou um estudo afim de listas semelhanças e diferenças entre os métodos supra citados. Posteriormente [FAGUNDES 2005] propôs um framework que apresenta as diferentes práticas estudadas, relacionadas conforme o processo incremental.

Dando continuidade, [MACHADO 2007] incrementou o estudo realizado por com mais três métodos: *Dynamic System Development Method* (DSDM), *Crystal Clear* e *Lean Software Development* (LSD), além de propor uma ferramenta computacional para auxiliar neste processo.

Abaixo serão descritos os métodos ágeis que compõe o framework de comparação e análise de metodologias ágeis.

2.1.1 SCRUM

O *Scrum* [FAGUNDES 2005] é um processo ágil, iterativo e incremental, para gerenciamento de projetos e desenvolvimento de software. O *Scrum* propõe um grupo de práticas e define alguns papéis com responsabilidades definidas.

Os papéis definidos pelo *Scrum* são:

- *Scrum Master*: é o líder técnico de desenvolvimento dentro de uma equipe *Scrum*. Também tem papel de facilitador, sendo responsável por resolver

impedimentos à realização do trabalho pela equipe e garantir que o processo seja respeitado;

- *Product Owner*: representa a voz do cliente no projeto, o principal intuito do responsável por este papel é trazer valor ao negócio, montando e priorizando o *Product Backlog* com os itens a serem desenvolvidos durante as *Sprints*.
- *Equipe*: é formada pelos desenvolvedores e sua função é desenvolver os itens selecionados em cada iteração.

As principais práticas propostas pelo *Scrum* são:

- *Product Backlog*: conjunto de funcionalidades priorizadas que devem ser desenvolvidas em um software. É utilizado para alimentar o Sprint Backlog.
- *Sprint Backlog*: é um subconjunto de funcionalidades selecionadas dentro do *Product Backlog* que serão implementadas durante o Sp*rint*;
- *Sprint*: também chamado de iteração, é o período onde são desenvolvidos os itens selecionados ao seu respectivo *Sprint Backlog*;
- *Planning Meeting*: reunião que ocorre no início de cada *Sprint* para definição de seu *Sprint Backlog* incluindo priorização das funcionalidades a serem desenvolvidas;
- *Daily Meeting*: reunião diária de curta duração entre os membros da equipe para expor possíveis problemas e dificuldades que estão sendo enfrentadas;
- *Revisão da Sprint*: momento em que ocorre a apresentação informal para o *Product Owner* do incremento desenvolvido durante a *Sprint*.

O processo do *Scrum* apresenta três fases:

- *PreGame*: fase em que ocorre a definição do sistema, equipe, ferramenta, avaliação dos riscos, treinamento e são definidos os itens iniciais do *Product Backlog*. Pode ser subdividido em Planejamento e Arquitetura;
- Desenvolvimento: momento em que são executadas as *Sprints* para desenvolvimento do *Product Backlog*;
- *PostGame*: entrega final do sistema após o termino de todas as *Sprints* planejadas.

2.1.2 EXTREME PROGRAMMING (XP)

O *Extreme Programming* (XP) [FAGUNDES 2005] é um método com atenção voltada principalmente à eficiência, dinamicidade e flexibilidade as recorrentes mudanças no decorrer de um projeto. Para isso define quatro valores básicos para seu sucesso: comunicação, simplicidade, feedback, respeito e coragem.

O XP sugere a adoção dos diversos papéis no desenvolvimento, sendo eles: programador, cliente, testador, rastreador, treinador, consultor e gerente. O XP propõe a organização do processo em cinco fases distintas:

Os papéis definidos pelo *XP* são:

- Exploração: momento em que os usuários descrevem as estórias de usuário para se ter um real entendimento do escopo do projeto, enquanto os desenvolvedores selecionam as tecnologias envolvidas;
- Planejamento: ocorre a definição de prioridade para as estórias de usuário bem como a definição de prazo para a próxima entrega;

- Iterações para Entrega: iterações curtas, de uma a quatro semanas, para modelagem, desenvolvimento e testes de unidade e aceitação e integração das estórias de usuário que compõe a entrega;
- Produção: começa ao fim da primeira iteração, onde o sistema já vai para produção no cliente;

Para que este processo acima descrito seja executado com sucesso o XP apresenta um conjunto de práticas e técnicas, sendo doze inicialmente definidas em 2000 e mais duas adicionadas posteriormente em 2004:

- Jogo de Planejamento: ocorre no inicio de cada iteração para planejar as atividades desenvolvidas, bem como a estimativa de tempo envolvido no desenvolvimento;
- Entregas Frequentes: em curtos e constantes espaços de tempo, devem ser realizadas entregas de versões ao cliente;
- Metáfora: o projeto deve fornecer ao menos uma metáfora para nortear e contextualizar o desenvolvimento efetuado pela equipe;
- Projeto Simples: é um principio básico que tem a premissa: Simples é melhor que complexo;
- Testes: Todos os programadores são responsáveis por escrever testes de unidades cobrindo todo o código por eles gerado, e os clientes por efetuar testes de aceitação;
- *Refactoring*: técnica de reestruturação de código para ficar mais reutilizável e compreensível, que deve ser efetuada sempre que for verificada possibilidade;

- Programação em Pares: todo ó código é produzido por dois programadores sentados lado a lado, utilizando apenas um computador, discutindo sobre o código implementado;
- Propriedade Coletiva: responsabilidade coletiva sobre o sistema em desenvolvimento;
- Integração Contínua: ao fim de um dia de desenvolvimento, todos os testes da aplicação devem ser executados, e em caso de sucesso, deve ser gerada uma versão funcional do sistema;
- Semana de 40-horas: não deve ser feito hora extra por mais de uma semana;
- Cliente Presente: o cliente deve estar sempre disponível para sanar dúvidas da equipe de desenvolvimento, afim de minimizar erros;
- Padrões de Codificação: a equipe deve seguir um padrão de código uniforme, com o intuito de facilitar a leitura posterior por todos os envolvidos.
- Reuniões em pé: reuniões diárias e rápidas, abordando apenas as tarefas realizadas e as próximas a serem desenvolvidas.
- Time coeso: a equipe deve ser formada por pessoas engajadas e de forma multidisciplinar, afim das habilidades de cada membro da equipe se complementarem durante o projeto.

2.1.3 AGILE MODELING (AM)

A AM [FAGUNDES 2005] é uma metodologia ágil totalmente focada na modelagem e documentação de um sistema. Para isso, fornece conselhos de como modelar sistemas de uma forma eficiente sem que o processo perca agilidade. Segue os mesmos quatro valores básicos do XP, a Comunicação, a Simplicidade, Feedback e Coragem, e traz

ainda um quinto valor, a Humanidade, referenciando o respeito dentro da equipe de trabalho.

A AM sugere três papéis para compor a equipe de modelagem:.

- Facilitador;
- Escrivão;
- Observador.

As práticas apresentadas pela AM para guiar a modelagem são divididas em duas partes: Básicas e Suplementares. As praticas básicas, que devem ser obrigatoriamente executadas são divididas nas três seguintes categorias:

- Práticas para modelagem iterativa:
  - Aplicar o(s) Artefato(s) Correto(s);
  - Criar Diversos Modelos em Paralelo;
  - Iterar em Outro Artefato;
  - Modelagem incremental;
- Práticas para um trabalho de equipe eficaz:
  - Modelagem em conjunto;
  - Participação Ativa do Cliente;
  - Posse Coletiva;
  - Visualização dos Modelos.
- Práticas que permitem a simplicidade:
  - Criação de Conteúdo Simples;
  - Apresentação de Modo Simples;
  - Utilização de Ferramentas Simples;
- Práticas para validar o trabalho:

- Considerar a Testabilidade
- Comprovar com Código.

As práticas suplementares podem ser usadas dependendo da necessidade do projeto e são divididas nas categorias:

- Práticas para melhorar a produtividade
  - Aplicar as Convenções de Modelagem;
  - Utilizar Padrões de Projeto;
  - Reuso dos Recursos Já Existentes;
- Práticas para documentação ágil:
  - Descartar Modelos Temporários;
  - Formalização dos Modelos de Contrato;
  - Atualizações Somente Quando Necessário;
- Práticas relacionadas com a motivação:
  - Modelar para Entender;
  - Modelar para Comunicar;

2.1.4 FEATURE DRIVEN DEVELOPMENT - FDD

O FDD [FAGUNDES 2005] é uma metodologia focada em pequenas iterações que resultam em entregas parciais do sistema funcionando. Para cumprir esta premissa fornece uma série de práticas que são:

- Modelagem dos Objetos de Domínio: construção de diagrama de classes UML, os diagramas de sequências;
- Desenvolvimento Através de Características: identificação das características do sistema ou seja as funcionalidades definidas pelos usuários;

- Propriedade Individual da Classe: define que cada classe ou conjunto de classes é de responsabilidade de um indivíduo;
- Equipes de Características: equipes cujos componentes possuem as propriedades das classes para a construção de determinada característica;
- Inspeções: devem ocorrer durante e ao final de cada interação;
- Construções Regulares: são efetuadas durante a execução de um conjunto de funcionalidades para detectar erros de integração;
- Administração de Configuração: utilização de sistemas de controle de versão do código implementado (Git, SVN, CVS, dentre outros);
- Relatórios de Resultados: disponíveis a todos membros do projeto;

Os papéis de responsabilidade propostos pelo FDD são divididos em três categorias:

- Chave: Gerente de Projeto, Arquiteto Principal, Gerente de Desenvolvimento, Programador Chefe, Proprietário de Classe e o Especialista no Domínio;
- Suporte: Gerente de Domínio, Gerente de Versão, Especialista na Linguagem, Coordenador de Configuração, *Toolsmith*, Administrador de Sistema;
- Adicional: Testadores, Desenvolvedores e Escritor Técnico.

O FDD, com foco no projeto e construção, propõe as seguintes etapas para um projeto:

- Desenvolvimento de um Modelo Global: tem o intuito de definir o contexto e requisitos do software;
- Construir uma Lista de Características: definição de uma lista completa de todas as características do produto;
- Planejar a Construção por Características: planejamento de execuções de características anteriormente definidas e distribuição aos proprietários;

- Projetar Cada Característica: geração de diagrama de sequência detalhado, além de atualizar e estudar o diagrama de classes;
- Construir Cada Característica: implementação e os testes.

2.1.5 ADAPTATIVE SOFTWARE DEVELOPMENT – ASD

O ASD [FAGUNDES 2005] é um processo dedicado à aprendizagem contínua, com ênfase na resposta a mudanças, reavaliações de premissas e grande colaboração entre desenvolvedores e clientes. Se diferencia por não definir um conjunto de papeis, apenas menciona a necessidade de um Patrocinador. Além disso, não define um conjunto de práticas, somente um conjunto de propriedades que caracterizam o processo de desenvolvimento adaptativo: Dirigido a missões, Baseado em característica, Iterativo, Prazo pré-fixado, Tolerância a mudança e Orientado a riscos.

Seu processo é dividido em ciclos de três fases, sendo elas:

- Especulação: consiste na definição objetivos, um plano baseado em características e os prazos de entrega de artefatos.
- Colaboração: diversas iterações para o desenvolvimento das características previamente definidas durante a Especulação;
- Aprendizado: Revisões de Qualidade e Entrega Final.

2.1.6 DYNAMIC SYSTEM DEVELOPMENT METHOD (DSDM)

O DSDM [MACHADO 2007] é uma metodologia ágil com foco no fornecimento de uma uma solução de qualidade em um curto espaço de tempo, utilizando combinados os conhecimentos de negócio do cliente e os conhecimentos técnicos da equipe de desenvolvimento.

Para atingir o seu foco, se embasa em algumas práticas, sendo elas:

- Timeboxing: é um intervalo de tempo pré-definido. O DSDM vê o projeto como um grande timebox, composto por diversos outros timeboxs de menor duração, com seu escopo de desenvolvimento bem definido.
- Moscow: é uma técnica de priorização de requisitos do sistema. Divide os requisitos em Precisa ter, Deve ter, Pode ter e Não terá, sendo priorizados nesta ordem;
- Modelagem: Ajuda a equipe a adquirir um bom conhecimento técnico sobre o domínio do problema. Não explicita nenhum conjunto de modelos para ser utilizado, porém determina que esta etapa não deve burocratizar o desenvolvimento.
- Prototipação: devem ser desenvolvidos desde o inicio, para facilitar a comunicação com o cliente. O DSDM divide os protótipos em quatro categorias: de Negócio, de Usabilidade, de Desempenho e Capacidade e de Capacidade Técnica.
- Workshops: é a principal técnica do DSDM, seu objetivo é fornecer um mecanismo para a equipe tomar decisões de qualidade em um curto espaço de tempo.
- Testes: devem ocorrer durante todo o período de desenvolvimento, e na sua grande maioria serem feitos por usuários não técnicos.
- Gerenciamento de Configurações: é imprescindível a gerência de configurações e documentação do sistema,

A equipe de desenvolvimento do DSDM é composta por onze papéis, sendo eles os seguintes:

- Patrocinador executivo;
- Visionário: usuário mais experiente que entende os objetivos de negócio;
- Usuário embaixador: informa os objetivos e organiza as políticas da equipe;
- Usuário consultor: conhece o ambiente onde o sistema será aplicado;
- Gerente de projeto;
- Coordenador técnico;
- Líder de equipe;
- Desenvolvedor;
- Redator: é responsável por anotar as decisões tomadas nos encontros.
- Facilitador: é o responsável por controlar o andamento dos workshops.

As fases de um projeto segundo o DSDM são:

- Pré-projeto: durante esta fase ocorre a definição inicial do problema de negócio, escopo e plano inicial para o estudo de viabilidade, orçamento e os responsáveis pelos papeis na equipe;
- Estudo de viabilidade: é executado antes do início do projeto e entre suas iterações, determinando o escopo e os limites de viabilidade. Durante o estudo de viabilidade são realizados: o relatório de viabilidade, o plano resumido e a lista de riscos, além de opcionalmente o protótipo de viabilidade;
- Estudo de negócio: visa obter um maior detalhamento do sistema a ser desenvolvido, além da priorização de requisitos e o plano de desenvolvimento;
- Modelagem funcional: nesta fase são desenvolvidos os protótipos dos requisitos funcionais e não funcionais conforme a priorização definida no estudo de negócio, sendo os protótipos bem sucedidos utilizados para guiar a próxima fase;

- Projeto e construção: neste momento ocorre o refinamento dos protótipos anteriores para abranger todos os requisitos, na prática pode acabar se sobrepondo com a fase de modelagem funcional;
- Implementação: Aprimora os protótipos anteriores, promovendo as alterações necessárias para deixar o sistema desenvolvido operacional;
- Pós-projeto: são as atividades de suporte e manutenção compreendidas após o término do desenvolvimento e entrega final do projeto.

2.1.7 CRYSTAL CLEAR

O Crystal Clear [MACHADO 2007] é uma metodologia que tem por objetivo principal a simplicidade, por isso não estabelece nenhuma de suas práticas propostas como obrigatória, apenas sugere algumas utilizadas com sucesso no desenvolvimento de projetos de software. Fica a cargo da equipe a seleção de práticas a serem utilizadas.

As práticas sugeridas pelo Crystal Clear são:

- Adaptação da Metodologia: tem por objetivo coletar informações sobre experiências anteriores para criar parâmetros iniciais para novos projetos. Para isto se embasa em duas atividades sequenciais, as Entrevistas de Projeto onde são coletados as informações e o Workshop de Adaptação de metodologia onde a equipe se reúne para selecionar os parâmetros iniciais do projeto.
- Workshop de Reflexão: reunião dos membros da equipe de forma periódica para analisar o andamento do projeto, identificar pontos de falha e propor ações efetivas para correção destes problemas.

- Planejamento Rápido: tem como objetivo definir o mapa e a duração de um projeto. Utiliza uma variação do Jogo de Planejamento da técnica *Extreme Programming*, com objetivo de tornar o processo de planejamento menos custoso possível.
- Delphi Estimation: técnica utilizada para efetuar estimativas do tempo necessário para o desenvolvimento de cada parte do software. Nela se reúne toda a equipe envolvida, com intuito de estimar o tamanho do sistema, o tempo necessário conforme o tipo de profissional envolvido, propor sugestões de datas de entregas parciais conforme dependências técnicas e de negócio além da organização de entregas de mesmo tamanho.
- Reunião Diária em Pé: uma curta reunião diária envolvendo os membros da equipe com intuito de dar visibilidade a todos sobre problemas que estejam ocorrendo.
- Design de Iteração Essencial: visa o design e modelagem da interface do sistema, com foco voltado ao usuário.
- Processo em Miniatura: sua utilização tem como objetivo a familiarização da metodologia Crystal Clear pelos membros da equipe. É sugerido que ocorra logo após a prática de Adaptação da Metodologia.
- Programação lado-a-lado: visa facilitar a comunicação entre os desenvolvedores se utilizando da ideia de dispor os computadores de forma que com uma simples virada de cabeça um desenvolvedor possa enxergar a tela do outro.

- Burn-Charts: seu objetivo é tornar visível o andamento do projeto por meio de gráficos, tornando assim mais simples a visualização do desenvolvimento do software frente a um cronograma firmado.

A equipe sugerida pela metodologia é formada pelos seguintes papéis:

- Patrocinador Executivo: responsável pelo financiamento do projeto.
- Usuário Embaixador: usuário com conhecimento operacional sobre o sistema em desenvolvimento, além de conhecimento sobre sua frequência de uso.
- Design Líder: desenvolvedor com maior experiência, com o papel de guiar os outros membros menos experientes da equipe.
- Programador Designer: faz o design dos requisitos além de efetuar a sua codificação.
- Coordenador: ocupação geralmente parcial de um membro da equipe que coordena a equipe como um todo.
- Especialista de Negócio: geralmente um papel executado pelo usuário embaixador, tem como principal função a resolução de dúvidas referentes ao negócio por parte da equipe.
- Testador e Redator: tem a função de testar o software bem como tomar nota das decisões tomadas durante os encontros da equipe para refinamento e análise futura.

O Crystal Clear é composto por um ciclo incremental e iterativo de desenvolvimento, compostos pelas seguintes fases:

- Chartering: é efetuado no início do projeto e tem como objetivos principais a definição da equipe principal e a construção do plano inicial do projeto.
- Intervalo de entrega: é composto por quatro atividades distintas, sendo elas o ajuste do plano inicial, um conjunto de uma ou mais iterações de desenvolvimento, a entrega efetiva ao usuário e uma reflexão sobre o produto em desenvolvimento.
- Iteração: seu objetivo é obter código integrado e testado. É composta de quatro partes, o planejamento de iteração, as atividades diárias, a integração do sistema e a sua finalização.
- Episódio do Desenvolvimento: é o período onde o desenvolvedor implementa uma pequena fração do sistema, realizando testes e fazendo a sua integração. A recomendação é de que episódios de desenvolvimento tenham menos de um dia de duração.

2.1.8 LEAN SOFTWARE DEVELOPMENT (LSD)

É considerada uma aplicação voltada a indústria de software do Lean Development (LD) voltada à indústria de software, um conjunto de práticas e princípios que teve origem na indústria automobilística japonesa. O LD pode ser considerado como a arte e a disciplina de assumir compromissos baseados em fatos ao invés de previsões [MACHADO 2007].

O Lean Software Development(LSD) é considerado a aplicação dos oito princípios do LD na indústria de software. Cada um dos princípios possui um grupo de praticas de suporte associadas.

Os princípios são:

- Iniciar Cedo: As atividades devem iniciar assim que possuírem detalhamento suficiente. As práticas associadas são:
  - Requisitos: Devem ser definidos em paralelo ao desenvolvimento.
  - Arquitetura: Deve ser mantida o mais simples possível, sendo evoluída conforme a necessidade fica aparente.
  - Disciplina: Para iniciar o desenvolvimento devem estar bem definidos diversos aspectos, tais como, controle de versão, execução de testes, integração continua e padrão de codificação.
- Aprender Constantemente: O objetivo não é prever o futuro, e sim estar preparado para responder as mudanças que virão. As práticas associadas são:
  - Desenvolvimento iterativo: Cada incremento deve ser testado, integrado e validado com o usuário.
  - Sincronização: Código sincronizado e integrado constantemente.
- Adiar Compromissos: As decisões devem ser tomadas no último momento possível, para minimizar os riscos envolvidos. As práticas associadas são:
  - O Último momento responsável: As decisões devem ficar para o último momento, a fim de se ter o máximo de conhecimento possível sobre o problema e tomar a melhor decisão possível.
  - Alimentar senso de como responder a mudanças: O código deve estar apto a receber mudanças rapidamente. Devem ser desenvolvidos testes automatizados para garantir a confiabilidade de todo o código após as alterações.

    - Escreva menos código: O mínimo possível de código deve ser escrito para implementar um requisito. Tem objetivo de manter o código o menor possível, facilitando assim a sua manutenção.
- Entregar Rápido: Efetuar entregas de software rapidamente, segue a ideia de entregar hoje o que o cliente necessita hoje. As práticas associadas são:
    - Teoria da fila: Tem objetivo de minimizar o tempo de espera por uma nova funcionalidade requerida.
    - Trabalho autodirigido: Em cada iteração o cliente define as prioridades e a equipe a capacidade de atender a demanda.
- Eliminar o desperdício: É considerado desperdício tudo aquilo que não agrega valor ao cliente. As práticas associadas são:
    - Caminhar na cadeia de valor: Deve eliminar os passos desnecessários nas etapas entre a requisição de uma funcionalidade pelo cliente e a sua entrega.
    - Identificar o desperdício: Identificar o desperdício durante o processo de desenvolvimento.
    - Escopo: minimizar o número de funcionalidades desenvolvidas e nunca utilizadas.
- Equipe com poder de decisão: Ninguém melhor que a equipe técnica para tomar decisões técnicas. As práticas associadas são:
    - Design do processo de criação: Cada membro deve ter autonomia para definir e melhorar seu processo.
    - Um processo de gerenciamento: Desenvolvedores devem se reunir com gerentes e propor mudanças afim de melhorar o processo.

    - Design do processo de software: Revisar sempre o processo de desenvolvimento em busca de falhas, promovendo assim seu continuo melhoramento.
    - Condescendência: Ninguém melhor que a equipe de desenvolvimento para saber do seu trabalho e problemas. Sua opinião deve ser respeitada.
    - Liderança: Deve existir um líder técnico com grande domínio do problema para sem consultado pelos outros membros.
    - Pericia: A equipe deve ser formada por pessoas altamente qualificadas para realizar o trabalho.
- Construindo com integridade: Não há espaço para trabalho descuidado. As práticas associadas são:
    - Repensando testes: Parte-se do princípio que se um erro pode ocorrer, então ele irá ocorrer, sendo assim são necessários meios de evitá-los sempre.
    - Desenvolvimento por características: Todos os testes devem ser executados durante todo o processo de desenvolvimento, evitando assim regressão de outras funcionalidades anteriormente implementadas.
- Evitar sub-otimização: Otimizar o sistema sempre olhando para o todo, e não somente para uma subparte. As práticas associadas são:
    - Desagregação: O sistema deve ser visto como um todo, e não em pequenos pedaços isolados.
    - O cartão de resultados do projeto: Dois aspectos precisam estar claro para todos os envolvidos: Quando o projeto estará completo? e O que é necessário para o projeto ser considerado um sucesso?

- Medidas de desempenho: É necessária a coleta de métricas de desempenho da equipe para promover o melhoramento contínuo do processo e identificar falhas e gargalos.
- Além dos limites da empresa: A empresa deve procurar melhorar, além de seus processos, também o de empresas parceiras.
- O propósito dos contratos: Contratos devem ser utilizados como um mecanismo para obter melhores resultados.
- Contrato de objetivos: Os objetivos para o sucesso do projeto devem estar estabelecidos, além de uma divisão justa de riscos e lucro envolvidos.

2.1.9 COMPARAÇÃO E ANÁLISE DOS MÉTODOS

O Framework de Práticas Ágeis foi proposto inicialmente em [Fagundes 2005], e posteriormente estendido em [Machado 2005] com a agregação de práticas ágeis de outros métodos ágeis. O framework consiste em um conjunto de práticas ágeis de desenvolvimento de software que permite a escolha das práticas mais convenientes para um determinado projeto no momento da definição do processo [Vilain 2007].

O agrupamento de práticas no framework é efetuado com base na lista de atividades do processo de desenvolvimento incremental. Estas atividades, ilustradas na Figura 2, formam o ciclo de desenvolvimento e são compostas pelo conjunto de práticas do framework.

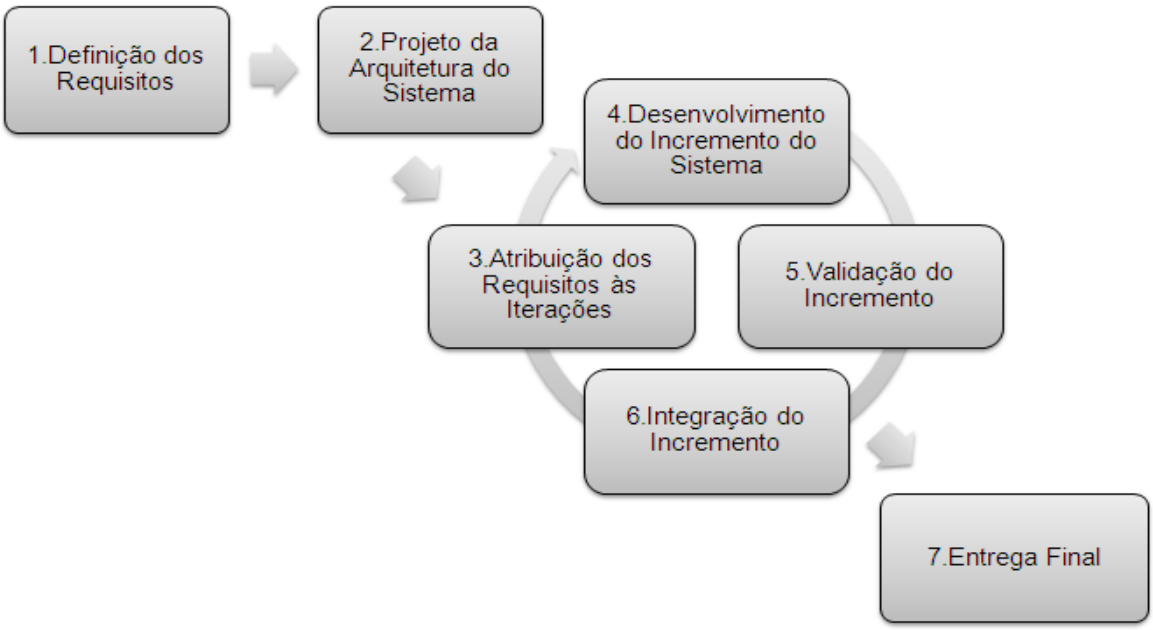

Figura 2 - Ciclo do framework

A seguir, estão as práticas do framework, juntamente com seu(s) métodos ágeis de origem, agrupadas de acordo com a sua atividade.

- Atividade de definição de requisitos:
  - Lista de requisitos: Scrum, FDD
  - Modelagem geral: FDD
  - Documentação inicial: AM
- Atividade de projeto da arquitetura do sistema:
  - Projeto de arquitetura do sistema: XP, Scrum, FDD, DSDM, Crystal Clear, LSD, AM
- Atividade de atribuição dos requisitos das iterações:
  - Planejamento da iteração: XP, Scrum, FDD, ASD, DSDM, Crystal Clear, and LSD
- Atividade de desenvolvimento do incremento do sistema:

    - Projeto de iteração: FDD
    - Estórias de usuário: XP e Crystal Clear
    - Casos de uso: FDD
    - Desenvolvimento: XP, Scrum, FDD, ASD, DSDM, Crystal Clear, e LSD
    - Testes de unidade: XP
    - Testes de aceitação: XP
    - Programação em pares: XP
    - Desenvolvimento lado a lado: Crystal clear
    - Refatoração: XP
    - Integração contínua: XP e Crystal Clear
    - Desenvolvimento coletivo do código: XP
    - Reuniões diárias: XP, Scrum e Crystal Clear
- Atividade de validação do incremento:
    - Inspeção de código: FDD e ASD
- Atividade de integração do incremento:
    - Reunião de revisão da iteração: Scrum, FDD e ASD.
- Atividade de entrega final:
    - Entrega do sistema: XP, Scrum, FDD, ASD, DSDM, e Crystal Clear
    - Documentação breve: XP, Scrum, AM e Crystal Clear.

Além disto, em [Vilain 2007], ficam definidas as seguintes práticas como obrigatórias: as práticas de Desenvolvimento e Entrega do sistema.

3 ANÁLISE DE AGILIDADE

De posse do framework de práticas ágeis [VILAIN 2007], [SHOEPPING 2012] explorou outra possibilidade: a avaliação de agilidade das práticas do framework. Para tanto, inicialmente identificou a adoção e efetividade de cada prática de maneira individual, de acordo com uma pesquisa online [AMBLER 2007]. Posteriormente, utilizando-se de dados da mesma pesquisa, foram abordados os relacionamentos e iterações entre as práticas, afim de identificar grupos de práticas mais efetivos.

O primeiro passo tomado em [SCHOEPPING 2012] foi o de correlacionar as práticas existentes no framework de práticas ágeis [VILAIN 2007] com as práticas da pesquisa online [AMBLER 2007]. Com isto, três grupos de práticas distintos foram identificados: Práticas idênticas às da pesquisa, Práticas muito próximas às da pesquisa e Práticas ausentes na pesquisa. Estas relações estão descritas respectivamente na Tabela 1, Tabela 2 e Tabela 3 abaixo.

Tabela 1 - Práticas idênticas às da pesquisa

Lista de requisitos	Modelagem inicial de requisitos ágeis
Projeto da arquitetura do sistema	Modelagem inicial de arquitetura ágil
Planejamento da iteração	Jogo de planejamento
Casos de uso	Casos de uso (detalhados)
Testes de unidade	Desenvolvimento orientado a testes
Testes de aceitação	Testes de aceitação pelo cliente
Programação em pares	Programação em pares
Refatoração	Refatoração de código
Integração contínua	Integração contínua de código
Reuniões diárias	Reuniões diárias em pé
Desenvolvimento coletivo de código	Propriedade coletiva de código

Inspeção de código	Inspeção de código
Desenvolvimento	Obrigatório
Entrega do sistema	Obrigatório

Tabela 2 - Práticas muito próximas às da pesquisa

Modelagem geral	Design evolutivo
Histórias do usuário	Casos de uso (leves)
Projeto da iteração	Design simples
Desenvolvimento lado a lado	Times reunidos

Tabela 3 - Práticas não existentes na pesquisa

Documentação inicial	--
Reunião de revisão da iteração	--
Documentação breve	--

3.1 ADOÇÃO E EFETIVIDADE DAS PRÁTICAS ÁGEIS INDIVIDUALMENTE

Após correlacionadas, pode-se extrair um gráfico com a adoção das práticas segundo a pesquisa de [AMBLER 2007] referenciado na Figura 3. Para gerar o gráfico, valores de 1 a 5 na pesquisa foram contabilizados como prática em uso, enquanto as práticas não respondidas ou que o entrevistado não quis responder foram contabilizadas como não utilizadas.

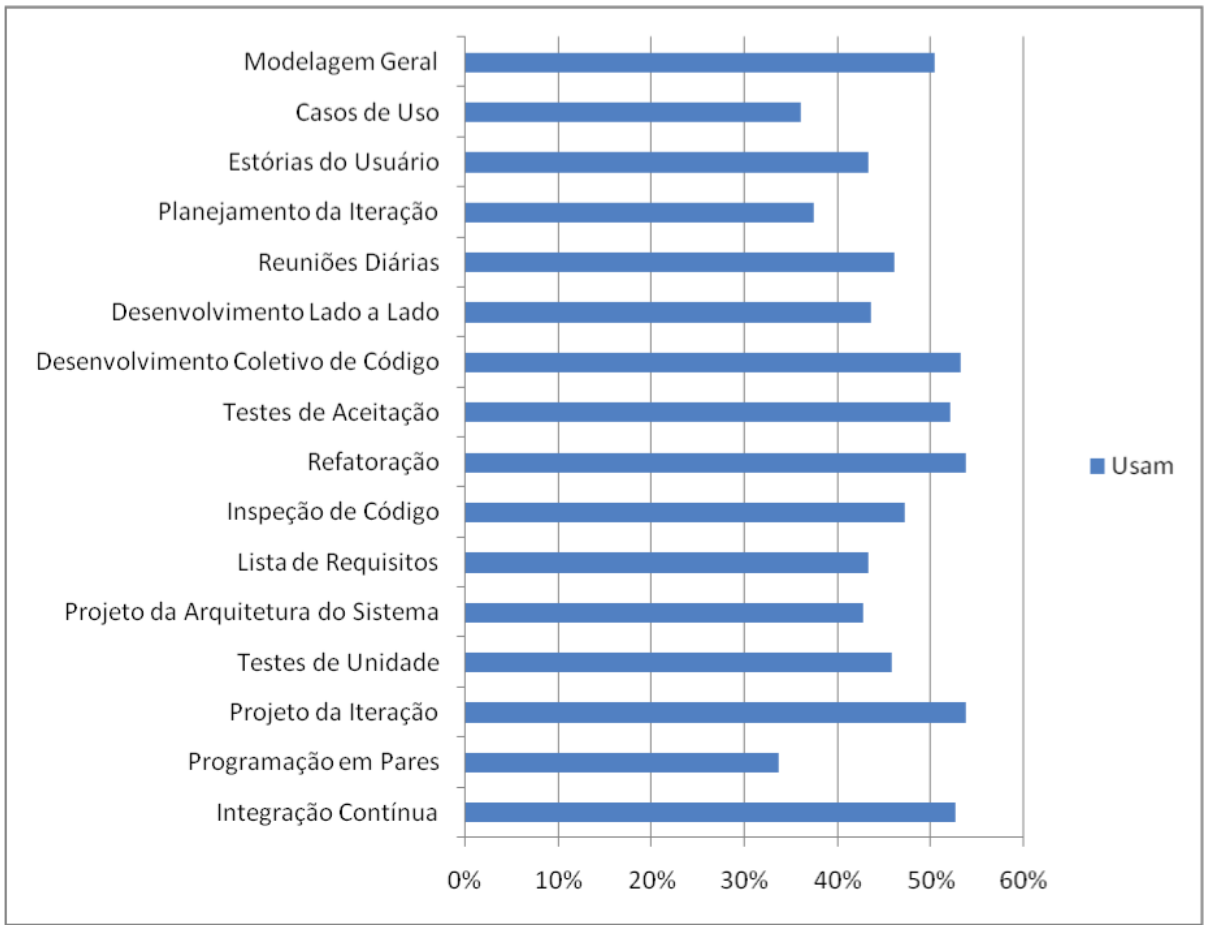

Figura 3 - Gráfico de adoção de práticas verificado em [SCHOEPPING 2012]

Para obtenção dos dados relativos a efetividade das práticas, foram utilizadas a fórmula abaixo, de acordo com as respostas dos entrevistados por [AMBLER 2007]. Posteriormente foi obtida a Tabela 4 contendo a efetividade das práticas do framework.

$$E = \frac{(Na * 5) + (Nb * 4) + (Nc * 3) + (Nd * 2) + (Ne * 1)}{Tr * Nr * Ns}$$

Onde:

E = efetividade geral da prática;

Na = número de respondentes que atribuíram cinco à efetividade da prática;

Nb = número de respondentes que atribuíram quatro à efetividade da prática;

Nc = número de respondentes que atribuíram três à efetividade da prática;

Nd = número de respondentes que atribuíram dois à efetividade da prática;

Ne = número de respondentes que atribuíram um à efetividade da prática;

Tr = total de respondentes que responderam a pergunta (exclui os que pularam);

Nr = número de respondentes que não responderam a pergunta;

Ns = número de respondentes que não souberam responder a pergunta.

Tabela 4 - Práticas numeradas utilizadas na Figura 3

Lista de requisitos	3,57
Modelagem geral	3,70
Documentação inicial	---
Projeto da arquitetura do sistema	3,60
Planejamento da iteração	3,49
Histórias do usuário	3,52
Casos de uso	2,96
Projeto da iteração	3,78
Testes de unidade	3,79
Testes de aceitação	3,94
Desenvolvimento	---
Programação em pares	3,40
Desenvolvimento lado a lado	4,00
Refatoração	3,81
Integração contínua	4,11
Reuniões diárias	3,88
Desenvolvimento coletivo de código	3,80
Inspeção de código	3,43
Reunião de revisão da iteração	---

Documentação breve	---
Entrega do sistema	---

3.2 MÉTRICAS DE EFETIVIDADE

De posse dos resultados acima [SHOEPPING 2012] seguiu os estudos visando obter a efetividade combinada das práticas ágeis acima listadas, isto é, quais práticas quando utilizadas simultaneamente tem um maior efeito positivo no processo que está sendo descrito. Para tanto, utilizou-se de uma técnica estatística chamada análise de agrupamentos, e mais especificamente o agrupamento em árvore, por não se saber inicialmente quantos grupos seriam formados, formando assim os dendogramas de grupos.

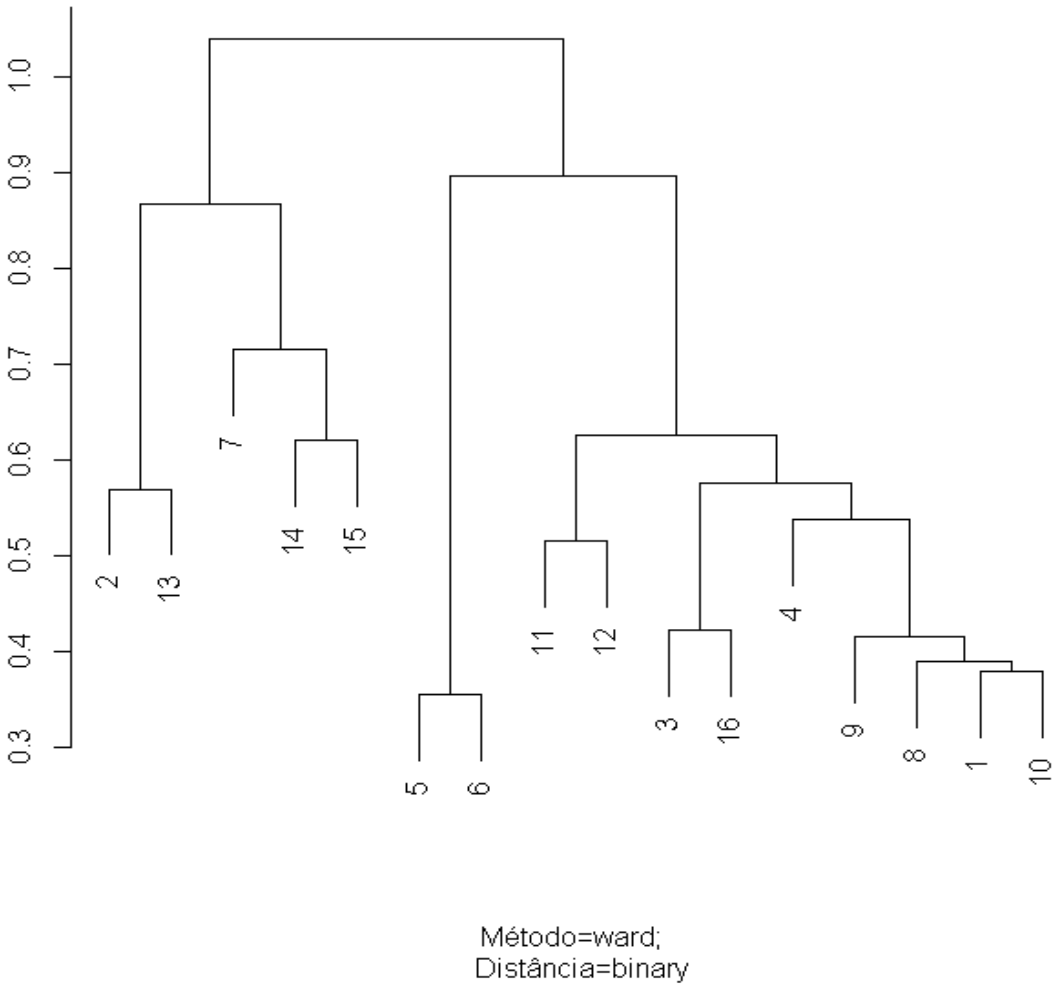

Figura 4 - Dendograma de práticas efetivas proposto por [SCHOEPPING 2012].

Tabela 5 - Práticas numeradas utilizadas na Figura 4

1	Integração contínua
2	Programação em pares
3	Projeto da iteração
4	Testes de unidade
5	Projeto da arquitetura do sistema
6	Lista de requisitos
7	Inspeção de código
8	Refatoração
9	Testes de aceitação
10	Desenvolvimento coletivo de código
11	Desenvolvimento lado a lado
12	Reuniões diárias
13	Planejamento da iteração
14	Histórias do usuário
15	Casos de uso
16	Modelagem geral

Após formar este dendograma, o passo subsequente foi o da formação efetiva dos grupos. Em um dendograma, quando mais abaixo segundo a escala da lateral (Height) da Figura 4, mais comumente as práticas próximas são efetivas quando utilizadas concorrentemente. A partir disto, utilizou-se o Height de 0.5 para obter os grupos descritos na Tabela 6

Tabela 6 - grupos de práticas formados pelo dendograma

A	5 + 6	Projeto da arquitetura do sistema
		Lista de requisitos
B	10+1+8+9	Desenvolvimento coletivo de código

		Integração contínua
		Refatoração
		Testes de aceitação
C	3+16	Projeto da iteração
		Modelagem geral
D	11+12	Desenvolvimento lado a lado
		Reuniões diárias

Após a formação destes grupos foram definidos alguns passos afim de maximizar a efetividade do processo proposto, sendo estes:

1. Selecionar as práticas obrigatórias do framework: Desenvolvimento e Entrega do sistema.
2. Escolher as práticas desejadas
3. Considerar as seguintes diretrizes para auxiliar a definição do processe;
    a. Caso todas as práticas do grupo B não tenham sido selecionadas, sugere-se que estas sejam selecionadas, em vista que este grupo contém as práticas com mais alto nível de efetividade.
    b. Se alguma prática do grupo A foi selecionada, porém não todas, sugere-se a adoção das outras práticas do grupo.
    c. Se alguma prática do grupo C foi selecionada, porém não todas, sugere-se a adoção das outras práticas do grupo.
    d. Se alguma prática do grupo D foi selecionada, porém não todas, sugere-se a adoção das outras práticas do grupo.

A ideia destes passos propostos não é estabelecer conjuntos pré-definidos de práticas a serem utilizadas, diminuindo assim a flexibilidade original do framework, e sim

auxiliar na seleção com combinações de práticas consideradas boas pelos usuários da pesquisa de [AMBER 2007].

4 PROJETO E DESENVOLVIMENTO DA FERRAMENTA

Neste capítulo será descrito o projeto da ferramenta, as principais regras de negócio, estórias de usuário, modelagem de classes e modelos de dados criados, e também as iterações propostas de desenvolvimento.

Para o desenvolvimento da ferramenta, o próprio Framework de Práticas Ágeis foi utilizado para a definição de um processo de desenvolvimento. As práticas selecionadas segundo as atividades do ciclo iterativo do framework foram:

- Modelagem geral:
    - Modelagem geral
- Projeto da arquitetura do sistema:
    - Projeto de arquitetura do sistema
- Atribuição dos requisitos das iterações:
    - Planejamento de iteração
- Desenvolvimento do incremento do sistema:
    - Estórias de usuário
    - Refactoring
    - Integração contínua
- Validação do incremento:
    - Inspeção de código
- Integração do incremento:
    - Reuniões de revisão de iteração
- Entrega final:
    - Entrega final do sistema

4.1 VISÃO GERAL

A ferramenta proposta tem como objetivo central auxiliar o usuário interessado em definir seu próprio modelo de processo nesta tarefa. Para isto, a ferramenta deve mostrar as práticas escolhidas e posteriormente utilizar-se das regras de sugestão de outras práticas simultâneas, de acordo com o descrito em [SCHOEPPING 2012].

Como os métodos ágeis estão constantemente evoluindo, para que não ocorra uma estagnação da ferramenta proposta, torna-se necessária a criação de perfil de administrador do sistema, para permitir a contínua melhoria do framework e adequação a estas mudanças e alterações.

Do ponto de vista do administrador, a ferramenta também deve ser suficientemente flexível e configurável para que possam ser inseridas novas práticas e grupos que venham a ser criados no futuro e passem a fazer parte do framework proposto em [VILAIN 2007].

Outro aspecto relevante, já que trata-se de uma ferramenta de domínio publico, é de que o código, configurações de ambiente de desenvolvimento e documentação devem ser o mais simples possíveis, visando a possibilidade de contribuições futuras de outros desenvolvedores. Para potencializar isto, todo o código fonte da aplicação foi escrito utilizando a língua inglesa, um padrão internacional.

4.2 REGRAS DE NEGÓCIO

Regras de negócio definem as políticas de um negócio, afim de satisfazer os objetivos de um sistema atendendo aos anseios dos seus usuários e provendo bom uso de recursos.

O conjunto de regras de negócio relativas ao desenvolvimento da ferramenta podem ser agrupadas em dois grupos distintos:

- Relativas à criação de um processo: todos os critérios para seleção de práticas, definidos em [VILAIN 2007], considerando a obrigatoriedade de práticas.
- Relativas à sugestão de práticas ao usuário: as sugestões das práticas a serem selecionadas seguem os critérios definidos por [SCHOEPPING 2012].

4.3 LINGUAGEM DE PROGRAMAÇÃO

A linguagem de programação escolhida para ser utilizada no desenvolvimento da ferramenta web em questão foi a Linguagem Java[3]. Java é uma linguagem de programação orientada a objetos, amplamente conhecida e utilizada, facilitando assim o desenvolvimento do projeto. Além disso, a escolha de Java, deve-se também ao fato da linguagem possuir um kit de desenvolvimento de software aberto, sob licença GNU (General Public License).

Pode-se ainda salientar o grande número de APIs já desenvolvidas e disponíveis para auxiliar no processo de desenvolvimento. Dentre as APIs utilizadas para criação da ferramenta podemos enfatizar uma em especial, o Spring Framework[4].

O Spring Framework é um framework open-source para a plataforma Java criado por Rod Johnson em 2002 juntamente com a publicação de seu livro Expert One-on-One J2EE Design and Development, baseia-se no padrão de inversão de controle e injeção de dependências. Ele utiliza o modelo MVC (do inglês Model, View and Controller) para organizar seus componentes e baseia-se nos padrões de projeto de inversão de controle e

[3] http://www.java.com
[4] www.springsource.org

injeção de dependências. O framework Spring se encarrega de instanciar as classes Java do projeto e definir dependências internas através de arquivos de configuração XML.

Outra grande vantagem do Spring Framework vem das diversas bibliotecas de apoio disponíveis, valendo a pena ser citada uma de suas principais, o Spring Security. Esta biblioteca provê principalmente funcionalidades referentes à autenticação de usuários e posterior autorização de acesso a conteúdo do site.

4.4 FERRAMENTAS E BIBLIOTECAS UTILIZADAS

Para facilitar a tarefa de desenvolvimento, diversas ferramentas e bibliotecas serão utilizadas. Dentre elas podem ser citadas como principais as seguintes:

- Eclipse IDE[5] (Integrated Development Environment): ide de desenvolvimento multi-linguagens amplamente utilizada em todo o mundo, com grande documentação disponível online, além de plug-ins disponíveis para outras ferramentas utilizadas, como Spring e Git por exemplo.
- Git[6]: ferramenta de controle de versões de código, criada por Linus Torvalds, para utilização no desenvolvimento do kernel do Linux. Atualmente é amplamente utilizado por projetos de código aberto e também em ambientes corporativos.
- Maven[7]: ferramenta para gerenciamento e automação de projetos em Java. A partir de um arquivo XML de configuração, o processo de construção de um projeto de software, suas dependências externas e internas e toda a sequência de construção é descrita. Outra grande vantagem do Maven é a possibilidade

[5] http://www.eclipse.org
[6] http://www.git-scm.com
[7] http://maven.apache.org

de utilização de repositórios externos para obtenção destas bibliotecas, facilitando assim o processo de construção de projetos Java.

- Hibernate[8]: é uma biblioteca para mapeamento objeto-relacional em Java. Seu principal objetivo é diminuir a complexidade do código em projetos Java que necessitem trabalhar com bancos de dados objeto-relacionais. Sua principal característica é a transformação de classes Java para tabelas em um banco de dados relacional, liberando assim o desenvolvedor deste trabalho.
- MySQL[9]: é um SGBD (Sistema de Gerenciamento de Bancos de Dados) relacional, que utiliza linguagem SQL (Linguagem de Consulta Estruturada) para manipulação dos dados. Atualmente é um dos bancos de dados mais difundidos no mundo e sendo também um software de domínio publico. Seu uso no sistema fica restrito ao ambiente de produção da aplicação. Foi utilizado para o ambiente de produção da aplicação.
- HSQLDB[10]: é um SGBD de código aberto, escrito totalmente em Java, que utiliza poucos recursos para sua execução, sua principal utilização é em ambientes desktop com poucos recursos computacionais disponíveis, ou também em ambiente de desenvolvimento de softwares mais complexos, tornando assim mais simples a sua configuração. Devido as suas características, o HSQLDB foi utilizado no ambiente de desenvolvimento da aplicação.

[8] http://www.hibernate.org
[9] http://www.mysql.com
[10] http://www.hsqldb.org

- Twitter Bootstrap[11]: é um conjunto de HTML, CSS e JavaScript que proporciona uma maneira rápida e eficiente de desenvolver o front-end de web sites e aplicações web. Nasceu de um projeto para documentar padrões de interface de usuário e experiência do usuário dentro do Twitter, porém tomou proporções muito maiores e hoje é um dos projetos abertos mais populares no GitHub.

4.5 DESENVOLVIMENTO

Nesta seção está descrito o ciclo de desenvolvimento da ferramenta, as principais funcionalidades identificadas, além das iterações montadas e diagramas de sequência demonstrando os principais fluxos da aplicação.

4.5.1 FUNCIONALIDADES E ESTÓRIAS DE USUÁRIO

Antes de iniciar a programação da ferramenta, foram levantadas as principais funcionalidades da ferramenta, e a partir destas, descritas estórias de usuário a serem implementadas priorizadas e implementadas nos ciclos iterativos posteriores.

As funcionalidades básicas identificadas quanto às necessidades de um usuário são:

- Identificar-se no sistema e ter acesso aos seus processos definidos;
- Definir e editar seus processos;
- Definir e editar seus *templates* de processos;
- Definir e editar processos criados à partir de *templates* previamente criados;
- Receber sugestões quanto à efetividade do conjunto de práticas selecionadas durante a definição de um processo.

[11] http://twitter.github.io/bootstrap/

Com a premissa da ferramenta de manter-se flexível à adição de novas práticas no framework e também alteração das práticas atualmente existentes, faz-se necessária a criação de um outro perfil de acesso no sistema: o administrador. Suas funcionalidades são as seguintes:

- Identificar-se como administrador no sistema;
- Definir e editar as atividades dos ciclos do framework (Figura 2);
- Definir e editar as práticas que compõe as atividades;
- Definir e editar os grupos de práticas a serem sugeridos ao usuário, segundo critérios de efetividade propostos por em [SCHOEPPING 2012].

Após o levantamento destas funcionalidades listadas acima, efetuou-se o levantamento das estórias de usuário a serem desenvolvidas, descritas na Tabela 7.

Tabela 7 - Estórias de usuário a serem desenvolvidas

1	Cadastro e login de usuário
2	Controle de acesso de usuário
3	Criação e manutenção de processos
4	Criação e manutenção de templates processos
5	Criação e manutenção de processos a partir de templates definidos
6	Criação e manutenção de atividades
7	Criação e manutenção de práticas
8	Criação e manutenção de grupos de atividades
9	Sugerir práticas afim de melhorar a efetivade do processo

As estórias de usuário definidas pela Tabela 7 foram organizadas em três iterações de desenvolvimento, descritas em detalhes nas próximas seções.

4.5.2 MODELO CONCEITUAL

Antes de iniciar o desenvolvimento da aplicação, foi desenvolvido o modelo conceitual da ferramenta. O modelo contempla os principais conceitos e associações necessários na aplicação e pode ser visto na Figura 5 abaixo.

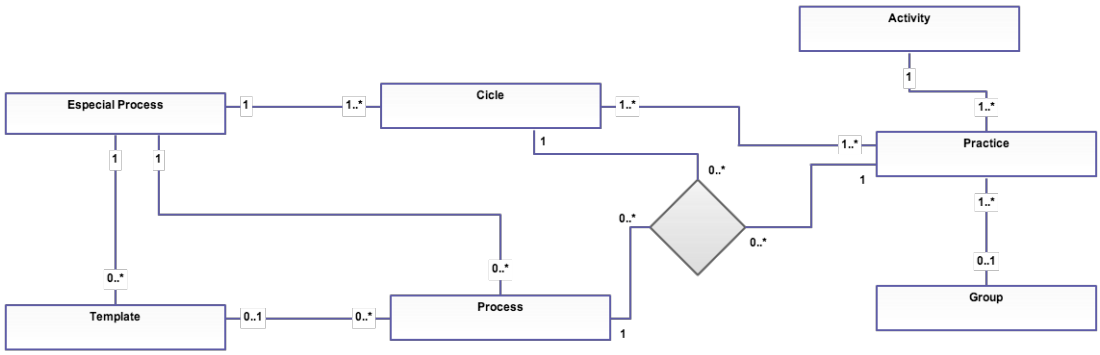

Figura 5 - Modelo Conceitual

4.5.3 ITERAÇÃO 1

A primeira iteração desenvolvida iniciou-se em 16/02/2013 e foi finalizada em 28/02/2013, e englobou três estórias de usuário: Cadastro e login de usuário do sistema, administrador ou projetista, Controle de acesso de usuário e Criação e manutenção de práticas. Não faz sentido iniciar pelas funcionalidades de um dos perfis de usuário do sistema, sem possibilitar o seu cadastro e login, além de isolamento de acesso entre os papéis.

Tabela 8 - Iteração 1

Data de Início: 16/02/2013		**Data de Término:** 28/02/2013
Tema: Infraestrutura de usuário e criação de prática		
Estórias a serem desenvolvidas		
	1	Cadastro e login de usuário
	2	Controle de acesso de usuário
	3	Criação e manutenção de práticas

4.5.3.1 CADASTRO E LOGIN DE USUÁRIO

Esta estória de usuário consiste na possibilidade de criação de uma conta de usuário no sistema, e posterior login utilizando-se desta. Quando acessar o sistema, o e-mail e senha serão solicitados ao usuário. Caso este já possua uma conta ativa, pode informá-los e iniciar o uso da ferramenta. Caso contrário, tem-se a opção de clicar em "Nova conta", preencher seu nome, e-mail e senha para obter as credenciais.

4.5.3.2 CONTROLE DE ACESSO DE USUÁRIO

Esta estória de usuário vem da necessidade do papel de administrador, cujas funcionalidades exigem um maior conhecimento do framework de práticas ágeis, sendo assim restritas afim de evitar definições errôneas que afetem aos outros usuários da ferramenta. O usuário administrador tem seu cadastro criado pela ferramenta automaticamente, e seus dados entregues à pessoa com estas competências. Todos os outros usuários tem o perfil de projetista e não podem acessar a área administrativa da ferramenta.

4.5.3.3 CRIAÇÃO E MANUTENÇÃO DE PRÁTICAS

A inclusão de uma nova prática no framework é uma atividade exclusiva do administrador do sistema, e tem como motivação a capacidade de adaptação da ferramenta às práticas de novos processos ágeis que futuramente surgirão, ou que ainda não tenham sido contemplados pelo framework [SHOEPPING 2012]. Na Figura 6 está descrito o diagrama de sequência para a criação de uma prática.

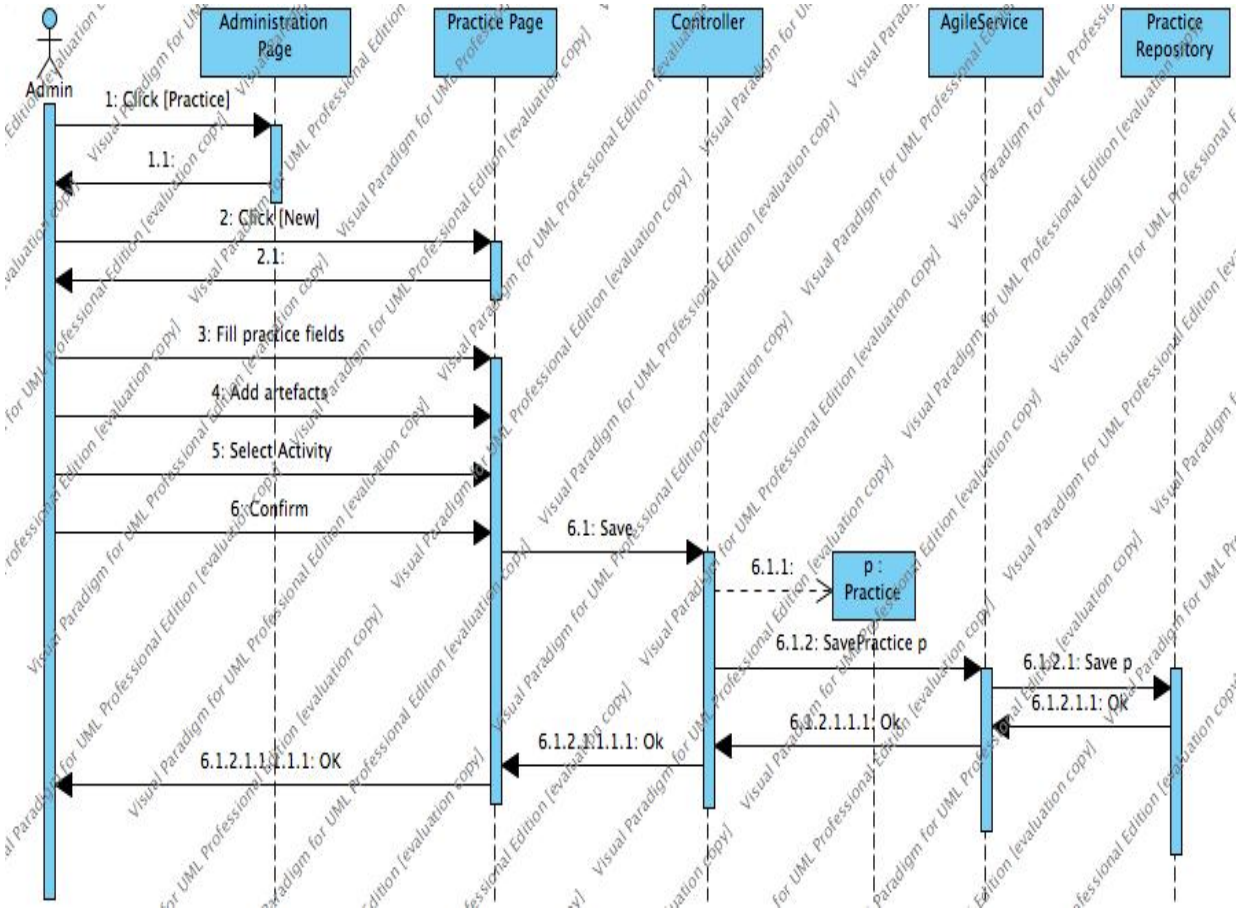

Figura 6 - Diagrama de sequência para inclusão de nova prática

O fluxo consiste em clicar no link “Practice” do menu de administração, posteriormente em “New” e preencher os campos da prática. Quando confirmado pelo usuário, o sistema chama a classe controladora de práticas, que efetua as validações sobre o objeto, e delega a ação de persistência do objeto para uma classe de serviço central, que posteriormente repassa o objeto ao repositório de prática para a inclusão efetiva na base de dados da aplicação.

4.5.4 ITERAÇÃO 2

A segunda iteração desenvolvida, iniciou-se em 01/03/2013 e foi finalizada em 16/03/2013, e englobou três estórias de usuário: Criação e manutenção de atividades, Criação e manutenção de grupos de atividades e Criação e manutenção de *templates* de processos. Optou-se por concluir a área administrativa da ferramenta e iniciar com a definição de *template* de processo.

Tabela 9 - Iteração 2

<table>
<tr><td colspan="3"></td></tr>
<tr><td colspan="2">Data de Início: 01/03/2013</td><td>Data de Término: 16/03/2013</td></tr>
<tr><td colspan="3">Tema: Área administrativa completa e templates de processo.</td></tr>
<tr><td rowspan="4">Estórias a serem desenvolvidas</td><td colspan="2"></td></tr>
<tr><td>6</td><td>Criação e manutenção de atividades</td></tr>
<tr><td>8</td><td>Criação e manutenção de grupos de atividades</td></tr>
<tr><td>4</td><td>Criação e manutenção de templates de processos</td></tr>
</table>

4.5.4.1 CRIAÇÃO E MANUTENÇÃO DE ATIVIDADES

Esta estória de usuário consiste na possibilidade de criação e manutenção de atividades. Estas atividades possuem um nome e são compostas por uma ou mais práticas segundo o framework de práticas ágeis. As atividades mostradas na Figura 2 deste trabalho serão criadas inicialmente diretamente na massa de dados inicial do sistema, sendo assim necessária a atuação do administrador nesta área somente quando ocorrerem mudanças.

4.5.4.2 CRIAÇÃO E MANUTENÇÃO DE GRUPOS DE PRÁTICAS

Como descrito na seção 3.2 deste trabalho, grupos de práticas são sugeridos ao usuário para aumentar a efetividade do processo sendo criado. Contudo, a criação destes grupos é tarefa exclusiva do administrador do sistema, pois esta inclusão deve seguir critérios estatísticos para ser válida. Na Figura 7 está descrito o diagrama de sequência para a definição de um grupo de práticas a serem sugeridos ao usuário.

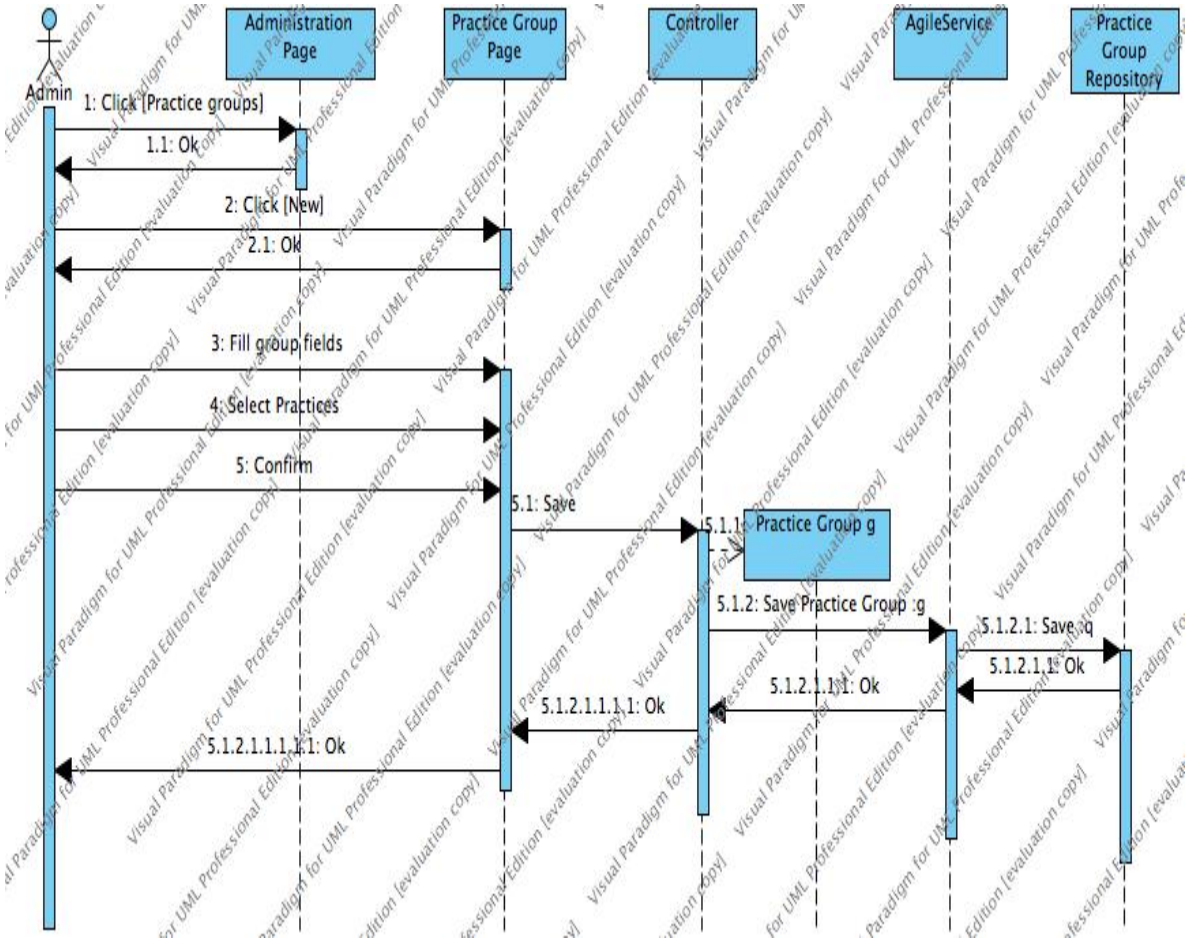

Figura 7 - Diagrama de sequência de criação de grupo de práticas

O fluxo consiste em clicar no link “Practice Groups” do menu de administração, posteriormente em “New”, preencher os campos do grupo e adicionar as práticas desejadas ao grupo. Quando confirmado pelo usuário, o sistema chama a classe controladora de grupos, que efetua as validações sobre o objeto, e delega a ação de persistência do objeto para uma classe de serviço central, que posteriormente repassa o objeto ao repositório de grupos para a inclusão efetiva na base de dados da aplicação.

4.5.4.3 CRIAÇÃO E MANUTENÇÃO DE TEMPLATES DE PROCESSOS

Esta estória de usuário consiste na possibilidade de criação e manutenção de templates de processos. Esses templates nada mais são que conjuntos pré definidos de práticas a serem referenciados posteriormente na definição de processos.

4.5.5 ITERAÇÃO 3

A terceira iteração desenvolvida iniciou-se em 17/03/2013 e foi finalizada em 03/04/2013, e tratou de finalizar as funcionalidades faltantes para definição de processos segundo o framework de práticas ágeis, a principal funcionalidade da ferramenta. As estórias de usuário que compõe esta iteração são: Criação e manutenção de processos, Criação e manutenção de processos a partir de *templates* definidos e Sugerir práticas afim de melhorar a efetividade do processo.

Tabela 10 - Iteração 3

Data de Início: 17/03/2013		**Data de Término:** 03/04/2013
Tema: Permitir a definição completa de um processo		
Estórias a serem desenvolvidas	**3**	Criação e manutenção de processos
	4	Criação e manutenção de processos a partir de *templates* definidos
	9	Sugerir práticas afim de melhorar a efetividade do processo

4.5.5.1 CRIAÇÃO DE PROCESSOS

Esta estória de usuário, a central da ferramenta, consiste na criação de um processo com base no framework de práticas ágeis.. Esta funcionalidade consiste na definição de um processo ágil, com base na seleção de práticas cadastradas previamente no sistema.

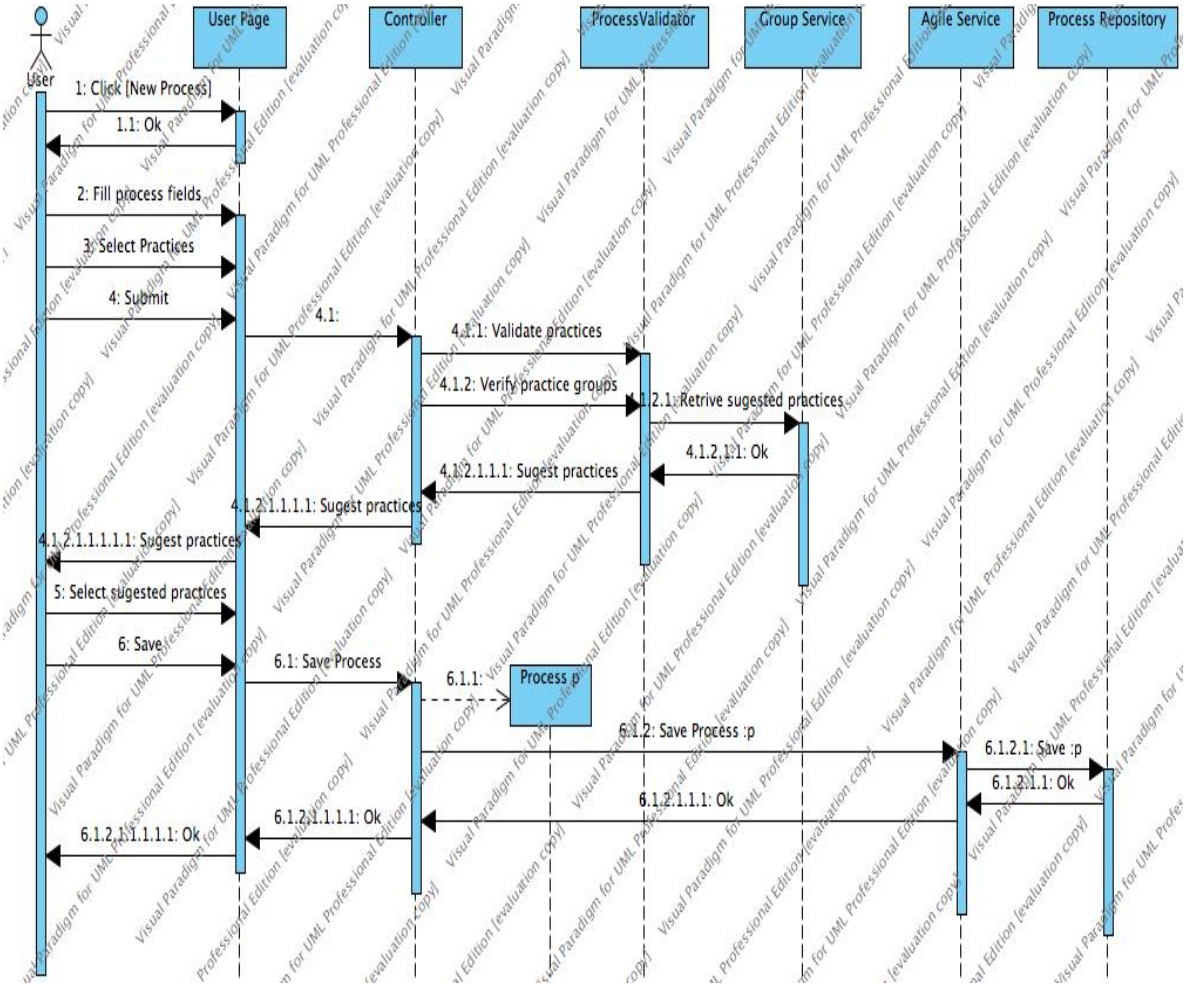

Figura 8 - Diagrama de sequência de criação de processo

O fluxo inicia-se quando o usuário clica em “New process”, preenche os campos de um processo, seleciona as práticas e submete ao servidor. Neste momento, a aplicação valida as práticas segundo as regras, impostas por [SHOEPPING 2012], quanto à

simultaneidade de algumas práticas e utiliza os grupos de práticas cadastrados para sugerir ao usuário a inclusão de outras, afim de melhorar a efetividade do processo montado. Após isso o usuário avalia as sugestões, inclui ou não estas práticas sugeridas e conclui o processo.

4.5.6 MODELAGEM DO SISTEMA

Após decorridas as três iterações acima citadas, o modelagem do sistema ficou como descrita na Figura 9. Pode-se notar a definição de duas entidades não citadas anteriormente: BaseEntity e NamedEntity. Estas entidades servem apenas para evitar a repetição dos atributos id e name, repetidos na maioria das entidades.

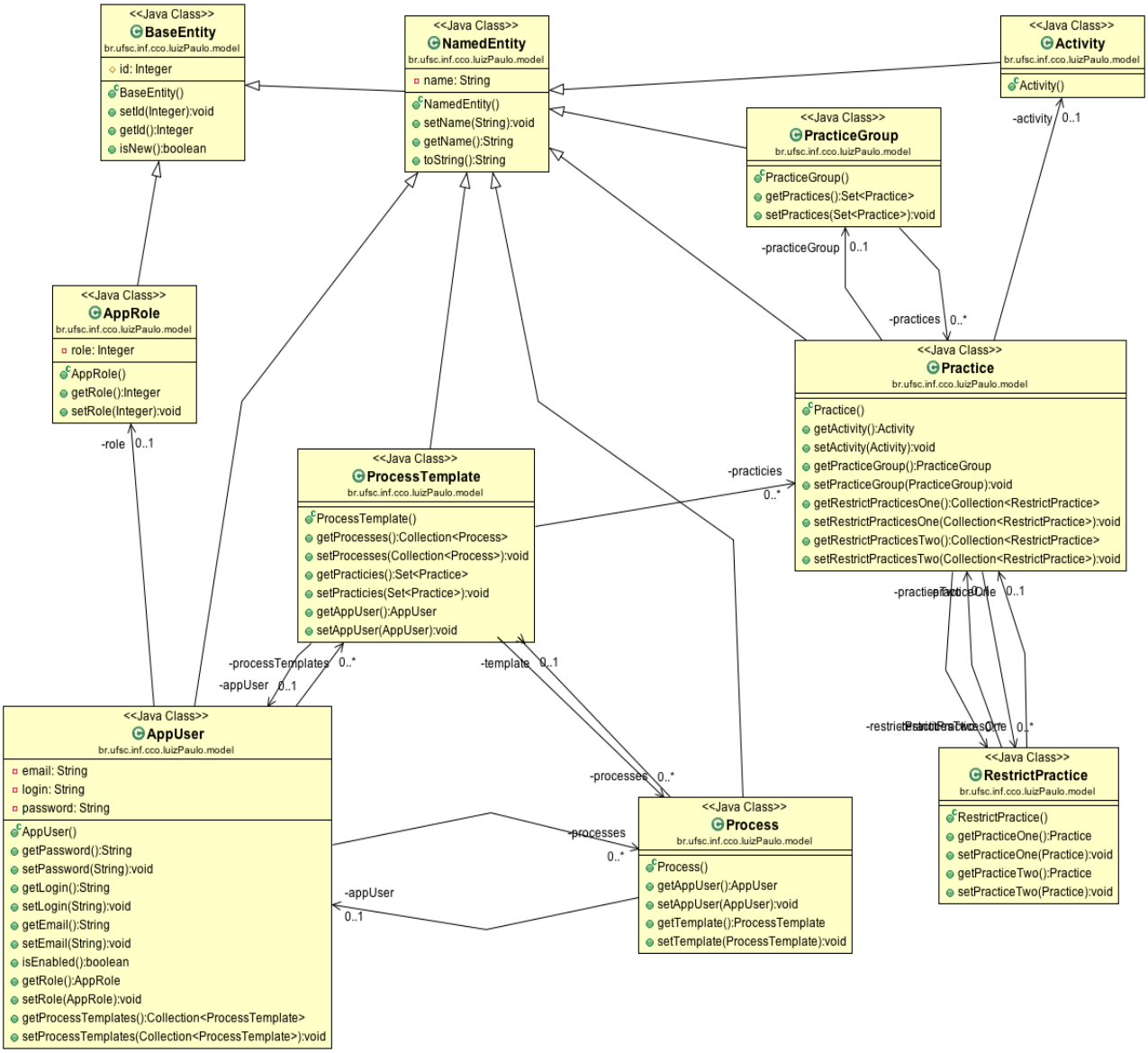

Figura 9 - Modelagem das classes de modelo da aplicação

5 FUNCIONAMENTO DA APLICAÇÃO

Neste capítulo será descrito o ambiente de desenvolvimento da ferramenta, suas configurações e aspectos técnicos, além de mostradas algumas imagens da aplicação em funcionamento.

Para testar a ferramenta, dois exemplos foram desenvolvidos. No primeiro foram definidas as atividades e práticas da atual versão do framework de práticas ágeis [SCHOEPPING 2012] e definidos processos ágeis a partir do framework [APÊNDICE 1]. No segundo exemplo, foram definidas as atividades e práticas do framework para linha de produtos de software [SOUZA 2013], e definidos processos utilizando este framework [APÊNDICE 2].

5.1.1 AMBIENTE DE DESENVOLVIMENTO

Nesta seção do trabalho será descrito o ambiente desenvolvimento da ferramenta, descrevendo o passo a passo para montagem de ambiente de desenvolvimento, organização de classes do projeto e montagem de artefato para deploy, visando facilitar futuros interessados em ampliar esta ferramenta.

5.1.1.1 CONFIGURAÇÃO DO AMBIENTE DE DESENVOLVIMENTO

Como descrito na seção 4.2.2, utilizou-se a IDE de desenvolvimento Eclipse, juntamente com o software de controle de versão git e o software Maven para gestão de dependências, afim de facilitar na montagem inicial do ambiente.

Para iniciar, utilizando o comando “git clone” pode-se obter o código fonte da ferramenta, disponível em [11]. Após isto, deve-se importar o projeto baixado como um projeto Maven dentro do Eclipse.

O Maven utiliza um arquivo chamado pom.xml na raiz do sistema para descrever dependências, plug-ins utilizados e artefatos a serem gerados. Utilizou-se o plug-in tomcat do maven para execução do projeto dentro do servidor web Apache Tomcat. Durante o desenvolvimento pode-se utilizar o comando “mvn tomcat:run” para subir a aplicação.

5.1.1.2 CONFIGURAÇÃO DO PROJETO

O projeto utiliza o framework de desenvolvimento web Spring MVC. Para tanto devem ser respeitadas as convenções de configuração impostas pelo mesmo, descritas em um arquivo web.xml, disponível em ”/src/main/webapp/WEB-INF/”, que muitas vezes torna-se extenso e complexo, por misturar diversos tipos de configuração necessárias. Com o intuito de minimizar esta complexidade, criou-se a estrutura de arquivos de configuração mostrada na Figura 10.

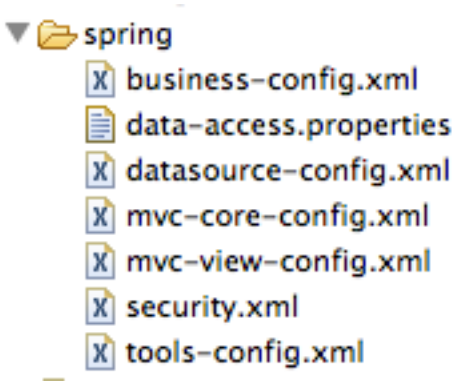

Figura 10 – Configurações do framework Spring MVC

A estrutura foi criada organizando os arquivos de acordo com a finalidade das configurações, sendo descrita da seguinte maneira:

- business-config.xml: Configurações da camada de negócio, tais como pacotes onde se encontram as classes de serviço, além do tipo de persistência utilizado pelo sistema;
- data-access.properties: Arquivo que isola as propriedades utilizadas para acesso a base de dados, tais como endereço, login e senha;
- datasource-config.xml: Arquivo contendo as definições de acesso à base de dados, utilizando as configurações do arquivo data-access.properties;
- mvc-core-config.xml: Contém as principais configurações do Spring MVC, tais como caminho para os controladores da aplicação e arquivos css e javascript utilizados;
- mvc-view-config.xml: Destina-se à configuração da camada de visão, incluindo caminhos para os arquivos desta camada;
- security.xml: Configurações referentes à autenticação e autorização de visualização de conteúdo da aplicação, necessárias devido a restrições de ações no sistema segundo o papel do usuário em questão;
- tools-config.xml: Guarda as configurações de ferramentas utilizadas pelo framework, como cache de acesso a dados dentre outras.

5.1.1.3 ORGANIZAÇÃO DOS PACOTES

Os pacotes de classes da aplicação foram organizados como demonstrado na Figura 11.

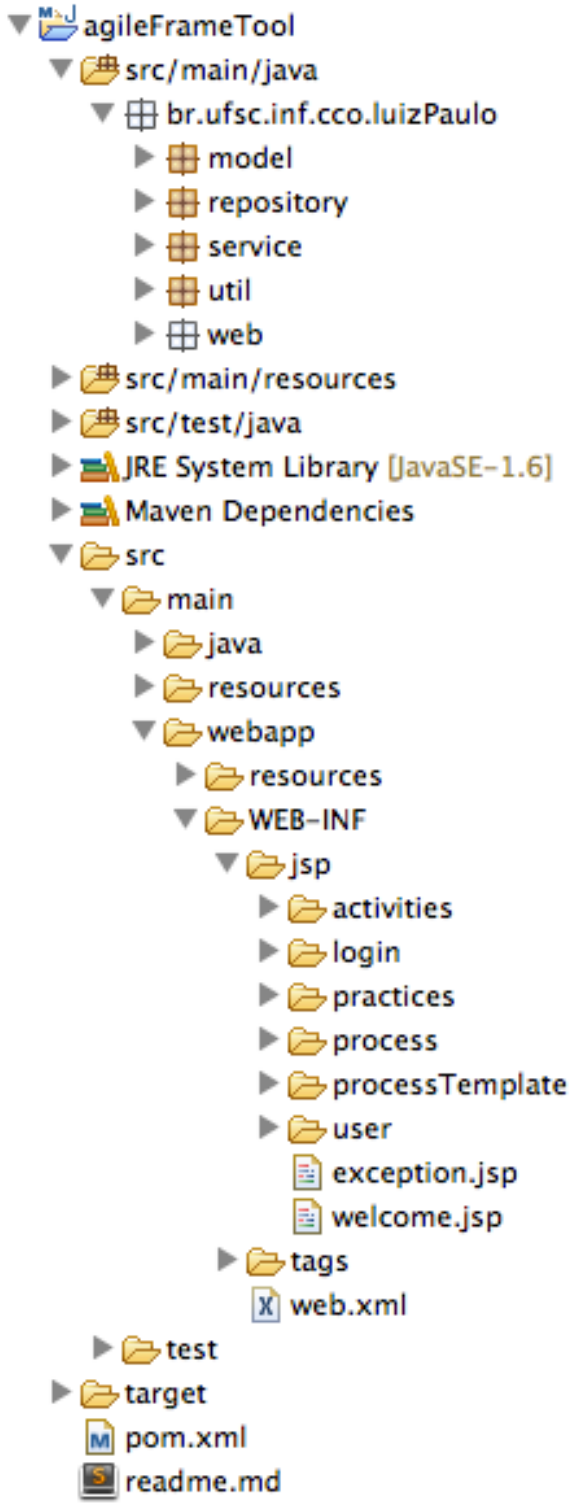

Figura 11 - Disposição de pacotes e arquivos da aplicação

Como evidenciado acima, os pacotes foram organizados de acordo com a responsabilidade das classes do sistema, podendo ser descritos da seguinte forma:

- Model: contém os modelos do sistema;
- Repository: contém as interfaces e implementações dos repositórios de arquivos na base de dados;
- Service: contém arquivos da camada de negócio da aplicação;
- Util: classes utilitárias da aplicação;
- Web: contém as classes utilizadas na iteração do usuário com a camada de visão do sistema
    - Controller: contém os controladores da aplicação;
    - Editor: contém classes auxiliares para associação de diferentes modelos em seus respectivos formulários de criação e edição;
    - Validators: mantém as classes responsáveis pela validação efetuada nos modelos durante a criação e edição dos mesmos.

Na Figura 11 pode-se também verificar a organização de arquivos da camada de visão da aplicação, que ficam localizados na pasta webapp expandida na figura acima. Os arquivos da camada de visão estão organizados de acordo com o modelo ao qual pertencem, além dos arquivos auxiliares contendo javascripts, folhas de estilo e fontes utilizadas pela aplicação.

5.1.2 SCREENSHOTS DA APLICAÇÃO

Nesta seção estão dispostas algumas imagens das principais telas da ferramenta desenvolvida, mostrando suas principais funcionalidades.

A Figura 12 mostra a tela inicial do sistema, na qual é solicitada a identificação de um usuário já existente, ou exibe o caminho para a criação de um novo usuário através do botão “Create an account”.

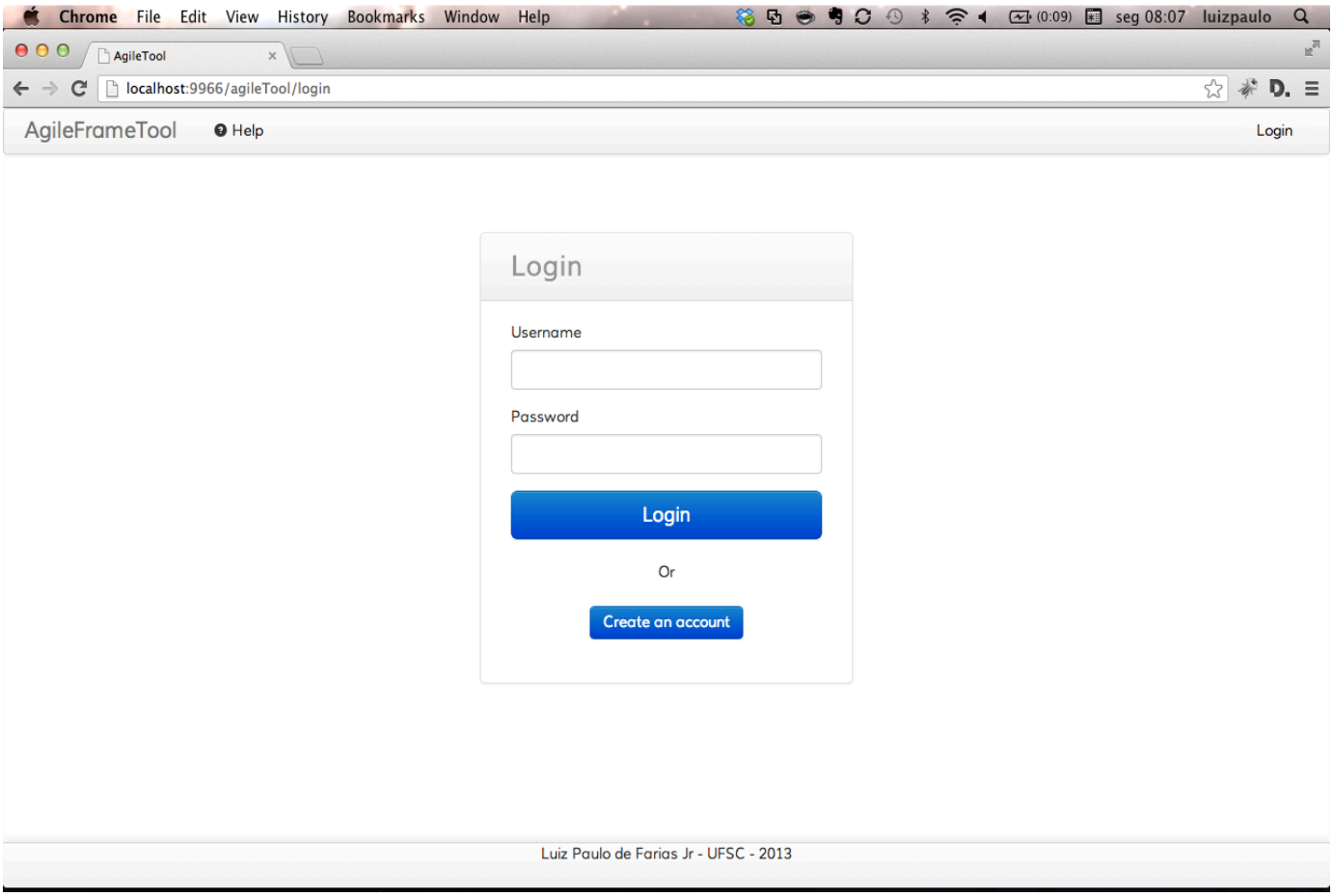

Figura 12 - Tela de login da aplicação

A Figura 13 mostra a tela de listagem de atividades, contendo um campo de busca das atividades e um link para acessar a edição de cada uma das atividades. Na parte inferior existe um botão para a tela de criação de nova atividade.

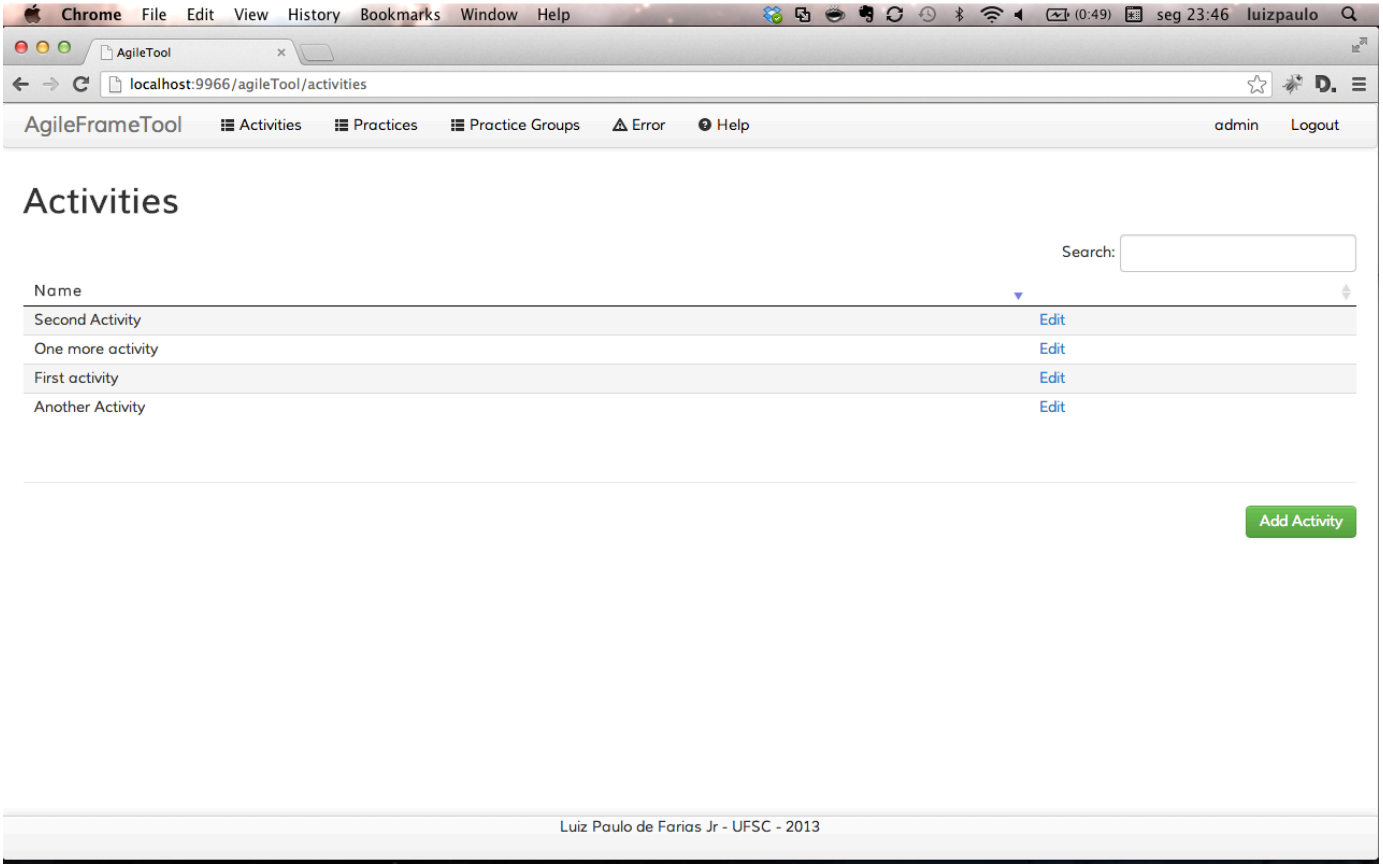

Figura 13 – Tela de listagem de atividades

A Figura 14 mostra a tela de criação de prática. Existe um campo para o nome da prática, um seletor para escolha de atividade e um seletor para adicionar práticas proibitivas entre sí.

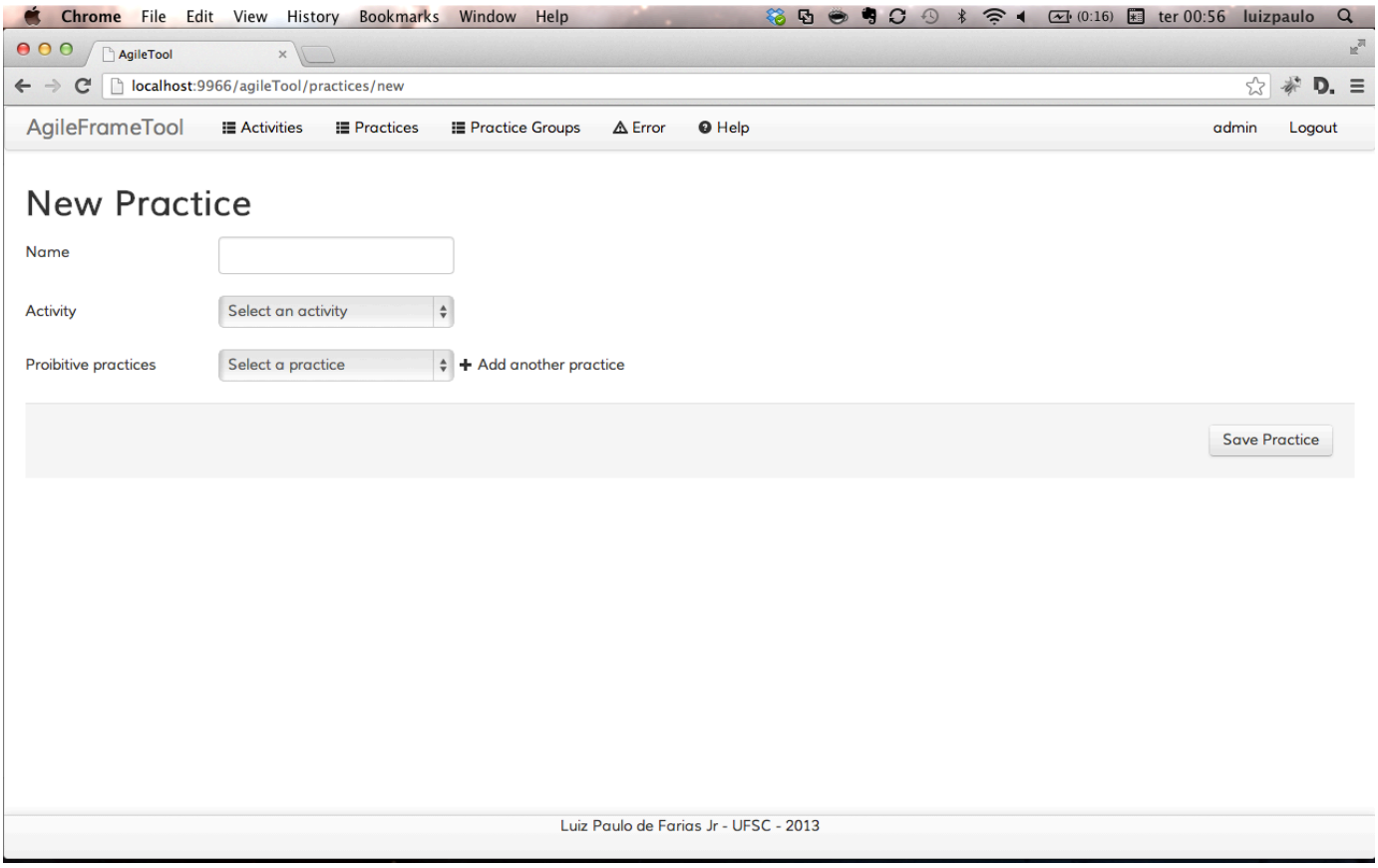

Figura 14 – Tela de criação de prática

A Figura 15 mostra a tela de criação de um *template* de processo. Para isso, o usuário deve informar o nome do *template* e adicionar as múltiplas práticas.

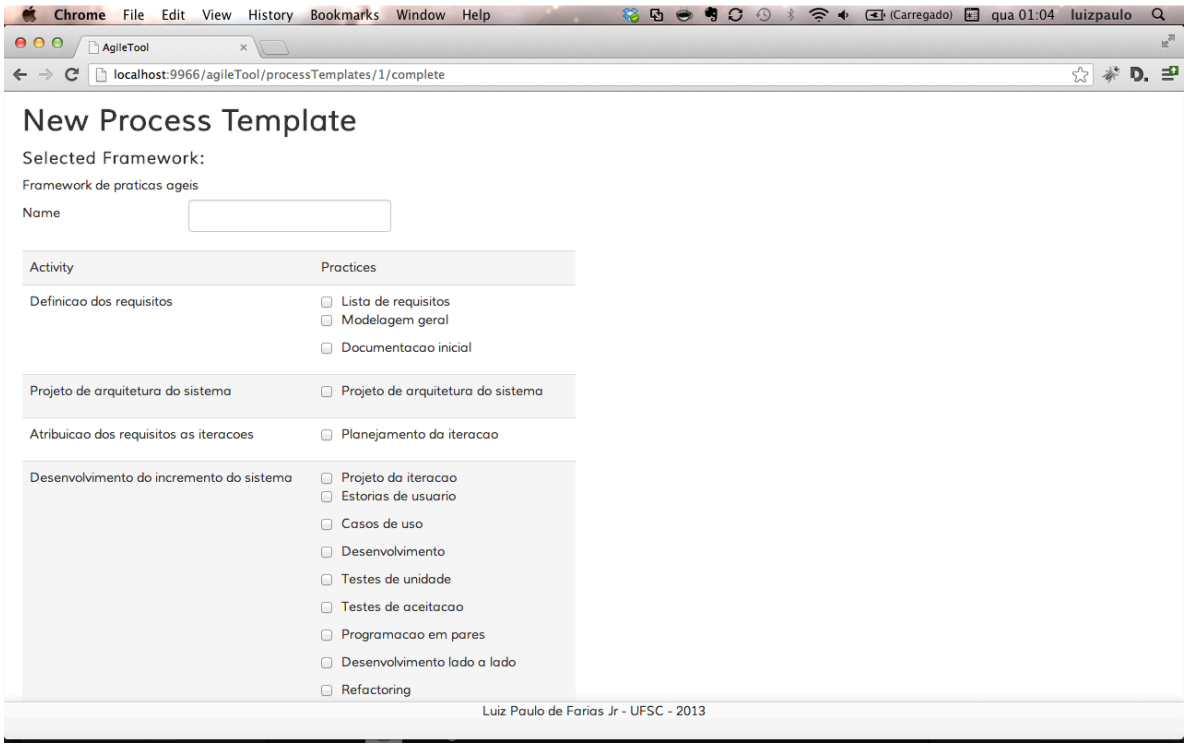

Figura 15 – Tela de criação de template processo

A Figura 16 mostra a tela de criação de processo. Existe um campo para o nome do processo, um seletor opcional para a escolha de um template previamente definido e um seletor para adicionar práticas.

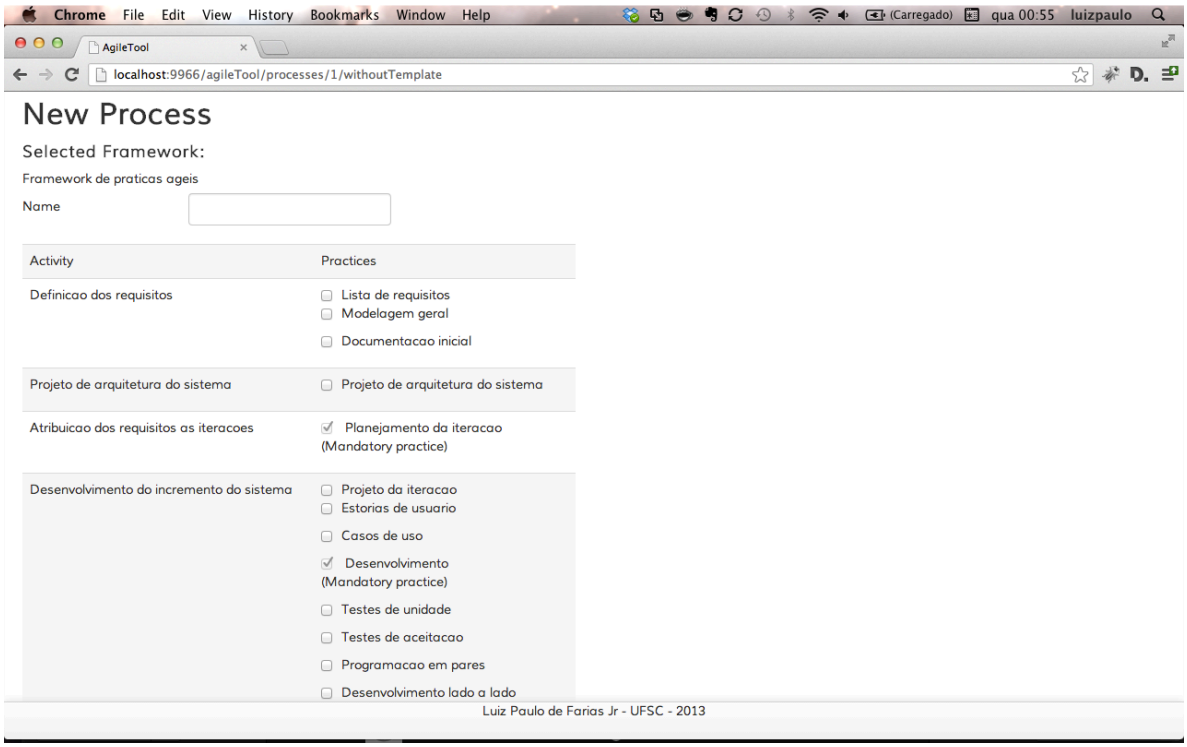

Figura 16 – Tela de criação de processo

A Figura 17 mostra a tela de criação de processo durante o momento de sugestão de práticas ao processo. A ferramenta verifica, de acordo com as práticas selecionadas pelo usuário, e traz novas práticas como sugestões abaixo.

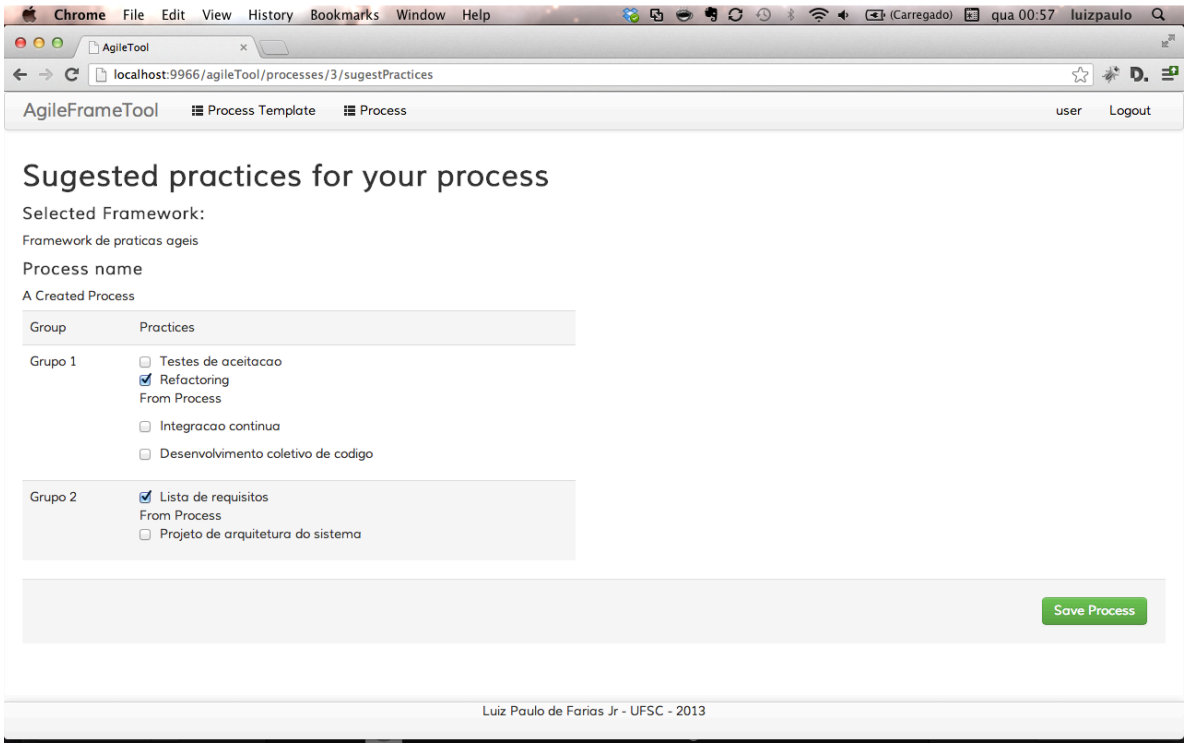

Figura 17 - Tela de sugestão de práticas ao processo

6 CONCLUSÃO

Métodos ágeis são cada vez mais adotados por empresas atualmente, porém o que se verifica é que usualmente as práticas definidas por um método ágil não atendem totalmente as demandas internas, bem como algumas práticas propostas acabam sendo descartadas. Desta forma, equipes acabam por definir seus próprios processos de desenvolvimento.

O framework proposto por [FAGUNDES 2005], estendido em [VILAIN 2007] e o posterior estudo sobre como aumentar a efetividade de um processo feito por [SCHOEPPING 2012] conseguem facilitar este processo, contudo acabam se tornando uma tarefa onerosa sem a utilização de uma ferramenta por haver uma grande gama de dados e se tratar de uma sequência complexa de passos para obter um processo ágil ideal.

Esta ferramenta web foi desenvolvida utilizando diversas práticas do framework, bem como estórias de usuário, desenvolvimento iterativo, dentre outras, mostrando a eficiência do framework proposto. Além disto, consegue auxiliar de forma eficaz esta tarefa, estando inclusive apta a absorver a constante expansão de métodos e práticas ágeis disponíveis no mercado.

A fim de testar a ferramenta, foram feitos dois exemplos. O primeiro consistiu em definir as atividades e práticas da atual versão do framework de práticas ágeis [SCHOEPPING 2012] e definidos processos ágeis a partir do framework [APÊNDICE 1]. No segundo exemplo, foram definidas as atividades e práticas do framework para linha de produtos de software [SOUZA 2013], e definidos processos utilizando este framework [APÊNDICE 2].

Visto isto, pode-se afirmar que a flexibilidade requerida para a ferramenta foi atingida, ambos os exemplos foram implementados com sucesso.

6.1 TRABALHOS FUTUROS

Uma possível expansão para a ferramenta é a inserção do conceito de gerência de artefatos relativos aos processos desenvolvidos pelo usuário, tais como, descrições de estórias, modelos produzidos, histórico de iterações, além da importação de artefatos de outros ambientes.

Outro aspecto que pode ser relevante a novos trabalhos é o aumento da colaboração entre usuários durante o desenvolvimento de um processo, com auxílio de ferramentas que possibilitem a criação e edição dos métodos de forma mais colaborativa entre usuários, afim de aumentar sempre a efetividade dos processos definidos.

Outra extensão também viável é a criação de um mecanismo de coleta de métricas sobre efetividade dos processos criados pelos usuários e retroalimentar o mecanismo de sugestão de práticas criado.

7 REFERÊNCIAS BIBLIOGRÁFICAS

[1] FAGUNDES, Priscila B., "*Framework for Comparing and Analyzing Agile Methods*." Master Thesis, UFSC, 2005.

[2] MACHADO, Thiago L., "Uma Ferramenta de Suporte ao Framework para Comparação e Análise de Métodos Ágeis." UFSC, 2005.

[3] BECK, Kent et al. *"Agile Manifesto"*. http://www.agilemanifesto.org, 2001.

[4] VILAIN, Patrícia; FAGUNDES, Priscila B.; MACHADO, Thiago L. "A Framework for Selecting Agile Practices and Defining Agile Software Processes." SEKE, 2007.

[5] SCHOEPPING, Guilherme; VILAIN, Patrícia. "Analisando a agilidade em processos ágeis." SBSI, 2011.

[6] B. Boehm. *Balancing Agility and Discipline: A Guide for the Perplexed*. 2 ed. Boston,MA: Addison-Wesley, 2004.

[7] http://www.agilealliance.org/

[8] Highsmith, Jim. "Agile Software Development" Ecosystems.Addison Wesley, 2002.

[9] http://www.progyan.com/agile.html

[10] SCHOEPPING, Guilherme. "Um Estudo Exploratório a Partir De Um Framework Para Seleção de Práticas Ágeis." UFSC, 2012

[11] https://github.com/luizpaulo/agileFrameTool

[12] SOUZA, Diego S., VILAIN, P. Selecting Agile Practices for Developing Software Product Lines. SEKE 2013.

APÊNDICE 1

Criação de um processo segundo o Framework de práticas ágeis [VILAIN 2007].

Para demonstrar o funcionamento da aplicação foram efetuados dois exemplos, o primeiro mostra a definição de um processo segundo o Framework de práticas ágeis[VILAIN 2007] com a adição da sugestão de grupos de práticas mais ágeis definido em [SCHOEPPING 2012].

Antes do projetista iniciar o trabalho, o sistema deve estar alimentado com as atividades, práticas e grupos de sugestão segundo o Framework de práticas ágeis. Os arquivos para criação destes itens já se encontram inclusos na ferramenta, poupando assim o trabalho inicial do administrador, porém deixando este apto a efetuar qualquer mudança necessária nestes itens. Nas figuras 18 a 21 subsequentes ficam evidenciados os itens atuals do Framework inseridos na ferramenta.

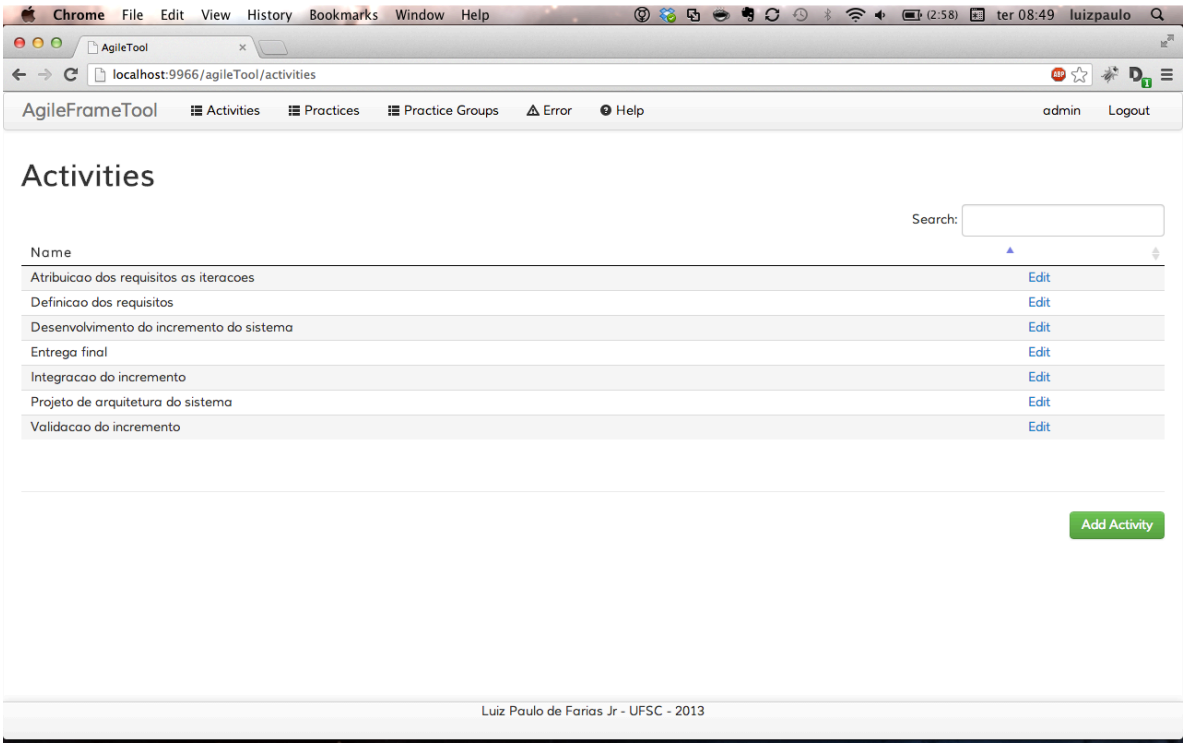

Figura 18 - Lista de atividades inseridas

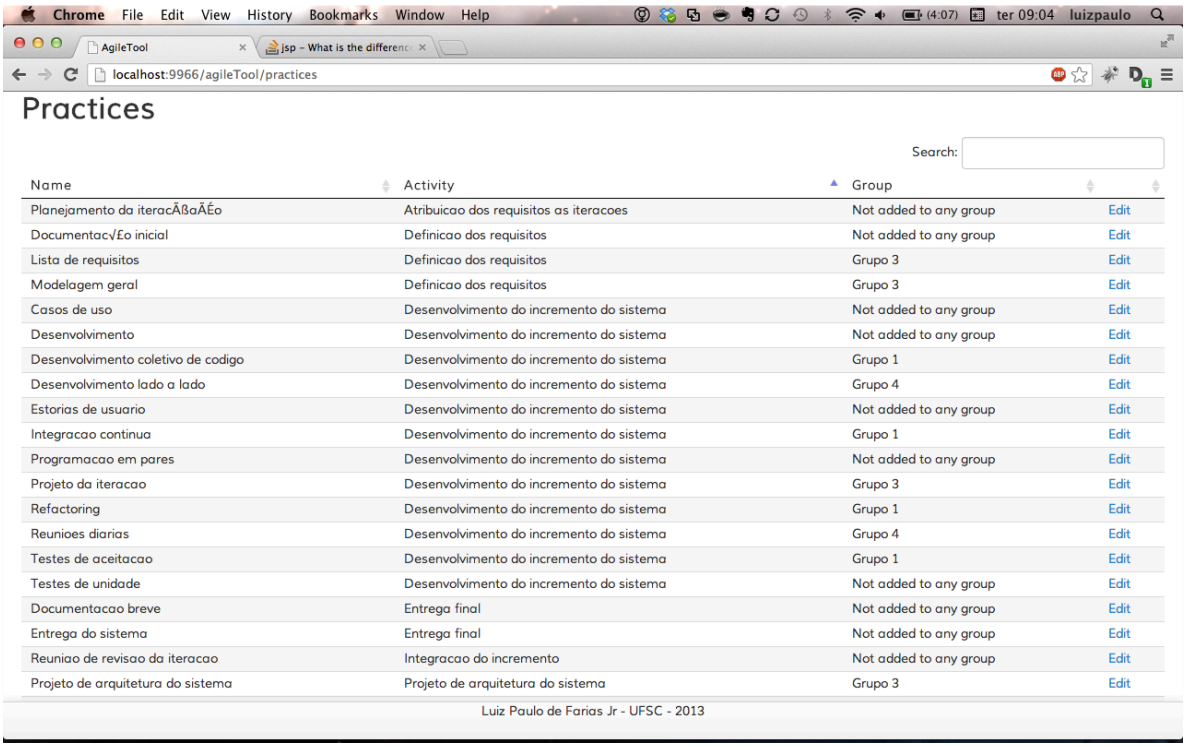

Figura 19 - Lista de práticas inseridas, com atividades e grupos definidos

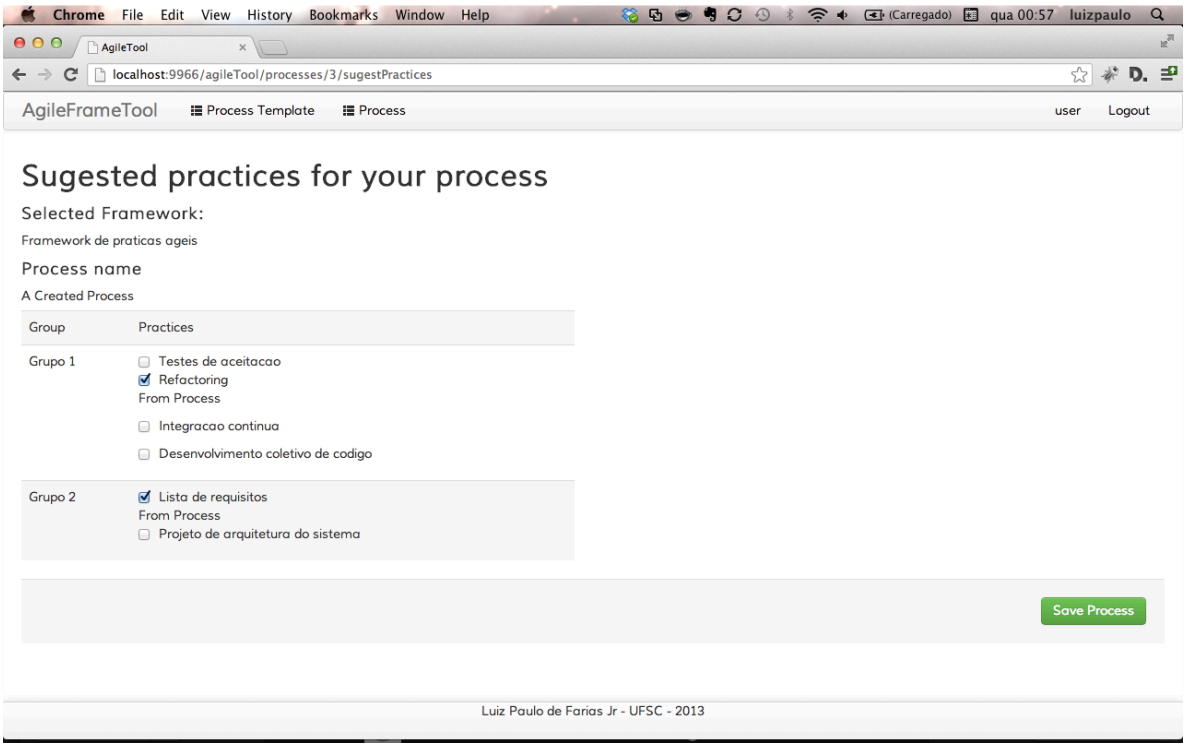

Figura 20 - Tela de sugestão de práticas segundo grupos

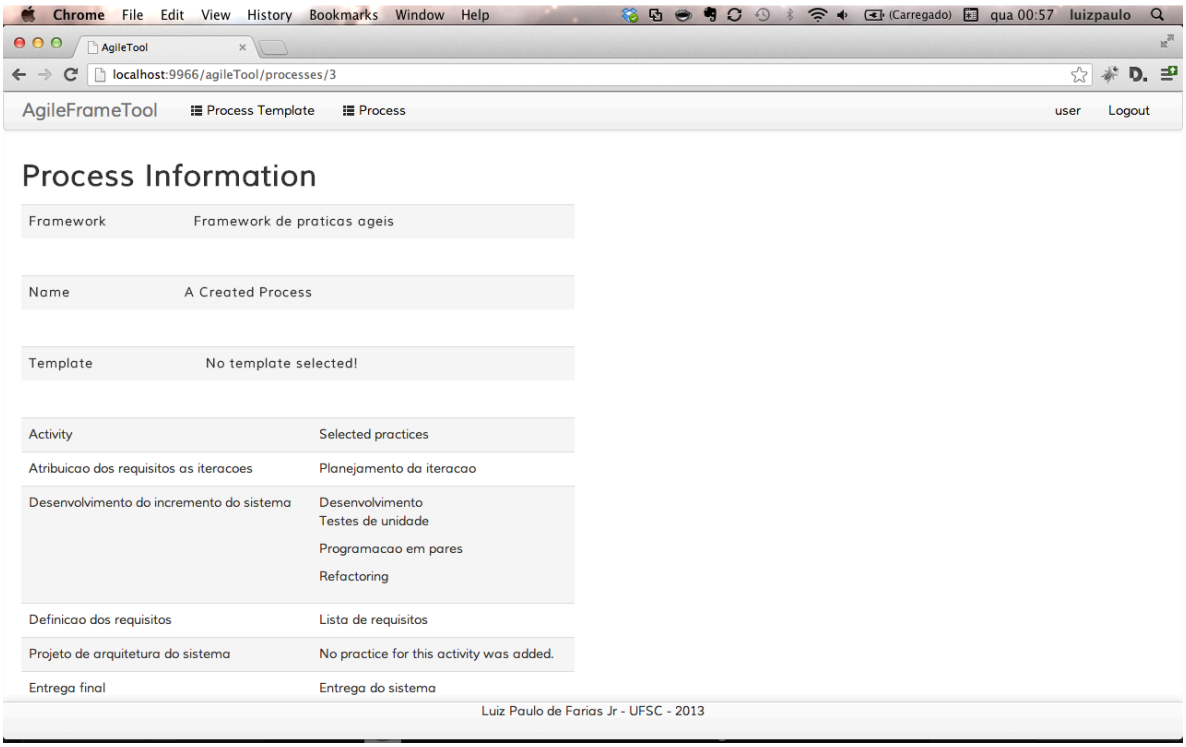

Figura 21 - Um processo definido

APÊNDICE 2

No apêndice 2, está demonstrada a utilização da ferramenta com o framework de para linha de produtos de software [SOUZA 2013], suas principais diferenças para o framework de práticas ágeis é a existência de dois ciclos distintos dentro do framework, além de uma maior quantidade de práticas obrigatórias e ainda não possuir um estudo para recomendação de práticas concorrentes durante sua utilização. O fluxo para o projetista é igual ao anteriormente descrito, porém utilizando-se dos dados específicos deste framework. Nas figuras abaixo estão evidenciadas estas diferenças.

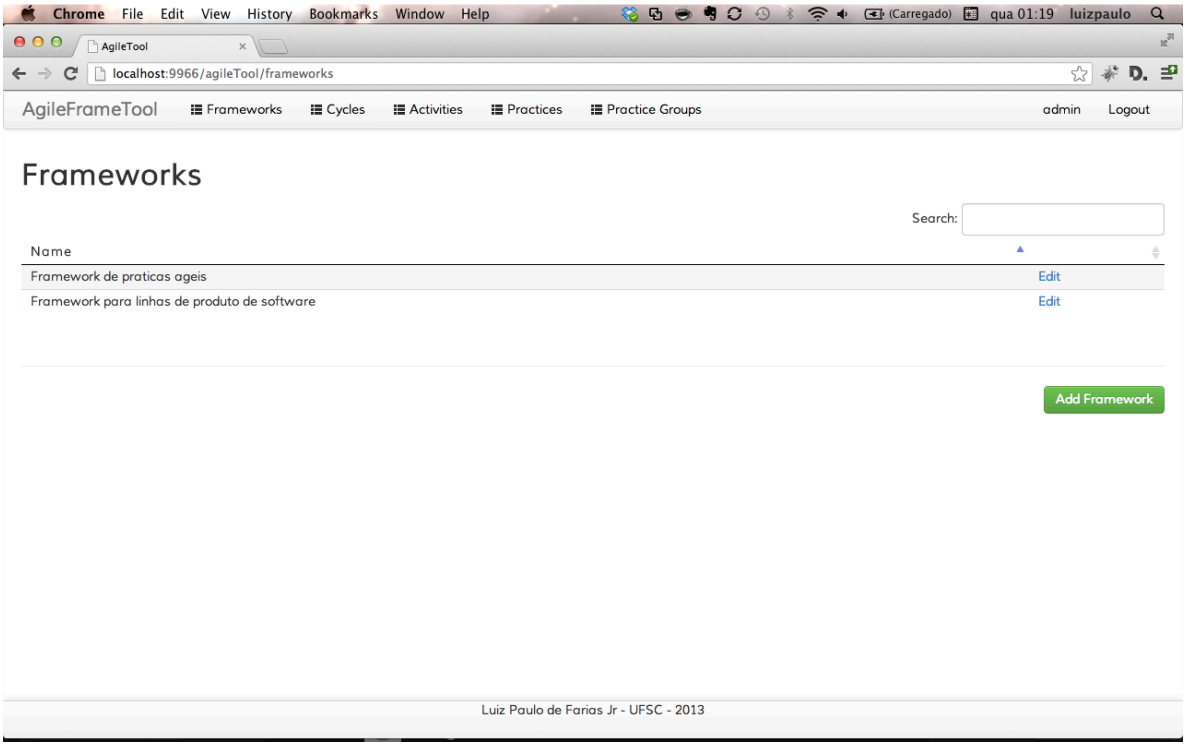

Figura 22 - Multiplos frameworks inseridos no sistema

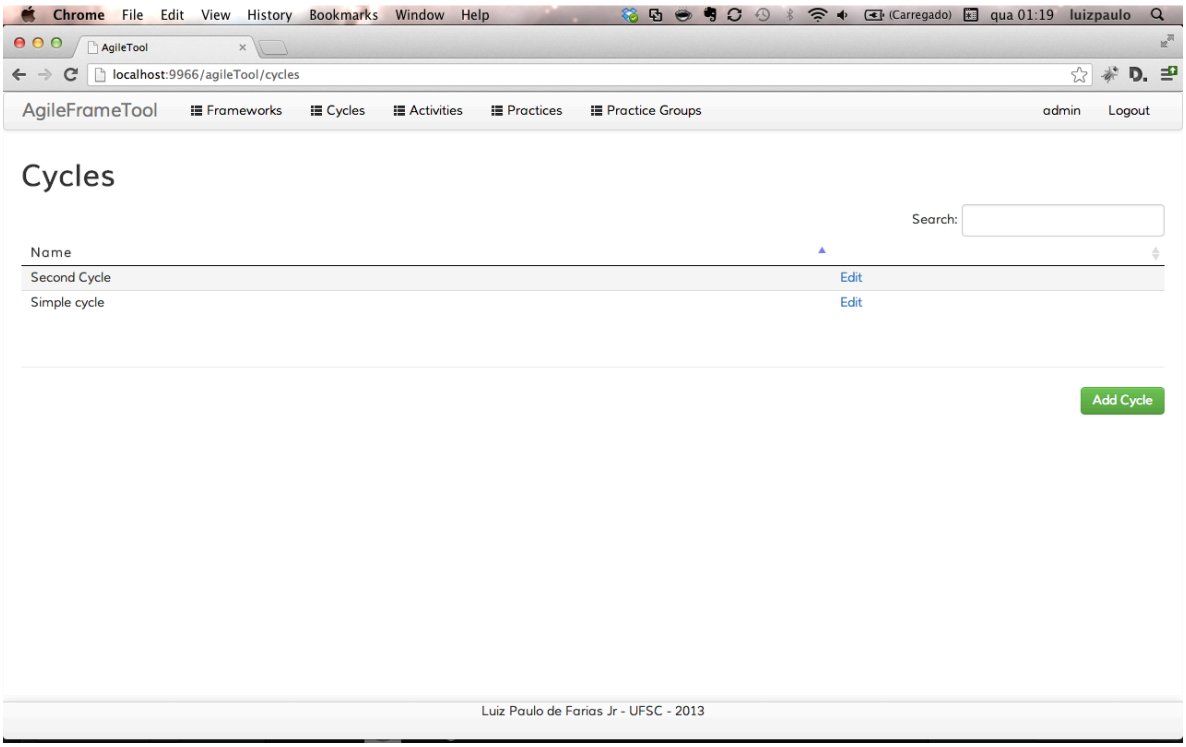

Figura 23 - Listagem de ciclos para o framework de linhas de produto de software