Gerenciamento Ágil de Projetos de Desenvolvimento de Softwares na Secretaria de Estado da Casa Civil

Claryanne A. Guimarães[1], Daniel Dias S. Rosa[1]

[1]Departamento de Computação – Universidade Federal de Sergipe (UFS)
Aracaju – SE – Brasil

claryanne@gmail.com, danieldiasba@yahoo.com.br,

Abstract. *This article presents the Agile managing of projects in the field of developing software, having the aim to provide a better view of the projects to the managers so that the developing of the software is better done. Some practices to the developing and managing of the Extreme Programming (XP) do Scrum were presented. With this practices we were able to analyse the developing of the software from Secretaria de Estado da Casa Civil de Sergipe (SECC), that from the results of this a new pattern for the agile managing of software developing was proposed.*

Resumo. *Este artigo apresenta o Gerenciamento Ágil de projetos no desenvolvimento de software como meio para os Gestores possuírem uma visão melhor dos projetos, tendo a oportunidade de gerenciar e controlar, tornando o desenvolvimento de software mais eficiente. Foram apresentadas algumas práticas para o desenvolvimento e gerência do Extreme Programming e do Scrum. Através dessas práticas permitiu nós avaliarmos o desenvolvimento de software da Secretaria de Estado da Casa Civil de Sergipe (SECC), que a partir dos resultados desta avaliação foi proposto um modelo para o gerenciamento ágil no desenvolvimento de software da SECC.*

1. Introdução

A Gerência de Projetos consiste de um conjunto de técnicas e habilidades na realização de atividades interligadas para alcançar um conjunto de objetivos pré-determinados.

Atualmente, existem duas abordagens para o gerenciamento de projetos: a do tipo "Clássico", na qual o projeto segue uma seqüência de etapas a serem concluídas; e a do tipo "Ágil" [Ferreira 2006] [Franco 2007] [Leal 2007] [Mattei 2007] [Filho 2007] onde o projeto é visto como um conjunto de pequenas tarefas, ao invés de um processo completo.

A abordagem do tipo "Ágil" será adotada na realização do objetivo deste trabalho por se considerar que seja mais adequada ao tamanho da equipe e à grande demanda uma vez que lida com projetos mais orientados a pessoas, adaptados às necessidades dos usuários e ao cumprimento dos prazos e custos, e apresenta simplicidade e transparência em seu desenvolvimento.

O objetivo principal deste artigo é realizar a adaptação de uma prática do gerenciamento ágil de projetos aplicado ao desenvolvimento de software na Secretaria de Estado da Casa Civil de Sergipe (SECC). O objetivo específico é descrever o

funcionamento do gerenciamento ágil de projetos aplicado ao desenvolvimento de software na SECC, proposto através de um projeto novo da SECC.

A justificativa para realização deste artigo é que a SECC não adota qualquer tipo de gerência, ainda que a demanda de projetos seja alta. Essa ausência afeta o tempo para entrega, a qualidade do produto e o relacionamento com o cliente. Projetos de desenvolvimento de software estão inseridos em ambientes de negócio bastante dinâmicos, sujeitos a mudanças constantes, no entanto necessita de uma gerência para poder controlar essas mudanças. Uma empresa que desenvolve projetos de softwares sem gerenciamento fica praticamente impossibilitada de atingir seus objetivos de negócio.

2. Gerenciamento Ágil de Projetos

Os modelos e métodos ágeis não são estruturados em padrões, o que torna o processo mais flexível e adaptável a necessidades emergentes. São estruturados de modo a atender a natureza mutável e dinâmica do processo de concepção do sistema [Ferreira e Lima 2006]. Diferentemente dos métodos clássicos que possuem fases delimitadas e bem separadas, no desenvolvimento ágil, as fase de concepção e desenvolvimento interagem durante todo o projeto, possibilitando desse modo uma interação constante entre os clientes e os desenvolvedores.

Desta forma observa-se que os métodos ágeis se diferenciam em alguns pontos em relação aos modelos clássicos de desenvolvimento de software como [Dias 2005]:

- **Os Métodos Ágeis são adaptativos e não preditivos:** diferentemente dos enfoques clássicos que defendem o planejamento integral do escopo no início do projeto e um controle rígido de mudanças, os planos dos Métodos Ágeis são elaborados e adaptados ao longo do projeto, permitindo e, algumas vezes estimulando, a incorporação das mudanças requeridas pelo cliente;
- **Os Métodos Ágeis são orientados a pessoas e não a processos:** os processos clássicos apresentam desenvolvimento de software, em geral, a pretensão de funcionar independentemente de quem os executa. Já os Métodos Ágeis levam em consideração os indivíduos, sendo elaborados para auxiliá-las;

Segundo Ferreira e Lima (2006) a adaptação á métodos ágeis é realizada através de desenvolvimento iterativo e interativo. A idéia central é trabalhar com iterações curtas. Cada iteração entrega ao seu final um produto completo e pronto para ser usado, que contém a implementação de um novo subconjunto de características. O uso de iterações curtas permite aos usuários e clientes fazerem uma avaliação do sistema logo que uma versão inicial é colocada em produção. Neste momento, usuários, clientes e desenvolvedores decidem sobre quais características devem ser adicionadas, quais devem ser modificadas, e até, quais devem ser retiradas do sistema. O sistema é desenvolvido da forma mais iterativa possível.

Portanto, os métodos ágeis consistem em projetos bem mais orientados a pessoas e, adaptados às necessidades dos usuários e ao cumprimento dos prazos e custos, apresentando simplicidade e transparência em seu desenvolvimento, com entrega de pequenas versões, permitindo mudanças ao longo do projeto e, uma maior aproximação do cliente.

Atualmente existe vários Métodos Ágeis de Desenvolvimento de Software, vale a pena destacar: *Extreme Programming* (XP) e *Scrum*. A seguir serão apresentados a principais características de cada um desses métodos.

2.1. *SCRUM*

Scrum consiste numa abordagem empírica, aplicada em ambientes voláteis e, desenvolvida de forma flexível e capaz de responder às mudanças, no gerenciamento de processos de desenvolvimento de software, onde as técnicas ficam a escolha dos programadores.

A seguir são descritos o conteúdo do framework *Scrum* composto em práticas artefatos, papéis, fases.

Segundo Schwaber (2009), os papeis são descritos em três como o *Scrum Master* como um treinador da equipe, o *Product Owner*: representa o cliente ou o patrocinador do projeto, o Time a equipe de desenvolvimento.

O ciclo de vida no *Scrum* está dividido em três partes [Martins 2007]:

- **Pré-game:** inicia-se com a definição da lista do *backlog* de produto, através de uma reunião de planejamento. Após a definição do *backlog* do produto, organiza-se o *backlog* do *Sprint*. Nesta etapa serão definidos, quais atividades serão realizadas no Game.
- ***Game:*** é a fase de desenvolvimento. Diariamente, durante o *sprint*, realizam-se as reuniões Diárias, onde a equipe se reúne e discute sobre o andamento do projeto. Nela, os membros da equipe falam brevemente de como vai o seu trabalho, explicitando problemas encontrados e qual será sua próxima atividade.
- **Pós-Game:** para fechar o ciclo (pós-game) é apresentado ao cliente o produto da iteração, ou seja, o pedaço executável de software ou incremento é analisado pelos usuários. Após a demonstração da nova parte do software ao cliente, inicia-se novamente o ciclo do *Scrum*.

De acordo com Martins (2007, p. 276-290) e Schwaber (2009 p. 4-5) os principais conceitos do *Scrum* são o *Backlog* do produto *Sprint*, *Backlog* da *Sprint*, Gráfico de *Burndown* (Gráfico da *sprint*) e Reunião de Planejamento da *sprint.*

O *Scrum* é um framework que não necessita de um grande investimento financeiro, seus processos e práticas são simples e objetivos, as ferramentas necessárias são básicas, podendo utilizar um quadro, cartão, editor de texto e planilhas. A flexibilidade de iteração com outros framework e adaptação a necessidade da equipe de desenvolvimento, a organização utiliza as práticas que achar melhor.

2.2. *Extreme Programming* (XP):

Indiscutivelmente, o método ágil de maior expressão voltado para pequenas e médias equipes de desenvolvimento de software, característico por reunir um conjunto de práticas de desenvolvimento já existentes e reconhecidas, levando ao extremo, ao limite. O XP baseia-se em práticas ou regras descritas a seguir [Pressman 2006].

Segundo Abrahamsson (2002 p. 18-21) o ciclo de vida XP é bastante curto e, à primeira vista, difere dos padrões dos modelos de processo convencionais. Onde as fases são: exploração, planejamento, primeira iteração, produção, manutenção e morte.

De acordo com Beck (2001, p. 105-111), os papéis do XP apresentam quatro atividades básicas produção de código, testes, *listening* (escutar o cliente) e desenho, são sete papéis definidos, com uma equipe composta de 4 a 12 membros: Programador, Cliente, Testador, *Tracker,* Técnico (*Coach*), Consultor e *Big Boss.*

A seguir são listadas as práticas do XP conforme descrito por Beck (2001), como Estórias ou Metáforas, Pequenas versões, Projeto Simples, Desenvolvimento orientado a Testes, Refatoração,Integração contínua, Código coletivo,Cliente Presente, Código padronizado e Reunião diária em pé ou *stand up meeting*.

3. Proposta de Gerenciamento Ágil de Projetos de Desenvolvimento de Softwares na Secretaria de Estado da Casa Civil

Nesta seção é apresentado a Secretaria de Estado da Casa Civil (SECC) e o modelo de gerenciamento ágil proposto para o desenvolvimento de software da SECC.

3.1. Apresentação da SECC e Descrição do Ambiente Encontrado

A SECC ocupa quase todo o palácio, e está dividida em várias Coordenadorias, Diretorias, Setores e Órgãos (ANEXO B). Ela tem por finalidade prestar assessoramento direto ao Governador do Estado no desempenho de suas atribuições, especialmente na supervisão e execução das atividades administrativas da Governadoria, no que se refere à Secretaria Particular do Governador e à Assessoria Especial de Assuntos e do Cerimonial.

Uma das Coordenadorias que possui destaque na tomada de decisão e no planejamento estratégico é a Coordenadoria de Infra-Estrutura em Tecnologia da Informação - COINF é competente para: monitorar o desempenho dos recursos computacionais e telecomunicação.

A situação encontrada na gestão do governo no ano de 2006, a Secretaria de Estado da Casa Civil não era provida de um setor especializado em desenvolvimento de softwares para acelerar e controlar os processos administrativos existentes no órgão.

No governo atual, no poder desde janeiro de 2007, foi criado a Coordenadoria Especial de Desenvolvimento de Tecnologias da Informação (CEDTI) para fins de desenvolvimento de soluções, com objetivo de propiciar aceleração e controle dos processos administrativos, auxiliando nas atribuições do governador. Hoje a equipe é formada por cinco integrantes, que exercem a função de Programador de Sistemas e um deles com a função de coordenador.

Apesar da criação da CEDTI em 2007 para desenvolvimento de softwares, está se encontra bastante desorganizada devido a vários fatores como:

- Não existe um padrão ou modelo desenvolvimento de software para elaborar os produtos. A não ser se basear em produtos elaborados anteriormente;

- Análise de requisitos é feita na maioria das vezes somente no início do projeto. Fazendo com que na maioria das vezes o produto não atinja as expectativas dos usuários;
- Não existe qualquer tipo de gerência para analisar como está sendo desenvolvido o projeto ou o software. Por isso os produtos são sempre entregue com atraso e sem estar dentro da qualidade desejada pelos usuários;
- Não há uma elaboração de uma documentação, a não ser um Diagrama de Entidade e Relacionamento (DER), que serve para identificar tudo que um software possui. Dificultando muitas vezes a manutenção do software ou até mesmo o seu entendimento;
- Os usuários têm resistências aos softwares desenvolvidos. Fazendo com que muito do que se é desenvolvido não tenha utilidade.

3.2. Modelo Proposto

Como base na especificação e nos problemas do ambiente de desenvolvimento da SECC, e do estado da arte dos métodos ágeis, percebeu-se que cada método possui sua particularidade, portanto o modelo vai utilizar duas metodologias, a *Scrum* para o gerenciamento do Projeto e o XP para o desenvolvimento do Projeto, com isso a interação das metodologias nos trabalhos realizados diariamente dentro dos *sprints* do projeto conforme a Figura 3.

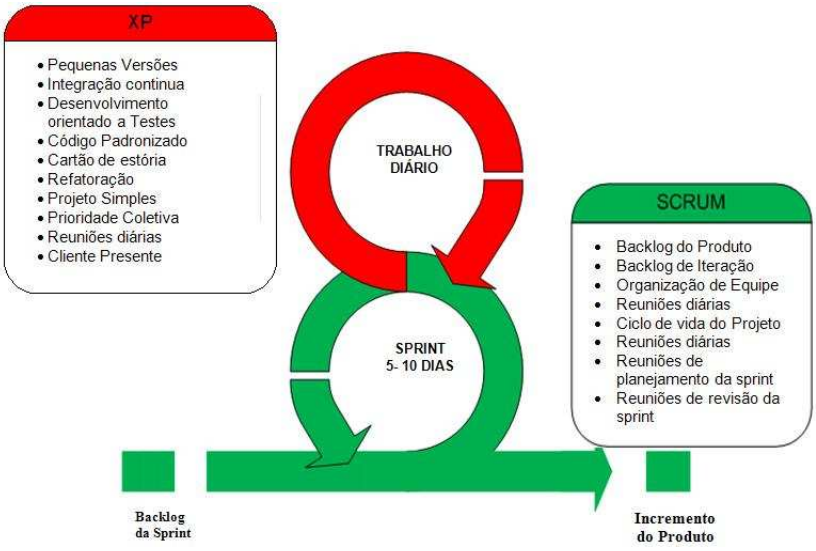

Figura 3: Interação entre o SCRUM e o XP no modelo

3.2.1. Papeis e Responsabilidades

No modelo proposto os papéis herdados são do *Scrum* onde trabalha com basicamente 3 (três) papéis principais bem definidos, a nomenclatura dos papéis foi alterada do Inglês para o Português para generalizar a proposta.

- **Líder do Time (*Scrum Master*):** Contato direto com o *Product Owner*, cuidar do time, garantir a comunicação; evitar impedimentos.

- **Gerente do Produto (*Product Owner*):** criar e compartilhar uma visão do projeto; tomar decisões sobre os itens *backlog* do Produto; indicar e priorizar itens de *backlog*; validar a entrega no final do sprint.
- **Time:** Estimar os itens do *backlog*; desenvolvimento do software.

3.2.2. Visão do Modelo Proposto

O modelo proposto tem como base o processo de gerenciamento de projetos especificado pelo *Scrum*, e o desenvolvimento do software pelo XP. A figura 4 identifica as entradas e saídas, suas relações e as fases do processo de desenvolvimento.

A nomenclatura utilizada no processo das fases de desenvolvimento do projeto é herdada do *Scrum*. O período sugerido para cada iteração é de 5 a 10 dias e durante a iteração deve ser definido e analisado a necessidade de cada cliente e o projeto.

Em cada iteração é percorrido o ciclo definido pela fase de pré-game, onde são detalhados os requisitos do próximo *sprint*, no desenvolvimento, onde os itens do *backlog* do *sprint* são transformados em incrementos do produto e o aprendizado adquirido é realimentado ao processo após a realização de reuniões de revisão do *sprint*.

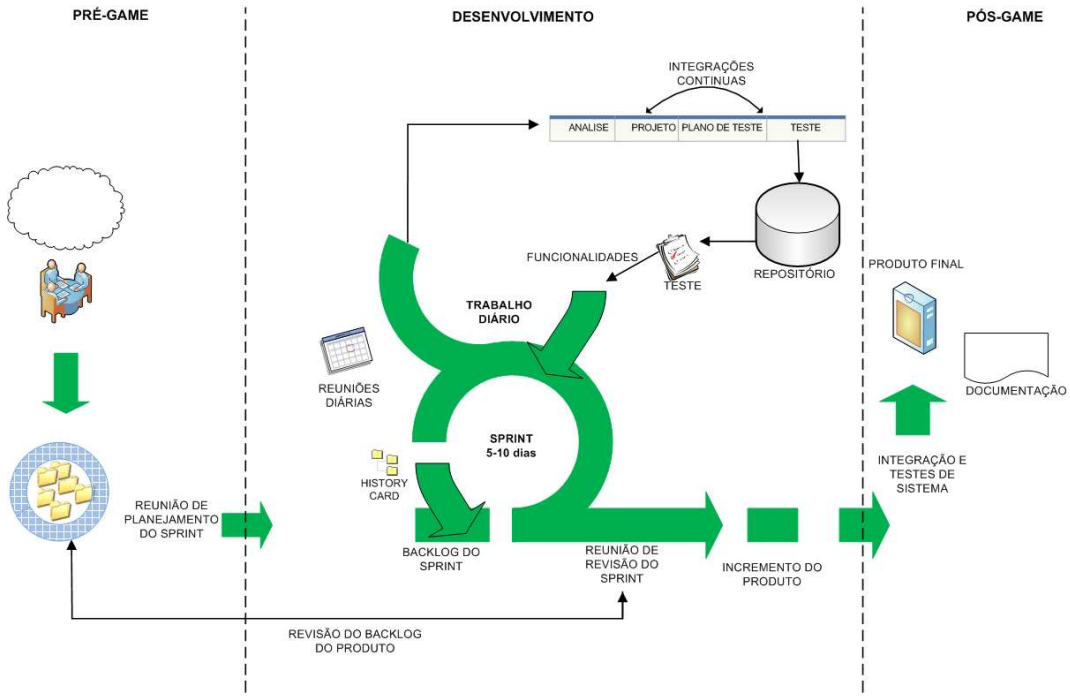

Figura 4: Visão Geral do Modelo Proposto

A seguir, são descritas cada uma das fases que compõem o modelo proposto, fornecendo uma visão do trabalho a ser executado em cada uma delas.

Pré-Game

Esta fase inicia-se com o levantamento de requisitos e a elaboração de uma lista das funcionalidades fornecidas pelo Gerente do Produto. Para isso foi utilizado a prática do XP de *user story card* (cartões de Estórias), onde o Gerente do Produto, com a ajuda dos interessados, descreve cada uma das funcionalidades do produto. A partir das tarefas originadas dos cartões é criado o *Backlog* do Produto. Nessa fase também são estabelecidos padrões, arquitetura, as tecnologias que serão utilizadas ao longo do projeto.

A fase do Pré-Game não é realizada apenas uma vez, antes do inicio de cada sprint, quando é reiniciado o ciclo de desenvolvimento, o time se reúne com o Gerente do Produto para planejar a próxima iteração. Nessa reunião de revisão do sprint, os requisitos do produto são revisados e uma visão para o próximo sprint é gerada; serve para elaborar uma nova expectativa ou até mesmo relembrar ao time qual propósito de cada um no projeto. Pode ser alterado, adicionado ou até mesmo excluindo algum item do *backlog* do produto.

Na reunião de planejamento do sprint é definido o escopo para ser trabalhado no próximo sprint, selecionado os itens do *backlog* do produto que serão incorporados à lista do *backlog* do *sprint*, que fica estável durante toda iteração.

Desenvolvimento

Na fase de desenvolvimento é onde a equipe executa o projeto, com entregas iterativas e incrementais de funcionalidade, permitindo a equipe gerar resultados verificáveis.

O desenvolvimento do incremento do produto é realizado através de algumas práticas e ferramentas do XP, que são utilizados nos trabalhos diários; as funcionalidades do produto são produzidas diariamente.

Ao final de cada sprint, os resultados obtidos são avaliados nas reuniões de revisão de sprint (nelas também estão inseridas as reuniões de retrospectiva da sprint), que o aprendizado obtido na reunião é aproveitado para reunião de planejamento do próximo sprint.

Nas reuniões de revisão de sprint também serão mostradas para o time o *burndown* da *sprint*, para que possa analisar de forma gráfica o andamento da *sprint*.

Pós-Game

O pós-game é quando acordado que o produto pode ser utilizado e não existe mais nenhuma funcionalidade na lista de *backlog* da sprint para ser implementada. Nessa fase é que são realizadas os testes do sistema e a documentação.

Quando todos os itens do *backlog* do produto são transformados em funcionalidades, o produto funcional é gerado e pronto para ser entregue ao cliente. Portanto o projeto é dado como encerrado, caso contrário todas as etapas (pré-game, desenvolvimento, pós-game) deverá ser realizadas novamente.

É importante incorporar o conhecimento desse projeto, como aprendizado em outros projetos.

Para documentação do sistema, utilizam-se os documentos disponíveis no XP e *Scrum* como os cartões de estórias, os testes de unidade e aceitação, o *backlog* do produto, *backlog* da *sprint*.

3.3. Aplicação do Modelo Proposto

Para a realização do modelo proposto, foi escolhido o projeto Suprimento de Fundos, que permitiu gerar resultados mais precisos sobre o modelo, devido ao seu maior grau de complexidade.

Iniciando o projeto de Suprimento de Fundos foi feita uma reunião inicial dos interessados com o coordenador de TI para ver a viabilidade de realizar o desenvolvimento deste projeto.

Após a reunião inicial foi gerado um documento com a definição do nome e uma visão geral do que é o produto. A partir daí deu-se início ao modelo proposto.

Iniciando o modelo foram definidos os papéis e responsabilidades de cada um no projeto: **Gerente do Produto**: Rafael Batista Santos; **Líder do time**: Daniel Dias Santa Rosa; **Time**: Alberto Santos Barbosa, Eliel da Silva Almeida Júnior e Rafael Ribeiro Deda.

Pré-Game – Iteração 1

Logo após a definição dos papéis e responsabilidades, foi realizada a reunião de planejamento da versão para entrega. Nesta etapa estabeleceu as características gerais e funcionalidades da versão. Para isso foi utilizado cartões de estórias, como mostra a figura 5.

User Story Card - SSF: Cadastrar Suprimento

Data: 04/12/2009	Tipo Atividade: Nova	Status: Concluído
Numero da US: 031	Tempo estimado: 11:30h	Tempo real: 17:30h
Prioridade: Alta		

Descrição: Página de cadastrar suprimento.
O cadastrar suprimento deve ter os seguintes campos: Data de Requisição (ao lado deste campo deve conter um botão que ao clicar deve aparecer um calendário), Projeto, Responsável, Base Legal, Unidade Orçamentária e Data de Certificado. Todos os campos serão obrigatórios, exceto Data de Certificado. O cadastro e alteração de elementos de um suprimento serão feitos dentro da própria página cadastrar suprimento. Antes de mostrar os campos do elemento será necessário clicar em um botão Adicionar, que automaticamente mostrará os campos do elemento. Também terão dois botões Inserir (ao clicar insere um suprimento) e Voltar (ao clicar retorna para página anterior).

Tarefas

Tarefa	Descrição	Tempo Estimado	Tempo de conclusão	Responsável	Status
031.01	Criar a página de cadastrar suprimento	2:30h	2:30h	Rafael	Concluído
031.02	Criar o consultar Inserir, alterar e excluir elemento na página de cadastrar suprimento.	9:00h	15:00h	Eliel	Concluído

Testes de Aceitação

1	Verificar se os campos obrigatórios de suprimento estão sendo validados	Concluído
2	Verificar se o botão Inserir está funcionando	Concluído
3	Verificar se o botão Voltar está funcionando	Concluído
4	Verificar se ao clicar nos campos de Data do suprimento está aparecendo o calendário	Concluído

Figura 5: Cartão de Estória de Cadastrar suprimento.

A partir das tarefas originadas dos cartões foi criado o Backlog do Produto, como mostra a figura 6

Backlog do Produto

Atualização 04/12/09

Iteração	Id	Item	Prioridade	Responsável	Horas estimadas	Situação
1	001.01	Criar o DER do SSF	Alta	Eliel	3	Não Iniciado
1	002.01	Criar o script do banco SSF	Alta	Alberto	3	Não Iniciado
1	003.01	Criar a classe Bean de Responsável	Alta	Alberto	2	Não Iniciado
1	003.02	Criar a classe Objeto de Responsável	Alta	Alberto	1,5	Não Iniciado
1	003.03	Criar a classe DAO de Responsável	Alta	Alberto	1,75	Não Iniciado
	004.01	Criar a página de consultar responsável	Média			Não Iniciado
	005.01	Criar a página de cadastrar responsável	Média			Não Iniciado
	006.01	Criar a página de alterar responsável	Média			Não Iniciado
	007.01	Criar a página de excluir responsável	Média			Não Iniciado
1	008.01	Criar a classe Bean de Base Legal	Alta	Eliel	0,75	Não Iniciado
1	008.02	Criar a classe Objeto de Base Legal	Alta	Eliel	0,5	Não Iniciado
1	008.03	Criar a classe DAO de Base Legal	Alta	Eliel	0,5	Não Iniciado
	009.01	Criar a página de consultar base legal	Média			Não Iniciado
	010.01	Criar a página de cadastrar base legal	Média			Não Iniciado
	011.01	Criar a página de alterar base legal	Média			Não Iniciado
	012.01	Criar a página de excluir base legal	Média			Não Iniciado
1	013.01	Criar a classe Bean de Projeto	Alta	Rafael	0,75	Não Iniciado
1	013.02	Criar a classe Objeto de Projeto	Alta	Rafael	0,5	Não Iniciado
1	013.03	Criar a classe DAO de Projeto	Alta	Rafael	0,5	Não Iniciado
	014.01	Criar a página de consultar Projeto	Média			Não Iniciado
	015.01	Criar a página de cadastrar projeto	Média			Não Iniciado
	016.01	Criar a página de alterar projeto	Média			Não Iniciado
	017.01	Criar a página de excluir projeto	Média			Não Iniciado
1	018.01	Criar a classe Bean de Unidade Orçamentária	Alta	Alberto	0,75	Não Iniciado
1	018.02	Criar a classe Objeto de Unidade Orçamentária	Alta	Alberto	0,5	Não Iniciado
1	018.03	Criar a classe DAO de Unidade Orçamentária	Alta	Alberto	0,5	Não Iniciado
	019.01	Criar a página de consultar unidade orçamentária	Baixa			Não Iniciado
	020.01	Criar a página de cadastrar unidade orçamentária	Baixa			Não Iniciado
	021.01	Criar a página de alterar unidade orçamentária	Baixa			Não Iniciado
	022.01	Criar a página de excluir unidade orçamentária	Baixa			Não Iniciado
1	023.01	Criar a classe Bean de Elemento de Despesa	Alta	Eliel	0,75	Não Iniciado
1	023.02	Criar a classe Objeto de Elemento de Despesa	Alta	Eliel	0,5	Não Iniciado
1	023.03	Criar a classe DAO de Elemento de Despesa	Alta	Eliel	0,5	Não Iniciado
	024.01	Criar a página de consultar elemento de despesa	Média			Não Iniciado
	025.01	Criar a página de cadastrar Elemento de Despesa	Média			Não Iniciado
	026.01	Criar a página de alterar Elemento de Despesa	Média			Não Iniciado
	027.01	Criar a página de excluir Elemento de Despesa	Média			Não Iniciado
1	028.01	Criar a classe Bean de Suprimento	Alta	Rafael	3	Não Iniciado
1	028.02	Criar a classe Objeto de Suprimento	Alta	Rafael	1,75	Não Iniciado
1	028.03	Criar a classe DAO de Suprimento	Alta	Rafael	1,5	Não Iniciado
1	029.01	Criar a classe Bean de Elemento	Alta	Eliel	5	Não Iniciado
1	029.02	Criar a classe Objeto de Elemento	Alta	Eliel	2	Não Iniciado
1	029.03	Criar a classe DAO de Elemento	Alta	Eliel	3	Não Iniciado
1	030.01	Criar a página de consultar suprimento	Alta	Rafael	3	Não Iniciado
1	031.01	Criar a página de cadastrar suprimento	Alta	Rafael	2,5	Não Iniciado
1	031.02	Criar o consultar Inserir, alterar e excluir elemento na página de cadastrar suprimento	Alta	Eliel	9	Não Iniciado
1	032.01	Criar a página de alterar suprimento	Alta	Rafael	1,5	Não Iniciado
1	032.02	Criar o consultar Inserir, alterar e excluir elemento na página de alterar suprimento	Alta	Eliel	4	Não Iniciado
1	033.01	Criar a página de excluir suprimento	Alta	Rafael	3	Não Iniciado
1	034.01	Criar a classe Bean de Comprovação	Alta	Alberto	5	Não Iniciado
1	034.02	Criar a classe Objeto de Comprovação	Alta	Alberto	2	Não Iniciado
1	034.03	Criar a classe DAO de Comprovação	Alta	Alberto	2	Não Iniciado
1	035.01	Criar a página de Consultar elementos para comprovação	Alta	Alberto	3	Não Iniciado
1	036.01	Criar a página de comprovar elemento	Alta	Alberto	9	Não Iniciado
	037.01	Criar Documento de Requisição Suprimento	Média			Não Iniciado
	038.01	Criar Documento de Comprovações Elemento	Média			Não Iniciado

Figura 6: Backlog do Produto na primeira iteração.

O Próximo passo foi a realização da reunião de planejamento da primeira *sprint*. Está reunião foi para que o Gerente do Produto determinasse quais tarefas de maior prioridade do *Backlog* do Produto o Time deveria realizar na primeira iteração, formando o *Backlog* do *Sprint*, como mostra a figura 7.

Backlog do Sprint - 1 — Horas dia: 6

Seg	Ter	Qua	Qui	Sex
		9/dez	10/dez	11/dez
14/dez	15/dez	16/dez	17/dez	18/dez

Atualização 18/12/09

Id	Item	Prioridade	Responsável	Horas estimadas	Dia 1	Dia 2	Dia 3	Dia 4	Dia 5	Dia 6	Dia 7	Dia 8	Situação
001.01	Criar o DER do SSF	Alta	Eliel	3	0	0	0	0	0	0	0	0	Finalizado
002.01	Criar o script do banco SSF	Alta	Alberto	3	0	0	0	0	0	0	0	0	Finalizado
003.01	Criar a classe Bean de Responsável	Alta	Alberto	2	1	0	0	0	0	0	0	0	Finalizado
003.02	Criar a classe Objeto de Responsável	Alta	Alberto	1,5	0,5	0	0	0	0	0	0	0	Finalizado
003.03	Criar a classe DAO de Responsável	Alta	Alberto	1,75	0,75	0	0	0	0	0	0	0	Finalizado
008.01	Criar a classe Bean de Base Legal	Alta	Eliel	0,75	0	0	0	0	0	0	0	0	Finalizado
008.02	Criar a classe Objeto de Base Legal	Alta	Eliel	0,5	0	0	0	0	0	0	0	0	Finalizado
008.03	Criar a classe DAO de Base Legal	Alta	Eliel	0,5	0	0	0	0	0	0	0	0	Finalizado
013.01	Criar a classe Bean de Projeto	Alta	Rafael	0,75	0	0	0	0	0	0	0	0	Finalizado
013.02	Criar a classe Objeto de Projeto	Alta	Rafael	0,5	0	0	0	0	0	0	0	0	Finalizado
013.03	Criar a classe DAO de Projeto	Alta	Rafael	0,5	0	0	0	0	0	0	0	0	Finalizado
018.01	Criar a classe Bean de Unidade Orçamentária	Alta	Alberto	0,75	0,75	0	0	0	0	0	0	0	Finalizado
018.02	Criar a classe Objeto de Unidade	Alta	Alberto	0,5	0,5	0	0	0	0	0	0	0	Finalizado
018.03	Criar a classe DAO de Unidade Orçamentária	Alta	Alberto	0,5	0,5	0	0	0	0	0	0	0	Finalizado
023.01	Criar a classe Bean de Elemento de Despesa	Alta	Eliel	0,75	0	0	0	0	0	0	0	0	Finalizado
023.02	Criar a classe Objeto de Elemento de Despesa	Alta	Eliel	0,5	0,25	0	0	0	0	0	0	0	Finalizado
023.03	Criar a classe DAO de Elemento de Despesa	Alta	Eliel	0,5	0,25	0	0	0	0	0	0	0	Finalizado
028.01	Criar a classe Bean de Suprimento	Alta	Rafael	3	2	2	1	0	0	0	0	0	Finalizado
028.02	Criar a classe Objeto de Suprimento	Alta	Rafael	1,75	1	1	0,5	0	0	0	0	0	Finalizado
028.03	Criar a classe DAO de Suprimento	Alta	Rafael	1,5	1	1	0,5	0	0	0	0	0	Finalizado
029.01	Criar a classe Bean de Elemento	Alta	Eliel	5	5	4	3	2	1	1	1	0	Finalizado
029.02	Criar a classe Objeto de Elemento	Alta	Eliel	2	2	1,5	1	1	1	1	0,5	0	Finalizado
029.03	Criar a classe DAO de Elemento	Alta	Eliel	3	3	2,5	2	2	2	1	1	0	Finalizado
030.01	Criar a página de consultar suprimento	Alta	Rafael	3	2,5	2,5	0	0	0	0	0	0	Finalizado
031.01	Criar a página de cadastrar suprimento	Alta	Rafael	2,5	1	1	1	0	0	0	0	0	Finalizado
031.02	Criar o consultar Inserir, alterar e excluir	Alta	Eliel	9	9	6,5	6,5	5,5	4,5	2,5	0	0	Finalizado
032.01	Criar a página de alterar suprimento	Alta	Rafael	1,5	1,5	1,5	1	0	0	0	0	0	Finalizado
032.02	Criar o consultar Inserir, alterar e excluir	Alta	Eliel	4	4	4	4	4	4	4	2	0	Finalizado
033.01	Criar a página de excluir suprimento	Alta	Rafael	3	3	3	3	1,5	0	0	0	0	Finalizado
034.01	Criar a classe Bean de Comprovação	Alta	Alberto	5	5	4	4	4	3,5	2	0	0	Finalizado
034.02	Criar a classe Objeto de Comprovação	Alta	Alberto	2	2	1,5	1,5	1,5	1,5	1	0	0	Finalizado
034.03	Criar a classe DAO de Comprovação	Alta	Alberto	2	2	1,5	1,5	1,5	1,5	1	0	0	Finalizado
035.01	Criar a página de Consultar elementos para	Alta	Alberto	3	3	3	0	0	0	0	0	0	Finalizado
036.01	Criar a página de comprovar elemento	Alta	Alberto	9	9	9	7	7	5,5	2	0	0	Finalizado

Figura 7: *Sprint* da primeira iteração.

Logo após a reunião de planejamento da sprint começa o desenvolvimento da sprint.

Desenvolvimento – Iteração 1

A meta da primeira sprint foi desenvolver tarefas relacionadas ao “coração” do produto (Cadastro de Suprimento e sua comprovação).

A primeira *sprint* foi realizada em 8 dias, durante a sua realização não ocorreram grandes problemas, foram realizadas reuniões diárias (Figura 8) de 15 minutos no final do expediente de trabalho, onde cada um mostrou o que tinha feito até o momento e o que iria fazer para o outro dia.

Durante o desenvolvimento da primeira *sprint* foram utilizadas diversas técnicas abordadas pelo XP como: Projeto Simples; Refatoração; Testes de Aceitação; Integração contínua; Cliente presente; Prioridade coletiva; Padrões de codificação;

Ao final da *sprint* foi realizada uma reunião de revisão da *sprint*. O Gerente do produto verificou o que foi feito e o que não foi. O Time relatou quais os problemas foram encontrados durante a *sprint* e o que foi feito para resolver. Isso ajudou a todos do time conhecer os problemas encontrados, fazendo com que não ocorresse mais durante as próximas *sprints*.

Durante a segunda parte da reunião o Líder do time procurou saber sobre cada membro do Time se durante o desenvolvimento da *sprint* passou alguma dificuldade no processo de desenvolvimento do modelo proposto, para que durante as próximas *sprints* o tornasse mais eficiente. Como quanto a isso nada foi relatado o processo continuou sendo o mesmo.

Após final da reunião de reunião da revisão da *sprint*, foi dado o inicio do Pós-Game.

Pós-Game – Iteração 1

Nessa etapa o incremento do produto (Figura 9) foi entregue aos interessados para que eles utilizassem e avaliassem. Também foi gerada uma documentação para entendimento de como executar o produto.

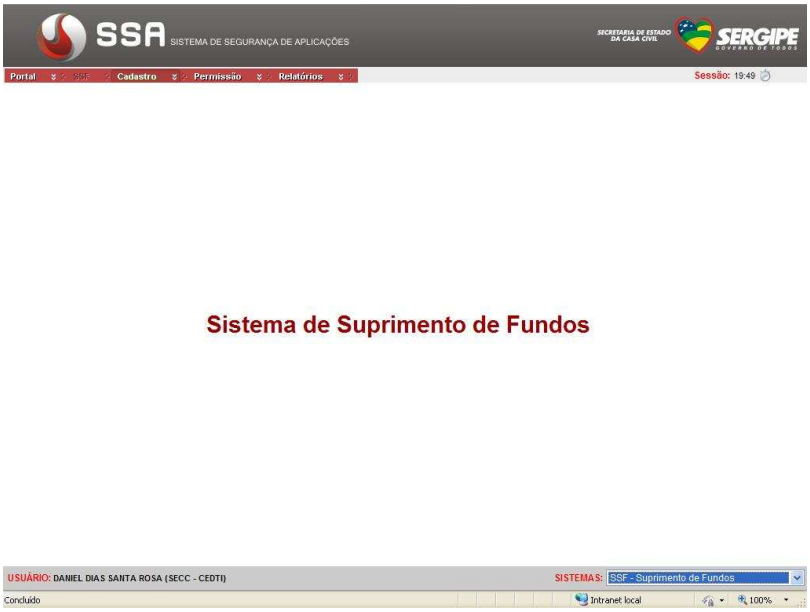

Figura 9: Tela Inicial do Sistema

Logo após o pós-game, a Iteração 2 foi iniciada. As iterações 2 e 3 seguiram praticamente os mesmos passos da iteração 1. Após o encerramento da iteração 3, como não existia mais nenhuma tarefa no *backlog* do produto o projeto foi dado como finalizado.

4. Conclusão

O Gerenciamento Ágil é bastante flexível, ele permite que uma empresa que o adote englobe características de vários métodos e modelos ágeis (XP, *Scrum* entre outros).

O desenvolvimento de um modelo proposto para SECC baseado no XP e *Scrum* mostrado no capítulo 3, permitiram ter uma melhor visão sobre os projetos de desenvolvimento de software (oportunidade de gerenciar, controlar e entender os projetos de uma maneira mais eficiente); uma maior participação e satisfação da equipe de desenvolvimento (oportunidade de trabalhar em um ambiente melhor fazendo a equipe mais eficiente e produtiva); uma maior participação e satisfação dos interessados (oportunidade de ter um maior interesse nos projetos e ajudando na suas definições); os

usuários tiveram a oportunidade de obter um produto mais próximo de suas necessidades. Devido a esses benefícios, gerou um produto de maior qualidade com a participação do cliente atingindo o sucesso do produto, produzindo também um padrão de codificação e integração.

Referências

Abrahamsson, P., Salo, O., Ronkainen, J. and Warsta J. Agile Software Development Methods: Review and Analysis, Espoo 2002, VTT Publications 478.

Beck, Kent; FOWLER, M. Extreme programming applied. Boston: Addison-Wesley, 2001.

Dias, Marisa Villas Bôas. Um novo enfoque para o gerenciamento de projetos de desenvolvimento de software. 2005. Disponível em: <http://www.teses.usp.br/teses/disponiveis/12/12139/tde-03012006-122134/>. Acessado em: 24 de agosto de 2009.

Ferreira, Renata Bastos && LIMA, Francisco de Paula Antunes. Metodologias Ágeis: Um Novo Paradigma de Desenvolvimento de Software. 2006. Disponível em: <http://www.cos.ufrj.br/~handrade/woses/woses2006/pdfs/10-Artigo10WOSES-2006.pdf>. Acessado em: 23 de julho de 2008.

Filho, Franklin Metodologias Ágeis de Desenvolvimento de Software: Um estudo comparativo entre Scrum e Exteme Programming. Fortaleza 2007, Trabalho de Conclusão de Curso- Curso de Sistemas de Informação, Faculdade Integrada do Ceara.

Franco, Eduardo Ferreira. Um modelo de gerenciamento de projetos baseado nas metodologias ágeis de desenvolvimento de softwares e nos princípios da produção enxuta. 2007. Disponível em: <http://www.teses.usp.br/teses/disponiveis/3/3141/tde-09012008-155823/>. Acessado em: 24 de agosto de 2009.

Muller, Elias. Uma aplicação do Scrum no SIMUPLAN. Disponível em: <http://inf.upf.br/gepes/downloads/TC_Elias_Muller_M%E9todos_%C1geis.pdf>. Acessado em: 26 de julho de 2009.

Mattei, Jaqueline Carvalho. Uma análise do método ágil Scrum sob a perspectiva da categoria Gerência de Processo do modelo CMMI. Disponível em: <http://inf.upf.br/gepes/downloads/TC_Elias_Muller_M%E9todos_%C1geis.pdf >. Acessado em: 26/11/2009.

Pressman, Roger S. Engenharia de Software. 6 ed., 2006

Schwaber, Ken. Guia do Scrum , Scrum Alliance, 2009