UNIVERSIDADE SAGRADO CORAÇÃO

HENRIQUE PEDON MAKAUSKAS

SOFTWARE WEB PARA APOIO DO GERENCIAMENTO DE PROJETOS COM FOCO NA METODOLOGIA ÁGIL SCRUM

BAURU

2013

HENRIQUE PEDON MAKAUSKAS

SOFTWARE WEB PARA APOIO DO GERENCIAMENTO DE PROJETOS COM FOCO NA METODOLOGIA ÁGIL SCRUM

Trabalho de Conclusão de curso apresentado ao centro de Ciências exatas e Sociais aplicadas como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação.

Orientadora: Prof.ª Ms. Elaine Cecília Gatto

BAURU

2013

Makauskas, Henrique Pedon
M2352s

Software web para apoio do gerenciamento de projetos com foco na metodologia ágil Scrum / Henrique Pedon Makauskas -- 2013.

61f. : il.

Orientadora: Profa. Me. Elaine Cecília Gatto.

Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) – Universidade do Sagrado Coração – Bauru – SP.

1. Gerenciamento de projetos. 2. Metodologia Ágil. 3. Scrum. I. Gatto, Elaine Cecília. II. Título.

Dedico este trabalho a todos que de alguma forma me ajudaram a conquistar mais essa etapa da minha vida e em especial aos meus pais que em nenhum momento deixaram de acreditar em mim.

AGRADECIMENTOS

Muito foram as pessoas que abriram as portas, caminhos e incentivaram meus estudos.

Agradeço a professora Dra. Patrícia Bellin Ribeiro que foi minha orientadora no começo desse trabalho e muito contribuiu para definição do tema e compartilhou seu conhecimento me indicando como iniciar um trabalho acadêmico.

Á professora Ms. Elaine Cecília Gatto que também foi minha orientadora e dispôs em muitos momentos de seu precioso tempo para me orientar e me ajudar a concluir esse trabalho, a professora meus sinceros agradecimentos.

Aos meus pais Carlos e Maria José e meu irmão Guilherme que muito contribuiu para o desenvolvimento do software.

Aos meus amigos que pacientemente entenderam minha ausência durante o andamento deste projeto.

Por fim, demonstro todo meu apreço aos profissionais da Universidade Sagrado do Coração, que diretamente ou indiretamente participaram do meu desenvolvimento profissional e acadêmico.

"Pessoas não fracassam. Elas simplesmente desistem."
(Henry Ford)

RESUMO

O gerenciamento de projetos na última década vem passando por uma grande transformação, principalmente na área de criação de softwares. As abordagens tradicionais que eram utilizadas já não atendem mais as tendências do desenvolvimento de softwares atuais, que trabalham de um forma empírica e com uma grande imprevisibilidade. Dentro desse contexto as metodologias ágeis se destacam, entre elas o Scrum aparece sendo o mais utilizado. Um método ágil que tem como principal objetivo entregar pequenas partes funcionais do sistema no menor tempo possível utilizando o empirismo, e tratando as mudanças como parte do processo de desenvolvimento de software. Este trabalho se propôs a desenvolver um software de apoio ao gerenciamento de projetos utilizando metodologia ágil Scrum. O sistema foi desenvolvido para Web, e apresenta os módulos de administração e projetos voltados para o uso da metodologia em questão. Para validar o projeto foi desenvolvido um questionário com casos de testes, onde profissionais de diversas área realizaram a análise tendo como resultados obtidos, falhas e melhorias encontradas, e também a avaliação do software como positiva.

Palavras-chave: Scrum. Metodologia ágil. Gerenciamento de projetos

ABSTRACT

Project management in the last decade has been undergoing a major transformation particularly in the area of software development . Traditional approaches that were used no longer meet the most current trends in software development, that requires an empirical way and unpredictability . Within this context agile methodologies stand out among them appears Scrum being the most used . An agile method that aims to deliver small functional parts of the system in the shortest possible time using empiricism , and treating the changes as part of the software development process . This study proposes to develop a support software project management using Scrum agile methodology . The system was developed for Web , and provides administration and projects modules. To validate the software a questionnaire with test cases where professionals from different area performed the analysis and the results were found as failures and improvements, and the evaluation of the software was positive.

Keywords: *Scrum. Agile. Project Management*

LISTA DE ILUSTRAÇÕES

Figura 1 - Aumentos específicos maior ou igual a 10% por implementar a abordagem Ágil 14
Figura 2 - Quais os percentuais de projetos ágeis obtiveram sucesso na perspectiva das organizações 14
Figura 3 - Qual a metodologia ágil as empresas estão seguindo.....15
Figura 4 - Camadas da engenharia de software 16
Figura 5 - Visão geral das áreas de conhecimento em gerenciamento de projetos e os processos de gerenciamento de projetos. 20
Figura 6 -Metodologia Scrum 24
Figura 7 - *Product Backlog* 29
Figura 8 - Simulação de uma solicitação e resposta 33
Figura 9 - Diagrama de caso de uso - Módulo Administração.....38
Figura 10 - Diagrama de caso de uso - Módulo Projetos 38
Figura 11 - Diagrama de atividade 39
Figura 12 - Modelo entidade relacionamento 40
Figura 13 -Tela de administração.....42
Figura 14 - Tela de permissões de acesso.....42
Figura 15 - Tela de gerenciamento dos projetos 43
Figura 16 - Tela de acompanhamento do projeto (*Sprints*) 43
Figura 17 - Tela de acompanhamento do projeto (*Product Backlog*) 44
Figura 18 -Tela de detalhes da atividade 45
Figura 19 - Resultado - Qual é a sua formação acadêmica? 48
Figura 20 - Resultado - Qual é a sua profissão? 48
Figura 21 - Resultado - Quantas horas por semana, em média, você utiliza o computador? 49
Figura 22 - Resultado - Há quanto tempo você navega na Internet? 49
Figura 23 - Resultado - Falhas encontradas. 50
Figura 24 - Resultado - Melhorias. 51
Figura 25 - Resultado - Usabilidade. 52
Figura 26 - Resultado - Avaliação Geral.....52

LISTA DE ABREVIATURAS E SIGLAS

PMI - *Project Management Institute*

PMBOK - *Project Management Body of Knowledge*

SWEBOK - *Software Engineering Body of Knowledge*

COBIT - *Control Objectives for Information and related Technology*

ITIL - *Information Technology Infrastructure Library*

SUMÁRIO

1 INTRODUÇÃO 11
1.1 Objetivos 13
1.3 Objetivos específicos 13
1.4 Justificativa 13

2 FUNDAMENTAÇÃO TEÓRICA 16
2.1 Engenharia de software 16
2.2 Gerenciamento de projetos 18
2.3 Metodologias ágeis 21
2.3.1 Scrum 24
2.3.2 Estrutura do processo Scrum 26

3 MATERIAIS E MÉTODOS 33
3.1 *World Wide Web* (Web) 33
3.2 Servidor Web - Apache 34
3.3 Linguagem de Programação PHP 34
3.4 Framework PHP 35
3.5 Banco de Dados 35

4 MODELAGEM DA APLICAÇÃO 36
4.1 Requisitos funcionais 36
4.1.1 Módulo de administração 36
4.1.2 Módulo de projetos 36
4.2 Diagramas 37

5 APLICAÇÃO 42

6 TESTES E RESULTADOS 46
6.1 Casos de teste 46
6.2 Resultados 47

CONSIDERAÇÕES FINAIS 53

REFERÊNCIAS 54

APÊNDECIE - QUESTIONÁRIO DE AVALIAÇÃO DO SOFTWARE 57

1 INTRODUÇÃO

A palavra projeto pode ser muitas vezes mal interpretada e assim podemos ter diferentes conceitos sobre a mesma, Exemplos: *Projeto de Vida* (significando um plano de como pretende viver daqui para frente), *Projeto de Pagamentos* (cronograma de pagamentos), entre outros.

Projeto é um conjunto de atividades ou medidas planejadas para serem executadas com responsabilidade de execução definidas a fim de alcançar determinados objetivos dentro de uma abrangência definida, num prazo de tempo limitado, com recursos específicos e criando algo novo (PRIKLADNICKI; ORTH, 2009).

Outro conceito para Projeto vem do *Project Management Institute* (Instituto de gerenciamento de projeto), segundo o qual “um projeto é um esforço temporário realizado para criar um produto ou serviço único”.

Todo projeto inicia com uma necessidade e uma visão de solução sempre levando em conta as limitações de tempo e custo. A primeira etapa é entender as necessidades do cliente e direcionar elas para dentro de um contexto da realidade computacional. Na segunda etapa é feito a analise de requisitos e definição técnica do projeto onde é levantado tudo o que é preciso para atingir os objetivos esperados pelo cliente. Na terceira etapa temos o desenvolvimento, e por fim a quarta etapa são realizado testes e a homologação por parte do cliente (MARTINS, 2007).

Atualmente o desenvolvimento de sistemas está longe de ser o ideal, em grande maioria os projetos são entregues com atraso, com orçamento estourado ou nem mesmo são entregues. Normalmente os sistemas quando são concluídos não atendem à todas as necessidades do cliente gerando projetos infinitos ou grande retrabalho por parte da equipe de desenvolvimento. Tudo isso gera um grande desconforto entre o cliente e a empresa que está desenvolvendo o sistema dificultando o relacionamento entre eles. Como resultado final há um cliente que não auxilia a equipe durante o desenvolvimento não dando o suporte necessário para atingir todos os objetivos (AMBLER, 2004).

Com a crise global que atingiu o mundo recentemente, o objetivo da maioria das empresas é cortar custos e assim aumentar os lucros. O custo destinado para projetos estão diminuindo forçando as empresas a procurarem mão de obra barata

em países da Ásia, Austrália e Europa Ocidental com taxas em média 25% menores que países ricos, esses países vem sendo a preferência para enfrentar o baixo orçamento destinado a projetos (ANDERSON, 2004).

Segundo dados levantados por Carvalho e Mello (2012), em uma pesquisa com mais de 30 mil projetos de desenvolvimento de sistemas desde 1994, mostram que uma taxa de sucesso para grandes projetos é estatisticamente nula e que para médios e pequenos projetos a taxa é de 55%. Os dados desse estudo mostram que o custo e tempo do projeto normalmente ultrapassam os valores definidos no início do projeto, além disso as funcionalidades planejadas nem sempre são implementadas em sua totalidade.

Baseado no contexto apresentado acima as empresas tem investido cada vez mais na área de gerência de projetos com foco em eliminar os problemas de projetos atrasados, que não atendem 100% da necessidade do cliente, com orçamento estourado, baixa qualidade e etc. (PRIKLADNICKI; ORTH, 2009).

Por décadas o desenvolvimento de software seguiu o modelo clássico de cascata para desenvolvimento de produtos. Nesse modelo, o projeto passa por diversas etapas (análise, design, documentação, codificação e testes) antes de ser entregue ao cliente, diminuindo a flexibilidade do processo e prejudicando a obtenção de repostas rápidas às exigências de mercado por parte da empresa. Em busca de um novo método de trabalho mais flexível e com documentação reduzida surgem as metodologias ágeis (LOESER, 2006).

> Para enfrentar esses desafios encontrados pelos desenvolvedores de software, um grupo inicial de 17 metodologias formou a *Agile software Development Alliance*, freqüentemente citada apenas como *Agile Alliance*, em fevereiro de 2001. Um aspecto interessante deste grupo é que todos vieram de diferentes áreas de formação e ainda assim são capazes de concordar em questões nas quais os metodologistas normalmente não concordam. Este grupo de pessoas definiu um manifesto para encorajar melhores meios de desenvolver software e, com base nesse manifesto, formulou um conjunto de princípios que definem critérios para os processos de desenvolvimento ágil de software, como a Modelagem Ágil (AMBLER, 2004).

Para esse projeto será abordada a metodologia ágil Scrum, por ser atualmente a metodologia mais difundida no mercado de trabalho. O Scrum é um conjunto de boas práticas para gerenciamento de projetos complexos o qual é fundamentado nas teorias empíricas de controle de processo, desta forma o

conhecimento é baseado nas lições aprendidas. O Scrum emprega uma abordagem iterativa e incremental para o desenvolvimento do projeto, que consiste em equipes do Scrum associadas a seus papéis, eventos, artefatos e regras (SCRUM.ORG, 2013).

1.1 Objetivo Geral

Desenvolver uma ferramenta de apoio para gerenciamento de projetos baseado na metodologia ágil Scrum.

1.2 Objetivos específicos

- Aprofundar os conhecimentos na metodologia ágil;
- Definir as atividades básicas necessárias para um projeto baseado em Scrum;
- Realizar a modelagem do sistema e banco de dados;
- Estudar as ferramentas necessárias para o desenvolvimento do projeto;
- Desenvolver o software.

1.3 Justificativa

Uma pesquisa feita pelo *The Standish Group* em 2012, mostra que apenas 39% dos projetos tiveram sucesso, 18% foram cancelados e 43% tiveram algum tipo de problema. Dentro dos problemas mais encontrados podemos destacar: requisitos incompletos, falta de apoio dos usuários, expectativas fora da realidade, mudanças do escopo do projeto, falta de apoio dos negócios e falta de recurso.

Como mostra as informações vistas até o momento, as metodologias ágeis indicam que, diferentemente das atividades de engenharia, nas quais as metodologias tradicionais são melhores, o processo de desenvolvimento de sistemas pode ser imprevisível dificultando a criação de um escopo inicial fixo como os métodos tradicionais orientam. As metodologias ágeis consideram mudanças como parte do desenvolvimento de programas de uma forma que elas possam ser adaptadas com facilidade.

Um pesquisa feita pela VersionOne em 2008, mostram que 89% das pessoas que utilizaram metodologias ágeis em seus projetos tiveram um ganho igual ou

superior a 10% na produtividade do desenvolvimento, esse e outros benefícios são ilustrados na Figura 1.

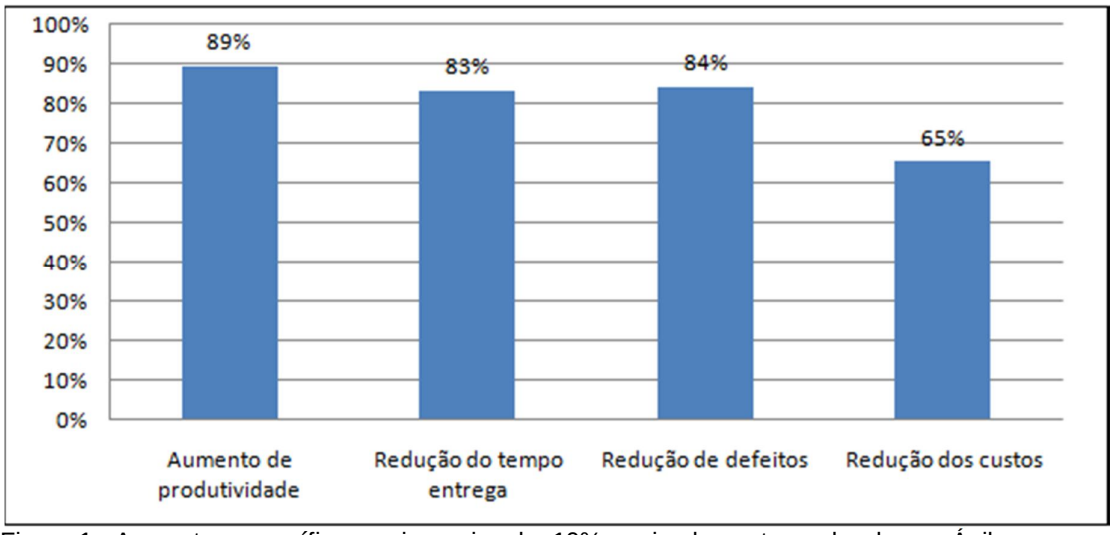

Figura 1 - Aumentos específicos maior ou igual a 10% por implementar a abordagem Ágil.
Fonte: VERSIONE(2008).

Nota-se na Figura 1 uma melhora significativa em vários itens importantes dentro de uma organização, como aumento de produtividade, redução de custos, redução de defeitos de software e aceleração do tempo de mercado. Também dentro dessa pesquisa outro dado que chama a atenção é referente a seguinte pergunta: "Quais os percentuais de projetos ágeis obtiveram sucesso na perspectiva da sua organização?", onde 55% das pessoas responderam que 90% ou 100% dos projetos que usam a agilidade obtém sucesso, podemos observar mais detalhes na Figura 2.

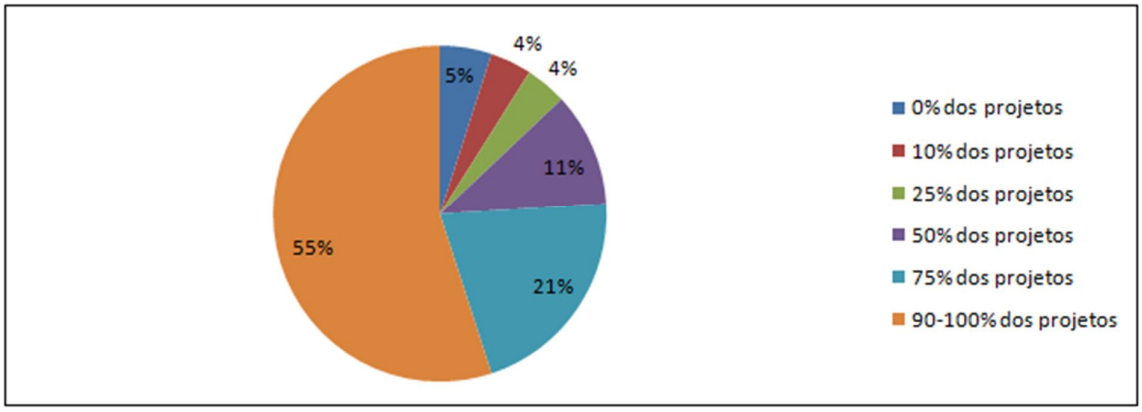

Figura 2 - Quais os percentuais de projetos ágeis obtiveram sucesso na perspectiva das organizações.
Fonte: VERSIONE(2008).

Nessa mesma pesquisa podemos notar que o Scrum é a metodologia mais utilizada, conforme apresenta a Figura 3.

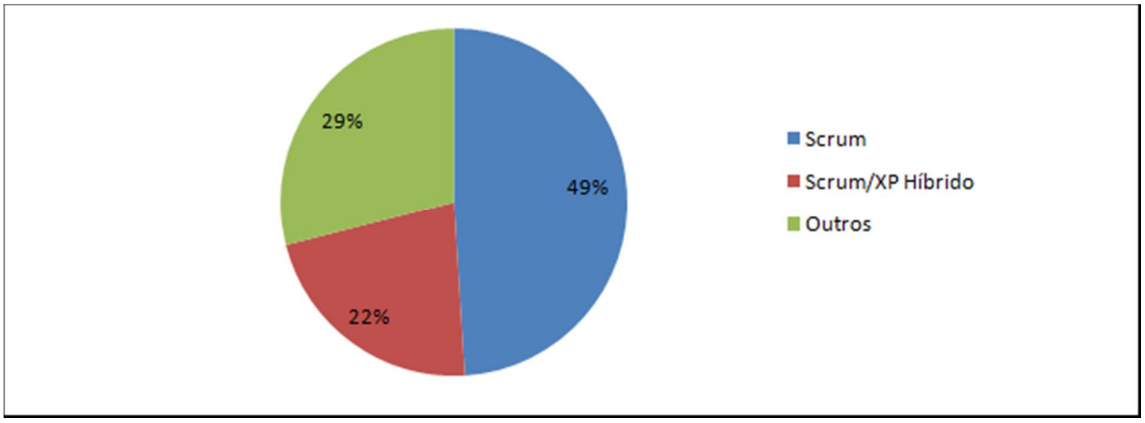

Figura 3 - Qual a metodologia ágil as empresas estão seguindo.
Fonte: VERSIONE(2008).

Com o mercado cada vez mais competitivo a necessidade de desenvolver programas com objetivo de redução de gastos e contratação de mão de obra especializada é cada vez mais claro. Com o crescimento do uso dessa metodologia fica evidente a necessidade de uma ferramenta que fornece o apoio a todo processo aumentando a produtividade, mantendo um histórico de informações e proporcionando uma melhor visão para todos envolvidos no projeto. As ferramentas simples mas que atendem as necessidades das equipes são preferíveis (FOWLER; BECK, 2001).

Assim chegamos a necessidade da criação de um *software* que ajudará no gerenciamento de projetos utilizando a metodologia ágil Scrum.

2 FUNDAMENTAÇÃO TEORICA

2.1 Engenharia de Software

A engenharia de *software* relaciona todos itens necessários para a criação de programas (SOMMERVILLE, 2008).

Todos os setores da economia tem utilizado a tecnologia da informação como diferencias estratégicos para aumentar a produtividade e também gerenciar diversos itens como a produção, máquinas, estoques, entre outros. Com base nesse cenário, produzir e manter software dentro de prazos, critérios de qualidade e custos adequados torna-se requisito obrigatório (SOMMERVILLE, 2008).

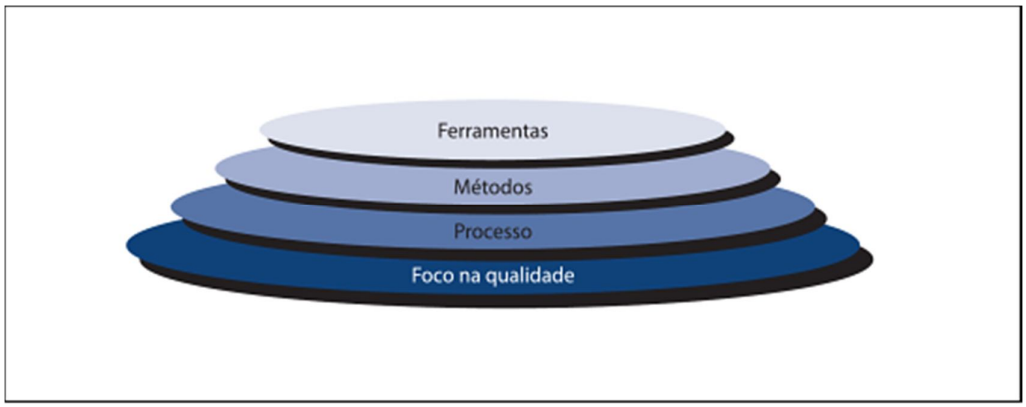

Figura 4 - Camadas da engenharia de software.
Fonte: PRESSMAN (2006).

A Figura 4 apresenta as camadas de engenharia de software. A primeira camada se refere às ferramentas que fornecem apoio automatizado ou semi-automatizado para os processos e métodos.

Segundo Pressman (2006):

> "quando as ferramentas são integradas de forma que a informação criada por uma ferramenta possa ser usada por outra, é estabelecido um suporte ao desenvolvimento de software chamado engenharia de software auxiliada por computador".

A segunda camada de métodos de engenharia de *software* representa as atividades, modelos e artefatos, "fornecem a técnica de como fazer para construir software", reunindo as tarefas que inclui entre elas análise de requisitos, projeto, construção de programas, etc. Os métodos orientam os processos recomendando práticas mais adequadas e atividades a serem seguidas.

A terceira camada a de processos é base da engenharia de software. O processo define a metodologia que será utilizada no desenvolvimento, possibilitando a criação de uma base para o controle do gerenciamento de projetos de software, gerando toda a documentação necessária.

A quarta e última camada é o item que sustenta a engenharia de software. A gestão da qualidade é um comprometimento que todas empresas devem possuir de uma forma que promova uma cultura de aperfeiçoamento contínuo de processos (PRESSMAN, 2006).

De acordo com Sommerville (2008), existem quatro atividades fundamentais que são comuns a todos os processos de software:

- Especificação de software: clientes e engenheiros definem o software a ser produzido e suas restrições;
- Desenvolvimento de software: o software é projetado e programado;
- Validação de software: o software é verificado para garantir que atende o que o cliente deseja;
- Manutenção de software: o software é adaptado para atender as modificações do cliente e do mercado.

Por fim, o apoio para todas as camadas da engenharia de software é o foco na qualidade do produto final gerado. A engenharia oferece processos, ferramentas e métodos de forma que o coordenador de projetos seja capaz de garantir que o software será feito com a qualidade.

Os processos de *software* são difíceis e são dependentes da percepção do coordenador de projetos e da característica do software que se deseja desenvolver. Cada sistema, variando do seu objetivo, vai exigir um processo de construção diferente e, apesar da variedade de modelos de processos de software, não há um processo ideal, forçando as organizações a realizarem adaptações para atender suas demandas de desenvolvimento.

Alguns *softwares* complexos precisam de um processo de desenvolvimento estruturado, ou mais rígidos, outros sistemas possuem requisitos que mudam rapidamente exigindo um processo mais flexível e ágil (SOMMERVILLE, 2008).

O guia Swebok desenvolvido pela *IEEE Computer Society* uma organização de profissionais da área de computação introduz a engenharia de software como uma disciplina a ser estudada como ciências da computação, gerenciamento de

projetos, matemática, etc., descrevendo as áreas de conhecimento e servindo como guia de conhecimento para o desenvolvimento de softwares (SWEBOK, 2004).

2.2 Gerenciamento de Projetos

Planejamento e gerenciamento de projetos fazem parte da vida das pessoas desde o começo dos tempos, sempre existiram projetos para serem gerenciados: construções, estradas a pavimentar e leis a serem escritas. Mesmo sem as ferramentas, técnicas e metodologias que temos hoje, as pessoas criaram linhas de tempo, reservaram materiais e recursos e analisaram os riscos envolvidos em seus projetos.

As organizações precisam, a cada dia, se atualizar e melhorar a forma como projetos são realizados com o objetivo de alcançar a excelência de produtos e/ou serviços. O objetivo das empresas é o cumprimento de metas com base em seu planejamento estratégico melhorando os resultados em suas entregas de produtos ou serviços prestados. Um gerenciamento de projetos eficaz pode contribuir neste desafio (FREITAS, 2011).

O Instituto de Gerenciamento de Projetos ou *Project Management Institute* - PMI (2008) define um projeto como um esforço temporário empreendido para criar um produto, serviço ou resultado único.

Temporário significa que um projeto tem início, meio e um final definidos. O final é alcançado quando os objetivos do projeto tiverem sido atingidos, quando se tornar claro que os objetivos do projeto não serão ou não poderão ser atingidos ou quando não existir mais a necessidade do projeto e ele for encerrado.

Produtos, serviços ou resultados exclusivos Um projeto cria entregas exclusivas, que são produtos, serviços ou resultados.

Os projetos podem criar:

1. Um produto ou objeto produzido, quantificável e que pode ser um item final ou um item componente;
2. Uma capacidade de realizar um serviço, como funções de negócios que dão suporte à produção ou à distribuição;
3. Um resultado, como resultados finais ou documentos. Por exemplo, um projeto de pesquisa desenvolve um conhecimento que

> pode ser usado para determinar se uma tendência está presente ou não, ou se um novo processo beneficiará a sociedade.

A elaboração progressiva é uma característica de projetos que integra os conceitos de temporário e exclusivo. Elaboração progressiva significa desenvolver em etapas e continuar por incrementos.

Segundo Koontz (1980) gerenciar baseia-se em realizar atividades e tarefas, que têm como finalidade fazer o planejamento e o controle das atividades de outras pessoas, para alcançar objetivos que seria impossível de ser alcançado se as pessoas atuassem sem supervisão.

Em palavras mais técnicas, o gerenciamento de projetos consiste na utilização do conhecimento, habilidades, ferramentas e técnicas às tarefas do projeto visando realizar todos os seus requisitos. O gerenciamento de projetos é a integração dos seguintes processos de gerenciamento de projetos: iniciação, planejamento, execução, monitoramento e controle, e encerramento (PMI, 2008).

O guia *Project Management Body of Knowledge* (PMBOK) é baseado nas boas práticas de gerenciamento de projetos desenvolvido pelo PMI. O PMBOK é muito conceituado e aceito por diversas empresas como sendo a base para conhecimento sobre gerência de projetos. Seu maior objetivo é identificar e reunir conhecimentos sobre a área, que é consenso, sendo aplicáveis para a maior parte dos projetos na maior parte do tempo. Outro propósito é fornecer um vocabulário único para a profissão, criando uma padronização dos termos utilizados. É estruturado em nove áreas de conhecimento em gerenciamento de projetos, são elas:

1. Integração: Contém os processos necessários para assegurar a coordenação entre os demais processos;

2. Escopo: Contém os processos necessários para definir e controlar o que está incluído ou não no projeto;

3. Tempo: Contém os processos necessários para concluir o projeto dentro do cronograma;

4. Custo: Contém os processos necessários para assegurar que o projeto se encerrará dentro do orçamento;

5. Qualidade: Contém os processos necessários para assegurar que o projeto satisfará as necessidades contratadas;

6. Recursos humanos: Contém os processos necessários para gerenciar os recursos humanos envolvidos no projeto;

7. Comunicações: Contém os processos necessários para assegurar a geração, disseminação e armazenamento das informações do projeto;

8. Risco: Contém os processos necessários para identificar, analisar e responder aos riscos do projeto;

9. Aquisições: Contém os processos necessários para aquisição de bens ou serviços fora da organização.

A figura 5 apresenta uma visão geral das áreas de conhecimento em gerenciamento de projetos e os processos de gerenciamento de projetos associados.

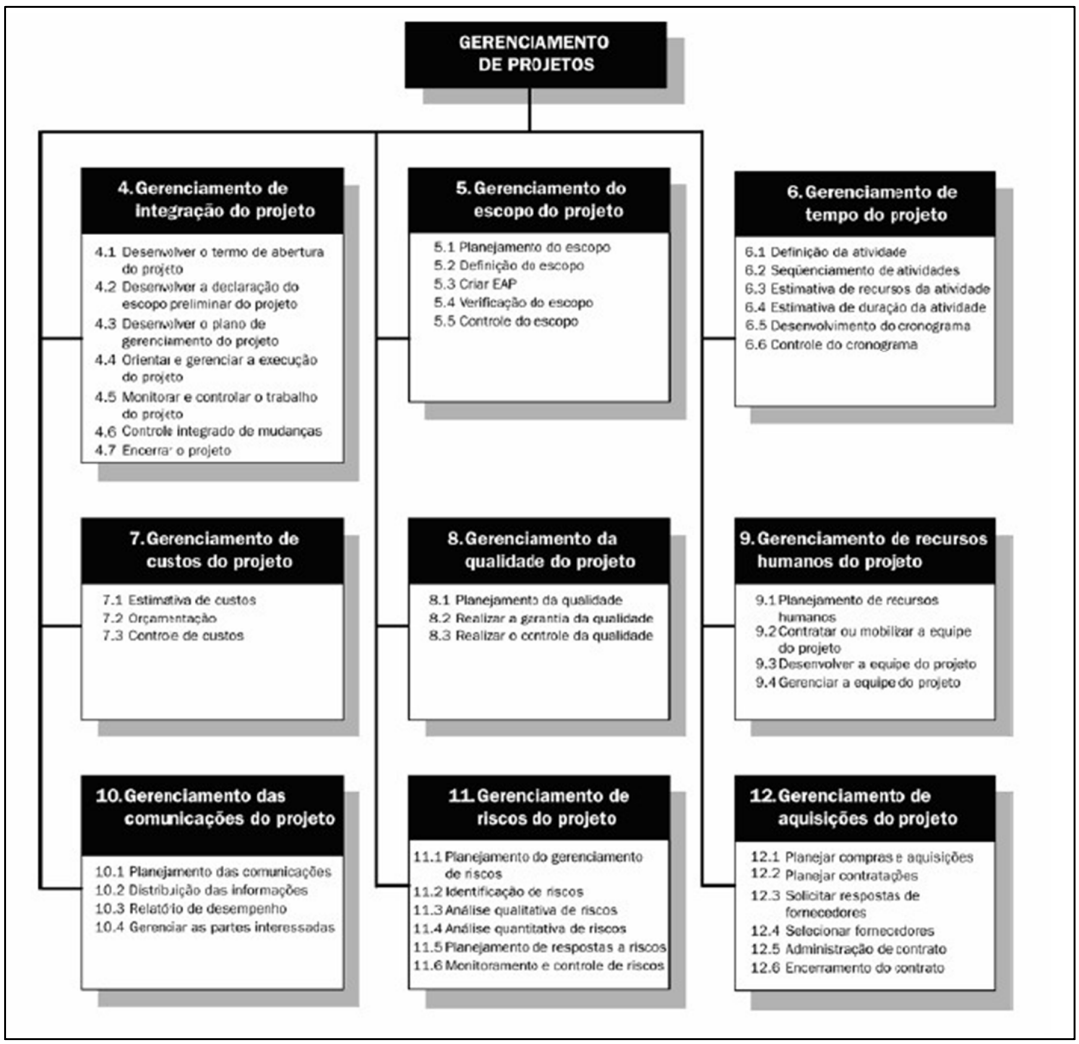

Figura 5 - Visão geral das áreas de conhecimento em gerenciamento de projetos e os processos de gerenciamento de projetos.
Fonte: PMI (2008).

O gerenciamento de projetos é um sistema interligado. A ligação dos processos do projeto precisam estar conectados e acontecendo em sequência já pré-estabelecida. O PMBOK define 44 processos de gerenciamento de projetos, estes processos estão divididos em cinco grupos (PMI, 2008), conforme descrito abaixo:

1. Grupo de processos de iniciação: São utilizados para reconhecer e iniciar um novo projeto ou fase;

2. Grupo de processos de planejamento: Define objetivo e planeja ações para alcançar estes objetivos;

3. Grupo de processos de execução: Coordena pessoas e recursos para realizar o projeto e produzir produtos e serviços;

4. Grupo de processos de monitoramento e controle: Monitora a execução do projeto para garantir que os objetivos sejam alcançados;

5. Grupo de processos de encerramento: Formaliza a aceitação do projeto e o encerra de forma ordenada.

Outros dois guias de boas práticas são: i) Cobit, que é direcionado a gestão da tecnologia da informação orientado ao negócio, possuindo diversas técnicas que servem de referência para aplicar no dia a dia de uma organização. Entre elas um guia com boas praticas para o gerenciamento (BOONEN; BRAND, 2007); ii) ITIL, voltado para a área de infraestrutura da tecnologia da informação é um conjunto de melhores práticas para o gerenciamento de serviços de TI (BON, 2007).

O PMBOK em conjunto com Cobit e ITIL são uma base para gerenciamento de projetos e largamente utilizados no mercado de trabalho, mas novas metodologias e processos apareceram com força, buscando atender a complexidade dos projetos executados que necessitam ser desenvolvidos em um curto espaço de tempo. Dentro de várias metodologias os métodos ágeis ganham bastante força e será abordado no próximo tópico.

2.3 Metodologias Ágeis

Metodologias ágeis para desenvolvimento de sistemas permitem à equipe manter o foco no desenvolvimento da aplicação, e não se preocupa tanto com a documentação. De uma forma iterativa o detalhamento do projeto é feito por partes, e não mais de uma única vez, atendendo uma grande necessidade que era a

flexibilidade do escopo dos projetos que mudavam rapidamente durante o desenvolvimento (PRESSMAN, 2006).

Segundo Martin Fowler e Kent Beck (2001) as mudanças dentro do desenvolvimento ágil são consideradas parte do processo pois são facilmente adaptadas e fortalecem o projeto. Os autores também afirmam que uma das grandes diferenças entre metodologias tradicionais e ágeis é a menor utilização de documentações extensas para criar um software.

O maior objetivo das metodologias ágeis de desenvolvimento de software é construir aplicações com menor tempo, maior produtividade e qualidade.

Um dos principais paradigmas quebrados é a forma de encarar as mudanças. Ela é algo que faz parte do desenvolvimento de sistemas, algo que sempre acontecerá, por isso é bem vinda. O escopo do projeto é flexível e pode ser alterado a qualquer momento do desenvolvimento, mesmo em fases finais (MANIFESTO ÁGIL, 2011).

Entre várias novas metodologias ágeis destacam-se: Scrum, *Extreme Programming* ou XP, Desenvolvimento Lean e Kanban.

Grande parte dessas metodologias foram criadas para atender necessidades que seus criadores enfrentavam no dia-dia do desenvolvimento de software e também ao fato de que grande número de projetos não tiveram o sucesso que era esperado ou até mesmo foram cancelados antes de chegar ao fim. Analisando os principais problemas que o desenvolvimento enfrentava criaram boas práticas, processos e elaboram metodologias para serem seguidas.

Em 2001, um grupo de pesquisadores auto denominados de Aliança Ágil (*Agile Alliance*), entre eles Martin Fowler, Alistair Cockburn, Jim Highsmith, Kent Beck, Mike Beedle entre outros, motivados pela suas experiências em desenvolvimento de software, iniciaram então uma discussão sobre como desenvolver software de uma forma mais rápida e eficaz, orientado principalmente à simplicidade. Esta discussão resultou no chamado Manifesto Ágil, que em síntese aponta:

> “Desejamos descobrir melhores caminhos para desenvolver software fazendo e ajudando outros a fazerem. Valorizamos os indivíduos e interações através de processos e ferramentas; O desenvolvimento de software deve possuir uma documentação compreensiva; A colaboração do cliente e respostas às mudanças através de um plano específico”.

Os princípios são:

- A prioridade é satisfazer o cliente, entregando o software antecipadamente. Para isso o sistema deve ser entregue em módulos, onde a prioridade será estabelecida junto com o mesmo.
- Mudanças de requisitos são bem vindas mesmo em projetos em atraso. As metodologias ágeis pregam que o software é do cliente, ou seja, ele tem o direito de solicitar modificações quando necessário, podendo assim, atender suas reais necessidades.
- A entrega do software deve ser freqüente, semanalmente, quinzenalmente ou mensalmente, preferencialmente no menor prazo possível. Para isso as metodologias ágeis trabalham com desenvolvimento incremental, ou seja, o sistema é implementado a cada nova funcionalidade definida.
- Especialistas de negócio e desenvolvedores devem trabalhar juntos ao longo de todo o projeto. Para isso, o cliente deve ser parte da equipe de desenvolvimento, estando disponível para esclarecer dúvidas e também acompanhar o projeto para verificar se suas necessidades estão sendo atendidas.
- Os projetos devem ser desenvolvidos por pessoas motivadas e aos mesmos devem ser dados todo o apoio e confiança para realização do trabalho. Para isso, o ambiente deve ser totalmente propício e agradável para a equipe sentir-se à vontade na realização de suas atividades.
- O mais eficiente e eficaz método para transmitir informação dentro de uma equipe de desenvolvimento é conversas “cara-a-cara”. Isso pode ser auxiliado com reuniões diárias dentro do projeto.
- A principal medida de progresso é o software funcionando.
- Contínua atenção a excelência técnica aumentam a agilidade e o bom design.
- Simplicidade – a arte de maximizar o valor do trabalho. Para isso, a equipe deve desenvolver o mínimo necessário para atender a necessidade do cliente.

- As melhores arquiteturas, requisitos e desenhos surgem de equipes auto organizadas.

Entre os métodos ágeis se destaca o Scrum, que será detalhado nesse trabalho, pois de acordo com dados apresentados nos capítulos anteriores é o método mais difundo hoje dentro do mercado atual.

2.3.1 Scrum

O Scrum é um método ágil que tem como principal objetivo entregar pequenas partes funcionais do sistema no menor tempo possível sempre levando em consideração o que é mais importante para negócio de uma forma iterativa e incremental (SCHWABER, 2009). A Figura 6 apresenta uma visão geral da metodologia Scrum e será detalhada nos próximos tópicos.

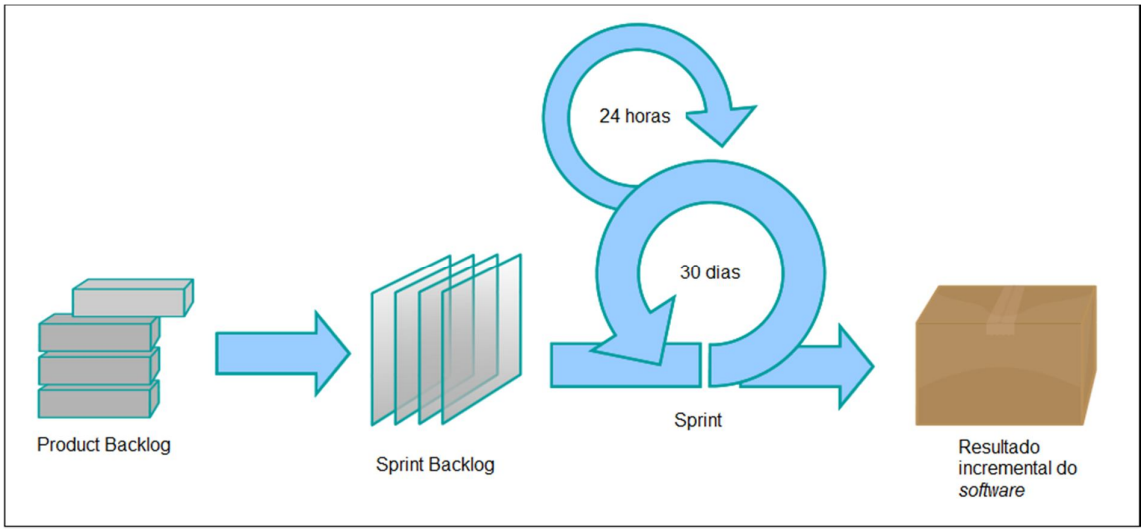

Figura 6 - Metodologia Scrum.
Fonte: SCHWABER(2004).

Papéis no Scrum

De acordo com Schwaber (2009), o *Product Owner*, ou dono do produto, é responsável pela criação do escopo do projeto e priorização das tarefas sempre maximizando o valor do produto e do trabalho da equipe de desenvolvimento, ele também representa todos os envolvidos no projeto e fornece todo detalhamento necessário para seguir com desenvolvimento.

O *Scrum Team*, ou time de desenvolvimento são profissionais auto-organizáveis e multifuncionais que tem como objetivo desenvolver uma versão

usável e incremental para um produto e também apresentar os resultados ao final de cada etapa.

E, por fim, o *Scrum Master* tem o objetivo de manter todo esse processo e boas praticas rodando sem nenhum tipo de interrupção bem como também ensinar a metodologia para o time de desenvolvimento, implementando essa nova cultura dentro das organizações.

Product Owner

Product Owner significa dono do produto. É o cliente ou um importante interessado do projeto, que representa os interesses de todos os envolvidos com o software. Suas principais responsabilidades são (ABRAHAMSSON et al., 2002):

- Definir as características e funcionalidades do produto;
- Concentrar as informações vindas das pessoas envolvidas no projeto ou do mercado, de maneira que se obtenha uma visão única dos requisitos do sistema;
- A maior responsabilidade é o ROI (*Return on Investment*) do projeto;
- Priorizar as funcionalidades do projeto de acordo com as necessidades dos usuários;
- Solicitar alterações a serem implementadas nas próximas iterações; e
- Aceitar ou rejeitar os resultados de cada iteração.

Scrum Master

O Scrum Master é o responsável por garantir que Scrum seja entendido e aplicado pois ele tem o papel de líder facilitador assegurando que todos processos do Scrum sejam realizados (SCRUM.ORG, 2013). Entre as principais responsabilidades do Scrum Master estão:

- Esclarecer o andamento do projeto;
- Difundir os valores do Scrum e suas práticas entre a equipe;
- Garantir as reuniões diárias;
- Remover obstáculos eventualmente encontrados;
- Proteger a equipe contra interferências internas;
- Conduzir os eventos realizados;
- Garantir a produtividade da equipe;

- Possibilitar uma cooperação estreita entre todos os papéis e funções.

Scrum Team

As equipes de desenvolvimento consiste de profissionais que realizam o trabalho de entregar uma versão usável que potencialmente incrementa o produto ao final de cada etapa. Somente integrantes da Equipe de Desenvolvimento criam incrementos.

As Equipes de Desenvolvimento tem as seguintes características:

- Eles são auto-organizadas. Ninguém (nem mesmo o *Scrum Master*) diz a Equipe de Desenvolvimento como desenvolver o sistema;
- Equipes de Desenvolvimento são multifuncionais, possuindo todas as habilidades necessárias, enquanto equipe, para criar o incremento do Produto.
- O Scrum não reconhece títulos para os integrantes da Equipe de Desenvolvimento que não seja o Desenvolvedor, independentemente do trabalho que está sendo realizado pela pessoa; Não há exceções para esta regra.
- Individualmente os integrantes da Equipe de Desenvolvimento podem ter habilidades especializadas e área de especialização, mas a responsabilidade pertence à Equipe de Desenvolvimento como um todo; e,
- Equipes de Desenvolvimento não contém sub-equipes dedicadas a domínios específicos de conhecimento, tais como teste ou análise de negócios.

2.3.2 Estrutura do Processo Scrum

O Scrum define processos importantes para serem seguidos no desenvolvimento do projeto. Esses processos baseados nas boas práticas para desenvolvimento de projetos com agilidade são estratégias fundamentais para o andamento do projeto e previsibilidade do que acontecerá (YOSHIMA, 2007).

O primeiro passo do Scrum começa com a criação da lista geral de funcionalidades que inicialmente deverão fazer parte do sistema, esta lista é chamada de *Prodcut Backlog*. Após finalizado essa lista é feita uma reunião de

planejamento chamada de *Sprint Planning Meeting*, onde os itens do *Product Backlog* é priorizado pelo *Product Owner*. A equipe que tem o papel de dizer se esses itens poderão entrar no Sprint ou não levando em conta a velocidade e capacidade da mesma. Essa lista planejada para a iteração é chamada de *Sprint Backlog*.

Durante a Sprint a equipe é responsável por fazer uma reunião diária para ajudar manter a comunicação e organização do grupo com relação ao projeto que está sendo desenvolvido assim como também identificar qualquer problema para ser resolvido o mais rápido possível.

No final de cada *Sprint* é realizado um reunião chamada de *Sprint review* ou revisão da *Sprint* onde será apresentado as funcionalidades para *Product Owner*. Esse clico é repetido inúmeras vezes até que o *Product Backlog* seja completamente atendido (KNIBERG, 2007).

Sprint

O Scrum trabalha com desenvolvimento iterativo e incremental, onde cada iteração é chamada de *Sprint*. O tempo de cada *Sprint* varia de acordo com a complexidade do produto levando em média de 2 a 4 semanas no máximo, sempre com objetivo bem definido e claro para toda a equipe (KNIBERG, 2007).

Ao final de cada iteração ou *Sprint* a equipe disponibiliza um pedaço do sistema totalmente usável que pode ou não ser utilizado como um incremento do produto.

Conforme Martins (2007), as principais características do *Sprint* são:

- O *Scrum Team* se compromete com os itens do *Product Backlog* durante a reunião de planejamento da *Sprint*. E esses itens não poderão ser alterados até o fim da *Sprint*;
- Caso seja verificado que a *Sprint* seja inviável ou infactível, o *Scrum Master* poderá realizar o encerramento anormal da *Sprint* e iniciar uma nova reunião de planejamento;
- Caso o *Scrum Team* perceba que não conseguirá finalizar os itens do *Product Backlog* com os quais se comprometeu, ele poderá consultar o *Product Owner* sobre quais itens poderiam ser removidos. Se forem

muitos itens, que possam comprometer a *Sprint*, o *Scrum Master* pode encerrar anormalmente o mesmo;

- Caso o *Scrum Team* perceba que ela pode trabalhar em mais itens do que os selecionados, pode consultar o *Product Owner* para identificar mais alguns itens que possam ser adicionados na *Sprint*;
- Durante a *Sprint,* o *Scrum Team* pode procurar por ajuda externa, para aconselhamento, informação e suporte;
- O *Scrum Team* gerencia a si próprio. Nenhuma pessoa externa pode forçar um direcionamento, dar instruções ou aconselhamentos a menos que seja solicitada;
- O *Scrum Team* possui duas responsabilidades administrativas durante a *Sprint*: participar das reuniões diárias e manter o *Sprint Backlog* atualizado num diretório onde todos do projeto possam ter acesso. Novas tarefas podem ser adicionadas ao *Sprint Backlog* se for necessário, desde que as estimativas de quanto faltam sejam sempre atualizadas.

Time Boxing

O *Time Boxing* é um espaço de tempo definido para equipe ter foco no objetivo que precisa ser realizado, alguns exemplos são (YOSHIMA, 2007):

- Sprint com prazo de 2 a 4 semanas
- Reunião diárias de no máximo 15 minutos
- Planejamento da Sprint em no máximo 8 horas

Product Backlog

O conceito mais importante no Scrum é o *Backlog* do produto (*Product Backlog*). O *Product Backlog* contém uma lista de itens definidos e priorizados pelo *Product Owner*, que incluem os requisitos funcionais e não funcionais de um sistema. Os itens do *Product Backlog* são chamados de histórias e todos os envolvidos no projeto podem incluí-los, porém somente o *Product Owner* pode priorizá-las. Essa lista não precisa ser definida por completa no início do projeto, podendo ser complementada com novas funcionalidades à medida que aumenta o conhecimento no produto.

O *Product Backlog* é a lista geral de funcionalidades que devem ser implementadas no produto.Essa lista é feita com pouco detalhe no inicio do projeto e pode mudar a qualquer momento. Os itens mais importantes ficam no topo da lista (KNIBERG, 2007).

Outro item importante do Scrum é o *Backlog* de Impedimentos, uma lista que contém todos os itens que impedem o progresso do projeto e geralmente estão associados a riscos. Estes itens não possuem uma priorização, mas estão associados a algum item de *Backlog* do produto ou a tarefas do item. O *Scrum Master* deve ter controle sobre esses itens, possibilitando ao time desenvolver o *Backlog* do produto sem impedimentos (ABRAHAMSSON et al., 2002).

A Figura 7 exemplifica itens do Product Backlog que deverão ser priorizados estimados pelo Product Owner e pela equipe do projeto respectivamente.

ID	Nome	Importância	Estimativa	Observação
1	Planejar projetos com SCRUM			
2	Acompanhar projetos com SCRUM			
3	O que é SCRUM			
4	Ferramentas para SCRUM			
5	SCRUM X PMBOK			

Figura 7 - *Product Backlog.*
Fonte: PICHLER (2010).

Priorização do *Product Backlog* pelo *Product Owner*

Os itens do backlog do produto podem ser priorizados por maior valor, risco, prioridade e necessidade. Os itens mais importantes ficam no topo da lista e determinam as próximas atividades a serem realizadas e também os itens que

precisam de mais detalhes para que todos possam entender e estimar a próxima tarefa a ser realizada (KNIBERG, 2007).

Estimativa do *Product Backlog*

Existem várias técnicas de estimativas, uma delas é o *Planning Poker*, estimando o trabalho em horas ou tamanho (ABRAHAMSSON et al., 2002).

O *Planning Poker* é uma forma de estimativa em conjunto, podendo ser feita como um jogo.Todos os membros do time podem participar democraticamente assim chegando em um consenso da estimativa de cada item.

Com um conjunto de cartas normalmente utilizando a sequência numérica de Fibonacci, onde existe uma maior diferença entre cada número, facilitando a diferença da complexidade de cada item, sendo que utilizando a sequência numérica convencional de 1 a 10 a diferente entre os número é muito próxima assim itens com complexidade 1 e 2 seria praticamente iguais (KNIBERG, 2007).

O jogo se baseia em selecionar um item do *backlog* do produto que o time acredita que seja o de menor complexidade e assim associar ele ao menor número da sequência de cartas tendo assim como comparar o grau de complexidade dos outros itens. Assim começa o jogo com cada membro mostrando o valor das cartas para cada item.

Quando todos os participantes escolherem suas cartas, elas são exibidas e caso haja um consenso o número da carta é associada à tarefa e assim é definido a complexidade de cada item. Em caso de divergência, cada participante explica o motivo das suas escolhas e uma nova rodada de escolhas é feita levando em conta as novas considerações e assim até o time chegar em um consenso sobre a complexidade do item. As rodadas só terminam quando todos os itens são estimados. O *Scrum Master* é o principal responsável por organizar esse processo (ABRAHAMSSON et al., 2002).

Sprint Backlog

O *Sprint Backlog*, ou *backlog* da Sprint, é um conjunto de itens do *backlog* do produto selecionados para *Sprint*. Esses itens são normalmente os que estão no topo da lista com maior prioridade e maior detalhamento. Cabe a equipe determinar

a quantidade de itens do *Product Backlog* que serão trazidos para o *backlog* da *Sprint*, já que é ela quem irá se comprometer a implementá-los.

O time é responsável por manter o *Backlog* da *Sprint* organizado e atualizado com as tarefas que vão sendo finalizadas e quanto tempo o time acredita que será necessário para completar as tarefas pendentes. É interessante ter pelo menos três *Sprints* priorizados e estimados para que o time não fique ocioso na transição entre uma *Sprint* e outra.

Execução da *Sprint*

A *Sprint* é iniciada com uma reunião de planejamento com duração máxima de 8 horas dividida em duas etapas. Nessa reunião todos os participantes do projeto discutem os itens que serão inseridos na *Sprint* assim como é feito a estimativa de tempo necessária para entrega do *Sprint*. Após essa etapa se inicia o desenvolvimento até o tempo limite estabelecido para termino da *Sprint*. No último dia as tarefas marcadas como prontas são analisadas pelo *Product Owner* e uma reunião de retrospectiva é realizada, com intuito de aperfeiçoar o processo (YOSHIMA, 2007).

Planejamento da Sprint (*Sprint Planning Meeting*)

O planejamento da *Sprint* tem duração máxima de 8 horas e é separada em duas etapas de 4 horas cada. Na primeira etapa o time juntamente com o *Product Owner* analisa o *backlog* do produto que já está ordenado por ordem de prioridade para verificar o que é possível realizar dentro do próximo *Sprint*, na segunda etapa do planejamento o time discute como será realizado o trabalho e como as atividades serão revertidas para incremento do produto que está sendo realizado.

No final do planejamento a equipe deve ser capaz de explicar para todos envolvidos no projeto como pretender trabalhar para completo o objetivo da *Sprint* (YOSHIMA, 2007).

Burndown Chart

Segundo Scrum.org (2013), o gráfico de *burndown* representa as tarefas da *Sprint* que estão sendo realizadas e indica a quantidade de trabalho que ainda precisa ser realizado para atingir 100% do objetivo da *Sprint*. O responsável por atualizar esse gráfico é a própria equipe de desenvolvimento e deve ser atualizado

diariamente para facilitar o acompanhamento das tarefas e garantir que os itens serão entregues dentro do prazo estimado.

Reunião Diária (*Daily Meeting*)

A reunião diária é um evento de no máximo 15 minutos onde os integrantes da equipe devem responder a três perguntas básicas:

- O que foi completado desde a última reunião ?
- O que será feito até a próxima reunião?
- Existe algum impedimento atual para você desenvolver seu trabalho?

A equipe utiliza as reuniões diárias para sempre verificar se tudo está dentro do planejado e se nada está impedindo o trabalho dos membros da equipe.

O *Scrum Master* deve assegurar que esta reunião esteja acontecendo mas o responsável por conduzir a reunião é a própria equipe de desenvolvimento.

Reunião de Revisão (Sprint Review)

A reunião de revisão é realizada no final da *Sprint* para verificar os itens marcados como feitos e realizar adaptações no *backlog* do produto caso haja necessidade. Com duração máxima de 4 horas o time e as partes interessadas no projeto colaboram sobre o que foi feito na *Sprint* com intuito de melhorar algum processo a ser realizado nas próximas *Sprints*.

O resultado dessa reunião é um *backlog* do produto atualizado facilitando o inicio da próxima *Sprint* (YOSHIMA, 2007).

Reunião de Retrospectiva (*Sprint Retrospective*)

A reunião de retrospectiva é uma oportunidade para o time verificar o que foi positivo e o que foi negativo durante a *Sprint* passada, criando assim um plano a ser executado na próxima *Sprint*.

O Scrum deve encorajar o time a realizar mudanças dentro do processo do Scrum, otimizando o processo de desenvolvimento.Ao final dessa reunião o time deve ter identificado as melhorias e o que será aplicado já no próximo *Sprint.*

3 Materiais e Métodos

3.1 *World Wide Web* (Web)

A Web representa uma forma de divulgar informações por meio de um sistema de documentos ou páginas em hipermídia que são interligados e executados na Internet.

A Web funciona basicamente com dois tipos de programas: o cliente e o servidor. O cliente através de programas chamados de navegadores tem acesso a todo conteúdo publicado na Web,como páginas com textos, imagens, animações, filmes e sons. Os servidores armazenam as páginas e podem exercer alguns tipos de controle sobre o que o usuários podem acessar. São computadores potentes instalados em universidades, empresas ou órgãos do governo conectados 24 horas por dia na Internet, assim os servidores recebem as requisições através do navegadores e retornam o conteúdo a ser visualizado (MARCONDES, 2001).

A Figura 8 ilustra a explicação acima.

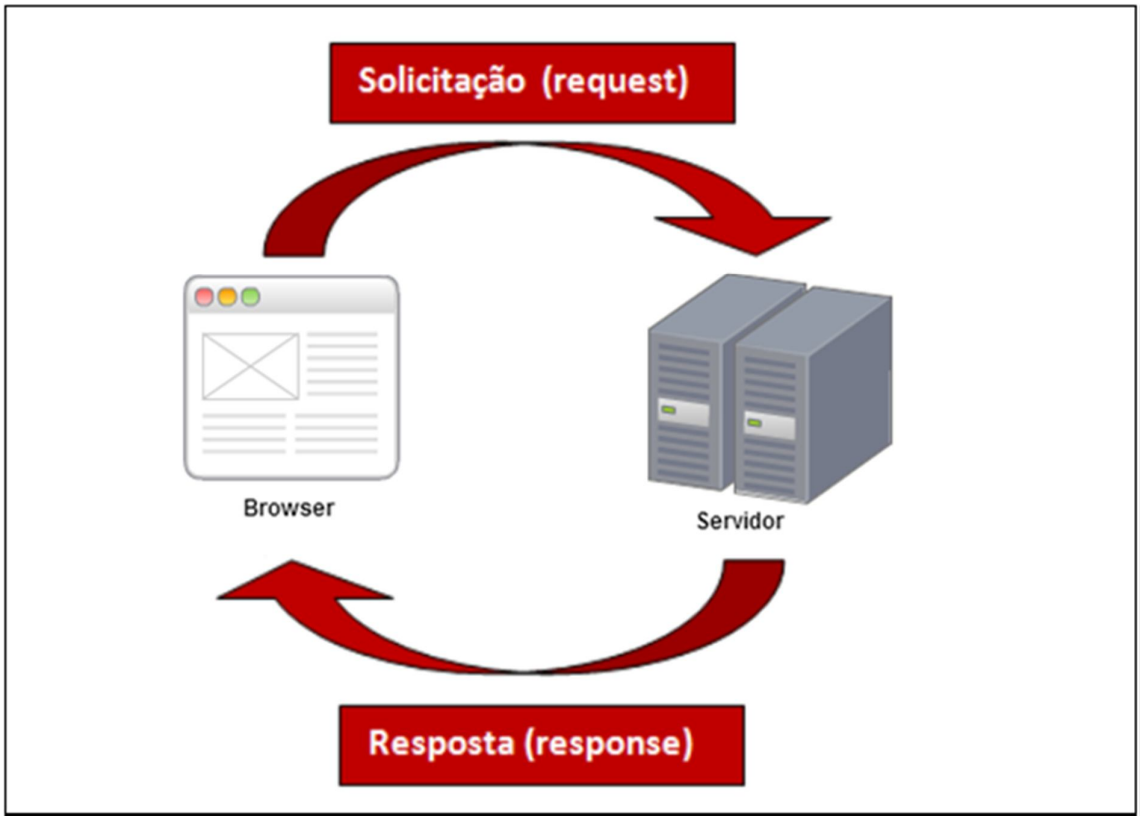

Figura 8 - Simulação de uma solicitação e resposta.
Fonte: Elaborado pelo autor.

A resposta pode ser uma página estática que significa que será o mesmo sempre que for requisitado e não possui nenhuma interação do usuário. Já as páginas com conteúdo dinâmico são construídas com a utilização de uma linguagem de programação que permite a interação dos usuários com as páginas Web.

3.2 Servidor Web - Apache

Atualmente em torno de 70% dos servidores Web do mundo utilizam o Apache, que é um dos servidores mais antigos, seguro e com diversos módulos, que adicionam suporte a vários recursos (MORIMOTO, 2006).

Para acessar os servidores é utilizado a URL (*Uniform resource Locator*) que é um endereço único na Web que indica o nome do arquivo, diretório, nome do servidor e o método pelo qual ele deve ser requisitado (MARCONDES, 2001).

Basicamente a função do servidor Web é traduzir uma URL em um nome de arquivo e em seguida devolver o conteúdo para o cliente, ou traduzir em um nome de programa que será executado dentro do servidor e após isso será devolvido o que produzido para o cliente que fez a requisição (LAURIE, 2013).

O servidor Apache é desenvolvido de forma colaborativa e é livremente disponível para implementação de melhorias, correções e alterações em seu código fonte. O projeto é desenvolvido em conjunto por um grupo de voluntários localizados em todo o mundo, através da Internet e da Web para se comunicar, planejar e desenvolver o servidor e sua respectiva documentação. Este projeto faz parte da *Apache Software Foundation* (APACHE.ORG, 2013).

O Apache se adapta a sites de todos os tamanhos e tipos, podendo ser executado uma única página pessoal ou várias páginas com milhões de visitantes regulares. O servidor também trabalha com conteúdo estático ou aplicações que geram conteúdo personalizado baseado na interação do visitante (LAURIE, 2013).

3.3 Linguagem de Programação PHP

PHP é uma linguagem utilizada para criar scripts do lado do servidor e foi projetada para criar páginas dinâmicas para Web. O PHP tem suporte a vários bancos de dados, como o MySQL e é largamente utilizando para criação de simples páginas pessoal como software complexos para grandes empresas.

Diferentemente do HTML que é processado pelo navegador do cliente o PHP é processado pelo servidor no qual o software está hospedado gerando um saída que é enviada aos navegadores. Uma grande vantagem do PHP é ser uma linguagem interpretada e não compilada como em outras linguagens facilitando e deixando mais ágil a realização de testes e modificações no sistema (LENGSTORF, 2009).

3.4 Framework PHP

Um framework para desenvolvimento de sistemas é um conjunto de códigos, classes, funções, técnicas e metodologias que tem como objetivo agilizar e facilitar o desenvolvimento de *softwares*.

As principais vantagens na utilização dessa tecnologia são: separação de apresentação e lógica, facilidade de geração de testes automatizados e geração de documentação. Utilizando a tecnologia de camadas MVC um acrônimo para *Model, view, Controller* (Modelo, Visão e Controlador) tem como objetivo separar todo desenvolvimento, seguindo padrões de projeto ou *design patterns* que é uma forma já testada e documentada para solucionar diversos problemas no desenvolvimento de sistemas.

Todas as requisições feitas pelo usuário chegam a *controller,* que manipula os dados utilizando o *model* que gerencia o acesso ao banco de dados e após isso chama a *view* para apresentação das informações processadas (MINETTO, 2007).

3.5 Banco de Dados

O banco de dados MySQL se tornou a base de dados mundial de código aberto mais popular por causa de seu alto desempenho, alta confiabilidade e facilidade de uso. É também a base de dados de escolha para uma nova geração de aplicações construídas sobre o LAMP (Linux, Apache, MySQL, PHP). Muitas das maiores organizações do mundo, incluindo Facebook, Google, Adobe, Alcatel Lucent e Zappos confiam no MySQL para poupar tempo e dinheiro, alimentando o seu volume de sites, sistemas críticos de negócios e pacotes de software.

O MySQL funciona em mais de 20 plataformas, incluindo Linux, Windows, Mac, Solaris, HP-UX, IBM AIX, dando-lhe o tipo de flexibilidade que coloca você no controle. (MYSQL.COM, 2013).

4 Modelagem da aplicação

O sistema de apoio ao gerenciamento de projetos utilizando a metodologia Scrum tem como principal objetivo auxiliar e facilitar a utilização do Scrum no desenvolvimento de *softwares*. Conforme descrito no capitulo anterior utilizando tecnologias para desenvolvimento Web o sistema pode ser utilizado por qualquer pessoa que tenha acesso a internet através de um computador.

4.1 Requisitos funcionais

Os requisitos funcionais descrevem o funcionamento do sistema e as principais funções disponibilizadas para os usuários. O sistema se divide em dois grandes módulos: Administração e Projetos.

4.1.1 Módulo de administração:

Esse módulo contém as funcionalidades de apoio ao módulo principal de projetos. Todas funcionalidades são cadastros básicos com funções de adicionar, editar e excluir os registros.

- Gerenciar Usuário: Cadastro geral dos usuários que utilizarão o sistema.
- Gerenciar Grupos: Cadastro de grupos que serão utilizados pelos cargos.
- Gerenciar Cargos: Cadastro dos papéis utilizados no Scrum e das permissões de acesso de cada papel.
- Gerenciar Empresas: Cadastro de empresas que serão utilizadas em cada projeto.

4.1.2 Módulo de projetos:

Esse é o principal módulo do sistema e contém todas as funcionalidades para o apoio a utilização da metodologia Scrum. Nesse módulo será possível gerenciar os projetos, gerenciar o *Product Backlog*, criar *Sprints*, cadastrar atas das reuniões diárias, cadastrar arquivos e documentos e gerenciar as horas utilizadas para o desenvolvimento.

- Cadastro de projetos: No cadastro do projeto é possível definir a prioridade, a empresa que pertence, os usuários responsáveis, prazo, data de inicio e data final.
- Tela de acompanhamento do projeto: Nessa tela é possível ter uma visão geral do projeto e contém informações de cadastro, da empresa, *Sprints*, cronograma, *Product Backlog*, atas e arquivos.
- *Product Backlog*: Essa funcionalidade permite o cadastro de atividades, que serão utilizadas para o desenvolvimento do projeto e serão selecionadas para criar um *Sprint*. Essa lista de atividades são ordenadas por prioridade, com a atividade mais importante no topo da lista.
- *Sprint*: O cadastro das *Sprints* determinam quais atividades serão desenvolvidas e o prazo de entrega para gerar um novo incremento do produto em desenvolvimento.
- Cronograma: Baseado nos prazos das *Sprints* é gerado um cronograma para facilitar o acompanhamento do projeto.
- Controle de horas: Dentro de cada atividade é possível controlar o tempo utilizado para desenvolvimento e gerar relatórios de horas por usuário e por projeto.
- Atas: Para controle das reuniões diárias é possível cadastrar em atas o que foi conversado e discutido.
- Arquivos: Essa funcionalidade facilita a centralização dos documentos e arquivos utilizado para o desenvolvimento do projeto.

4.2 Diagramas

A seguir será apresentado os diagramas utilizados para o desenvolvimento do *software:* i) diagrama de caso de uso; ii) diagrama de atividade; iii) diagrama de entidade relacionamento.

O diagrama de caso de uso é representação das funcionalidades que cada usuário tem acesso dentro do sistema. O administrador é responsável pelos itens básicos de funcionamento como cadastro de usuários, grupos, cargos e permissões. Ilustrado na figura 9.

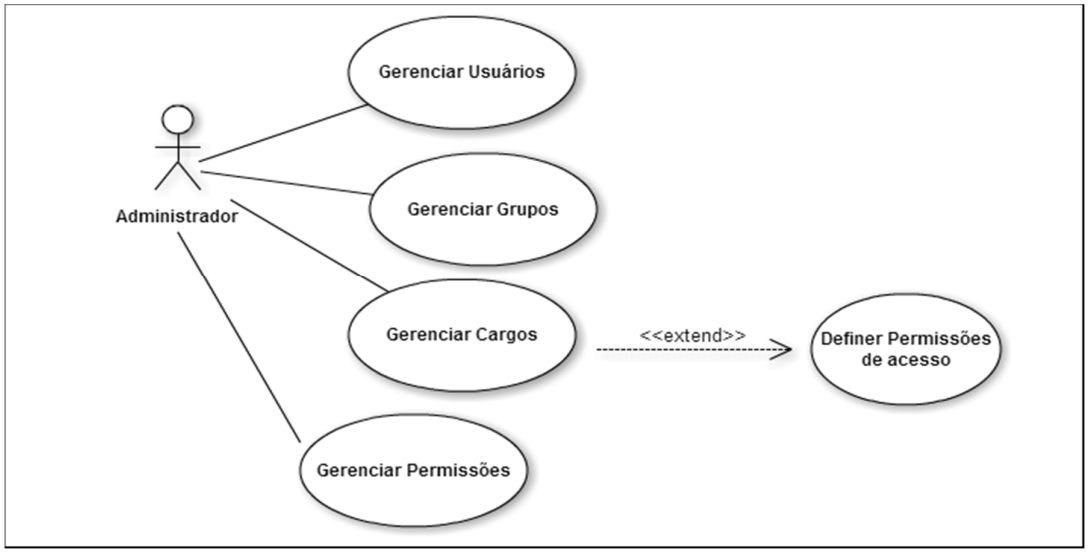

Figura 9 - Diagrama de caso de uso - Módulo Administração.
Fonte: Elaborado pelo autor (2013).

No diagrama ilustrado na figura 10, temos os membros da metodologia Scrum interagindo com a funcionalidades do sistema. Basicamente o *Product Owner* é responsável pelas atividades e prioridades, o time de desenvolvimento pela produção do incremento de *software* no final de cada *Sprint* e o *Scrum Master* é o responsável geral por manter todo processo funcionando.

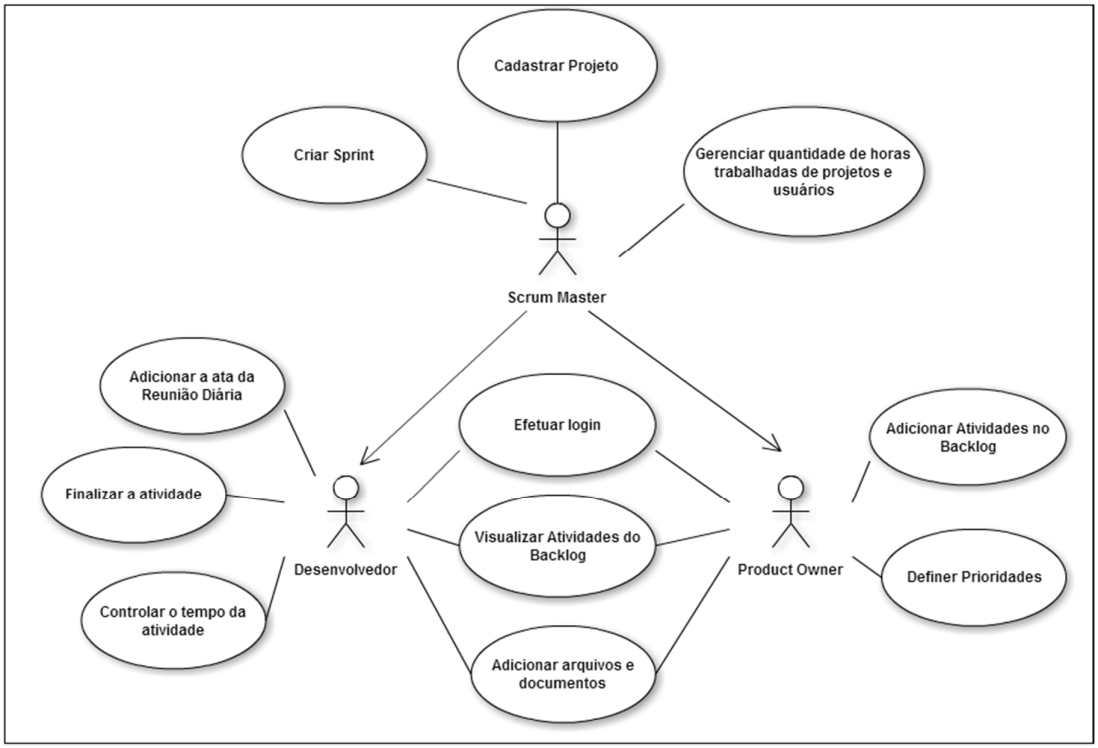

Figura 10 - Diagrama de caso de uso - Módulo Projetos.
Fonte: Elaborado pelo autor (2013).

A figura 11, apresenta as atividades realizadas pelo usuários do sistema, durante todo o desenvolvimento do projeto até que ele seja finalizado. O Scrum Master é responsável pela criação do projeto e das Sprints, já o *Product Owner* é responsável pelas atividades e priorização e o Scrum Team pelo desenvolvimento.

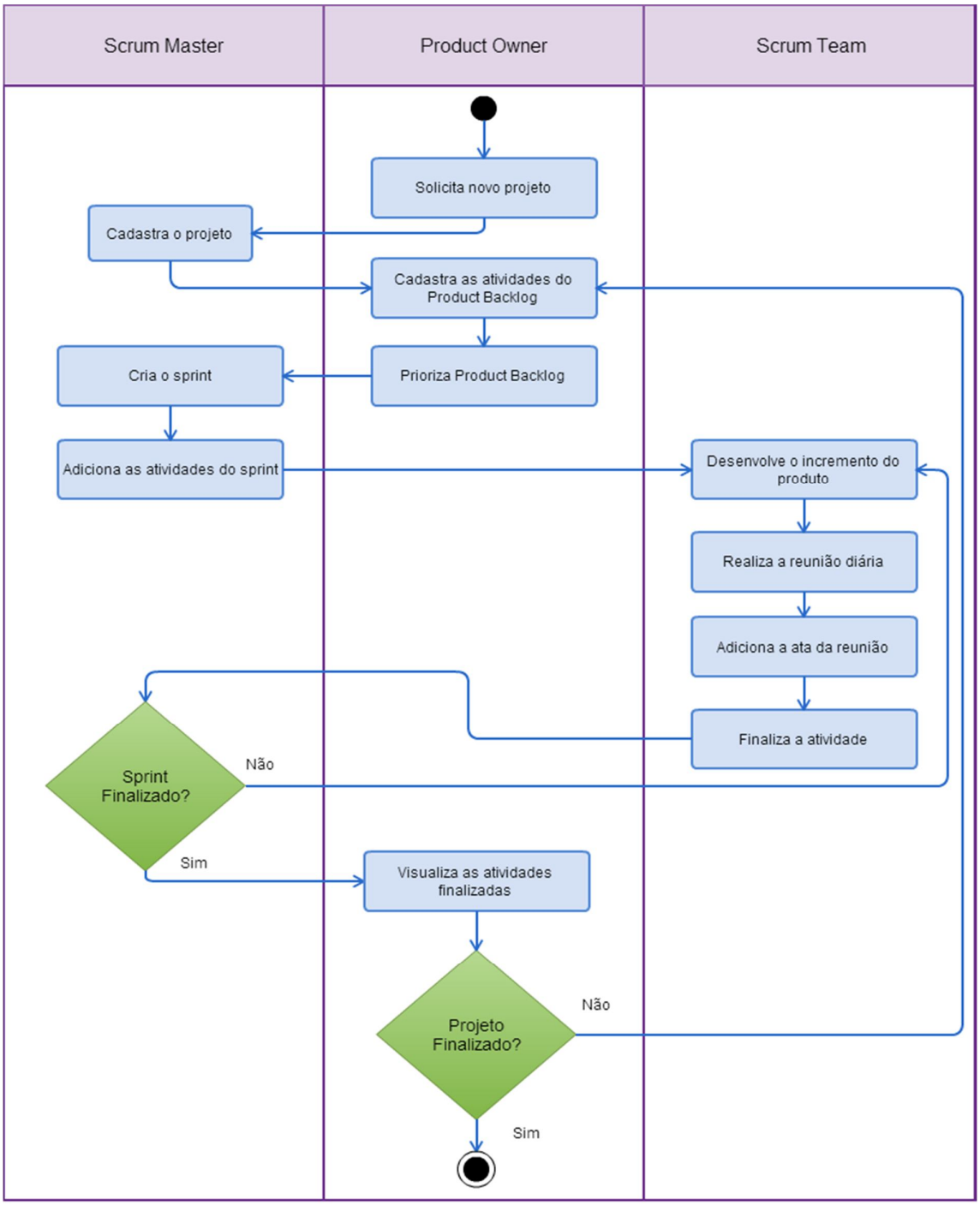

Figura 11 - Diagrama de atividade.
Fonte: Elaborado pelo autor (2013).

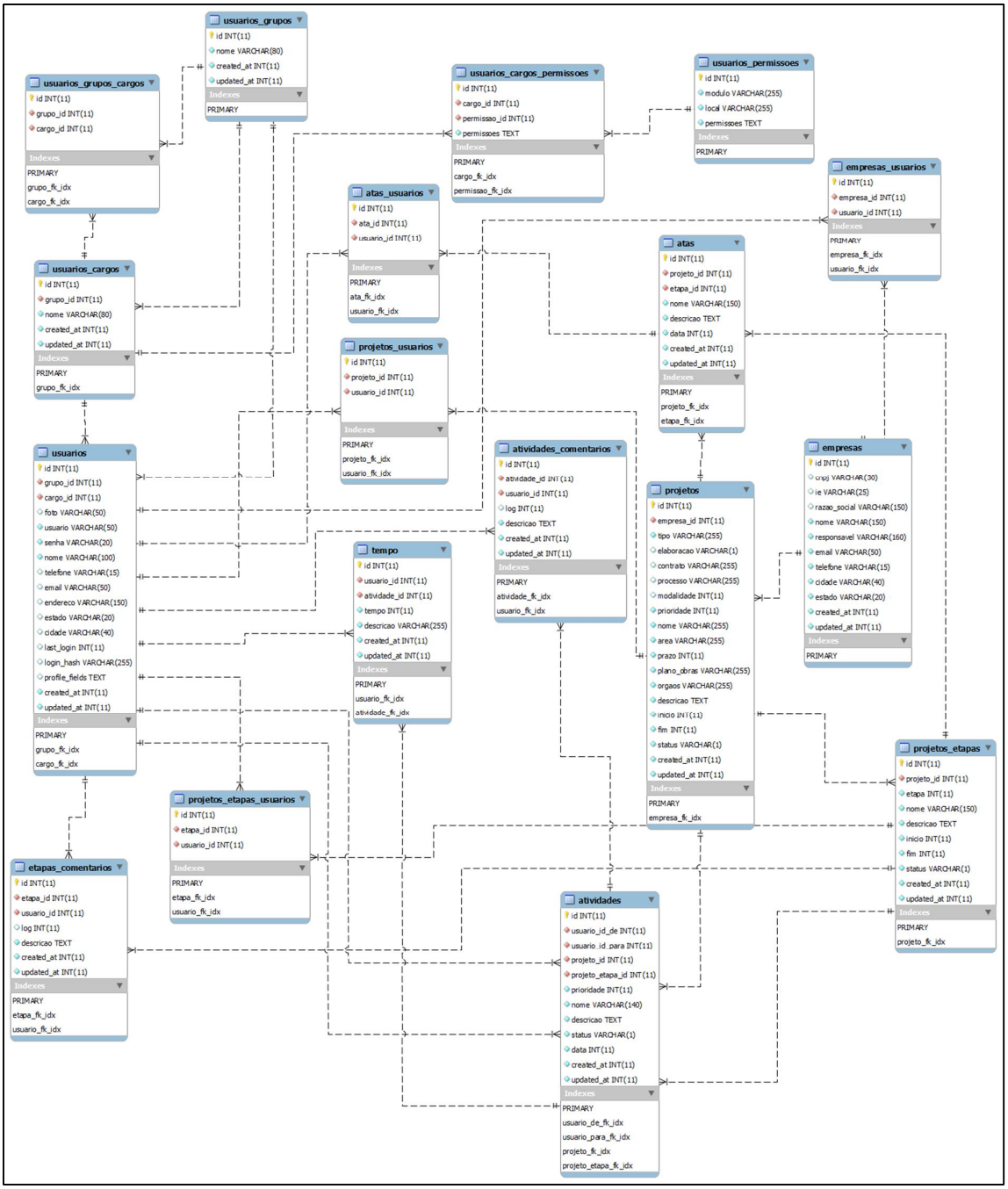

Figura 12 - Diagrama entidade relacionamento.
Fonte: Elaborado pelo autor (2013).

A ilustração da Figura 12 é a representação do banco de dados mostrando o relacionamento entre as entidades. As entidades são os itens que possuem características próprias e que se relacionam entre si. Essas podem ser pessoas, objetos, conceitos, eventos, etc..

No projeto em discussão as principais entidades são Projetos, Usuários e Atividades que vão se relacionar entre si e com outras entidades para criar a estrutura de armazenamento de informações do sistema.

5 Aplicação

Este capítulo descreve as principais funcionalidades, apresentando imagens das telas do sistema.

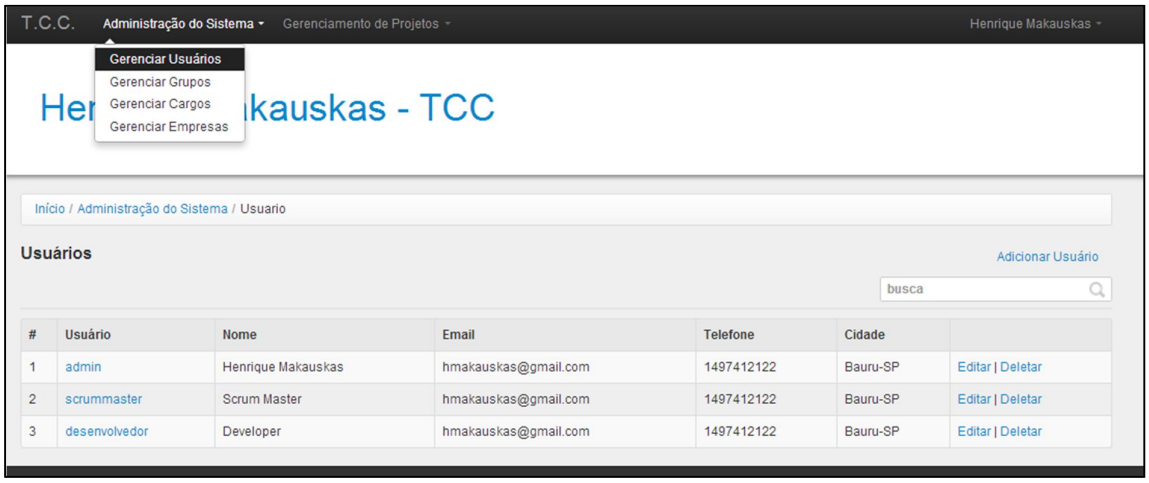

Figura 13 - Tela de administração.
Fonte: Elaborado pelo autor (2013).

A Figura 13, ilustra o módulo de administração do sistema, que abrange as funcionalidades de gerenciamento de usuários, cargos, grupos e empresas.

Nesse módulo também é possível definir diferentes permissões de acordo com o cargo do usuário, personalizando os níveis de acesso, conforme a Figura 14.

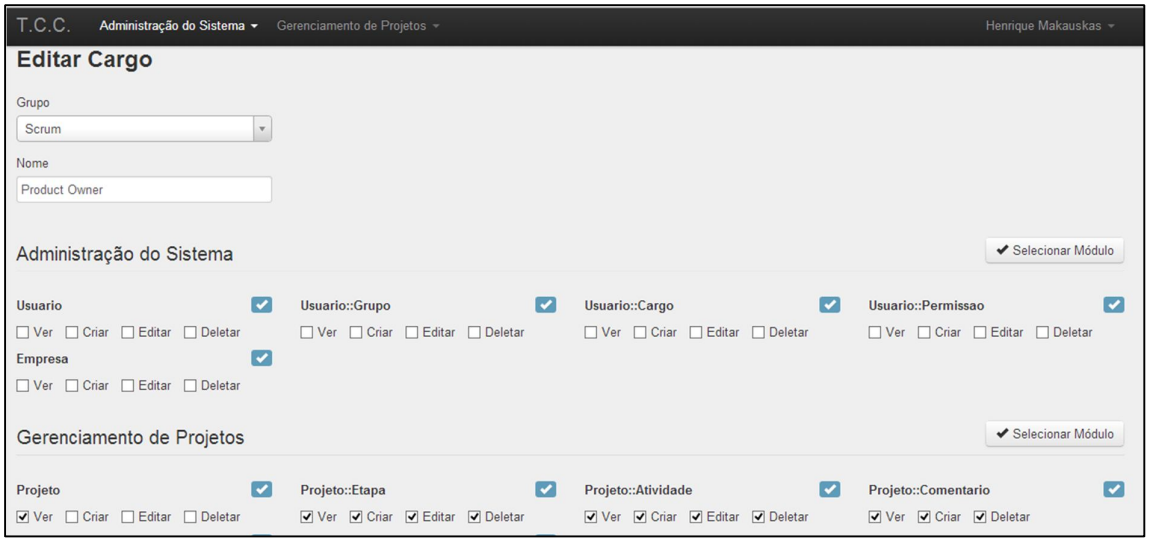

Figura 14 - Tela de permissões de acesso.
Fonte: Elaborado pelo autor (2013).

As próximas figuras apresentam o módulo de projetos, que contém as funcionalidades de apoio a metodologia Scrum.

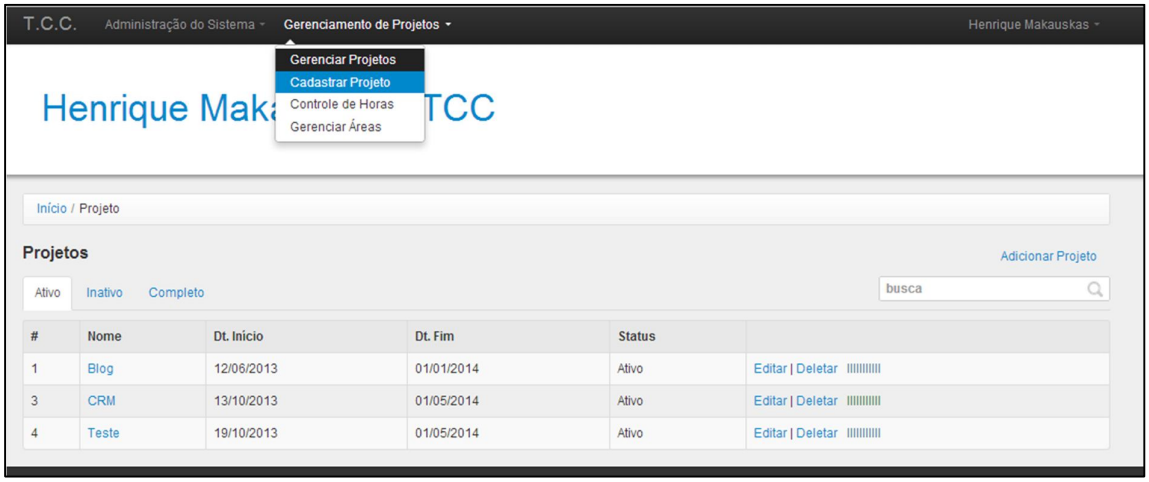

Figura 15 - Tela de gerenciamento dos projetos.
Fonte: Elaborado pelo autor (2013).

A Figura 15 apresenta a tela de gerenciamento de projetos, que contém as funcionalidade de criação, definição de prazos, definição dos responsáveis e prioridade.

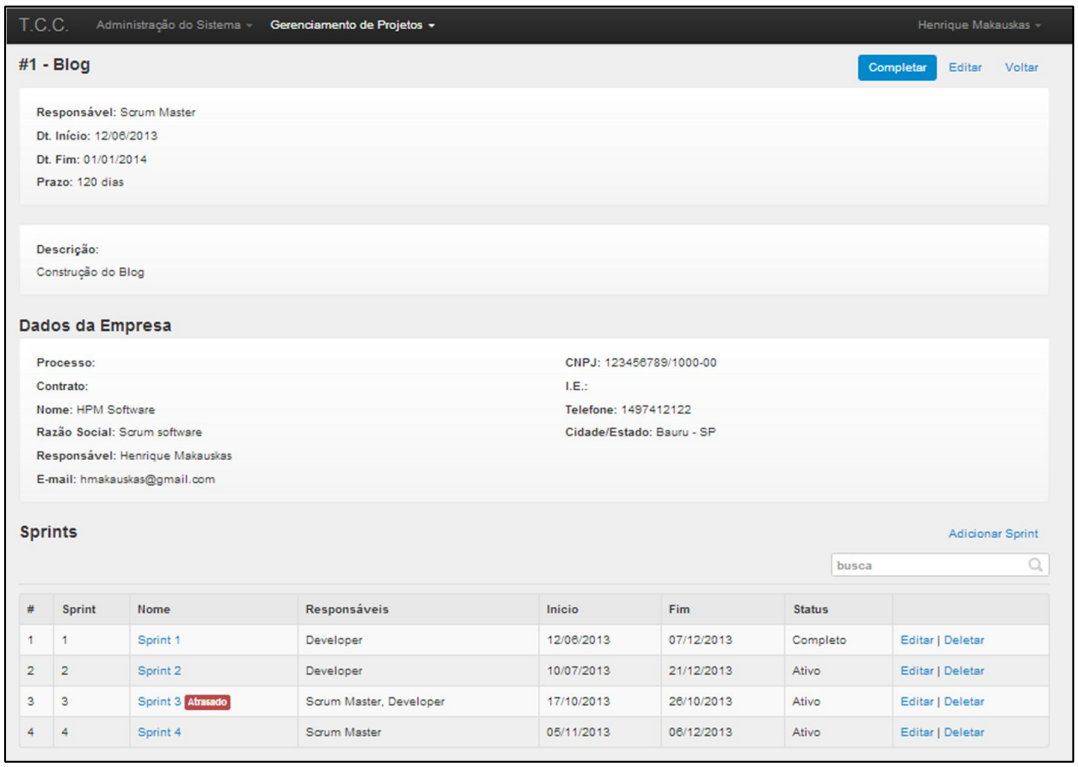

Figura 16 - Tela de acompanhamento do projeto (*Sprints*).
Fonte: Elaborado pelo autor (2013).

A tela de acompanhamento do projeto mostra as informações previamente cadastradas sobre o projeto como nome do projeto, responsáveis, prazo e dados da empresa.

Conforme a Figura 16 e 17, o cronograma do projeto é criado de acordo com cada *Sprint* adicionada no projeto facilitando a visão do prazo total do projeto. As atividades são adicionadas dentro do *Product Backlog* e posteriormente na reunião de planejamento da *Sprint* cada item é selecionado para desenvolvimento gerando ao fim de cada *Sprint* um incremento do sistema.

As reuniões diárias, de planejamento e retrospectiva podem ser adicionadas no formato de atas dentro do projeto para consulta, assim como arquivos, documentos e figuras com o objetivo de serem compartilhadas com a equipe.

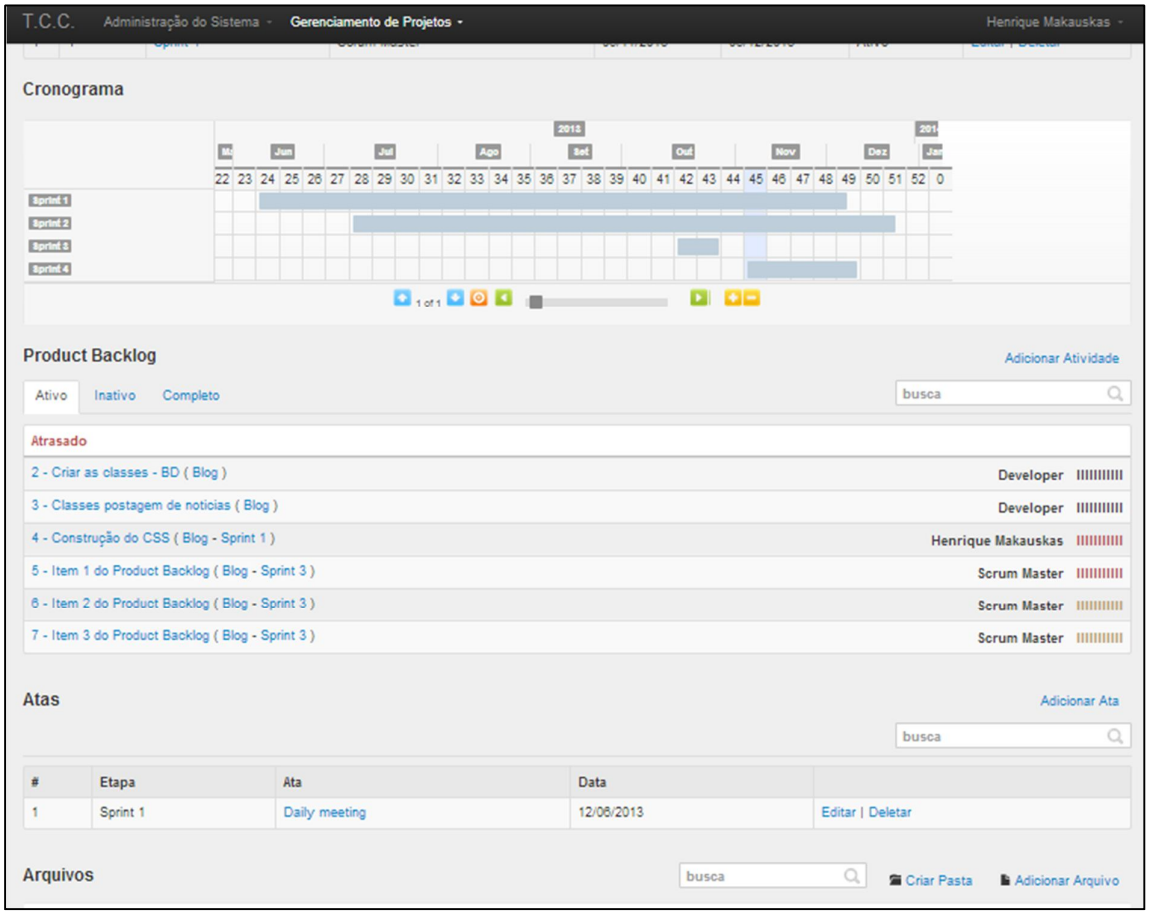

Figura 17 - Tela de acompanhamento do projeto (*Product Backlog*).
Fonte: Elaborado pelo autor (2013).

Todos os processos descritos acima visão facilitar o desenvolvimento de um projeto utilizando a metodologia aqui apresentada, englobando os principais

artefatos como criação do *Product Backlog*, *Sprints*, reuniões diárias e a fácil visualização do projeto como um todo.

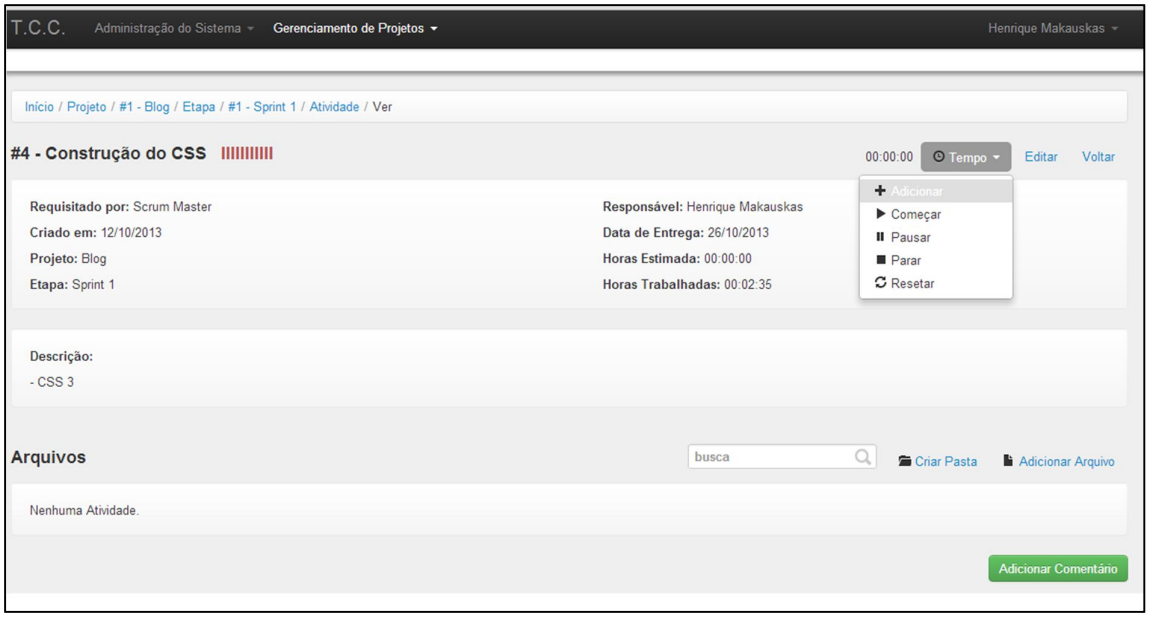

Figura 18 - Tela de detalhes da atividade.
Fonte: Elaborado pelo autor (2013).

Por fim a Figura 18 ilustra a tela de descrição da atividade que contém informações detalhadas de qual o objetivo da tarefa.

É possível também estimar e controlar o tempo de cada atividade com a funcionalidade de cronômetro, controlando assim a produtividade de cada desenvolvedor.

6 Testes e Resultados

Uma parte importante de todo desenvolvimento de software são os testes, que tem como principais objetivos encontrar falhas, definir melhorias e validar funcionalidades.

Para esse projeto foi utilizado a técnica chamada de teste de caixa-preta ou teste funcional. Essa técnica se baseia nas especificações das funcionalidades do projeto, e são realizadas a partir da visão do cliente, sem a necessidade de verificar códigos ou ter conhecimento técnico sobre o desenvolvimento.A análise consiste em criar casos de testes onde são definidas as entradas de dados no sistema, gerando assim saídas ou respostas que devem estar de acordo com as funcionalidades apresentadas (BADGETT; SANDLER; MYERS, 2011).

A técnica de teste de caixa-preta auxilia na verificação geral das funcionalidades do sistema, identificando requisitos incompletos ou inconsistentes, requisitos que não estão explicitamente indicados mas são funcionalidades consideradas como implícitas e teste de entrada de dados inválidos, sempre com na base visão do cliente (RAMESH; DESIKAN, 2006).

O teste funcional envolve dois passos principais: identificar as funções que o sistema deve realizar e criar os casos de teste para checar se essas funções estão funcionando corretamente.

6.1 Casos de teste

Os casos de testes foram desenvolvidos de acordo com as funcionalidades do sistema, visando encontrar falhas, melhorias e validar as funções existentes no sistema.

Para aplicar os testes e analisar os resultados, foi desenvolvido um questionário de avaliação do software (APÊNDICE A) contendo perguntas e casos de testes a serem realizados com o objetivo do avaliador analisar todas as partes do sistema. O questionário foi dividido em 4 seções:

- Seção 1 - Perfil do avaliador: Objetivo é coletar as informações do avaliador para identificar o perfil e experiência com computador e internet.

- Seção 2 - Módulo de administração:Objetivo é validar as funcionalidades do módulo de administração.
- Seção 3 - Módulo de projetos: Objetivo é validar as funcionalidades do módulo de projetos.
- Seção 4 - Visão Global do software: Objetivo é identificar a experiência do avaliador após realizar os casos de teste.

Visando facilitar a compreensão dos resultados utilizamos os seguintes termos para identificar os itens:

1. OK: Nenhum problema encontrado, funcionalidade de acordo com as especificações
2. FALHA: Algum problema foi encontrado e deve ser corrigido na próxima versão.
3. MELHORIA: Nenhum problema encontrado, mas foi identificado um ponto de melhoria no sistema para as próximas versões.

Os testes foram realizados por profissionais de diversas áreas que possuem diferentes experiências com computadores e sistemas para internet. As pessoas escolhidos tem o objetivo de abranger visões diferentes de acordo sua área de formação. Os profissionais formados na área de computação tendem a ter uma visão mais técnica sobre a análise, já os profissionais de outras áreas uma visão mais voltada ao usuário final de computador.

6.2 Resultados

A seguir é apresentado o resultado obtido com o questionário de avaliação do software (APÊNDICE A).Os dados foram consolidados e apresentados em formato de gráficos e tabelas para uma melhor visualização.

Os primeiros gráficos mostram o perfil de cada avaliador, seguindo para as tabelas de falhas e melhorias que foram encontradas durantes os casos de teste e, por fim, é apresentado o resultado de usabilidade e avaliação geral que foram obtidos após realização dos testes.

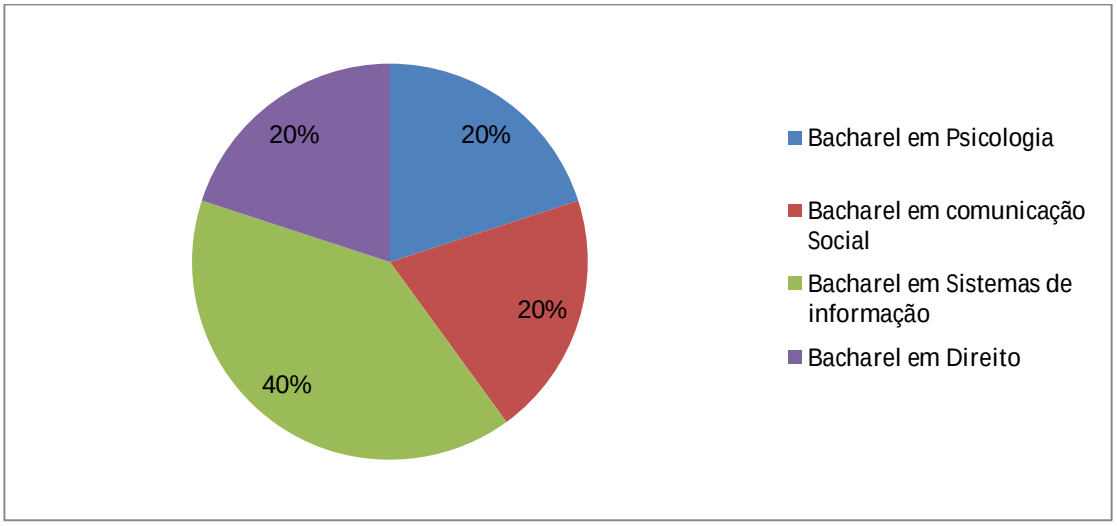

Figura 19 - Resultado - Qual é a sua formação acadêmica?
Fonte: Elaborado pelo autor (2013).

A Figura 19, apresenta a formação acadêmica de cada avaliador com o objetivo de identificar o conhecimento e a experiência adquirida de cada um. Nota-se que 40% são estudantes ou graduados no curso de Bacharel em Sistemas de informação pois são profissionais com conhecimento técnico na área de tecnologia da informação criando um equilíbrio entre a visão técnica e não técnica durante os testes.

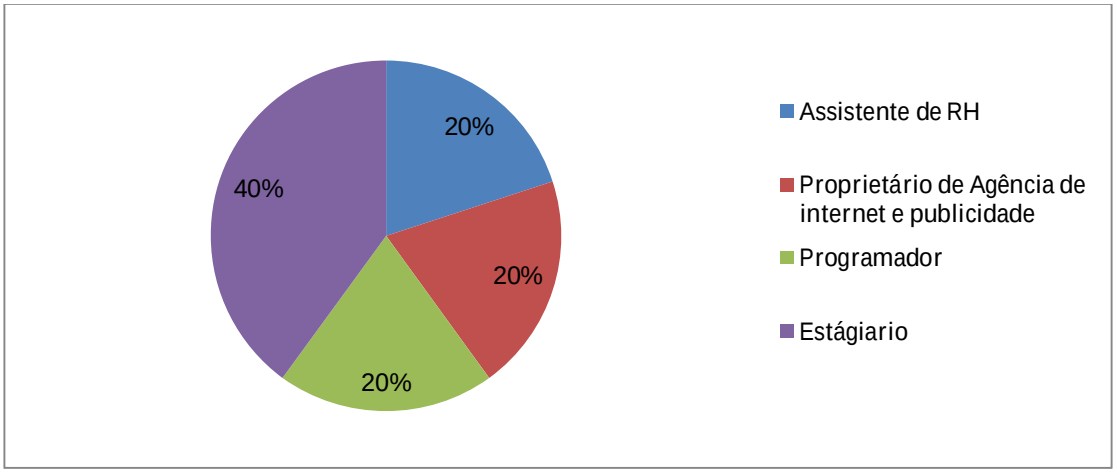

Figura 20 - Resultado - Qual é a sua profissão?
Fonte: Elaborado pelo autor (2013).

A Figura 20, apresenta a profissão de cada avaliador visando compreender a aplicabilidade do computador em seu dia a dia.

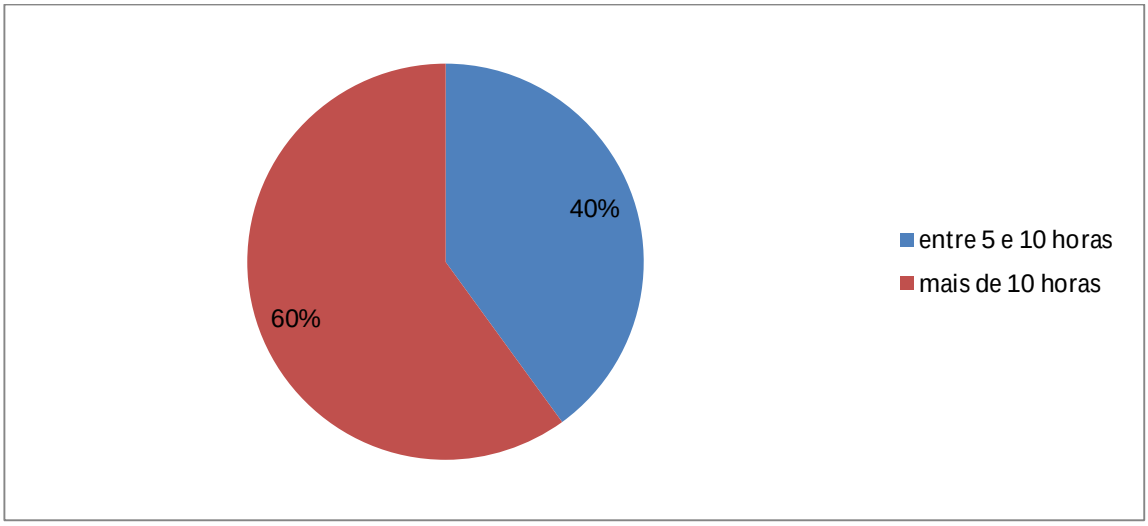

Figura 21 - Resultado - Quantas horas por semana, em média, você utiliza o computador?
Fonte: Elaborado pelo autor (2013).

Conforme apresentado na Figura 21, todos os avaliadores utilizam o computador a mais de 5 horas por semana, sendo que 60% utilizam mais que 10 horas. Assim é possível identificar que todos possuem um amplo conhecimento na utilização do computador e consequentemente em softwares em geral.

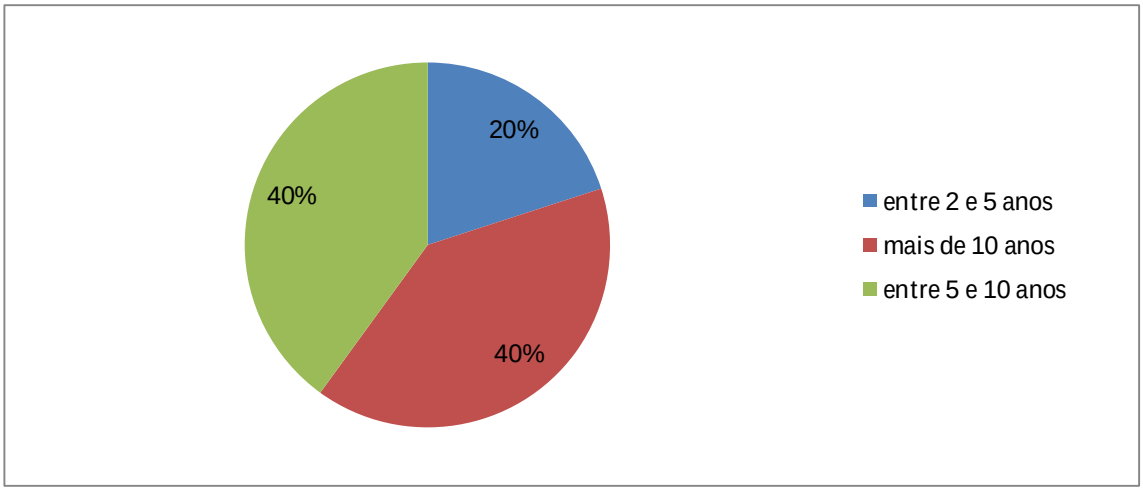

Figura 22 - Resultado - Há quanto tempo você navega na Internet?
Fonte: Elaborado pelo autor (2013).

Seguindo a mesma tendência do resultado apresentado acima, a Figura 22 mostra que todos os avaliadores tem experiência com internet, sendo que 40% já utilizam a mesma por mais de 10 anos. Mostrando que possuem habilidades para encontrar falhas e melhorias no software em questão.

Local do sistema	**Descrição**	***Status***
Cadastro de usuário	Não aparece os cargos	Corrigido na versão atual
Product Backlog	Não mostrar atividades que foram adicionadas em um *Sprint*	Corrigido na versão atual
Cargos e permissões	Não está funcionando a configuração de permissões de acesso	Corrigido na versão atual
Atividades	Não está funcionando o controle de horas gastas dentro do atividade	Corrigido na versão atual
Product Backlog	Está aparecendo *Sprints* de outros projetos	Corrigido na versão atual
Projetos	Página não encontrada no momento de visualizar um projeto	Corrigido na versão atual
Projetos	Não aparece os arquivos adicionados	Corrigido na versão atual

Figura 23 - Resultado - Falhas encontradas
Fonte: Elaborado pelo autor (2013).

A Figura 23, apresenta as falhas encontradas durante os casos de testes. A primeira coluna, local do sistema, representa o local onde as falhas foram encontradas dentro do sistema, a segunda coluna, de descrição é a explicação de como o erro aconteceu. Essas duas colunas tem o objetivo de identificar a falha para que o programador seja capaz de simular o erro e assim corrigi-lo.

Por fim, a terceira coluna *status* demonstra que todas as falhas encontradas foram corrigidas na versão atual do sistema.

Local do sistema	**Descrição**	***Status***
Geral	Não aparecer as opções do menu quando o usuário não tiver permissão de acesso	Aplicado na versão atual
Cadastros	Identificar com asterisco (*) campos obrigatórios	Aplicado na versão atual
Cadastro de empresas	Adicionar máscara de entrada de dados para campos de CNPJ, CPF	Será aplicado na próxima versão
Sprints	Selecionar itens do *Product Backlog* no momento de criação do *Sprint*	Será aplicado na próxima versão
Cadastro de empresa	O campo de CNPJ deveria ser obrigatório	Aplicado na versão atual
Cadastro de projetos	Não permitir o cadastro com datas retroativas	Será aplicado na próxima versão
Projetos	Adicionar o gráfico de *Burndown* para acompanhamento do *Sprint*	Será aplicado na próxima versão
Geral	Adicionar uma dica de funcionamento do campo quando o mesmo estiver selecionado	Será aplicado na próxima versão

Figura 24 - Resultado - Melhorias
Fonte: Elaborado pelo autor (2013).

A Figura 24, segue a mesma lógica apresentada acima para as duas primeiras colunas. Para essa tabela a terceira coluna *status* indica as melhorias que foram aplicadas na versão atual e as melhorias que serão aplicadas nas próximas versões do sistema.

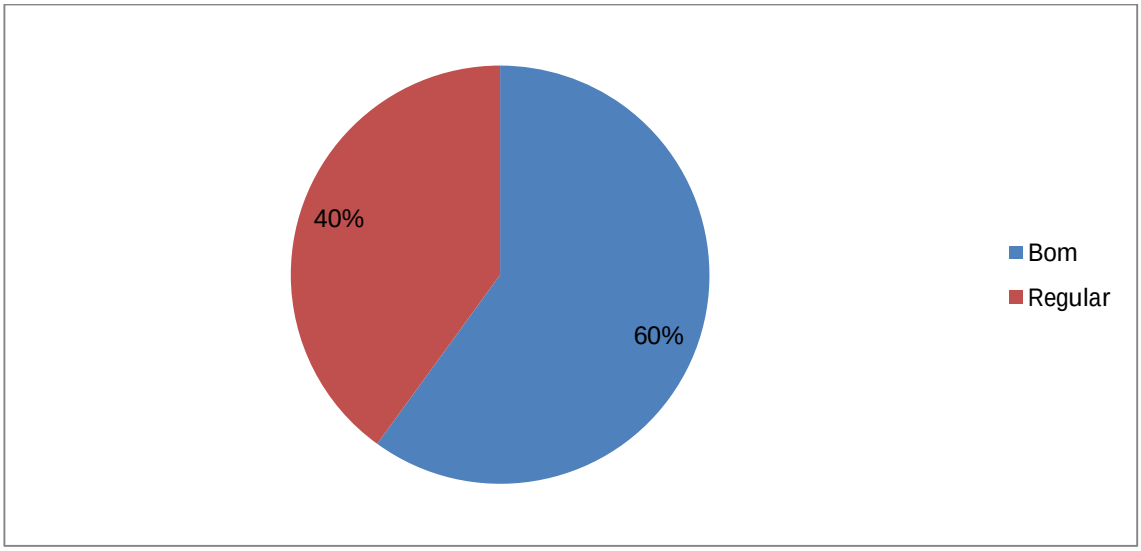

Figura 25 - Resultado - Usabilidade.
Fonte: Elaborado pelo autor (2013).

A Figura 23, ilustra a avaliação da usabilidade obtidos com os testes realizados. Nota-se que é necessário trabalhar para as próximas versões melhorias na questão de usabilidade.

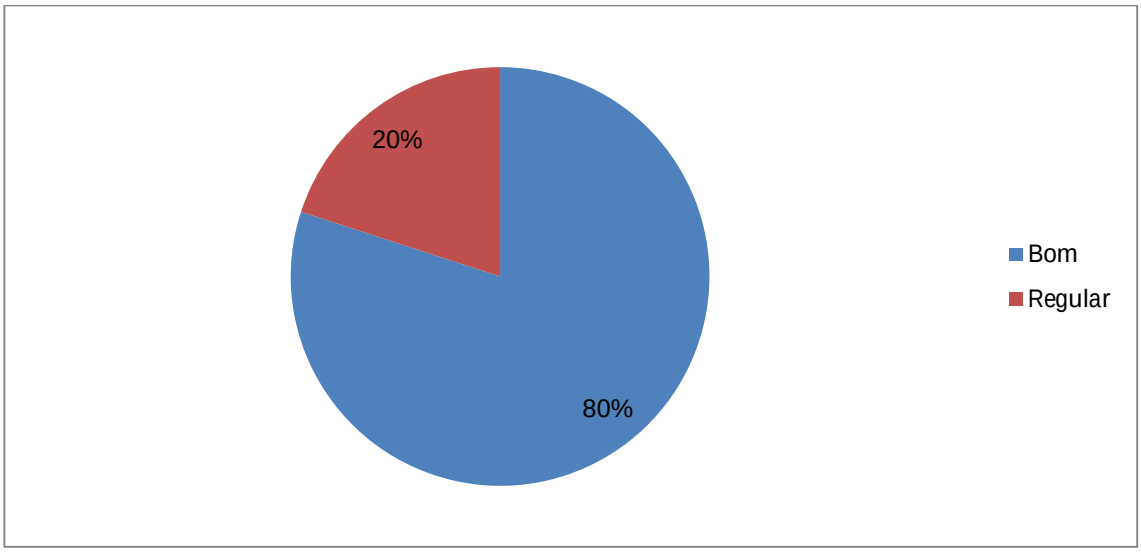

Figura 26- Resultado - Avaliação Geral.
Fonte: Elaborado pelo autor (2013).

De uma maneira geral, a Figura 24 indica que o software atende seus objetivos sendo que 80% dos avaliadores classificaram o sistema como bom. É claro que existe muito espaço ainda para melhorias e novas funcionalidades que podem agregar valor ao produto final.

CONSIDERAÇÕES FINAIS

O objetivo deste trabalho foi realizar um estudo sobre a metodologia ágil Scrum e desenvolver um software para o apoio a sua utilização.

Como base nos estudos apresentados foi possível identificar que hoje existe uma deficiência no desenvolvimento e na entrega dos projetos, c omo atrasos, softwares com má qualidade e até mesmo cancelamentos, causando grandes prejuízos financeiros.

As metodologias ágeis surgem para amenizar esses problemas trazendo boas práticas que devem ser utilizadas durante todo o desenvolvimento de um projeto. Estudos mostram grandes melhorias na utilização dessas metodologias e a quebra de um grande paradigma que são as mudanças durante o desenvolvimento de sistemas.

Neste projeto foi abordado sobre a metodologia ágil Scrum, que pesquisas mostram ser a mais utilizada nos dias atuais. Assim para facilitar a utilização do Scrum foi desenvolvido um software de apoio, assegurando que as boas práticas estão sendo utilizadas.

Os resultados obtidos através do questionário com casos de testes, apresentam uma boa avaliação do software e também indicam melhorias a serem aplicadas em futuras atualizações, agregando um maior valor na sua utilização.

Todos os dados aqui apresentados foram frutos de um processo de leitura e pesquisa sobre gerenciamento de projetos, metodologias ágeis e Scrum.

REFERÊNCIAS

PRIKLADNICKI, Rafael; ORTH, Afonso Inácio. **Planejamento & gerência de projetos.** Porto Alegre: EDIPUCRS, 2009.

MARTINS, José Carlos Cordeiro. **Técnicas Para Gerenciamento de Projetos de Software.** Rio de Janeiro: Brasport, 2007.

KNIBERG, H. **Scrum and XP from the trenches**: How we do Scrum. InfoQ, 2007. Disponível em <http://www.infoq.com/minibooks/scrum-xp-from-the-trenches>. Acesso em: 25 Ago. 2013.

AMBLER, Scott W. **Modelagem Ágil**: Práticas eficazes para a programação eXtrema e o processo unificado. São Paulo: Bookman, 2004.

ANDERSON, David James. **Agile Management for Software Engineering**: Applying the Theory of Constraints for Business Results. New Jersey: Prentice Hall Professional, 2004.

CARVALHO, Bernardo Vasconcelos de; MELLO, Carlos Henrique Pereira. **Aplicação do método ágil scrum no desenvolvimento de produtos de software em uma pequena empresa de base tecnológica**. Gest. Prod., São Carlos , v. 19, 2012. Disponível em <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0104-530X2012000300009&lng=en&nrm=iso>. Acesso em 15 Set. 2013.

LOESER, Anne. **Project Management and Scrum**: A Side by Side Comparison, 2006. Disponível em < http://hosteddocs.ittoolbox.com/AL12.06.06.pdf >. Acesso em 10 Out. 2013.

FOWLER, Martin; BECK, Kent. **Planning Extreme Programming.** Boston: Addison-Wesley Professional, 2001.

Guia do Scrum. **Scrum.org**, 2013. Disponível em <https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/2013/Scrum-Guide-Portuguese-BR.pdf >. Acesso em 29 Set. 2013.

SOMMERVILLE,Ian. **Engenharia de software.** 8. ed. São Paulo: Addison Wesley, 2008.

PRESSMAN, Roger S. **Engenharia de software**: Uma Abordagem Profissional. 6. ed. São Paulo: McGraw Hill Brasil, 2006.

KOONTZ, Harold; O'DONNEL, Cyril. **Principios de administração**: uma análise das funcões administrativas. São Paulo: Livraria Pioneira Editora, 1980.

BON, Jan Van. **ITIL, Volume 3**: A Pocket Guide. Amersfoort: Van Haren Publishing, 2007.

FREITAS, Carlos Augusto. **Certificação CAPM**: para membros de equipes e novos gerentes de projetos. Rio de Janeiro: Brasport, 2011.

YOSHIMA, Rodrigo. **Gerenciamento de Projetos com Scrum**. 2007. Disponível em < http://www.aspercom.com.br/ead/mod/resource/view.php?id=245>. Acesso em 12 Set. 2013.

SCHWABER, Ken. **Agile Project Management with Scrum.** O'Reilly Media, Inc., 2009.

SWEBOK. **Um Guide to the Software Engineering Body of Knowledge**. IEEE Computer Society, 2004.

PICHLER, Roman. **Agile Product Management with Scrum**: Creating Products that Customers Love. Addison-Wesley Professional, 2010.

MARCONDES, Christian Alfim. **Programando em HTML 4.0.** 6. ed. São Paulo: Editora Érica, 2001.

WELLING, Luke; THOMSON, Laura. **PHP e MySQL Desenvolvimento Web**. 3. ed. Rio de Janeiro: Elsevier, 2005.

Chaos Manifesto. **The Standish Group,** 2013 Disponível em <http://versionone.com/assets/img/files/ChaosManifesto2013.pdf>. Acesso em 10 Out. 2013.

MORIMOTO, Carlos E. **Redes e Servidores Linux.** 2. ed. Porto Alegre: Sul Editores, 2006.

LAURIE, Ben; LAURIE, Peter. **Apache**: The Definitive Guide. 3. ed. Sebastopol: O'Reilly Media, Inc., 2002.

MINETTO, Elton Luís. **Frameworks para Desenvolvimento em PHP**. Disponível em <http://www.martinsfontespaulista.com.br/anexos/produtos/capitulos/243418.pdf>. Acesso em 15 Out. 2013.

MENDES, João Ricardo Barroca; VALLE, André Bittencourt do; FABRA, Marcantonio. **Gerenciamento de projetos.** Rio de Janeiro: Editora FGV, 2009.

LENGSTORF, Jason. **PHP for Absolute Beginners.** New York: Apress, 2009.

3rd Annual Survey The State of Agile Development. **VersionOne,** 2008 Disponível em <http://www.versionone.com/pdf/3rdannualstateofagile_fulldatareport.pdf>. Acesso em 23 Set. 2013.

BADGETT, Tom; SANDLER, Corey; MYERS, Glenford J. **The Art of Software Testing.** 3. ed. New Jersey: John Wiley & Sons, 2011.

Apache HTTP Server Project. **Apache.org**, 2013. Disponível em <https http://httpd.apache.org/ABOUT_APACHE.html >. Acesso em 18 Out. 2013.

RAMESH, Gopalaswamy; DESIKAN Srinivasan. **Software Testing: Principles and Practice.** Delhi: Pearson Education India, 2006.

Manifesto para o desenvolvimento ágil de software. **Manifesto Ágil**. 2011. Disponível em < http://www.manifestoagil.com.br/>. Acesso em 19 Set. 2013.

ABRAHAMSSON, P. et al. **Agile software development methods**: review and analysis. Espoo: VTT Publications, 2002.

PMI. **Um Guia do Conhecimento em gerenciamento de projeto**: (Guia PMBOK) Quarta Edição. Project Management Institute, 2008.

About MySQL. **Mysql.com**, 2013. Disponível em <http://www.mysql.com/about/>. Acesso em 18 Out. 2013.

BOONEN, Harry; BRAND, Koen. **IT Governance Based on Cobit 4.1**: A Management Guide. Amersfoort: Van Haren Publishing, 2007.

APÊNDECIE - QUESTIONÁRIO DE AVALIAÇÃO DO SOFTWARE

Objetivo do Teste:	Garantir a funcionalidade apropriada do alvo do teste, incluindo navegação, entrada de dados, processamento, e apresentação final.
Técnica:	Executar cada caso de teste usando dados válidos e inválidos, para verificar o seguinte: ▪ Os resultados esperados ocorrem quando dados válidos são usados ▪ As mensagens de erro ou aviso apropriadas são exibidas quando dados inválidos são usados. ▪ Cada regra de negócio é aplicada apropriadamente
Critério de Finalização:	▪ Todos os testes planejados foram executados. ▪ Todos os defeitos identificados foram tratados.
Considerações Especiais:	Nós casos de teste utilizar as seguintes identificações: ▪ OK: Nenhum problema encontrado, funcionalidade de acordo com as especificações ▪ FALHA: Algum problema foi encontrado e deve ser corrigido na próxima versão. ▪ MELHORIA: Nenhum problema encontrado, mas foi identificado um ponto de melhoria no sistema para as próximas versões.

Seção 1: Perfil do avaliador

1 - Qual é a sua formação acadêmica ? _____

2 - Qual é a sua profissão? _____

3 - Quantas horas por semana, em média, você utiliza o computador?

( ) menos de 2 horas ( ) entre 5 e 10 horas

( ) entre 2 e 5 horas ( ) mais de 10 horas

4 - Há quanto tempo você navega na Internet?

( ) Menos de 2 anos ( ) entre 5 e 10 anos

( ) entre 2 e 5 anos ( ) mais de 10 ano

Seção 2: Módulo de administração

Caso 1 - Cadastro de usuários e grupos

Objetivo Geral: Validar a funcionalidade de cadastro de usuários e grupos.

ITEM	OBJETIVO	RESULTADO
Inclusão de usuários e grupos	Adicionar usuários e grupos	
Edição de usuários e grupos	Editar dados cadastrais dos usuários e grupos	
Remoção de usuários e grupos	Deletar usuários ou grupos	
Validação dos campos obrigatórios	Verificar se nenhum cadastro está em branco e validar se os campos obrigatórios são solicitados	
Validar entrada de dados	Não permitir o cadastro de dados inválidos	

Caso 2: Gerenciamento de cargos e permissões

Objetivo Geral: Validar a funcionalidade de cadastro de cargos e permissões. Para fins de teste utilize os cargos: *Scrum Master*, *Product Owner* e *Scrum team*, definindo as permissões de acesso para cada um.

ITEM	OBJETIVO	RESULTADO
Inclusão de cargos	Adicionar cargos	
Edição dos cargos	Editar dados de cada cargo	
Remoção dos cargos	Deletar cargos	
Validação dos campos obrigatórios	Não permitir o cadastro dos cargos sem os campos obrigatórios	
Permissões de acesso	Configurar as permissões	

Caso 3: Gerenciamento de empresas

Objetivo Geral: Validar a funcionalidade de cadastro de usuários e grupos.

ITEM	OBJETIVO	RESULTADO
Inclusão de empresas	Adicionar empresas	
Edição de empresas	Editar dados cadastrais	
Remoção de empresas	Deletar empresas do banco de dados	
Validação dos campos obrigatórios	Não permitir o cadastro de empresas sem os campos obrigatórios	
Validar entrada de dados	Não permitir o cadastro de dados inválidos	

Seção 3: Módulo de projetos

Caso 4: Gerenciamento de projetos

Objetivo Geral: Principal item do sistema, verificar as funcionalidades básicas de cadastro, analisando os campos específicos como prioridade, prazo, datas de começo e fim e responsáveis do projeto.

ITEM	OBJETIVO	RESULTADO
Inclusão de projetos	Adicionar projetos	
Edição de projetos	Editar dados cadastrais	
Remoção de projetos	Deletar projetos	
Validação dos campos obrigatórios	Não permitir o cadastro de projetos sem os campos obrigatórios	
Validar entrada de dados	Não permitir o cadastro de dados inválidos	
Prioridade do projeto	Definir a prioridade do projeto	
Responsáveis	Definir os responsáveis do projeto	
Prazo / Data de inicio e fim	Definir prazos e data de inicio e fim	

Caso 5: *Sprints*

Objetivo Geral: Os *Sprints* são períodos de tempo determinados onde é desenvolvido as atividades definidas de acordo com as prioridades. As atividades escolhidas para o desenvolvimento ficam dentro do *Sprint Backlog*.

ITEM	OBJETIVO	RESULTADO
Inclusão do *Sprint*	Adicionar um *Sprint*	
Edição do *Sprint*	Editar dados do *Sprint*	
Remoção do *Sprint*	Deletar um *Sprint*	
Validação dos campos obrigatórios	Não permitir o cadastro de um *Sprint* sem os campos obrigatórios	
Sprint Backlog	O *Sprint Backlog* deve conter as atividades a serem desenvolvidas	

Caso 6: *Product Backlog*

Objetivo Geral: Adicionar itens ao *Product Backlog* que deve estar ordenado por prioridade e com a possibilidade de adicionar cada atividade dentro de uma *Sprint*.

ITEM	OBJETIVO	RESULTADO
Inclusão de atividades	Adicionar usuários	
Edição de atividades	Editar dados cadastrais	
Remoção de atividades	Deletar usuários	
Validação dos campos obrigatórios	Não permitir o cadastro de atividades sem os campos obrigatórios	
Ordenação por prioridade	A lista deve estar ordenada por prioridade decrescente	
Inclusão da atividade em um *Sprint*	Uma atividade deve ser inserida em um *Sprint*	
Responsável pela atividade	Incluir um usuário responsável pelo desenvolvimento	

Caso 7: Atas e arquivos

Objetivo Geral: Funcionalidades de apoio ao processo de desenvolvimento. As atas são responsáveis pela as reuniões diárias, reuniões de planejamento de *Sprints* e qualquer outro item que deve ser documentado para consulta posterior, assim como os arquivos que tem como objetivo centralizar os documentos utilizados para o projeto.

ITEM	OBJETIVO	RESULTADO
Inclusão de atas e arquivos	Adicionar atas e arquivos	
Edição de atas	Editar dados cadastrados nas atas	
Remoção de atas e arquivos	Deletar atas e arquivos	
Validação dos campos obrigatórios	Não permitir o cadastro de atas sem os campos obrigatórios	
Criação de pastas para os arquivos	Adicionar os arquivos em pastas	

Caso 8: Tela de acompanhamento do projeto

Objetivo Geral: A tela de acompanhamento tem todas as informações pertinentes ao projetos que está sendo desenvolvido.

ITEM	OBJETIVO	RESULTADO
Informações do projeto	Mostrar informações do projeto	
Dados da empresa	Mostrar dados da empresa	
Sprints	Mostrar os *Sprints* por ordem crescente	
Cronograma	Utilizar o cronograma para ilustrar o intervalo de tempo dos *Sprints*	
Product Backlog	Não permitir o cadastro de dados inválidos	
Atas	Mostrar as atas cadastradas no projeto	
Arquivos	Mostrar os arquivos cadastrados no projeto	

Caso 9: Tela de acompanhamento da atividade e controle de horas

Objetivo Geral: Na tela de atividade fica a descrição do que ser feito e informações da data de entrega, horas estimadas e o controle de horas trabalhadas.

ITEM	OBJETIVO	RESULTADO
Informações sobre a atividade	Mostrar as informações necessárias para o desenvolvimento da atividade	
Controle de horas trabalhadas	Controlar o tempo gasto na atividade	

Seção 4: Visão Global do software

Caso 10: Usabilidade

Objetivo Geral: Avaliar a facilidade de utilização do sistema para conseguir atingir objetivos acima realizados.

( ) Ótimo ( ) Ruim

( ) Bom ( ) Muito ruim

( ) Regular

Caso 11: Avaliação geral

Objetivo Geral: Avaliar de uma forma global o software

( ) Ótimo ( ) Ruim

( ) Bom ( ) Muito ruim

( ) Regular

Assinatura: _____

Software Web para apoio do Gerenciamento de Projetos com Foco na Metodologia Ágil Scrum

Henrique Pedon Makauskas

Banca examinadora: Prof.ª Ms. Elaine Cecília Gatto, Prof.º Dr.º Elvio Gilberto da Silva, Prof.º Ricardo Saggioro.

Centro de Ciências Exatas e Sociais, Ciência da Computação, Universidade do Sagrado Coração, Bauru-SP, Brasil.

Abstract. *This article describes the development of software to support the project management with a focus on Scrum. Within this context is presenting a brief study on project management that in the last decade has been undergoing major changes and also a study on agile methodologies gaining new fans each year. Currently the Scrum methodology is the most widespread in the labor market. Scrum employs an iterative and incremental approach to project development.*

Resumo. *Este artigo descreve o desenvolvimento de um software para o apoio ao gerenciamento de projetos com foco na metodologia Scrum. Dentro desse contexto é apresentando um breve estudo sobre gerenciamentos de projetos que na última década vem passando por grandes modificações e também um estudo sobre metodologias ágeis que vem ganhando novos adeptos a cada ano. Atualmente o Scrum é a metodologia mais difundida no mercado de trabalho. O Scrum emprega uma abordagem iterativa e incremental para o desenvolvimento de projetos.*

1. Introdução

Nos últimos anos, com a evolução dos computadores, das técnicas e ferramentas, a produção de software confiável e entregue dentro dos prazos e custos estipulados ainda é improvável. Quando os projetos são realizados respeitando prazos de entrega e custos, a qualidade corre o risco de ser duvidosa, pois pode ter sido efetuada em cima de muita pressão, aumentando as possibilidades de erros [Soares, 2004; Ambler, 2004].

Um estudo realizado pelo *The Standish Group* em 2012, mostra que apenas 39% dos projetos tiveram sucesso, 18% foram cancelados e 43% tiveram algum tipo de problema. Dentro dos problemas mais encontrados podemos destacar: requisitos incompletos, falta de apoio dos usuários, expectativas fora da realidade, mudanças do escopo do projeto, falta de apoio dos negócios e falta de recurso.

Durante muito tempo o desenvolvimento de software seguiu o modelo clássico de cascata para o desenvolvimento de produtos. Posteriormente, investigou-se um novo método de trabalho mais flexível e com documentação reduzida, surgindo assim, as metodologias ágeis [Ambler, 2004].

Uma pesquisa feita pela VersionOne em 2008, apontam os seguintes benefícios: i) 89% das pessoas que utilizaram metodologias ágeis em seus projetos tiveram um ganho igual ou superior a 10% na produtividade do desenvolvimento; ii) 83% tiveram

uma redução do tempo de entregar superior a 10%; iii) 84% reduziram a quantidade de defeitos encontrados e; iv) 65% tiveram redução dos custos. Nessa mesma pesquisa é apontado que 49% das pessoas utilizam Scrum como metodologia ágil.

Os métodos ágeis atualmente utilizados para desenvolvimento de sistemas podem ser capazes de aumentar a satisfação do cliente, adicionando maior valor ao produto final, produzindo um software de alta qualidade e acelerando os prazos de desenvolvimento de projetos [Libardi, Barbosa, 2010].

O presente artigo abordará a metodologia ágil *Scrum* por ser atualmente a metodologia mais difundida no mercado de trabalho. De acordo com a *Scrum*.org (2013) o *Scrum* é um conjunto de boas práticas para gerenciamento de projetos complexos, fundamentado nas teorias de controle de processo, com o conhecimento baseado nas lições aprendidas.

Esse artigo está organizado da seguinte maneira. A seção 2 aborda sobre as metodologias ágeis. A seção 3 apresenta como funciona o Scrum. Na sequência a seção 4 apresenta um descritivo sobre aplicação e suas funcionalidades. A seção 5 aborda como os testes foram executados. Na seção 6 demonstra os resultados obtidos com a realização dos testes. E, por fim, na seção 7 são apresentadas as considerações finais.

2 Metodologias Ágeis

A maior característica das metodologias ágeis é que elas são adaptativas ao invés de serem preditivas. As metodologias ágeis têm como característica a adaptação de novos fatores durante o desenvolvimento do projeto, sem tentar analisar previamente tudo o que pode ou não acontecer no decorrer do desenvolvimento. Essa análise prévia é custosa e pode se tornar um problema quando for necessário fazer alterações nos planejamentos. O maior objetivo das metodologias ágeis de desenvolvimento de software é construir aplicações com menor tempo, maior produtividade e qualidade. Essas metodologias trabalham com constante *feedback*, o que permite adaptar rapidamente a eventuais mudanças nos requisitos. Entre várias novas metodologias ágeis destacam-se: *Scrum*, Extreme Programming ou XP, Desenvolvimento Lean e Kanban [Libardi, Barbosa, 201].

Entre os métodos ágeis destaca-se o *Scrum*, sendo o objetivo de estudos e do presente artigo, pois, de acordo com pesquisas é o método mais difundido hoje dentro do mercado atual.

3 Scrum

O *Scrum* é um método ágil que tem como principal objetivo entregar pequenas partes funcionais do sistema no menor tempo possível sempre levando em consideração o que é mais importante para negócio de uma forma iterativa e incremental [Schwaber, 2009].

O coração do *Scrum* é a iteração. A cada iteração a equipe analisa os requisitos, a tecnologia e suas habilidades e então se dividem para construir e entregar o melhor software possível, adaptando-se diariamente conforme apareçam complexidades, dificuldades e surpresas [Libardi, Barbosa, 2010].

Scrum implementa sua estrutura iterativa e incremental através de três papéis: o *Product Owner é* o dono do produto, pode ser o cliente ou um interessado do projeto, que representa os interesses de todos os envolvidos com o software; o *Scrum Team* é a equipe de desenvolvimento que consiste em profissionais que realizam o trabalho de

entregar uma versão usável, que potencialmente incrementa o produto ao final de cada etapa; e o *Scrum Master* é responsável pelo processo *Scrum*, por ensiná-lo a todas as pessoas envolvidas no projeto, por implementá-lo, fazendo dele uma cultura na organização [Libardi, Barbosa, 2010].

No *Scrum*, um projeto se inicia com uma visão simples do produto que será desenvolvido. O *Product Owner* transforma essa visão em uma lista de requisitos funcionais e não funcionais. Essa lista, chamada de *Product Backlog* é priorizada de uma forma que os itens que gerem maior valor ao produto tenham maior prioridade.

Todo o trabalho é feito em *Sprints* que são iterações de 2 a 4 semanas. Cada *Sprint* se inicia com uma reunião chamada *Sprint Planning Meeting* na qual o *Product Owner* e a equipe decidem o que será desenvolvido nesta *Sprint*. Na reunião são apresentados os itens de maior prioridade e o *Scrum Team* seleciona os itens que serão entregues ao final da *Sprint* dividindo-os em tarefas que compõem então o *Sprint Backlog*. Durante a *Sprint* a equipe é responsável por fazer uma reunião diária para ajudar manter a comunicação e organização do grupo com relação ao projeto que está sendo desenvolvido, assim como identificar qualquer problema para ser resolvido o mais rápido possível. No final de cada *Sprint* é realizada uma reunião chamada de *Sprint review* ou revisão da *Sprint* onde serão apresentadas as funcionalidades para o *Product Owner*. Esse ciclo é repetido inúmeras vezes até que o *Product Backlog* seja completamente atendido [Libardi, Barbosa, 2010].

4. Aplicação

O sistema de apoio ao gerenciamento de projetos utilizando a metodologia Scrum tem como principal objetivo auxiliar e facilitar a utilização do Scrum no desenvolvimento de softwares.

4.1 Requisitos funcionais

Os requisitos funcionais apresentam as principais funções disponibilizadas para os usuários. O sistema se divide em dois grandes módulos: Administração e Projetos.

- **Módulo de administração:** esse módulo contém as funcionalidades de apoio ao módulo de projetos, que são:
    1. Gerenciamento de usuários: Cadastro geral dos usuários
    2. Gerenciamento de grupos: Cadastro de grupos que serão utilizados pelos cargos.
    3. Gerenciamento de cargos e permissões: Cadastro dos papéis utilizados no Scrum e das permissões de acesso de cada um.
    4. Gerenciamento de empresas: Cadastro de empresas que serão utilizadas em cada projeto.

- **Módulo de projetos:** esse é o principal módulo do sistema e contém as funcionalidades para o apoio a utilização da metodologia Scrum, que são:
    1. Cadastro de projetos: No cadastro do projeto é possível definir a prioridade, a empresa que pertence, os usuários responsáveis, prazo, data de inicio e data final.

2. Tela de acompanhamento do projeto: Nessa tela é possível ter uma visão geral do projeto e contém informações de cadastro, da empresa, Sprints, cronograma, *Product Backlog*, atas e arquivos.
3. *Product Backlog*: Essa funcionalidade permite o cadastro de atividades, que serão utilizadas para o desenvolvimento do projeto e serão selecionadas para criar um *Sprint*. Essa lista de atividades é ordenada por prioridade, com a atividade mais importante no topo da lista.
4. *Sprints*: O cadastro das *Sprints* determinam quais atividades serão desenvolvidas e o prazo de entrega para gerar um novo incremento do produto em desenvolvimento.
5. Cronograma: Baseado nos prazos das *Sprints* é gerado um cronograma para facilitar o acompanhamento do projeto.
6. Controle de horas: Dentro de cada atividade é possível controlar o tempo utilizado para desenvolvimento e gerar relatórios de horas por usuário e por projeto.
7. Atas: Para controle das reuniões diárias é possível cadastrar em atas o que foi conversado e discutido
8. Arquivos: Essa funcionalidade facilita a centralização dos documentos e arquivos utilizados para o desenvolvimento do projeto.

Para uma melhor visualização a figura 1 ilustra as funcionalidades que o administrador utiliza dentro do sistema.

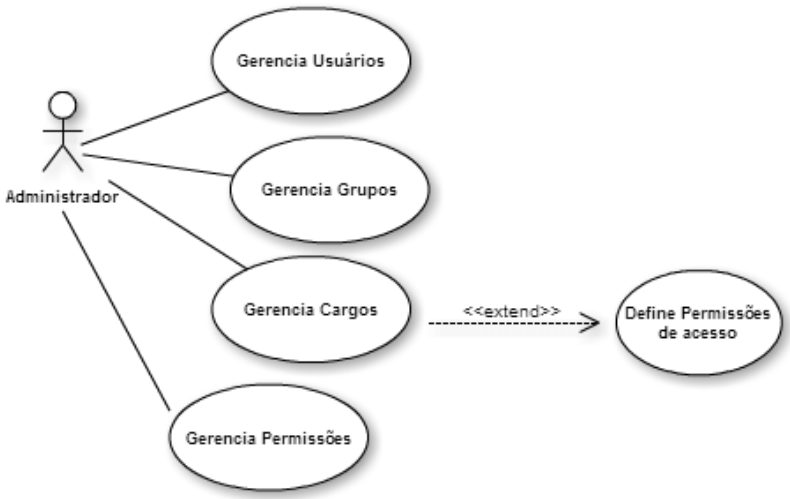

Figura 1 - Diagrama de caso de uso – Módulo de Administração. Fonte: Elaborado pelo autor.

A figura 2 ilustra os papéis da metodologia Scrum interagindo com as funcionalidades do sistema, onde o *Product Owner* é responsável pelas atividades e prioridades, o desenvolvedor ou *Scrum Team* pela produção do incremento de software no final de cada *Sprint* e o *Scrum Master* é responsável por manter essa estrutura funcionando.

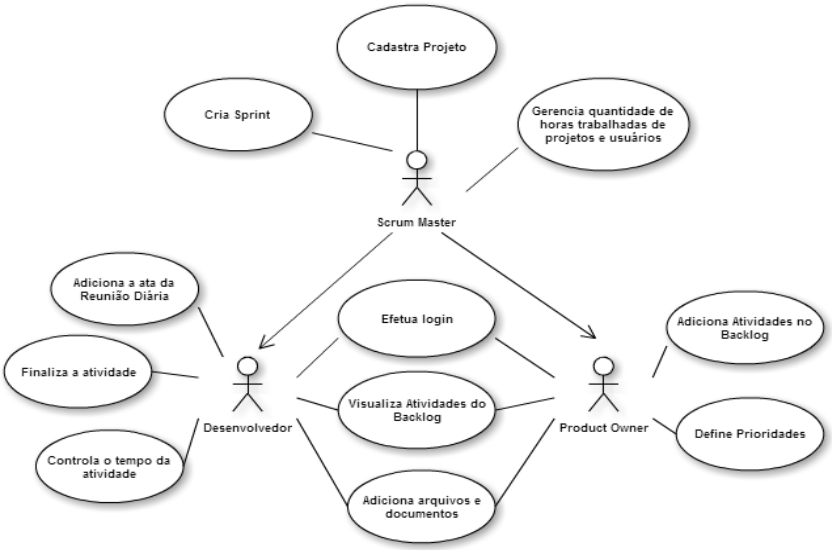

Figura 2 - Diagrama de caso de uso – Módulo de Projetos. Fonte: Elaborado pelo autor.

4.1 Apresentação do software de apoio

A seguir é apresentada a principal da tela do software onde ficam localizadas as principais funcionalidades relacionadas a metodologia Scrum.

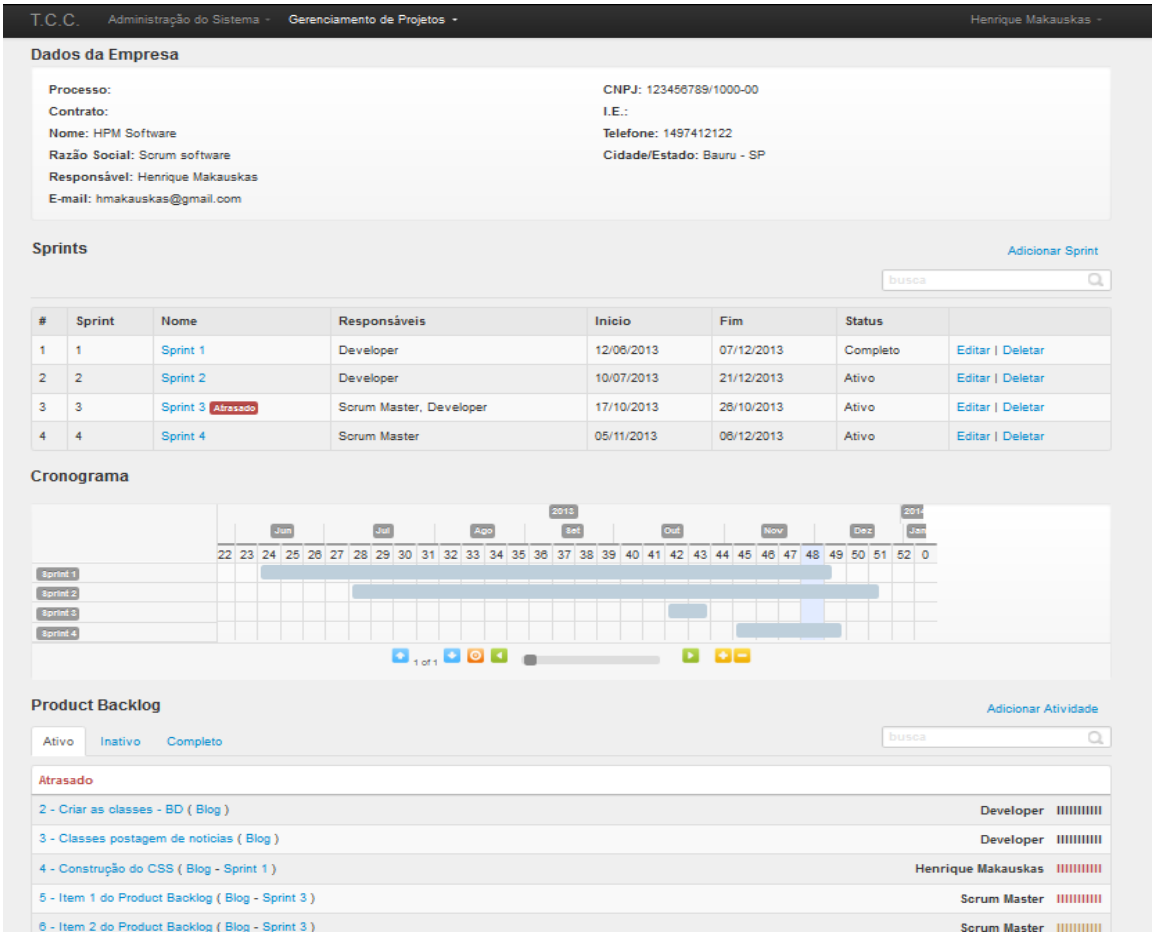

Figura 3 - Tela de acompanhamento do projeto. Fonte: Elaborado pelo autor.

A figura 3 é chamada dentro do sistema de tela de acompanhando do projeto, onde é possível ter uma visão geral de todo o projeto em uma única tela buscando atingir uma melhor usabilidade e facilitar o acesso à informação. Através dessa tela é possível ter acesso às informações do projeto como prazo de entrega, responsável geral, dados da empresa, gerenciar os *Sprints*, visualizar o cronograma, gerenciar o *Product Backlog*, adicionar atas das reuniões diárias ou de planejamento e por fim adicionar arquivos e documentos referentes ao projeto em desenvolvimento.

Outras telas do sistema são: as telas de cadastros de usuários onde é possível verificar usuários cadastrados; tela de cadastro dos grupos; tela de cadastros dos cargos e configuração das permissões de acesso; tela de gerenciamento das empresas; tela de acompanhamento do *Sprint*, onde é feito o gerenciamento do *Sprint Backlog* que são as atividades que foram retiradas do *Product Backlog;* tela de acompanhamento das atividades, que é onde ficam os detalhes do que precisa ser feito para atingir os objetivos propostos para a tarefa em questão e também onde é feito o controle de horas gastas através da ferramenta de monitoramento do tempo; por fim a tela de detalhes das atas registradas com as informações das reuniões diárias, de planejamento e retrospectiva que é realizada no final de cada *Sprint*.

5. Testes

Uma parte importante de todo desenvolvimento de software são os testes, que tem como principais objetivos encontrar falhas, definir melhorias e validar funcionalidades.

Para esse projeto foi utilizado à técnica chamada de teste de caixa-preta ou teste funcional. Essa técnica se baseia nas especificações das funcionalidades do projeto, e são realizadas a partir da visão do cliente, sem a necessidade de verificar códigos ou ter conhecimento técnico sobre o desenvolvimento. A análise consiste em criar casos de testes onde são definidas as entradas de dados no sistema, gerando assim saídas ou respostas que devem estar de acordo com as funcionalidades apresentadas [Badgett; Sandler; Myers, 2011].

Os casos de testes foram desenvolvidos de acordo com as funcionalidades do sistema, visando encontrar falhas, melhorias e validar as funções existentes no sistema.

Para aplicar os testes e analisar os resultados, foi desenvolvido um questionário de avaliação do software contendo perguntas e casos de testes a serem realizados com o objetivo de o avaliador analisar todas as partes do sistema.

Os testes foram realizados por profissionais de diversas áreas que possuem diferentes experiências com computadores e sistemas para internet. As pessoas escolhidas tem o objetivo de abranger visões diferentes de acordo sua área de formação. Os profissionais formados na área de computação tendem a ter uma visão mais técnica sobre a análise, já os profissionais de outras áreas uma visão mais voltada ao usuário final de computador.

6 Resultados

Os resultados obtidos foram muito importantes para a evolução do software, validação da usabilidade, identificação de falhas e sugestão de melhorias. Como o teste foi realizado por usuários técnicos e não técnicos, foi possível observar e comparar as dificuldades encontradas e identificar pontos de melhorias dentro do sistema para atender todos os tipos de usuários.

Os principais dados coletados através do formulário são apresentados abaixo:

- Perfil dos avaliadores: 40% eram formados em áreas relacionadas à computação, e 100% utilizam o computador pelos menos 5 horas na semana, sendo que 80% utiliza a internet há mais de cinco anos.
- Falhas: Foram encontradas sete falhas que foram corrigidas na versão atual do sistema. Alguns problemas relatados foram nas configurações de permissões de acesso, nos cadastros de usuários e *Product Backlog*, na visualização dos projetos e no controle de horas de cada atividade.
- Melhorias: Durante os testes foram destacadas sete melhorias das quais três foram aplicadas na versão atual que basicamente estão ligadas a questão de usabilidade e quatro melhorias que são mais complexas e demandariam alto tempo de desenvolvimento serão aplicadas nas próximas versões do sistema.
- Usabilidade: a avaliação da usabilidade obtida com os testes realizados mostrou que 60% apontaram como "bom" e 40% como "regular", indicando que melhorias futuras em relação à usabilidade precisam ser aplicadas.
- Avaliação Geral: em um contexto geral o software foi bem avaliado com 80% dos avaliadores classificando o sistema como "bom".

7. Considerações finais

Atualmente existe uma deficiência no desenvolvimento e na entrega dos projetos, como atrasos, softwares com má qualidade e até cancelamentos, causando grandes prejuízos financeiros.

A excessiva competitividade na área de desenvolvimento de software faz com que as empresas busquem sempre o aperfeiçoamento de seus serviços para driblarem a concorrência. Fatores importantes como prazo, qualidade e custo são os atuais diferenciais que podem ser atingidos utilizando-se metodologias ágeis de desenvolvimento, que surgem para amenizar esses problemas trazendo boas práticas que devem ser utilizadas durante todo o desenvolvimento de um projeto.

Dentre os métodos ágeis optou-se por estudar o método *Scrum* e concluiu-se que com a utilização do mesmo, houve facilitação no desenvolvimento de softwares e assegurou-se que as boas práticas estão sendo utilizadas para isso.

Referências

Ambler, S. W. (2004). "Modelagem Ágil: Práticas eficazes para a programação eXtrema e o processo unificado", São Paulo: Bookman.

Guia do Scrum. (2013) "Scrum.org".

Libardi, P. L. O., Barbosa V. (2010). "Métodos ágeis". Faculdade de Tecnologia, Universidade Estadual de Campinas.

Panchihak, M. L. (2013). "Benefícios do gerenciamento de projetos". Instituto Curitiba de Informática.

PMI. (2008) “Um Guia do Conhecimento em gerenciamento de projeto: (Guia PMBOK)” 4ª ed. Project Management Institute.

Prikladnicki, R., Orth, A. I. (2009). “Planejamento & gerência de projetos”. Porto Alegre: EDIPUCRS.

Rohden, R. B. (2013) “A importância da engenharia de software na computação nos dias atuais”. Departamento de Tecnologia, Universidade Regional do Noroeste do Estado do Rio Grande do Sul (UNIJUI).

Schwaber, K. (2009). “Agile project management with Scrum”. O'Reilly Media, Inc.

The Standish Group. (2013). “Chaos Manifesto”. Disponível em <http://versionone.com/assets/img/files/ChaosManifesto2013.pdf>. Acesso em 10 Out. 2013.

Soares, M. S. (2004). “Metodologias Ágeis Extreme Programming e Scrum para o Desenvolvimento de Software”, Revista Eletrônica de Sistemas de Informação, v.3, n.1.

Sommerville, I. (2008) “Engenharia de software”. 8. ed. São Paulo: Addison Wesley.

Marcondes, Christian Alfim. (2001) “Programando em HTML 4.0”. 6. ed. São Paulo: Editora Érica.

Morimoto, Carlos E. (2006) “Redes e Servidores Linux”. 2. ed. Porto Alegre: Sul Editores.

Laurie, Ben; Laurie, Peter. (2002) “Apache: The Definitive Guide”. 3. ed. Sebastopol: O'Reilly Media, Inc.

Lengstorf, Jason. (2009) “PHP for Absolute Beginners”. New York: Apress,.

Badgett, Tom; Sandler, Corey; Myers, Glenford J.(2011) “The Art of Software Testing.“ 3. ed. New Jersey: John Wiley & Sons.

VersionOne. (2008) “3rd Annual Survey The State of Agile Development”. Disponível em <http://www.versionone.com/pdf/3rdannualstateofagile_fulldatareport.pdf>. Acesso em 23 Set. 2013.