



UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

ESCOLA DE INFORMÁTICA APLICADA

Sistema de Apoio ao Planejamento de Projetos Baseado no Método de Simulação de  
Monte Carlo

Bruno Magalhães de Castro Dutra  
Carlos Eduardo de Andrade Paes Leme

**Orientador**

Alexandre Luis Correa

RIO DE JANEIRO, RJ – BRASIL

JUNHO DE 2014

Sistema de Apoio ao Planejamento de Projetos Baseado no Método de Simulação de  
Monte Carlo

Bruno Magalhães de Castro Dutra

Carlos Eduardo de Andrade Paes Leme

Projeto de Graduação apresentado à Escola de  
Informática Aplicada da Universidade Federal do  
Estado do Rio de Janeiro (UNIRIO) para obtenção do  
título de Bacharel em Sistemas de Informação.

Aprovada por:

---

Prof. Alexandre Luis Correa, D. Sc. (UNIRIO)

---

Prof. Gleison dos Santos Souza, D.Sc. (UNIRIO)

---

Prof. Márcio de Oliveira Barros, D.Sc. (UNIRIO)

RIO DE JANEIRO, RJ – BRASIL.

JUNHO DE 2014

## **Agradecimentos**

Bruno Magalhães de Castro Dutra:

À minha família, pelo apoio, incentivo e ajuda nessa jornada universitária.

À minha namorada pelo apoio, ajuda e cobrança nos momentos finais.

Aos amigos da faculdade, pela ajuda nos momentos complicados e pela descontração que fizeram com que esse caminho não fosse tão cansativo e trabalhoso.

Carlos Eduardo De Andrade Paes Leme:

A Deus, pelo apoio.

À minha família, pelo apoio, incentivo e paciência.

À minha namorada pelo apoio e compreensão.

Aos meus amigos por toda ajuda oferecida.

Aos professores do corpo docente do BSI pelos conhecimentos adquiridos e dedicação e, especialmente ao nosso orientador, o professor Alexandre Correa pela dedicação, atenção, paciência, ajuda e transmissão de conhecimentos.

## RESUMO

O gerenciamento de projetos utilizando métodos ágeis pode não ser algo simples ou trivial. Em particular, no método SCRUM, a utilização de estimativas para a realização de epics, histórias ou tarefas é crucial para o planejamento adequado dos releases e sprints realizados ao longo do projeto. Em muitos planejamentos de projetos, as estimativas de esforço para produção de determinado item ou realização de uma atividade são realizadas considerando apenas um valor, o que pode gerar problemas nas definições do conjunto de requisitos que podem ser desenvolvidos considerando o esforço estimado para cada requisito e a capacidade de produção da equipe. Este projeto busca reduzir esses problemas com o desenvolvimento de uma ferramenta que apoie o planejamento de releases e sprints de um projeto a partir de um gráfico de probabilidade acumulado gerado pela aplicação do método de simulação de Monte Carlo, considerando que a estimativa de esforço associada a cada item alocado a um Release ou Sprint é feita conforme uma distribuição de probabilidade (no escopo deste trabalho, será utilizado a distribuição triangular). Sendo assim, este trabalho permite a avaliação de diferentes cenários de alocação de itens de trabalho a um release ou sprint a partir da análise da probabilidade de conclusão desses itens no prazo esperado.

**Palavras-chave:** Gerenciamento de projetos, Simulação de Monte Carlo, Métodos Ágeis, Estimativas em Projeto.

## **ABSTRACT**

The project management using agile methods cannot be something simple or trivial. In particular, the Scrum method, the use of estimates for performing epics, stories or tasks is crucial for proper planning releases and sprints performed throughout the project. In many planning projects, estimates of effort to produce a particular item or performing an activity is performed considering only one value, which can create problems in the definitions of the set of requirements that can be developed based on the estimated effort for each requirement and production capacity of the team. This project seeks to reduce these problems by developing a tool that supports planning releases and sprints from one project using a cumulative probability plot, generated by applying the method of Monte Carlo simulation, whereas the estimate of effort associated with each item allocated to a Release or Sprint is taken as a probability distribution (in this work, the triangular distribution is used). Thus, this work allows the evaluation of different scenarios allocation of work items to a release or sprint from the analysis of the probability of finding these items in the expected time.

**Keywords:** Project Management, Monte Carlo Simulation, Agile Methods, Project Estimates.

## Índice

1. Introdução .....	1
1.1 Contexto .....	1
1.2 Motivação.....	1
1.3 Objetivos .....	2
1.4 Organização do texto.....	2
2.1 Métodos Ágeis .....	4
2.2 SCRUM.....	8
2.2.1 Papéis SCRUM .....	8
2.2.2 Dinâmica do Processo .....	9
2.2.3 Artefatos do SCRUM .....	11
2.3 SCRUM e Estimativas .....	13
2.4 Ferramentas Similares .....	17
2.5 Distribuição Triangular .....	18
2.6 Método de Simulação de Monte Carlo.....	19
2.6.1 Introdução.....	19
2.6.2 História .....	20
2.6.3 Aplicações e Áreas de utilização.....	20

2.6.4	Algoritmo de Monte Carlo .....	21
3	O Sistema .....	25
3.1	Visão Geral da Solução .....	25
3.2	Tecnologias Utilizadas .....	26
3.2.1	Linguagem de Programação C# .....	26
3.2.2	Google Charts.....	26
3.2.3	Microsoft Entity Framework.....	27
3.2.4	LINQ to Entities .....	28
3.2.5.	SQL SERVER 2008 .....	28
3.3	Modelo de Classes .....	28
3.4	Modelo de Casos de Uso.....	30
3.5	Módulos do Sistema.....	33
3.5.1	<i>Login</i> .....	33
3.5.2	Cadastro de Usuário .....	34
3.5.3	Tela Principal - Projeto .....	34
3.5.4	Cadastro de novo projeto - Projeto.....	35
3.5.5	Tela principal de um Projeto – Projeto.....	36
3.5.6	Cadastro de <i>Release</i> .....	37

3.5.7	Cadastro de <i>Sprint</i> .....	37
3.5.8	<i>Epic</i> .....	38
3.5.9	História .....	43
3.5.10	Tarefa .....	47
3.5.11	Simulação de Estimativas.....	52
4	Conclusão.....	58
4.1	Contribuições .....	58
4.2	Limitações .....	58
4.3	Trabalhos Futuros .....	59
4.4	Considerações Finais.....	59
Anexo I	– Arquivos de Importação .....	63

## Índice de Tabelas

Tabela 1 - Exemplo de <i>Product Backlog</i> .....	15
Tabela 2 - Exemplo de <i>Sprint Backlog</i> .....	16
Tabela 3 - Comparação de ferramentas similares.....	18
Tabela 4 - Exemplo de Atividades com as estimativas de 3 pontos.....	23
Tabela 5 - Exemplo de Alguns Cenários da Simulação de Monte Carlo .....	23

## Índice de Figuras

Figura 1 - Processo de iteração do SCRUM.....	10
Figura 2 – Fórmula para o Cálculo de Dias Ideais .....	14
Figura 3 - Algoritmo da Distribuição Triangular Utilizada.....	19
Figura 4 - Algoritmo de Monte Carlo.....	22
Figura 5 - Algoritmo de distribuição triangular.....	22
Figura 6 - Exemplo Gráfico de Probabilidade Acumulada .....	24
Figura 7 - Diagrama de Classes do Sistema .....	30
Figura 8 - Diagrama de Casos de Uso .....	31
Figura 9 - Tela de <i>Login</i> de Usuário.....	33
Figura 10 - Tela de Cadastro de Usuário.....	34
Figura 11 - Tela Principal de Projetos .....	35
Figura 12 - Tela de Cadastro de Novo Projeto .....	35
Figura 13 - Tela Principal do Projeto Escolhido .....	36
Figura 14 - Menu de Opções .....	36
Figura 15 – Tela de Cadastro Release .....	37
Figura 16 - Tela de Cadastro Sprint .....	38
Figura 17 - Tela Principal <i>Epic</i> .....	39

Figura 18 - Cadastro de Epic .....	40
Figura 19 - Arquivo Importação <i>Epic</i> .....	41
Figura 20 - Importar <i>Epic</i> .....	41
Figura 21 - Importação inválida .....	41
Figura 22 - Associação <i>Epic</i> – <i>Release</i> .....	42
Figura 23 - Editar Epic .....	43
Figura 24 - Tela Principal História .....	44
Figura 25 - Cadastro de História.....	45
Figura 26 - Arquivo Importação História .....	46
Figura 27 - Importar História.....	46
Figura 28 - Associação História – Sprint e/ou Epic .....	47
Figura 29 - Editar História.....	47
Figura 30 - Tela Principal Tarefa .....	48
Figura 31 - Cadastro de Tarefa .....	49
Figura 32 - Arquivo Importação Tarefa .....	50
Figura 33 - Importar Tarefa .....	50
Figura 34 - Associação Tarefa – Sprint e/ou História .....	51
Figura 35 - Editar Tarefa .....	51

Figura 36 - Tela Inicial de Projeto, destacando como realizar a simulação .....	52
Figura 37 - Tela Inicial de Simulação .....	52
Figura 38 - Tela de Simulação, com as <i>epics</i> listadas .....	53
Figura 39 - Tela da curva de probabilidade acumulada x tempo, destacando o ponto com a probabilidade para ser realizado no determinado tempo .....	54
Figura 40 - Tela de escolha de atividades temporais, destacando a data a partir da qual serão considerados.....	55
Figura 41 - Tela com as histórias associadas ao Sprint e sem associação.....	55
Figura 42 - Tela com curva de probabilidade acumulada x tempo para as Histórias do Sprint Selecionado.....	57

# 1. Introdução

## 1.1 Contexto

Os métodos ágeis surgiram como uma alternativa para lidar com a crescente complexidade dos sistemas e redução de prazos, de forma a possibilitar o desenvolvimento de projetos com maior produtividade e flexibilidade em ambientes propícios à sua utilização.

Nos últimos anos observa-se uma crescente adoção de métodos ágeis, em particular SCRUM e Kanban, na gestão, planejamento e execução de projetos, especialmente na área de desenvolvimento de software. Este trabalho se insere no contexto da atividade de planejamento de projetos que empregam o SCRUM como método de gestão.

## 1.2 Motivação

A motivação para a realização deste projeto é a crescente adoção de métodos ágeis no desenvolvimento de software que exigem o emprego de diversas práticas para a condução ideal do projeto. Uma delas consiste no planejamento e acompanhamento baseado em estimativas permanentemente monitoradas e revistas ao longo do projeto. Muitas organizações ainda têm dificuldade em cumprir os prazos originalmente acordados, e parte dessa dificuldade origina-se na qualidade das estimativas utilizadas no planejamento e definição dos compromissos. A definição do que pode ser feito em um determinado intervalo de tempo é uma tarefa muito complexa, em especial, devido à diversidade de variáveis e às incertezas que estão envolvidas. Dessa forma, a motivação para esse trabalho foi desenvolver uma solução que auxiliasse no planejamento de *releases* e *sprints* de um projeto baseado no método SCRUM de forma a considerar as incertezas envolvidas.

### **1.3 Objetivos**

O objetivo do trabalho foi construir um ferramental de apoio ao planejamento de escopo de cada iteração/fase de um projeto baseado em uma abordagem probabilística que visa melhor acomodar as diversas incertezas existentes em um projeto.

Com a adoção de um modelo probabilístico, utilizando estimativas de 3 pontos para cada item de trabalho, o modelo pode calcular, com certo grau de confiança, o esforço total necessário para a realização desses itens. Com isso, espera-se que a tendência de problemas com relação ao cronograma ou a atrasos seja atenuada, uma vez que é comum encontramos projetos onde as estimativas são definidas como um valor único, muitas vezes derivados de experiências pessoais anteriores ou de um suposto consenso da equipe. Porém o risco de super ou subdimensionar o esforço associado a um item não pode ser desprezado.

Como as metodologias ágeis são extremamente dinâmicas, as estimativas podem ser alteradas a qualquer momento. Essa é mais uma motivação para o uso de um modelo probabilístico, visto que ele se adapta melhor às condições de um ambiente de certa forma caótico. O modelo permite que, a partir das estimativas para um determinado conjunto de itens, o esforço total estimado possa ser recalculado em diferentes cenários, facilitando assim avaliar o grau de risco ao assumir determinados compromissos de entrega.

### **1.4 Organização do texto**

O presente trabalho está estruturado em capítulos e, além deste capítulo, está estruturado da seguinte forma:

- Capítulo 2: apresenta a fundamentação teórica para o entendimento dos termos empregados neste trabalho, uma introdução sobre os princípios ágeis e dos métodos ágeis, uma explicação das principais características do método ágil SCRUM. Além disso, o capítulo apresenta os conceitos fundamentais do método de simulação de Monte Carlo.
- Capítulo 3: apresenta o sistema produzido por este trabalho, as tecnologias utilizadas para implementação, suas funcionalidades e principais definições.

- Capítulo 4: reúne as considerações finais, assinala as contribuições do trabalho, algumas limitações e sugere possibilidades de trabalhos futuros.

## 2 Fundamentação Teórica

A crescente demanda por sistemas que satisfaçam as necessidades dos clientes, aliada à pressão por seu desenvolvimento em prazos cada vez menores em ambientes com certo grau de volatilidade nos requisitos são algumas motivações que justificam o crescente interesse e adoção de métodos ágeis de desenvolvimento. Este capítulo apresenta os princípios ágeis e enumera os principais métodos ágeis atualmente disponíveis, além de descrever as principais características de um deles, o SCRUM, que define o contexto de processo de desenvolvimento considerado para a implementação do software desenvolvimento no presente trabalho.

### 2.1 Métodos Ágeis

Segundo Vasco et. al. (2006), os métodos ágeis valorizam os aspectos humanos no desenvolvimento do projeto, promovendo um alto grau de interação entre os membros da equipe de desenvolvimento, bem como enfatizando a necessidade de se estabelecer um relacionamento de cooperação com o cliente. Isso se dá pelo emprego de práticas que visam promover o seguinte conjunto de valores publicado no Manifesto Ágil (Beck et. al., 2001):

- “Indivíduos e interações são mais importantes que processos e ferramentas”;
- “Software em funcionamento é mais importante que documentação abrangente”;
- “Colaboração com o cliente é mais importante que negociação de contratos”;
- “Responder a mudanças é mais importante que seguir um plano”;

Além desse conjunto de valores, o Manifesto Ágil propõe 12 princípios que os processos de desenvolvimento ágil devem seguir, denominados princípios ágeis (Beck et. al., 2001):

- *“Satisfazer o cliente através de entrega contínua e adiantada de software que agregue valor.”;*
- *“Mudanças nos requisitos são bem-vindas, ainda que ocorram tardiamente no desenvolvimento. Processos ágeis aproveitam as mudanças para oferecer vantagem competitiva para o cliente.”;*
- *“Entregar frequentemente software funcionando, em intervalos de poucas semanas a poucos meses, com preferência à menor escala de tempo.”;*
- *“Pessoas do negócio e desenvolvedores devem trabalhar em conjunto, diariamente, ao longo de todo o projeto.”;*
- *“Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o apoio necessário, e confie que eles concluirão o trabalho.”;*
- *“O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversas face-a-face.”;*
- *“Software funcionando é a principal medida de progresso.”;*
- *“Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante por um tempo indefinido.”;*
- *“Atenção contínua à excelência técnica e ao bom projeto técnico propicia mais agilidade.”;*
- *“Simplicidade - a arte de maximizar a quantidade de trabalho não realizado - é essencial.”;*
- *“As melhores arquiteturas, requisitos e projetos detalhados emergem de equipes auto-organizáveis.”;*
- *“Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e, então, refina e ajusta seu comportamento de forma apropriada.”;*

Esses métodos propõem uma abordagem de desenvolvimento que reduz investimentos em documentação excessiva e burocrática, enfatizando não apenas a interação entre as pessoas, mas também as atividades que efetivamente trazem valor e contribuem para a produção de software com qualidade [Beck et. al. 2001].

Dentre os métodos ágeis atualmente existentes, os mais conhecidos são:

- *Adaptive Software Development* (ASD) – consiste em um método que tenta trazer uma nova maneira de abordar o desenvolvimento de software em uma organização, promovendo um paradigma adaptativo, oferecendo soluções para o desenvolvimento de grandes e complexos sistemas. O método incentiva o desenvolvimento iterativo e incremental, com prototipagem constante [Highsmith,2000];
- *Agile Modeling* (AM) - é uma abordagem para a realização de atividades de modelagem, que tenta adequar as práticas de modelagem às bases da filosofia ágil. O foco principal em AM é incentivar os desenvolvedores a produzirem modelos avançados o suficiente para servir de apoio às necessidades de projeto e documentação. O objetivo é manter a quantidade de modelos e de documentação o mais baixo possível. Questões culturais são abordadas, descrevendo formas de incentivar a comunicação, a organização das estruturas de equipe e as formas de trabalho [Ambler,2002];
- *Crystal Family* - define uma série de métodos, a partir dos quais, o método mais adequado para cada projeto individual deve ser selecionado. Além dos métodos, a abordagem *Crystal* apresenta princípios para a adaptação desses métodos a diferentes circunstâncias de projeto. Cada método do *Crystal Family* é marcado com uma cor que indica o 'peso' do método. *Crystal* sugere a escolha de um método de cor apropriada para um projeto, baseando-se no seu tamanho e importância. Projetos de maior dimensão tendem a necessitar de mais coordenação e métodos mais pesados do que projetos menores. Os métodos da família *Crystal* são abertos para quaisquer práticas de desenvolvimento, ferramentas ou produtos de trabalho, permitindo assim a integração, como por exemplo, do XP e de práticas SCRUM [Cockburn,2002];

- *Dynamic Systems Development Method (DSDM)* - tem como ideia fundamental fixar custo, qualidade e prazo de um projeto e utilizar a técnica MoSCoW (*Must, Should, Could, Won't*) de priorização para definir o escopo esperado para o produto resultante do projeto de forma alinhada com os recursos disponíveis e com o nível de qualidade esperado. DSDM pode ser visto como o primeiro método de desenvolvimento de software realmente ágil [Stapleton,1997];
- *Extreme Programming (XP)* – consiste em um conjunto de práticas bem conhecidas da engenharia de software para o desenvolvimento de software bem sucedido, ainda que em situações de definição vaga de requisitos ou de constante mudanças nos requisitos. O método estabelece um conjunto de valores e práticas que tendem a gerar resultados de maior qualidade quando utilizadas em conjunto. Dentre as principais práticas do XP podem ser citadas: iterações curtas com a rápida produção do produto em pequenos incrementos, *feedback* constante, participação efetiva do cliente, constante comunicação e coordenação, reestruturação contínua do código, integração contínua e testes automatizados, propriedade coletiva do código e programação em pares [Beck,2000];
- *Feature-Driven Development (FDD)* - é um método de desenvolvimento de software orientado a processos para o desenvolvimento de sistema de negócios críticos. A abordagem FDD se concentra na modelagem e no desenvolvimento do projeto. A abordagem FDD incorpora o desenvolvimento iterativo com as práticas que se acredita serem eficazes na organização. A mistura específica desses aspectos faz com que os processos FDD sejam únicos para cada caso. Ele enfatiza aspectos de qualidade durante todo o processo e inclui entregas frequentes e tangíveis, juntamente com um acompanhamento preciso da evolução do projeto [Palmeret. al. 2002].

Além dos métodos já descritos, o SCRUM é outro método ágil bastante utilizado na indústria, cujo foco é a gerência do projeto. Como as formas de planejamento de projeto utilizadas pelo SCRUM foram o alvo deste trabalho, a próxima seção descreve alguns detalhes importantes desse método.

## 2.2 SCRUM

Segundo Schwaber (1995), SCRUM é uma jogada do Rugby na qual os jogadores se unem em posições específicas para colocar uma bola, que estava praticamente perdida, de volta ao jogo. O interessante dessa jogada é que o time é responsável por organizar sua própria formação. Inspirado nesse princípio de trabalho em equipes auto-gerenciáveis, SCRUM é um modelo de desenvolvimento de software baseado em uma ou mais equipes pequenas trabalhando de forma intensiva e interdependente.

No SCRUM, as equipes de trabalho são pequenas e auto-organizadas com o objetivo de maximizar a comunicação, minimizar a supervisão e maximizar o compartilhamento de conhecimento. O processo tem que se adaptar às tecnologias definidas e às modificações requeridas pelo cliente, com o objetivo de oferecer, de forma ágil e incremental, um produto melhor. O desenvolvimento e os recursos humanos envolvidos são divididos em partições claras de baixo acoplamento. Testes e documentação são produzidos à medida em que o produto é construído de forma iterativa. [Schwaber,1995].

### 2.2.1 Papéis SCRUM

Segundo Schwaber (2004), existem três papéis no SCRUM: o *Product Owner*, o *Team*, e o *Scrum Master*. Todas as responsabilidades de gestão em um projeto são divididas entre esses três papéis:

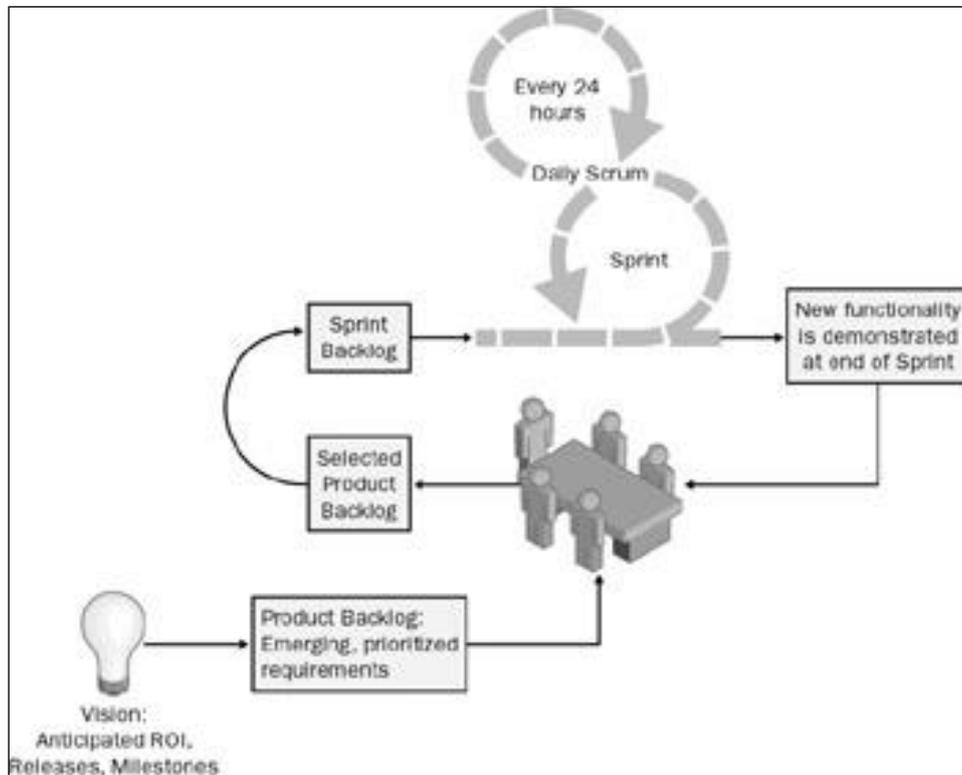
- O *Product Owner* é responsável por representar os interesses de todos os envolvidos no projeto e seu sistema resultante. O *Product Owner* consegue financiamento inicial e permanente para o projeto, criando o projeto inicial, definindo os requisitos gerais, o retorno sobre o investimento (ROI), objetivos e planos de liberação de versões. Ele também define a relação dos requisitos mais importantes que devem ser desenvolvidos em cada iteração do processo de desenvolvimento [Schwaber, 2004].
- *Team* (equipe) é responsável pelo desenvolvimento do produto. Equipes SCRUM tem auto-gestão, auto-organização e são multifuncionais. Além disso, a equipe é responsável por definir como transformar os requisitos em software

funcionando ao final de uma iteração, bem como gerenciar seu próprio trabalho para chegar a esse objetivo. Os membros da equipe são coletivamente responsáveis pelo sucesso de cada iteração e do projeto como um todo [Schwaber, 2004].

- O *Scrum Master* é responsável pelo processo SCRUM. Ele provê o ensino do método de trabalho para todos os envolvidos no projeto, garante a implementação do SCRUM para que ele seja inserido na cultura da organização, aponta os benefícios esperados, e ainda, é responsável por assegurar a institucionalização das regras e práticas da metodologia ágil. Além disso, ele é responsável por resolver os impedimentos apontados pela equipe ao longo de uma iteração [Schwaber, 2004].

### **2.2.2 Dinâmica do Processo**

As práticas do SCRUM são definidas com base em um processo iterativo e incremental, cuja estrutura geral é ilustrada na Figura 1. O círculo inferior representa uma iteração que corresponde à realização de um conjunto de atividades de desenvolvimento, cujo resultado é um incremento de funcionalidades agregadas ao produto. O produto, portanto, é o resultado produzido ao longo de uma sequência de iterações. O círculo superior representa a inspeção diária que ocorre durante cada iteração, que consiste em uma reunião na qual os membros da equipe se atualizam sobre o andamento das atividades uns dos outros e fazem as devidas adaptações e correções de rumo. A referência para o planejamento e a execução de uma iteração é uma lista de requisitos, denominada *Product Backlog*. As iterações se repetem até que o projeto acabe ou não seja mais financiado [Schwaber, 2004].



**Figura 1 - Processo de iteração do SCRUM**

Segundo Schwaber (2004), no início de um projeto baseado no SCRUM, existe apenas uma visão geral e ainda vaga do sistema a ser construído, que depois vai se tornando menos abstrata e mais efetiva à medida em que o projeto avança. O *Product Owner* formula um plano geral, que inclui a definição do *Product Backlog* contendo uma lista dos requisitos funcionais e não funcionais do produto. Os itens do *Product Backlog* devem ser priorizados conforme o valor que cada item agrega ao negócio.

De acordo com Schwaber (2004), o projeto é dividido em atividades que a equipe deverá executar durante os *Sprints*. Os *Sprints* são iterações que normalmente possuem a duração de 30 dias consecutivos, podendo ser mais curtas (e.g. 15 dias). Cada *Sprint* é iniciado com uma reunião denominada Reunião de Planejamento de *Sprint*, onde toda a equipe, o *Product Owner* e o *Scrum Master* participam e decidem os itens do *Product Backlog* que deverão estar incorporados ao produto ao final do *Sprint*. Uma vez definidos os itens do *Product Backlog* que serão desenvolvidos no *Sprint*, os membros da equipe planejam as atividades necessárias para que esse objetivo seja atingido. O conjunto de tarefas que serão realizadas em um determinado *Sprint* é conhecido como *Sprint Backlog*.

Após o planejamento inicial do *Sprint*, a equipe executa as tarefas planejadas e acompanha seu andamento utilizando uma dinâmica conhecida como *Daily SCRUM*, que consiste em uma reunião diária, de aproximadamente 15 minutos de duração, na qual cada membro da equipe responde a três perguntas:

- O que você fez desde o último *Daily SCRUM*?
- O que você planeja fazer até o próximo *Daily SCRUM*?
- Existe algum impedimento para realização das tarefas?

Essa dinâmica tem o objetivo de sincronizar o trabalho de toda a equipe, seguindo o princípio de promover equipes auto-gerenciadas e auto-organizadas.

Ao final de cada *Sprint*, as partes envolvidas no projeto fazem duas reuniões onde o produto e o processo de trabalho são avaliados. Na reunião conhecida como *Sprint Review*, a equipe apresenta os resultados alcançados ao final do *Sprint*, mantendo o *Product Owner* informado sobre o andamento do projeto. Os resultados alcançados correspondem a requisitos efetivamente incorporados ao produto e que podem ser demonstrados ao *Product Owner*. Nessa reunião, a equipe também tem a possibilidade de validar o produto construído e de esclarecer dúvidas com o *Product Owner*. O *Product Owner*, por sua vez, pode tomar decisões sobre prioridades, novos requisitos e mudanças nos requisitos em função do que foi apresentado pela equipe. Resumindo, essas reuniões servem para validar as funcionalidades realizadas e definir prioridades para ações futuras. A segunda reunião, *Sprint Retrospective*, envolve a equipe e o *Scrum Master* e objetiva avaliar criticamente o processo de trabalho e identificar oportunidades de melhorar a forma com que o produto será desenvolvido nas próximas iterações [Schwaber, 2004].

### **2.2.3 Artefatos do SCRUM**

Segundo Schwaber (2004), o SCRUM define dois artefatos importantes para o planejamento e acompanhamento de um projeto: *Product Backlog* e *Sprint Backlog*.

#### ***Product Backlog***

Este artefato corresponde à lista de requisitos do produto. O responsável pelo conteúdo do *Product Backlog* é o *ProductOwner*, que define a priorização e a alocação dos seus itens aos *Sprints* e às liberações do projeto. O *Product*

*Backlog* evolui ao longo do desenvolvimento do projeto e sua existência está vinculada à existência do produto, ou seja, o *Product Backlog* existe enquanto o produto existir [Schwaber, 2004]. Tipicamente, além da definição do requisito, um item do *Product Backlog* possui duas informações necessárias ao planejamento do projeto: valor para o negócio, definido pelo *Product Owner*, que será utilizado como referência para priorização dos itens na alocação dos itens ao longo dos *Sprints*; esforço, definido pela equipe, consiste em um número que indica, tipicamente de forma relativa, o esforço necessário para o desenvolvimento do item.

### ***Sprint Backlog***

Este artefato registra o plano de trabalho contendo o conjunto de tarefas necessárias para realizar os itens do *Product Backlog* definidos como o incremento da funcionalidade que deverá ter sido agregado ao produto ao final do *Sprint*. Na segunda parte da reunião de planejamento do *Sprint*, a equipe define uma lista inicial de tarefas e faz uma estimativa do esforço em horas necessário para sua realização. As tarefas devem ser definidas de forma que a duração estimada de cada uma esteja entre 4 a 16 horas. Tarefas com duração estimada maior que 16 horas indicam um grau de desconhecimento sobre a real complexidade da tarefa e devem ser refinadas em outras tarefas. Somente a equipe pode mudar o *Sprint Backlog*, o que confere o caráter de auto-gerenciamiento da equipe. O *Sprint Backlog* é altamente visível, e expressa a imagem em tempo real do trabalho que a equipe planeja realizar durante o *Sprint* [Schwaber, 2004].

### **Histórias**

Segundo Cohn (2004), uma história descreve uma funcionalidade que gera valor para o usuário ou cliente de um software. As histórias de usuário são criadas pelo cliente, ou com muita participação dele e apoio da equipe, no momento do levantamento de requisitos. O ideal é que seja sempre apenas uma pessoa a criar as histórias e priorizá-las para repassar a equipe de desenvolvimento. As histórias são a materialização das necessidades do cliente

em relação ao software. Ou seja, são as especificações detalhadas das funcionalidades desejadas.

### ***Epics***

Um *epic* pode ser considerado com uma grande ou um conjunto de histórias do usuário. Ele possui um nível pequeno de detalhamento e dará origem a vários itens do Product Backlog à medida em que for detalhado ao longo do projeto [Cohn, 2006].

### **Tarefas**

As tarefas consistem no menor detalhe das atividades do projeto. Após a definição das histórias a equipe define as tarefas necessárias para que as histórias sejam realizadas [Cohn, 2006].

### **Releases**

Os releases são entregas do projeto (conjunto de funcionalidades) que agregam valor ao produto e já podem ser utilizados pelo usuário. Eles são definidos por uma data de início e fim (data de entrega) [Schwaber, Beedle, 2001].

### ***Sprints***

Conforme mencionado anteriormente, o *sprint* consiste no plano de trabalho contendo o conjunto de tarefas necessárias para realizar os itens de *backlog* em um determinado período de tempo. Os *sprints* são definidos pela equipe de forma conjunta. Ao contrário dos releases, os *sprints* não consistem obrigatoriamente em uma entrega de uma “versão” do produto, ainda assim, os *sprints* tem uma data de início e fim e geralmente duram duas semanas [Schwaber, 2004].

## **2.3 SCRUM e Estimativas**

Segundo Cohn (2006), é muito comum tentar realizar estimativas com alto grau de detalhamento, mas isso pode não ser tão interessante quando se faz uso de métodos

ágeis de projeto. Estimativas em projetos ágeis são definidas geralmente em grupo para tentar minimizar posições individuais que tendem tanto para o otimismo quanto para o pessimismo.

Existem diversas técnicas para realizar estimativas em projetos ágeis como, por exemplo, dias ideais, *planning poker* e pontos de função.

- Dias Ideais ou Horas Ideais - Dia Ideal ou Hora Ideal consiste no volume de trabalho que é realizado em um dia (ou uma hora) sem nenhum tipo de interrupção ou distração [Fonseca et. al., 2008]. Dessa forma, um dia ideal é um dia de trabalho onde a história estimada é a única coisa na qual o desenvolvedor irá trabalhar. Considera-se que tudo que o profissional necessitar estará à mão imediatamente e sem qualquer tipo de interrupção. Dias ideais não são dias reais e, portanto, estimativas nessa unidade de medida não podem ser diretamente convertidas em tempo real. Quando utiliza-se dias ideais para estimativas, deve-se estar atento ao fato de que, a despeito do nome, dia ideal não é uma unidade de tempo, mas sim de esforço ou de tamanho de uma história. Segundo Martins (2007) o cálculo destes dias estimados utiliza a seguinte fórmula:

$$DE = \frac{IED}{1 - IED\_REAL\%}$$

Onde:

DE: quantidade de dias estimado para concluir a tarefa;

IED: prazo necessário para implementar o item, esse prazo é definido pela equipe;

IED\_REAL%: percentual que indica a estimativa de quanto tempo do dia o desenvolvedor ficara dedicado a implantação do item.

### Figura 2 – Fórmula para o Cálculo de Dias Ideais

- *Planning Poker* - no *Planning Poker*, a estimativa utiliza o *Product Backlog* e um conjunto de cartas. Muitos utilizam a sequência Fibonacci para numerar essas cartas (0, 1, 2, 3, 5, 8, 13, 20, 40 e 100) [PlanningPoker, 2009]. Cada membro da equipe deve possuir um conjunto dessas cartas, pois cada item do

*Product Backlog* terá um valor definido por meio de rodadas de estimativas realizadas pelos membros. Para cada item do *Product Backlog*, os membros relacionam uma carta com o valor que acham ser o ideal. Essa estimativa é relativa, isto é, dados dois itens A e B aos quais tenham sido atribuídos dois valores, 2 e 1, respectivamente, isso significa que estima-se que o esforço associado a A é o dobro de B. Depois que toda equipe jogar, é discutido o valor ideal para aquele item. Caso a equipe não chegue a um consenso, são realizadas mais rodadas até a equipe chegar a um consenso. O *Planning Poker* é concluído quando todos os itens do *Product Backlog* dentro do escopo do planejamento (*Release* ou *Sprint*) tiverem sido estimados;

- Pontos – A técnica de estimativas por pontos, segundo Dekkers (1998), consiste na definição dos Pontos de Função (PF) que medem o tamanho funcional do software, onde o tamanho funcional pode ser definido como uma medida de tamanho de software realizada a partir da funcionalidade implementada em um sistema sob o ponto de vista do usuário.

O *Product Backlog* apresentado na Tabela 1 define um conjunto de itens e suas respectivas estimativas de esforço em dias ideais.

Nome	Descrição	Esforço (dias)
Cadastros	Realização dos cadastros necessários para a utilização do sistema	65
Modelos Lógicos	Documentar os modelos lógicos necessários para o sistema.	47
Simulador de Monte Carlo	Criação das funcionalidades necessárias para realizar a Simulação de Monte Carlo de dados que será utilizado	80

**Tabela 1 - Exemplo de *Product Backlog***

Ao realizar o planejamento de um Release, consideramos o cenário onde existe um prazo estabelecido no qual deseja-se entregar um novo release do produto cumprindo um conjunto de requisitos definidos no *Product Backlog*. Neste contexto, o

problema é definir o conjunto de requisitos que podem ser desenvolvidos considerando o esforço estimado para cada requisito e a capacidade de produção da equipe.

Como um *Release* é normalmente construído por meio de várias iterações (*Sprints*), o mesmo problema ocorre em uma escala de tempo menor no planejamento do *Sprint*, onde desejamos definir o conjunto de requisitos (subconjunto dos requisitos alocados no planejamento do *Release*) que poderão ser desenvolvidos no intervalo de tempo definido pelo *Sprint*.

O planejamento de um *Sprint* é dividido em duas partes. Na primeira define-se o conjunto de itens do *Product Backlog* a serem incorporados ao produto, enquanto na segunda planeja-se todas as tarefas necessárias para a realização desses itens. Esse conjunto de tarefas forma o *Sprint Backlog*.

O exemplo ilustrado na Tabela 2 apresenta algumas atividades deste projeto de graduação, as quais devem ser encaixadas no *sprint* “Implementação das Classes de Negócio”, que tem início em 1/9/2013 e término em 10/10/2013, totalizando um intervalo de 39 dias.

Nome	Descrição	Esforço
Classe Perfil	Implementar a classe de domínio relativo ao Perfil de um usuário	8
Classe Release	Implementar a classe de domínio relativo ao Release	17
Classe User	Implementar a classe de domínio relativo ao Usuário	10
Classe UserProject	Implementar a classe de domínio relativo ao relacionamento entre Usuário e Projeto	7
Classe Epic	Implementar a classe de domínio relativo a Epic	10
Classe Sprint	Implementar a classe de domínio relativo ao Sprint	9
TOTAL		61

**Tabela 2 - Exemplo de *Sprint Backlog***

Ao somar todas as atividades envolvidas, o total obtido é 61 dias, ou seja, o total das atividades estimadas está além do intervalo definido pelo *sprint* (39 dias), considerando que as estimativas pontuais sejam precisas, o que não é muito provável.

Para ajudar no planejamento do *Release* e dos *Sprints*, este projeto propõe uma solução que apóie o usuário a avaliar a estimativa total de esforço no planejamento de *Release* e de *Sprints* a partir de um gráfico de probabilidade acumulado gerado pela aplicação do método de simulação de Monte Carlo, considerando que a estimativa de esforço associada a cada item alocado a um *Release* ou *Sprint* é feita conforme uma distribuição de probabilidade (no escopo deste trabalho, será utilizada a distribuição triangular). Os conceitos utilizados nessa técnica são descritos nos tópicos a seguir.

## 2.4 Ferramentas Similares

Em decorrência do trabalho, foram vistas algumas ferramentas que se aproximavam do escopo do sistema desenvolvido neste trabalho, no entanto com algumas restrições, como a utilização de estimativas pontuais e o planejamento de *Releases* e *Sprints* sem o auxílio de uma distribuição de probabilidade para ajudar no planejamento e nos cenários E-SE.

As ferramentas analisadas foram Axosoft Scrum Software (<http://www.axosoft.com/scrum>), Histimate (Soares, M e Almeida, R [2013]) e Scrum Half (<http://myscrumhalf.com/?lang=pt>), para uma breve comparação das principais funcionalidades, como visto na Tabela 3.

	Axosoft	Histimate	Scrum Half	Sistema de Apoio ao Planejamento de Projetos Baseado no Método de Simulação de Monte Carlo
Controla Epics, Histórias e Tarefas	<b>Sim</b>	<b>Parcial</b>	<b>Parcial</b>	<b>Sim</b>
Estimativas Pontuais	<b>Sim</b>	<b>Sim</b>	<b>Sim</b>	<b>Não</b>
Estimativas 3 pontos	<b>Não</b>	<b>Não</b>	<b>Não</b>	<b>Sim</b>
Geração de cenários E-SE	<b>Não</b>	<b>Não</b>	<b>Não</b>	<b>Sim</b>
Cadastro de Usuários	<b>Sim</b>	<b>Sim</b>	<b>Sim</b>	<b>Sim</b>
Utilização de Modelos Probabilísticos para planejamento de <i>Releases</i> e <i>Sprints</i>	<b>Não</b>	<b>Não</b>	<b>Não</b>	<b>Sim</b>

**Tabela 3 - Comparação de ferramentas similares**

## 2.5 Distribuição Triangular

A distribuição triangular é uma distribuição probabilística contínua definida por dois limites, um limite inferior (valor mínimo) e um limite superior (valor máximo) e por uma moda (que neste trabalho corresponderá ao valor do caso mais provável).

Essa distribuição é muito utilizada na tomada de decisões de negócios ou no gerenciamento de riscos, visto que normalmente os limites (mínimo e máximo) são conhecidos, e não se sabe muito sobre a distribuição do resultado, devendo-se utilizar uma distribuição uniforme. No entanto, como no âmbito deste trabalho podemos definir o caso mais provável, realizamos a simulação utilizando uma distribuição triangular.

$$f(x|a, b, c) = \begin{cases} \frac{2(x-a)}{(b-a)(c-a)} & \text{for } a \leq x \leq c \\ \frac{2(b-x)}{(b-a)(b-c)} & \text{for } c \leq x \leq b \\ 0 & \text{otherwise} \end{cases}$$

**Figura 3 - Algoritmo da Distribuição Triangular Utilizada**

Conforme ilustrado na Figura 3, os limites superior (letra b), inferior (letra a) e o valor médio (letra c) podem ser associados aos termos utilizados neste trabalho. O limite inferior equivale ao pior caso, o limite superior corresponde ao melhor caso e a média é o caso mais provável (ainda que o caso mais provável não seja necessariamente a média do melhor caso com o pior caso).

## **2.6 Método de Simulação de Monte Carlo**

O sistema desenvolvido neste trabalho visa apoiar as atividades de planejamento de *Releases* e de *Sprints* em um projeto que segue o método SCRUM, considerando a utilização do método de simulação de Monte Carlo para avaliar probabilisticamente as expectativas do escopo estimado para conclusão em um *release* ou *sprint*, com base nas estimativas de esforço dos itens do *Product Backlog* e das tarefas do *Sprint Backlog*.

Esta seção apresenta brevemente os conceitos relacionados ao Método (simulação) de Monte Carlo. Será apresentada uma breve história sobre o método, as suas principais aplicações e o algoritmo usado neste trabalho, finalizando com a apresentação do conceito de estimativa de 3 pontos que, embora não esteja diretamente associado ao método, foi utilizado na geração dos números pseudoaleatórios requeridos pelo método.

### **2.6.1 Introdução**

O método ou simulação de Monte Carlo é o conjunto ou classe de algoritmos computacionais que dependem de uma amostragem aleatória para poder obter os resultados por meio de cálculos. O método consiste no uso de números pseudoaleatórios gerados por computadores, o que justifica a forte dependência da computação para a utilização prática desse método.

Os algoritmos de Monte Carlo utilizam como base a Lei dos Grandes Números. Essa lei estabelece que ao se gerar um grande número de amostras, em algum momento a distribuição aproximada desejada será obtida [Teknomo].

A simulação possui três características comuns em todas as suas variações:

- A amostra é gerada de forma aleatória;
- A distribuição de entrada é conhecida;
- A realização de experimentos numéricos.

### **2.6.2 História**

A simulação surgiu a partir da variação de outros experimentos, principalmente através do experimento da Agulha de Buffon, método para estimar o valor de  $\pi$  ao derrubar-se agulhas em um piso feito de tábuas paralelas de madeira. O método foi estudado pela primeira vez em 1930, por Enrico Fermi para estudar a difusão de nêutrons. Posteriormente, em 1946, físicos do laboratório científico de Los Alamos, apesar de terem a maioria dos dados para resolver o problema, foram incapazes de obter um resultado através de métodos matemáticos convencionais ou determinísticos onde dois dos cientistas realizaram uma pesquisa para tentar obter uma resposta, Jonh von Neumann e Stanislaw Ulam [Eckhardt, Roger,1987]. O trabalho recebeu o nome de Monte Carlo por conta dos cassinos de Monte Carlo em Mônaco, onde o tio de Stanislaw Ulam pegou dinheiro para apostar. O método foi parte central das simulações utilizadas no projeto Manhattan para o desenvolvimento da bomba atômica e posteriormente no desenvolvimento da bomba de hidrogênio.

### **2.6.3 Aplicações e Áreas de utilização**

O método de Monte Carlo é aplicado principalmente em problemas que necessitem de números elevados de amostras aleatórias e possuam um grau de incerteza significativo nos dados de entrada e com grande grau de liberdade [Monte Carlo Method -[http://en.wikipedia.org/wiki/Monte\\_Carlo\\_method](http://en.wikipedia.org/wiki/Monte_Carlo_method)]. As principais áreas de aplicação do método são:

- Ciências Físicas: utilizado principalmente para cálculos de cromodinâmica quântica, que auxiliam no desenvolvimento de escudos térmicos e formas aerodinâmicas, modelagem molecular – auxiliando nos

cálculos das teorias de campo estatístico dos sistemas de partículas e de polímeros simples e utilizado também na evolução de galáxias e transmissão de radiação de micro-ondas por meio da superfície planetária;

- Engenharia: usado na análise de sensibilidade, análise de rendimento energético (utilizado principalmente na área eólica), nos impactos da poluição do diesel comparado com a gasolina, na área de robótica - onde o método pode auxiliar a determinar a posição de um robô e no planejamento de redes wireless – provando que o design da rede funciona dentro de uma variedade de cenários;
- Computação Gráfica: ajuda a renderizar uma cena 3D, traçando aleatoriamente caminhos de luz pelo cenário;
- Estatística aplicada: usado ajudar na implementação de testes de hipóteses que se mostram mais eficientes que os testes exatos e na comparação com estatísticas concorrentes, que tem pequenas amostras de dados em condições reais.

#### **2.6.4 Algoritmo de Monte Carlo**

O algoritmo de Monte Carlo descrito neste trabalho funciona em conjunto com a distribuição triangular para variáveis pseudoaleatórias. [Adaptado de <http://www.codeproject.com/Articles/32654/Monte-Carlo-Simulation>] O mesmo utiliza um conjunto de atividades (Epics, Histórias ou Tarefas), onde se define uma estimativa de 3 pontos para cada uma dessas atividades (melhor caso, caso mais provável e pior caso), além de um número gerado aleatoriamente (compreendido entre 0.0 e 1.0), segundo a Figura 4.

```

public static IList<double> monteCarloDouble(int iterationNumber, IList<Epic> projectItens)
{
    Random r = new Random();
    IList<double> resultsValues = new List<double>();
    for (int i = 0; i < iterationNumber; i++)
    {
        var result = 0.0;
        foreach (Epic projectItem in projectItens)
        {
            double minimun = (double)projectItem.PiorCaso;
            double mostProbable = (double)projectItem.CasoMedio;
            double maximum = (double)projectItem.MelhorCaso;

            result += Triangular.triangular(minimun, mostProbable, maximum,r);
        }
        resultsValues.Add(result);
    }

    return resultsValues;
}

```

**Figura 4 - Algoritmo de Monte Carlo**

Segundo a Figura 4 o algoritmo realiza uma chamada ao algoritmo de distribuição triangular, visto na Figura 5, que calcula um valor para compor uma variável que será o valor acumulado das probabilidades de um determinado conjunto de atividades em uma iteração. A cada iteração, o valor acumulado é salvo em uma lista encadeada (“resultValues” visto na Figura 4) e o conjunto de valores da lista nos dá o gráfico de distribuição acumulada.

```

public static double triangular(double Min, double Mode, double Max,Random r)
{
    // Declarations
    double R = 0.0;
    // Initialise

    R = r.NextDouble(); //between 0.0 and 1.0 gaussian
    // Triangular
    if (R == ((Mode - Min) / (Max - Min)))
    {
        return Mode;
    }
    else if (R < ((Mode - Min) / (Max - Min)))
    {
        return Min + Math.Sqrt(R * (Max - Min) * (Mode - Min));
    }
    else
    {
        return Max - Math.Sqrt((1 - R) * (Max - Min) * (Max - Mode));
    }
}

```

**Figura 5 - Algoritmo de distribuição triangular**

Considerando que um *backlog* tenha um conjunto de  $n$  entradas, cada entrada é associada a uma variável pseudoaleatória para a qual deve ser feita uma estimativa de três pontos correspondentes ao pior caso, melhor caso e caso mais provável (Tabela 4). A cada iteração da simulação, é feita uma amostragem de cada variável pseudoaleatória assumindo uma distribuição triangular de probabilidade definida pela estimativa de três pontos associada. O esforço total do conjunto de itens do *backlog* é dado pela soma dos valores amostrados para cada variável pseudoaleatória (Tabela 5). Esse processo é realizado por  $N$  iterações, onde sugere-se realizar ao menos 10.000 iterações.

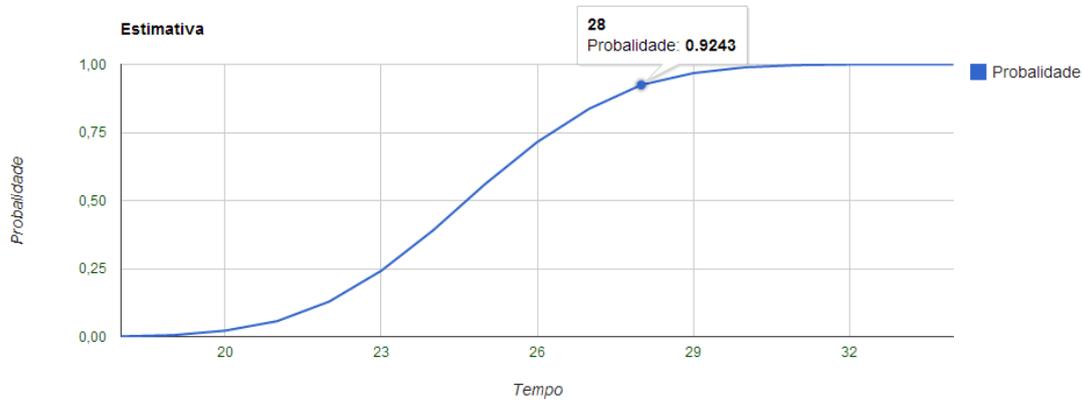
Nome Atividade	Melhor Caso	Caso mais provável	Pior Caso
Diagrama de Classes	2	4	6
Diagrama de Casos de Uso	4	7	9
Classe User	6	8	10
Classe Perfil	3	5	8
Classe UserProject	1	3	7

**Tabela 4 - Exemplo de Atividades com as estimativas de 3 pontos**

Nome da Atividade	Valor do Cenário 1	Valor do Cenário 2	Valor do Cenário 3
Diagrama de Classes	5	3	3
Diagrama de Casos de Uso	6	5	8
Classe User	8	7	8
Classe Perfil	5	5	6
Classe UserProject	5	1	3
<b>Total</b>	<b>29</b>	<b>21</b>	<b>28</b>

**Tabela 5 - Exemplo de Alguns Cenários da Simulação de Monte Carlo**

Uma vez realizadas todas as iterações, obtém-se um conjunto de valores a partir do qual será calculada a frequência acumulada. O produto resultante é o gráfico de distribuição acumulada, onde cada ponto representará seu valor e sua probabilidade acumulada.



**Figura 6 - Exemplo Gráfico de Probabilidade Acumulada**

O gráfico aponta, no eixo das abscissas, a probabilidade e o eixo das coordenadas o tempo, sendo assim, o gráfico produzido indica com qual confiança (probabilidade) o conjunto de atividades escolhido será realizado no tempo indicado. No exemplo da Figura 6, o gráfico mostra que a probabilidade deste conjunto de tarefas serem realizadas em no máximo 28 dias (tempo) é de 92,43%.

Contudo, o usuário pode se apoiar neste gráfico analisando os resultados obtidos e assim avaliar alternativas inserindo ou removendo itens que serão considerados neste planejamento, realizando a simulação após estas movimentações. Desta forma, permite-se uma análise E-SE ao usuário do seu planejamento com apoio probabilístico.

## 3 O Sistema

Este capítulo descreve o software desenvolvido, apresentando as suas funcionalidades e as tecnologias utilizadas. A seção 3.1 apresenta uma breve visão da solução. A seção 3.2 descreve as tecnologias utilizadas na implementação. A seção 3.3 apresenta o modelo de dados, a seção 3.4 o modelo de casos de uso e a seção 3.5 apresenta os detalhes de interação das funcionalidades do sistema.

### 3.1 Visão Geral da Solução

Este trabalho apresenta uma ferramenta que visa apoiar as atividades de planejamento de *releases* e *sprints* de um projeto desenvolvido com o método SCRUM ao fornecer uma visão probabilística do esforço necessário para realizar as histórias alocadas a um release ou sprint ou para concluir as tarefas planejadas para um sprint.

Para que essa visão probabilística pode ser gerada pelo sistema, os dados do projeto, os itens que compõem o *Product Backlog*, os dados do *release* ou *sprint* que estiver sendo planejado (período de realização), as tarefas planejadas para o *sprint* e as estimativas tanto para os itens do *Product Backlog* como para as tarefas do *Sprint Backlog* devem ser entrados ou importados para o sistema. Conforme descrito no capítulo anterior, as estimativas de esforço do projeto (tanto para itens do *Product Backlog* quando para as tarefas de um *Sprint*) são definidas pelo usuário utilizando três pontos: valor no melhor caso; valor no pior caso e valor mais provável.

Uma vez que essas informações tenham sido entradas no sistema, o usuário poderá gerar o gráfico de probabilidade acumulada escolhendo os itens do *Product Backlog* (se estiver analisando escopo de um *Release* ou *Sprint*) ou itens do *Sprint Backlog* (se estiver analisando capacidade de realização de tarefas) que farão parte da simulação de Monte Carlo. Dessa forma, a ferramenta permite que a realização de

análises E-SE, por meio da avaliação da variação dos resultados em função da adição ou remoção de itens do *backlog*, apoiando o processo de definição do escopo de um *release* ou *sprint* do projeto.

## **3.2 Tecnologias Utilizadas**

Esta seção apresenta as principais tecnologias utilizadas na construção do sistema.

### **3.2.1 Linguagem de Programação C#**

C# foi a linguagem de programação escolhida para a construção do projeto por ter grande compatibilidade com as outras tecnologias e possuir bons recursos de desenvolvimento rápido de aplicações web.

O C# é uma linguagem de programação criada para o desenvolvimento de uma variedade de aplicações que executam sobre o .NET Framework. C# é uma linguagem simples, poderosa, com “tipagem” segura e orientada a objetos. As várias inovações no C# permitem o desenvolvimento rápido de aplicações, mantendo a expressividade e a elegância do estilo de linguagens C.

A biblioteca de classes do .NET Framework fornece acesso a vários serviços do sistema operacional e outras classes úteis e bem estruturadas que aceleram significativamente o ciclo de desenvolvimento.

### **3.2.2 Google Charts**

Esta ferramenta foi escolhida para a construção do gráfico de distribuição acumulada de probabilidade. O Google Charts é uma poderosa ferramenta da Google que permite a construção de gráficos de maneira simples. Ela possui desde gráficos de linha simples a complexos mapas de árvores hierárquicas prontos para uso.

Neste projeto, o Google Charts foi usado com JavaScript incorporado à página web. Gráficos são expostos como classes JavaScript, sendo que o Google Charts oferece muitos tipos de gráficos. Os gráficos são altamente interativos e é possível criar eventos que permitem conectá-los de forma a criar *dashboards* complexos ou outras experiências integradas com a página de uma aplicação. Os gráficos são processados usando tecnologia HTML5/SVG para fornecer compatibilidade *cross-browser* (incluindo VML para versões mais antigas do Internet Explorer) e portabilidade entre

plataformas para iOS e Android. Os usuários não precisam de plug-ins ou qualquer software, basta utilizar o próprio navegador web.

Todos os tipos de gráficos são preenchidos com dados usando a classe *DataTable*, tornando mais fácil alternar entre os diferentes tipos de gráficos. O *DataTable* fornece métodos para a classificação, modificação e filtragem de dados, e pode ser preenchido diretamente de uma página web, um banco de dados, ou qualquer provedor de dados que suporte o protocolo *Datasource* Ferramentas de Gráfico. Esse protocolo inclui uma linguagem de consulta SQL-like e é implementado pelo Google *Spreadsheets*, Google *FusionTables*, e provedores de dados de terceiros, como o *SalesForce*, por exemplo.

### **3.2.3 Microsoft Entity Framework**

Para a implementação da modelagem de classes e implementação de código para a criação física das entidades no banco de dados foi utilizado o Microsoft Entity Framework, que possui excelente integração com o SQL SERVER (banco de dados utilizado neste sistema) simplificando as funções de acesso a banco de dados.

O Microsoft Entity Framework é uma ferramenta de mapeamento objeto relacional (ORM – *Object Relational Management*), que permite aos desenvolvedores trabalhar com classes (entidades) que correspondem a tabelas em um banco de dados, tornando transparente o acesso a esses dados e principalmente, eliminando a necessidade de escrever código de banco de dados (*SELECT*, *INSERT*, *UPDATE*, *DELETE*) na aplicação. Com o Entity Framework, os desenvolvedores manipulam os dados através de classes que são mapeadas com as tabelas do banco de dados, simplificando o acesso e a manipulação desses dados, pois o desenvolvedor já trabalha naturalmente com objetos, propriedades e coleções no seu desenvolvimento em C#.

A comunicação do Entity Framework com o banco de dados é feita através do ADO.Net Provider, que funciona como um “driver” do banco de dados, normalmente desenvolvido pelo próprio fabricante do banco, ou em alguns casos por um terceiro. Sendo assim, todos os comandos submetidos pelo Entity Framework são “traduzidos” para a linguagem do banco de dados através do seu *provider*, gerando os comandos SQL mais adequados a cada operação e principalmente, comandos que tenham o máximo de desempenho.

### 3.2.4 LINQ to Entities

Esta tecnologia foi utilizada neste projeto juntamente com o Entity Framework para simplificar o acesso pelas classes de negócio das funções básicas de banco de dados, provendo generalização, customização e aproveitamento de código.

O LINQ to Entities fornece suporte à LINQ (Consulta Integrada à Linguagem) que permite aos desenvolvedores escreverem consultas no modelo conceitual do Entity Framework usando Visual Basic ou Visual C#. As consultas no Entity Framework são representadas por consultas de árvore de comando, que são executadas no contexto de objeto. O LINQ to Entities converte consultas do LINQ (Consulta Integrada à Linguagem) para consultas de árvore de comando, executa as consultas no Entity Framework e retorna os objetos que podem ser usados pelo Entity Framework e pelo LINQ.

### 3.2.5. SQL SERVER 2008

Este foi o sistema gerenciador de banco de dados utilizado no projeto, pois o mesmo permite uma melhor integração com as tecnologias abordadas anteriormente. Por possuir uma integração nativa, esta ferramenta foi totalmente utilizada com o Entity Framework permitindo grande agilidade de modelagem, construção e utilização do banco de dados.

O Microsoft SQL Server Express é uma edição gratuita com muitos recursos do SQL Server que é ideal para aprendizado, desenvolvimento, habilitação de área de trabalho, aplicativos Web e de servidores pequenos. A versão do SQL Server Express inclui a versão completa do SQL Server Management Studio.

## 3.3 Modelo de Classes

Esta parte descreve o modelo lógico do banco de dados (Figura 7), podendo ser interpretada como o diagrama de classes, visto que utilizou-se o Microsoft Entity Framework que faz o mapeamento do modelo orientado a objetos para o modelo relacional.

A entidade principal é **Projeto (Project)** que concentra o maior número de associações. A associação com **Usuário (User)** remete ao gestor do projeto. A

associação com **Release** define a sequência de releases produzidas pelo projeto. A associação com **Epic** representa o *Product Backlog* do projeto, enquanto que as associações com **História e Tarefa** referenciam todas as atividades planejadas ou realizadas do projeto, sendo a primeira associação relativa ao escopo do *Sprint* e a segunda em relação as tarefas necessárias para a realização do *sprint*.

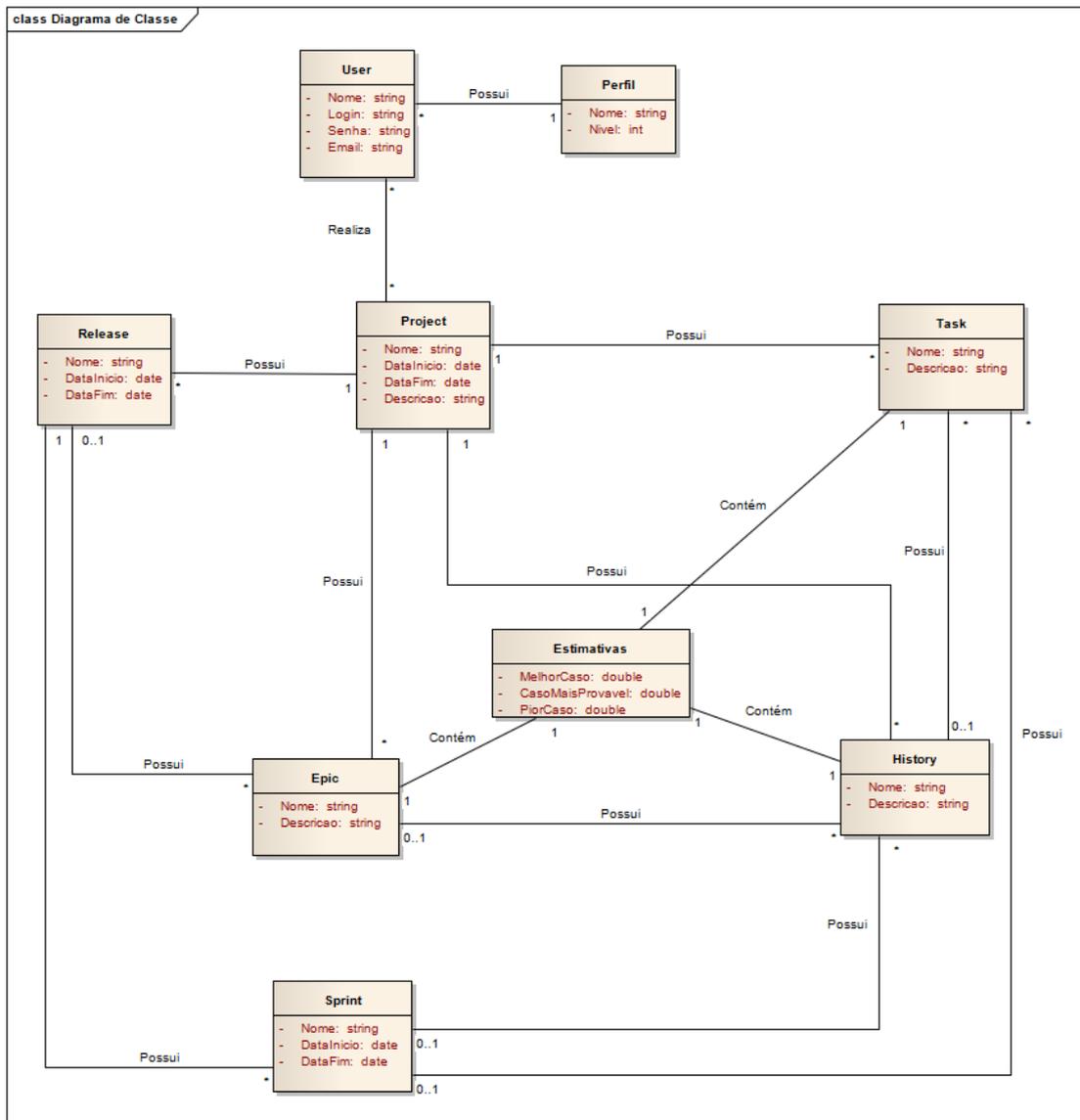
A associação da entidade **Release** com a entidade **Sprint** registra a decomposição de um *release* (médio prazo) em uma sequência de *sprints* (curto prazo).

A Associação da classe **Estimativas** com as classes **Epic, História e Tarefa**, ocorre por essas três classes terem em comum esses atributos, portanto foi derivada em uma classe externa e não em atributos para dar uma maior reutilização.

A associação de **Epic** com **Release** representa o conjunto de *epics* (requisitos de maior nível de abstração) alocados para construção em um determinado Release. Um *Epic* pode se associar a diversas instâncias de História de forma que essas históricas fazem a decomposição de um *Epic* em elementos menores e que podem ser alocados a um *sprint* específico.

A entidade **História (History)** está associada com a entidade **Sprint**, registrando a alocação de histórias do *Product Backlog* que devem ser construídas no *Sprint*. Uma história também pode estar associada a várias instâncias de **Tarefa (Task)**, registrando todas as tarefas necessárias para a conclusão da história.

A entidade **Tarefa (Task)** associa-se com a entidade **Sprint**, permitindo o registro das tarefas que estão alocadas ao *Sprint*.



**Figura 7 - Diagrama de Classes do Sistema**

### 3.4 Modelo de Casos de Uso

As funcionalidades do sistema são definidas pelo modelo de casos de uso apresentado na figura 8.



- **Caso de uso Realizar Login**

Realiza o login do usuário no sistema, onde o mesmo deve passar o seu nome de usuário e a sua senha, sendo que o caso de uso validar usuário é incluído para verificar a autenticidade do mesmo.

- **Casos de uso Manter *Release* e Manter *Sprint***

Estes casos de uso incluem dois outros casos de uso que são o caso de uso de cadastro de novos itens e o caso de uso de edição dos itens cadastrados. A remoção não foi implementada também a fim de evitar problemas de uma complexidade maior, que estão descritos nas limitações do projeto. Basicamente, esses casos de uso permitem manter atualizados os cadastros de *releases* e *sprints* os relacionamentos entre esses elementos.

- **Manter Epic, Manter História e Manter Tarefas**

Estes casos de uso incluem outros três casos de uso que são o caso de uso de cadastro de novos itens, o caso de uso de edição dos itens cadastrados e a importação de itens através de um padrão de um arquivo txt. A remoção não foi implementada também a fim de evitar problemas de uma complexidade maior, que estão descritos nas limitações do projeto. Basicamente, esses casos de uso permitem manter atualizados os cadastros de epics, histórias, tarefas e os relacionamentos entre esses elementos. Em especial o caso de uso de cadastro de novo item, inclui um outro caso de uso que é o de **Estimar**.

- **Caso de uso Estimar**

Este caso de uso é incluído pelos casos de uso de cadastro de *Epic*, História e Tarefa, além do mais ele também estende o caso de uso **Simular**, visto que as informações utilizadas por este caso de uso vêm, no caso de uso aqui descrito.

- **Caso de uso Manter Projeto**

Este caso de uso inclui dois casos de uso, que é o cadastro de novos projetos e a remoção de projetos. A edição de um projeto não foi implementada, sendo considerada como uma melhoria para o futuro.

- **Caso de uso Realizar Simulação**

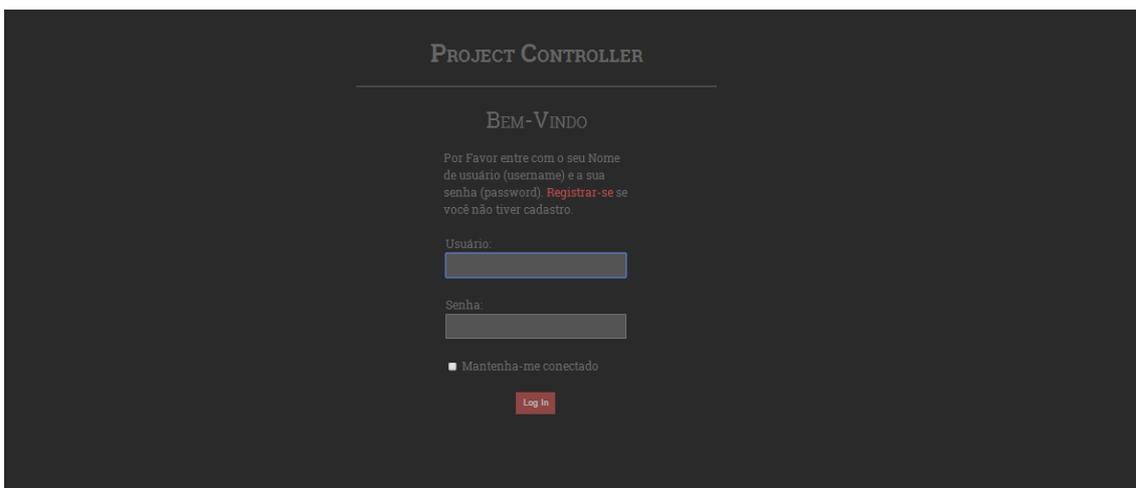
A realização da simulação é a principal funcionalidade do sistema. Neste caso de uso, os algoritmos de distribuição triangular e Monte Carlo são invocados para gerar a saída que irá compor o gráfico de probabilidade acumulada apresentado para o usuário. Este caso de uso também inclui outro caso de uso que é o **Plotar Gráfico**.

### 3.5 Módulos do Sistema

Esta seção descreve as funcionalidades do sistema.

#### 3.5.1 *Login*

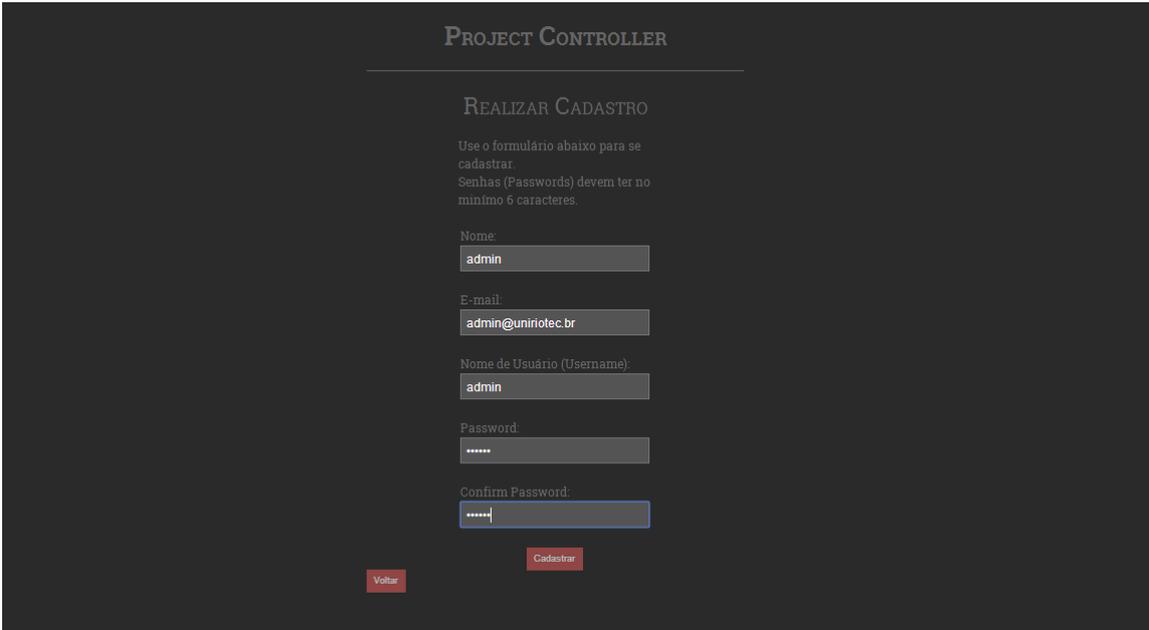
Este módulo consiste na apresentação da tela de *login* para acesso ao sistema (primeira tela exibida ao se acessar a ferramenta). O usuário pode realizar o *login* e direcionar para a tela de projetos ou clicar no link de “Registrar-se” para criar um novo *login*.



**Figura 9 - Tela de *Login* de Usuário**

### 3.5.2 Cadastro de Usuário

Conforme mencionado no item *Login*, se o usuário não tiver um *login* cadastrado, ele deverá fazer o seu cadastramento. Após clicar no link “Registrar-se” (Figura 9– *Login*) o sistema fará o direcionamento para a tela ilustrada pela Figura 8, onde o usuário deverá preencher o cadastro com os campos indicados e clicar em “Cadastrar” ou “Voltar” (Figura 8). Após realizar essas atividades, o *login* já estará apto para autorizar a entrada no sistema. No exemplo da Figura 10, o usuário “admin” foi criado.



PROJECT CONTROLLER

REALIZAR CADASTRO

Use o formulário abaixo para se cadastrar.  
Senhas (Passwords) devem ter no mínimo 6 caracteres.

Nome:  
admin

E-mail:  
admin@unirotec.br

Nome de Usuário (Username):  
admin

Password:  
\*\*\*\*\*

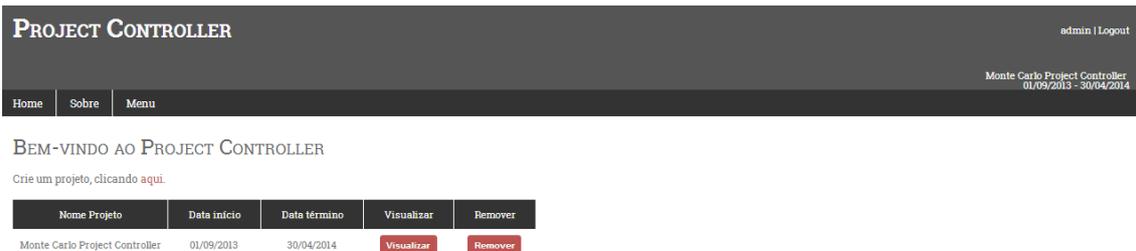
Confirm Password:  
\*\*\*\*\*

Voltar Cadastrar

**Figura 10 - Tela de Cadastro de Usuário**

### 3.5.3 Tela Principal - Projeto

Ao fazer o *login* no sistema, o usuário será direcionado para a tela principal (Figura 11). Nessa tela o menu está localizado na parte superior e abaixo são listados os projetos do usuário. Se o mesmo não tiver nenhum projeto associado, pode-se criar um novo projeto realizando um click no link “aqui” (em vermelho – Figura 11). Se o usuário já tiver um projeto associado, basta pressionar o botão visualizar (Figura 11) e o mesmo será direcionado para a tela que detalha o projeto (item 3.5.5). O usuário pode excluir o projeto clicando em remover (Figura 11).



**Figura 11 - Tela Principal de Projetos**

### 3.5.4 Cadastro de novo projeto - Projeto

Nesse item é descrita a parte de cadastramento de um projeto. Nesse caso será utilizado como exemplo o projeto “Monte Carlo Project *Controller*”. O usuário define as características do projeto como nome, data (início e fim) e descrição. Após preencher todos os dados, o usuário deve clicar em “Salvar” (em vermelho – Figura 12). Após o clique em salvar, o usuário será direcionado para a tela que detalha o projeto (item 3.5.5)

**CADASTRO DE PROJETO**

Descrição	Valor
Nome:	<input type="text" value="Monte Carlo Project Controller"/>
Data de Início:	<input type="text" value="01/09/2013"/>
Data de Término:	<input type="text" value="10/05/2014"/>
Descrição	<input type="text" value="Construção da ferramenta que através das estimativas de tempo do projeto definidas pelo usuário haja uma simulação delas gerando uma análise (gráfico de probabilidade acumulada) que permite que o usuário enxergue probabilisticamente com qual possibilidade o projeto, ou parte dele, será realizado em um determinado período de tempo de acordo com definição das estimativas."/>

Salvar e Sair

**Figura 12 - Tela de Cadastro de Novo Projeto**

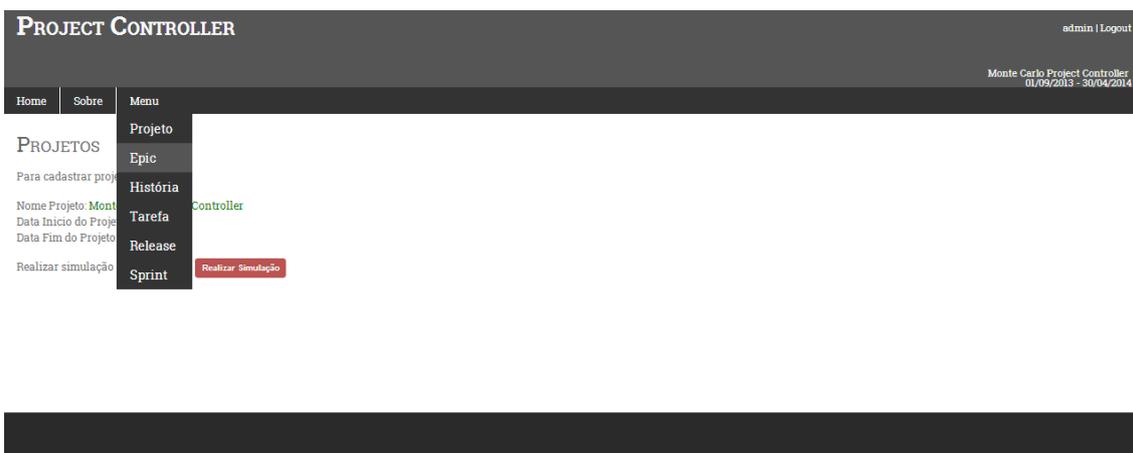
### 3.5.5 Tela principal de um Projeto – Projeto

Após a criação do projeto, o usuário será direcionado para a tela principal do projeto escolhido (Figura 13), onde o usuário pode visualizar os dados do projeto, além de poder criar um novo projeto no link “aqui” (em vermelho – figura 13) ou realizar a simulação de estimativas do projeto (botão “Realizar Simulação” – Figura 13).



**Figura 13 - Tela Principal do Projeto Escolhido**

Ainda nessa área (Figura 14), é possível acessar via menu as telas principais de *Epic*, *História*, *Tarefa*, *Release* e *Sprint* do projeto.



**Figura 14 - Menu de Opções**

### 3.5.6 Cadastro de *Release*

Acessando a tela principal de *Release* no menu (Figura 14), o usuário pode cadastrar um *Release* preenchendo um nome, data de início e data de término. Para concluir o cadastro, basta pressionar o botão “Salvar”. No exemplo ilustrado pela Figura 15, será cadastrado o release “Elaboração de Modelos Conceituais”.



The screenshot shows the 'PROJECT CONTROLLER' header with navigation links for 'Home', 'Sobre', and 'Menu'. Below this is the 'CADASTRO DE RELEASE' section. It features a table with two columns: 'Descrição' and 'Valor'. The 'Nome' field is filled with 'Elaboração dos Modelos Conceituais'. The 'Data de Início' field is filled with '01/09/2013' and the 'Data de Término' field is filled with '10/11/2013'. At the bottom, there is a 'Salvar e Sair' button and a red 'Salvar' button.

**Figura 15 – Tela de Cadastro Release**

### 3.5.7 Cadastro de *Sprint*

Acessando a tela principal de *Release* no menu (Figura 14), o usuário pode realizar a criação de *Sprint*, preenchendo o nome, a data de início, data de término. Além disso, pode-se associar um *Release* ao *Sprint*. A Figura 16 ilustra um exemplo de criação do *sprint* “Diagramas UML” associado ao release “Elaboração dos Modelos Conceituais” criado anteriormente.

Home	Sobre	Menu
------	-------	------

## CADASTRO DE SPRINT

Descrição	Valor
Nome:	<input type="text" value="Diagramas UML"/>
Data de Início:	<input type="text" value="01/09/2013"/>
Data de Término:	<input type="text" value="10/10/2013"/>
Release:	<input type="text" value="Sem Associação"/> <ul style="list-style-type: none"> <li style="background-color: #007bff; color: white; padding: 2px;">Elaboração dos Modelos Conceituais</li> <li>Implementação de Padrão Arquitetural</li> <li>Simulação de Monte Carlo</li> <li>Sem Associação</li> </ul>
<input type="button" value="Salvar"/>	

**Figura 16 - Tela de Cadastro Sprint**

### 3.5.8 *Epic*

Esta seção descreve as funcionalidades relacionadas à estrutura *Epic*. Um *Epic* corresponde a uma macro funcionalidade que tipicamente não pode ser desenvolvida em um único *Sprint*, devendo ser dividido em histórias.

- Tela Principal *Epic*

Para acessar a tela principal de gestão de *Epics*, deve-se clicar no menu principal (Figura 14). Na tela resultante, ilustrada pela Figura 17, o usuário pode cadastrar um novo *Epic* (clicando no link “Cadastrar”), importar um arquivo .txt (clicando no link “Importar *Epics*”) contendo *epics*, ou ainda, associar *epics* já existentes com releases.

BEM-VINDO A TELA INICIAL CONTENDO TODOS OS RECURSOS DOS EPICS

Cadastro de Epics: [Cadastrar](#)Importação de Epics: [Importar Epics](#)

## ASSOCIAÇÃO DE EPIC A RELEASE

Release: 

Nome	Descrição	Melhor Caso	Caso Médio	Pior Caso	Release Associada	
Diagramas UML Estático	Representar, em UML, o comportamento estatico do sistema através de um diagrama de classes.	5	7	9		<a href="#">Editar</a>
Diagrama UML Dinamico	Representar, em UML, o comportamento dinamico do sistema através de um diagrama de casos de uso	5	7	9		<a href="#">Editar</a>
Modelo Entidade-Relacionamento	Representar o modelo de banco de dados que sera utilizado	12	18	23		<a href="#">Editar</a>
Domínio de Dados	Implementar as classes de domínio do sistema	34	42	50		<a href="#">Editar</a>
Business Object das Classes de Domínio	Implementar os BOs das classes de negocio	40	47	55		<a href="#">Editar</a>

1 2

**Figura 17 - Tela Principal *Epic***

- Cadastro de *Epics*

Acessando o cadastro de Epics (opção “Cadastro de *Epics*” - Figura 17), o usuário é direcionado para a tela de cadastro de *Epics* (Figura 18). Para cadastrar um novo epic, é necessário preencher o nome, descrição, estimativa de esforço no melhor caso, caso mais provável e pior caso para a realização desse epic. A Figura 2 apresenta o exemplo de cadastro do *epic* “Diagramas UML Estático”. Uma vez preenchidos os campos, o usuário poderá clicar no botão “Salvar e Sair” para concluir o cadastro, clicar em “Salvar e Continuar” para continuar cadastrando outros epics ou voltar para a tela principal (clitando em “Voltar”).

# PROJECT CONTROLLER

Home
Sobre
Menu

Cadastrar Epics

Rótulo	Valores
Nome:	<input style="width: 90%;" type="text" value="Diagramas UML Estático"/>
Descrição:	<div style="border: 1px solid #ccc; padding: 5px; min-height: 100px;">Representar, em UML, o comportamento estático do sistema através de um diagrama de classes.</div>
Melhor Caso:	<input style="width: 90%;" type="text" value="5"/>
Caso mais Provável:	<input style="width: 90%;" type="text" value="7"/>
Pior Caso:	<input style="width: 90%;" type="text" value="9"/>
Salvar e Decompor em Histórias	<input style="background-color: #800000; color: white; padding: 5px 15px;" type="button" value="Salvar e Sair"/>
Salvar e continuar criando Epics	<input style="background-color: #800000; color: white; padding: 5px 15px;" type="button" value="Salvar e Continuar"/>
Voltar para tela inicial de Epics	<input style="background-color: #800000; color: white; padding: 5px 15px;" type="button" value="Voltar"/>

**Figura 18 - Cadastro de Epic**

- Importação de Epic

Acessando a tela de importação de epic na tela principal de Epic, o usuário pode importar um arquivo .txt contendo epics para o sistema. Para isto, o usuário deve criar um arquivo .txt no formato apresentado na Figura 19, selecionar o mesmo e clicar em “Enviar” (Figura 120).

```
importar_epic - Bloco de notas
Arquivo Editar Formatar Exibir Ajuda
Nome;Descricao;MelhorCaso;CasoMedio;PiorCaso
Diagrama UML Dinâmico;Representar, em UML, o comportamento dinâmico do sistema através de um diagrama de casos de uso;5;7;9
Modelo Entidade-Relacionamento;Representar o modelo de banco de dados que será utilizado;12;18;23
Domínio de Dados;Implementar as classes de domínio do sistema;34;42;50
Business Object das Classes de Domínio; Implementar os BOS das classes de negócio;40;47;55
Apresentação do Sistema;Implementar as páginas e controles visuais do sistema;70;84;92
Business Object relativo a Simulação de Monte Carlo;Implementar os BOS das classes relacionadas a Simulação de Monte Carlo;15;2:
Apresentação da Simulação de Monte Carlo;Implementar a página e controle relativos a simulação de Monte Carlo;18;27;33
```

**Figura 19 - Arquivo Importação Epic**



**Figura 20 - Importar Epic**

Se o arquivo não seguir o formato indicado, o sistema não permitirá a importação apresentando o erro ocorrido. Na figura 21, observa-se o exemplo de uma importação de um arquivo na extensão .sql.



**Figura 21 - Importação inválida**

- Associação de *Epic* a um Release

Na tela principal de *Epic* o usuário pode associar *epics* a *releases* clicando na opção Editar (Figura 122).

PROJECT CONTROLLER admin | Logout

Monte Carlo Project Controller  
01/09/2013 - 30/04/2014

Home Sobre Menu

BEM-VINDO A TELA INICIAL CONTENDO TODOS OS RECURSOS DOS EPICS

Cadastro de Epics: [Cadastrar](#)

Importação de Epics: [Importar Epics](#)

ASSOCIAÇÃO DE EPIC A RELEASE

Release: Sem Associação

	Descrição	Melhor Caso	Caso Médio	Pior Caso	Release Associada	
Diagramas UML Estático	Representar, em UML, o comportamento estatico do sistema através de um diagrama de classes.	5	7	9		<a href="#">Editar</a>
Diagrama UML Dinamico	Representar, em UML, o comportamento dinamico do sistema através de um diagrama de casos de uso	5	7	9		<a href="#">Editar</a>
Modelo Entidade-Relacionamento	Representar o modelo de banco de dados que sera utilizado	12	18	23		<a href="#">Editar</a>
Dominio de Dados	Implementar as classes de dominio do sistema	34	42	50		<a href="#">Editar</a>
Business Object das Classes de Dominio	Implementar os BOs das classes de negocio	40	47	55		<a href="#">Editar</a>

1 2

**Figura 22 - Associação *Epic* – *Release***

Ao clicar em “Editar” em um *Epic* da lista, os dados do *Epic* selecionado podem ser editados (Figura 22). O usuário pode modificar os valores dos atributos do *Epic* ou associar o *Epic* a um Release na lista *drop down* de releases. No exemplo da Figura 22, o *epic* “Diagramas UML Estático” será associado ao release “Elaboração dos Modelos Conceituais”. Nessa tela (Figura 23), o usuário ainda pode utilizar o filtro de exibição de *epics* por release, ou seja, exibir somente as *epics* associadas ao release escolhido.

BEM-VINDO A TELA INICIAL CONTENDO TODOS OS RECURSOS DOS EPICS

Cadastro de Epics: [Cadastrar](#)  
 Importação de Epics: [Importar Epics](#)

#### ASSOCIAÇÃO DE EPIC A RELEASE

Release:

Nome	Descrição	Melhor Caso	Caso Médio	Pior Caso	Release Associada	
Diagramas UML Estático	Representar, em UML, o c	<input type="text" value="5"/>	<input type="text" value="7"/>	<input type="text" value="9"/>	Elaboração dos Modelos Conceituais	<a href="#">Salvar</a>
Diagrama UML Dinamico	Representar, em UML, o comportamento dinamico do sistema atraves de um diagrama de casos de uso	5	7	9	Elaboração dos Modelos Conceituais	<a href="#">Editar</a>
Modelo Entidade-Relacionamento	Representar o modelo de banco de dados que sera utilizado	12	18	23	Implementação de Padrão Arquitetural	<a href="#">Editar</a>
Domínio de Dados	Implementar as classes de dominio do sistema	34	42	50	Simulação de Monte Carlo	<a href="#">Editar</a>
Business Object das Classes de Domínio	Implementar os BOs das classes de negocio	40	47	55	Sem Associação	<a href="#">Editar</a>

1 2

**Figura 23 - Editar Epic**

### 3.5.9 História

Esta seção descreve as funcionalidades relacionadas a gerência de Histórias do *Product Backlog*.

- Histórias - Tela Principal

Para acessar a tela principal de Histórias, deve-se clicar no menu principal (Figura 14). A partir dessa tela principal (Figura 24), o usuário pode cadastrar uma nova História (clicando no link “Cadastrar”), importar um arquivo .txt contendo histórias (clicando no link “Importar Histórias”) ou associar histórias existentes com *Sprints* e/ou *Epics*.

BEM-VINDO A TELA INICIAL CONTENDO TODOS OS RECURSOS DAS HISTÓRIAS

Cadastro de Histórias: [Cadastrar](#)  
Importação de Histórias: [Importar Histórias](#)

#### ASSOCIAÇÃO DE HISTÓRIA A SPRINT E EPIC

Sprint:

Nome	Descrição	Melhor Caso	Caso Médio	Pior Caso	Sprint Associado	Epic Associado	
Diagrama de Classes	Representar, em UML, as classes necessarias para o sistema funcionar.	2	4	6			<a href="#">Editar</a>
Diagrama de Casos de Uso	Representar, em UML, as funcionalidades necessarias para o sistema funcionar.	4	7	9			<a href="#">Editar</a>
Classe User	Implementar a classe de dominio relativo ao Usuario	6	8	10			<a href="#">Editar</a>
Classe Perfil	Implementar a classe de dominio relativo ao Perfil de um usuario	3	5	8			<a href="#">Editar</a>
Classe UserProject	Implementar a classe de dominio relativo ao relacionamento entre Usuario e Projeto	1	3	7			<a href="#">Editar</a>

1 2 3 4 5 6

**Figura 24 - Tela Principal História**

- Cadastro de História

Acessando o cadastro de História (opção “Cadastro de Histórias” - Figura 24), o usuário é direcionado para a tela de cadastro de História (Figura 2452). Para cadastrar uma nova história, é necessário preencher o nome, descrição, estimativa de esforço no melhor caso, caso mais provável e pior caso para a realização dessa história. Após isso o usuário poderá clicar no botão “Salvar e Sair” para concluir o cadastro, clicar em “Salvar e Continuar” para continuar cadastrando outras histórias ou voltar para a tela principal (clique em “Voltar”).

## Cadastrar História

Rótulo	Valores
Nome:	<input type="text" value="Diagrama de Classes"/>
Descrição:	<pre>Representar, em UML, as classes necessarias para o sistema funcionar.</pre>
Melhor Caso:	<input type="text" value="2"/>
Caso mais Provável:	<input type="text" value="4"/>
Pior Caso:	<input type="text" value="6"/>
Salvar e Decompor em Tarefas	<input type="button" value="Salvar e Sair"/>
Salvar e continuar criando Histórias	<input type="button" value="Salvar e Continuar"/>
Voltar para tela	<input type="button" value="Voltar"/>

**Figura 25 - Cadastro de História**

- Importação de História

Acessando a tela de importação de história na tela principal de História, o usuário pode importar para o sistema um arquivo .txt contendo histórias. Para isto, o usuário deve criar um arquivo .txt no formato apresentado na Figura 26 e importá-lo utilizando a tela apresentada na Figura 27.

```

Nome;Descricao;MelhorCaso;CasoMedio;PiorCaso
Diagrama de Casos de Uso;Representar, em UML, as funcionalidades necessarias para o sistema funcionar;4;7;9
Classe User;Implementar a classe de dominio relativo ao Usuario;6;8;10
Classe Perfil;Implementar a classe de dominio relativo ao Perfil de um usuário;3;5;8
Classe UserProject;Implementar a classe de dominio relativo ao relacionamento entre Usuário e Projeto;1;3;7
Classe Release;Implementar a classe de dominio relativo ao Release;8;13;17
Classe Sprint;Implementar a classe de dominio relativo ao Sprint;4;6;9
Classe Project;Implementar a classe de dominio relativo ao Projeto;8;12;17
Classe Epic;Implementar a classe de dominio relativo a Epic;5;8;10
Classe History;Implementar a classe de dominio relativo a Historia;4;8;10
Classe Task;Implementar a classe de dominio relativo a Tarefa;6;8;10
Classe UserBO;Implementar a classe que contem as regras de negócio do usuario;6;8;10
Classe PerfilBO;Implementar a classe que contem as regras de negócio relativo ao Perfil de um usuário;3;5;8
Classe UserProjectBO;Implementar a classe que contem as regras de negócio relativo ao relacionamento entre usuário e Projeto;1;3;7
Classe ReleaseBO;Implementar a classe que contem as regras de negócio relativo ao Release;8;13;17
Classe SprintBO;Implementar a classe que contem as regras de negócio relativo ao Sprint;4;6;9
Classe ProjectBO;Implementar a classe que contem as regras de negócio relativo ao Projeto;8;12;17
Classe EpicBO;Implementar a classe que contem as regras de negócio relativo a Epic;5;8;10
Classe HistoryBO;Implementar a classe que contem as regras de negócio relativo a Historia;4;8;10
Classe TaskBO;Implementar a classe que contem as regras de negócio relativo a Tarefa;6;8;10
Apresentacao de Usuario;Implementar as telas de apresentacao relativo ao Usuario;6;8;10
Apresentacao de Release;Implementar as telas de apresentacao relativo ao Release;8;13;17
Apresentacao de Sprint;Implementar as telas de apresentacao relativo ao Sprint;4;6;9
Apresentacao de Projeto;Implementar as telas de apresentacao relativo ao Projeto;8;12;17
Apresentacao de Epic;Implementar as telas de apresentacao relativo a Epic;5;8;10
Apresentacao de Historia;Implementar as telas de apresentacao relativo a Historia;4;8;10
Apresentacao de Tarefa;Implementar as telas de apresentacao relativo a Tarefa;6;8;10

```

**Figura 26 - Arquivo Importação História**



**Figura 27 - Importar História**

- Associar História a um *Sprint* e/ou *Epic*

Na tela principal de História, o usuário pode associar histórias a *sprints* e/ou *epics* existentes. Para tal, basta clicar no botão “Editar” correspondente à história desejada (Figura 28) e editar as informações da história (Figura 29). O usuário pode modificar os valores dos atributos da história ou associar a história a um *sprint* e/ou *epic* nos respectivos campos de seleção. No exemplo ilustrado na Figura 29, a história “Diagrama de Classes” será associada ao *Sprint* “Diagramas UML” e ao *Epic* “Diagramas UML Estático”. Nessa tela de histórias, o usuário também pode utilizar o filtro de exibição de histórias por *sprint* para exibir somente as histórias associadas ao *sprint* escolhido.

BEM-VINDO A TELA INICIAL CONTENDO TODOS OS RECURSOS DAS HISTÓRIAS

Cadastro de Histórias: [Cadastrar](#)  
 Importação de Histórias: [Importar Histórias](#)

ASSOCIAÇÃO DE HISTÓIA A SPRINT E EPIC

Sprint:

	Descrição	Melhor Caso	Caso Médio	Pior Caso	Sprint Associado	Epic Associado	
Diagrama de Classes	Representar, em UML, as classes necessarias para o sistema funcionar.	2	4	6			<a href="#">Editar</a>
Diagrama de Casos de Uso	Representar, em UML, as funcionalidades necessarias para o sistema funcionar	4	7	9			<a href="#">Editar</a>
Classe User	Implementar a classe de dominio relativo ao Usuario	6	8	10			<a href="#">Editar</a>
Classe Perfil	Implementar a classe de dominio relativo ao Perfil de um usuario	3	5	8			<a href="#">Editar</a>
Classe UserProject	Implementar a classe de dominio relativo ao relacionamento entre Usuario e Projeto	1	3	7			<a href="#">Editar</a>

1 2 3 4 5 6

Figura 28 - Associação História – Sprint e/ou Epic

BEM-VINDO A TELA INICIAL CONTENDO TODOS OS RECURSOS DAS HISTÓRIAS

Cadastro de Histórias: [Cadastrar](#)  
 Importação de Histórias: [Importar Histórias](#)

ASSOCIAÇÃO DE HISTÓIA A SPRINT E EPIC

Sprint:

Nome	Descrição	Melhor Caso	Caso Médio	Pior Caso	Sprint Associado	Epic Associado	
Diagrama de Classes	Representar, em UML,	2	4	6	Implementacao das Classes de Negocio	Diagrama UML Dinamico	<a href="#">Salvar</a>
Diagrama de Casos de Uso	Representar, em UML, as funcionalidades necessarias para o sistema funcionar	4	7	9		Diagramas UML Estáticos	<a href="#">Editar</a>
Classe User	Implementar a classe de dominio relativo ao Usuario	6	8	10		Diagrama UML Dinamico	<a href="#">Editar</a>
Classe Perfil	Implementar a classe de dominio relativo ao Perfil de um usuario	3	5	8		Modelo Entidade-Relacionamento	<a href="#">Editar</a>
Classe UserProject	Implementar a classe de dominio relativo ao relacionamento entre Usuario e Projeto	1	3	7		Domínio de Dados	<a href="#">Editar</a>
						Business Object das Classes de Dominio	<a href="#">Editar</a>
						Apresentacao do Sistema	<a href="#">Editar</a>
						Business Object relativo a Simulacao de Monte Carlo	<a href="#">Editar</a>
						Apresentacao da Simulacao de Monte Carlo	<a href="#">Editar</a>
						Sem Associação	<a href="#">Editar</a>

1 2 3 4 5 6

Figura 29 - Editar História

### 3.5.10 Tarefa

Esta seção descreve as funcionalidades relacionadas à gestão de Tarefas.

- Tarefa - Tela Principal

Para acessar a tela principal de Tarefa, deve-se clicar na opção Tarefas no menu principal (Figura 14). Na tela de tarefas (Figura 30), o usuário pode cadastrar uma nova Tarefa (clcando no link “Cadastrar”), importar para o sistema um arquivo .txt (clcando no link “Importar Tarefas”) contendo tarefas ou associar tarefas existentes com *sprints* e/ou histórias.

BEM-VINDO A TELA INICIAL CONTENDO TODOS OS RECURSOS DAS TASKS

Cadastro de Task: [Cadastrar](#)  
Importação de Tasks: [Importar Tasks](#)

#### ASSOCIAÇÃO DE TAREFA A SPRINT E HISTÓRIA

Sprint:

Nome	Descrição	Melhor Caso	Caso Médio	Pior Caso	Sprint Associado	História Associada	
Definição das classes	Definir as classes que serão modeladas	2	4	6			<a href="#">Editar</a>
Definição dos atributos	Definir os atributos das classes modeladas	4	7	9			<a href="#">Editar</a>
Definição dos relacionamentos e das cardinalidades	Definir os relacionamentos e as cardinalidades entre eles, das classes modeladas	5	8	11			<a href="#">Editar</a>
Definição do escopo dos casos de uso	Definir os o que os casos de uso representarão no cenário principal	4	7	9			<a href="#">Editar</a>
Avaliar os casos de uso de extensão e de inclusão	Definir os casos de uso que são de extensão e de inclusão	2	4	6			<a href="#">Editar</a>

1 2 3 4 5

**Figura 30 - Tela Principal Tarefa**

- Cadastro de Tarefa

Acessando o cadastro de Tarefa (opção “Cadastro de Tarefas” - Figura 30), o usuário é direcionado para a tela de cadastro de Tarefa (Figura 31). Para cadastrar uma nova tarefa, é necessário preencher o nome, descrição, estimativa de esforço no melhor caso, caso mais provável e pior caso na realização dessa tarefa. Após isso o usuário poderá clicar no botão “Salvar e Sair” para concluir o cadastro, clicar em “Salvar e Continuar” para continuar cadastrando outras tarefas ou voltar para a tela principal (clique em “Voltar”). No exemplo ilustrado pela Figura 28, será cadastrada a tarefa “Definição de Classes”.

# PROJECT CONTROLLER

Home
Sobre
Menu

## Cadastro de Tarefa

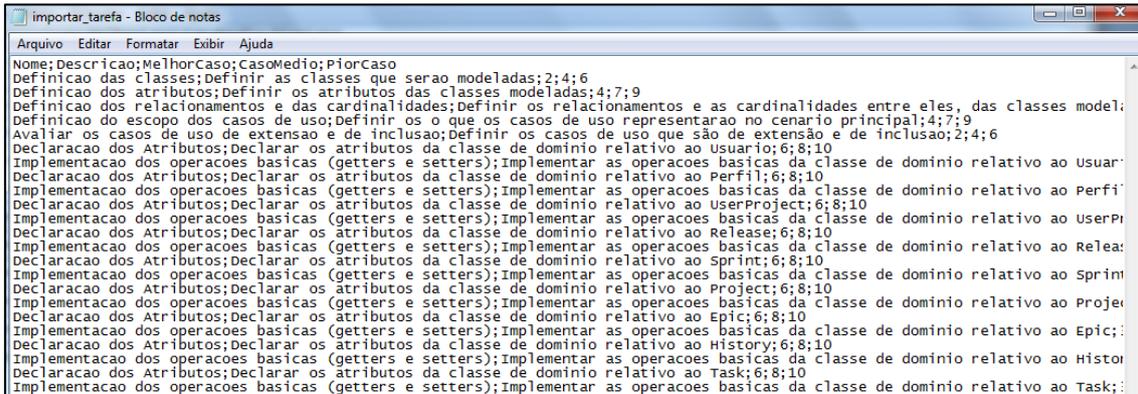
Descrição	Valor
Nome	<input type="text" value="Definicao das classes"/>
Descrição da Tarefa	<div style="border: 1px solid #ccc; padding: 5px; min-height: 100px;">Definir as classes que <u>serao</u> modeladas.</div>
Melhor Caso	<input type="text" value="2"/>
Caso mais Provável	<input type="text" value="4"/>
Pior Caso	<input type="text" value="6"/>
Salvar e Sair	<input type="button" value="Salvar e Sair"/>
Salvar e continuar criando Tasks	<input type="button" value="Salvar e Continuar"/>
Voltar para tela inicial de Tasks	<input type="button" value="Voltar"/>

**Figura 31 - Cadastro de Tarefa**

- Importação de Tarefa
 

Acessando a tela de importação de tarefa na tela principal de Tarefas, o usuário pode importar para o sistema um arquivo .txt contendo tarefas. Para isso,

o usuário deve criar um arquivo .txt no seguinte formato (Figura 32) e enviar o arquivo na tela apresentada na Figura 33.



```
Arquivo Editar Formatar Exibir Ajuda
Nome;Descricao;MelhorCaso;CasoMedio;PiorCaso
Definicao das classes;Definir as classes que serao modeladas;2;4;6
Definicao dos atributos;Definir os atributos das classes modeladas;4;7;9
Definicao dos relacionamentos e das cardinalidades;Definir os relacionamentos e as cardinalidades entre eles, das classes modeladas;4;7;9
Avaliar os casos de uso de extensao e de inclusao;Definir os casos de uso que são de extensao e de inclusao;2;4;6
Declaracao dos Atributos;Declarar os atributos da classe de dominio relativo ao Usuario;6;8;10
Implementacao dos operacoes basicas (getters e setters);Implementar as operacoes basicas da classe de dominio relativo ao Usuario;6;8;10
Declaracao dos Atributos;Declarar os atributos da classe de dominio relativo ao Perfil;6;8;10
Implementacao dos operacoes basicas (getters e setters);Implementar as operacoes basicas da classe de dominio relativo ao Perfil;6;8;10
Declaracao dos Atributos;Declarar os atributos da classe de dominio relativo ao UserProject;6;8;10
Implementacao dos operacoes basicas (getters e setters);Implementar as operacoes basicas da classe de dominio relativo ao UserProject;6;8;10
Declaracao dos Atributos;Declarar os atributos da classe de dominio relativo ao Release;6;8;10
Implementacao dos operacoes basicas (getters e setters);Implementar as operacoes basicas da classe de dominio relativo ao Release;6;8;10
Declaracao dos Atributos;Declarar os atributos da classe de dominio relativo ao Sprint;6;8;10
Implementacao dos operacoes basicas (getters e setters);Implementar as operacoes basicas da classe de dominio relativo ao Sprint;6;8;10
Declaracao dos Atributos;Declarar os atributos da classe de dominio relativo ao Project;6;8;10
Implementacao dos operacoes basicas (getters e setters);Implementar as operacoes basicas da classe de dominio relativo ao Project;6;8;10
Declaracao dos Atributos;Declarar os atributos da classe de dominio relativo ao Epic;6;8;10
Implementacao dos operacoes basicas (getters e setters);Implementar as operacoes basicas da classe de dominio relativo ao Epic;6;8;10
Declaracao dos Atributos;Declarar os atributos da classe de dominio relativo ao History;6;8;10
Implementacao dos operacoes basicas (getters e setters);Implementar as operacoes basicas da classe de dominio relativo ao History;6;8;10
Declaracao dos Atributos;Declarar os atributos da classe de dominio relativo ao Task;6;8;10
Implementacao dos operacoes basicas (getters e setters);Implementar as operacoes basicas da classe de dominio relativo ao Task;6;8;10
```

**Figura 32 - Arquivo Importação Tarefa**



**Figura 33 - Importar Tarefa**

- Associar Tarefa à *Sprint* e/ou História

Na tela principal de Tarefas, o usuário pode associar tarefas a *sprints* e/ou histórias criadas. Para isso, basta clicar no botão “Editar” correspondente à tarefa desejada (Figura 34). Após clicar em “Editar”, o usuário pode modificar os valores dos atributos da tarefa ou associar a tarefa a um *sprint* e/ou história nas caixas de seleção (Figura 35). Nessa tela, o usuário também pode utilizar o filtro de exibição de tarefa por *sprint* para exibir somente as tarefas associadas ao *sprint* escolhido. No exemplo ilustrado pela Figura 35, a tarefa “Definição das Classes” será associada ao *sprint* “Diagramas UML” e à história “Diagrama de Classes”.

BEM-VINDO A TELA INICIAL CONTENDO TODOS OS RECURSOS DAS TASKS

Cadastro de Task: [Cadastrar](#)  
 Importação de Tasks: [Importar Tasks](#)

ASSOCIAÇÃO DE TAREFA A SPRINT E HISTÓRIA

Sprint:

	Descrição	Melhor Caso	Caso Médio	Pior Caso	Sprint Associado	História Associada	
Definicao das classes	Definir as classes que serao modeladas	2	4	6	Diagramas UML		<a href="#">Editar</a>
Definicao dos atributos	Definir os atributos das classes modeladas	4	7	9			<a href="#">Editar</a>
Definicao dos relacionamentos e das cardinalidades	Definir os relacionamentos e as cardinalidades entre eles, das classes modeladas	5	8	11			<a href="#">Editar</a>
Definicao do escopo dos casos de uso	Definir os o que os casos de uso representaro no cenario principal	4	7	9			<a href="#">Editar</a>
Avaliar os casos de uso de extensao e de inclusao	Definir os casos de uso que sao de extensao e de inclusao	2	4	6			<a href="#">Editar</a>

1 2 3 4 5

Figura 34 - Associação Tarefa – Sprint e/ou História

PROJECT CONTROLLER admin | Logout

Monte Carlo Project Controller  
01/09/2013 - 30/04/2014

Home Sobre Menu

BEM-VINDO A TELA INICIAL CONTENDO TODOS OS RECURSOS DAS TASKS

Cadastro de Task: [Cadastrar](#)  
 Importação de Tasks: [Importar Tasks](#)

ASSOCIAÇÃO DE TAREFA A SPRINT E HISTÓRIA

Sprint:

Nome	Descrição	Melhor Caso	Caso Médio	Pior Caso	Sprint Associado	
Definicao das classes	Definir as classes que se	<input type="text" value="2"/>	<input type="text" value="4"/>	<input type="text" value="6"/>	Diagramas UML	<a href="#">Salvar</a>
Definicao dos atributos	Definir os atributos das classes modeladas	4	7	9		<a href="#">Editar</a>
Definicao dos relacionamentos e das cardinalidades	Definir os relacionamentos e as cardinalidades entre eles, das classes modeladas	5	8	11		<a href="#">Editar</a>
Definicao do escopo dos casos de uso	Definir os o que os casos de uso representaro no cenario principal	4	7	9		<a href="#">Editar</a>
Avaliar os casos de uso de extensao e de inclusao	Definir os casos de uso que sao de extensao e de inclusao	2	4	6		<a href="#">Editar</a>

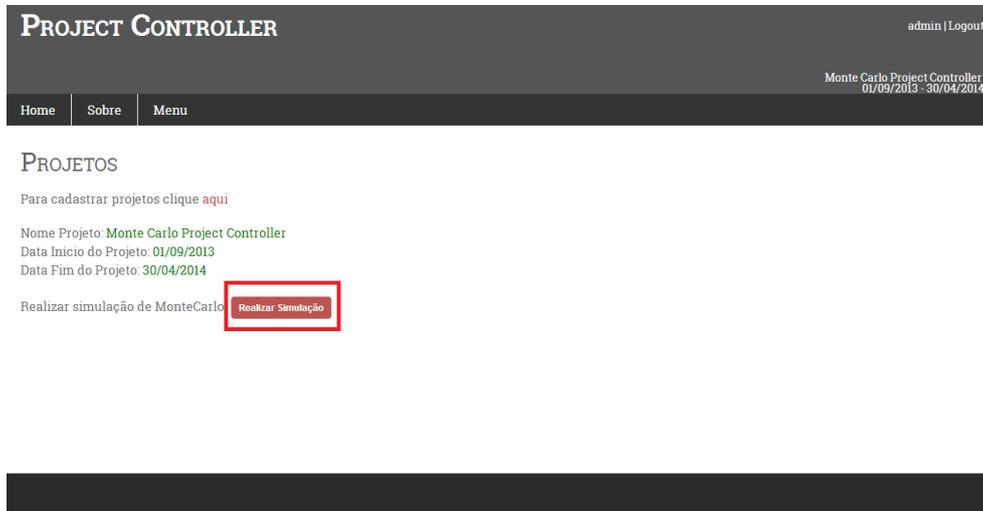
1 2 3 4 5

Diagrama de Classes  
 Diagrama de Casos de Uso  
 Classe User  
 Classe Perfil  
 Classe UserProject  
 Classe Release  
 Classe Sprint  
 Classe Project  
 Classe Epic  
 Classe History  
 Classe Task  
 Classe UserBO  
 Classe PerfilBO  
 Classe UserProjectBO  
 Classe ReleaseBO  
 Classe SprintBO  
 Classe ProjectBO  
 Classe EpicBO  
 Classe HistoryBO  
 Classe TaskBO

Figura 35 - Editar Tarefa

### 3.5.11 Simulação de Estimativas

Para se executar a simulação de Monte Carlos, o usuário deve clicar “Realizar Simulação” na tela principal de projetos (Figura 36).



**Figura 36 - Tela Inicial de Projeto, destacando como realizar a simulação**

A Figura 37 apresenta a tela onde o usuário pode selecionar opções para a realização da simulação. Pode-se optar por realizar a simulação por requisitos/atividades (selecionando *Epics*, *Histórias* e *Tarefas*) ou por fases/iterações do projeto (selecionando *Releases* ou *Sprints*).



**Figura 37 - Tela Inicial de Simulação**

Para realizar a simulação com base nos requisitos/atividades, o usuário seleciona na caixa de seleção o tipo de elemento a ser utilizado (*Epic*, *História*

ou Tarefa). O sistema apresenta a relação dos elementos do tipo selecionado cadastrados no sistema. A Figura 38 apresenta um exemplo onde são listados os *Epics* com as suas respectivas estimativas e os Releases respectivamente associados.

Monte Carlo Project Controller  
01/09/2013 - 30/04/2014
Home | Sobre | Menu

---

PÁGINA DE SIMULAÇÃO DE MONTE CARLO - MONTE CARLO PROJECT CONTROLLER

Epics OK

Escreva a quantidade de iterações que deseja:  Simular

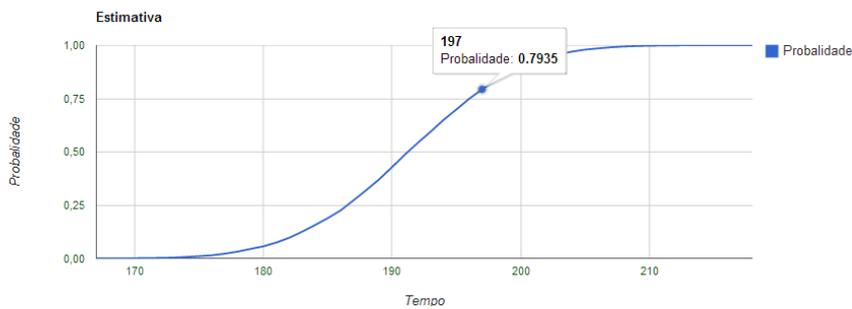
	Nome	Melhor Caso	Mais Provável	Pior Caso	Associado	
<input type="checkbox"/>	Diagramas UML Estático	5	7	9	Elaboração dos Modelos Conceituais	<span style="background-color: #f00; color: white; padding: 2px 5px;">Editar</span>
<input type="checkbox"/>	Diagrama UML Dinamico	5	7	9	Elaboração dos Modelos Conceituais	<span style="background-color: #f00; color: white; padding: 2px 5px;">Editar</span>
<input type="checkbox"/>	Modelo Entidade-Relacionamento	12	18	23	Elaboração dos Modelos Conceituais	<span style="background-color: #f00; color: white; padding: 2px 5px;">Editar</span>
<input type="checkbox"/>	Dominio de Dados	34	42	50	Implementação de Padrão Arquitetural	<span style="background-color: #f00; color: white; padding: 2px 5px;">Editar</span>
<input type="checkbox"/>	Business Object das Classes de Dominio	40	47	55	Implementação de Padrão Arquitetural	<span style="background-color: #f00; color: white; padding: 2px 5px;">Editar</span>
<input type="checkbox"/>	Apresentacao do Sistema	70	84	92		<span style="background-color: #f00; color: white; padding: 2px 5px;">Editar</span>
<input type="checkbox"/>	Business Object relativo a Simulacao de Monte Carlo	15	23	29	Simulação de Monte Carlo	<span style="background-color: #f00; color: white; padding: 2px 5px;">Editar</span>
<input type="checkbox"/>	Apresentacao da Simulacao de Monte Carlo	18	27	33		<span style="background-color: #f00; color: white; padding: 2px 5px;">Editar</span>

**Figura 38 - Tela de Simulação, com as *epics* listadas**

O usuário seleciona os itens da lista que deverão ser considerados na simulação e informa a quantidade de interações que devem ser realizadas para gerar o gráfico de distribuição de probabilidade acumulada. O sistema apresenta o gráfico resultante, conforme ilustrado na Figura 39. A partir desse gráfico, o usuário pode analisar os resultados obtidos e avaliar outras alternativas inserindo ou removendo itens que serão considerados no planejamento e rodando novamente a simulação. No exemplo da Figura 39, o conjunto de *Epics* selecionados tem 79,35% de probabilidade de ser concluído em até 197 unidades de tempo (dias ou horas ideais, por exemplo).

Escreva a quantidade de iterações que deseja:

<input checked="" type="checkbox"/>	Nome	Melhor Caso	Mais Provável	Pior Caso	Associado	<input type="button" value="Editar"/>
<input type="checkbox"/>	Diagramas UML Estático	5	7	9	Elaboração dos Modelos Conceituais	<input type="button" value="Editar"/>
<input type="checkbox"/>	Diagrama UML Dinamico	5	7	9	Elaboração dos Modelos Conceituais	<input type="button" value="Editar"/>
<input type="checkbox"/>	Modelo Entidade-Relacionamento	12	18	23	Elaboração dos Modelos Conceituais	<input type="button" value="Editar"/>
<input checked="" type="checkbox"/>	Domínio de Dados	34	42	50	Implementação de Padrão Arquitetural	<input type="button" value="Editar"/>
<input checked="" type="checkbox"/>	Business Object das Classes de Domínio	40	47	55	Implementação de Padrão Arquitetural	<input type="button" value="Editar"/>
<input checked="" type="checkbox"/>	Apresentacao do Sistema	70	84	92		<input type="button" value="Editar"/>
<input checked="" type="checkbox"/>	Business Object relativo a Simulacao de Monte Carlo	15	23	29	Simulação de Monte Carlo	<input type="button" value="Editar"/>
<input type="checkbox"/>	Apresentacao da Simulacao de Monte Carlo	18	27	33		<input type="button" value="Editar"/>



**Figura 39 - Tela da curva de probabilidade acumulada x tempo, destacando o ponto com a probabilidade para ser realizado no determinado tempo**

A realização da simulação também pode ser realizada tendo como referência um release ou *sprint*, conforme opção feita pelo usuário na caixa de seleção (Figura 40). O usuário informa uma data de referência e o sistema apresenta todos os releases ou *sprints*, conforme o tipo selecionado, cujo período (data início - data término) englobem a data informada. No exemplo apresentado na Figura 38, a parte superior da tela apresenta os *sprints* compreendidos no período definido a partir da data informada pelo usuário.

PÁGINA DE SIMULAÇÃO DE MONTE CARLO - MONTE CARLO PROJECT CONTROLLER

Sprint

Selecione a Data, a partir da qual irá se considerar as atividades:

Escreva a quantidade de iterações que deseja:

Maio 2014						
D	S	T	Q	Q	S	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31



**Figura 40 - Tela de escolha de atividades temporais, destacando a data a partir da qual serão considerados**

O usuário escolhe um elemento da lista (*sprint* ou *release*) e seleciona o botão "Visualizar" correspondente. Em seguida, o usuário opta pelo tipo dos elementos associados ao *sprint* ou release que serão considerados na simulação, ou seja, neste momento o usuário pode optar por utilizar *epics*, histórias ou tarefas. No exemplo da Figura 341, o usuário selecionou o *sprint* "Implementação de Classes de Negocio" e, em seguida, optou pela utilização de histórias (em amarelo na figura 41).

Monte Carlo Project Controller  
01/09/2013 - 30/04/2014

Home Sobre Menu

PÁGINA DE SIMULAÇÃO DE MONTE CARLO - MONTE CARLO PROJECT CONTROLLER

Sprint

Selecione a Data, a partir da qual irá se considerar as atividades:

Nome	Data Inicial	Data Término	Visualizar
Diagramas UML	01/09/2013	30/10/2013	<input type="button" value="Visualizar"/>
Implementação das Classes de Negocio	01/11/2013	26/12/2013	<input type="button" value="Visualizar"/>

História

Escreva a quantidade de iterações que deseja:

#	Nome	Melhor Caso	Mais Provável	Pior Caso	Associado	Ações
01	Classe User	5	8	10	Implementação das Classes de Negocio	<input type="button" value="Editar"/>
02	Classe Perfil	3	5	8	Implementação das Classes de Negocio	<input type="button" value="Editar"/>
03	Classe UserProject	1	3	7	Implementação das Classes de Negocio	<input type="button" value="Editar"/>
04	Classe Release	8	13	17	Implementação das Classes de Negocio	<input type="button" value="Editar"/>
05	Classe Sprint	4	6	9		<input type="button" value="Editar"/>
06	Classe Project	8	12	17		<input type="button" value="Editar"/>
07	Classe Epic	5	8	10		<input type="button" value="Editar"/>

**Figura 41 - Tela com as histórias associadas ao Sprint e sem associação**

O sistema apresenta uma lista de histórias, onde as primeiras histórias listadas são aquelas que estão associadas ao *Sprint* selecionado. O restante da lista apresenta histórias ainda não associadas a nenhum *Sprint*. O objetivo é permitir a avaliação do esforço das histórias associadas ao *Sprint* em relação ao intervalo de tempo determinado pelo *Sprint*. Neste momento, o usuário está avaliando a probabilidade de uma determinada combinação de histórias poder ser realizada no prazo determinado pelo *Sprint*, o que poderá fazê-lo experimentar diferentes cenários, acrescentando ou removendo histórias ao escopo do *Sprint*.

Observe que o gráfico produzido (Figura 42) é similar ao obtido anteriormente com a seleção de *Epics* (Figura 38). Entretanto, a avaliação feita por meio de *releases/sprints* guia o processo de seleção dos itens envolvidos na simulação (*epics*, histórias ou tarefas) a partir de uma data de referência, a partir da qual seleciona-se um intervalo de tempo planejado no projeto (*release* ou *sprint*), permitindo a seleção dos elementos em uma lista onde o usuário consegue visualizar rapidamente aqueles que estão associados ao intervalo selecionado (*release* ou *sprint*) daqueles que ainda não tenham sido planejados ou associados a uma *sprint* ou *release* específico do projeto.

## PÁGINA DE SIMULAÇÃO DE MONTE CARLO - MONTE CARLO PROJECT CONTROLLER

Sprint

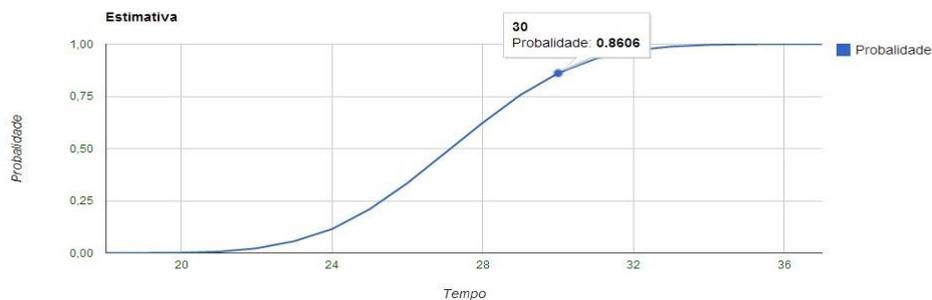
Selecione a Data, a partir da qual irá se considerar as atividades:

Nome	Data Início	Data Término	Visualizar
Diagramas UML	01/09/2013	10/10/2013	<input type="button" value="Visualizar"/>
Implementacao das Classes de Negocio	11/11/2013	29/12/2013	<input type="button" value="Visualizar"/>

Historia

Escreva a quantidade de iterações que deseja:

<input type="checkbox"/>	Nome	Melhor Caso	Mais Provável	Pior Caso	Associado	<input type="button" value="Editar"/>
<input checked="" type="checkbox"/>	Classe User	6	8	10	Implementacao das Classes de Negocio	<input type="button" value="Editar"/>
<input checked="" type="checkbox"/>	Classe Perfil	3	5	8	Implementacao das Classes de Negocio	<input type="button" value="Editar"/>
<input checked="" type="checkbox"/>	Classe UserProject	1	3	7	Implementacao das Classes de Negocio	<input type="button" value="Editar"/>
<input checked="" type="checkbox"/>	Classe Release	8	13	17	Implementacao das Classes de Negocio	<input type="button" value="Editar"/>
<input type="checkbox"/>	Classe Sprint	4	6	9		<input type="button" value="Editar"/>
<input type="checkbox"/>	Classe SprintBO	4	6	9		<input type="button" value="Editar"/>
<input type="checkbox"/>	Classe ProjectBO	8	12	17		<input type="button" value="Editar"/>
<input type="checkbox"/>	Classe EpicBO	5	8	10		<input type="button" value="Editar"/>
<input type="checkbox"/>	Classe HistoryBO	4	8	10		<input type="button" value="Editar"/>
<input type="checkbox"/>	Classe TaskBO	6	8	10		<input type="button" value="Editar"/>
<input type="checkbox"/>	Apresentacao de Usuario	6	8	10		<input type="button" value="Editar"/>
<input type="checkbox"/>	Apresentacao de Release	8	13	17		<input type="button" value="Editar"/>
<input type="checkbox"/>	Apresentacao de Sprint	4	6	9		<input type="button" value="Editar"/>
<input type="checkbox"/>	Apresentacao de Projeto	8	12	17		<input type="button" value="Editar"/>
<input type="checkbox"/>	Apresentacao de Epic	5	8	10		<input type="button" value="Editar"/>
<input type="checkbox"/>	Apresentacao de Historia	4	8	10		<input type="button" value="Editar"/>
<input type="checkbox"/>	Apresentacao de Tarefa	6	8	10		<input type="button" value="Editar"/>



**Figura 42 - Tela com curva de probabilidade acumulada x tempo para as Histórias do Sprint Selecionado**

## 4 Conclusão

### 4.1 Contribuições

Este projeto produziu uma ferramenta de software para auxiliar o processo de planejamento de *releases* e *sprints*, no contexto de projetos baseados no método SCRUM, por meio de uma análise probabilística operacionalizada pelo método de simulação de Monte Carlo. Ao invés de utilizar estimativas pontuais, o planejamento passa a usar uma estimativa de 3 pontos e, assumindo uma distribuição triangular de probabilidade, o sistema gera a distribuição de probabilidade acumulada em um número significativo de cenários gerados pelo método. O sistema proposto permite a avaliação de diferentes opções de alocação de itens do *Product Backlog* a um *Sprint* ou *Release* que está sendo planejado, de forma similar a uma análise E-SE, oferecendo informações que visam apoiar um planejamento mais realístico dos compromissos que podem ser assumidos em cada marco do projeto.

### 4.2 Limitações

O trabalho apresenta limitações, dentre as quais podemos destacar:

- O sistema não valida se as estimativas dos itens “filhos” são superiores às estimativas dos itens “pais”. Por exemplo, se a soma das estimativas de histórias associadas a um *epic* for superior à estimativa do *epic* à qual elas estão associadas;
- O sistema não permite a remoção de itens de modo a evitar problemas de remoção de itens já realizados ou a inserção em *sprints* ou releases posteriores à data corrente. Por exemplo, o sistema não permite inserir tarefas em um *Sprint* que já foi finalizado;

- O Sistema não controla diferentes tipos de usuários bem como suas horas de trabalho;
- O sistema não possui uma funcionalidade de exportação da simulação de Monte Carlo.

### 4.3 Trabalhos Futuros

Neste tópico destacam-se algumas evoluções interessantes decorrentes deste trabalho, que pode ser encontrado no endereço <https://code.google.com/p/project-management-system-monte-carlo-sim/source/checkout> onde pode-se fazer o download do projeto para a realização de trabalhos futuros:

- A implementação de um controle de usuários, por meio do qual os gerentes de projetos poderiam cadastrar os funcionários, de modo que estes pudessem fornecer as suas estimativas;
- A implementação de um controle mais robusto de releases e *sprints*, onde as datas e o tempo disponível sempre sejam respeitados;
- A implementação da importação das tarefas, histórias e *epics* via arquivo em formato XML.
- A implementação de uma funcionalidade para verificar se as tarefas filhas estão sendo mais complexas que a pai, permitindo que o gerente possa atualizar a tarefa pai dinamicamente ou rever as tarefas filhas;

### 4.4 Considerações Finais

A proposta da ferramenta descrita neste trabalho é uma forma interessante e satisfatória para o auxílio do planejamento da distribuição de atividades de um projeto ou parte dele, mesmo existindo pontos que precisam ser aperfeiçoados. Este trabalho nos proporcionou um grande aprendizado de tecnologias como Entity Framework, C#, .Net e GoogleChart. Além disso, o uso da técnica de Simulação de Monte Carlo nos abriu novos horizontes na área de gerência de projetos e na forma de tratar o problema de estimativas em planejamento de projetos.

## Referências Bibliográficas

- ABRAHAMSSON, P., WARSTA, J., SIPONEN, M. T., RONKAINEN, J., “New directions on agile methods: a comparative analysis” (2003), Proceedings of the 25th International Conference on Software Engineering (ICSE-03), IEEE Computer society, 2003. Retirado de <[http://agile.vtt.fi/docs/publications/2003/2003\\_icse03\\_new\\_directions\\_on\\_agile\\_methods.pdf](http://agile.vtt.fi/docs/publications/2003/2003_icse03_new_directions_on_agile_methods.pdf)>. Acessado em: 26 jun. 2011.
- BECK, K., BEEDLE, M., van BENNEKUN, A., COCKBURN, W., FOWLER, M., GRENNING, J., HIGHSMITH, J., HUNT, A., JEFFRIES, R., KERN, J., MARICK, B., MARTIN, R. C., MELLOR, S., SCHWABER, K., SUTHERLAND, J., THOMAS, D., “**Manifesto for Agile Software Development**”. Retirado de <<http://agilemanifesto.org/iso/ptbr/>>. Acessado em: 26 jun. 2011.
- VASCO, C.G., VITHOFT, M. H., ESTANTE, P.R., “**Comparação entre Metodologias RUP e XP**”. Curitiba, PUCPR, 2006.
- SCHWABER, K., “**Agile Project Management with Scrum**”, Microsoft Press, Redmond, WA, 2004.
- SCHWABER, K., “**Scrum Development Process**,” OOPSLA'95 Workshop on Business Object Design and Implementation, 1995.
- BASILI, V.R., CALDIERA G., ROMBACH H.D., “**Goal Question Metric Paradigm**”, Encyclopedia of Software Engineering, 2 Volume Set, John Wiley & Sons, Inc, 1994.
- GOMES, A., OLIVEIRA, K., ROCHA, A. R., “**Avaliação de Processos de Software Baseada em Medições**”, XV Simpósio Brasileiro de Engenharia de Software, Rio de Janeiro, p. 84 - 99, 2001.

- COHN, M., “**User Stories Applied: For Agile Software Development**”, New York: Addison-Wesley, 2004.
- CARLETON, A.D., FLORAC, W. A., “**Measuring the Software Process: Statistical Process Control for Software Process Improvement**”, 1st ed. Addison-Wesley Professional, 1999. 250p.
- HIGHSMITH, J. A., “**Adaptive Software Development: A Collaborative Approach to Managing Complex Systems**”, New York, NY: Dorset House Publishing, 2000.
- COCKBURN, A., “**Agile Software Development**”, Boston: Addison-Wesley, 2002.
- AMBLER, S., “**Agile Modeling: Effective Practices for Extreme Programming and the Unified Process**”, New York: John Wiley & Sons, Inc. New York, 2002.
- STAPLETON, J., “**Dynamic systems development method**”, The method in practice: Addison Wesley, 1997.
- BECK, K., “**Extreme Programming Explained: Embrace Change**”, 2000.
- PALMER, S., R., FELSING, J., M., “**A Practical Guide to Feature-Driven Development**”, 2002.
- BERGHOUT, E., SOLLINGEN, R., “**The Goal / Question / Metric Method: A practical Guide for Quality Improvement of Software Development.**”, London: McGraw-Hill. 1999. 195 p.
- Eckhardt, R., (1987). “**Stan Ulam, John von Neumann, and the Monte Carlo method**”. *Los Alamos Science, Special Issue* (15): 131–137. Retirado de <[http://en.wikipedia.org/wiki/Monte\\_Carlo\\_method](http://en.wikipedia.org/wiki/Monte_Carlo_method)> Acessado em 22/03/2014. Documento original em <[http://library.sciencemadness.org/lanl1\\_a/lib-www/pubs/00326867.pdf](http://library.sciencemadness.org/lanl1_a/lib-www/pubs/00326867.pdf)> Acessado em 22/03/2014.

Fishman, G.S. (1996) **“Monte Carlo, Concept, Algorithms, and Applications”**, Springer, New York

Teknomo, K., **“What is Monte Carlo Simulation?”**. Retirado de  
<<http://people.revoledu.com/kardi/tutorial/Simulation/MonteCarlo.html>> Acessado em  
09/07/2014.

## Anexo I – Arquivos de Importação

Os arquivos de importação seguem um padrão, para que possam ser considerados válidos pelo sistema. Nome;Descrição;Valor da Estimativa de Melhor Caso;Valor da Estimativa do Caso Mais Provável;Valor da Estimativa de Pior Caso;

- Para a importação dos *Epics* do Projeto foram utilizados estes dados:  
Cadastros;Realização dos cadastros necessários para a utilização do sistema;70;83;105;  
Modelos Lógicos;Documentar os modelos lógicos necessários para o sistema.;41;52;66;  
Simulador de Monte Carlo;Criação das funcionalidades necessárias para realizar a Simulação de Monte Carlo.;63;79;126;

- Para a importação das Histórias do Projeto foram utilizados estes dados:

Cadastro de Usuário;Conjunto de componentes para a realização do cadastro de usuários.;10;17;22;

Cadastro de Epics;Conjunto de componentes para a realização do cadastro de Epics.;13;19;24;

Cadastro de Histórias;Conjunto de componentes para a realização do cadastro de Histórias.;13;17;19;

Cadastro de Tarefas;Conjunto de componentes para a realização do cadastro de Tarefas.;11;15;18

Cadastro de Releases;Conjunto de componentes para a realização do cadastro de Releases.;15;25;30;

Cadastro de Sprints;Conjunto de componentes para a realização do cadastro de Sprints.;23;27;36;

- Para a importação das Tarefas do Projeto foram utilizados estes dados:  
Diagrama de Casos de Uso;Representar, em UML, as funcionalidades necessárias para o sistema funcionar;4;7;9;  
Classe User;Implementar a classe de domínio relativo ao Usuario;6;8;10;  
Classe Perfil;Implementar a classe de domínio relativo ao Perfil de um usuário;3;5;8;

Classe UserProject;Implementar a classe de dominio relativo ao relacionamento entre Usuário e Projeto;1;3;7;

Classe Release;Implementar a classe de dominio relativo ao Release;8;13;17;

Classe Sprint;Implementar a classe de dominio relativo ao Sprint;4;6;9;

Classe Project;Implementar a classe de dominio relativo ao Projeto;8;12;17;

Classe Epic;Implementar a classe de dominio relativo a Epic;5;8;10;

Classe History;Implementar a classe de dominio relativo a Historia;4;8;10;

Classe Task;Implementar a classe de dominio relativo a Tarefa;6;8;10;

Classe UserBO;Implementar a classe que contem as regras de negócio do Usuario;6;8;10;

Classe PerfilBO;Implementar a classe que contem as regras de negócio relativo ao Perfil de um usuário;3;5;8;

Classe UserProjectBO;Implementar a classe que contem as regras de negócio relativo ao relacionamento entre Usuário e Projeto;1;3;7;

Classe ReleaseBO;Implementar a classe que contem as regras de negócio relativo ao Release;8;13;17;

Classe SprintBO;Implementar a classe que contem as regras de negócio relativo ao Sprint;4;6;9;

Classe ProjectBO;Implementar a classe que contem as regras de negócio relativo ao Projeto;8;12;17;

Classe EpicBO;Implementar a classe que contem as regras de negócio relativo a Epic;5;8;10;

Classe HistoryBO;Implementar a classe que contem as regras de negócio relativo a Historia;4;8;10;

Classe TaskBO;Implementar a classe que contem as regras de negócio relativo a Tarefa;6;8;10;

Apresentacao de Usuario;Implementar as telas de apresentacao relativo ao Usuario;6;8;10;

Apresentacao de Release;Implementar as telas de apresentacao relativo ao Release;8;13;17;

Apresentacao de Sprint;Implementar as telas de apresentacao relativo ao Sprint;4;6;9;

Apresentacao de Projeto;Implementar as telas de apresentacao relativo ao Projeto;8;12;17;

Apresentacao de Epic;Implementar as telas de apresentacao relativo a Epic;5;8;10;

Apresentacao de Historia;Implementar as telas de apresentacao relativo a Historia;4;8;10;

Apresentacao de Tarefa;Implementar as telas de apresentacao relativo a Tarefa;6;8;10;