UNIVERSIDADE ESTADUAL DA PARAÍBA
CAMPUS VII GOVERNADOR ANTONIO MARIZ – PATOS-PB
CENTRO DE CIÊNCIAS EXATAS E SOCIAIS APLICADAS
CURSO DE LICENCIATURA EM COMPUTAÇÃO

CARLOS ALBERTO FERREIRA DOS SANTOS FILHO

EASY PROCESS E SUA UTILIZAÇÃO NO DESENVOLVIMENTO DO SOFTWARE BANCO DE SIMULAÇÕES

PATOS – PB
2011

CARLOS ALBERTO FERREIRA DOS SANTOS FILHO

EASY PROCESS E SUA UTILIZAÇÃO NO DESENVOLVIMENTO DO SOFTWARE BANCO DE SIMULAÇÕES

Trabalho de Conclusão de Curso apresentado ao Curso de Licenciatura em Computação da Universidade Estadual da Paraíba, em cumprimento à exigência para obtenção do grau de Licenciado em Computação.

Orientador: Esp. Vitor Abílio Sobral D. Afonso

PATOS – PB

2011

S231e SANTOS FILHO, Carlos Alberto Ferreira dos.

Easy Process e sua Utilização no Desenvolvimento do Software Banco de Simulações / Carlos Alberto Ferreira dos Santos Filho.
Patos: UEPB, 2011.
50f

Monografia (trabalho de conclusão de curso -
- (Tcc) - Universidade Estadual da Paraíba.
Orientadora: Prof. Esp. Vitor Abílio Sobral D. Afonso

1. Engenharia de Software 2. Easy Process – YP
I. Titulo II. Afonso, Vitor Abílio Sobral Dias

CDD 005.1

CARLOS ALBERTO FERREIRA DOS SANTOS FILHO

EASY PROCESS E SUA UTILIZAÇÃO NO DESENVOLVIMENTO DO SOFTWARE BANCO DE SIMULAÇÕES

Trabalho de Conclusão de Curso apresentado ao Curso de Licenciatura em Computação da Universidade Estadual da Paraíba, em cumprimento à exigência para obtenção do grau de Licenciado em Computação.

Aprovada em 17 / 11 / 2011.

Vitor Abílio Sobral Dias Afonso
Prof. Esp. Vitor Abílio Sobral Dias Afonso / UEPB
Orientador

Weverton Rubens Souto Pereira
Prof. Weverton Rubens Souto Pereira / UEPB
Examinador

Pedro Ivanio Firmino Pereira de Oliveira
Prof. Pedro Ivanio Firmino Pereira de Oliveira / UEPB
Examinador

AGRADECIMENTOS

Agradeço principalmente a Deus, pela vida a mim concedida e por mi proteger em todos os obstáculos e desafios encontrado ao longo desse percurso.

A minha família: pais, irmãos, esposa e filha, e em especialmente a minha mãe, pois alem de ajudar financeira em todos os aspectos, mi deu a mão quando enxergou possíveis quedas adiante, por ter mi conduzido lado-a-lado rumo a esta conquista, dando sempre exemplo total de vida.

Agradeço em geral a todos os que se fizeram presente nesta conquista.

RESUMO

Esta pesquisa objetiva fornecer uma visão global sobre Easy Process – YP analisando seus fundamentos e práticas. Easy Process é uma metodologia ágil de desenvolvimento de sistemas, aplicável em equipes pequenas e medias. E um processo original, diferentes dos tradicionais contendo algumas praticas controvérsia YP. Para uma aplicação pratica mais fácil, são descritas possíveis formas de se implantar a metodologia. Foi utilizado um processo de desenvolvimento de software que considera os pontos importantes e aplicáveis a projetos no embasamento teórico do Easy Process. Ao término, e aplicado à construção do sistema BANCO DE SIMULAÇÔES o processo YP para analise de seus resultados

Palavras-chaves: Easy Process – YP. Modelagem Ágil. Engenharia de Software

ABSTRACT

This research aims to provide an overview of Easy Process - YP analyzing its fundamentals and practices. Easy Process is an agile systems development, applicable in small and medium-sized teams. And a unique process, containing some non-traditional practices controversy YP. For a practical application easier, we described possible ways to implement the methodology. We used a software development process that considers the important points and applicable to projects in the theoretical foundation of the Easy Process. At the end, and applied to the system construction process SIMULATIONS BANK YP for analysis of results

Key-words: Easy Process – YP. Agile Modeling. Engineering of Software.

SUMÁRIO

1 INTRODUÇÃO

A Engenharia de Software (ES) tem propósito como a aplicação de uma abordagem sistemática, disciplinada e quantificável no desenvolvimento, operação e manutenção de software, possuindo representação ou abstração dos objetos e atividades envolvidas no processo de software. Além disso, oferece uma forma mais abrangente e fácil de representar o gerenciamento de processo de software e conseqüentemente o progresso do projeto.

Os processos na ES se caracterizam em dois tipos: processos pesados, onde se tem maior controle o que o torna burocrático e processo leve, menor controle maior velocidade e simplicidade.

As metodologias de desenvolvimento de software sofrem constantes modificações ao longo da historia da informática.

Com surgimento da orientação a objeto e adoção dos desenvolvedores e analistas fez-se necessário a criação de novas metodologias, que organizassem o desenvolvimento de software. Sabendo-se que a orientação a objeto utiliza-se do reaproveitamento de código. Dentre as novas metodologias que surgiram para atender esta demanda, destacam-se os Modelos Ágeis, entre eles o easYProcess (YP), criado pela Professora DsC. Francilene Procópio Garcia pertencente ao curso de Ciência da Computação da Universidade Federal de Campina Grande (UFCG).

A metodologia criada pela supracitada professora, tem por finalidade o desenvolvimento de processos pequenos, que envolvem uma equipe reduzida e que possa ser concluído num espaço de tempo curto como no caso das universidades que o tempo é menor para desenvolver o projeto, micro empresas de software também se utiliza dessa metodologia por isso que YP é uma alternativa interessante.

A utilização de uma metodologia diminui os riscos, uma vez que toda equipe vai estar de maneira organizada, diminuindo o tempo do processo, aumentando a qualidade e garantindo a funcionalidade do mesmo. Existem vários problemas no desenvolvimento de software que, na maioria das organizações, são entregues fora do prazo planejado e da previsão orçamentária.

Um processo de desenvolvimento define o fluxo de trabalho, as atividades, os artefatos e as regras para os envolvidos no trabalho. O processo de desenvolvimento de software possui os seguintes objetivos (**CONALLEN, 2000):**

- Prover direção sobre a ordem das atividades da equipe;
- Especificar quais artefatos deve ser desenvolvido;
- Direcionar as tarefas de desenvolvedores e da equipe como um todo;
- Oferecer critérios para monitorar e mensurar os produtos do projeto e das atividades.

É imprescindível a utilização de um processo no desenvolvimento de software, mas muitas vezes, aplicar metodologias consagradas, como o RUP (Rational Unified Process), se torna inviável para o desenvolvimento de softwares menores. As metodologias existentes no mercado são muito complexas para pequenos softwares, uma vez que envolvem diversas fases e tipos diferentes de documentos. Para sistemas reduzidos, a utilização de um ambiente extremamente burocrático, repleto de formulário e de definições pode fazer com que os custos não justifiquem os benefícios.

O presente estudo visa apresentar uma metodologia ágil para o desenvolvimento de software, baseada na simplicidade aliada à disciplina.

Contudo, pondo em prática o conceito de processo ágil de desenvolvimento de software foi desenvolvido um sistema alvo de simulação - BANCO DE SIMULAÇÕES, objetivando ajudar professores e alunos no estudo da disciplina de Física, integrante do Ensino Médio. A utilização do sistema se dá através de uma barra contendo menus, com diversas funcionalidades próprias para cada assunto. Para tanto, o software desenvolvido se baseou no processo ágil YP, procurando explorar todos os seus aspectos quanto à aplicação de suas práticas no desenvolvimento do mesmo.

1.1 JUSTIFICATIVA

Devido ao curto espaço de tempo existente nos períodos letivos das universidades, pequenas produções de software necessitam do uso de uma metodologia que acompanhe agilmente o desenvolvimento de softwares, que neste caso, tem duração de 06 (seis) meses.

Portanto, essa pesquisa vem mostrar o quanto é essencial e útil o processo ágil YP, suprindo todas as necessidades.

Enfim, torna-se clara a importância do easYProcess quando se trata de uma equipe pequena e o curto espaço de tempo disponível. Posteriormente, será mostrada a sua utilização.

1.2 OBJETIVOS

1.2.1 OBJETIVO GERAL

Avaliar a aderência da metodologia ágil de desenvolvimento easYProcess, através do qual foi desenvolvido um sistema alvo de simulações - BANCO DE SIMULAÇÕES.

1.2.2 OBJETIVOS ESPECÍFICOS

- Detalhar a metodologia aplicada sob desenvolvimento do sistema de simulações – BANCO DE SIMULAÇÕES.
- Identificar as características de alguns processos ágeis de desenvolvimento de software e, principalmente a metodologia easYProcess.
- Apresentar e analisar a aplicação do processo easYProcess através do sistema – BANCO DE SIMULAÇÕES.
- Identificar as vantagens e desvantagens da utilização do YP no processo.

1.3 ESTRUTURA DO TRABALHO

O segundo capitulo descreve uma visão geral sobre o processo de desenvolvimento de software e sobre metodologias ágeis: contexto das mudanças, manifesto ágil, princípios, ponto-chave e exemplos.

O terceiro capitulo apresenta as tecnologias e ferramentas utilizadas no desenvolvimento do sistema BANCO DE SIMULAÇÕES, enfatizando a metodologia YP.

O quarto capitulo descreve o desenvolvimento do software de simulações, apresentando problema, a arquitetura, a descrição do projeto, os conceitos de YP que foram ou não aplicados, implementação das funcionalidades.

O quinto capitulo descreve as considerações finais e contribuições.

2 PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

O processo de desenvolvimento de software é formado por um conjunto de passos parcialmente ordenados, artefatos, grupo de pessoas, recursos, organizações e restrições, sendo como objetivo produção de software como resultado final requerido, como veremos nas futuras descrições deste trabalho.

Segundo Pressman (2005), a engenharia de software é uma disciplina que estabelece o uso de conceitos da engenharia com o objetivo de desenvolver software de maneira sistemática e com baixo custo, gerando um produto confiável e eficiente. Seus elementos fundamentais são:

- Métodos: Descrevem como fazer. Tarefas com planejamento, estimativa de projeto, análise de requisitos de software e de sistema, projeto da estrutura de dados, arquitetura de programa e algoritmo de processamento, teste e manutenção.
- Ferramentas: As ferramentas de engenharia de software fornecem apoio automatizado ou semi-automatizado aos métodos.
- Procedimentos: Realiza a ligação entre os métodos e as ferramentas.

Este conjunto de elementos e o meio como são agregados define os diversos processos e paradigmas da engenharia de software. De maneira geral, podemos considerar o Processo de Desenvolvimento de Software contento três fases. Na de definição é estabelecido o domínio do problema e realizado o planejamento do projeto. A segunda é a fase de desenvolvimento onde e realizado a construção definitiva do software. A última é a fase de manutenção, onde mudanças corretivas e melhorias são realizadas.

A engenharia de software é um ramo da engenharia que se concentra no desenvolvimento de softwares tendo como base a utilização de modelos de projetos que foram bem sucedidos, o que resulta numa boa relação de custo-benefício. Segundo Sommerville (2003), a Engenharia de Software é uma disciplina que visa todos os aspectos da produção de software, e seu foco está nas teorias, métodos e ferramentas para o desenvolvimento profissional do software.

Existem várias Metodologias de Desenvolvimento utilizadas atualmente no processo de desenvolvimento de software. Um processo tão complexo quanto

cansativo e que exige um planejamento para compor talo, que é a escolha da metodologia mais adequada ao desenvolvimento de um sistema e à equipe integrante do mesmo. A metodologia YP (easYProcess) é o método que foi utilizado com este objetivo. Para apresentar, compreender como é avaliada essa metodologia ágil de desenvolvimento de software, o presente estudo estará analisando seus conceitos e práticas e, ainda, aplicando-os no Sistema BANCO DE SIMULAÇÕES.

2.1 METODOLOGIAS ÁGEIS

Este trabalho relata a metodologia easYProcess e sua aplicação no software BANCO DE SIMULAÇÕES. Como YP é um processo ágil, este capitulo é dedicado à introdução dos conceitos que envolvem as metodologias ágeis, abordando também, o contexto do surgimento deste novo paradigma na Engenharia de Software.

2.2 CONTEXTO DAS MUDANÇAS

De um modo geral, os projetos de desenvolvimento de software tem sido de preocupação constantemente para os clientes dos sistemas, gerentes de projeto e para os próprios desenvolvedores. Os prazos são estão sempre revogados, nunca na data de entrega, longas fazes de análises de requisitos, estouro no orçamento do projeto, fases de teste insuficientes, muita às vezes cancelamento do projeto, produto necessariamente mais caro, com alta taxa de defeito de requisitos que não satisfazem as necessidades reais dos clientes são apenas alguns exemplos que servem pra meramente ilustrar a gravidade dos problemas encontrados durante o desenvolvimento de software.

Para lidar com esses problemas no decorrer do processo de desenvolvimento do software, inúmeros profissionais da área de computação estabeleceram práticas e princípios baseados na Engenharia para a produção de software confiável e funcionalmente eficiente para que possa dar credibilidade e estabilidade ao cliente.

Segundo (PRESSMAN, 2005), paradigmas como ciclo de vida clássico deveriam ter posto fim a vários problemas de desenvolvimento de software, já que proviam fases ordenadas e bem definidas como: Engenharia de Sistemas, Análise de Requisitos, Projeto de Software, Codificação, Testes e Manutenção.

Verifica-se que durante ciclo de vida clássico, 60% dos custos são de desenvolvimento e 40% de teste, sendo que para softwares customizados, os custos de evolução excedem os de desenvolvimento (**SOMERVILLE, 2003**). Por tanto o que foi mencionado reforça a contradição citada por Pressman, já que o longo planejamento inicial era feito para evitar custos futuros.

Contudo, com o surgimento de novas tecnologias de desenvolvimento de novas técnicas de programação e de Engenharia de Software (orientação a aspectos, desenvolvimento orientado a teste, refatoração, padrões de projetos entre outros), podemos verificar que os custos de manutenção podem ser considerados menores.

Por tanto, voltando ao assunto sobre os custos no processo de desenvolvimento de softwares, com a redução dos custos por parte da manutenção reforça a seguinte dúvida: Por que não é possível a entrega de software útil de qualidade e com eficiência?

2.2.1 MANIFESTO ÁGIL

As principais manifestações: o rompimento ás resistência aos processos usuais, tornando o processo de desenvolvimento mais simples e natural para os desenvolvedores, e possibilitando a mudança de requisitos durante o projeto, refletindo as necessidades do cliente e minimizando os riscos de fracasso do projeto.

O manifesto Agile definido pelas simples declarações de valores que definem preferências, não alternativas, encorajando o enfoque de certas áreas, mas sem eliminar outras (AMBLER, 2004) São elas.

- Indivíduos e interações valem mais que processos e ferramentas;
- Um software funcionando vale mais do que documentação abrangente;
- A colaboração do cliente vale mais que a renegociação do contrato;
- Responder a mudanças vale mais que seguir um plano;

A Agile Alliance com base neste manifesto definiu o conjunto de princípios que são seguidos nos processos de desenvolvimento ágil de software.

2.2.2 PRINCÍPIOS

A seguir veremos os princípios que são atribuídos às metodologias ágeis:

- A maior prioridade é a satisfação do cliente através da entrega rápida e contínua de software útil;
- A mudança de requisitos será bem recebida em qualquer momento do desenvolvimento, mesmo em uma fase mais avançada no desenvolvimento
- A entrega de software deverá se frequente, em poucas semanas ou meses, mas sempre preferindo a menor escala;
- Pessoas que entendem o negócio e desenvolvedores devem trabalhar juntos diariamente;
- Construa o projeto com pessoas motivadas, dando todo suporte necessário a elas e confiança;
- O método mais eficiente e eficaz de entrega / disseminar a informação em uma equipe de desenvolvimento é a conversa frente-a-frente;
- Software em funcionamento deve ser a métrica de progresso;
- Processos ágeis promovem substancialmente o desenvolvimento. Clientes, desenvolvedores e usuários deveriam ser capazes de manter a paz indefinidamente;
- Atenção continua a excelência técnica e o bom projeto aumentam a agilidade;

- Simplicidade – a arte de maximizar a a quantidade de trabalho a não ser feito – é essencial;
- As melhores arquiteturas, requisitos, e projetos emergem de equipes organizadas;
- Em intervalos regulares, a equipe deve refletir sobre como podem ser mais efetivos, então devem ajustar seu comportamento de acordo com isso.

As metodologias ágeis se adequar a tais princípios ou correm o risco de não pertencer a tal classificação, já que são metodologias baseadas no comportamento humano natural.

Portanto, as principais metodologias ágeis com easYProcess, estes princípios que norteiam as metodologias ágeis foram extraídos das experiências práticas destes autores com suas metodologias.

2.2.3 PONTO CHAVE

Na verdade os valores e princípios da Agile Alliance não explicitam o que é o ponto-chave das metodologias ágeis: a comunicação. As metodologias ágeis sugerem o jogo cooperativo, comunicativo e inventivo com recursos limitados ou simplesmente jogo cooperativo da comunicação com os seguintes objetivos claros (COCKBURN, 2002):

- Entregar o software com qualidade e valor ao cliente com objetivo principal;
- Preparar-se para a próxima jogada com objetivo secundário.

Desta forma, esta nova definição se aproxima mais do domínio e entendimento humano. Os participantes seriam: programadores, patrocinadores, gerentes, especialista, consultores, designer e testadores. Cada pessoa da equipe deve ser comunicativo, interagindo e desenvolvendo com todos os participantes.

Para as metodologias ágeis, quando se substitui ou adicionam-se novas pessoas a um projeto, desta podem não conseguir descobrir pela documentação o estado em que o desenvolvimento se encontra, porque as pessoas possuem comportamentos diferentes e não reagem com funções lineares (COCKBURN, 2002).

Assim, a comunicação e considerada um fator tão crucial para o projeto que muitas metodologias ágeis delimitam inclusive o ambiente de desenvolvimento.

2.2.4 EXEMPLOS

Em comparação com as metodologias tradicional de software, as metodologias ágeis é menos "orientado-a-documentos" e mais "orientado-a-código". Portanto, sendo isso um reflexo de duas diferencias mais profundas entre os dois estilos: Métodos ágeis são mais adaptativos que preditivos, acolhendo bem as mudanças e mais "orientado-a-pessoas", confiando mais em habilidades, competência e colaboração, que "orientado-a-processos" (PAETSCH, 2003). Abaixo uma descrição de algumas metodologias ágeis.

- **easYProcess:** O EasyProcess é uma metodologia de desenvolvimento de software criada com o intuito de atender o desenvolvimento de projetos acadêmicos. Esta metodologia foi idealizada pela Professora Drª Francilene Procópio Garcia da Universidade Federal de Campina Grande (UFCG) e foi concebida no ambiente do grupo PET Computação desta mesma universidade no ano de 2003.

- **eXtreme Programming** (BECK, 2000): É uma disciplina de desenvolvimento de software baseada nos valores de simplicidade, comunicação, feedback e coragem. Elas funciona reunindo toda equipe sobre a práticas simples, com feedback suficiente para que todos saibam onde estão no desenvolvimento.

- **Modelagem Ágil** (AMBLER, 2004): tem seu foco em um conjunto de básicos e suplementares princípios e práticas de modelagem. A Modelagem Ágil é inspirada em XP, pontuando que as mudanças no desenvolvimento de software são comuns e devem afetar a forma de desenvolvimento, e por extensão, a forma de modelar.

- **Scrum** (PAETSCH, 2003): é um método para gerenciar o processo de desenvolvimento de sistemas, aplicando idéias da teoria de controle de processos industriais como flexibilidade, adaptabilidade e produtividade. Scrum não propõe nenhuma técnica particular de implementação, focando em como uma equipe deve trabalhar junta para produzir trabalho de qualidade em ambientes de mudança.

- **Crystal Methodologies** (COCKBURN, 2002): são uma família de diferentes metodologia. Alistair Cockburn, seu desenvolvedor, acredita que para cada tipo diferente de projeto deva existir uma metodologia que se adéque mais. As metodologia estão indexadas por diferentes cores para indicar sua “dureza” (métrica originada do “tamanho” do projeto e seu “peso”).

- **Feature Driven Development** (PAETSCH, 2003): é um processo de interações curtas (duas semanas) focando no projeto (designer) e na fase de uma construção. Feature Driven Development (FDD) não recomendada um modelo de processo especifico. FDD consiste de cinco processos sequenciais:

  - Desenvolvimento de um modo geral;
  - Construção de uma lista de funcionalidades (Features);
  - Planejamento por funcionalidade;
  - Projeto por funcionalidade.
  - Construção por funcionalidade.

- **Dynamic Systems Development Method** (PAETSCH, 2003): provê um framework para o desenvolvimento rápido de aplicações. Dynamic

Systems Development Method (DSDM) consiste de cinco fases, e suas interações são denominadas time boxes, que duram entre poucos dias ou semanas

- **Adaptive Software Development (ASD)** (PAETSCH, 2003): provê um framework para o desenvolvimento de sistemas grandes e complexos com coordenação suficiente que evite caos no projeto seus princípios vêm de pesquisa em desenvolvimento iterativo, encorajando tal tipo de desenvolvimento como a construção prototipagem constante. O processo ASD contém três fases não-lineares e sobrepostas: especulação, colaboração e aprendizagem (PAETSCH, 2003).

2.3 CONCLUSÃO

Portanto, após ter mostrado todas estas outras metodologias, fez-se uso da easY Process no projeto por se adequar melhor ao desenvolvimento em questão. Outro motivo, está no fato de que trata-se de uma equipe formada unicamente por uma pessoa, tendo que desempenhar todas as funções exigidas na metodologia e, pelo pouco tempo disponível para finalizar o já referido projeto acadêmico.

3 TECNOLOGIAS E FERRAMENTAS

No decorrer do desenvolvimento do Sistema BANCO DE SIMULAÇÃO, foram utilizadas algumas metodologias, disciplina educacional (Física), tecnologias e ferramentas, que ao longo do processo necessitaram de estudos / aprendizado de umas aprimoramentos de outras tais como: JAVA, disciplina educacional (Física) e easY Process – YP, foco principal desta monografia.

3.1 EASY PROCESS (YP)

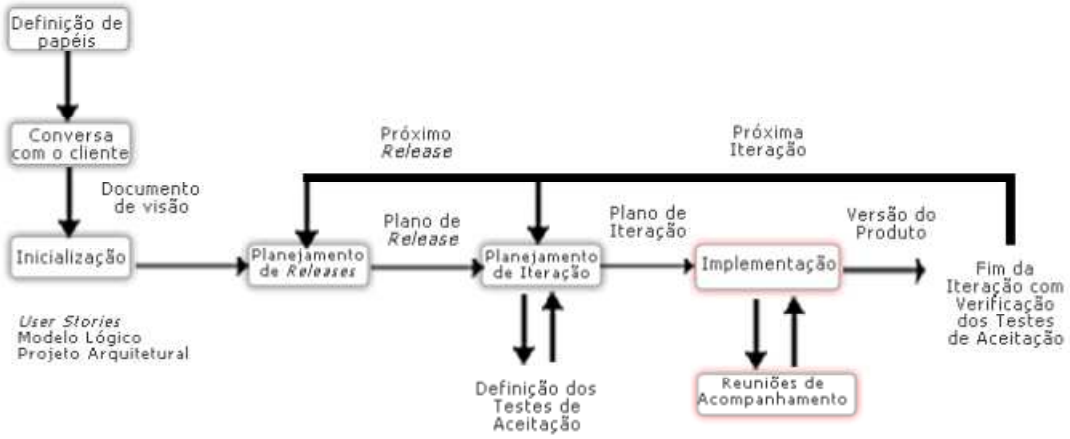
Esta metodologia foi aplicada no desenvolvimento do software BANCO DE SIMULAÇÂO e, será apresentada detalhadamente, a fim de explicar sua escolha, sua adequação ao referido Sistema e, consequentemente, sua utilização.

3.1.1 HISTÓRICO

Easy Process é uma metodologia ágil de desenvolvimento de software. A necessidade de se utilizar melhores práticas para desenvolvimento de software no meio acadêmico, que possibilitem maior sucesso na implementação de projetos oferecidos em algumas disciplinas, foi a principal motivação para a criação do *easYProcess*. Tal metodologia foi idealizada pela Drª Francilene Procópio Garcia, responsável pelo Grupo de Pesquisa e Desenvolvimento em Engenharia de Software do Departamento de Sistemas e Computação (DSC) da Universidade Federal de Campina Grande (UFCG), tendo em vista que está metodologia foi apoiada em outras praticas metodológicas: RUP, XP e Agile Modeling.

3.1.2 FUNDAMENTOS

Figura 1: Modelo do easYProcess



Fonte: <http://www.dsc.ufcg.edu.br/~yp>

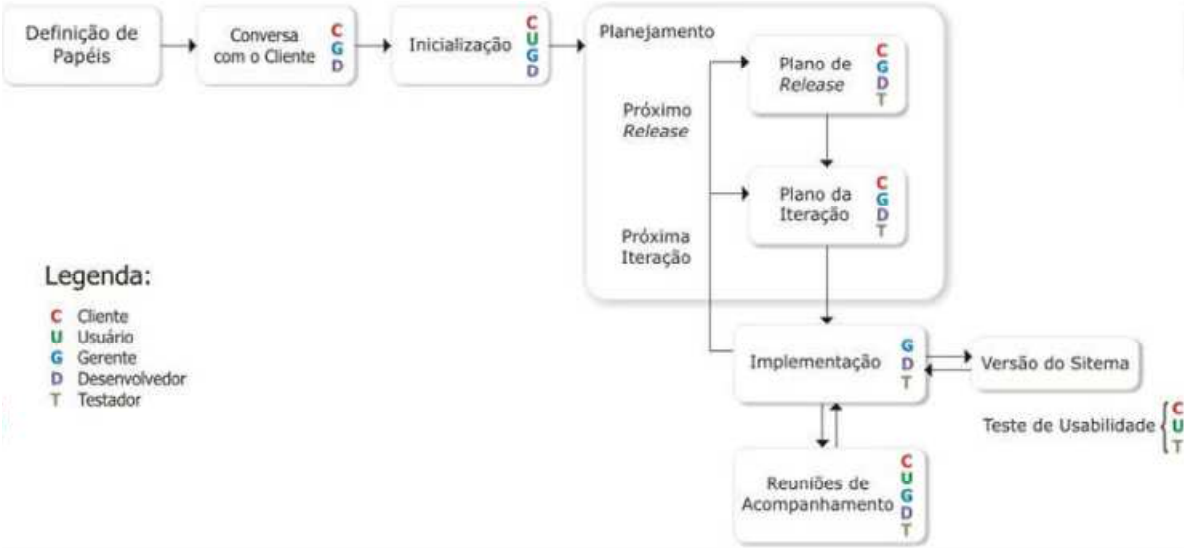
Esta figura representa o fluxograma do easY Process, como são definidos os papeis, conversa com o cliente, implementação, planejamento e reunião de acompanhamento, mostrando como segue e definido as ações.

3.1.3 DEFINIÇÕES DE PAPÉIS

Ao montar uma equipe de desenvolvimento de software sugere-se uma divisão de tarefas entre os membros da mesma, de forma que cada um assuma um determinado papel no desenvolvimento. Um papel constitui um conjunto de responsabilidades que determina qual será o comportamento de uma pessoa durante parte do processo. No **YP** recomenda-se a presença de cinco papéis: **cliente**, **usuário**, **gerente**, **desenvolvedor** e **testador**.

A alocação dos papéis deve ser feita de acordo com as necessidades e escopo do projeto, levando-se em conta as habilidades e características de personalidade das pessoas envolvidas no processo. A equipe de desenvolvimento deve ser estruturada de forma a se obter a maior produtividade possível.

Figura 2: Mostra como é aplicações dos Papéis em YP



Fonte: <http://www.dsc.ufcg.edu.br/~yp>

A imagem mostra como os papéis são aplicados na metodologia YP. Todos os papéis são definidos segundo as aptidões de cada membro da equipe, seguindo um padrão e desenvolvendo.

3.1.4 CONVERSA COM O CLIENTE

E a primeira conversa com o cliente e os desenvolvedores e resto da equipe. Tem por objetivo fazer com que o cliente o os desenvolvedores tenham uma idéia comum a respeito do sistema que será desenvolvido. Este processo acontece a todo tempo, antes de se dar inicio ao processo de desenvolvimento, durante o mesmo e depois do processo para ver se realmente foram desenvolvidos todos os artefatos exigidos.

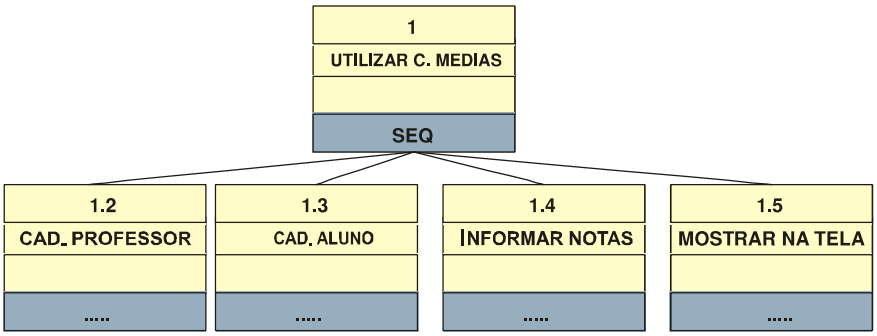
3.1.5 INICIALIZAÇÃO

É um conjunto de 5 atividades: (1) **Modelagem da Tarefa**; (2) definição das ***User Stories*** do sistema, (3) geração do **Protótipo da Interface**, (4) elaboração do **Projeto Arquitetural**; e por fim, (5) construção de um **Modelo Lógico de Dados**, caso exista um banco de dados envolvido.

Para que os desenvolvedores possam ver o futuro sistema de acordo com aspectos de diversas naturezas. Ou seja, passam e entender: como a tarefa funciona, quais são as funções do sistema (*User Stories*), como se dá a interação do usuário com o sistema, como os módulos do sistema se relacionam entre si e com módulo ou sistemas externos, como o sistema deve ser testado, e como o sistema deve ser estruturado logicamente.

3.1.5.1 MODELO DE TAREFAS

Figura 3: Exemplo de um Modelo de tarefas



Fonte: LIMA; SILVA; 2010.

O modelo de tarefa é uma representação dos artefatos do sistema, ou seja, o desenvolvimento do modelo da tarefa é precedido pela análise da tarefa, que consiste no estudo da tarefa a fim de melhorar a compreensão dos desenvolvedores com relação ao uso do sistema. Para a criação desse modelo de tarefas são utilizadas algumas ferramentas tais como: o ITAOS que é utilizado para a criação das tabelas. Veja o exemplo da criação de um modelo de tarefas.

3.1.5.2 USER STORIES

Tabela 1: Exemplo de um Modelo de User Stories

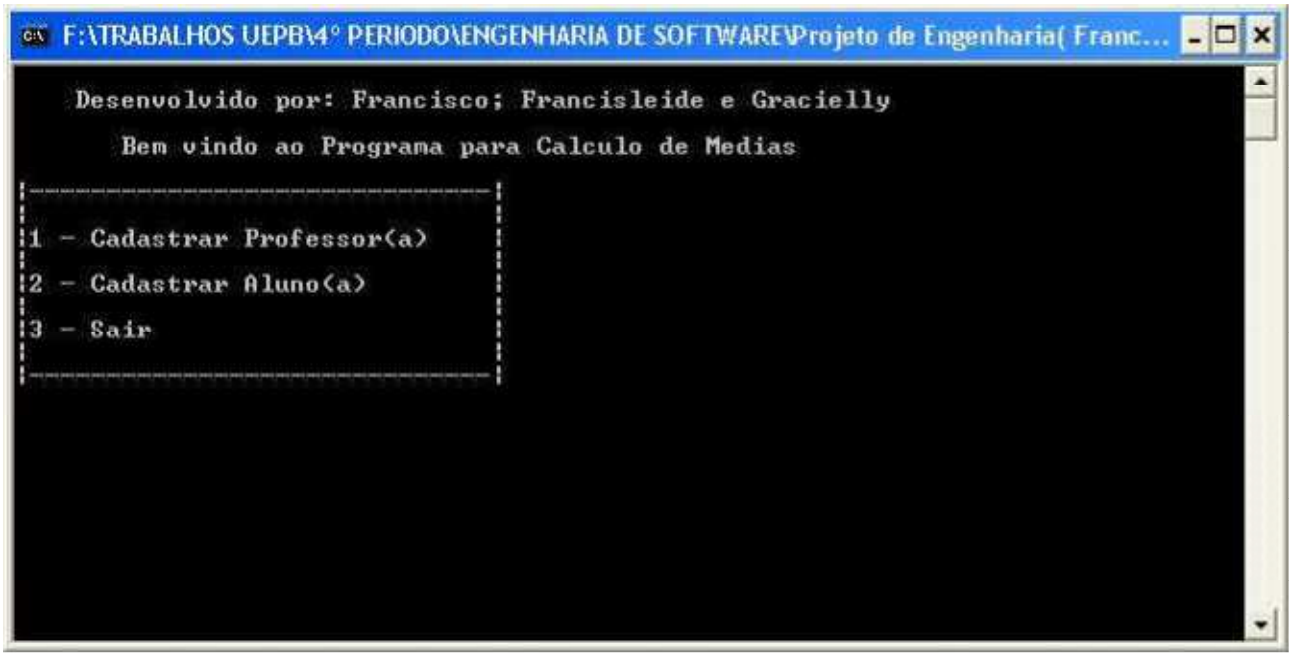
US 01	Estudar linguagem de programação (C++) e elaborar mecanismos de testes a serem utilizados. Gerar exemplos como resultados. **Estimativa inicial: 13h**
TA 1.1	Analisar se os exemplos gerados satisfazem os clientes de modo a confirmar o uso das tecnologias citadas.
US 02	Implementar funcionalidade de cadastro de professor **Estimativa inicial 12h**
TA 2.1	Cadastrar um professor com todos os seus dados corretos (Cadastro Efetuado com Sucesso) e mostrar na tela os dados.
TA 2.2	Cadastrar um professor sem informar todos os campos obrigatórios (Cadastro não deve ser efetuado)
US 03	Implementar funcionalidade de cadastro de aluno **Estimativa inicial 12h**
TA 3.1	Cadastrar um Aluno com todos os seus dados corretos (cadastrado efetuado com sucesso) e mostrar todos os dados na tela e sua situação acadêmica,
TA 3.2	Cadastrar um Aluno sem informar todos os campos obrigatórios

Fonte: LIMA; SILVA; 2010.

As User Stories são as especificações feitas pelo cliente a fim de definir o que o sistema deve ter/fazer. Isso que dizer que são todas as funcionalidades requeridas pelo cliente, na sua primeira conversa, e implementadas pelo seus desenvolvedores. Em seguida exemplo de um modelo de User Stories:

3.1.5.3 PROTÓTIPO DA INTERFACE

Figura 4: Exemplo de um Modelo de Protótipo de Interface

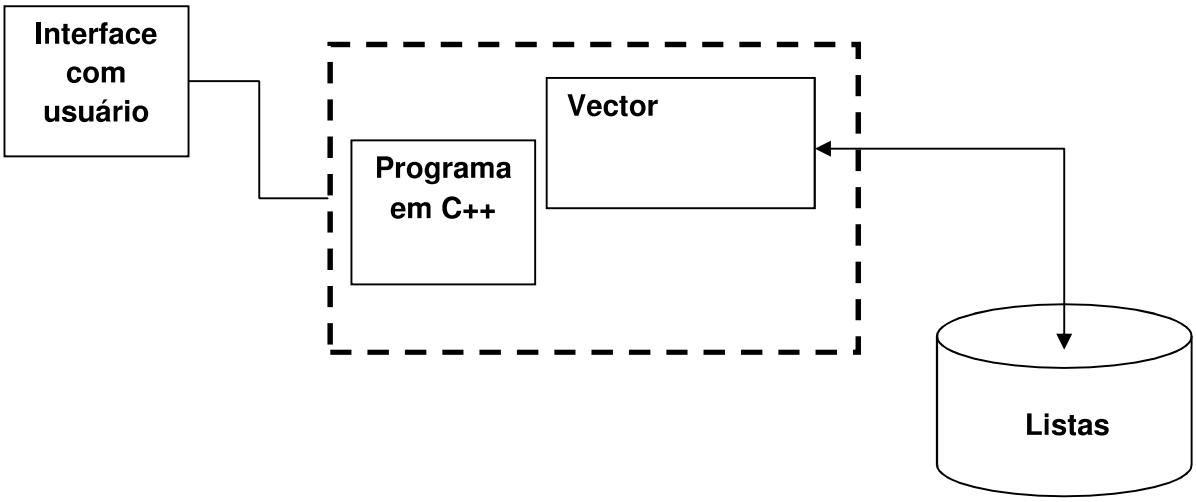


Fonte: LIMA; SILVA; 2010.

Tal processo é apenas um esboço que possam ser construído com um baixo custo de tempo os protótipos são bastante eficientes na hora da construção do sistema. Veja um exemplo de um protótipo de interface

3.1.5.4 PROJETO ARQUITETURAL

Figura 5: Exemplo de um Modelo de Projeto Arquitetural



Fonte: LIMA; SILVA; 2010.

Este processo tem como propósito geral descrever o funcionamento do sistema num alto nível de abstração. Ele é utilizado quando se deseja explicitar como as partes do sistema a serem desenvolvidas interagem entre si ou com outros sistemas, ou seja, nessa etapa é onde descrevemos também o tipo de linguagem.

3.1.5.5 CONSTRUÇÃO DE UM MODELO LÓGICO DE DADOS

Tal modelo mostra como será distribuído os dados e como será o fluxo sobre os mesmos.

3.1.6 PLANEJAMENTO

O planejamento que descreve de forma clara e objetiva um programa das atividades que serão realizadas ao longo do desenvolvimento do sistema. No YP a fase de planejamento é composta por dois planejamentos, o de *release* e o da iteração, no qual existem 3 *releases,* cada um contendo 2 iterações de 2 semanas.

É útil na organização, no planejamento a curto e em longo prazo e no cumprimento das atividades do projeto. Auxilia o gerente a avaliar o desempenho de toda a equipe de desenvolvimento quanto ao andamento do projeto.

O cliente e toda a equipe de desenvolvimento participam do planejamento do *release* e da iteração. Mas é responsabilidade do gerente documentar e disponibilizar de forma clara estes planos.

3.1.6.1 MATRIZ DE COMPETÊNCIA

Tal processo visa atribuir as competências de integrante da equipe, tendo em ainda que essas competências são distribuídas de acordo com os potenciais.

3.1.7 IMPLEMENTAÇÃO

É a realização das atividades estabelecidas no plano da iteração. Tem como principal artefato o código do sistema. Para que a implementação do sistema resulte em uma boa codificação é necessário que a equipe de desenvolvimento considere algumas práticas: Integração Contínua, Boas Práticas de Codificação, Propriedade Coletiva de Código e Testes.

Para que, a partir das atividades estabelecidas no plano da iteração, se obtenha uma versão codificada do sistema. Desenvolvimento e Testadores. Durante todo o período de codificação, implementação.

O tempo de implementação está associado à duração da Iteração. Considerando que, o **YP** sugere 3 releases, cada um com duas iterações de 2 semanas cada, o tempo total necessário para a implementação são de, em média, 3 meses.

3.1.7.1 INTEGRAÇÃO CONTÍNUA

O YP propõe a integração contínua com o objetivo de melhorar o gerenciamento do projeto e o trabalho dos desenvolvedores, caso estes não possuam horários em comum. Para tanto é necessário que os módulos gerados do sistema sejam integrados continuamente, com o apoio de alguma ferramenta de

controle de versão. Na verdade, é interessante que a integração seja feita cada vez que algum código novo seja implementado e esteja pelo menos sem erro de compilação para não comprometer o andamento do projeto.

Com a integração contínua a equipe não necessitará de um novo encontro para integrar tudo o que foi implementado separadamente, além de diminuir a quantidades de erros na implementação devido à integração de código.

3.1.7.2 BOAS PRÁTICAS DE CODIFICAÇÃO

Uma vez que YP sugere a propriedade coletiva de código, deve-se ter em mente a necessidade da geração de um código limpo (evitar: linhas de código desnecessárias, repetição de trechos de código, implementação de funcionalidades não condizentes com as *User Stories*, tarefas não condizentes com o Modelo da Tarefa) e de fácil entendimento. Com isso, um membro da equipe de desenvolvimento que não tenha codificado aquela parte do código será capaz de entendê-la e modificá-la, sem que seja necessário muito tempo ou esforço. Sugere-se aplicar ao projeto as práticas de *Design* Simples, Padrões de Codificação, Padrões de Projeto e Refatoramento, de forma que o código gerado possa ser considerado limpo.

3.1.7.2.1 DESIGN SIMPLES

Consiste em perseguir a geração do melhor código possível, ou seja, de fácil entendimento, sendo praticamente auto-explicativo, exigindo apenas comentários essenciais.

3.1.7.2.2 PADRÕES DE CODIFICAÇÕES

É bastante positivo que, antes de começar a codificar, a equipe de desenvolvimento defina um padrão de codificação. Deve-se determinar, por exemplo, o tamanho da endentação utilizada, a forma como devem ser posicionados os parênteses e chaves, a maneira como devem ser nomeados os métodos e variáveis.

3.1.7.2.3 PADRÕES DE PROJETO

O uso de soluções previamente pensadas por grandes projetistas da Orientação a Objetos fornece a possibilidade de reutilizar micro-arquiteturas de classes, objetos, suas funcionalidades e suas colaborações.

3.1.7.2.4 REFATORAMENTO

Pode ser visto como uma pequena modificação no código do sistema que não altera o seu comportamento funcional, mas que melhora algumas qualidades não-funcionais, tais como: simplicidade, flexibilidade, desempenho e clareza do código. Essa é uma prática bastante eficiente quando o objetivo é a limpeza de código.

3.1.7.3 PROPRIEDADE COLETIVA DE CÓDIGO

Propriedade coletiva é a habilidade de qualquer membro da equipe poder alterar o código elaborado por outros membros durante a implementação. Isto implica dizer que o código é de posse coletiva, ou seja, toda a equipe é responsável por ele.

Esta prática é uma excelente ferramenta para a melhoria contínua de código, além de facilitar a recuperação de falhas, pois possíveis trechos de código problemáticos podem não ser claros para um dos desenvolvedores, mas podem ser facilmente percebidos pelos outros membros da equipe.

3.1.8 TESTE

Testes têm por finalidade verificar se todos os requisitos do sistema forma implementados corretamente, ou seja, trata-se de uma análise das pré e pós-condições diante do contexto no qual está inserido o módulo a ser testado. A prática de efetuar bons testes assegura, na medida do possível, a qualidade e a corretude do sistema, além da satisfação do cliente e do usuário. Para tanto, é necessário testar o sistema sob diferentes aspectos (funcionalidade, usabilidade, carga, controle de acesso, desempenho, entre outros), o que classifica os teste em diferentes

categorias, cada um com um objeto especifico de verificação. O YP recomenda que os **testes de unidade**, **de aceitação** e **de usabilidade** sejam feitos, não impedindo a realização dos demais, caso necessário.

3.1.8.1 TESTE DE UNIDADE

Testam a estrutura interna do código, ou seja, a lógica e o fluxo de dados. Sendo assim, tais testes validam as menores partes do sistema (os métodos, classes ou trechos de códigos).

3.1.8.2 TESTE DE ACEITAÇÃO

Contemplam um conjunto de situações definidas pelo cliente sobre como medir o sucesso do projeto. Descrevem cenários que devem ser suportados pelo sistema, e estão sempre associados às *User Stories*. As características destes testes devem ser extraídas do cliente da maneira mais transparente possível.

3.1.8.3 TESTE DE USABILIDADE

Baseia-se em uma combinação de um conjunto de técnicas que incluem observação, questionários, entrevistas e testes com o usuário. Os testes de usabilidade verificam se os objetivos de usabilidade especificados pelo cliente e pelo usuário foram satisfeitos, além de averiguar como está sendo o processo de interação entre o usuário e o sistema.

3.1.9 REUNIÕES DE ACOMPANHAMENTO

Reunião semanal que visa recolher e analisar métricas. É aqui que se faz uso do *Big Chart* e da Tabela de Alocação de Atividades (TAA) para examinar o andamento do projeto. A preocupação com os riscos e possíveis mudanças deve ser pauta da reunião, assim como a necessidade explícita de refatoramento.

3.2 CONCLUSÃO

A adoção do YP como metodologia no desenvolvimento do sistema BANCO DE SIMULAÇÔES foi a mais acertada possível, visto que as características do projeto adequavam-se diretamente com o que é proposto pelo processo ágil.

O sistema era composto por uma equipe pequena e totalmente interada com o cliente sempre presente e cujo trabalho priorizou medidas de prevenção e controle através de planejamento, simplicidade e teste.

Outros fatores que contribuíram para a escolha de YP no BANCO DE SIMULAÇÔES foram o prévio conhecimento e fácil acesso à metodologia.

4 O SISTEMA BANCO DE SIMULAÇÕES

4.1 INTRODUÇÃO

Neste capitulo são apresentados todos os procedimentos, as técnicas de pesquisa, os métodos de desenvolvimento e as tecnologias, expressadas nos capítulos anteriores. Há também, o processo de coleta de dados com destaque do cliente, a documentação, os artefatos gerados, analisando todos os requisitos necessários para o desenvolvimento desse Sistema.

Easy Process foi adotada como processo por vários indicadores de compatibilidade com a estrutura existente no software. Dentre eles, a presença do cliente, por se tratar de projeto acadêmico que conta com apenas um membro na equipe, a exigência de menos documentação e o ambiente ser dinâmico, colaborativo e aprazado.

O desenvolvimento do presente software se deu pelo aluno de graduação Carlos Alberto Ferreira dos Santos Filho da Universidade Estadual da Paraíba (UEPB). A mesma situa-se na cidade de Patos-PB, que conta com diversos cursos dentre eles Licenciatura Plena em Computação, curso ao qual o desenvolvedor do projeto está vinculado.

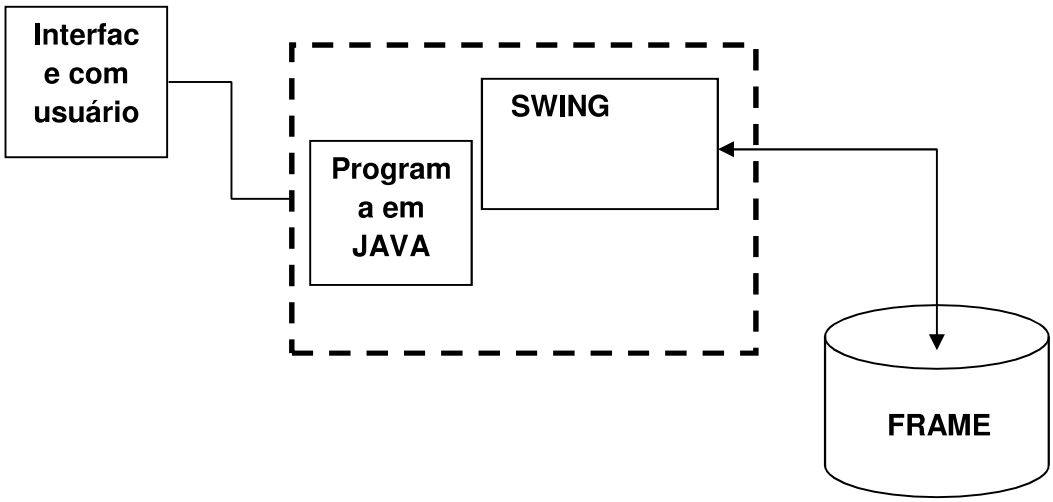
Para o desenvolvimento do software seja factível, busca-se atingir todos os itens relacionados a qualidade de software: funcionalidade, manutenibilidade, usabilidade, eficiência, entre outros.

4.2 PROBLEMA

O sistema visa, facilitar a compreensão dos alunos de física, separando-o pelos respectivos papel e organizando suas tarefas. Contendo o menu próprio (personalizado).

4.3 ARQUITETURA DO PROJETO

Figura 6: Projeto Arquitetural do Sistema BANCO DE SIMULAÇÔES



Fonte: Autor da Pesquisa, 2011.

A arquitetura JAVA foi empregada no projeto para suprir as necessidades de distribuir e configurar a execução da aplicação para satisfazer vários volumes de transação. Por se tratar de uma linguagem Orientada a Objeto e permitir com que as aplicações se desenvolvam.

4.4 LEVANTAMENTOS DE REQUISITOS

Os requisitos apresentam as funcionalidades e restrições que buscam atingir as necessidades do cliente. As restrições são descritas pelo cliente que estabelece os objetivos a serem alcançados no desenvolvimento do software.

Esses requisitos são subdivididos em funcionais, quando identificam as ações que o sistema deve assumir e não-funcionais, quando visam às propriedades e qualidade de software.

A seguir, serão apontados os requisitos do sistema BANCO DE SIMULAÇÕES.

4.4.1 REQUISITOS FUNCIONAIS

Os requisitos funcionais foram levantados em reunião com o cliente onde foi conciliado o que deveria ser desenvolvido com a forma como seria realizado o trabalho, estabelecendo metas, restrições e prazos.

As ferramentas, tecnologias e metodologia empregadas no sistema. A aplicação das mesmas colaborou na elaboração de artefatos necessários ao desenvolvimento do sistema BANCO DE SIMUÇÃO.

- Simulações com interações direta, requeridas pelo cliente.
- Um exercício proposto sobre os assuntos e simulações.

4.4.2 REQUISITOS NÃO FUNCIONAIS

A complexidade de um software é determinada em parte por sua funcionalidade (requisitos funcionais), ou seja, o que o sistema faz, e em parte por requisitos gerais (requisitos não funcionais) que fazem parte do desenvolvimento do software com o custo, performance, confiabilidade, manutenibilidade, portabilidade, custos operacionais entre outros

- Não possuem mapeamento direto nas funcionalidades;
- Não são fáceis de detectar;
- Devem ser observados ao longo do seu desenvolvimento;
- Hardware – Computadores, servidores;
- Usabilidade – As interfaces devem ser amigáveis ao cliente. Deve provê facilidade de uso;
- Confiabilidade – Os dados de todo o sistema devem ser persistentes e estar bem protegidos e seguros contra invasões e acessos não-autorizados.

4.5 DESCRIÇÃO DO PROJETO

Um processo de desenvolvimento de software é um método para organizar as atividades relacionadas com a criação, entrega e manutenção de sistemas de software. O processo de software é um conjunto de atividades e resultados

associados que auxiliam na sua produção. O resultado do processo é um produto que reflete a forma como o processo foi conduzido. Para Sommerville (2003), se um processo de desenvolvimento de software for escolhido impropriamente, certamente os resultados obtidos serão insatisfatórios.

4.5.1 DESENVOLVIMENTO DO SISTEMA

A metodologia de desenvolvimento adotada foi a Easy Process (YP), que enfatiza o desenvolvimento ágil e flexível, com feedback bastante rápido entre a equipe de desenvolvimento e cliente. Essa metodologia foi escolhida, entre outros motivos pelo fato de se tratar de um projeto acadêmico com um curto espaço de tempo 06 (seis) meses e da equipe de desenvolvimento ser de pequeno porte e pela proximidade com o cliente, que tem a disponibilidade de acompanhar de perto o desenvolvimento e decidir quais são suas necessidades mais urgentes.

O processo fácil foi implementado quase no seu todo no desenvolvimento do sistema. A aplicabilidade das práticas de Easy Process será descrita a seguir:

- **Definições de Papéis –** Foram definidos os papéis, seguindo a qualidade e a importância de cada um deles. Os papéis de cliente, usuário, desenvolvedor, gerente e testador;
- **Conversa com o Cliente** - Tem por objetivo fazer com que o cliente o os desenvolvedores tenham uma idéia comum a respeito do sistema que será desenvolvido. Assim podendo torna o trabalho mais rápido e viável.
- **Inicialização –** Para que os desenvolvedores possam ver o futuro sistema de acordo com aspectos de diversas naturezas. Ou seja, passam e entender: como a tarefa funciona, quais são as funções do sistema (*User Stories*).
- **Planejamento –** O planejamento que descreve de forma clara e objetiva um programa das atividades que serão realizadas ao longo do desenvolvimento do sistema. Foram planejadas todas as releases do sistema.
- **Implementação** - É a realização das atividades estabelecidas no plano da iteração. Tem como principal artefato o código do sistema. Foram desenvolvida pelo programador sempre em contato com o cliente.

- **Reuniões de Acompanhamento** - Reunião semanal que visa recolher e analisa. Foram realizadas diversas reuniões visando a melhora do sistema, tendo em vista que são de suma importância para software.
- **Teste** - Testes têm por finalidade verificar se todos os requisitos do sistema foram implementados corretamente, ou seja, trata-se de uma análise das pré e pós-condições diante do contexto no qual está inserido o módulo a ser testado. O YP recomenda que os **testes de unidade**, **de aceitação** e **de usabilidade** sejam feitos, não impedindo a realização dos demais, caso necessário**.**

O primeiro passo constituiu-se da análise e implementação, onde foram criadas todas as classes métodos contendo suas respectivas funcionalidades. A equipe direcionou-se para o desenvolvimento do sistema onde foi trabalhada a parte de programação do projeto, utilizando Orientação a Objetos.

4.5.1.1 PÁGINA INICIAL

Quando o sistema é iniciado, a tela que constitui o acesso a todas as funcionalidades existentes no sistema (software).

Assim quando qualquer o usuário abrir o sistema (software) terá acesso a todo em si vendo que o software não disponibiliza de senha para logar no sistema (Software).

Portanto, o pedido do cliente será atendido, onde o sistema proposto pode ser acessado por qualquer usuário.

4.5.1.2 IMPLEMENTAÇÃO DAS DEMAIS FUNCIONALIDADES

Tendo em vista que se tratando de software educacional as suas respectivas funcionalidades: assuntos, simulação sobre os diversos assuntos e exercícios os respectivos foram aplicadas as seguintes entidades:

- Cliente: o cliente adquire o sistema (software);
- Usuário: quem irá usar o sistema;
- Operação: atividade que será atribuída a um usuário;

- Programa: sistema (software) adquirido pelo cliente;
- Programador: membro da equipe de desenvolvimento;
- Funcionalidade: requisito que será aplicado ao sistema, a pedido do cliente.

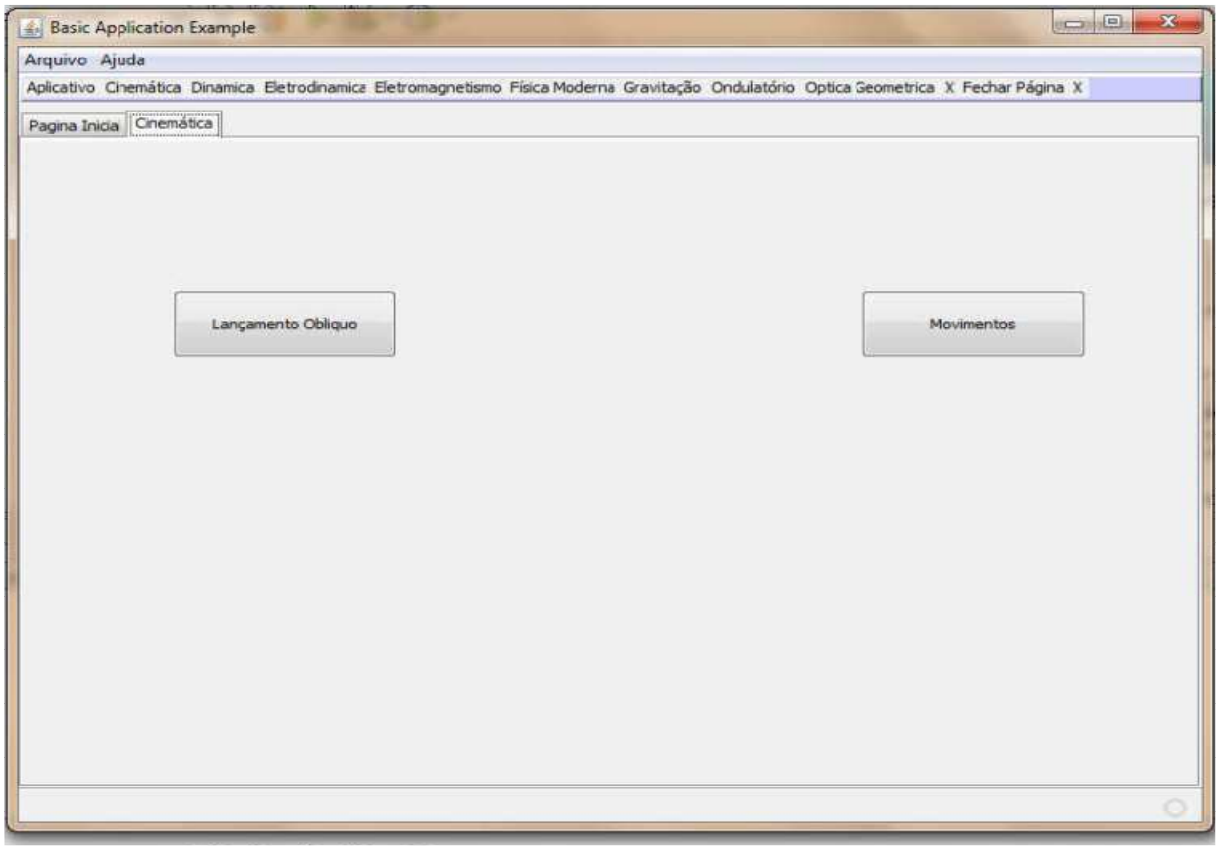
A seguir, são apresentadas as telas do produto / sistema BANCO DE SIMULAÇÔES:

Figura 7: Tela inicial do Sistema

Fonte: Autor da Pesquisa, 2011.

A figura acima representa a tela inicial do sistema BANCO DE SIMULAÇÔES, tendo em vista todas as suas funcionalidades, botões com acessos as funcionalidades e a todo o sistema.

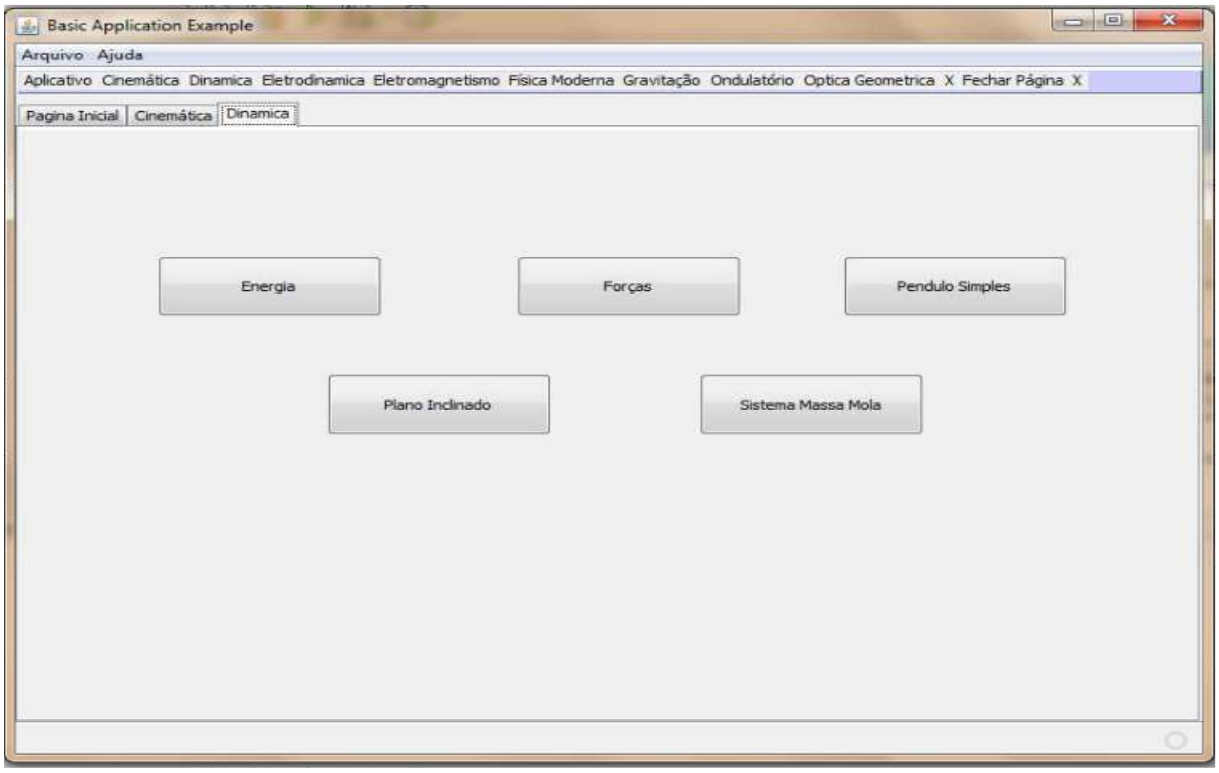
Figura 8: Tela Projeção de Cinemática



Fonte: Autor da Pesquisa, 2011.

A figura acima representa a tela projeção de cinemática do sistema BANCO DE SIMULAÇÔES, tendo em vista todas as suas funcionalidades, botões com acessos as funcionalidades e as simulações referentes à cinemática.

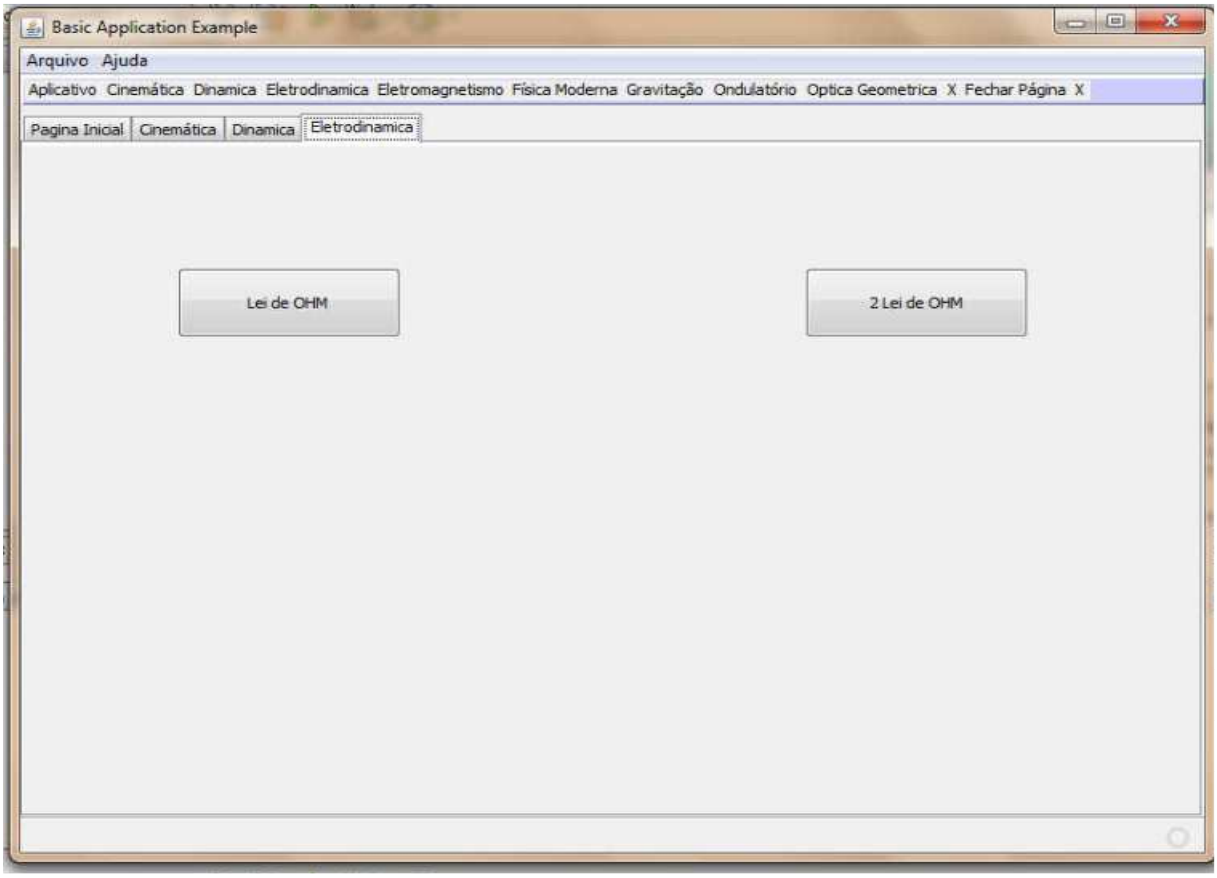
Figura 9: Tela Projeção de Dinâmica



Fonte: Autor da Pesquisa, 2011.

A figura acima representa a tela projeção de dinâmica do sistema BANCO DE SIMULAÇÔES, tendo em vista todas as suas funcionalidades, botões com acessos as funcionalidades e as simulações referentes à dinâmica.

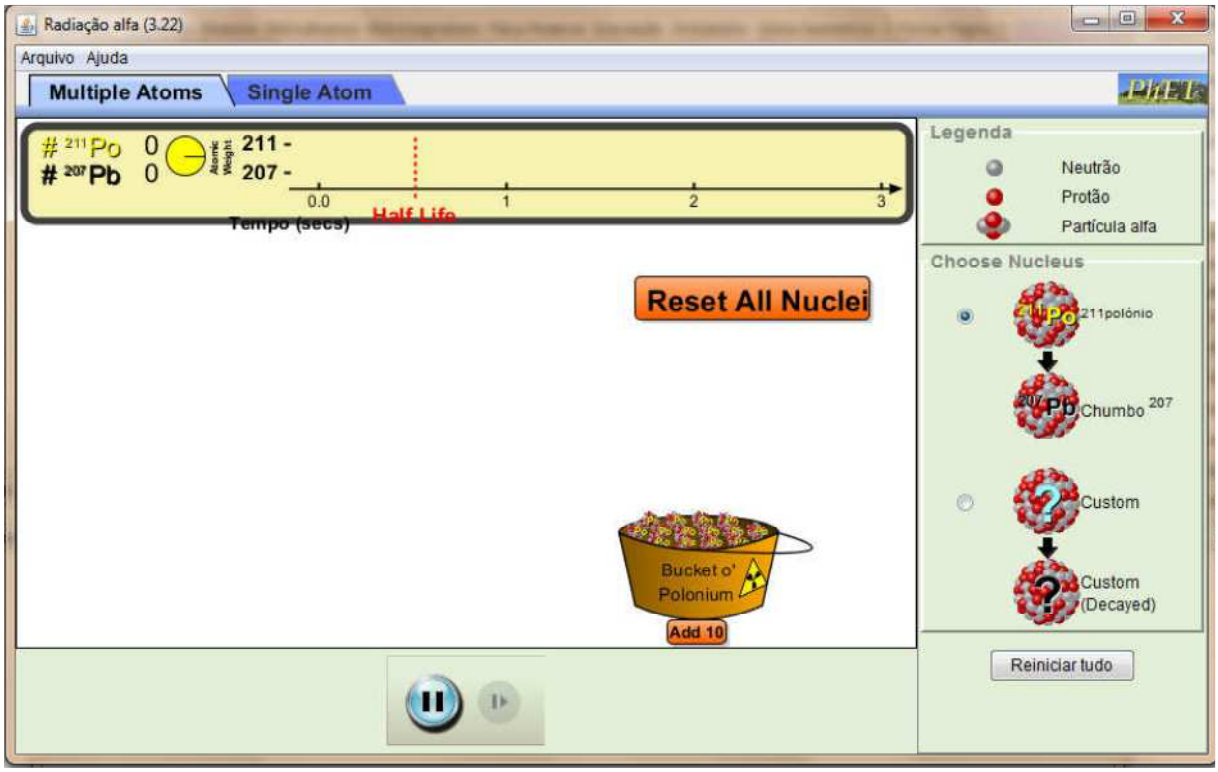
Figura 10: Tela de Projeção de Eletrodinâmica



Fonte: Autor da Pesquisa, 2011.

A figura acima representa a tela projeção de eletrodinâmica do sistema BANCO DE SIMULAÇÔES, tendo em vista todas as suas funcionalidades, botões com acessos as funcionalidades e as simulações referentes à eletrodinâmica.

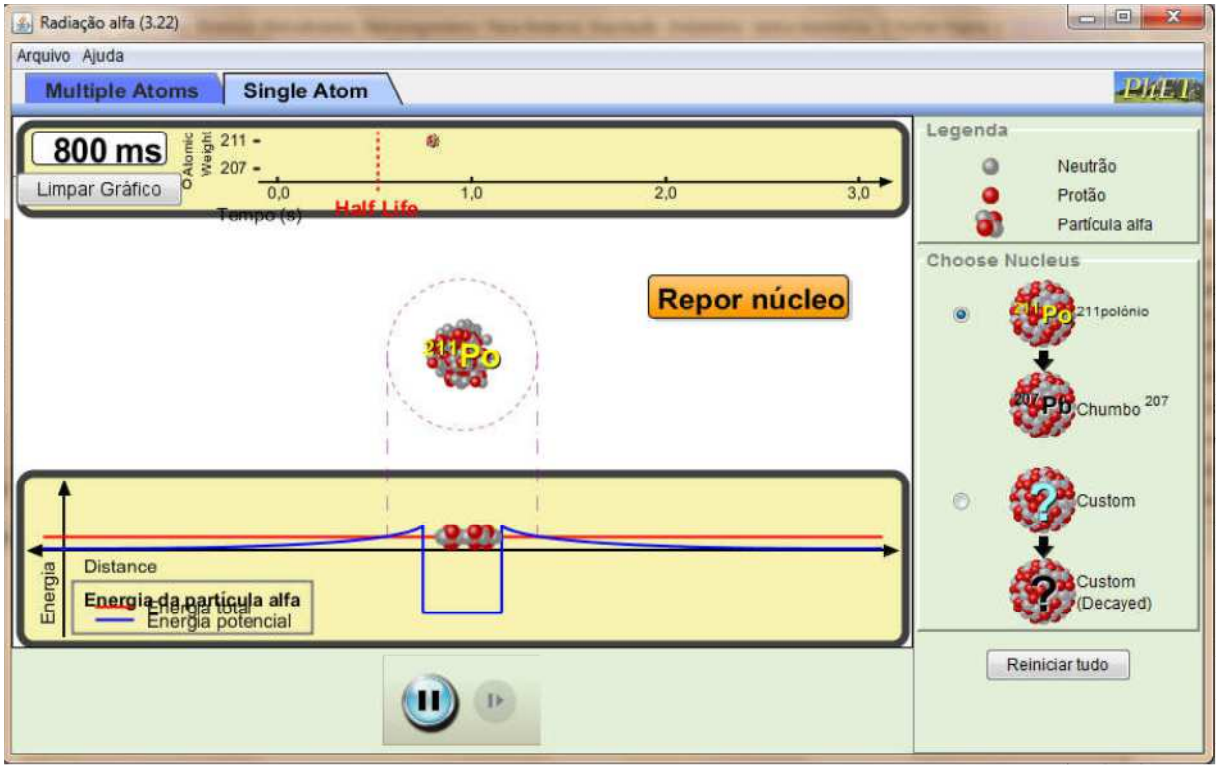
Figura 11: Tela de Simulação de Radiação Alfa



Fonte: Autor da Pesquisa, 2011.

A figura acima representa a tela simulação de radiação alfa do sistema BANCO DE SIMULAÇÔES, tendo em vista todas as suas funcionalidades, botões com acessos as funcionalidades e as simulações referentes à simulação de radiação alfa.

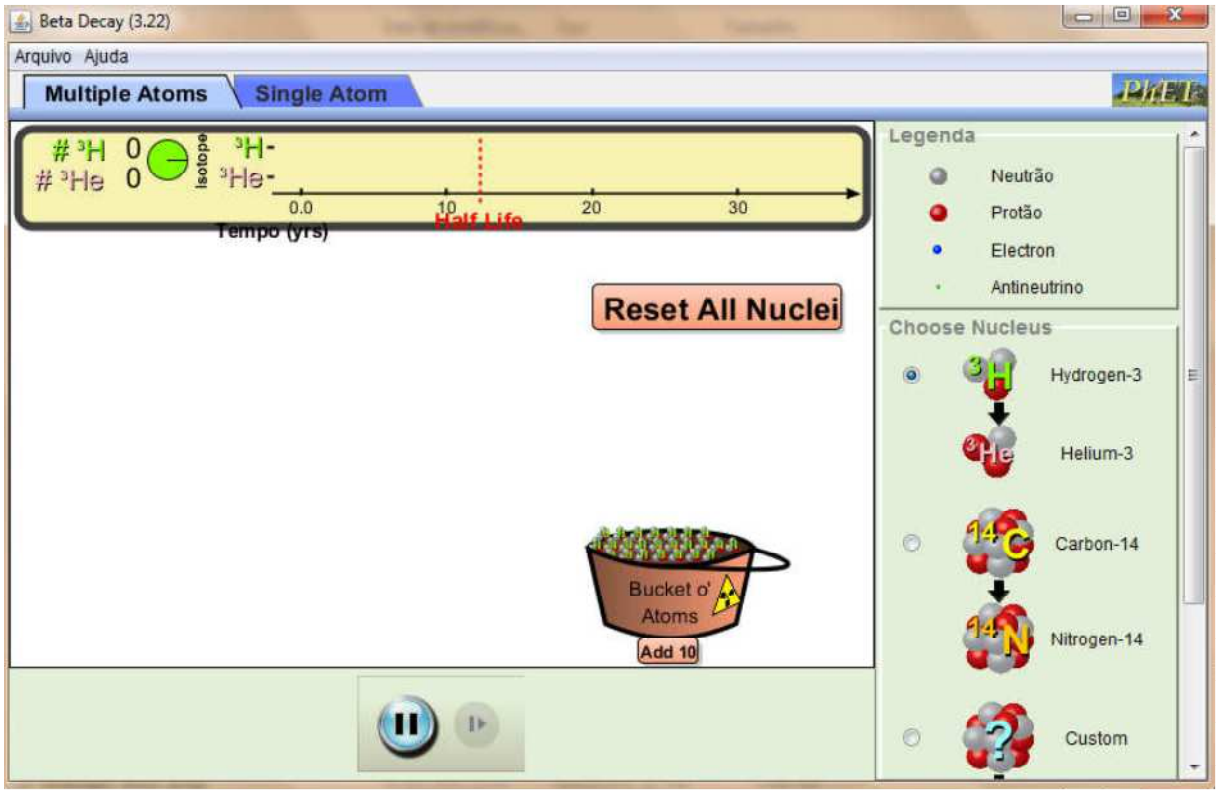
Figura 12: Tela de simulação de Radiação Alfa 2



Fonte: Autor da Pesquisa, 2011

A figura acima representa a tela simulação de radiação alfa 2 do sistema BANCO DE SIMULAÇÔES, tendo em vista todas as suas funcionalidades, botões com acessos as funcionalidades e as simulações referentes à simulação de radiação alfa 2.

Figura 13: Tela de simulação Beta



Fonte: Autor da Pesquisa, 2011.

A figura acima representa a tela simulação de radiação beta do sistema BANCO DE SIMULAÇÔES, tendo em vista todas as suas funcionalidades, botões com acessos as funcionalidades e as simulações referentes à simulação de radiação beta.

4.5.1.3 ARTEFATOS DO SISTEMA

Tabela 2: Modelo de User Stories

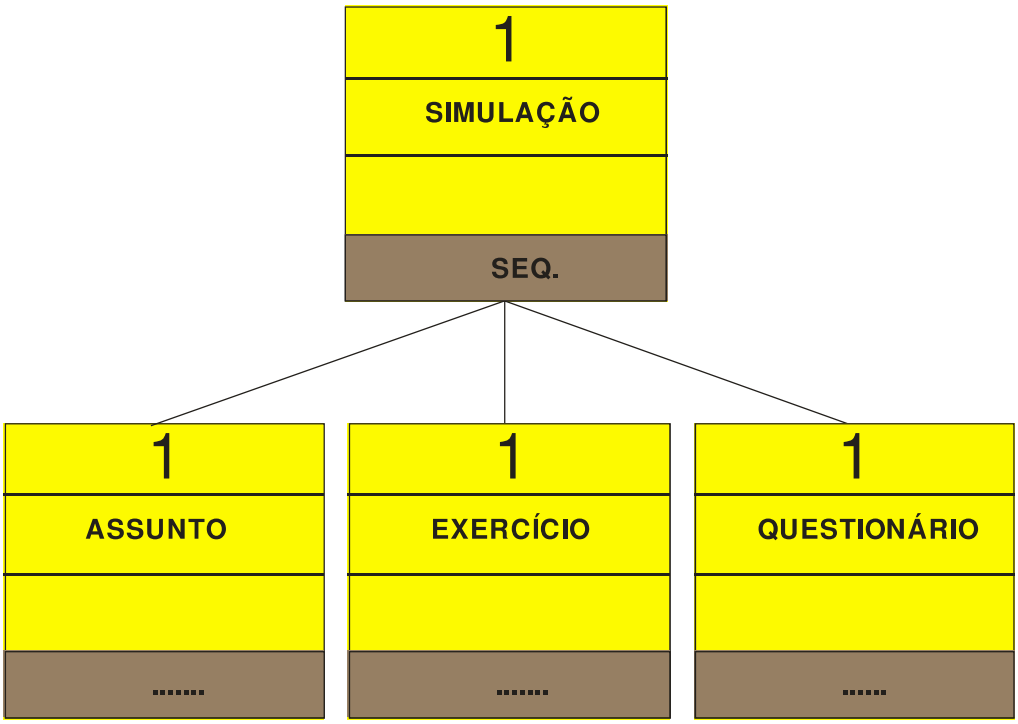
US 01	Estudar linguagem de programação (JAVA) e elaborar mecanismos de testes a serem utilizados. Gerar exemplos como resultados. **Estimativa inicial: 15h**
TA 1.1	Analisar se os exemplos gerados satisfazem os clientes de modo a confirmar o uso das tecnologias citadas.
US 02	Implementar funcionalidade de exercícios dos assuntos referidos **Estimativa inicial 12h**
TA 2.1	Cadastrar perguntas com todos os seus dados corretos (Efetuado com Sucesso) e mostrar na tela os dados.
TA 2.2	Resultado obtido pelo usuário mediante total de acertos (Efetuado com Sucesso) e mostrar resultado na tela
US 03	Implementar funcionalidade de Simulação

	Estimativa inicial 12h
TA 3.1	Criar uma função na qual todas as simulações sejam distribuídas de acordo com qual assunto pertença (Efetuado com Sucesso).

Fonte: Autor da Pesquisa, 2011.

A figura acima representa a tabela user stories do sistema BANCO DE SIMULAÇÔES, tendo em vista todas as suas funcionalidades, e estimativas de tempo (prazos).

Figura 14: Modelo de Tarefa



Fonte: Autor da Pesquisa, 2011.

A figura acima representa a modelo de tarefas do sistema BANCO DE SIMULAÇÔES, tendo em vista todas as suas funcionalidades, e a produção do software.

4.5.2 DESENVOLVIMENTO DA EQUIPE

A equipe trabalhou bem entrosada, onde se primou pelo debate de idéias, dúvidas e opiniões. Quanto ao trabalho, a divisão foi sugestiva vendo que cada membro da equipe adquiriu um papel conforme suas respectivas capacidades, com interação e dando suporte aos demais. O estudo foi conjunto, refletido assim, em um aprendizado mais rápido e satisfatório.

4.5.3 DIFICULDADES

Quanto a às dificuldades, foram relacionados à distância geográfica, já que membros da equipe reside em outra cidade. Mais dificuldade no tocante ao aprendizado da disciplina educacional e, ainda, adaptação aos prazos estipulados pelo cliente.

4.5.4 RISCOS

No que diz respeito a riscos, eles existem e foram trabalhados ao máximo, com o prazo de entrega sendo, sem dúvida, um deles. Outro seria a confirmação do real aprendizado de tecnologia mais complexas e disciplina educacional para produção de artefatos tendo ainda em vista outros riscos de menos força mais devidamente tradado.

4.5.5 TECNOLOGIA E DICIPLINA EDUCACIONAL APRENDIDA

As tecnologias aprendidas foram: NetBeans e CorelDraw. Suas dificuldades se concentram no fato de ter que as aprendes para produção do produto. Quanto à disciplina educacional teve sua dificuldade no processo de que se tratando de algo que diverge da área da computação e o torna difícil.

4.5.6 RESULTADOS OBTIDOS

O resultado obtido com o processo de desenvolvimento do software foi à satisfação do cliente, tendo em vista que software entregue no prazo e com suas funcionalidades implementadas. As funcionalidades implementadas foram as seguintes:

- Uma pagina com exercício proposto sobre o assunto e simulações;
- Pagina com simulaçõe

O software BANCO DE SIMULAÇÕES precisa ter seus alicerces de desenvolvimento embasados em organização, estratégia e controle. O uso de YP (easYProcess) em sincronia com outras tecnologias possibilitou alcançar esses objetivos e, além disso, propiciou conhecimento mais profundo e experiência profissional.

5 CONSIDERAÇÕES FINAIS

Easy Process é certamente em alguns aspectos uma metodologia que rompe paradigmas antigos da Engenharia de Software à medida que seu desenvolvimento já está consolidado.

O YP é uma metodologia voltada para o aumento de qualidade e produtividade no desenvolvimento de software. Para implementar o YP é preciso fazer a equipe se unir em torno de algumas práticas simples. Ela pode ser implementada aos poucos, porém a maior parte das práticas são essenciais. Nem todos os projetos são bons candidatos para o uso do YP.

Por rompe paradigmas tão antigos e propor práticas tão diferentes, YP não tem seu valor reconhecido unanimemente entre a comunidade de analistas e desenvolvedores.

A visão para o desenvolvimento de software apresentada por YP promete: reduzir o risco do projeto com seu planejamento e teste, aumenta a produtividade das equipes e a qualidade do software que é entregue dentro do prazo e cronogramas planejados.

Adotando uma abordagem minimalista, YP requer uma alta disciplina e espírito colaborativo entre os membros da equipe. Levando-se em consideração os valores, princípios e requisitos necessários, é possível tirar proveito de seu principal ponto forte: o desenvolvimento do software em ambientes estáveis que exige respostas rápidas às constantes mudanças que lhe são inerentes.

Portanto, trata-se de uma metodologia que tem seu mérito por propor soluções inovadoras, humanas, criativas, objetivas e ágeis e que trazem resultados reais em alguns contextos, além de romper corajosamente paradigmas ultrapassados do desenvolvimento de sistemas e dar novos ares a esta ciência.

5.1 CONTRIBUIÇÕES

Para a elaboração do projeto, foi feito um estudo sobre a linguagem de programação orientada a objeto JAVA e a disciplina educacional de Física que foi o núcleo central deste projeto.

As pesquisas referentes às ferramentas, material e tecnologias envolvidas nessa monografia compõem as principais contribuições e, colaborando ainda, como material de estudos para trabalhos futuros.

5.2 LIMITAÇÕES

Uma das principais limitações existentes foi desconhecimento de algumas tecnologias, ferramentas e disciplina educacional, complicando ainda mais devido ao alto grau de complexidade e/ou difícil acesso de algumas. Outras limitação foi quanto à distancia geográfica e dos empregos tanto dos membros da equipe desenvolvedora, quanto do cliente.

REFERÊNCIAS

AMBLER S. W. **Modelagem Ágil – praticas eficazes para a Programação eXtrema e o Processo Unificado.** São Paulo: Addison Wesley, 2004.

BECK, K. **Extreme programming Explained: Embrace Change.** [S.l.] Addison Wesley Professional, 2004.

COCKBURN, A. **Agile Software Development.** 3th ed. [S.l.] Addison Wesley Professional, 2002.

CONALLEN, J. **Building Web Applications With UML.** Addison Wesley, 2000.

LIMA, A. F; SILVA, M. A. F; **Analisando e Utilizando o easYProcess(YP),** 2010.

PRESSMAN, R. S. **Engenharia de Software.** São Paulo: Makron Books, 2005.

SOMMERVILLE, Ian. **Engenharia de Software.** 6ª ed. São Paulo: Pearson, 2003.

YP, easY Process. 2003 disponível em: http://www.dsc.ufcg.edu.br/~yp. Acesso em: 25 de maio de 2011.