**UNIVERSIDADE ESTADUAL DA PARAÍBA**
**CAMPUS VII - GOVERNDOR ANTÔNIO MARIZ**
**CENTRO DE CIÊNCIAS EXATAS E SOCIAIS APLICADAS - CCEA**
**CURSO DE LICENCIATURA EM COMPUTAÇÃO**

**ANGÉLICA FELIX MEDEIROS**

# ELICITAÇÃO DE CRITÉRIOS ESSENCIAIS PARA A ADAPTAÇÃO DE UMA METODOLOGIA ÁGIL PARA O DESENVOLVIMENTO DE SOFTWARE EDUCATIVO

PATOS – PB
2012

**ANGÉLICA FELIX MEDEIROS**

# ELICITAÇÃO DE CRITÉRIOS ESSENCIAIS PARA A ADAPTAÇÃO DE UMA METODOLOGIA ÁGIL PARA O DESENVOLVIMENTO DE SOFTWARE EDUCATIVO

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

Orientador: Prof. MSc. Pablo Ribeiro Suárez

PATOS – PB
2012

M488e MEDEIROS, Angélica Felix

Elicitação de Critérios Essenciais para a Adaptação de uma Metodologia Ágil para o Desenvolvimento de Software Educativo.

Patos: UEPB, 2012.

82f.

Monografia (TRABALHO Acadêmico Orientado - (TAO) - Universidade Estadual da Paraíba.

Orientador: prof. Msc. Pablo Ribeiro Suárez

1. Engenharia de Software 2. Engenharia de Software Educativo I. Titulo II. Suárez, Pablo Ribeiro.

CDD 005.1

**ANGÉLICA FELIX MEDEIROS**

# ELICITAÇÃO DE CRITÉRIOS ESSENCIAIS PARA A ADAPTAÇÃO DE UMA METODOLOGIA ÁGIL PARA O DESENVOLVIMENTO DE SOFTWARE EDUCATIVO

Monografia apresentada 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 03/09/2012

Prof. MSc. Pablo Ribeiro Suárez / UEPB
Orientador

Profª. MSc. Cheyenne Ribeiro Guedes Isidro Abilio / UEPB
Examinadora

Profª. MSc. Janine Vicente Dias / UEPB
Examinadora

## DEDICATÓRIA

Ofereço este trabalho a todos que direta ou indiretamente me ajudaram a concluir mais esta jornada, primeiramente a Deus; ao meu orientador, pelo incentivo e apoio; e aos familiares, namorado e amigos por compreenderem a minha ausência, durante a realização deste trabalho.

## AGRADECIMENTOS

À Deus, por ter me proporcionado esta incrível experiência, por ter iluminado em cada decisão a ser tomada, clareando o meu caminho durante esta jornada e me guiando sempre pelo melhor caminho.

Aos meus pais, avós, irmãos e demais familiares por terem compartilhado os meus ideais, me incentivando a prosseguir na jornada, fossem quais fossem os obstáculos;

Ao professor, orientador e amigo Pablo Ribeiro Suárez pelo tempo e incentivo dedicados que tornaram possível a conclusão desta monografia. Obrigada por ter me confiado as suas boas ideias e genialidade.

A todos os meus professores da Universidade Estadual da Paraíba que foram tão importantes na minha vida acadêmica e no desenvolvimento desta monografia, me direcionando ao caminho do conhecimento.

Aos meus colegas de turma, em especial a Betoven, por ter me ajudado no projeto utilizado para o estudo de caso deste trabalho. Agradeço pela oportunidade de conviver com cada um de vocês, por todas as vezes que não me deixaram desistir nem temer os desafios. Agradeço os sorrisos, lágrimas, sonhos e até decepções compartilhadas, Não poderia ter melhores companheiros de guerra, espero que as amizades formadas durem tanto quanto foram intensas.

A minha colega de turma, Cúmplice Intelectual e hoje amiga, Ayslânya. Nós sorrimos, sofremos, choramos e vencemos juntas. Você esteve ao meu lado nos momentos mais difíceis desta caminhada, me apoiando incondicionalmente a cada passo. A estrada foi longa, foram intermináveis horas de estudo, de trabalhos e planos mirabolantes para um futuro enriquecedor que nunca chega... Mas juntas encontramos a força e o incentivo necessário para que continuássemos caminhando. Espero que todos os nossos sonhos se realizem e que nosso mestrado venha todo trabalhado na agilidade e na fortuna.

Aos meus amigos que acompanharam este tempo de ausência, por força das minhas ocupações acadêmicas ao longo destes longos anos, e entenderam o isolamento nos intermináveis momentos de estudo.

Por fim, agradeço em especial ao meu amigo, companheiro e namorado, Vinicius, pela compreensão nos momentos difíceis, pelo ombro amigo para ouvir meus desabafos e silêncios, pelos momentos de ausência e extremos de alegria e de tristeza. O meu "muito obrigado" é tão sincero e tão intenso quanto o nosso amor.

Loucura é fazer a mesma coisa e esperar um resultado diferente.
Albert Einstein

## RESUMO

A Tecnologia da Informação vem trazendo grandes mudanças, sobretudo na maneira como as pessoas interagem e aprendem. Contudo, ainda é um grande desafio produzir softwares que atendam as necessidades pedagógicas do processo educacional. A principal dificuldade está em desenvolver Softwares Educativos (SE) que contemplem todas as necessidades de aprendizagem. Este trabalho destaca a importância de adotar práticas da Engenharia de Software em projetos de desenvolvimento de SE, analisando os métodos ágeis como possíveis soluções para os problemas relacionados a estes aplicativos. Existem no mercado vários métodos disponíveis que utilizam a abordagem ágil e que apresentam uma série de atividades semelhantes no seu processo de desenvolvimento. Um destes métodos é o Processo de Desenvolvimento de Software EasYProcess (YP) que foi criado com o intuito de atender o desenvolvimento de projetos acadêmicos. Por isso é simples e de fácil entendimento pelos integrantes da equipe, porém é completo e robusto a ponto de gerar produtos de qualidade, sendo ainda passível de adaptação ao ambiente de aplicação. Tendo em vista os aspectos observados, este trabalho analisa o YP, como uma alternativa para o desenvolvimento de softwares educativos. No entanto, a construção de um SE aborda questões específicas que diferem dos sistemas tradicionais e, portanto, foi necessário elencar alguns critérios para adaptar este processo ao desenvolvimento de software educativo surgindo assim o YPEduc. A metodologia da pesquisa envolveu um estudo de caso onde o YPEduc foi posto em prática num miniprojeto de desenvolvimento de SE, além de um estudo exploratório para mensurar sob um ponto de vista pedagógico a sua contribuição. Como conclusão, têm-se a importância da implantação do YPEduc por alunos da UEPB na disciplina de Engenharia de Software quando são realizados projetos de desenvolvimento de SE, para que possam ser mensuradas as verdadeiras contribuições desta metodologia.



**PALAVRAS-CHAVE:** Informática na Educação, Software Educativo, Metodologias Ágeis, YPEduc.

## ABSTRACT

The Information Technology is bringing big changes, especially in the way how people interact and learn. However, it is still a great challenge to produce software which respond the pedagogical needs of the educational process. The main difficulty is to develop educational software (ES) which address all the learning process needs. This paper emphasizes the importance of adopting practices of software engineering in SE development projects, analyzing fast methods as possible solutions to problems related to these software. There are several methods available in the market which use the fast approach and present a series of similar activities in their development process. One of these methods is the Software Development Process EasYProcess (YP) which was created with the intention of assisting the development of academic projects. So it is simple and easy to understand by team members, but it is complete and robust enough to generate high quality products, being also capable of adapting to the environment of application. Considering the observed aspects, this paper analyzes the YP as an alternative for the development of educational software. However, the construction of an ES approaches specific issues that differ from the traditional systems, and therefore, it was necessary to list some criteria in order to adapt this process to developing educational software, thus resulting in the YPEduc. The research methodology involved a case study where the YPEduc was put into practice in a mini development ES project, and also an exploratory study to measure under a pedagogical point of view its contribution. As a conclusion, we highlight the importance of deploying YPEduc by UEPB students in the subject of software engineering when development projects in ES are performed, which can be measured for the real contributions of this methodology.



**KEYWORDS:** Informatics in Education, Educational Software, Agile Methodologies, YPEduc.

## LISTA DE FIGURAS

**Figura 1 -** As camadas da Engenharia de Software 19
**Figura 2** - Atividades básicas do desenvolvimento de software 20
**Figura 3 -** Ciclo de vida de software **22**
**Figura 4** - Ciclo de vida simplificado do XP 27
**Figura 5 -** Ciclo de desenvolvimento em Scrum 28
**Figura 6** - Síntese do Fluxo do Processo de Desenvolvimento YP 30
**Figura 7** - Zonas de sombra em projeto de software 32
**Figura 8** - Fluxo do Processo easYProcess 36
**Figura 9 -** Fluxo do Processo YPEduc 45
**Figura 10** - Tela Inicial 59
**Figura 11** - Mudança de nível 01 59
**Figura 12 -** Mudança de nível 02 60
**Figura 13** - Telas do Jogo 60
**Figura 14 -** Tela Final 61
**Figura 15 -** Modelo Lógico de Dados 64

## LISTA DE QUADROS

**Quadro 1** - Critérios de Adaptação nas Etapas no YP....44
**Quadro 2 -** Escopo do Problema....51
**Quadro 3** - Definição de Papéis....51
**Quadro 4 -** Documento de Visão....52
**Quadro 5 -** Definição dos Requisitos....53
**Quadro 6 -** Perfil do Aluno....54
**Quadro 7 -** Perfil do Professor Mediador....55
**Quadro 8 -** Objetivos de Usabilidade....56
**Quadro 9** - Modelagem da Tarefa....57
**Quadro 10 -** User Stories e Testes de Aceitação....58
**Quadro 11 -** Projeto Arquitetural....61
**Quadro 12** - Plano de Release....64
**Quadro 13 -** Plano de Iteração....65
**Quadro 14 -** Análise de Risco....69
**Quadro 15 -** BIG CHART....70
**Quadro 16 -** Teste de Usabilidade....70
**Quadro 17 -** Teste de Aprendizagem....71
**Quadro 18 -** Teste de Qualidade Pedagógica....72
**Quadro 19** - Análise do Processo de Desenvolvimento YPEduc....74

**SUMÁRIO**

**1 INTRODUÇÃO** ..... 12
1.1 CENÁRIO TÉCNICO-CIENTÍFICO ..... 12
1.2 DEFINIÇÃO DA PROBLEMÁTICA ..... 13
1.3 OBJETIVOS ..... 14
1.4 JUSTIFICATIVA ..... 15
1.5 METODOLOGIA ..... 15
1.6 ESTRUTURA DO TRABALHO ..... 16
**2 PROCESSO DE DESENVOLVIMENTO DE SOFTWARE EDUCATIVO** ..... 18
2.1 ENGENHARIA DE SOFTWARE ..... 18
2.2 ENGENHARIA DE SOFTWARE E SOFTWARE EDUCATIVO ..... 20
2.3 METODOLOGIAS ÁGEIS ..... 24
2.3.1 Extreme Programming (XP) ..... 25
2.3.2 Scrum ..... 27
2.3.3 easYProcess ..... 29
2.4 AGILIDADE NO DESENVOLVIMENTO DE SOFTWARE EDUCATIVO ..... 31
**3 ADAPTAÇÃO DO EASYPROCESS PARA O DESENVOLVIMENTO DE SOFWARE EDUCATIVO** ..... 35
3.1 EASYPROCESS ..... 35
3.1.1 As Fases do Processo ..... 36
3.2 CRITÉRIOS DE ADAPTAÇÃO ..... 39
3.3 O YP ADAPTADO AO CENÁRIO EDUCACIONAL: YPEDUC ..... 45
3.3.1 Identificação do Escopo do Problema ..... 46
3.3.2 Especificação de Papéis ..... 46
3.3.3 Conversa com o Cliente ..... 46
3.3.4 Inicialização ..... 47
3.3.5 Planejamento (Release e Iteração) ..... 48
3.3.6 Implementação ..... 49
3.3.7 Versão do Produto ..... 49
**4 ESTUDO DE CASO** ..... 51
4.1 MINIPROJETO PARA O DESENVOLVIMENTO DO SOFTWARE EDUCATIVO: RECICLA ..... 51
4. 2 ANÁLISE DO PROCESSO DE DESENVOLVIMENTO YPEDUC ..... 73

**5 CONCLUSÕES E CONSIDERAÇÕES FINAIS** ..... 76
5.1 CONTRIBUIÇÕES ..... 76
5.2 LIMITAÇÕES ..... 77
5.3 TRABALHOS FUTUROS ..... 77
**REFERENCIAS** ..... 78

# 1 INTRODUÇÃO

## 1.1 CENÁRIO TÉCNICO-CIENTÍFICO

Os avanços tecnológicos têm modificado profundamente o contexto educacional, aumentando a necessidade de agregar recursos computacionais às práticas educativas. Para Rezende (2000), a tecnologia educacional deve adequar-se às necessidades dos projetos político pedagógicos, colocando-os a serviço de seus objetivos e nunca os determinando.

Para Bassani et al (2006), a maioria dos aplicativos comerciais enfatizam o processo, a execução de tarefas, o apoio à tomada de decisão, entre outros, enquanto o software educativo visa atender e promover a aprendizagem.

Com a demanda do mercado de software exigindo que tecnologias se aprimorem de forma acelerada, a produção de software se tornou muito complexa, exigindo que técnicas fossem desenvolvidas para melhorar a sua produção. Neste sentido, diversos estudos vêm sendo desenvolvidos almejando relacionar definições da área da Engenharia de Software com as necessidades da área educacional.

A partir da necessidade de processos cada vez mais confiáveis e eficientes, e com o objetivo de agilizar o desenvolvimento de software, surgiu o Movimento Ágil (AGILE MANIFESTO, 2001) que propõe um novo paradigma de desenvolvimento, no qual é priorizada a produção de código ao invés de extensa documentação, além de várias outras características.

Um processo ágil implica em um método adaptativo e leve, que facilmente responde a mudanças (LARMAN, 2002), optando por uma documentação apropriada evitando redundâncias e excessos, para que auxilie efetivamente o desenvolvimento do software (COSTA FILHO et al, 2005).

Tais métodos têm se tornado cada vez mais populares e sendo cada vez mais utilizados. É fundamental ressaltar ainda que os métodos classificados como ágeis possuem em comum o fato de serem aplicados em projetos não muito complexos, utilizando ciclos iterativos curtos, planejamento guiado por funcionalidades, tolerância a mudanças e proximidade entre a equipe e o cliente (HIGHSMITH, 2002).

Estas características fazem das metodologias ágeis possíveis soluções para os problemas relacionados ao desenvolvimento de softwares educativos. No entanto, ainda não se encontram diretrizes detalhadas de como aplicá-los a softwares educacionais.

Segundo Bassani et al (2006), algumas metodologias vêm sendo propostas para o processo de desenvolvimento de software educativo, entretanto, a autora destaca que tais metodologias são tendenciosas quanto aos aspectos educacionais e psicológicos, desviando-se dos aspectos computacionais.

Garcia et al (2004) afirma que a academia é o ambiente mais propício para que a teoria seja aliada à prática a fim de proporcionar ao estudante uma visão real do trabalho com Engenharia de Software.

Por isso tudo, estudos envolvendo o processo de desenvolvimento de softwares educativos vêm ganhando mais espaço no meio acadêmico.

E partindo do pressuposto que o sucesso é mais facilmente obtido quando se utiliza um processo de desenvolvimento de software adequado, se faz necessária uma metodologia que contemple os benefícios da agilidade, além de ser capaz de sistematizar o processo de desenvolvimento de software educativo no meio acadêmico.

## 1.2 DEFINIÇÃO DA PROBLEMÁTICA

O software educativo é um recurso de imensurável importância na aprendizagem, trazendo de forma lúdica e prazerosa o desenvolvimento prático dos conhecimentos escolares, podendo auxiliar o educador como um recurso a mais em sua prática educativa no processo de ensino-aprendizagem (SILVA, 2009).

No entanto, alguns softwares educativos não passam por processos de desenvolvimento que garantam uma qualidade pedagógica efetiva. Lacerda (2007) enfatiza que uma das principais causas para tais problemas é a falta de diretrizes para o processo de desenvolvimento deste tipo de aplicação.

Devido à especificidade do uso destes produtos, as questões colocadas à equipe de desenvolvimento são muito peculiares, pois além das dificuldades gerais inerentes ao desenvolvimento de software (por exemplo: incorporação de novos requisitos aos protótipos, manutenção de softwares, etc.) soma-se uma nova dimensão: a educacional (PERRY, 2006).

A partir da necessidade de processos cada vez mais rápidos, confiáveis e eficientes, com o objetivo de agilizar o desenvolvimento de software, surgiu o Movimento Ágil. Soares (2004) sintetiza a ideia dos métodos ágeis, reforçando o enfoque nas pessoas e não em processos ou algoritmos. Além disso, existe a preocupação de gastar menos tempo com documentação e mais com a implementação. Uma característica das metodologias ágeis é que elas são adaptativas ao invés de serem preditivas. Com isso, elas se adaptam a novos fatores

decorrentes do desenvolvimento do projeto, ao invés de procurar analisar previamente tudo o que pode acontecer no decorrer do desenvolvimento.

Alguns processos representantes do desenvolvimento ágil são XP (Extreme Programming) e o SCRUM, que são consideradas metodologias compatíveis e complementares, não sendo raro vermos empresas que utilizam ambas as metodologias. O XP é mais focado em práticas integradas de engenharia de software enquanto que o SCRUM foca-se nas práticas de gerenciamento do projeto (NETO, 2008).

Outro exemplo entre as metodologias ágeis é o easYProcess (PET/UFCG, 2007), que foi uma metodologia desenvolvida através de um projeto da Universidade Federal de Campina Grande (UFCG).

O easYProcess (YP) é uma metodologia de desenvolvimento de software criada com o intuito de atender ao desenvolvimento de projetos acadêmicos e por isso é simples e de fácil entendimento pelos integrantes da equipe, porém é completo e robusto a ponto de gerar produtos de qualidade.

Tendo em vista os aspectos observados, levantou-se a seguinte questão: O YP é passível a adaptações a fim de atender as especificidades do desenvolvimento de um software educativo? Para respondê-la, traçaram-se os objetivos da pesquisa de maneira que se harmonizassem e contribuíssem com tal propósito.

## 1.3 OBJETIVOS

Este trabalho tem como objetivo geral elencar critérios de adaptação para o processo de desenvolvimento de Software easYProcess, a fim de adequar a sua aplicação ao desenvolvimento de software educativo.

Para tal, os seguintes objetivos específicos serão realizados:

- Propiciar uma reflexão sobre a importância do planejamento no projeto de softwares educacionais;
- Descrever o processo de desenvolvimento YP adaptando suas etapas para o contexto educacional;
- Avaliar a metodologia adaptada através da sua implantação em um miniprojeto de desenvolvimento de software educativo;
- Analisar sob um ponto de vista pedagógico a contribuição do YPEduc;

## 1.4 JUSTIFICATIVA

Possibilitar um aprendizado interativo e dinâmico é um dos grandes desafios para o novo contexto educacional em que a sociedade se encontra, e neste sentido vale ressaltar, que dentre os recursos existentes, o computador é um dos mais poderosos, porque além do texto e da imagem, coloca à disposição dos usuários o som, o movimento e a interatividade.

Para tanto, surge a necessidade do desenvolvimento de softwares educativos nos quais professores e alunos tenham a possibilidade de participar durante todo o processo de planejamento e modelagem do mesmo, bem como do design de uma interface mais favorável para aplicação educacional, para que assim, este tipo de software contenha particularidades que assegurem a sua aplicabilidade e usabilidade, possibilitando maior desempenho no processo de aprendizagem.

Dessa forma, utilizar outro olhar para o processo de desenvolvimento de softwares educativos pode ser imprescindível para que importantes alternativas apareçam e facilitem o entendimento da situação educativa mediada por tecnologias vivenciada em sala de aula, tendo como consequência softwares de fato inovadores.

Dado o exposto, se faz necessária uma metodologia que contemple os benefícios da agilidade, além de ser capaz de sistematizar o processo de desenvolvimento de software educativo no meio acadêmico.

## 1.5 METODOLOGIA

A pesquisa é uma atividade direcionada para a investigação de problemas teóricos ou práticos através da utilização de processos científicos. Ela surge de uma duvida ou problema e, com o uso do método cientifico, busca uma resposta ou solução (CERVO et al, 2007, p. 57).

Gil (2010) propõe alguns critérios para a classificação dos tipos de pesquisas, deste modo, quanto aos objetivos mais gerais o estudo é exploratório e quanto ao método aplicado adotou-se o estudo de caso.

A presente pesquisa visa a aquisição de conhecimentos voltada para a aplicação numa situação específica, além disto, como um dos objetivos deste trabalho é mensurar sob um ponto de vista pedagógico a contribuição do YPEduc, o estudo exploratório foi de extrema relevância, já que o mesmo proporciona maior familiaridade com o problema apresentado, com vistas a torná-lo mais explícito ou a construir hipóteses (GIL, 2010).

Uma vez que este trabalho fundamenta-se na concepção de buscar melhorias para o desenvolvimento de softwares educativos na instituição, o procedimento técnico utilizado foi o estudo de caso, permitindo uma melhor compreensão do ambiente e, consequentemente, abrindo espaço para um posterior estudo.

Para GIL (2010), praticamente toda pesquisa acadêmica necessita em algum momento da realização de pesquisa bibliográfica, sendo assim, este trabalho teve como primeiro passo o levantamento da literatura existente a respeito das áreas envolvidas pelo tema proposto, dentre elas, Engenharia de Software, Metodologias Ágeis e Processos de Desenvolvimento de Softwares Educativos.

No segundo momento foi feita a análise em algumas metodologias ágeis com o objetivo de compará-las, e desta forma encontrar pontos passíveis de adaptação para aplicações no contexto educacional.

Realizada a identificação destes pontos, a metodologia com maior passividade de adequação foi adaptada para o desenvolvimento de softwares educativos, enfatizando a importância do uso de técnicas e métodos que envolvam o usuário no processo de desenvolvimento do SE.

Por fim, foi desenvolvido um miniprojeto para o desenvolvimento de um software educativo utilizando o YPEduc. Tal desenvolvimento teve como intuito analisar a adequação dos critérios levantados à metodologia ágil, com isso a validação conseguida não versou em torno da metodologia em si, sendo para isso necessário um desenvolvimento real que tivesse um cliente real, um domínio específico real de qualquer área de conhecimento e uma equipe multidisciplinar disponível para a elaboração do sistema capaz de sanar as necessidades presentes neste domínio. Em suma, na prática foi feita uma análise sob um ponto de vista pedagógico acerca da viabilidade e da contribuição da metodologia adaptada para, no setor acadêmico, se obter software educativo.

## 1.6 ESTRUTURA DO TRABALHO

Este capítulo 1 apresentou a introdução do presente trabalho, expondo o cenário-técnico-científico, definindo a problemática, delimitando os objetivos, enfatizando a sua justificativa e determinando a metodologia da pesquisa.

O capítulo 2 apresenta o estado da arte, que será construído a partir dos estudos bibliográficos já realizados, fundamentando a reflexão e a argumentação.

No capítulo 3, serão elencados os critérios de adaptação propostos para que a metodologia easYProcess se ajuste ao desenvolvimento de software educativo.

No capítulo 4 é apresentado o estudo de caso, onde a metodologia será avaliada através de um miniprojeto e serão discutidos os resultados da pesquisa.

Por fim, no capítulo 5 serão mostradas as conclusões obtidas com o trabalho, suas contribuições e as sugestões para trabalhos futuros.

# 2 PROCESSO DE DESENVOLVIMENTO DE SOFTWARE EDUCATIVO

Este capítulo está dividido em quatro seções. Na primeira, são apresentados conceitos de Engenharia de Software, enaltecendo os benefícios de suas práticas. Na segunda seção, o tema abordado é Engenharia de Software e Software Educativo, ressaltando a importância de utilizar técnicas e métodos específicos em projetos de desenvolvimento de SE. Na terceira, são apresentados os conceitos que envolvem as Metodologias Ágeis, bem como suas principais características e práticas, destacando ainda três representantes deste movimento: Extreme Programming, SCRUM e easYProcess. Na quarta e última parte deste capítulo, é feita uma abordagem sobre a Agilidade no Desenvolvimento de Software Educativo, apresentando os principais pontos relacionados a este tema.

## 2.1 ENGENHARIA DE SOFTWARE

Sommerville (2003, p.5) define muito bem a Engenharia de Software como sendo uma área da informática preocupada com a especificação, o desenvolvimento e a manutenção de softwares, ou seja, todos os aspectos da sua produção. Ela se dispõe de tecnologias e práticas, como gerência de projetos, com o objetivo de tornar o software mais organizado, produtivo e de mais qualidade.

A Engenharia de Software é peça fundamental no planejamento de software, pois para um software ser um produto de qualidade deve possuir algumas características chave, dentre elas: eficiência, facilidade de manutenção, facilidade de uso, nível de confiança (LACERDA, 2007).

Neste sentido, os fundamentos que amparam a Engenharia de Software envolvem o uso de modelos abstratos e precisos que permitem especificar, projetar, implementar e manter sistemas de software, avaliando e garantindo suas qualidades.

Carvalho (2001) acrescenta ainda que a Engenharia de Software enquanto disciplina, reúne metodologias, métodos e ferramentas a serem utilizados, visando solucionar problemas inerentes ao processo de desenvolvimento e ao produto de software.

Para Pressman (2001), a Engenharia de Software pode ser entendida através de camadas que abrangem três elementos fundamentais: ferramentas, métodos e processos. Conforme a Figura 1, cada um destes elementos corresponde a uma camada. Os elementos representados nas três primeiras camadas devem possibilitar ao gerente de software o domínio

do processo de desenvolvimento de software e proporcionar ao desenvolvedor embasamento para a construção de software de qualidade.

**Figura 1 -** As camadas da Engenharia de Software


**Fonte:** Próprio Autor adaptado de PRESSMAN (2001)

As ferramentas proporcionam apoio automatizado ou semiautomatizado aos métodos, enquanto estes abrangem um extenso conjunto de tarefas que incluem: planejamento e estimativa de projeto, análise de requisitos, projeto da estrutura de dados, arquitetura de programa e algoritmo de processamento, codificação, teste, manutenção, etc.

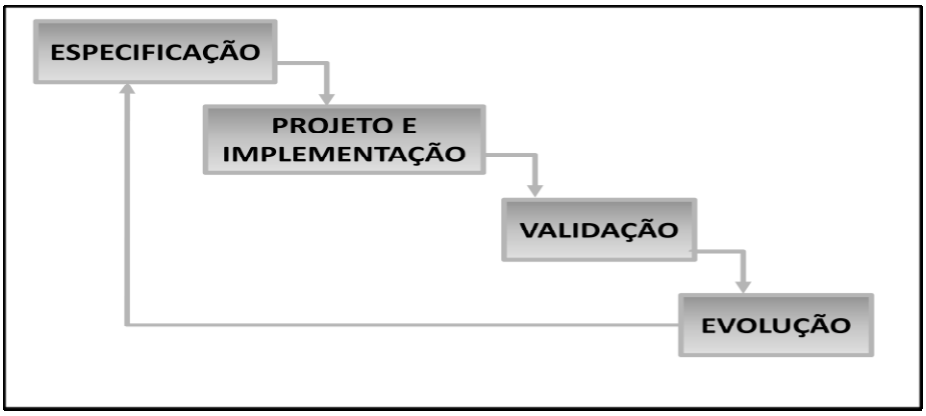
Sommerville (2003) sintetiza os métodos de Engenharia de Software como sendo abordagens para o desenvolvimento de software e que possui como objetivo facilitar a produção de um software a fim de torná-lo um produto de alta qualidade e ter uma boa relação custo-benefício.

O processo é a camada essencial da Engenharia de Software (PRESSMAN, 2001), pois determina ações práticas a serem realizadas pela equipe de projeto. Desta forma, um processo define a sequência em que os métodos serão aplicados, como os artefatos serão entregues, como a qualidade será garantida e a adaptabilidade às mudanças, e ainda determina os pontos que irão possibilitar aos gerentes de software mensurar o progresso do desenvolvimento.

Um processo de desenvolvimento de software é representado por um modelo, enquanto que o modelo é operacionalizado por meio de uma metodologia (GIRAFFA et al, 2005), onde podem ser vistos modelos de processos de desenvolvimento como sendo uma descrição simplificada do processo de software, ou seja, é uma abstração do processo real que está sendo descrito.

Ao longo da história da Engenharia de Software foram concebidos vários destes modelos e todos compartilham de atividades básicas como Especificação, Projeto e Implementação, Validação e Evolução, como mostra a Figura 2.

**Figura 2 -** Atividades básicas do desenvolvimento de software




**Fonte:** Próprio Autor adaptado de SOMMERVILLE (2003)

Para Sommerville (2007), estas atividades são instanciadas com o mesmo nome ou com nome diferente e são organizadas de modo peculiar nos diversos processos de desenvolvimento e realizam as seguintes funções:

- Especificação: Onde é definida a funcionalidade do software e as restrições em sua execução;
- Projeto e implementação: O software deve ser produzido de modo que atenda as devidas especificações;
- Validação: Deve garantir que o software faz o que o cliente almeja;
- Evolução: O software deve evoluir para atender às necessidades versáteis do cliente.

Existem diversos modelos de processo de desenvolvimento de software, e cada modelo pode ter mais de uma metodologia que o operacionaliza. A metodologia estabelece a sucessão das atividades e seus relacionamentos entre si, definindo o momento em que os métodos e as ferramentas serão utilizados (PRESSMAN, 2001).

Neste sentido, um processo de desenvolvimento de software deve ser executado respeitando um modelo previamente determinado, adotando uma metodologia que se adapte às necessidades e objetivos existentes, e tudo isto deve servir de norte para o emprego adequado dos métodos e das ferramentas, tendo sempre como foco a qualidade.

## 2.2 ENGENHARIA DE SOFTWARE E SOFTWARE EDUCATIVO

Graças ao surgimento da computação pervasiva, o computador passou de mero artefato tecnológico a elemento da cultura humana, se estabelecendo, inclusive, dentro do

espaço escolar como um agente decisivo no processo de ensino. A escola calcada apenas no saber do professor e dos livros, não corresponde mais a uma sociedade que respira tecnologia (LAGO, 2004, p.4).

As novas tecnologias mostram que, quando utilizadas adequadamente, auxiliam no processo da construção do conhecimento, tornando o processo de ensino-aprendizagem mais estimulante e eficaz (JUCÁ, 2006), tais tecnologias podem ser representadas desde uma ferramenta de apoio para resolução de problemas, até um complexo instrumento de ensino capaz de motivar e estimular o aluno a construir seu próprio conhecimento.

Sendo assim, com a introdução do computador como mediador didático, desenvolveram-se softwares específicos para serem utilizados em contextos de ensino–aprendizagem, e estes passaram a ser denominados de Softwares Educativos.

Gomes e Wanderley (2003) definem Softwares Educativos (SE) como sendo a classe de interfaces educativas ou conjunto de artefatos criados para funcionarem enquanto mediadores em atividades educativas de formação em áreas distintas do conhecimento. E destacam ainda que a função de um software educativo é a de promover aprendizagem para o uso, mas também analisar a aprendizagem de conceitos específicos que ocorrem com o uso do software.

Para atingir tal objetivo, deve-se produzir um software onde (OLIVEIRA et al, 2001):

- O processo de ensino-aprendizagem seja um processo dinâmico e ativo;
- O próprio aluno seja o construtor do seu conhecimento;
- O professor seja o facilitador do processo de aquisição do conhecimento do aluno.

Nesta perspectiva, para que um software seja utilizado com finalidade educacional ou em atividades curriculares, de modo a atender às necessidades da área de aplicação a que se destina existem alguns desafios como qualidade, manutenção e desenvolvimento burocrático.

Por tudo isso, cabe ressaltar que nem todos os softwares disponíveis no mercado e intitulados como educacionais atendem as necessidades pedagógicas. Muitos deles são dotados de alta tecnologia, mas não possuem características pedagógicas que auxiliem no processo de ensino-aprendizagem, nem uma interface intuitiva que facilite seu uso por parte dos alunos.

Benitti et al (2005) enfatiza ainda que a construção de um SE aborda questões específicas que diferem dos sistemas tradicionais e, portanto, tais questões devem ser consideradas em seu processo de desenvolvimento.

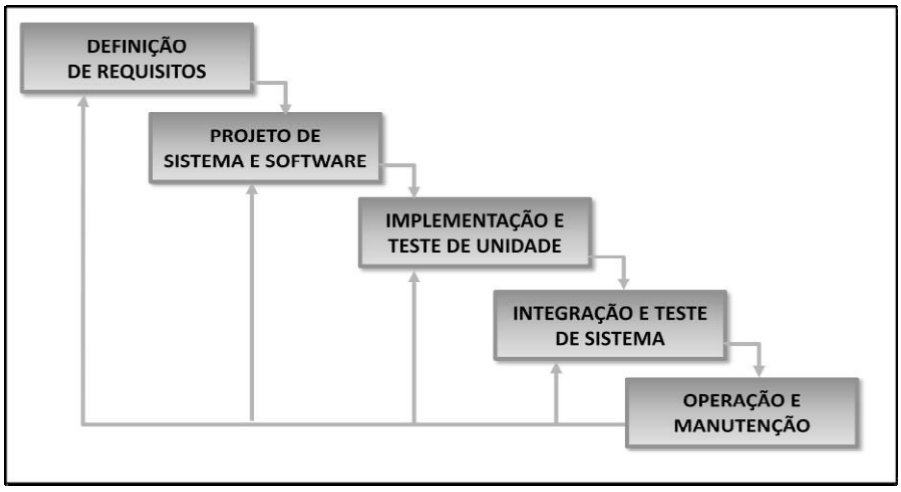
Para Gama (2007), é possível avaliar o aspecto pedagógico em softwares educativos levando em consideração os seguintes critérios:

- Objetividade: o conteúdo deve ser objetivo, com informações claras;
- Sequenciamento Instrucional: as informações devem seguir uma sequencia lógica e didática;
- Motivação: o aluno deve ser incentivado a interagir, criando expectativas e se comprometendo com o processo de aprendizagem;
- Estruturação: Os caminhos/operações do software devem ser determinados de modo que reduzam a possibilidade de erros;
- Legibilidade: as informações devem estar em linguagem apropriada para o aluno e graficamente legível;
- Avaliação: o aluno deve ter a possibilidade de se autoavaliar.

Em se tratando de softwares educativos, o processo de desenvolvimento tem que abranger tanto o funcionamento do sistema propriamente dito, quanto os mecanismos pedagógicos e didáticos que constituem a base de todo instrumento de ensino e de aprendizagem.

Sommerville (2007) define o ciclo de vida de um software como a sequência de atividades que compõem o desenvolvimento de um software. Como mostra a Figura 3, no caso de software tradicional, o ciclo de vida inclui as seguintes etapas: Definição de requisitos, projeto de sistemas e software, implementação e teste de unidade, integração e teste e sistema, operação e manutenção.

**Figura 3 -** Ciclo de vida de software




**Fonte**: Próprio Autor adaptado de SOMMERVILLE (2007)

No entanto a literatura demonstra que esta metodologia não pode ser aplicada a um SE, pois enfatizam essencialmente aspectos computacionais, a partir da identificação dos dados de entrada do sistema, especificação do processamento e a apresentação de dados de saída.

A metodologia adequada para a especificação e desenvolvimento de um software educativo deve contemplar aspectos computacionais e educacionais de interface e colaboração. Levando em consideração estes aspectos Bassani et al (2006, pág. 7) aponta os seguintes passos para o ciclo de vida de um SE:

a) Levantamento de requisitos: compreende a abordagem pedagógica, definição do tipo de software, definição do público-alvo, profundidade e abrangência do conteúdo e perspectivas de colaboração. Uma das intenções deste passo é descrever o software em termos de usabilidade, considerando proximidade com o usuário e seus modelos de aluno e com o contexto de aplicação.

b) Análise: Detalha o funcionamento do software, a fim de orientar na prototipação do software que abrange o design das telas, de interação, usabilidade, etc. Para a modelagem são sugeridos os seguintes procedimentos: Modelagem de cenários, Diagrama classes (UML), Modelo entidade-relacionamento (ER), Projeto de interface.

c) Projeto/Prototipação: Nesta fase são enraizados cada uma das funcionalidades e cenários previstos. É imprescindível a participação do usuário para que seja possível realizar os ajustes necessários. Esta análise conjunta possibilita um melhor conhecimento a cerca do perfil dos usuários, a adaptabilidade do ambiente e da interface, e realizar as correções necessárias nos requisitos de acessibilidade e usabilidade.

d) Testes/Validação: certifica-se nesta ocasião se os requisitos do software foram efetivamente atendidos.

Outro fator importante é destacado por Campos (2001) quando este afirma que para a elaboração de um software educativo de qualidade deve-se contar com uma equipe multidisciplinar envolvendo profissionais de informática, professores de conteúdo, de didática e alunos. Pois é indiscutível que cada um destes profissionais, em suas respectivas áreas, inclusive o aluno, possui seus próprios critérios de qualidade.

Campos (2001) observa ainda que para avaliar um software educativo é necessário levar em consideração as questões de tecnologia, Educação, Psicologia, Ciência Cognitiva.

Desta forma, projetos de desenvolvimento de software educacional, além de envolver em seu desenvolvimento uma equipe multidisciplinar, os produtos de software devem refletir

os objetivos educacionais propostos e o ambiente de aprendizagem almejado, criando situações que estimulem o desenvolvimento das habilidades desejadas.

Vale ressaltar ainda que além de todos estes aspectos observados, a constante busca por agilidade e adaptabilidade no processo de produção de software tem impulsionado a preferência por metodologias de desenvolvimento que são concebidas com o intuito de garantir uma maior interação e um desenvolvimento mais ágil de aplicações computacionais (FERREIRA E LIMA, 2006), este tema será aprofundado na próxima seção.

## 2.3 METODOLOGIAS ÁGEIS

O termo Metodologia Ágil tornou-se conhecido em fevereiro de 2001, quando foi criada a Aliança Ágil e instituído o Manifesto Ágil para o desenvolvimento de software, tal paradigma possui valores e práticas diferenciadas que segundo Parsons et al (2007) em alguns casos podem auxiliar projetos de desenvolvimento de software a apresentarem resultados mais satisfatórios.

Conforme Highsmith (2002), os valores do Manifesto Ágil são:

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

Além destes quatros valores pontos destacados, o manifesto ágil também apresenta doze princípios que aos quais os métodos ágeis de desenvolvimento de software devem se apoiar (AGILE MANIFESTO, 2001):

- A prioridade é satisfazer ao cliente através de entregas rápidas e frequentes;
- Mudanças nos requisitos são aceitas;
- Software deve ser entregue funcionando com a menor frequência, sempre na menor escala de tempo;
- As equipes de negócio e os desenvolvedores devem trabalhar juntos diariamente no projeto;
- Manter uma equipe motivada, fornecendo o ambiente e o suporte necessário e confiando que realizarão o trabalho;
- O modo mais eficiente e eficaz de transmitir informações dentro e fora da equipe de desenvolvimento é a comunicação face a face;

- Ter o software funcionando é a principal medida de progresso;
- Processos ágeis promovem o desenvolvimento em um ritmo sustentável.
- Atenção ininterrupta com a excelência técnica e bom design aprimoram a agilidade;
- Simplicidade é essencial;
- Os melhores requisitos, arquiteturas e design nascem de equipes organizadas;
- Em intervalos regulares, a equipe deve refletir sobre como se tornar mais competente.

Com base nos princípios descritos acima, cada um dos métodos ágeis apresenta um conjunto de atividades a serem seguidas durante o processo de desenvolvimento do sistema. Sommerville (2001) reconhece que esses métodos são uma forma estruturada de se desenvolver software com o objetivo de facilitar a sua produção com alta qualidade.

A essência do manifesto é o sentido da nova abordagem de desenvolvimento de software, amparada na agilidade, na flexibilidade, nas habilidades de comunicação e na competência de fornecer novos produtos e serviços de valor ao mercado, em curtos períodos de tempo. Desta forma, estes processos não estão vinculados apenas a pontos filosóficos da Engenharia de Software, mas são organizados e centrados em aspectos pautados a uma célere produtividade sem que exista prejuízo de qualidade.

Dentre as metodologias ágeis existentes, para este trabalho foram escolhidas algumas das que são mais utilizadas do mercado: XP (Extreme Programming) e o Scrum. Além da metodologia easYProcess (YP) que atende o desenvolvimento de projetos acadêmicos.

### 2.3.1 Extreme Programming (XP)

A metodologia XP é recomendada para equipes pequenas e médias, que implementam software baseado em requisitos não inteiramente definidos e que se modificam rapidamente (BECK, 2004).

Esta metodologia estima os valores da comunicação entre as pessoas, a simplicidade do software e do próprio processo, o feedback constante e contínuo e a coragem (COCKBURN e HIGHSMITH, 2001).

Além dos valores apresentados anteriormente a metodologia XP determina um conjunto de princípios que devem ser adotados pelas equipes de projetos. Os princípios fundamentais são: Feedback rápido, assumir simplicidade, mudança incremental, abraçar mudanças e trabalho de qualidade (BECK, 2000), tais princípios servirão para amparar na escolha de alternativas para solucionar problemas durante o andamento do projeto.

A Extreme Programming possui também doze práticas que versam na essência principal do processo e que foram designadas com embasamento nos ideais defendidos pelos valores e princípios apresentados acima.

O seu autor enumera as práticas de XP como: jogo do planejamento, releases pequenos, metáfora, projetos simples, testes constantes, refatoramento, programação em pares, propriedade coletiva do código, integração contínua, semana de quarenta horas, cliente no local, padrões de codificação.

Algumas destas práticas requerem muita disciplina e são interdependentes, completando-se e apoiando-se. Por isto, podem existir riscos na adoção de somente uma parte do conjunto de práticas.

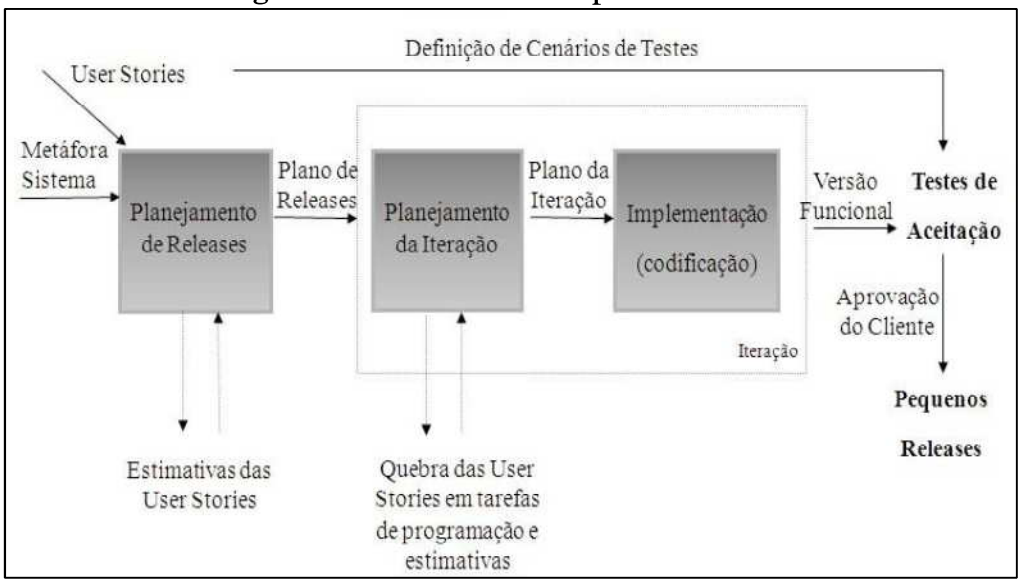
A gerência em XP é dividida em dois papéis: o treinador e o rastreador, além dos papéis gerenciais, uma equipe que utiliza XP é formada por outros papéis, tais como programador, cliente, testador e consultor (ABRAHAMSSON et al, 2002).

Um projeto XP avança algumas fases durante o seu ciclo de vida, onde estas são constituídas de várias tarefas que são executadas. O ciclo do XP é dividido em seis fases:

1. Exploração: O cliente escreve cartões de estórias, cada um descrevendo uma funcionalidade esperada para o primeiro release.
2. Planejamento: Neste momento acontece a definição de prioridades entre as estórias junto com o cliente, onde os programadores avaliam o esforço e o cronograma para cada uma das estórias.
3. Iterações para Release: Nesta fase ocorrem várias iterações até o primeiro release ser concluído. Na primeira iteração é determinado o sistema com toda a arquitetura, nas iterações posteriores as funcionalidades serão acrescentadas de acordo com as prioridades estabelecidas.
4. Validação para Produção: Nesta etapa são realizados testes extensivos e verificações para validação do software para ser utilizado em ambientes de produção.
5. Manutenção: Após o primeiro release para produção, há novas edições do sistema com novas funcionalidades.
6. Morte: Quando não há mais estórias a serem implementadas e o cliente está satisfeito com o sistema.

O ciclo de vida XP é curto e pode se destacar em um ambiente onde as mudanças de requisitos do sistema são fatores dominantes. A Figura 4 representa, de forma simplificada, o ciclo de vida do processo XP:

**Figura 4 -** Ciclo de vida simplificado do XP




**Fonte:** Próprio autor adaptado de WELLS (2002)

A partir da definição incremental dos requisitos do sistema fornecidos pelos usuários (User Stories) é produzida a metáfora do sistema que proporciona uma visão geral, em formato simples, que possa ser compreendida por programadores e clientes.

Na fase de planejamento, os requisitos são minuciosamente especificados à medida que fornecidos, em seguida, os testes são preparados a partir destas especificações e a fase de codificação é iniciada com o propósito de atender a esses testes, de forma que é estabelecida uma relação conexa e contínua entre as fases de teste e codificação.

### 2.3.2 Scrum

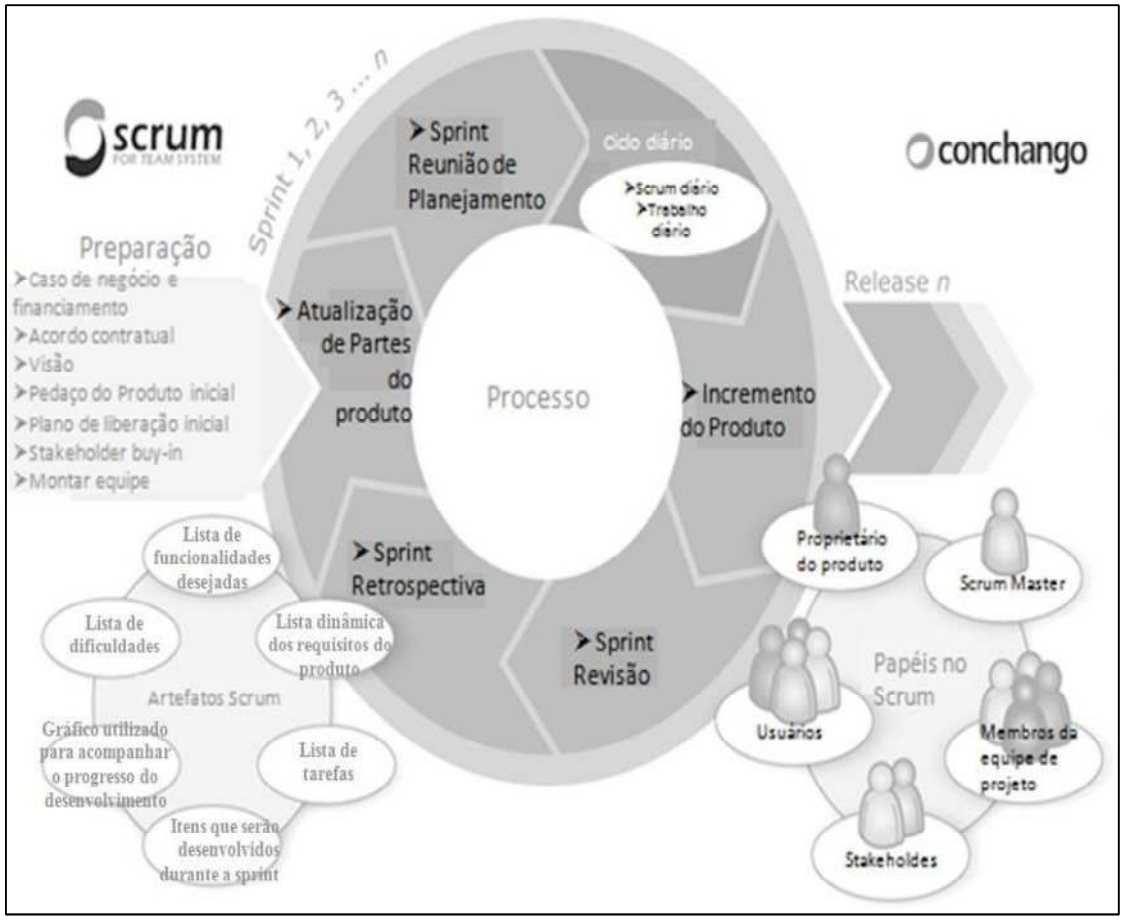
O Scrum foi desenvolvido para gerenciar o processo de desenvolvimento de software em ambientes em que os requisitos estão em constante mudança, sendo apropriado para equipes pequenas, com até dez integrantes (ABRAHAMSSON et al, 2002).

O foco da metodologia é encontrar uma forma de trabalho dos membros da equipe para produzir o software de forma flexível e em um ambiente em constante mudança.

Deste modo, a ideia fundamental do Scrum é que o desenvolvimento de softwares envolve muitas variáveis técnicas e do ambiente, como requisitos, recursos e tecnologia, que podem mudar durante o processo. Isto torna o processo de desenvolvimento imprevisível e complexo, requerendo flexibilidade para acompanhar as mudanças (SCHWABER, 1995).

O Scrum não exige ou fornece métodos ou práticas específicas de desenvolvimento de software, mas exige certas práticas de gerenciamento como mostra a Figura 05:

**Figura 5 -** Ciclo de desenvolvimento em Scrum




**Fonte:** Próprio Autor adaptado de http://consultingblogs.emc.com/colinbird/ (2010)

Conforme Soares (2004) descreve muito bem, o ciclo de vida do Scrum é baseado em três fases principais, divididas em subfases:

- Pré-planejamento: os requisitos são especificados em um documento chamado backlog, de forma a serem priorizados e a partir disto são realizadas estimativas de esforço para a implementação de cada requisito. O planejamento inclui também, a definição da equipe de desenvolvimento e as ferramentas a serem utilizadas para tal, os possíveis riscos do projeto e as necessidades de treinamento. Por fim é sugerida uma arquitetura de desenvolvimento. As alterações nos requisitos especificados no backlog são identificadas, assim como seus eventuais riscos;
- Desenvolvimento: as variáveis técnicas e do ambiente identificadas de antemão são analisadas e controladas no decorrer do desenvolvimento. Nesta etapa o software é desenvolvido em ciclos (sprints) em que novas funcionalidades são adicionadas. Cada um desses ciclos é implementado de forma que primeiramente faz-se a análise,

em seguida o projeto, implementação e testes. Cada um desses ciclos é esquematizado para durar de uma semana a um mês;

- Pós-planejamento: são realizadas reuniões para avaliar o avanço do projeto e demonstrar o software atualizado para os clientes. Nesta fase são feitas as etapas de integração, testes finais e documentação.

Segundo o seu autor Schwaber e Beedle (2002), O Scrum não é um processo previsível, ele não define o que fazer em toda circunstância. Por não fornecer métodos e práticas específicas de desenvolvimento, o Scrum se torna um método mais flexível, já que o desenvolvimento pode ser tratado da forma que for melhor para a organização.

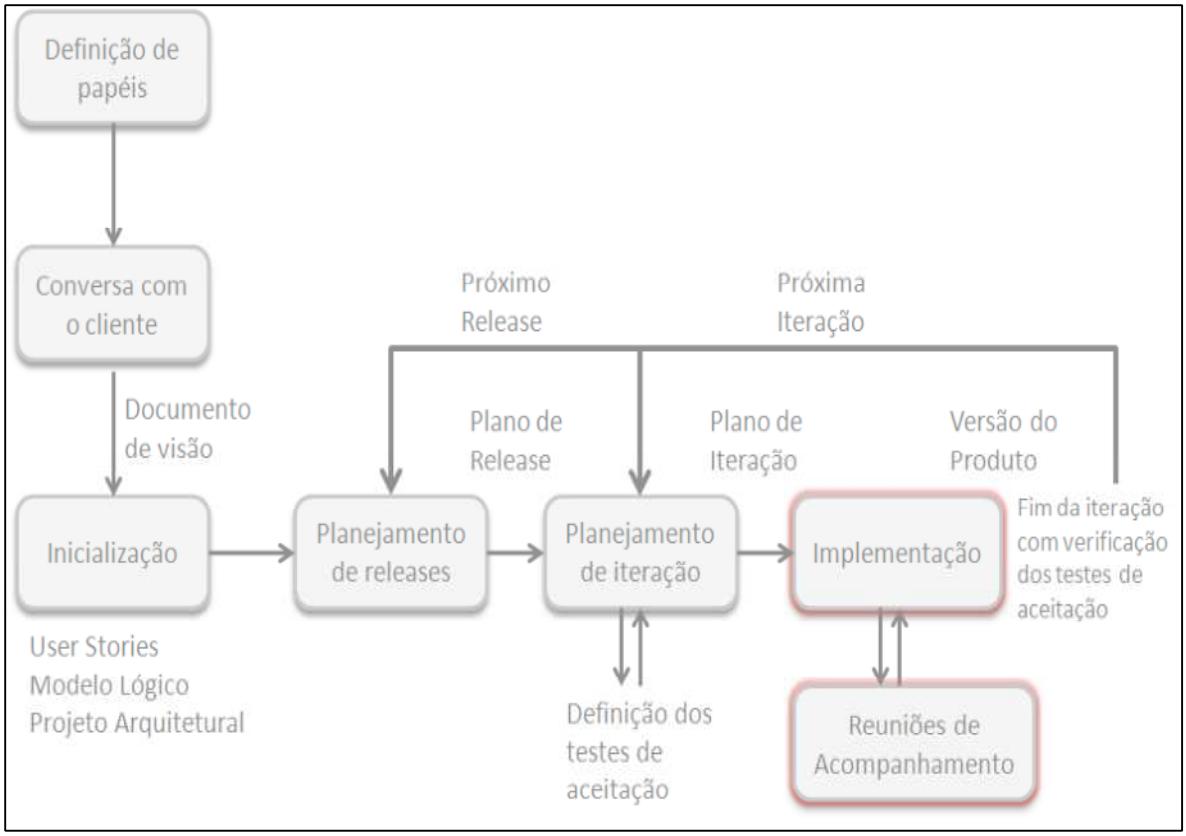
### 2.3.3 easYProcess

O easYProcess (YP) é um processo de software simplificado, desenvolvido através de um projeto da Universidade Federal de Campina Grande (PET/UFCG, 2007), amparado em práticas do XP (eXtreme Programming), RUP (Rational Unified Process) e AM (Agile Modeling), para possibilitar o desenvolvimento de software de forma ágil.

O YP foi determinado com base nas particularidades de um projeto para desenvolvimento de software acadêmico, mas mesmo sendo desenvolvido para projetos curtos e simples, é uma metodologia robusta e completa, com processo bem definido e bastante simples de aprender e implantar.

As fases do easYProcess são descritas por Garcia et al (2004) como: definição de papéis, conversa com o cliente, inicialização, planejamento, implementação, finalização da iteração e versão do produto, tais fases estão representadas na Figura 06.

**Figura 6** – Síntese do Fluxo do Processo de Desenvolvimento YP




**Fonte:** Próprio autor adaptado do PET/UFCG em (PET/UFCG, 2007).

As especificações de cada etapa deste processo estão sintetizadas a seguir (PET/UFCG, 2007):

O processo se inicia com a definição de papéis, onde o YP recomenda os seguintes papéis: cliente, usuário, testador, desenvolvedor e gerente. Depois deve ser realizada uma conversa com o cliente, onde conhecimentos sobre o escopo do problema são obtidos. A partir daí, a equipe produzirá o documento de visão, que deve ser validado pelo cliente, para que funcione como um roteiro para a equipe de desenvolvimento.

Em seguida, dá-se inicio a fase de Inicialização, momento no qual o cliente define as User Stories e são produzidos o projeto arquitetural e o modelo lógico de dados, este último apenas se necessário. A equipe deve realizar uma estimativa inicial do tempo para implementação de cada User Stories fornecida pelo cliente. Baseada nesta estimativa haverá a verificação da viabilidade de desenvolvimento do projeto no escopo e tempo definidos.

Dá-se inicio então ao planejamento, fase composta por dois planos, o de release e o de iteração. O planejamento de um release só ocorre após o término do anterior, e da mesma forma para as iterações. No planejamento de release alocamos as User Stories de acordo com

a priorização do cliente. No planejamento de iteração as User Stories alocadas são quebradas em tarefas, e o cliente deve definir os testes de aceitação para cada User Stories.

Para a Implementação, o processo prega o uso de algumas práticas, tais como: Design Simples, Padrões de Codificação, Padrões de Projeto, Refatoramento e Propriedade Coletiva de Código, a fim de produzir um código com mais qualidade. Há uma grande ênfase na parte de testes, tanto de unidade, que validam pequenos módulos do sistema, como de aceitação, que de fato representam a satisfação ou não do cliente diante do que foi desenvolvido.

O andamento do processo deve ser coordenado pelo gerente através da Reunião de Acompanhamento que visa recolher e analisar métricas. Nesta reunião faz-se uso do Big Chart, que deve ser gerado no mínimo uma vez por semana, da TA, para o acompanhamento das tarefas, e da Tabela de Riscos, onde deve ser realizado o acompanhamento dos riscos anteriormente levantados e dos novos riscos que surgirem.

Nas reuniões de acompanhamento, que coincidem com o fim de iteração, é necessária a presença do cliente pois este fará a verificação dos testes de aceitação, podendo dar uma User Stories por concluída ou não. Deve ser destacado que a cada plano de iteração ou release o ciclo referente ao planejamento e implementação se repete.

Pode-se perceber que essa metodologia possui algumas semelhanças com as outras metodologias apresentadas, como por exemplo, necessidade de produção de poucos artefatos, necessidade de reuniões de acompanhamento, dentre muitas outras. No entanto, como já foi mencionado é uma metodologia oriunda do meio acadêmico e ideal para ser aplicada com equipes ainda menores que o delimitado pelo XP e pela metodologia Scrum, se adaptando melhor, então, ao caso estudado. Por essas razões, foi escolhido como objeto desse estudo.

## 2.4 AGILIDADE NO DESENVOLVIMENTO DE SOFTWARE EDUCATIVO

No mercado brasileiro, graças à iniciativa do governo através de projetos de pesquisa e incentivo de iniciativa privada, alguns softwares educativos vêm sendo desenvolvidos (MORAES, 1998).

Conforme Koscianski e Soares (2007) desde a década de 70 o desenvolvimento de software em geral se confronta com uma série de problemas em sua construção e utilização, dentre eles:

- Cronogramas não respeitados;
- Projetos difíceis de serem seguidos, e por conta disto são abandonados;
- Versões que não realizam o combinado com o cliente;

- Aplicativos que não correspondem à expectativa;
- Programas difíceis de utilizar, por isso são descartados;
- Softwares que simplesmente deixam de funcionar.

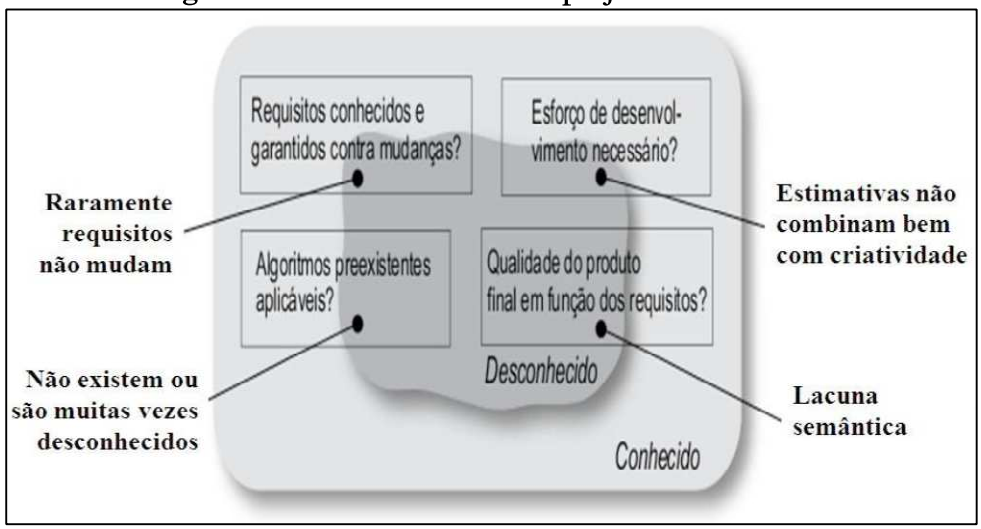
Trazendo para o contexto do software educativo, além dos problemas citados acima, também são encontrados problemas como padronização excessiva, falta de planejamento ao caráter educacional, desenvolvimento burocrático e baixa qualidade da documentação.

A fraqueza de muitos artefatos educacionais é que eles não se baseiam em uma análise adequada das necessidades do aluno (BOYLE et al, 2006). Por isso é importante que durante o projeto possam ser trazidas algumas questões à tona, tais como, quais são as dificuldades de aprendizagem que os alunos enfrentam? Como a utilização de Software Educativo pode ajudar aos alunos a sanar tais dificuldades? Os Softwares Educativos oferecem uma nova oportunidade de aprendizado?

O projeto de sistemas de softwares educativos deveria ser abordado de forma sistemática integrando aspectos de diversas dimensões de fatores humanos numa abordagem que incorpore aspectos teóricos e técnicos (TCHOUNIKINE, 2002), para que assim sejam dinâmicos e atendam as necessidades específicas quanto à interação com o usuário e realmente exerçam impacto sobre a aprendizagem.

Analisando a Figura 7 é possível detectar que as principais dificuldades no desenvolvimento de SE começam durante as etapas iniciais do projeto, principalmente porque uma mudança nas necessidades declaradas por um usuário pode repercutir em vários elementos da estrutura do programa (KOSCIANSKI e SOARES, 2007). Os autores nomeiam as principais dificuldades no desenvolvimento de SE como zona de sombra.

**Figura 7-** Zonas de sombra em projeto de software




**Fonte:** Próprio Autor adaptado de KOSCIANSKI e SOARES (2007)

Neste sentido, a metodologia utilizada para o desenvolvimento de SE deve auxiliar a equipe de desenvolvimento a encontrar a melhor forma de atingir o objetivo do software, respeitando as necessidades de aprendizagem e especificações do projeto. Para tanto, metodologias de desenvolvimento têm sido criadas, testadas e adaptadas há décadas (BOYLE et al, 2006).

Fuggetta (2000) concorda que processos precisam continuamente passar por mudanças e refinamentos para aumentar a sua habilidade de lidar com requisitos e expectativas da organização.

Para Boyle et al (2006) é necessário atualizar os métodos para o desenvolvimento ágil, que é estruturado e adaptável às circunstâncias do contexto educacional. O autor defende ainda que atingir essa flexibilidade é útil para se concentrar inicialmente nas funções-chave do desenvolvimento como análise das necessidades do aluno, desenvolvimento, entrega e avaliação.

Em contrapartida, fica clara a falta de diretrizes superiores para execução de um método de desenvolvimento de SE, sendo necessário que a equipe de projeto opte por uma metodologia simples, que possa ser rapidamente assimilada e aplicada, para que assim facilite e torne ágil a implantação e manutenção do processo. De modo que a equipe não necessite alterar suas práticas de desenvolvimento em função da implantação do método, e que este seja passível de adaptação, para que assim possam ser respeitadas as especificidades das etapas do processo de desenvolvimento de SE.

Tendo em vista os aspectos analisados, desenvolver um software com caráter efetivamente educativo de forma ágil e eficiente significa observar detalhes em cada uma das etapas de produção do mesmo, escolhendo formalismos que potencializem as características educacionais, e produzindo apenas artefatos realmente necessários durante o processo.

Por outro lado, constata-se que apesar da importância da utilização de metodologias de desenvolvimento amparada em formalismos e ferramentas CASE para o desenvolvimento ágil de software educativo, a principal mudança que deve ser realizada neste segmento é de caráter comportamental, tendo em vista que os pontos críticos deste problema envolvem essencialmente a maneira como a equipe de projeto utiliza as informações inerentes do contexto educativo.

Uma vez que se tenha conhecimento destas melhorias, é importante que sejam elencados elementos essenciais que interferem na concepção deste tipo de software, para que tais elementos possam ser inseridos em uma metodologia de desenvolvimento.

Portanto, seria necessário adaptar uma metodologia para utilização neste contexto, gerando um conjunto de recomendações que pudessem ser inseridas em uma metodologia que além de ágil, fosse utilizada no âmbito acadêmico para que desta forma tais recomendações pudessem ser incorporadas na formação de futuras equipes de projeto, de maneira que viessem a se tornar boas práticas de desenvolvimento de software educativo.

Diante disto, destaca-se o fato do processo de desenvolvimento YP ter sido planejado para academia e idealizado para o uso por equipes com nenhum conhecimento ou prática no uso de metodologias de desenvolvimento e processos. Neste sentido, ele foi desenvolvido para ser simples, mas completo o suficiente para que o aluno tenha um contato com uma metodologia séria de desenvolvimento, o que o torna passível a adaptações para diferentes contextos.

Em adição a tais aspectos, o YP ainda se encontra em consonância com as zonas de sombra em projeto de software, que são apontadas por Koscianski e Soares (2007) como as principais dificuldades no desenvolvimento de SE.

Por tudo isto, o processo de desenvolvimento YP será abordado com mais detalhes: seu processo, papéis, práticas e princípios, com base em (GARCIA et al., 2004) e (PET/UFCG, 2007), logo após, serão apresentadas algumas recomendações para que o YP possa ser melhor aplicado para o desenvolvimento de software educativos.

# 3 ADAPTAÇÃO DO EASYPROCESS PARA O DESENVOLVIMENTO DE SOFWARE EDUCATIVO

Este capítulo está dividido em três seções. Na primeira, o Processo de Desenvolvimento de Software easYProcess (YP) é apresentado de forma detalhada. Na segunda seção, alguns critérios foram elencados para que o YP fosse adaptado para o desenvolvimento de software educativo, ressaltando a importância de envolver professores e alunos durante todo o projeto. Na terceira e última parte deste capítulo, o YP adaptado ao cenário educacional (YPEduc) é apresentado, expondo cada etapa com as adaptações propostas.

## 3.1 EASYPROCESS

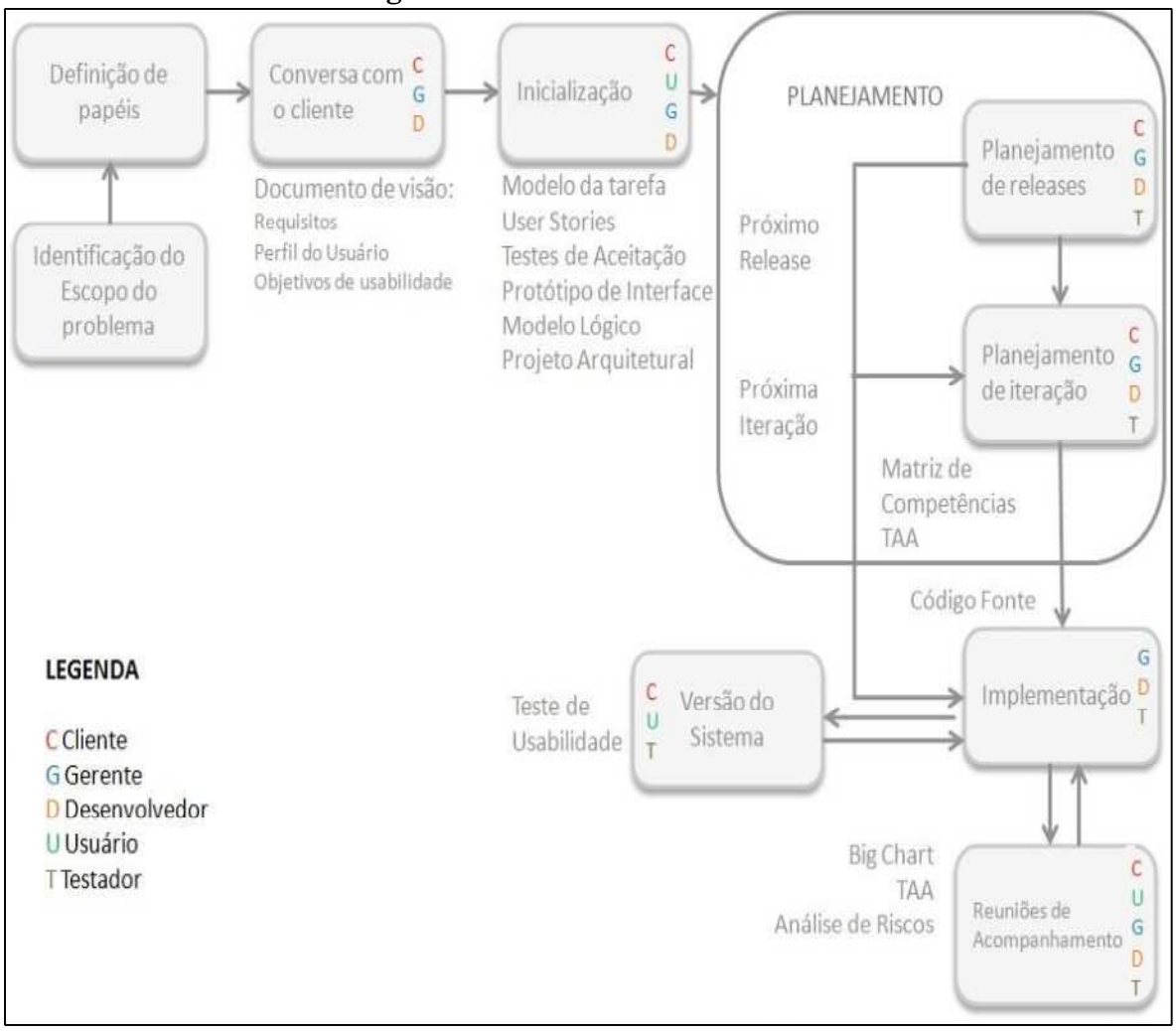
O processo de desenvolvimento YP é composto por oito fases bem definidas. Tais fases, em sua maioria, são pequenas e simples e é exatamente essa coesão que o faz um processo tão simples.

Outra característica marcante deste processo são as reuniões de acompanhamento, que proporciona grande flexibilidade e adaptabilidade ao processo, prevenindo com antecedência erros e diminuindo potenciais riscos. Outro ponto interessante do YP é a quantidade de artefatos, que além de relativamente pequena, possui um bom número de artefatos gerenciais, o que auxilia e incentiva o controle do processo.

Em relação à divisão de papéis, o YP facilita a formação de equipes pequenas de desenvolvimento. Cada papel possui responsabilidades diferentes associadas, e como visto anteriormente, um papel não é exclusivo de uma única pessoa, sendo que uma pessoa pode ter mais de um papel dentro da equipe. Alguns papéis possuem maior importância em determinadas etapas do que em outras.

Essa distribuição de importância e responsabilidades de cada papel em etapas do processo e os detalhes do fluxo deste processo são mostrados na Figura 8, e em seguida serão explicadas cada uma das fases com mais detalhes:

**Figura 8** - Fluxo do Processo easYProcess




**Fonte:** Próprio autor adaptado do PET/UFCG em (PET/UFCG, 2007).

### 3.1.1 As Fases do Processo

A Identificação do Escopo do problema é a fase inicial do processo, e é nela que se pesquisa e se adquire conhecimentos necessários sobre o problema para que se possa então preparar um roteiro de perguntas para a Conversa com o Cliente.

Na fase de Definição de Papéis é definido quem faz o quê dentro do processo. Esta fase é relativamente simples e por isso não costuma durar muito tempo.

A Conversa com o Cliente tem o objetivo de fazer com que os desenvolvedores e os clientes tenham um conhecimento em comum do sistema a ser desenvolvido. É nesta primeira reunião com o cliente que se captam informações como: requisitos funcionais e não funcionais, perfil do usuário, objetivos de usabilidade, testes de aceitação, entre outros. Após

esta etapa é então criado o Documento de Visão, primeiro artefato do processo, este documento concentra as ideias gerais sobre o que o sistema se propõe a fazer.

Este documento deve conter informações como o perfil do usuário final do sistema, uma análise inicial de riscos do projeto, requisitos funcionais e não funcionais, características gerais do produto, uma matriz inicial de competências dos envolvidos na equipe, objetivos de usabilidade, e mais outras informações que possam ajudar a equipe. Este documento pode ser usado também como um comparativo ao final do projeto, para saber o quão distante do planejado ficou o que foi concretizado.

Na fase de Inicialização é realizado um modelo da tarefa do sistema, para que então seja feito um protótipo da interface do sistema. Após a aceitação desta interface do sistema, são definidas, então, as User Stories com seus respectivos testes de aceitação. Enfim é desenvolvido um projeto arquitetural do sistema, bem como um possível modelo lógico, no caso de haver persistência de dados envolvida.

Essa etapa gera diversos artefatos e conta com a presença do cliente na sua maior parte, durando então o suficiente para que se tenha uma ideia das tarefas a serem desenvolvidas pelos desenvolvedores com bastante clareza, bem como um protótipo da interface e um modelo de dados bem definido.

O Planejamento, como pode ser vista na Figura 8, é dividido em duas fases: o planejamento de Release e o planejamento de Iteração, onde um release e uma iteração ocorrem em paralelo, sendo um release composto de "n" iterações. Estas fases do planejamento têm finalidades distintas:

- Planejamento de Releases: o planejamento de release consiste em definir o período e o escopo de um release, e então dividi-lo em iterações. Nesta fase são definidos quais User Stories serão implementados no release planejado;
- Planejamento de Iterações: no planejamento de iterações, as User Stories definidas no planejamento de release são divididas em atividades menores e alocadas para os membros envolvidos. Nesta etapa também é atualizada a Tabela de Alocação de Atividades (ou TAA).

A Matriz de Competências é uma tabela simples que mostra as competências de cada desenvolvedor do projeto. Estas competências devem ser relacionadas às tecnologias utilizadas no projeto. Esta matriz serve também como métrica do aprendizado e evolução dos desenvolvedores, pois se realizada ao início e ao fim do projeto, pode ser utilizada para a

comparação das competências de cada membro em dois tempos diferentes, constatando, então, quem aprendeu o quê.

A Tabela de Alocação de Atividades (TAA) representa a alocação de atividades para cada desenvolvedor. É ela quem sinaliza quem faz o quê durante uma iteração. Ela mostra uma descrição simples da atividade, quem é o responsável pela sua implementação, qual a estimativa de tempo, qual o status da tarefa, e, quando finalizada a tarefa, o tempo real despendido para a realização da mesma. A TAA mostra também quais os testes de aceitação associados à quais tarefas e o status de cada um destes testes.

Na etapa de Implementação é produzido o principal artefato de todo o projeto: o código-fonte do sistema. Nesta etapa são aplicadas as principais práticas usadas no desenvolvimento com metodologias ágeis.

O Código fonte é o objetivo de todo o processo, este deve ser sucinto e claro o suficiente para que sirva de comunicação entre os desenvolvedores. Como o YP incentiva a prática da Propriedade Coletiva de Código, todos os desenvolvedores terão acesso a qualquer parte dele, logo, é interessante que o código fonte seja de fácil entendimento.

Durante a Reunião de Acompanhamento ocorre o acompanhamento do projeto pelo Gerente com a participação de toda a equipe de desenvolvimento, e, quando coincidir com o final de uma iteração ou release, conta também com a presença dos clientes. Esta reunião serve também para atualizar o estado da Tabela de Alocação de Atividades e do Big Chart.

A TAA é um valioso objeto de análise do gerente durante a reunião de acompanhamento mensal, pois é a partir dela que é possível diagnosticar tarefas problemáticas e/ou desenvolvedores com problemas para cumprir suas tarefas.

O Big Chart é o artefato gerencial de todo o processo, pois se trata de um acumulador, em forma de gráfico, de métricas de todo o desenvolvimento do projeto. Ele marca a evolução das principais métricas que podem ser capturadas em cada reunião de acompanhamento.

A Análise de Riscos, desenvolvida pelo Gerente, é um documento que mostra os riscos associados a determinados aspectos do projeto, este documento é normalmente uma planilha simples, com a descrição dos riscos, prioridade, responsável, possível solução ou providência para o mesmo, e um estado (ou status) do problema, que pode assumir valores como: Resolvido, Superado, Vigente, Abortado, entre outros.

A etapa Versão do Sistema serve para a finalização e testes finais de uma eventual versão do sistema, que entrará em produção. Nesta etapa que são desenvolvidos eventuais manuais de utilização ou documentação, executados testes de integração, entre outras tarefas relacionadas, sendo também nesta etapa que são realizados os testes de usabilidade na versão

do sistema. O YP recomenda que os Testes de Unidade, de Aceitação e de Usabilidade sejam feitos.

O Teste de Unidade verifica 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). Os Testes 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. Os Testes de Usabilidade abrangem práticas de entrevistas, questionários e observações de uma situação de uso do sistema, em uma simulação com usuários reais que pertencem ao perfil levantado no documento de visão do projeto.

Quando comparada com outras metodologias, o YP possui um número relativamente pequeno de artefatos, priorizando mais a comunicação entre os membros da equipe e a interação com o cliente. O conjunto de artefatos do YP é basicamente composto de elementos de controle, para facilitar a gerência de todo o processo, e o acompanhamento preciso do estado do desenvolvimento. O uso de cada artefato está relacionado com as etapas do processo na Figura 8.

Com tudo isto, o YP parece inicialmente uma metodologia madura o suficiente para o uso em ambiente de produção, apesar de possuir apenas alguns anos de existência. Entretanto, como nunca foi usada em um ambiente de produção de software educativo, sua implantação em um ambiente diferente deve ser estudada e possíveis adaptações podem vir a ser feitas para adequá-la.

## 3.2 CRITÉRIOS DE ADAPTAÇÃO

Com a revisão da literatura ficou evidente que alguns pontos são de extrema relevância num processo de desenvolvimento de educativo, surgindo à necessidade de modificar alguns pontos do YP com o objetivo de contemplar características peculiares deste tipo de aplicativo computacional.

Algumas questões específicas foram levantadas como indispensáveis para que o software seja efetivamente educativo e por isso devem ser levadas em consideração no decorrer do projeto de desenvolvimento, dentre elas: o processo de ensino-aprendizagem deve ser um processo dinâmico e ativo; o próprio aluno precisa ser o construtor do seu conhecimento; o professor assume o papel de facilitador do processo de aquisição do conhecimento do aluno.

Neste sentido, outro fator em destaque após a revisão da literatura é a necessidade da mudança comportamental por parte da equipe de projeto, tendo em vista que os pontos críticos deste problema envolvem essencialmente a maneira como as informações inerentes ao contexto educativo são utilizadas pela equipe.

Para um melhor entendimento, os critérios sugeridos serão organizados em subseções, de forma que estes serão encaixados nas etapas do YP.

a) Identificação do Escopo

É fundamental ressaltar a importância da Identificação e delimitação do Escopo do Problema, pois ao elaborar um material educacional é necessário que se tenha em mente as características inerentes ao conteúdo que será abordado para que sejam identificados os seus objetivos, pois o software deverá ser um instrumento para atingi-los. Deve-se observar também qual o papel do software no processo de aprendizagem, como ele vai ser utilizado e qual o grau de importância que o SE terá em relação a este processo.

b) Definição de Papéis

Uma primeira mudança deve ser posta em prática na fase de Definição de Papéis, abrindo espaço para a presença de uma equipe multidisciplinar, onde professores e especialistas da educação possam participar da concepção do Software Educativo, tendo em vista que com o auxílio de profissionais da área de educação se torna muito mais simples definir pontos pedagógicos essenciais para esta problemática.

Ainda convém lembrar que o esforço na constituição de uma equipe multidisciplinar e multiprofissional com especialistas da Educação torna-se complexo, devido ao aumento de pessoas envolvidas no processo.

Por tudo isto, os componentes desta equipe devem antes de tudo desenvolver competências relacionais de interação e colaboração, que permitam o melhor desenvolvimento das competências individuais. Diante da necessidade de reunir pessoas de formações distintas para construir um material que deve atender aos objetivos do processo de ensino aprendizagem, a preocupação com as relações interpessoais, os processos de comunicação entre os saberes se tornam fundamentais, evitando assim os ruídos que podem interferir nas ações e objetivos traçados no planejamento.

O papel da equipe multidisciplinar será de elaborar e adaptar o conteúdo propriamente dito, textos, atividades, citações, gráficos, tabelas e demais elementos necessários para que o tema apresentado contemple os objetivos do SE, verificando ainda se este atende aos objetivos traçados pelo projeto pedagógico.

c) Conversa com o Cliente

Durante a Conversa com o Cliente, de onde resultará diversos artefatos importantes para o projeto de software, a equipe multidisciplinar deve participar de toda a parte pedagógica do processo, pois dada a complexidade dos fatores envolvidos no projeto de um software educativo, mesmo técnicas sistemáticas apresentam problema em sua execução.

Tais sistemas, conforme visto, possuem uma grande variedade de tipos de requisitos a serem identificados, relacionados à aprendizagem de conceitos e ao contexto de uso. A Análise de Requisito é um processo que engloba todas as atividades que contribuem para a produção de um documento de requisitos e sua manutenção ao longo do tempo.

Com a revisão de literatura ficou evidente que a fraqueza de muitos artefatos educacionais é que eles não se baseiam em uma análise adequada das necessidades do aluno em relação ao conteúdo que será passado.

Neste contexto, no desenvolvimento de um software educativo a Análise de Requisitos ganha ainda mais importância, pois é neste momento que são extraídas informações relacionadas às principais funcionalidades do sistema (requisitos funcionais) e aos requisitos não funcionais, portanto é na analise de requisitos que se determina o caráter educacional do software a ser desenvolvido e qual serão os métodos de aprendizagem do mesmo.

Assim, a Análise de Requisitos em softwares educativos deve respeitar questões como público-alvo, contexto, conteúdo, avaliação e equipe multidisciplinar a fim de aprimorar o desenvolvimento desses softwares para que os mesmos possuam o caráter educacional desejado e possam assim auxiliar o educador no processo de ensino-aprendizagem.

Para produzir o próximo artefato, Perfil do Usuário, geralmente é elencado um conjunto de informações relacionadas às características de potenciais usuários do sistema. Tais características devem revelar as habilidades, as limitações, as preferências, os interesses dos usuários, assim como o conhecimento prévio dos mesmos com relação a tarefa a ser realizada e ao uso de sistemas computacionais.

Para o Perfil do Usuário deste tipo de projeto, diferente do que acontece em projetos de softwares comerciais, além do perfil do aluno que utilizará o software deve ser levado em

consideração o perfil do professor mediador. Pois com o estudo da literatura ficou evidente que para revolucionar o processo de ensino-aprendizagem é necessário que haja a integração entre aluno- ferramenta- professor.

Outro fator que chama atenção neste artefato, é que apenas são levadas em considerações características físicas e de conhecimento prévio do usuário em relação ao manuseio do computador, contudo, para o projeto de SE deve ser considerado também o conhecimento prévio em relação ao conteúdo abordado pelo software.

Os Objetivos de Usabilidade são um conjunto de metas de usabilidade, mensuráveis, que devem ser alcançados pelo sistema. Geralmente se referem à eficácia, eficiência, segurança, aprendizado, memorização do sistema. No entanto, em softwares educativos, se faz necessário que ocorra também a valorização da aprendizagem do conteúdo que está sendo transmitido pelo sistema, neste sentido, durante a definição dos objetivos de usabilidade deve ser levada em conta além da aprendizagem do sistema, a aprendizagem do conteúdo educacional. Para tanto, é indispensável que professor e aluno também participem da produção deste artefato.

d) Inicialização

A fase de Inicialização tem como resultado O Modelo da Tarefa, as User Stories, os Testes de Aceitação, o Protótipo de Interface, o Modelo Lógico e o Projeto Arquitetural.

O Modelo da Tarefa é um artefato que envolve o entendimento dos desenvolvedores sobre a aplicação, e nada difere em projetos de softwares educativos para comerciais, por este motivo, tal artefato não sofrerá adaptação.

As User stories são especificações feitas pelo cliente a fim de definir o que o sistema deve ter/fazer. Este artefato deve ser gerado baseando-se nas necessidades do usuário, que são manifestadas por problemas e possibilidades dos usuários, e o contexto de uso, que inclui as características de futuros usuários, suas tarefas atuais e seu ambiente. O contexto de uso educacional necessita ser considerado e respeitado.

Para a Prototipação da Interface, o YP preconiza a construção de protótipos simples (esboço), que possam ser construídos com baixo investimento de tempo e recurso sem requerer qualquer habilidade técnica.

No entanto, em SE a prototipação merece uma atenção especial, pois a interface com o usuário é fundamental para o sucesso de um software de caráter educativo, neste sentido, é importante ressaltar que na fase de prototipação da interface deve ser levado em consideração

as múltiplas inteligências, o perfil de cada aluno, a fim de contemplar o Máximo de alunos possível com o aprendizado intuitivo.

O Modelo Lógico de Dados e Modelo Arquitetural são etapas que não diferem de acordo com o caráter do software, seja ele educativo ou comercial, por isto não haverá adaptações para estas etapas.

e) Planejamento

Durante a etapa de Planejamento, onde ocorre o Planejamento de Releases e o de Iteração, a equipe deve estar ciente do tempo disponível para o desenvolvimento do projeto, e a partir de então, com o cliente, definir o número de releases e iterações necessárias para a conclusão do mesmo. O YP propõe pequenos releases na tentativa de garantir uma maior interação entre a equipe de desenvolvimento e o cliente. Um benefício desta prática é a redução do impacto das mudanças de requisitos. O caráter do software não gera impacto nesta etapa, por isto não haverá adaptação para o contexto educativo.

Os artefatos que resultam desta fase é a matriz de competências e a TAA, mas estes artefatos também não sofrerão adaptações.

f) Implementação

A etapa de implementação de um SE também não difere de comercial, por isto nesta etapa também não acontecera adaptações.

g) Reuniões de Acompanhamento

As reuniões de acompanhamento são de fundamental importância em um SE e toda equipe deve participar, no entanto não há divergências quando o software concebido for educativo ou comercial, por isto não haverá adaptação para esta fase.

i) Versões do Sistema

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 easYProcess reconhece que é necessário testar o sistema sob diferentes aspectos, classificando os teste em diferentes categorias, cada um com um objeto especifico de verificação e recomenda que os Testes de Unidade, de Aceitação e de Usabilidade sejam feitos, não impedindo a realização dos demais, caso necessário. Sendo assim, considerando o caráter educacional do SE é evidente a necessidade de também se testar a aprendizagem do conteúdo abordado pelo software, através de um Teste de Aprendizagem.

Outro fator importante que pode ser mensurado através de testes seria a qualidade pedagógica, este teste seria realizado com professores e trataria de questões genéricas que avaliassem se os objetivos proposto pelo software foi alcançado, se a apresentação do conteúdo está clara e simples, se o nível de dificuldade está respeitando o perfil dos alunos e se o software é motivador e estimula o aprendizado.

O quadro 1 resume os critérios de adaptação do YP para o desenvolvimento de se, encaixando cada critério nas etapas do processo.

Quadro 1 - Critérios de Adaptação nas Etapas no YP

| ETAPAS | CRITÉRIOS DE ADAPTAÇÃO |
|---|---|
| **Definição de Papéis** | ✓ Constituição de uma equipe multidisciplinar e multiprofissional com especialistas da educação; |
| **Conversa com o cliente** | ✓ A equipe multidisciplinar deve participar da conversa com o cliente;<br>✓ A análise de requisitos deve respeitar questões como público-alvo, contexto, conteúdo e avaliação;<br>✓ Além do Perfil do Aluno deve ser levado em consideração o perfil do Professor Mediador;<br>✓ No Perfil do Usuário deve ser considerado também o conhecimento prévio em relação ao conteúdo abordado pelo software;<br>✓ Durante a definição dos Objetivos de Usabilidade deve ser levada em conta além a aprendizagem do conteúdo educacional;<br>✓ É indispensável que professor e aluno também participem da produção dos objetivos de usabilidade; |
| **Inicialização** | ✓ Considerar o contexto de uso educacional na concepção das User Stories;<br>✓ Na fase de prototipação da interface, deve ser levado em consideração as múltiplas inteligências e o perfil dos alunos;<br>✓ A equipe multidisciplinar deve participar da produção das User Stories e da Prototipação da interface; |
| **Planejamento** | ✓ Sem alterações; |
| **Implementação** | ✓ Sem alterações; |

**Quadro 1 -** Critérios de Adaptação nas Etapas no YP (Continuação)

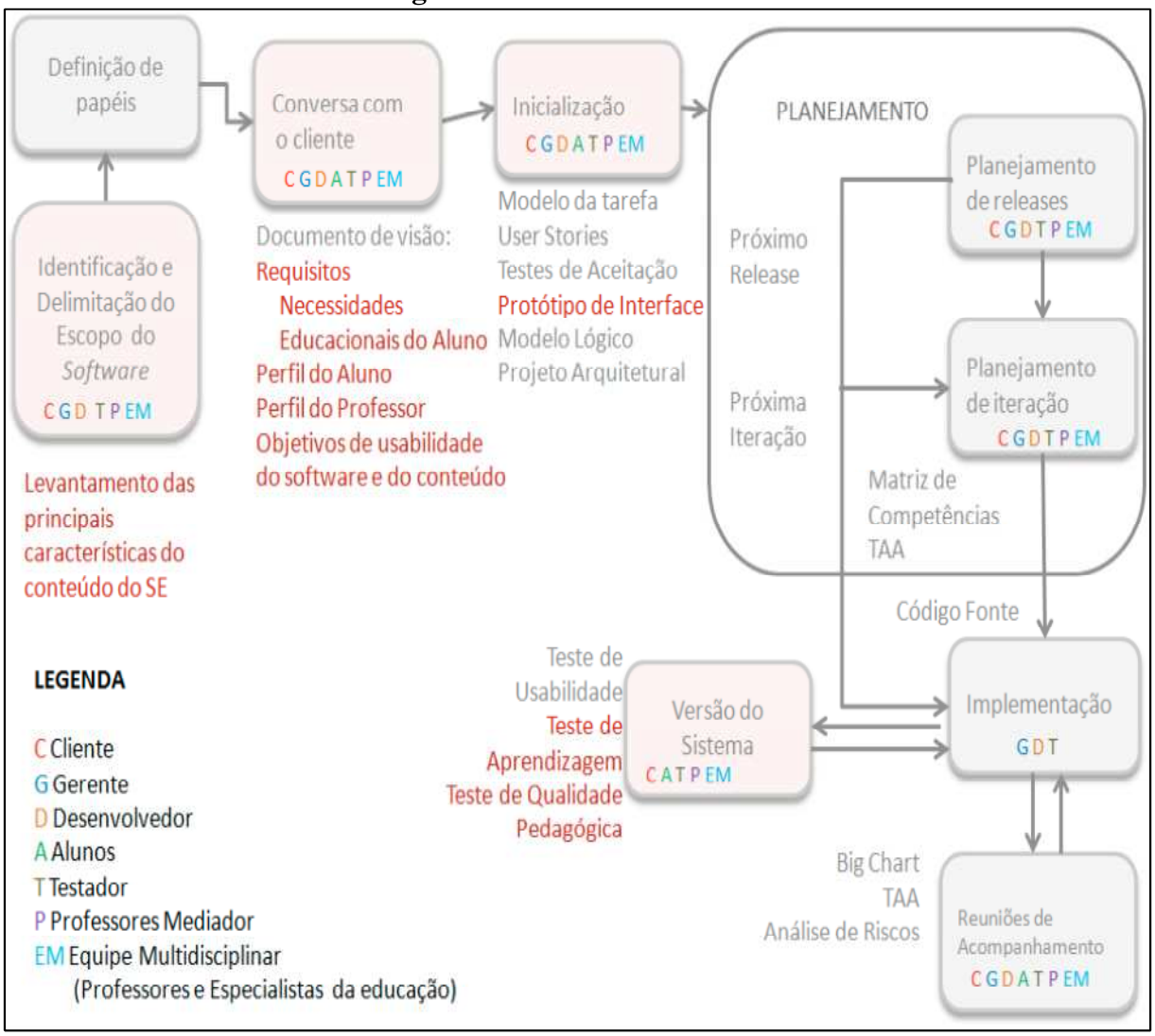
| | |
|---|---|
| **Reuniões de Acompanhamento** | ✓ Sem alterações; |
| **Versão do Sistema** | ✓ Realiar Teste de Aprendizagem e Teste de Qualidade Pedagógica. |

Fonte: Próprio Autor.

## 3.3 O YP ADAPTADO AO CENÁRIO EDUCACIONAL: YPEDUC

Nesta seção é apresentado o Fluxo de Trabalho do YPEduc, as etapas ou artefatos que sofreram alterações irão aparecer em destaque com letra vermelha. Em seguida, é explicada cada uma das fases com mais detalhes.

**Figura 9 -** Fluxo do Processo YPEduc




**Fonte:** Próprio Autor

### 3.3.1 Identificação do Escopo do Problema

Além da identificação do Escopo do software, são definidas as características inerentes ao conteúdo que será abordado para que sejam identificados os seus objetivos. Deve-se estabelecer também qual o papel do software educativo no processo de aprendizagem, como ele vai ser utilizado e qual o grau de importância que o mesmo tem em relação a este processo. Toda a equipe deve participar desta etapa.

### 3.3.2 Especificação 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 o processo. No YPEduc recomenda-se a presença de: cliente, usuário (aluno), gerente, desenvolvedor, testador, Professor (Mediador) e Equipe Multidisciplinar (Professor Conteúdista e Pedagogos ou Especialistas da Educação).

### 3.3.3 Conversa com o Cliente

A primeira conversa com o cliente consiste em uma coleta inicial de informações sobre o sistema que será desenvolvido. Devem participar desta etapa o cliente, gerente, Desenvolvedor, Testador, Aluno, Professor Mediador e Equipe Multidisciplinar.

Os artefatos produzidos durante esta etapa são:

- Visão sobre o Software Educativo: A partir dessa conversa o cliente e os desenvolvedores passam a ter um entendimento comum a respeito do sistema. O desenvolvedor deve tentar extrair do cliente a maior quantidade de informação possível sobre o sistema solicitado, ou seja, deve buscar por informações relacionadas às principais funcionalidades do sistema (requisitos funcionais), aos requisitos não funcionais mais importantes, ao perfil do usuário, aos objetivos de usabilidade, aos testes de aceitação, entre outros. A equipe multidisciplinar deve participar de toda a parte pedagógica do processo.
- Levantamento dos Requisitos Funcionais e Não Funcionais: A análise de requisitos em softwares educativos deve respeitar questões como público-alvo, contexto, conteúdo e avaliação.

- Levantamento do Perfil do Usuário/Aluno: Para o projeto de SE, no Perfil do Usuário deve ser considerado também o conhecimento prévio em relação ao conteúdo abordado pelo software e diferente do que acontece em projetos de softwares comerciais.
- Levantamento do Perfil do Professor Mediador: O perfil do professor mediador deve trazer informações sobre seu conhecimento sobre o conteúdo, sua experiência com tecnologias na educação, as principais dificuldades encontradas para lecionar o conteúdo abordado pelo SE, dentre outras.
- Levantamento dos Objetivos de Usabilidade: Durante a definição dos objetivos de usabilidade deve ser levada em conta além da aprendizagem do sistema a aprendizagem do conteúdo educacional. É indispensável que professor e aluno também participem da produção dos objetivos de usabilidade.

### 3.3.4 Inicialização

Logo após a equipe de desenvolvimento ter uma ideia geral sobre o problema a ser resolvido, devem ser iniciadas algumas atividades de análise do sistema. Devem participar desta etapa o cliente, gerente, Desenvolvedor, Testador, Aluno, Professor Mediador e Equipe Multidisciplinar.

O objetivo é fazer um Modelo da Tarefa que sirva de base para a construção do protótipo da interface que tem a função de para receber o feedback do cliente nas fases iniciais de implementação e listar todas as User Stories do sistema (e seus respectivos Testes de Aceitação) a fim de desenvolver uma arquitetura e um modelo lógico de dados capazes de acomodar mudanças e que sejam estáveis o suficiente para que riscos sejam minimizados.

Os artefatos produzidos durante esta etapa são:

- Modelagem da Tarefa: É uma representação, hierárquica, de como a tarefa é realizada pelo usuário. A construção do modelo da tarefa é precedida pela análise da tarefa, que consiste no estudo da tarefa a fim de melhorar a compreensão dos desenvolvedores com relação à natureza da tarefa, as partes que a compõe, e a ordem com que devem ser realizadas. Durante a análise da tarefa aspectos relacionados às características da tarefa, ao perfil do usuário e aos objetivos de usabilidade devem ser considerados.
- Levantamento das User Stories e Testes de Aceitação: São especificações feitas pelo cliente a fim de definir o que o sistema deve ter/fazer. O uso no contexto educacional

necessita ser considerado e respeitado para a produção das User Stories. Cada User Story deve estar associada no mínimo a um Teste de Aceitação.

- Geração do Protótipo da Interface: É uma representação gráfica, não necessariamente funcional, de um sistema que ainda não foi implementado. Em um SE, na fase de prototipação da interface deve ser levado em consideração às múltiplas inteligências e o perfil dos alunos, a fim de contemplar o máximo de alunos possível com o aprendizado intuitivo. Para a produção deste artefato, podem ser utilizados alguns métodos de design centrado no usuário e que incorporem questões relativas a modelos cognitivos do processamento humano. Alguns métodos de prototipação são explicados por Rocha e Baranauskas (2000), dentre eles estão o Design baseado em Guidelines, Design baseado em Cenários e o Design Participativo.
- Elaboração do Projeto Arquitetural: Este artefato tem como objetivo descrever o funcionamento do sistema num alto nível de abstração. Ele é útil quando se deseja explicitar como as partes do sistema a serem desenvolvidas interagem entre si ou com outros sistemas.
- Geração do Modelo Lógico de Dados: Se o sistema a ser desenvolvido envolve um banco de dados, o desenvolvedor deve construir o modelo lógico de dados capaz de representar a estrutura lógica do banco de dados.

### 3.3.5 Planejamento (Release e Iteração)

Concluídas as atividades de Inicialização, deve ser iniciado o Planejamento do projeto. Primeiramente, a equipe deve estar ciente do tempo disponível para o desenvolvimento do projeto, e a partir de então, com o cliente, definir o número de releases e iterações necessárias para a conclusão do mesmo.

Devem ser realizados pequenos releases na tentativa de garantir uma maior interação entre a equipe de desenvolvimento e o cliente, sendo assim, as User Stories alocadas em cada release só podem ser consideradas como finalizadas após a realização dos testes de aceitação.

O plano da iteração consiste em quebrar as User Stories em atividades menores (caso necessário) e em seguida alocar os membros da equipe de desenvolvimento como responsáveis pela realização de cada atividade. A alocação da tarefa ao desenvolvedor pode ser feita com o auxílio da Matriz de Competências, uma tabela simples que informa quais as habilidades dos membros da equipe de desenvolvimento e o tempo que cada um disponibiliza para o projeto.

Devem participar desta etapa o cliente, gerente, Desenvolvedor, Testador, Professor Mediador e Equipe Multidisciplinar.

### 3.3.6 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, Testes e Pequenos Releases. Devem participar desta etapa o gerente, Desenvolvedor e Testador.

### 3.3.7 Versão do Produto

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. Devem participar desta etapa o cliente, Testador, Aluno, Professor Mediador e Equipe Multidisciplinar.

Nesta fase, é necessário que além dos Testes de Usabilidade também ocorram os Testes de Aprendizagem e de Qualidade Pedagógica, criado pela equipe multidisciplinar para avaliar a aprendizagem dos alunos e o caráter pedagógico do software.

- 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. Os cenários de teste de usabilidade devem buscar avaliar o desempenho de usuários típicos na realização de tarefas comuns.
- Testes de Aprendizagem: Verifica se os alunos estão realmente aprendendo o conteúdo substancial do aplicativo, ou seja, é avaliado o que o aluno retém de conhecimento após a sua utilização. Diversos métodos de avaliação podem ser aplicados para o Teste de Aprendizagem, como provas, questionários, minitestes, dinâmicas, dentre outras.
- Testes de Qualidade Pedagógica do Software Educativo: Este teste verifica os aspectos pedagógicos (facilidade no acesso às informações, adequação a faixa etária, clareza nas informações, tipo de exercícios), do conteúdo (fidelidade ao objeto, coerência de apresentação do conteúdo, correção dos exercícios, organização dos conteúdos, promoção da criatividade e motivação dos usuários) e aspectos gerais

(alcança os objetos propostos, contribui para a aprendizagem dos conteúdos apresentados, preço compatível). Para o Teste de Qualidade Pedagógica existem alguns métodos estruturados indicados para a avaliação de qualidade de softwares educacionais, dentre eles a TICESE - Técnica de Inspeção Ergonômica de Software Educacional - (GAMEZ et al, 1999) e a Ficha de Avaliação de Software Educativo (Metodologia de Avaliação de Software Educativo Adaptado LECT/ USP).

# 4 ESTUDO DE CASO

Este capítulo apresenta os resultados obtidos frente à implantação do YPEduc em um miniprojeto de desenvolvimento de um software educativo, o Recicla. Neste capítulo ão avaliados os critérios propostos na seção anterior para cada etapa do YP, no entanto por se tratar de uma monografia, a autora assumirá praticamente todos os papéis do projeto e este estudo de caso terá um pequeno escopo com o intuito de verificar apenas se todos os pontos importantes foram acomodados no processo.

## 4.1 MINIPROJETO PARA O DESENVOLVIMENTO DO SOFTWARE EDUCATIVO: RECICLA

O primeiro artefato construído no projeto de desenvolvimento do software Recicla foi referente ao escopo do problema, onde foram levantadas as características iniciais do sistema, como mostra o quadro 2.

**Quadro 2 -** Escopo do Problema

| IDENTIFICAÇÃO DO ESCOPO DO PROBLEMA |
|---|
| O sistema a ser desenvolvido é de caráter educacional e visa a orientação de crianças na fase pré - escolar, com idade entre 04 e 06 anos. O objetivo principal do sistema é chamar atenção e ao mesmo tempo incentivar a coleta seletiva do lixo, fazendo com que através de um joguinho interativo e divertido as crianças entendam a importância da reciclagem para o meio ambiente. |

**Fonte:** Próprio Autor

Em seguida, foi realizada a definição de papéis com o intuito de dividir tarefas e responsabilidades do projeto, este artefato está sendo mostrado no quadro 3.

**Quadro 3** - Definição de Papéis

| PAPÉIS NO PROCESSO YPEDUC | |
|---|---|
| Pablo Ribeiro | Cliente |
| Angélica Felix | Gerente, Testador, Professor (Conteudista), Pedagogo e Usuário (aluno) e Professor Mediador. |
| Betoven Oliveira | Desenvolvedor |

**Fonte:** Próprio Autor

Feita a divisão de papéis, houve a primeira conversa com o cliente, para que a partir dela fosse criado o artefato que serviria de base para todo o projeto, a Descrição do Sistema como mostra o quadro 4.

**Quadro 4** – Documento de Visão

| **VISÃO SOBRE O JOGO RECICLA** |
|---|
| Com o advento dos tempos modernos e a revolução industrial o meio ambiente vem sofrendo mudanças em decorrência da agressão e do descaso da sociedade. Tendo em vista a não viabilidade de uma desindustrialização surge à demanda de uma pratica que proporcione o desenvolvimento sustentável, a reciclagem. As maiores vantagens da reciclagem são a minimização da utilização de fontes naturais, muitas vezes não renováveis; e a redução da quantidade de resíduos que necessita de tratamento final, como aterramento, ou incineração.<br>A coleta seletiva é o termo utilizado para o recolhimento dos materiais que são passíveis de serem reciclados, previamente separados na fonte geradora, essa separação evita a contaminação dos materiais reaproveitáveis, aumentando o valor agregado destes e diminuindo os custos de reciclagem, muitos materiais podem ser reciclados e os exemplos mais comuns são o papel, o vidro, o metal e o plástico.<br>Um dos fatores que inviabilizam esse processo é a falta de conscientização da população para essa separação adequada do lixo, por isso seria importante da inicio a essa conscientização nos primeiros anos de vida da criança onde seu desenvolvimento cognitivo influenciará na formação da sua personalidade. Daí surgiu à necessidade de um software educacional que despertasse essa responsabilidade ecológica nas crianças, já em fase pré – escolar entre 04 e 06 anos, de forma que este software as oriente para a forma adequada de tratar o lixo.<br>O software deve possuir dinâmica de jogo com o intuito de envolver o usuário, deve possuir uma aparência agradável com diversidade de cores, som e imagens, além de três níveis de dificuldade e uma forma de orientação visual que familiarize a criança com o computador e respeite a sua fase cognitiva, não sendo estritamente necessária a leitura, deve haver também duas formas de interação no jogo através do teclado e do mouse para que a criança possa desenvolver as duas habilidades ou escolher a forma de sua preferência.<br>O jogo consiste na queda de objetos recicláveis e o usuário deve mover o conjunto de lixeiras para que o objeto em queda adentre o recipiente correto, reproduzindo sons para acertos e erros. A cor de fundo deve ser branca para facilitar a visualização dos objetos e as lixeiras devem possuir as suas cores oficiais. No jogo deve haver um botão sair que finalize o jogo de imediato.<br>Cada objeto depositado em sua devida lixeira acumula dez pontos e ao contabilizar 130, 300 e 500 pontos subirá para o segundo nível, terceiro nível e finalizará o jogo respectivamente. Esse avanço de nível influencia na velocidade da queda dos objetos e ao passar de nível aparecerá uma mensagem lhe dando os parabéns e mostrando objetos que foram confeccionados a partir da reciclagem, a fim de que a criança tenha uma real noção do que pode ser feito a partir da reutilização do lixo.<br>Neste caso temos dois componentes um de interação que é representado pelo movimento das lixeiras, pela queda dos objetos, pelos sons, pelas mensagens de parabéns e pelo avanço de nível, o outro componente é de informação representada pelo guia de ajuda, nível de dificuldade e pela contabilidade dos pontos.<br>O papel do software no processo de aprendizagem será o de ensinar as crianças como utilizar as lixeiras de coleta seletiva, as ajudando a memorizar as cores e os tipos de materiais que devem ser colocados em cada lixeira. A interface deve ser de simples utilização para que as crianças possam usufruir do aplicativo dentro e fora de sala de aula.<br>As tecnologias estão por toda parte e é necessário ressaltar ainda o quanto é importante utilizar esta tecnologia também no processo de ensino aprendizagem como uma forma de aproximar as crianças da escola, tendo em vista que elas facilitam os processos e fascinam pelas suas mil e uma possibilidades. |

**Fonte:** Próprio Autor

A definição dos requisitos funcionais e não funcionais do sistema segue no quadro 5

**Quadro 5 -** Definição dos Requisitos

| **REQUISITOS FUNCIONAIS** | |
|---|---|
| Componente de interação | Queda de objetos;<br>Movimento das lixeiras;<br>Sons;<br>Avanço de nível;<br>Mensagens de incentivo; |
| Componente de informação | Display de pontuação;<br>Guia de usabilidade;<br>Display de nível de dificuldade; |
| **REQUISITOS NÃO FUNCIONAIS** | |
| **REQUISITOS** | **DESCRIÇÃO** |
| Facilidade de uso | Poucos botões de acesso;<br>Interação e cores intuitivas; |
| Desempenho | Boa velocidade;<br>Baixo consumo de memória; |
| Portabilidade | Todos os sistemas com a JVM instalada; |
| Entrega | Dois meses; |
| Implementação | Linguagem Java;<br>Método iterativo e incremental; |
| Padrões | Não se aplica ao sistema; |
| Espaço | 173 Kb; |
| Confiabilidade | Pouca propensão a erros; |
| Interoperabilidade | Não se aplica ao sistema; |
| Ético | Software de interesse público; |
| Privacidade | Respeito aos direitos autorais; |
| Segurança | Não se aplica ao sistema; |
| Público-alvo | Crianças na fase pré - escolar, com idade entre 04 e 06 anos; |
| Contexto | O software poderá ser utilizado em sala de aula ou em casa; |
| Conteúdo | Reciclagem; |
| Avaliação | Através do comparativo das pontuações; |

**Fonte:** Próprio Autor

O perfil do aluno e do professor-mediador são também artefatos resultantes da primeira conversa com o cliente. Os formulários para levantamento do Perfil do Aluno e do Professor Mediador seguem nos quadros 6 e 7, respectivamente.

Para coletar dados para o perfil do aluno e do professor-mediador, foi necessário que o professor-mediador respondesse um questionário adaptado do Laboratório de Interfaces Homem-Máquina, UFCG que normalmente é utilizado para levantamento do perfil do usuário.

**Quadro 6 -** Perfil do Aluno

| **FORMULÁRIO PARA O LEVANTAMENTO DO PERFIL DO ALUNO** |
|---|
| **Características do usuário, escolhidas pelo projetista, de acordo com a relevância, para o projeto.**<br>**Levantamento baseado em:**<br>Fatos( ) Opinião do usuário( ) Dados medidos ou observados( )<br>Opinião do Professor( X ) |
| **Parte I – Características Gerais** |
| **Faixa etária:** 04 a 06 anos.<br>**Sexo:** Feminino/ Masculino.<br>**Habilidades específicas necessárias para executar a tarefa:**<br>**Níveis de percepção:** Visão.<br>**Habilidades motoras:** Um mínimo possível de coordenação motora.<br>**Grau de instrução (ex. técnico, superior):** Fase pré – escolar.<br>**Função desempenhada:** Aprendiz / aluno.<br>**Objetivos do aplicativo:** Despertar a responsabilidade ecológica nas crianças.<br>**Motivações do aluno:** Aprender sobre coleta seletiva e reciclagem.<br>**Preferências (ex. uso do teclado, preenchimento do formulário):** Teclado e/ou mouse. |
| **Parte II - CONHECIMENTO CONCEITUAL necessário à execução das tarefas** |
| CONHECIMENTO SEMÂNTICO<br>Nível de Experiência (noviço, experiente, especialista) |
| **Função:** Noviço **Método:** Noviço<br>**Tarefa:** Noviço **Computadores:** Noviço<br>**Ferramentas utilizadas na execução das tarefas (ou similares):** Noviço |
| CONHECIMENTO SINTÁTICO<br>Nível de experiência |
| **Uso de teclado e mouse:** Noviço **Uso de dispositivos especiais de interação:** Noviço<br>**Uso de terminologia específica:** Noviço |
| **PARTE III – ESTILO COGNITIVO** |
| **Aprendizado (treinamentos):** Treinamento informal em sala de aula.<br>**Capacidade de solucionar problemas (sozinho, com ajuda):** Sozinho.<br>**Capacidade de reter o aprendizado (alta, média, baixa):** Alta. |
| PERSONALIDADE |
| **Nível de curiosidade (baixo, médio, elevado):** Elevado.<br>**Nível de persistência (baixo, médio, elevado):** Baixo.<br>**Nível de inovação (baixo, médio, elevado):** Médio.<br>**Inovador** ( ) **Conservador**( ) **Impulsivo**( X ) **Reflexivo**( ) |
| CONHECIMENTO PRÉVIO |
| **Quanto às letras, o aluno:**<br>Não reconhece nenhuma( ) Reconhece algumas(X) Reconhece Todas( )<br>**Quanto aos números, o aluno:**<br>Não reconhece nenhum( ) Reconhece alguns( ) Reconhece Todos( X )<br>**Quanto ao grau de leitura do aluno:**<br>( X )Não sabe/Está aprendendo ( ) Intermediário ( )Alfabetizado ( )Leitor Crítico<br>**O aluno terá aulas sobre o tema abordado antes de utilizar o software:** Sim(X) Não( )<br>**O aluno já apresentou algum tipo de curiosidade/interesse sobre o tema:** Sim(X) Não( )<br>**Qual o nível de conhecimento que o aluno possui sobre o tema:**<br>Baixo( X ) Médio( ) Alto( ) Especialista( )<br>**O aluno será avaliado após a utilização do software:** Sim(X) Não( ) |

**Fonte:** Formulário adaptado do © 2001, Laboratório de Interfaces Homem-Máquina, UFCG

**Fonte:** Próprio Autor

**Quadro 7 -** Perfil do Professor Mediador

**FORMULÁRIO PARA O LEVANTAMENTO DO PERFIL DO PROFESSOR**

**Características do professor, escolhidas pelo projetista, de acordo com a relevância, para o projeto.**
**Levantamento baseado em:**
Fatos( ) Opinião do usuário( ) Dados medidos ou observados( )
Opinião do Professor( X )

**Parte I – Características Gerais**

**Qual a sua formação Acadêmica:**
**Quantos anos de Experiência como docente:** Feminino/ Masculino.
**Fez algum curso de informática: Sim**( X ) **Não**( )
**Qual o seu grau de instrução tecnológica:** Baixo( ) Médio( X ) Alto( )
**Quanto costuma utilizar o computador:**
Raramente( ) Semanalmente( ) Diariamente( X ) Várias vezes por dia( )
**Objetivos do aplicativo:** Despertar a responsabilidade ecológica nas crianças.
**Motivações do aluno:** Conscientizar as crianças para a importância da coleta seletiva.
**Preferências (ex. uso do teclado, preenchimento do formulário):** Teclado e/ou mouse.

**Parte II - CONHECIMENTO CONCEITUAL necessário à execução das tarefas**

CONHECIMENTO SEMÂNTICO
Nível de Experiência (noviço, experiente, especialista)

**Função:** Experiente **Método:** Experiente
**Tarefa:** Experiente **Computadores:** Experiente
**Ferramentas utilizadas na execução das tarefas (ou similares):** Experiente

CONHECIMENTO SINTÁTICO
Nível de experiência

**Uso de teclado e mouse:** Experiente **Uso de dispositivos especiais de interação:** Experiente
**Uso de terminologia específica:** Experiente

**PARTE III – ESTILO COGNITIVO**

PERSONALIDADE

**Nível de curiosidade (baixo, médio, elevado):** Elevado.
**Nível de persistência (baixo, médio, elevado):** Elevado.
**Nível de inovação (baixo, médio, elevado):** Médio.
**Inovador** ( ) **Conservador**( ) **Impulsivo**( ) **Reflexivo**( **X** )

CONTEÚDO

**Qual o seu nível de conhecimento sobre o tema:**
Baixo( ) Médio( ) Alto( X ) Especialista( )
**Já foi ministrada alguma aula sobre este tema:** Sim ( X ) Não( )
**Em geral, os alunos apresentam alguma dificuldade em aprender este conteúdo:**
Não( ) Sim(X) Qual: __A memorização das cores das lixeiras para cada tipo de material reciclável._____
**Qual tipo de Avaliação você realizará após a utilização do software:**
Prova( ) Perguntas Orais( ) Conversa informal( X ) Seminários( )
Questionários( ) Nenhuma( ) Outras( )_____

**Fonte (Formulário):** Formulário adaptado do © 2001, Laboratório de Interfaces Homem-Máquina, UFCG
**Fonte (informações):** Próprio Autor

Outro artefato resultante desta etapa do projeto são os objetivos de usabilidade do jogo Recicla. Os Objetivos de Usabilidade e a mensuração destes estão apresentados no quadro 8.

**Quadro 8 -** Objetivos de Usabilidade

| OBJETIVOS DE USABILIDADE DO JOGO RECILA | |
|---|---|
| **OBJETIVOS** | **MENSURAÇÃO** |
| Reduzir a taxa de erros | Propensão a erros reduzida pela interface. |
| Facilitar o aprendizado | Cores e imagens intuitivas. |
| Adequar o conteúdo | Respeito com o nível de conhecimento do usuário. |
| Aumentar a satisfação subjetiva do usuário | Consultar usuário após 1ª impressão com o sistema a fim de otimizá-lo em novas versões. |
| Manter a clareza na estrutura | Guia de usabilidade de fácil entendimento e presente no próprio jogo. |
| Ser atrativo ao usuário | Consultar usuário com relação a interface gráfica. |
| Possuir telas simples | Interface intuitiva e de fácil interação. |
| Manter consistência nas sequencias de ações necessárias para realizar uma tarefa | Simples acesso e manipulação. |
| Disponibilizar boa documentação | Documentação simples incorporada no próprio software. |
| Reduzir a dificuldade de aprendizagem dos alunos com relação ao conteúdo abordado | Exploração do conteúdo de forma simples e objetiva. |
| Sanar as dúvidas que surgirem quanto ao conteúdo do software | Na ajuda deve conter uma explicação sucinta sobre o conteúdo. |

Fonte: Próprio Autor

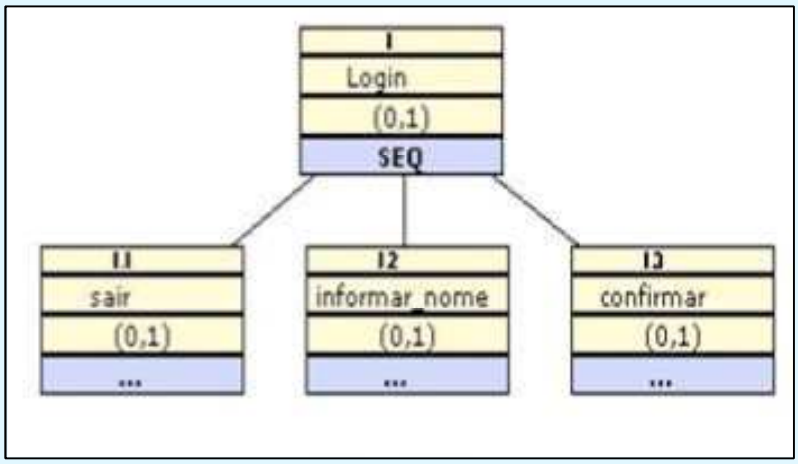
A próxima fase do projeto é a Inicialização, é importante que neste momento a equipe de projeto já tenha uma ideia geral do sistema que esta sendo desenvolvido, pois é agora que começam as atividades de análise do sistema. Os artefatos que foram produzidos nesta fase

foram: Modelagem da tarefa, User Stories e Testes de Aceitação, Protótipo de Interface, Projeto Arquitetural e Modelo Lógico de Dados.

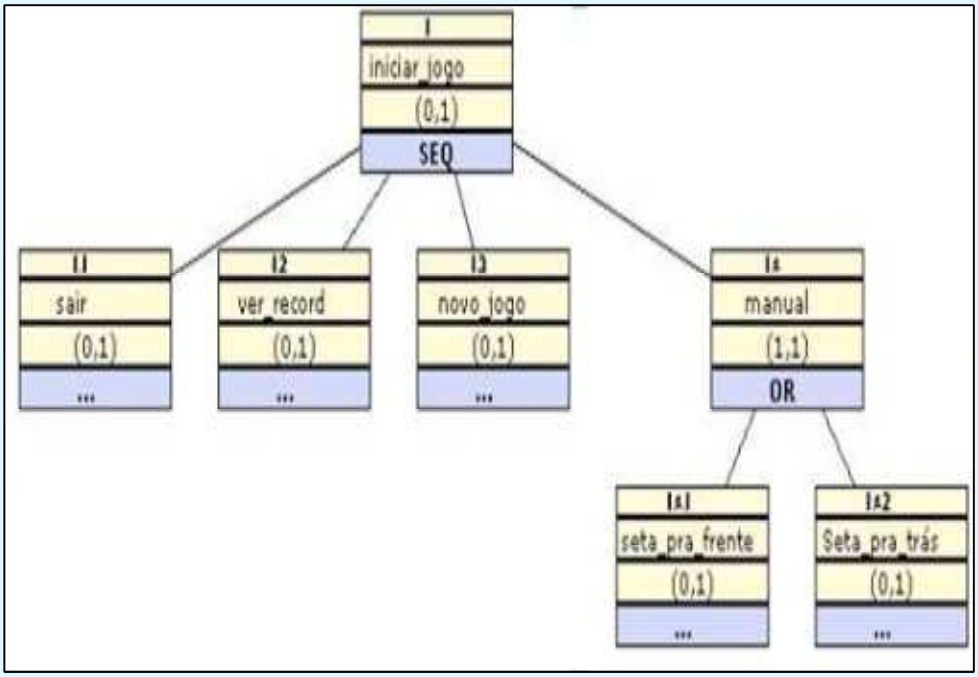
A modelagem da tarefa foi realizada pelo desenvolvedor através da Ferramenta para Modelagem de Tarefas ITAOS (MEDEIROS et al, 2003) e esta exposta no quadro 9.

**Quadro 9** - Modelagem da Tarefa

MODELO DE TAREFA: ABRIR APLICATIVO




MODELO DE TAREFA: INICIAR JOGO




**Fonte:** Próprio Autor

Nesta etapa também são definidas as User Stories e os Testes de Aceitação, estes artefatos foram produzidos pelo cliente, desenvolvedor e testador, mas o professor mediador e o aluno também colaboraram. Tal artefato segue no quadro 10.

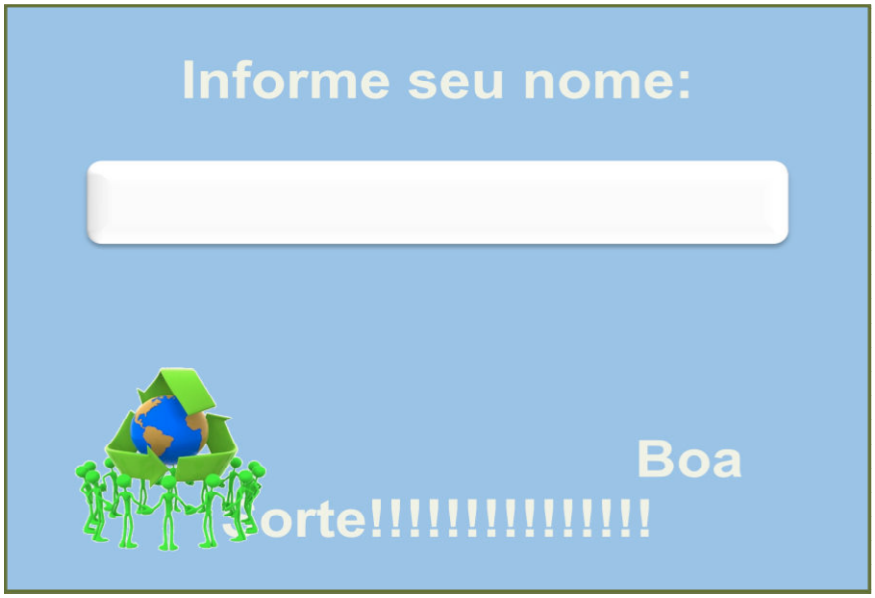
**Quadro 10 -** User Stories e Testes de Aceitação

| DEFINIÇÃO DAS USER STORIES E SEUS RESPECTIVOS TESTES DE ACEITAÇÃO | |
|---|---|
| **US01** | **Implementar a queda dos recicláveis na tela.** |
| TA1.1 - Verificar se os recicláveis se alternam após a queda.<br>TA2.2 – Verificar se os recicláveis caem em lugares alternados. | |
| | **Estimativa inicial:** 8h |
| **US02** | **Implementar um conjunto de lixeiras que deve ser manipulado a fim de coletar os recicláveis que descem na tela.** |
| TA2.1 - Averiguar se os limites de deslocamento das lixeiras correspondem ao esperado.<br>TA2.3 – Averiguar se as lixeiras são corretamente movimentadas a partir do teclado e do mouse. | |
| | **Estimativa inicial:** 10h |
| **US03** | **Implementar displays de pontuação e fase.** |
| TA3.1 – Averiguar se os valores do contador de pontos estão sendo "setados" no display de pontuação.<br>TA3.2 – Averiguar se o display de fase é modificado a partir dos valores especificados.<br>TA3.3 – Averiguar se o contador de pontos funciona se e somente se os recicláveis caem na lixeira correta.<br>TA3.4 – Averiguar se as telas de parabéns e incentivo são abertas após as fases serem conquistadas. | |
| | **Estimativa inicial:** 8h |
| **US04** | **Implementar guia simples de orientação ao uso.** |
| TA4.1 – Averiguar se o guia é ativado a partir de um simples clique e se satisfaz o cliente. | |
| | **Estimativa inicial:** 3h |
| **US05** | **Implementar fechamento alternativo do jogo.** |
| TA5.1 – Averiguar se o jogo é fechado com as mesmas características do fechamento default e se satisfaz o cliente. | |
| | **Estimativa inicial:** 1h |
| **US06** | **Implementar visão de recordes.** |
| TA6.1 – Testar entradas e visões no SGBD antes de ligá-lo ao jogo.<br>TA6.2 – Testar entradas e visões (diversas vezes) através do jogo. | |
| | **Estimativa inicial:** 6h |
| **US07** | **Implementar os sons do jogo.** |
| TA7.1 – Averiguar a saída de som para os cliques.<br>TA7.2 – Averiguar os sons após a passagem de fase e finalização(zeragem) do jogo. | |
| | **Estimativa inicial:** 8h |

Fonte: Próprio Autor

O Protótipo de Interface é um artefato importantíssimo para a usabilidade do software, pois a interface com o usuário é um requisito fundamental para o sucesso do software. A produção dos protótipos exigiu a colaboração do cliente, aluno, equipe multidisciplinar, professor mediador e desenvolvedor. Este artefato foi produzido através dos métodos do Design Participativo ilustrado em Rocha e Baranauskas (2000). Estes artefatos estão ilustrados nos Figuras 10, 11, 12, 13 e 14.

**Figura 10** - Tela Inicial




**Fonte:** Próprio Autor

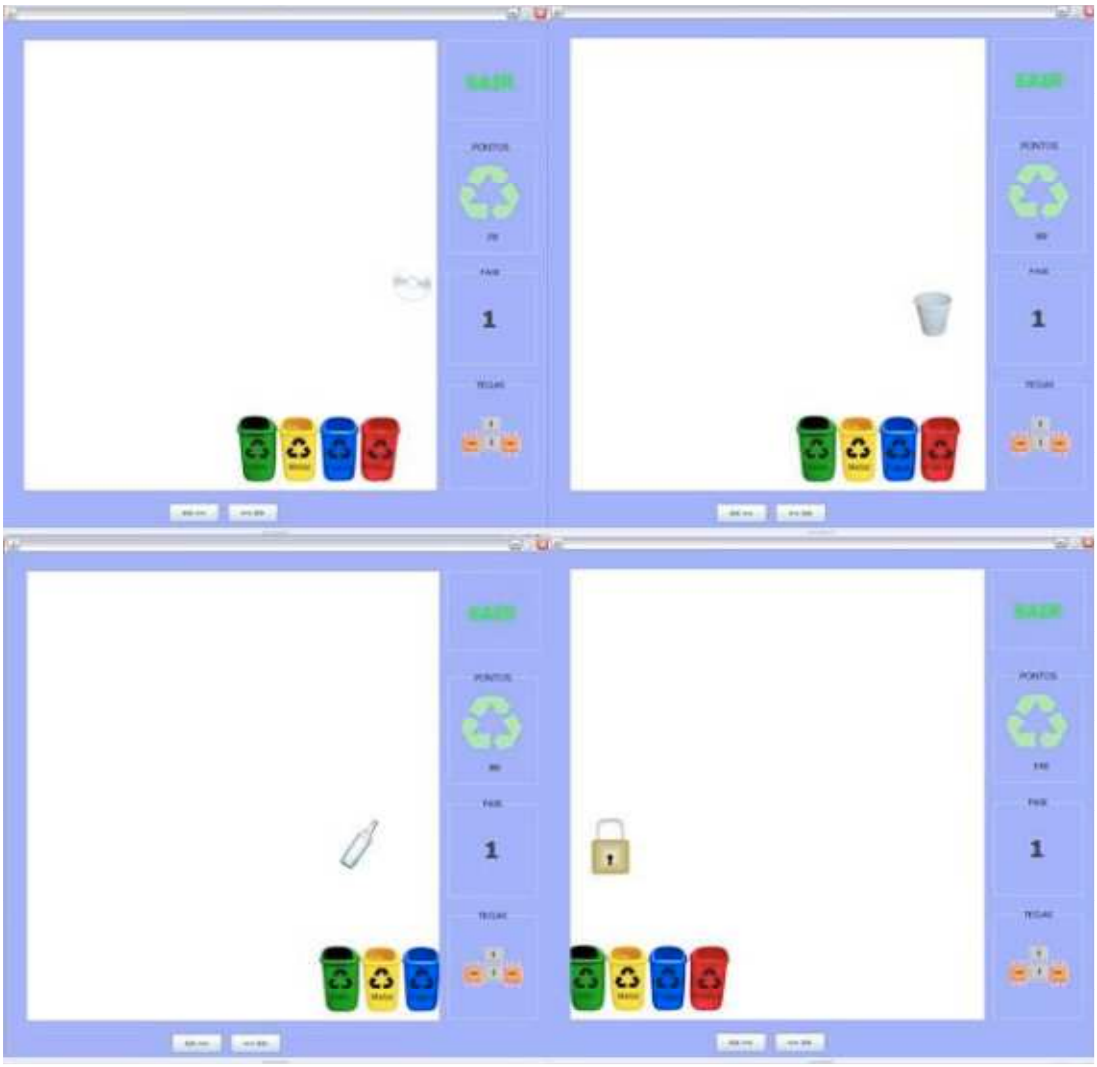
**Figura 11** - Mudança de nível 01


**Fonte:** Próprio Autor

**Figura 12 -** Mudança de nível 02


**Fonte:** Próprio Autor

**Figura 13** – Telas do Jogo



**Fonte:** Próprio Autor

**Figura 14 -** Tela Final


**Fonte:** Próprio Autor

O Projeto Arquitetural objetiva descrever o funcionamento do Recicla num alto nível de abstração, este artefato foi produzido pelo desenvolvedor, a partir de conversas com o cliente e os artefatos gerados anteriormente. O projeto arquitetural do Recicla esáa sendo mostrado no quadro 11 e foi realizado com base no questionário do professor Jacques Sauvé do Departamento Sistemas e Computação da UFCG - Perguntas a Fazer ao Elaborar um Projeto Arquitetural.

**Quadro 11 -** Projeto Arquitetural

| **SOBRE ENTIDADES EXTERNAS AO SISTEMA** |
|---|
| **Quais sistemas externos devem ser acessados?** O SGBD MySql para a persistência dos recordes; a persistência de imagens e sons será em arquivo dentro do próprio *.jar<br>**Como serão acessados?** A persistência no SGBD será auxiliada pelo JDBC<br>**Há integração com o legado a ser feita? Como?** Não |
| **DETERMINAÇÃO DE OPORTUNIDADES PARA O REUSO DE SOFTWARE** |
| **Há interesse/conveniência/tempo em aproveitar tais oportunidades?** SIM!<br>**Como isso pode ser feito?** Para a GUI usaremos o pacote javax.swing; usaremos também o JDBC para criar a comunicação entre SGBD e nosso sistema.<br>**Construindo um framework?** Não. |

**Quadro 11 -** Projeto Arquitetural (Continuação)

**SOBRE A ORGANIZAÇÃO GERAL DO SISTEMA**

**O sistema é centralizado ou distribuído?** Centralizado.
**Como modularizar em subsistemas?** Será utilizado o paradigma OO para facilitar
**Como minimizar acoplamento entre os pedaços?** Será criada uma classe para comunicar a camada gráfica com a camada de negócio e o JDBC do Java para comunicar a camada de negócio com a camada de persistência de dados. Lembrando que um sistema pode ser visto como sendo composto de três grandes camadas lógicas.
**Tais camadas serão lógicas ou terão existência física separada?**
**Camada Gráfica**: corresponde à interface com o usuário; são as telas do software que permitiram uma boa usabilidade por parte do usuário do jogo.
**Camada de Negócio**: é nela que se encontra o algoritmo do software. É também essa camada que permitirá uma comunicação com a interface (através de uma classe que servirá de ponte) e os dados.
**Camada de Persistência de Dados**: haverá persistência em dois tipos de dados. As informações de recordes serão armazenadas através de SGBD (a saber – MySQL) e as imagens serão armazenas em diretório contido dentro do próprio arquivo *.jar. A persistência dos dados caberá a essa camada.
**Qual é o nível de acoplamento, frequência de interações, volumes de dados trocados entre as camadas?** Baixo/ razoável/ poucos
**A programação será feita com qual paradigma? OO?** Sim!
**Que linguagens e ferramentas serão usadas?** A linguagem será Java, na plataforma JSE (Java Standard Edition); Será utilizado a ferramenta NetBeans como IDE do projeto.
**Como será feito a alocação de memória para os elementos do programa?** Estaticamente, com exceção dos dados persistidos através do SGBD.
**Onde são armazenados os strings?** Alguns em classe auxiliar, outros em banco de dados.
**Há considerações especiais de segurança que afetam o sistema?** Não

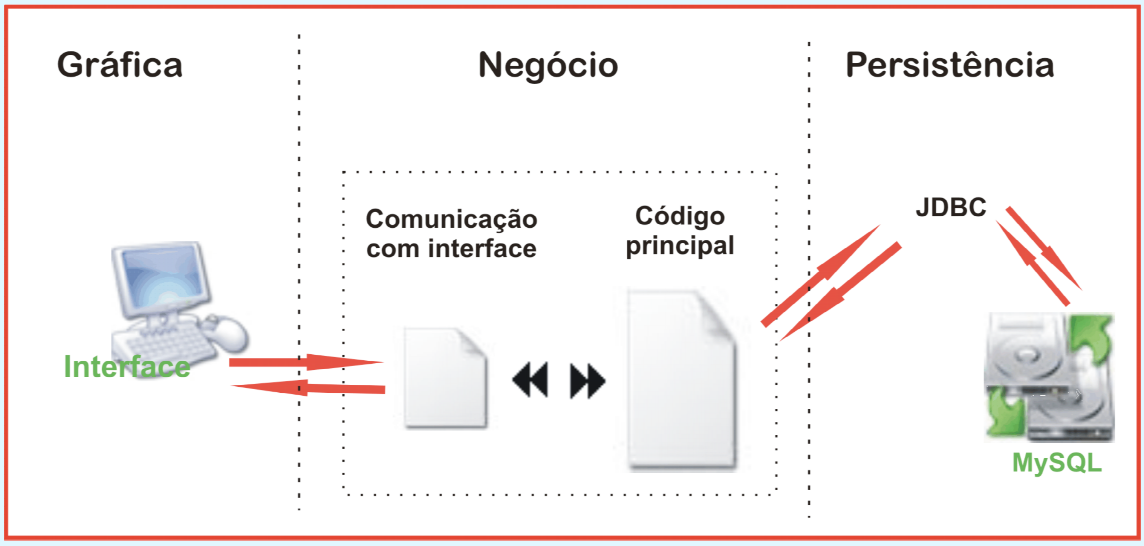
**SOBRE A CAMADA DE INTERFACE**

**O sistema será acessado usando que tipos de clientes?** Público infantil.
**Como fazer a interface gráfica?** Com o auxílio do pacote javax.swing
**Com que ferramentas?** IDE NetBeans
**Com que componentes visuais?** Serão usadas imagens.
**Serão desenvolvidos? Comprados?** Todas as imagens serão criadas pela equipe de desenvolvimento e/ou baixadas da internet – desde que sejam liberadas para quaisquer propósitos pelos autores.
**Há considerações de localização (para outras línguas ou outros países)?** Não.

**SOBRE A CAMADA DE LÓGICA DA APLICAÇÃO**

**Quais são os componentes principais a fazer?** Camada de Negócio, pois nesta camada se encontra o algoritmo do software. É também essa camada que permitirá uma comunicação com a interface (através de uma classe que servirá de ponte) e os dados.
**Há uso de threads?** Sim, para os sons e movimento das imagens.

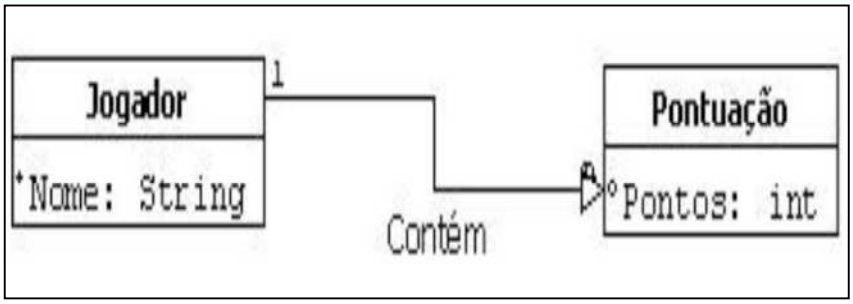
**Quadro 11 -** Projeto Arquitetural (Continuação)

| **SOBRE A CAMADA DE DADOS PERSISTENTES** |
|---|
| **Quais são as fontes de dados? Externas? Internas?Existentes? Novas?** Imagens e áudio serão internos (algumas existentes e outras criadas pela equipe). Haverá informações de origem externa – dados do usuário para recordes.<br>**Que estratégia de persistência será usada:**<br>**Em arquivos?** Imagens e áudio, sim.<br>**Usando um SGBD?** Dados do usuário para recordes.<br>**Qual paradigma de SGBD usar?** Relacional.<br>**Como será feita a população dos bancos de dados**? Através da aplicação. |
| **PERGUNTAS ADICIONAIS** |
| **Como será resolvida a instalação do produto?** O programa ser portátil. Um *.jar que deverá ser usado em qualquer Sistema Operacional que tenha a mais recente versão da JRE.<br>**O que será comprado, feito, modificado?** Nada/classes e imagens/algumas imagens.<br>**Como será a distribuição do software?** Entregue diretamente ao cliente que a distribuirá como quiser. |
| **IMAGEM DO PROJETO ARQUITETURAL** |
|  |

**Fonte** (Questionário): Sauvé, Jacques - Perguntas a Fazer ao Elaborar um Projeto Arquitetural
**Fonte** (Respostas): Próprio Autor

O modelo lógico de dados é uma representação das informações da área do software, deve ser independente da tecnologia implementada devido a constante mudança dos produtos tecnológicos. A produção deste artefato foi responsabilidade do desenvolvedor. O modelo lógico de dados do Recicla está representado na Figura 15.

**Figura 15 -** Modelo Lógico de Dados




**Fonte:** Próprio Autor

Concluídas as atividades de inicialização, deu-se início ao planejamento do projeto. Toda a equipe do projeto participa do Planejamento do Release e o da Interação. Tais artefatos estão sendo mostrados nos quadros 12 e 13 respectivamente.

**Quadro 12** - Plano de Release

| **RELEASE 01: 14/06 – 16/06** | **GERENTE – ANGÉLICA FÉLIX** | |
|---|---|---|
| **Iteração** | **User Story** | **Período** |
| Iteração 01 | US 01 | 14/06 |
| Iteração 02 | US 02 | 15/06 – 16/06 |
| **RELEASE 02: 17/06 – 18/06** | **GERENTE – ANGÉLICA FÉLIX** | |
| **Iteração** | **User Story** | **Período** |
| Iteração 03 | US 03 | 17/06 |
| Iteração 04 | US 04, USO 05 | 18/06 |
| **RELEASE 01: 19/06 –20 /06** | **GERENTE – ANGÉLICA FÉLIX** | |
| **Iteração** | **User Story** | **Período** |
| Iteração 05 | US 06 | 19/06 |
| Iteração 06 | US 07 | 20/06 |

**Fonte:** Próprio Autor

**Quadro 13 -** Plano de Iteração

**ITERAÇÃO 01 – (14/06/2012)**

**USO 01 - Implementar a queda dos objetos recicláveis na tela**

| **Testes de aceitação** | | **Status** |
|---|---|---|
| **TA1.1** | Verificar se os recicláveis se alternam após a queda. (SUCESSO) |  |
| **TA1.2** | Verificar se os recicláveis caem em lugares alternados. (SUCESSO) |  |

| **Atividade** | **Descrição** | **Responsável** | **Estimativa de tempo** | **Tempo real** | **Status** |
|---|---|---|---|---|---|
| A1.1 | Implementar a interface para a queda de objetos | Angélica | 3h | 3,5 |  |
| A1.2 | Implementar a funcionalidade para a queda de objetos | Betoven | 4h | 4h |  |
| A1.3 | Instalar aplicação | Betoven | 1h | 0,5 |  |

**ITERAÇÃO 02 – (15/06/2012 – 16/06/2012)**

**USO 02 -** Implementar um conjunto de lixeiras que deve ser manipulado a fim de coletar os recicláveis que descem na tela.

| **Testes de aceitação** | | **Status** |
|---|---|---|
| **TA2.1** | Averiguar se os limites de deslocamento das lixeiras correspondem ao esperado. (SUCESSO) |  |
| **TA2.2** | Averiguar se as lixeiras são corretamente movimentadas a partir do teclado e do mouse.(SUCESSO) |  |

**Quadro 13 -** Plano de Iteração (Continuação)

| Atividade | Descrição | Responsável | Estimativa de tempo | Tempo real | Status |
|---|---|---|---|---|---|
| A2.1 | Implementar a interface para a manipulação do conjunto de lixeiras. | Angélica | 3h | 3h |  |
| A2.2 | Implementar a funcionalidade para a manipulação do conjunto de lixeiras. | Betoven | 7h | 8h |  |
| **ITERAÇÃO 03 – (17/06/2012)** | | | | | |
| **USO 03 -** Implementar displays de pontuação e fase. | | | | | |

| Testes de aceitação | | Status |
|---|---|---|
| **TA3.1** | Averiguar se os valores do contador de pontos estão sendo "setados" no display de pontuação. (SUCESSO) |  |
| **TA3.2** | Averiguar se o display de fase é modificado a partir dos valores especificados. (SUCESSO) |  |
| **TA3.3** | Averiguar se o contador de pontos funciona se e somente se os recicláveis caem na lixeira correta. (SUCESSO) |  |
| **TA3.4** | Averiguar se as telas de parabéns e incentivo são abertas após as fases serem conquistadas. |  |

| Atividade | Descrição | Responsável | Estimativa de tempo | Tempo real | Status |
|---|---|---|---|---|---|
| A3.1 | Implementar a interface para a contagem de pontos. | Betoven | 2h | 2h |  |

**Quadro 13 -** Plano de Iteração (Continuação)

| | | | | | |
|---|---|---|---|---|---|
| A3.2 | Implementar a funcionalidade para contagem de pontos. | Betoven | 2h | 3h |  |
| A3.3 | Implementar a interface para avanço de fase. | Angélica | 2h | - |  |
| A3.4 | Implementar a funcionalidade para avanço de fase. | Angélica | 2h | 3h |  |
| **ITERAÇÃO 04 – (18/06/2012)** | | | | | |
| **USO 04 -** Implementar guia simples de orientação ao uso.<br>**USO 05 -** Implementar fechamento alternativo do jogo. | | | | | |

| **Testes de aceitação** | | **Status** |
|---|---|---|
| **TA4.1** | Averiguar se o guia é ativado a partir de um simples clique e se satisfaz o cliente. (SUCESSO) |  |
| **TA4.2** | Averiguar se o jogo é fechado com as mesmas características do fechamento default e se satisfaz o cliente. |  |

| **Atividade** | **Descrição** | **Responsável** | **Estimativa de tempo** | **Tempo real** | **Status** |
|---|---|---|---|---|---|
| A4.1 | Implementar a interface para guia rápido no na própria tela do jogo. | Angélica | 1h | 1h |  |
| A4.2 | Implementar a funcionalidade guia rápido. | Angélica | 2h | 1,5h |  |

**Quadro 13 -** Plano de Iteração (Continuação)

| | | | | | |
|---|---|---|---|---|---|
| A4.3 | Implementar a funcionalidade fechamento alternativo do jogo. | Angélica | 1h | 1h |  |
| **ITERAÇÃO 05 – (19/06/2012)** | | | | | |
| **USO 06 -** Implementar visão de recordes. | | | | | |

| **Testes de aceitação** | | **Status** |
|---|---|---|
| **TA6.1** | Testar entradas e visões no SGBD antes de ligá-lo ao jogo. |  |
| **TA6.2** | Testar entradas e visões (diversas vezes) através do jogo. |  |

| **Atividade** | **Descrição** | **Responsável** | **Estimativa de tempo** | **Tempo real** | **Status** |
|---|---|---|---|---|---|
| A6.1 | Gerar script para configuração do MySQL. | Angélica | 2h | - |  |
| A6.2 | Vincula o banco de dados a aplicação. | Betoven | 4h | - |  |
| **ITERAÇÃO 06 – (20/06/2010)** | | | | | |
| **USO 07 -** Implementar os sons do jogo. | | | | | |

| **Testes de aceitação** | | **Status** |
|---|---|---|
| **TA7.1** | Averiguar a saída de som para os cliques. |  |
| **TA7.2** | Averiguar os sons após a passagem de fase e finalização(zeragem) do jogo. |  |

**Quadro 13 -** Plano de Iteração (Continuação)

| Atividade | Descrição | Responsável | Estimativa de tempo | Tempo real | Status |
|---|---|---|---|---|---|
| A7.1 | Implementar os sons da aplicação. | Angélica e Betoven | 8h | - |  |

**Fonte: Próprio Autor**

Toda a equipe de desenvolvimento deve participar da reunião, pois as discussões se darão em torno da produção de seus membros, assim como das situações vivenciadas por cada um deles durante a semana. Para facilitar a avaliação, é rigorosamente recomendado fazer uso da análise de riscos (Quadro 14).

**Quadro 14 -** Análise de Risco

| Data | Risco | Prioridade | Responsável | Status | Providência/ solução |
|---|---|---|---|---|---|
| 14/06 | Implementar a queda dos objetos recicláveis na tela. | Alta | Todos da equipe | Superado | Estudar e tentar superar o risco considerando o tempo necessário para iteração. |
| 18/06 | Implementar a tela para avanço de fase. | Baixa | Angélica | Abortado | Devido ao tempo e a prioridade a tarefa foi abortada. |
| 19/06 | Uso da tecnologia desconhecida. (MySQL). | Média | Todos | Abortado | Devido ao tempo e a falta de conhecimento suficientes em banco de dados a tarefa foi abortada. |
| 20/06 | Implementação dos sons. | Baixa | Todos | Abortado | Devido ao tempo e a prioridade a tarefa foi abortada. |

**Fonte:** Próprio Autor

Outro artefato que facilita a avaliação é o Big Chart (Quadro 15), pois este permite que possíveis mudanças inerentes ao projeto sejam levantadas durante a reunião.

**Quadro 15** – Big Chart

| Data | Classes | Scripts | Testes de aceitação | Testes de unidade | User Stories | Observação |
|---|---|---|---|---|---|---|
| 14/06 | 0 | 0 | 0 | 0 | 0 | Inicio da implementação |
| 17/06 | 7 | 0 | 7 | 10 | 3 | Abortada a criação das telas de avanço de fase, por problemas técnicos. |
| 21/06 | 14 | 0 | 12 | 30 | 5 | Abortada a implementação dos sons do jogo e a vinculação do banco de dados. |

**Fonte:** Próprio Autor

Para que todas as esferas do software fossem avaliadas foram realizados Testes de Usabilidade, Teste de Aprendizagem e o Teste de Qualidade Pedagógica. O teste de Usabilidade verifica se os objetivos de usabilidade especificados pelo cliente e pelo usuário foram satisfeitos, tal artefato segue no quadro 16.

**Quadro 16 -** Teste de Usabilidade

| ROTEIRO DAS ATIVIDADES | |
|---|---|
| DESCRIÇÃO INICIAL | Neste Teste de Usabilidade, você assumirá as funções de aluno da pré-escola e atuará como usuário do jogo Recicla. Este jogo possibilita escolher em qual lixeira da coleta seletiva o lixo será colocado com o intuito de auxiliar a aprendizagem dos alunos a respeito do assunto.<br><br>**OBS.:** Caso encontre alguma dificuldade que não comprometa a realização da atividade, não se preocupe e siga em frente. |
| ATIVIDADE 01: Efetuar Login no sistema | **Roteiro:** Nesta atividade você irá entrar no Recicla informando o seu nome.<br>**Instruções**<br>○ Abra o Recicla clicando duas vezes no ícone referente ao mesmo;<br>○ Informe seu nome;<br>○ Verifique se o programa abriu corretamente. |
| ATIVIDADE 02: Jogar Nível 01 | **Roteiro:** Nesta atividade você irá escolher em qual lixeira da coleta seletiva o lixo que está caindo será colocado.<br>○ Mova as lixeiras com as setas do teclado (direita e esquerda) ou clicando nos botões de seta no próprio jogo;<br>○ Verifique se o lixo está caindo corretamente;<br>○ Verifique se as lixeiras estão se movendo corretamente; |

**Quadro 16** – Teste de Usabilidade (Continuação)

| | |
|---|---|
| ATIVIDADE 03: Jogar Nível 02 | **Roteiro:** Nesta atividade você irá escolher em qual lixeira da coleta seletiva o lixo que está caindo será colocado.<br>• Mova as lixeiras com as setas do teclado (direita e esquerda) ou clicando nos botões de seta no próprio jogo;<br>• Verifique se o lixo está caindo corretamente e mais rápido que no nível anterior;<br>• Verifique se as lixeiras estão se movendo corretamente; |
| ATIVIDADE 04: Jogar Nível 03 | **Roteiro:** Nesta atividade você irá escolher em qual lixeira da coleta seletiva o lixo que está caindo será colocado.<br>• Mova as lixeiras com as setas do teclado (direita e esquerda) ou clicando nos botões de seta no próprio jogo;<br>• Verifique se o lixo está caindo corretamente e mais rápido que no nível anterior;<br>• Verifique se as lixeiras estão se movendo corretamente; |
| ATIVIDADE 05: Fechar o jogo | **Roteiro:** Nesta atividade você irá fechar jogo clicando no botão SAIR no próprio jogo ou no botão fechar na barra de tarefas. |

**Fonte:** Próprio Autor

O Teste de Aprendizagem verifica se os usuários do SE estão realmente aprendendo o conteúdo substancial do aplicativo, o roteiro utilizado para este teste segue no quadro 17.

**Quadro 17 -** Teste de Aprendizagem

| ROTEIRO DAS ATIVIDADES | |
|---|---|
| DESCRIÇÃO INICIAL | Neste Teste de Aprendizagem será avaliado o conhecimento retido após a utilização do jogo Recicla. |
| ATIVIDADE 01: realizar questionários orais. | **Roteiro:** Nesta atividade você realizará questionários com os alunos que envolvam coleta seletiva. |
| ATIVIDADE 02: avaliação das respostas | **Roteiro:** Nesta atividade você avaliará a quantidade de respostas certas e erradas, atentando para a intensidade de respostas. A avaliação ocorrerá de maneira contínua, a partir da observação do desempenho e interesse dos alunos em relação às atividades desenvolvidas além da participação em sala. |

**Fonte:** Próprio Autor

O Teste de Qualidade Pedagógica (Quadro 18), como o próprio nome sugere, verifica os aspectos pedagógicos. Para este teste foi utilizada uma adaptação da metodologia proposta pela Escola do Futuro da USP, LECT-Laborat Laboratório Ensino Ciências e Tecnologia.

**Quadro 18 -** Teste de Qualidade Pedagógica

| **S** | **PP** | **P** | **PM** | **N** | **NA** |
|---|---|---|---|---|---|
| Sim | Parcialmente com POUCAS restrições | Parcialmente | Parcialmente e com MUITAS restrições | Não | Não se aplica |

| | **QUESITOS PARA AVALIAÇÃO DO SOFTWARE.** | **S** | **PP** | **P** | **PM** | **N** | **NA** |
|---|---|---|---|---|---|---|---|
| 1 | O software favorece a Memorização | X | | | | | |
| 2 | A dificuldade dos exercícios propostos é adequada quanto à faixa etária? | X | | | | | |
| 3 | O erro é tratado no software como processo de aprendizagem? | | | X | | | |
| 4 | O erro é apontado ao aluno no software como algo negativo? | | | | | X | |
| 5 | A proposta de solução problema é relacionada ao conteúdo? | | | | | X | |
| 6 | São apresentados múltiplos caminhos para a solução de problemas? | | | | X | | |
| 7 | O software proporciona interação entre diferentes disciplinas? | | | | X | | |
| 8 | Os conteúdos são abordados de forma clara facilitando a compreensão? | X | | | | | |
| 9 | Erros conceituais? | | | | | X | |
| 10 | Erros gramaticais ou ortográficos? | | | | | X | |
| 11 | Apresenta discriminação / preconceito? | | | | | X | |
| 12 | A Navegação pelo software é fácil? | X | | | | | |
| 13 | O contexto é adequado ao design e são atraentes (interface)? | X | | | | | |
| 14 | É possível a retomada da atividade em um determinado ponto em outro momento? | | | | | X | |
| 18 | O manual de utilização tem linguagem apropriada e de fácil entendimento? | | | | | | X |
| 20 | Funciona em rede? | | | | | X | |
| 21 | Os recursos de hipertexto e hiperlink funcionam e estão adequados? | | | | | | X |
| 22 | A ajuda ao usuário sobre o conteúdo é adequada e aparece em todo o software? | | | | | X | |
| 23 | O software propicia a interação entre Aprendiz x Agente de Aprendizagem | | X | | | | |
| 24 | O software propicia a interação entre Aprendiz x Agente de Aprendizagem X Grupo | | X | | | | |
| 25 | O software propicia a interação entre Aprendiz X Máquina | | X | | | | |

**Fonte:** Próprio Autor

## 4. 2 ANÁLISE DO PROCESSO DE DESENVOLVIMENTO YPEDUC

Uma metodologia de desenvolvimento tenta disciplinar um processo definido, no entanto o fato de ser baseado em pessoas exige flexibilidade. Por isso, mensurar a implantação de uma metodologia não é simples. Ainda mais quando se trata do projeto de softwares educativos, pois envolvem ainda mais pessoas tanto na utilização do software quanto na sua concepção.

Propõe-se para este trabalho que a avaliação do sucesso da implantação do YPEduc seja a constatação do correto seguimento dos passos propostos e respeito aos requisitos elencados, além da comodidade dos envolvidos ao utilizá-la.

A implantação do YPEduc no estudo de caso deste trabalho teve como intuito analisar a adequação dos critérios levantados à metodologia ágil. Com isso a validação conseguida não versou a cerca da metodologia em si, pois para isto seria necessário um desenvolvimento real que tivesse um cliente real, um domínio específico real de qualquer área de conhecimento e uma equipe multidisciplinar disponivel para a elaboração do sistema capaz de sanar as necessidades presentes neste domínio.

Como foi visto, o esforço na constituição de uma equipe multidisciplinar e multiprofissional com especialistas da educação torna-se complexo, devido à variação do trabalho solitário a prática da interatividade multiprofissional e multifacetada da equipe de produção.

Daí surge uma divergência entre a indispensável presença de uma fundamentação pedagógica que permeie todo o desenvolvimento e que define a forma de interação do SE com o professor e aluno e a dificuldade de relacionar esta diversidade de perfis durante o processo de desenvolvimento.

É possível observar que não foram feitas recomendações específicas para os artefatos Modelo da Tarefa, Modelo lógico de Dados, Projeto Arquitetural, Testes de Usabilidade, Matriz de Competências e Tabela de Alocação de Atividades e nas fases de Planejamento e Implementação. Isto ocorreu pois apesar de também serem importantes, não possuem pontos específicos a processos de desenvolvimento de Softwares Educativos, portanto as recomendações especificadas no YP podem ser reutilizados no desenvolvimento de uma aplicação utilizando o YPEduc.

Outro fator importante no YPEduc é a valorização da interface gráfica em softwares educativos, considerando que esta é a embalagem, ou a capa do produto final e que essa questão deve ser tratada com os devidos cuidados, pois, por mais eficiente que seja um SE, se

sua imagem na tela não for agradável aos olhos dos usuários, então pode haver uma rejeição desse sistema.

Para melhor visualizar a análise dos critérios essenciais a uma metodologia ágil de desenvolvimento de software educativo, os critérios elencados serão divididos no quadro 19 sendo classificados em Facilmente Aplicável (FA), Parcialmente Aplicável (PA) e Dificilmente Aplicável (DA), seguindo da justificativa da classificação.

**Quadro 19** - Análise do Processo de Desenvolvimento YPEduc

| **ANÁLISE DOS CRITÉRIOS ESSENCIAIS A UMA METODOLOGIA ÁGIL DE DESENVOLVIMENTO DE SOFTWARE EDUCATIVO** | | |
|---|---|---|
| **Critérios** | **Classificação** | **Justificativa** |
| Participação da equipe multidisciplinar no projeto do software | PA | Esta interatividade multiprofissional é de fundamental importância no processo, no entanto é algo complexo de ser colocado em prática; |
| Valorizar informações relacionadas à aprendizagem de conceitos e ao contexto de uso durante a conversa com o cliente | FA | É um critério indispensável para a efetivação do caráter educativo do software, alem de ser uma atitude simples de ser posta em prática. |
| Considerar questões como público-alvo, contexto, conteúdo e avaliação durante a Análise de Requisitos | FA | Respeitar estas questões sugeridas enfatizam o caráter educacional desejado, auxiliando a quebrar a padronização que tende a acontecer nestes processos e para ser posta em prática não exige muito da equipe de projeto. |
| Durante o levantamento do Perfil do aluno deve-se obter informações sobre o conhecimento prévio em relação ao conteúdo abordado pelo software | PA | A dificuldade deste critério está em avaliar o conhecimento prévio do aluno, pois dependendo da forma que esta avaliação seja realizada ela tende a nivelar os alunos deixando de respeitar a individualidade do processo de ensino-aprendizagem. |
| Levar em consideração o perfil do professor mediador, obtendo informações sobre as dificuldades encontradas pelo professor ao lecionar o conteúdo abordado pelo SE | FA | Este critério sugere um artefato simples de ser produzido e que traz grandes benefícios para o resultado final. |
| Durante a definição dos objetivos de usabilidade deve ser levada em conta além da aprendizagem do sistema a aprendizagem do conteúdo educacional | FA | Este critério sugere apenas que se respeite o caráter educacional do software, sendo simples de ser colocado em prática. |
| Professor e aluno também devem participar da produção dos objetivos de usabilidade | DA | Quanto mais pessoas precisam interagir e dividir responsabilidade mais transtornos podem acontecer. |

**Quadro 19** - Análise do Processo de Desenvolvimento YPEduc (Continuação)

| | | |
|---|---|---|
| O contexto de uso educacional necessita ser considerado e respeitado para a produção das User Stories | FA | Este critério sugere apenas que se respeite o caráter educacional do software, sendo simples de ser colocado em prática. |
| Na fase de prototipação da interface deve ser levado em consideração as múltiplas inteligências e o perfil dos alunos, a fim de contemplar o máximo de alunos possível com o aprendizado intuitivo | PA | A valorização da interface gráfica em softwares educativos é fundamental para o sucesso do software, no entanto este critério dificulta um pouco o processo de desenvolvimento. |
| É necessário que além dos testes tradicionais, também ocorram Testes de Aprendizagem e Testes de Qualidade Pedagógica, criados pela equipe multidisciplinar para avaliar a aprendizagem dos alunos e o caráter pedagógico, respectivamente. | FA | Este critério é indispensável para avaliar se os usuários do SE estão realmente aprendendo o conteúdo substancial do aplicativo, além de sugerir a verificação dos aspectos pedagógicos da aplicação. Além de trazer inúmeros benefícios para o produto final este é um critério de fácil aplicação, pois não exige esforço da equipe do projeto. |

Analisando os resultados obtidos, é importante observar que os critérios elencados tendem a amadurecer com o tempo e a prática deste processo será essencial para a evolução e utilidade da proposta.

# 5 CONCLUSÕES E CONSIDERAÇÕES FINAIS

## 5.1 CONTRIBUIÇÕES

A fraqueza de muitos artefatos educacionais é que eles não se baseiam em uma análise adequada das necessidades do aluno desenvolver um software com caráter efetivamente educativo de forma ágil e eficiente significa observar detalhes em cada uma das etapas de produção do mesmo, escolhendo formalismos que potencializem as características educacionais, e produzindo apenas artefatos realmente necessários durante o processo.

Com todos os aspectos observados, constata-se que apesar da importância da utilização de metodologias de desenvolvimento amparada em formalismos e ferramentas para o desenvolvimento ágil de software educativo, a principal mudança que deve ser realizada neste segmento é de caráter comportamental, tendo em vista que os pontos críticos deste problema envolvem essencialmente a maneira como a equipe de projeto utiliza as informações inerentes do contexto educativo.

Neste sentido, a proposta do YPEduc marca um avanço e o início de organização do processo de desenvolvimento de aplicações educacionais. Através dela é possível perceber pontos essenciais a este processo que aliadas às revisões de literatura contribuem para a evolução do processo e do conjunto de especificações propostas. É importante observar que o conjunto de critérios elencados tende a amadurecer com o tempo e a prática do desenvolvimento deste processo será de grande auxílio para a evolução e utilidade da proposta.

Foi possível observar que ainda há muito para evoluir no que diz respeito a adequação de ambientes de desenvolvimento de software para auxílio ao processo de ensino-aprendizagem. E por todos esses aspectos, deve-se utilizar outro olhar para o processo de construção de softwares educativos pode permitir importantes alternativas para o entendimento do funcionamento da situação educativa mediada por tecnologias vivenciada em sala de aula, tendo consequência materiais didáticos de fato inovadores.

Pretende-se, portanto, que este trabalho sirva de contribuição para reflexões que possam tirar proveito dos conhecimentos e dos autores aqui referenciados. E, por fim, que o mesmo possa criar lacunas e suscitar questionamentos que induzam a incursões no sentido de se chegar a um maior entendimento sobre novas formas de utilização das tecnologias digitais a serviço da educação.

## 5.2 LIMITAÇÕES

Por se tratar de uma monografia, a autora assumiu praticamente todos os papéis do projeto e o estudo de caso realizado teve uma pequena proporção com o intuito de verificar apenas se todos os pontos importantes foram acomodados no processo, deixando a desejar na validação da metodologia adaptada.

Ainda sobre o estudo de caso, não houve possibilidade de interagir com a equipe multidisciplinar proposta para fundamentar o caráter educativo do software, havendo assim perda neste quesito da concepção do software. O tempo foi outro fator limitante no desenvolvimento deste trabalho, pois como está retratado no Big Chart do projeto, não foi possível implementar todas as especificações desejadas para o software Recicla.

## 5.3 TRABALHOS FUTUROS

As contribuições apresentadas representam um primeiro esforço para o uso de processos específicos para o desenvolvimento de softwares educativos. Apesar dos benefícios apresentados por meio dos estudos realizados, este trabalho pode ser continuado através de outros estudos experimentais utilizando a metodologia adaptada - YPEduc. Com novos estudos podem surgir novas estratégias de refinamento dos critérios essenciais elencados neste trabalho.

É importante destacar também que o YPEduc seja avaliado para fins didáticos, sendo posto em prática por alunos da UEPB na disciplina de Engenharia de Software quando são realizados projetos de desenvolvimento de SE, para que assim possam ser mensuradas as verdadeiras contribuições desta metodologia.

Seria interessante também que fosse realizado um estudo comparativo entre o YPEduc e outros processos de desenvolvimento de softwares educativos para que assim possam surgir novas adaptações e reflexões sobre este tipo de processo de desenvolvimento.

## REFERENCIAS


ABRAHAMSSON, P.; SALO, O.; RONKAINEN, J.; WARSTA, J. **Agile Software Development Methods:** Reviews and Analysis. Espoo: VTT Publications, 2002.

AGILE MANIFESTO. Manifesto for Agile Software Development. Agile Alliance,2001. Disponível em: <http://www.agilemanifesto.org/>. Acessado em: 15 de Março de 2012.

**Avaliação de Software Educativo** - Escola do Futuro da USP, LECT-Laborat Laboratório Ensino Ciências e Tecnologia. Disponivel em: http://www.slideshare.net/Ediclei/ficha-avaliao-de-software-educativo. Acesso em: 13 de Junho de 2012.

BASSANI, Patricia. B. Scherer ; PASSERINO, L. M. ; PASQUALOTTI, P. ; RITZEL, M.; **Em busca de uma proposta metodológica para o desenvolvimento de software educativo colaborativo**. RENOTE. Revista Novas Tecnologias na Educação, v. 4, p. 1-10, 2006.

BECK, Kent.; **Extreme Programming Explained.**,New York: 2000, Addison Wesley.

BECK, Kent.; **Programação extrema (XP) explicada: acolha as mudanças.** Porto Alegre, RS; Bookman, 2004.

BENITTI, Fabiane Barreto Vavassori ; SEÁRA, Everton Flávio Rufino ; SCHLINDWEIN, Luciane Maria . **Processo de desenvolvimento de software educacional:** proposta e experimentação. RENOTE. Revista Novas Tecnologias na Educação, Porto Alegre, v. 3, n. 1, p. 1-10, 2005.

BOYLE, T.; COOK, J.; WINDLE, R.; Wharrad, H.; JEEDER, D.; ALTON, Rob.. **An Agile method for developing learning objects**. Proceedings of the 23rd annual ascilite conference: Who's learning? Whose technology? Sydney, Australia, 2006.

CAMPOS, G. H. B.; CAMPOS, F. C. A.; **Qualidade de software Educacional**. In: Ana Regina Cavalcanti da Rocha. (Org.). Qualidade de software: Teoria e Prática. Campinas: Makron, 2001. Disponível em: http://www.cciencia.ufrj.br/publicacoes/Artigos/EduBytes95/ QualidadeSE.htm; Acesso em: 10 de Maio de 2012.

CARVALHO, Ariadne M. B. R., CHIOSSI, Thelma. C. S. **Introdução à Engenharia de Software.** São Paulo: Editora da Unicamp, 2001, 148 p.

CERVO, A. L.; BERVIAN, P. A.; DA SILVA, R.; **Metodologia Científica.** 6 ed. – São Paulo: Prentice Hall, 2006. v. 1. 162 p.

COCKBURN, A. e HIGHSMITH, J. "Agile Software Development: The Business of Innovation", IEEE Computer, Sept., (2001), pp. 120-122.

COSTA FILHO, Edes Garcia da; PENTEADO, R. A. D.; ANACLETO, J. C.; BRAGA, R. T. V.; **Padrões e Métodos Ágeis:** Agilidade no processo de desenvolvimento de software. In: Fifth Latin American Conference on Pattern Languages of Programs, 2005, Campos do Jordão - SP. Anais da 5ª Conferência Latino-Americana em Linguagens de Padrões para Programação, 2005. V. 1.

Disponível em: http://sugarloafplop2005.icmc.usp.br/papers/9673.pdf; Acesso em: 25 de fevereiro 2012.

FERREIRA, R. B. ; LIMA, F. P. A.; **Metodologias Ágeis:** Um Novo Paradigma de Desenvolvimento de Software. In: SBQS 2006, 2006, Vilha Velha. Woses 2006. Rio de Janeiro: PESC/COPPE, 2006. v. 2. p. 107-115.

**Formulário para o levantamento do Perfil do Usuário;** 2003, Laboratório de Interfaces Homem-Máquina, UFCG. Disponível em: http://www.dee.ufcg.edu.br/~scaico/facisa/ihm/ perfil_do_usuario.pdf; Acesso em: 15 de Junho de 2012.

FUGGETTA, A. **Software Process:** A Roadmap, In: Proceedings of The Future of; 2000.

GAMA, C. L. da;. **Método de construção de objetos de aprendizagem com aplicação em métodos numéricos**. Tese de doutorado, Curitiba: UFPR, 2007. Disponível em: http://www.ppgmne.ufpr.br/arquivos/teses/9.pdf; Acesso em: 14 de Abril de 2012.

GAMEZ, L.; **Técnica de Inspecção de Conformidade Ergonômica de Software Educacional.** 1998; Disponível em: http://www.cin.ufpe.br/~case/artigos/Avaliacao%20e% 20Classificacao/ manual %20 ticese.pdf; Acesso em: 12 de Maio de 2012.

GARCIA et al.; **easYProcess:** Um Processo de Desenvolvimento para Uso no Ambiente Acadêmico. In: XXII WEI - Workshop de Educação em Computação, XXIV Congresso da Sociedade Brasileira de Computação, 2004, Salvador. SBC, 2004.

GIL, A. C.; **Como elaborar projetos de pesquisa.** 5 ed. - São Paulo: Atlas, 2010. v. 1. 171 p.

GIRAFFA, L. M. M. ; MARCZAK, S. S. ; PRIKLADNICKI, R. . **PDS-E:** Em Direção a um Processo para Desenvolvimento de Software Educacional. In: XXV Congresso da SBC - Workshop sobre Informática na Escola (WIE), 2005, São Leopoldo. Anais do XXV Congresso da SBC, 2005. v. 1. p. 2833-2841.

GOMES, A. S.; WANDERLEY, E. G.; **Elicitando requisitos em projetos de Software Educativo**. In: WIE 2003 Workshop Brasileiro de Informática Educativa, 2003, Campinas. Ciência, Tecnologia e Atalhos para o futuro - Anais do XXIII Congresso da Sociedade Brasileira de Computação. Campinas : SBC, 2003. v. V. p. 227-238.

HIGHSMITH, Jim. **Agile Software Development Ecosystems.** Adisson-Wesley, 2002.

JUCÁ, S. C. S.; **A Relevância dos Softwares educativos na Educação Profissional**. Ciências & Cognição (UFRJ), v. 08, p. 03, 2006. Disponível em: www.cienciasecognicao.org.br. Acesso em: 14/01/2012

KOSCIANSKI, A.; SOARES, M. S.; **Qualidade de Software.** 2. ed. São Paulo: Novatec, 2007. v. 1. 395 p.

LACERDA, R. A.; **Proposta de um modelo para análise de requisitos de software educativo;** 2007; Universidade de Brasília, UNB, Brasil. (Dissertação).

LAGO, Samuel R. **Educação hoje** – uma reflexão para pais e educadores. In: Gazeta do Povo, 2004.

LARMAN, Craig. **Applying UML And Patterns.** 2nd Edition, 2002.

MEDEIROS, F. P. A.; BARBOSA, P. C.; LULA JR, B.; **iTAOS** - Ferramenta para Modelagem da Tarefa. 2003.

MORAES, M. A. **Informática, a angústia do usuário**. Revista Developers, n. 31, ano 3, p.66, Março, 1998.

NETO, Erasmo Isotton. **Scrumming -** Ferramenta Educacional para Ensino de Práticas de SCRUM. 2008. Trabalho de Conclusão de Curso. (Graduação em Bacharelado em Sistemas de Informação) - Pontifícia Universidade Católica do Rio Grande do Sul. Orientador: Rafael Prikladnicki.

OLIVEIRA, Celina Couto de; COSTA, José Wilson da; MOREIRA, Márcia. **Ambientes informatizados de aprendizagem:** produção e avaliação de software educativo. São Paulo, SP: Papirus, 2001. 144 p.

PARSONS, D., RYU, H. e LAL, R. ; **The impact of methods and techniques on outcomes from agile software development projects**. In: Organizational Dynamics of Technology-Based Innovation: Diversifying the Research Agenda, v.235 of IFIP, p. 235–249. Springer Boston; 2007.

SAUVÉ, Jacques Philippe; **Perguntas a Fazer ao Elaborar um Projeto Arquitetural** – Disponível em: http://jacques.dsc. ufcg.edu.br/ cursos/proj/gerenciadesenv/arquitetural.htm. Acesso em: 15 de Maio de 2012**.**

PERRY, G. T; **Proposta de uma Metodologia Participativa para o Desenvolvimento de Software Educacional**; 2006; Universidade Federal do Rio Grande do Sul, UFRGS, Brasil. (Dissertação).

PET/UFCG. **easYProcess: Um processo de desenvolvimento de software.** Apostila. 2007. Disponível em: < http://www.mps.xpg.com.br/YP.pdf >. Acesso em: 14 de Março de 2012.

PRESSMAN, Roger S. **Software Engineering: a practitioner's approach**. EUA: McGraw Hill, 2001. 860 p.

REZENDE, F.; **As Novas Tecnologias na Prática Pedagógica sob a Perspectiva Construtivista**. Ensaio. Pesquisa em Educação em Ciências, Belo Horizonte, v. 2, n. 1, p. 75-98, 2000.

ROCHA, H. V.; BARANAUSKAS, M. C. C.; **Design e Avaliação de Interfaces Humano-Computador.** 1. ed. São Paulo: Escola de Computação - 12, Sp, 2000, 2000. 242p .

SCHWABER, K. **Scrum Development Process**, OOPSLA'95 Workshop on Business Object Design and Implementation. Springer-Verlag, 1995.

SCHWABER, K; e BEEDLE, M; **Agile Software Development with SCRUM**. Prentice-Hall, 2002.

SILVA, R. J. S.; **Avaliação de Software Educacional:** critérios para definição da qualidade do produto. Simpósio Nacional ABCiber, v. 1, p. 1-15, 2009.

SOARES, M. S.; **Metodologias Ágeis Extreme Programming e Scrum para o Desenvolvimento de Software.** RESI. Revista Eletrônica de Sistemas de Informação, v. 3, p. 1-8, 2004.

SOMMERVILLE, I**. Engenharia de Software.** São Paulo: Addison Wesley, 6ª Edição, 2001.

SOMMERVILLE, Ian. **Engenharia de Software.** São Paulo: Addison Wesley, 2003. 592 p.

SOMMERVILLE, I. **Engenharia de Software.** São Paulo: Addison Wesley, 8ª Edição, 2007.

TCHOUNIKINE P., **Pour une ingénierie des Environnements Informatiques pour l'apprentissage humain**, Information-Interaction-Intelligence, vol. 2, n. 1, 2002.

WELLS, D. **Extreme Programming: a gentle introduction.** Disponível em: http://www.extremeprogramming.org; Acesso em: 20 de Fevereiro de 2012.