UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE ESPECIALIZAÇÃO EM COMPUTAÇÃO

ESTUDO COMPARATIVO ENTRE A MODELAGEM EXTREME PROGRAMMING E RATIONAL UNIFIED PROCESS

MONOGRAFIA

Josué Marques Passini

Santa Maria, RS, Brasil
2007

ESTUDO COMPARATIVO ENTRE A MODELAGEM EXTREME PROGRAMMING E RATIONAL UNIFIED PROCESS

por

Josué Marques Passini

Monografia apresentada ao Curso de Especialização em Computação, área de concentração em Sistemas de Informações para Web, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de
Especialista em Sistemas de Informação para Web.

Orientadora: Prof.ª Oni Reasilvia Sichonany

Santa Maria, RS, Brasil.

2007

Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Especialização em Computação

A Comissão Examinadora, abaixo assinada, aprova a Monografia

ESTUDO COMPARATIVO ENTRE A MODELAGEM EXTREME PROGRAMMING E RATIONAL UNIFIED PROCESS.

Elaborada por:
Josué Marques Passini

Como requisito parcial para obtenção do grau de
Especialista em Sistemas de Informação para Web

COMISSÃO EXAMINADORA:

Oni Reasilvia Sichonany, Mestre.
(Presidente/Orientadora)

João Carlos Damasceno Lima, Mestre.

Iria Brucker Roggia, Mestre.

Santa Maria, 31 de Janeiro de 2007.

RESUMO

Monografia
Programa de Pós-Graduação em Ciência da Computação
Universidade Federal de Santa Maria

ESTUDO COMPARATIVO ENTRE A MODELAGEM EXTREME PROGRAMMING E RATIONAL UNIFIED PROCESS.

Autor: JOSUÉ MARQUES PASSINI
Orientadora: ONI REASILVIA SICHONANY
Data e Local da Defesa: Santa Maria, 31 de Janeiro de 2007.

Os sistemas de informação desempenham papéis fundamentais dentro das organizações. Devido a este fato aumentou a demanda para auxiliar na tarefa da administração e operação das empresas. Sendo assim, surgem desafios quanto à qualidade dos sistemas de informação que devem ser entregues aos clientes. Para tanto, são criadas diversas formas de gestionar o processo de desenvolvimento de software com qualidade. Este trabalho apresenta um estudo comparativo entre os processos de engenharia de software Rational Unified Process (RUP) e Extreme Programming (XP) sob o ponto de vista da modelagem do sistema. Nele, busca-se verificar os reais ganhos em tempo de modelagem ao utilizar um método ágil. O RUP consiste em uma abordagem do ciclo de vida adequado a UML. A meta do RUP é permitir a produção de software da mais alta qualidade que atenda às necessidades do usuário final de acordo com planejamento e orçamentos previsíveis [5]. O XP surgiu através do manifesto por um desenvolvimento ágil de software que prega, entre diversos aspectos, a satisfação do consumidor através de entregas rápidas e contínuas do produto de software. Também procura preparar os desenvolvedores para receber bem mudanças nos requisitos, mesmo com o desenvolvimento em andamento. O trabalho foi estruturado em quatro capítulos: Revisão de conceitos (RUP e XP); Estudo comparativo entre RUP e XP (comparação do ciclo de vida, comparação das fases do ciclo de vida); Estudo de Caso (sistema proposto, aplicação do RUP, aplicação do XP) e conclusões.

Palavras Chaves: engenharia de software; Rational Unified Process; Extreme Programming.

ABSTRACT

Monography
Programa de Pós-Graduação em Ciência da Computação
Universidade Federal de Santa Maria

COMPARATIVE STUDY BETWEEN THE MODELING OF EXTREME PROGRAMMING AND RATIONAL UNIFIED PROCESS.

Autor: JOSUÉ MARQUES PASSINI
Orientadora: ONI REASILVIA SICHONANY
Data e Local da Defesa: Santa Maria, 31 de Janeiro de 2007.

The Information Systems play a fundamental role among today's organization. Due to this fact the demand increased to help the administration task and the operation of the enterprises. Therefore, challenges come up, in relation to the quality of the System itself, which should be delivered to the clients to achieve the software development process with quality and various forms of managing it where created. This work presents a comparative study between the software engineering process Rational Unified Process (RUP) and Extreme Programming (XP) under the system modelling point of view. In the study we intent to verify the real modelling time gain when the agile modeling method is used. The RUP consists of a different approach of the life cycle adequate to the UML. The RUP objective is to produce the most high quality software production and supply the final user with predictable planning and budget. The organization of XP was based in the Agile Manifest development software which among various aspects teaches the satisfaction of the clients through rapid and continuous delivery of the software products. It also prepares the developers to well receive the required changes even with the project on course. The work was structured in four chapters: Bibliography Revision (RUP and XP); Comparative Study Between RUP and XP (Comparison of life cycle, Comparison of phases of life cycle); Case Study (Example system, RUP use and XP use) and Conclusion.

Key Words: Software Engineering ; Rational Unified Process; Extreme Programming.

LISTA DE FIGURAS

Figura 1 – Ciclo de Vida do RUP proposto na UML 16
Figura 2 – User Stories proposto por Kent Beck 23
Figura 3 – User Stories proposto por Scott Ambler 23
Figura 4 – Exemplos de CRC cards proposto por Scott Ambler 24
Figura 5 – Exemplo de CRC model proposto por Scott Ambler 24
Figura 6 – Processo da prototipagem proposto por Scott Ambler 25
Figura 7 – Exemplo de um esboço de uma interface proposto por Scott Ambler 26
Figura 8 – Exemplo de um protótipo de uma interface proposto por Scott Ambler 26
Figura 9 – Processo de Engenharia de Software na visão do RUP 31
Figura 10– Ciclo de Vida XP – por Scott Ambler 31
Figura 11– Modelo de Negócios e os Casos de Uso 37
Figura 12– Diagrama de seqüência para o Caso de Uso *Ler E-Mail de Editais* 38
Figura 13– Diagrama de Classes para o Sub-Sistema de Licitações. 39
Figura 14– User Story Ler E-Mail 40
Figura 15– CRC Card que descreve Produto 41
Figura 16– CRC Model para o Sub-Sistema de Licitações 41
Figura 17– Interface Gráfica Sugerida pelo cliente 42

LISTA DE TABELAS

Tabela 1 – Relações entre fases do RUP e XP 32

LISTA DE ANEXOS

Anexo A – Exemplo de User Stories utilizado no estudo de caso
Anexo B – Estudo de Caso aplicando RUP
Anexo C – Estudo de Caso aplicando XP

SUMÁRIO

1 INTRODUÇÃO 10
2 REVISÃO DE CONCEITOS 11
2.1 RATIONAL UNIFIED PROCESS 11
2.1.1 Visão Geral 11
2.1.2 Rup & UML 13
2.1.3 Levantamento de Requisitos 13
2.1.4 Modelagem e Documentação 14
2.1.5 Ciclo de Vida 16
2.1.6 Testes 18
2.2 EXTREME PROGRAMMING 19
2.2.1 Visão Geral 19
2.2.2 Levantamento de Requisitos e Modelagem 22
2.2.3 Ciclo de Vida 27
2.2.4 Testes 28
2.2.5 Documentação 29
3 ESTUDO COMPARATIVO ENTRE RUP E XP 30
3.1 Comparação do Ciclo de Vida 30
3.2 Comparação das Fases do Ciclo de Vida 32
4 ESTUDO DE CASO 35
4.1 Sistema – Sub-Sistema de Controle de Licitações 35
4.2 Aplicação RUP 36
4.3 Aplicação XP 40
5 CONCLUSÃO 43
6 REFERÊNCIAS BIBLIOGRÁFICAS 45

1 INTRODUÇÃO

Os sistemas de informação desempenham papéis fundamentais dentro das organizações nos três níveis da administração: operacional, tático e estratégico, independente do porte dela. Devido a este fato aumentou a demanda por sistemas de informação para auxiliar na tarefa da administração e operação.

Para atender tal aumento de demanda são criados inúmeros sistemas: alguns implementam funcionalidades genéricas, outros são específicos para uma empresa e suas peculiaridades.

Genericamente, o desenvolvimento de um sistema passa por algumas fases, que vão desde o entendimento do objetivo central do software, passando pela sua implementação e utilização plena até chegar ao momento em que este não se faz mais necessário ou tenha se tornado ineficiente, chegando à morte. Este conjunto de fases é chamado de Ciclo de Vida de um projeto. O controle destas atividades que envolvem o ciclo de vida de um projeto é chamado de Processo. Existem diversos processos de gestão de sistemas de informação. Pode-se citar alguns como: Personal Software Process (PSP), Capability Maturity Model (CMM), Software Process Improvement and Capability Determination (SPICE – ISO), Rational Unified Process (RUP), Extreme Programming (XP) entre diversos outros.

A engenharia de software e os processos que visam implementar a qualidade na produção de software têm como objetivo desenvolver sistemas que atendam as reais necessidades dos clientes dentro de um prazo e custo estimado. A qualidade do software está diretamente ligada a estas três variáveis: requisitos, tempo e custo.

Este trabalho visa realizar um estudo comparativo entre dois processos de engenharia de software: RUP e XP. O foco do estudo está centrado em como os processos tratam a modelagem dos sistemas de informação e quais são as reais vantagens do ponto de vista de tempo quando se utilizam métodos ágeis.

O trabalho está organizado em uma revisão dos conceitos de RUP e XP, posteriormente faz-se um comparativo entre os dois processos e por fim realiza-se um estudo de caso para extrair as devidas conclusões.

2 REVISÃO DE CONCEITOS

Os sistemas de informação são parte vital na gestão das organizações modernas. Independente do tamanho, cada vez mais elas precisam deles para reagir aos problemas e oportunidades do ambiente de negócios [1]. Por conseguinte, a demanda por sistemas tem aumentado.

Para que os sistemas sejam desenvolvidos de forma eficaz e eficiente estuda-se a engenharia de software. Ela por sua vez, visualiza os problemas e as soluções para questões como: processo de produção, qualidade, aspectos humanos envolvidos, custo, métricas, etc.

Quando se trata de desenvolvimento de sistemas de informação deve-se verificar a importância do planejamento e o controle sobre estas e outras atividades.

Os problemas associados ao desenvolvimento de um software são de tal dimensão que é fundamental a definição e a aplicação de princípios, regras e estratégias que conduzam a melhorias significativas em todo processo de desenvolvimento do projeto [2]. Existem diversas metodologias ou processos de produção de sistemas de informação no mercado; cada qual com suas vantagens e desvantagens que afetam diretamente o ciclo de vida do sistema.

Ciclo de Vida consiste nas fases que um software passa desde a sua idealização até sua morte. Existem diversos Ciclos de Vida apresentados por diferentes autores cada qual com diferentes números de fases e nomenclaturas. De modo geral, os sistemas de informação seguem um determinado ciclo de vida: concepção, construção, implantação, implementação, utilização plena, declínio, manutenção e morte [3]. A *Concepção* consiste no nascimento do sistema ou software; a *Construção* trata-se da análise e programação; na *Implantação* realizam-se os testes e disponibiliza-se o produto aos clientes; na *Implementação*, faz-se os ajustes necessários pós-implantação; a *Maturidade* e utilização plena é o momento onde extrai-se todo potencial do sistema desenvolvido; no *Declínio* o sistema passa a ter dificuldade de continuidade; a *Manutenção* vem para tentar manter o sistema ativo e por fim, a *Morte* que é a descontinuidade do sistema [3].

Neste estudo verificar-se-á, comparativamente, dois processos de desenvolvimento de sistemas de informação: RUP e XP.

2.1 RATIONAL UNIFIED PROCESS

2.1.1 Visão geral

Trata-se de um processo para engenharia de software. Fornece bases para a determinação de tarefas e responsabilidades para empresas de desenvolvimento de software [4].

Consiste em uma abordagem do ciclo de vida adequado a UML. A meta do RUP é permitir a produção de software da mais alta qualidade que atenda às necessidades do usuário final, de acordo com o planejamento e orçamentos previsíveis [5].

O RUP absorve várias das melhores práticas do desenvolvimento moderno de software adequado a uma variedade de projetos e organizações. Nesta situação podem ser apresentadas seis práticas [4]:

- Desenvolvimento iterativo do software;
- Gestão de requisitos;
- Desenvolvimento centrado na arquitetura;
- Modelagem visual do software;
- Verificação contínua da qualidade do software e
- Gestão de alterações do software.

O RUP possui as características necessárias para extrair da UML seu maior potencial. Ele é: Orientado a Casos de Uso, Centrado na Arquitetura e Iterativo/Incremental. Orientado a Casos de Uso significa que esses casos são utilizados como o principal artefato para o estabelecimento do comportamento desejado do sistema; Centrado na Arquitetura significa que a arquitetura do sistema é utilizada como principal artefato para a conceituação, construção, gerenciamento e a construção do sistema em desenvolvimento; Processo Iterativo é aquele que envolve o gerenciamento de seqüências de versões executáveis e, por fim, Processo Incremental é aquele que envolve a integração contínua da arquitetura do sistema para a produção dessas versões, de maneira que cada nova versão incorpora os aprimoramentos incrementais em relação às demais [5].

2.1.2 RUP & UML

A Unified Modeling Language (UML) é uma linguagem padrão para a elaboração da estrutura de projetos de software. UML pode ser empregada para a visualização, especificação, construção e documentação de artefatos que façam uso de sistemas complexos de software. É uma linguagem muito expressiva, abrangendo todas a visões necessárias ao desenvolvimento e implantação desses sistemas. A UML é apenas uma linguagem e, portanto, é somente uma parte de um método para desenvolvimento de software. A UML independe de processo, apesar de ser perfeitamente utilizada em processo orientado a Casos de Uso, centrada na arquitetura, iterativa e incremental [5].

O RUP é um processo de engenharia de software desenvolvido pelos três idealizadores da UML: Ivar Jacobson, James Rumbaugh e Grady Booch, sendo o resultado de mais de 30 anos de experiência acumulada. É o primeiro processo de desenvolvimento a explorar integralmente as capacidades do padrão UML [10].

2.1.3 Levantamento de Requisitos

O desafio no levantamento de requisitos é desenvolver modelos que expressem as reais necessidades do sistema em construção. Os Casos de Uso são modelos mais apropriados para tal tarefa, capturam requisitos funcionais naturalmente; também podem ser representados os requisitos não-funcionais que tenham alguma ligação no contexto do Caso de Uso. Para aqueles requisitos não-funcionais que não se encaixam no contexto dos Casos de Uso, existe ainda um documento chamado Requisitos Suplementares [11].

Os Casos de Uso levam a pensar em *Quem* é o usuário e *Qual* objetivo/missão que ele deve atender.

Para determinar os requisitos de um sistema utilizando os modelos de Caso de Uso deve-se: (fonte UML)

- Estabelecer o contexto do sistema, identificando os atores ao seu redor;
- Para cada ator, considera-se o comportamento de que se espera ou requer que o sistema proporcione;
- Nomeiam-se estes comportamentos comuns como Casos de Uso;

- Faz-se a fatoração do comportamento comum em novos Casos de Uso utilizados pelos outros; faz-se a fatoração do comportamento variante em novos Casos de Uso que estendem os fluxos da linha principal;
- Faz-se a modelagem desses Casos de Uso, atores e relacionamentos em um diagrama de Casos de Uso;
- Incluem-se adornos nesses Casos de Uso com notas declarando requisitos não-funcionais; poderá ser necessário anexar alguns deles a todo o sistema.

Com este trabalho realizado pode-se prosseguir para as próximas tarefas: análise e projeto, implementação e teste. Estes Casos de Uso conduzirão os trabalhos através destas tarefas iteração por iteração [11].

2.1.4 Modelagem e Documentação

A modelagem é uma parte central de todas atividades que levam à implantação de um bom software. Constroem-se modelos para comunicar a estrutura e o comportamento desejado do sistema; para visualizar e controlar a arquitetura do sistema; para compreender melhor o sistema em elaboração e por fim, para gerenciar riscos. (fonte UML)

O RUP é o processo. A modelagem utiliza uma linguagem: a UML. Ela é uma linguagem-padrão para elaboração da estrutura de projetos de software. Pode ser empregada para a visualização, especificação, construção e documentação de artefatos. (fonte UML)

A UML fornece alguns diagramas que são utilizados na modelagem dos sistemas. Um diagrama é a apresentação gráfica de um conjunto de elementos, geralmente representados como gráficos vértices (itens) e arcos (relacionamentos). São desenhados para permitir a visualização de um sistema sob diferentes perspectivas. A UML possui nove diagramas [5]:

- Diagrama de Classes;
- Diagrama de Objetos;
- Diagrama de Casos de Uso;
- Diagrama de Seqüência
- Diagrama de Colaboração;

- Diagrama de Gráficos de Estado;
- Diagrama de Atividades;
- Diagrama de Componentes;
- Diagrama de Implantação.

Neste estudo, para fins de comparação entre os dois processos, utilizar-se-á os diagramas de Classes, Casos de Uso e Seqüência.

Um *diagrama de Classe* exibe conjunto de classe, interfaces e colaborações, bem como seus relacionamentos. Abrangem uma visão estática da estrutura do sistema [5].

Um *diagrama de Casos de Uso* exibe um conjunto de caso de uso e atores (um tipo especial de classe) e seus relacionamentos. São importantes para a determinação e modelagem do comportamento do sistema e abrangem uma visão estática dele [5].

O *diagrama de Seqüência* e diagrama de Colaboração fazem parte dos diagramas de Iterações, consistindo de um conjunto de objetos e seus relacionamentos, incluindo as mensagens que podem ser trocadas entre eles. O *Diagrama de Seqüência* enfatiza a ordenação temporal das mensagens [5].

Todos estes diagramas supracitados constituem a modelagem do sistema, ou seja, a utilização da linguagem UML para definição do projeto do software. Porém, o RUP abrange diversas outras tarefas que por sua vez necessitam de controle e documentação. Para isto existem modelos (artefatos) específicos para ele. Alguns são derivados da UML outros são mais específicos para controle do projeto. Existem nove modelos para o RUP [5]:

- Modelo de Negócio;
- Modelo de Domínio;
- Modelo de Caso de Uso;
- Modelo de Análise (opcional);
- Modelo de Projeto
- Modelo de Processo (opcional);
- Modelo de Implantação;
- Modelo de Implementação e
- Modelo de Teste.

2.1.5 Ciclo de Vida

O RUP é um sistema Orientado a Caso de Uso, Centrado na Arquitetura e Iterativo/Incremental. Este processo pode ser desmembrado em Fases. Existem quatro fases no Ciclo de Desenvolvimento: Iniciação ou Concepção, Elaboração, Construção e Transição. A Figura 1 apresenta o Ciclo de Vida do RUP. Abaixo seguem a conceituação de Fase, Iterações e Ciclo de Desenvolvimento.

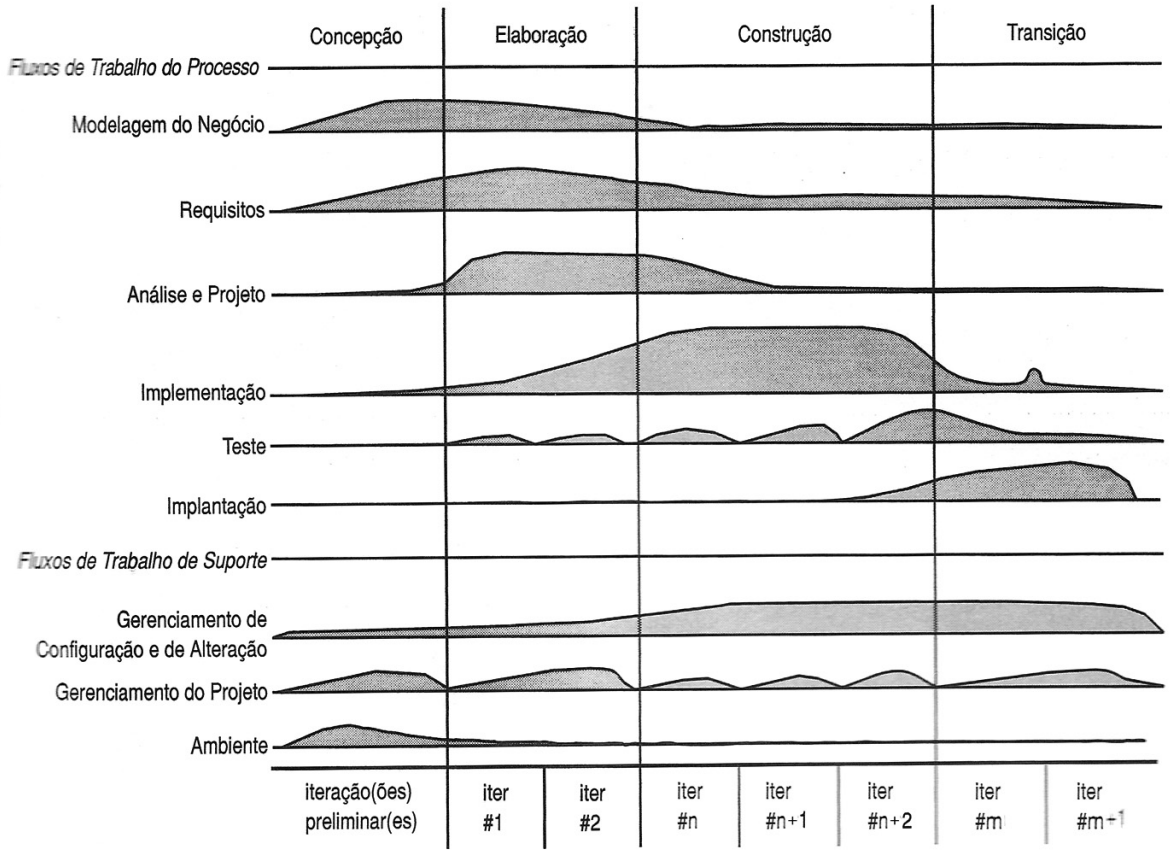


Figura 1 – Ciclo de Vida do RUP.

Fases:

Uma fase é o período de tempo entre dois importantes marcos do progresso do processo em que um conjunto bem-definido de objetivos é alcançado, artefatos são concluídos e decisões são tomadas em relação à passagem para a fase seguinte. O RUP é composto por quatro fases [5]:

1. Concepção: ocorre a definição do caso de negócio para o sistema e delimita o escopo do projeto. No final da fase de concepção, examinam-se os

objetivos do ciclo de vida do projeto e decide-se se deve prosseguir com o desenvolvimento em plena escala.

2. Elaboração: as metas da fase de elaboração são a análise do domínio do problema, o estabelecimento da fundação de uma arquitetura sólida, o desenvolvimento do plano do projeto e a eliminação dos elementos de mais alto risco do projeto. As decisões de arquitetura devem ser feitas com uma compreensão de todo o sistema.
3. Construção: durante a fase de construção, desenvolve-se, de forma iterativa e incremental, um produto completo, pronto para a transição à sua comunidade de usuários. No final da fase de construção, decide-se se o software, ambientes e usuários estão prontos para se tornarem operacionais.
4. Transição: durante a fase de transição o software torna-se disponível a comunidade de usuários. Após o sistema ser colocado nas mãos de seus usuários finais, sempre surgem questões que requerem algum desenvolvimento adicional, com a finalidade de ajustar o sistema, corrigir alguns problemas identificados ou concluir algumas características propostas no início.

Iterações:

A iteração é um ciclo completo de desenvolvimento, resultando em uma versão (interna ou externa) de um produto executável que constitui um subconjunto do produto final em desenvolvimento e cresce de modo incremental de uma iteração para outra para se tornar o sistema final [5].

Ciclos de Desenvolvimento:

Uma passagem pelas quatro principais fases chama-se Ciclo de Desenvolvimento. Um sistema pode possuir apenas um ciclo (passar uma vez pelas fases concepção, elaboração, construção e transição) ou mais de um, quando passa a ser chamado de ciclos de evolução [5].

2.1.6 Testes

Trata-se da verificação do produto da implementação. Especificamente os testes atendem aos seguintes propósitos: planejar testes para identificar as demandas do sistema bem como a integração dele; desenhar e implementar testes que possam ser executados automaticamente; e realizar testes para identificar possíveis erros para encaminhamento da correção deles [11].

Dentro do ciclo de vida do projeto os testes podem ocorrer na fase de concepção no momento da definição do escopo do projeto; ocorrem com uma freqüência significativa nas fases de Elaboração e Construção; e na fase de Transição a fim de identificar falhas no sistema.

O Modelo de Teste deve ser planejado para executar os testes automaticamente utilizando a ajuda de sistemas para este fim. O Modelo de Teste pode possuir os seguintes artefatos: Caso de Teste, Procedimento de Teste e Componente de Teste.

Caso de Teste: especifica um caminho para realizar um teste do sistema. Considera as entradas, saídas e condições de teste.

Procedimento de Teste: especifica como proceder com um ou mais Caso de Teste ou partes dele. Os Procedimentos de Teste podem ser utilizados para outros Casos de Teste semelhantes onde valores de entrada e saídas diferem.

Componente de Teste: automatiza um ou vários Procedimentos de Teste ou ainda, partes deles. Os Componentes de Teste podem ser desenvolvidos utilizando linguagens Scripts, linguagens de programação ou ainda utilizando Sistemas de Automação de Testes.

2.2 EXTREME PROGRAMMING

2.2.1 Visão Geral

Em fevereiro de 2001 um grupo de dezessete estudiosos da área de softwares reuniram-se em Snowbird, UT, EUA, para discutir o crescimento dos assim chamados métodos peso-leve. Ficou decidido que usariam o termo ágil para descrever esta nova geração de métodos. Na mesma ocasião foi escrito o *Manifesto por um Desenvolvimento Ágil de Software*, definindo os valores e princípios dos processos ágeis [6].

Os princípios definidos no manifesto foram [7]:

1. A prioridade é satisfazer o consumidor através de entregas rápidas e contínuas do produto de software;
2. Receber bem mudanças nos requisitos, mesmo com o desenvolvimento em andamento. Processos Ágeis mudam conforme as necessidades de alcançar vantagens competitivas para os clientes;
3. Entregar o software freqüentemente, entre algumas semanas a poucos meses. Com uma preferência de um *timescale* mais curto;
4. Desenvolvedores e usuários finais devem trabalhar juntos diariamente durante todo o projeto.
5. Construir projetos próximos de indivíduos motivados. Fornecer a estas pessoas um ambiente que elas necessitam pra trabalhar e confiar que elas concluirão o trabalho.
6. O método mais eficiente e eficaz de tornar fluente a informação dentro de um grupo de desenvolvimento é a conversação cara-a-cara.
7. Avalia-se o progresso, preliminarmente, através do trabalho de produção de software.
8. Processos Ágeis promovem um desenvolvimento sustentável. Os patrocinadores e os desenvolvedores devem manter constantemente as medições.
9. Manter a atenção na excelência técnica e *design* para aumentar a agilidade.
10. Simplicidade: a arte de maximizar o aumento de trabalho não feito.
11. As melhores arquiteturas, requerimentos e *designs* surgem a partir de grupos bem organizados internamente.

12. Em intervalos regulares, a equipe deve refletir como pode se tornar mais eficaz, e assim, fazer os devidos ajustes no comportamento da equipe.

A XP segue algumas práticas que podem ser definidas da seguinte forma:

1. Jogo do planejamento: trata-se de um diálogo evolutivo sobre o que é possível fazer e o que é desejável. Existem dois tipos de decisões a serem tomadas: decisões de negócios e técnicas. As de negócios definem o escopo do projeto, prioridades, composição das versões e data de entrega. As técnicas, que antecedem as decisões de negócios, são as estimativas, conseqüências, processos (de trabalho utilizado) e cronograma detalhado.
2. Entregas freqüentes: prioridade é entregar cada vez mais rápidas funcionalidades que tragam maior valor ao sistema.
3. Metáfora: cada projeto XP é guiado por uma metáfora abrangente. A metáfora serve para apresentar uma visão para todos os envolvidos no projeto. A metáfora substitui o conceito de “Arquitetura” do sistema.
4. Projeto Simples: procura-se desenvolver um projeto da seguinte forma: que execute todos os testes; que não tenha lógica duplicada; que expresse todas as intenções importantes para os programadores e tenha o menor número possível de classes e objetos. Sendo assim, pode-se chamar de um projeto simples.
5. Testes: são sempre automatizados. Podem ser Testes de Unidade – escritos pelos programadores – e Testes de Funcionalidades – escritos principalmente pelos clientes.
6. Refatoração: trata-se de desenvolver códigos que podem ser alterados mantendo os testes em execução. A refatoração é utilizada para simplificar o código. A refatoração ocorre voluntariamente – onde demanda esforço do programador, porém evita aborrecimentos futuros – ou por necessidade – quando o sistema passa por problemas de código, ou duplicações, entre outros.
7. Programação em Pares: significa uma pessoa programando e outra, ao lado, analisando estrategicamente o trabalho. Estes pares são dinâmicos, variam de acordo com a necessidade.

8. Propriedade Coletiva: surgiu para resolver o problema tanto da propriedade individual quanto da ausência de propriedade. Na primeira, o problema era a concentração do conhecimento do código na mão de poucas pessoas e, por conseguinte a evolução lenta do código; na segunda, o código evoluía rapidamente, porém, devido a diversas modificações, tornava-se instável. Com a propriedade coletiva, a responsabilidade de melhorar o código passa a ser de todos. O sistema é responsabilidade de todos. Quando se verifica uma oportunidade de melhoria do código, deve-se fazer imediatamente.
9. Integração Contínua: integra-se o código constantemente, de preferência num período que vai de horas a no máximo um dia de desenvolvimento. Faz-se a integração por par de programadores. Ao efetuar a integração, os testes revelarão a situação atual. Se os testes não chegarem a 100% deve-se retomar o desenvolvimento para posterior integração.
10. Semana de 40 horas: pretende-se organizar a carga de trabalho para que não seja necessário fazer horas extras. Supõe-se que horas extras sejam necessárias esporadicamente. No momento em que estas começam a ser solicitadas freqüentemente, significa a existência de algum problema maior no projeto.
11. Cliente presente: um cliente real deve ficar com a equipe, estar disponível para resolver questões, resolver disputas e definir prioridades de menor escala. Cliente real significa o verdadeiro cliente do sistema. Aquela pessoa que utilizará o produto final do desenvolvimento. Porém, disponibilizar pessoas que devem estar trabalhando na sua atividade fim nem sempre é possível. Deve-se buscar o entendimento (equilíbrio) para conseguir disponibilizar clientes reais o máximo possível para trabalhar com os programadores.
12. Padrões de Codificação: deve-se adotar um padrão de codificação devido ao fato de muitas pessoas estarem constantemente trabalhando nele. Regras fáceis, que facilitem a comunicação e que possa ser adotado voluntariamente por todo a equipe.

2.2.2 Levantamento de Requisitos e Modelagem

Dentro do ciclo de vida de um projeto ágil podem existir dois tipos de levantamento de requisitos: modelagem de requisitos iniciais e modelagem em tempo de desenvolvimento [9].

A primeira refere-se aos requisitos iniciais, bem como a definição do escopo geral do projeto e o que o sistema realmente deverá fazer. Nesta atividade, procura-se visualizar os principais requisitos do projeto o que pode levar até dias. A documentação não é exigida, pode ser desenvolvido posteriormente, caso necessário. Para realizar este trabalho apresenta-se três ferramentas: *User Stories, Inicial Domain Model e User Interface Model*. O primeiro, são cartões onde são descritos os principais requisitos que o programador precisa para estimar um tempo de desenvolvimento. O segundo, identifica as principais regras do negócio, entidades e relacionamentos, utiliza-se os cartões CRC (*Class Responsability Collaborator cards*). Por fim, o modelo de interface gráfica através de esboços sugeridos pelos clientes.

A segunda trata-se de levantamento de requisitos que devem ser visualizados dentro de uma iteração. Cada iteração sugere um levantamento de requisitos específico para determinada tarefa. Entende-se que visualizando o problema a fundo torna-se mais fácil o entendimento do problema e, por conseguinte mais preciso a descrição dos requisitos. Para realizar tal tarefa o programador (ou a dupla de programadores) deve entrevistar os clientes. Estas entrevistas são rápidas e pontuais e são chamadas de *Model storming sessions*.

O levantamento de requisitos está diretamente ligado ao usuário do sistema. Pessoas que realmente conhecem o funcionamento da empresa/sistema. Os programadores são os facilitadores. Esses sugerem soluções, instigam os usuários a pensar/re-pensar o problema e a solução.

Existem diversos artefatos que podem ser gerados no levantamento de requisitos, cada qual num determinado momento do desenvolvimento como citado anteriormente. Tais artefatos são descritos abaixo:

User Stories: São cartões que determinam as funcionalidades do sistema e processo da atividade em questão. Trata-se de um cartão onde se registram apenas as informações necessárias para o programador estimar o tempo de desenvolvimento. As informações são escritas pelos clientes. Possui um número de

identificação, um título da atividade que representa, a descrição na forma de “passos”, a prioridade e o tempo estimado (o qual será determinado pelo programador). A apresentação destes cartões pode vir de várias formas, na figura 2 o proposto por Kent Beck e na figura 3 o proposto por Scot Ambler.

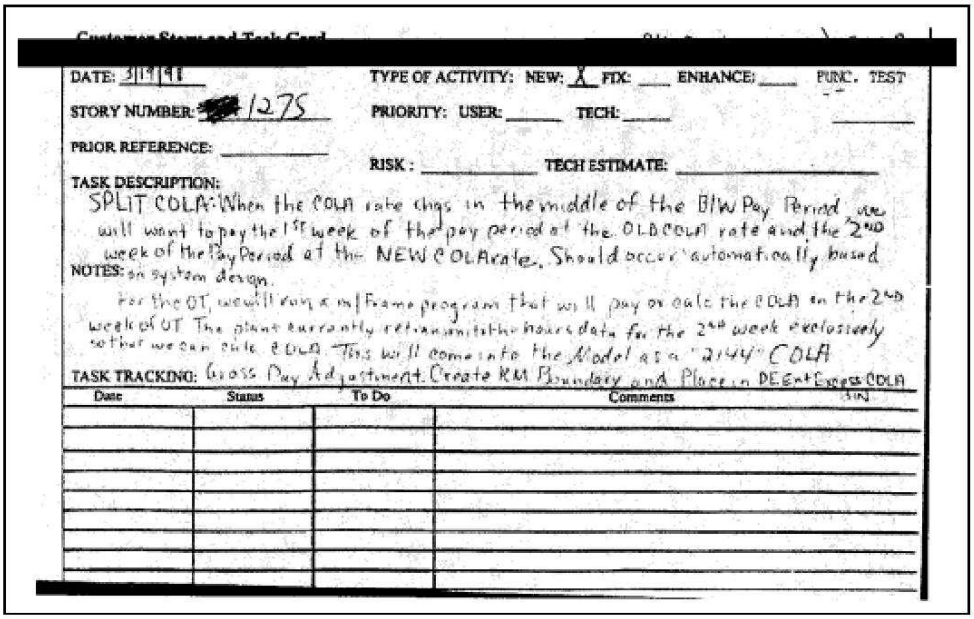


Figura 2 – User Stories proposto por Kent Beck.

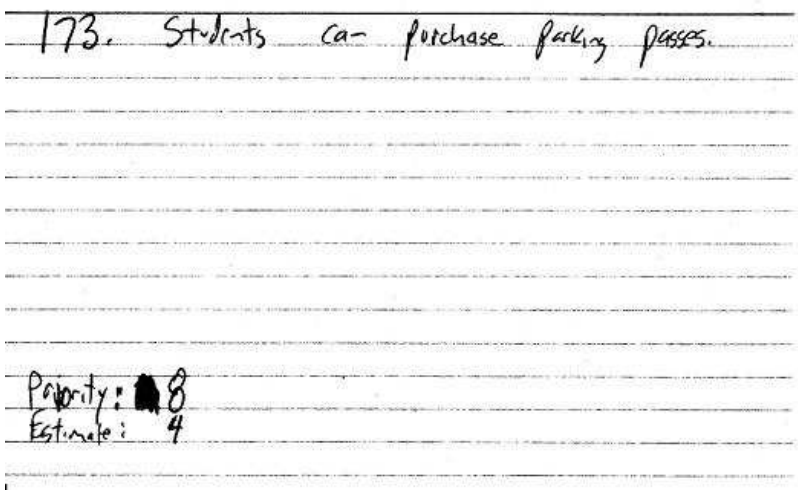


Figura 3 – User Stories proposto por Scot Ambler

A partir de uma lista de User Stories com tempo estimado e prioridade definida, o cliente determina a ordem de desenvolvimento de acordo com sua necessidade.

Class Responsability Collaborator cards: trata-se de um modelo que apresenta a visão geral do sistema, suas entidades e relacionamentos. O CRC model é composto por diversos CRC cards, esses são divididos em três partes: Nome da Classe, Responsabilidades e Colaboradores. Esta técnica utilizada antigamente para ensino dos conceitos de orientação a objetos foi bem aceita na modelagem de requisitos em XP devido à participação dos Usuários na elaboração do projeto. Na figura 4, visualiza-se um CRC card e na figura 5, um CRC model.

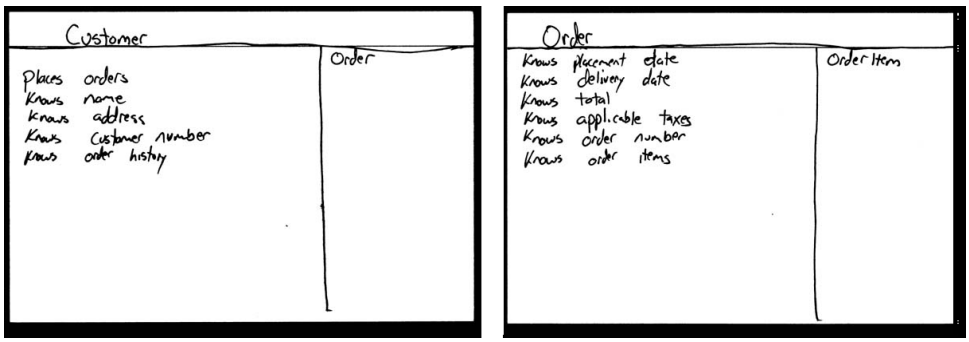


Figura 4 – Exemplos de CRC cards.

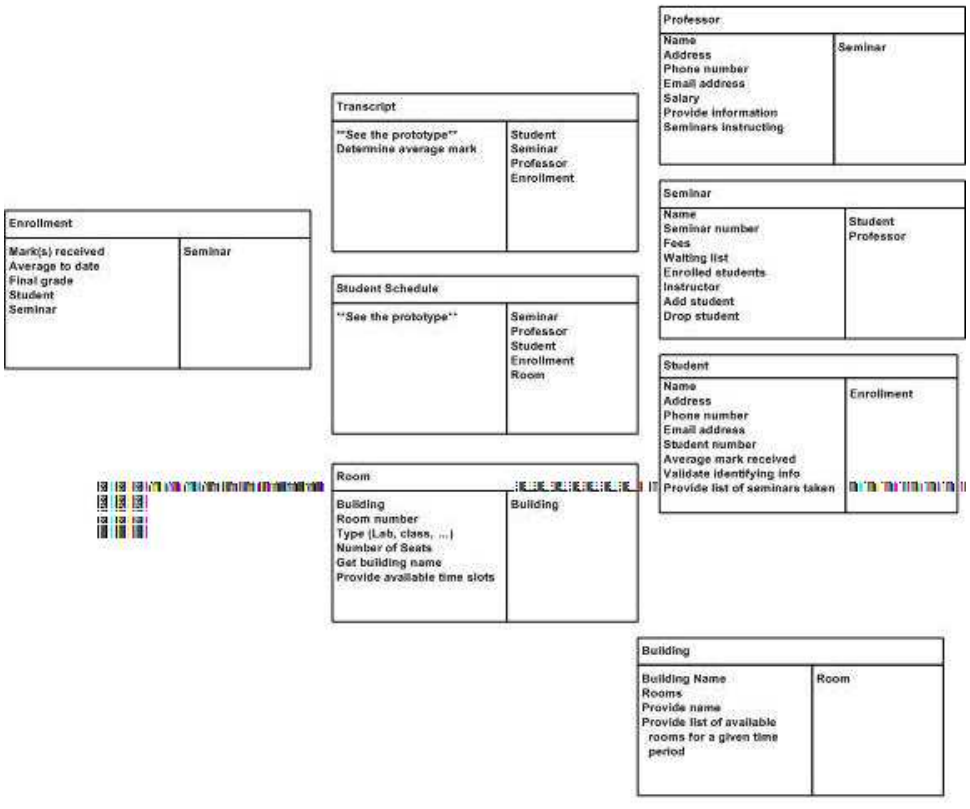


Figura 5 – Exemplo de CRC model.

User Interface Prototype: a prototipagem de interfaces gráficas permite instigar o usuário a simular como o sistema deve parecer. Essas facilitam a comunicação entre programador e usuário do sistema entre outros benefícios. O processo da prototipagem possui quatro passos: determinação das necessidades, construção do protótipo, desenvolvimento do protótipo e teste de aceite, como mostra a figura 6.

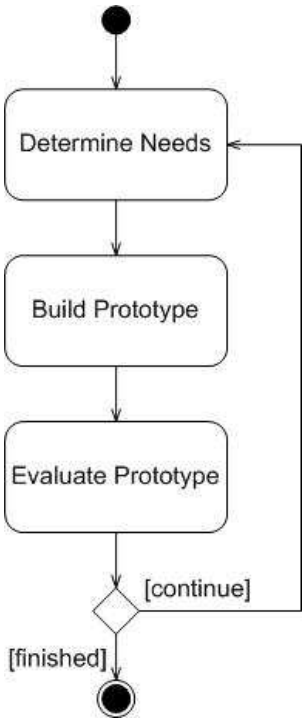


Figura 6 – Processo da prototipagem.

Inicialmente, os protótipos podem ser esboçados num papel para posterior desenvolvimento. Na figura 7, apresenta-se um esboço de uma interface sugerida pelo cliente. Após este esboço, tornam-se claras as necessidades de interfaces do cliente. Assim, segue-se para o momento de construção do protótipo em linguagem de alto nível escolhida para o desenvolvimento do projeto. Na figura 8 apresenta-se um exemplo de protótipo de interface de usuário.

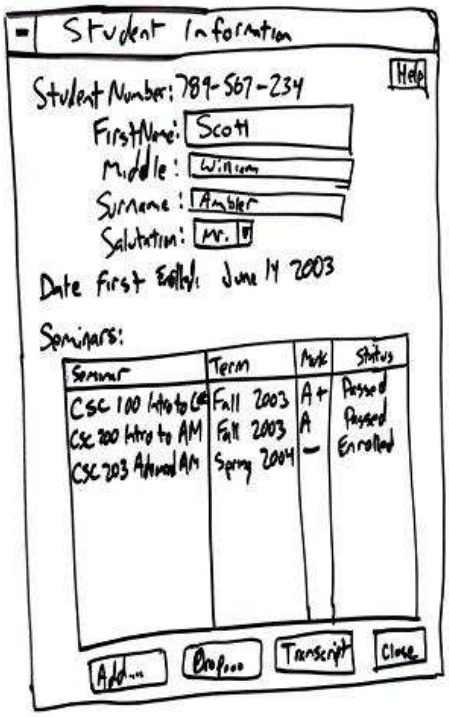


Figura 7 – Exemplo de um esboço de uma interface.

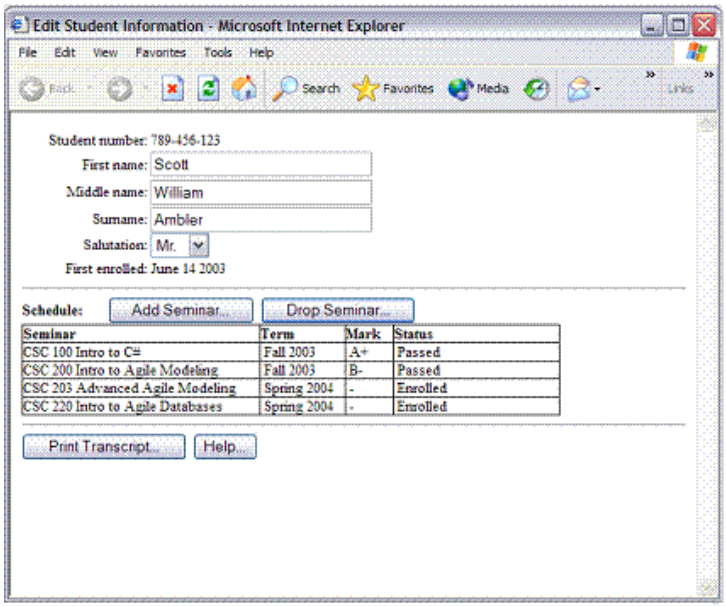


Figura 8 – Exemplo de um protótipo de uma interface.

O terceiro passo, evolução do protótipo, permite ao cliente verificar se o resultado é satisfatório, fazer sugestões e incluir/excluir propriedades. Este passo pode ser extremamente rápido. Após este passo a interface passa por um teste de aceite.

2.2.3 Ciclo de Vida

O Ciclo de Vida da XP é apresentada em seis fases: Exploração, Planejamento, Iterações para a primeira Entrega, Produção, Manutenção e Morte [8]:

1. Exploração: para iniciar a fase de produção da equipe precisa estar preparado para isto: é preciso acreditar que conseguirá terminar o programa. Nesta fase, a equipe de desenvolvimento testa as tecnologias, arquiteturas, limites de performance etc que poderão ser utilizadas na produção enquanto o cliente define os *Cartões de Histórias*: documentos escritos em papel que expressam as funcionalidades do sistema do ponto de vista dele. Com esta prática será possível estimar o tempo para desenvolvimento de cada tarefa, aumentando a confiança da equipe.
2. Planejamento: trata-se de fazer os clientes e os programadores chegarem a um acordo sobre a data na qual o menor e mais valioso conjunto de histórias estará terminado. O produto desta fase é um cronograma de comprometimento. Se a Exploração tiver sido bem feita, provavelmente, esta fase demandará pouco tempo para ser concluída.
3. Iterações para a Primeira Entrega: O cronograma de comprometimento será dividido em iterações. A primeira iteração define a arquitetura do sistema. Para as demais iterações deve escolher as historias que definam o sistema como um todo, mesmo que ao final produza-se um arcabouço do sistema. De preferência, ao final de uma iteração o cliente já terá realizado um teste de funcionalidade e poderá executar o sistema. Ao fim da última iteração o projeto estará pronto para entrar em produção.
4. Produção: Ao entrar em produção o sistema passa a ser conduzido com um feedback de menor tempo. O tempo das iterações diminuirá. As estimativas de tempo serão mais precisas. O tempo de evolução do software diminuirá.
5. Manutenção: Trata-se do estado normal de um projeto XP. Simultaneamente produzem-se novas funcionalidades, mantém-se o sistema existente em execução e incorporam-se novas pessoas à equipe. Neste momento, o sistema passa por refatorações, experimentam-se novas arquiteturas, os clientes criam novas histórias e surgem novas versões.
6. Morte: Existem, pelo menos, duas possibilidades de morte do sistema: a primeira, quando o cliente não consegue mais produzir histórias que tornem

o sistema melhor. A segunda, no caso do sistema não agregar mais valor ao negócio do cliente de uma forma economicamente viável. Para efetuar uma morte bem conduzida, gera-se um roteiro do sistema para eventuais consultas futuras.

2.2.4 Testes

Considerando teste em XP verifica-se que estes são Isolados e Automáticos. Isolados, pois não interagem com outros para que no caso de um erro, este não ocasione erros sucessivos. Testes automáticos visam apresentar resultados não-qualitativos do comportamento.

Neste sentido deve-se realizar teste do que realmente importa, ou seja, não é necessário testar tudo em um sistema. Um bom teste é aquele que contraria as expectativas: quando se espera o correto funcionamento e ele falha; quando se espera que ele falhe e o funcionamento ocorre corretamente [8].

Os testes devem ser escritos pelos programadores e pelos clientes da seguinte forma: os programadores escrevem testes por métodos e os clientes escrevem os testes por história.

Os programadores escrevem testes nas seguintes circunstâncias:

- Se a interface de um método não está muito clara, escreve-se um teste antes de escrever o método;
- Se a interface é clara, mas imagina-se que a implementação será um pouco complicada, escreve-se um teste antes de escrever o método;
- Caso existam circunstâncias inusitadas onde o código deve funcionar como esperado, escreve-se um teste para comunicar as circunstâncias;
- Caso um problema seja descoberto depois, escreve-se um teste que isole o problema;
- Caso esteja prestes a refatorar o código, não saiba-se ao certo como ele irá se comportar, e não exista teste para o aspecto em questão, escreve-se um teste primeiro;

Os clientes escrevem testes por histórias. Estes verificam o que deve ser feito para determinada história fazer o planejado.

São dois os tipos de testes: os de unidades e os funcionais. Os testes de unidade são escritos pelos programadores e sempre se executam na sua plenitude. Testes funcionais devem ser escritos pelos clientes, porém estes não conseguem escrevê-los sozinhos.

Os testes de unidade verificam o código e devem ser executados na sua plenitude. Ao ocorrer uma falha, foca-se na resolução desta para poder seguir o trabalho do desenvolvimento.

Os testes funcionais não se executam na sua plenitude. São mensurados por porcentagens. Ao longo do tempo vão evoluindo e se aproximando dos 100% .

2.2.5 Documentação

A XP tem por objetivo fazer entregas rápidas e contínuas do produto de software. Isto pode sugerir que não exista documentação no processo. Os idealizadores do XP alertam para este mal-entendido.

A documentação existe na seguinte forma: na primeira iteração, com as Users Stories, CRC cards/model, Interfaces Model; na demais iterações, na forma de comentários dentro do código fonte e documentos gerados pelas reuniões com os clientes; e por fim, quando ocorre a chamada Morte do sistema, faz-se um documento compilando todos aspectos envolvidos nele para, se necessária, posterior consulta.

3 COMPARAÇÃO ENTRE RUP E XP

Neste estudo, a comparação entre o RUP e XP se dá nos aspectos do ciclo de vida de cada processo. Nesta análise pode-se verificar quais e qual o momento em que os artefatos são gerados. Inicialmente, faz-se uma comparação entre os ciclos de vida dos dois processos e por fim, compara-se as fases que se correspondem funcionalmente.

3.1 COMPARAÇÃO DO CICLO DE VIDA

Ciclo de Vida de um processo de engenharia de software é a estrutura que abrange toda o projeto de um sistema de informação. Genericamente, contempla diversas fases como levantamento de requisitos, análise, desenvolvimento, testes, implantação etc. Existem diversas teorias que apresentam ciclos de vida. Cada qual tem uma particularidade, porém todas visam o mesmo objetivo: entrega de um sistema de informação que atenda as necessidades do patrocinador dentro do prazo e orçamento estimado.

Neste estudo, abordamos o RUP e XP. Cada um deles possui determinado número de fases que representam atividades dentro do processo de construção do sistema. Alguns têm mais fases que outros. Neste caso, precisamente, o RUP apresenta quatro fases: Concepção, Elaboração, Construção e Transição; já o XP apresenta seis fases: Exploração, Planejamento, Iterações para a primeira entrega, produção, manutenção e morte.

O RUP e o XP são classificados como processos Iterativos e Incrementais. Isto significa que o entendimento da solução evolui com o passar do tempo, as funções do sistema são adquiridas com o aumento deste entendimento e o sistema vai-se tornando cada vez melhor.

Mesmo os dois modelos sendo Iterativos e Incrementais, deve-se observar a diferença de como são aplicados estes conceitos em cada processo: No RUP, cada iteração demanda um ciclo completo de desenvolvimento; No XP, as iterações são feitas dentro do ciclo geral do projeto. A figuras 9 e 10 apresentam o ciclo de vida do RUP e XP respectivamente.

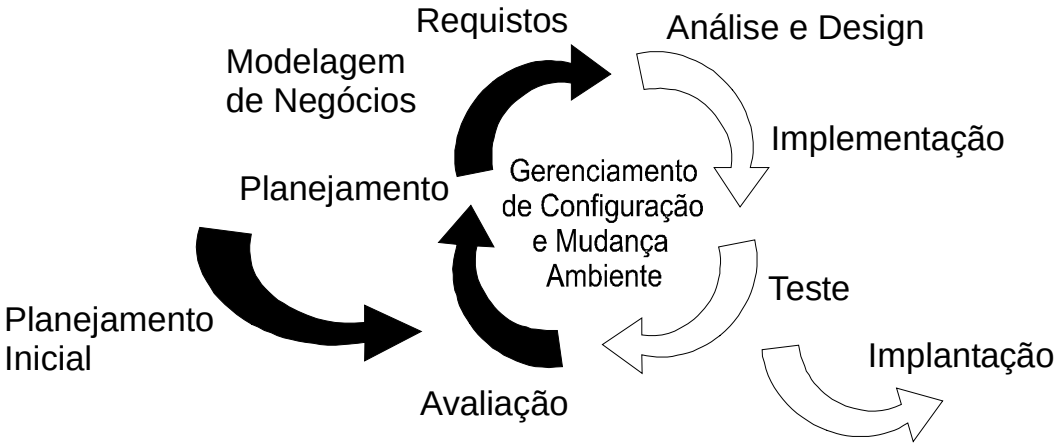


Figura 9 – Processo de Engenharia de Software na visão do RUP.

A comparação entre os dois ciclos de vida será efetuada verificando-se as fases correspondentes em cada um deles. No RUP são quatro fases: Concepção, Elaboração, Construção e Transição. O XP apresenta seis fases: Exploração, Planejamento, Iterações para a primeira entrega, produção, manutenção e morte.

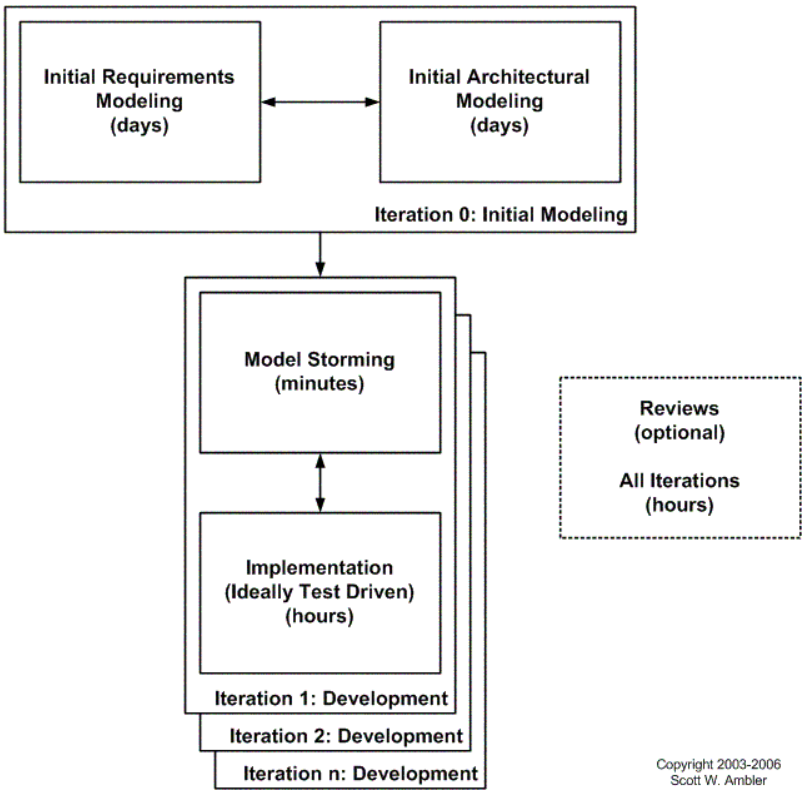


Figura 10 – Ciclo de Vida XP.

Considerando as fases do RUP já apresentadas, existem nove Disciplinas que são as tarefas no ciclo de vida do desenvolvimento do projeto, são elas: modelagem do negócio, requisitos, análise e design, implementação, teste, implantação, gerência de configuração e mudança, gerenciamento do projeto e ambiente. De acordo com o foco deste trabalho, utilizaremos as três primeiras disciplinas: modelagem do negócio, requisitos e análise/design.

3.2 COMPARAÇÃO DAS FASES DO CICLO DE VIDA

Para efetuar a comparação por fase, procurar-se-á identificar na disciplina em questão do RUP a fase correspondente no XP. Devido à arquitetura de cada ciclo, idealizada por seus criadores, uma tarefa pode ser encontrada em mais de uma fase ou vice-versa. A partir do estudo, verificou-se uma semelhança funcional entre as seguintes fases/tarefas/disciplinas em questão como apresentado na tabela 1.

Tabela 1 – Relações entre fases do RUP e XP.

	Fase	
	RUP	**XP**
Visualização do negócio e objetivos do projeto	Modelagem do Negócio	Exploração – metáfora; Planejamento
Levantamento de Requisitos	Requisitos – Use Cases	Exploração – User Stories
Análise	Análise e Design – Modelo de Análise: Diagramas de Classe, Iterações, etc.	Exploração – User Stories, Inicial Domain Model e User Interface Model; Iterações para Primeira Entrega

Visualização do Negócio e Objetivos do Projeto/Sistema:

Para visualizar o negócio e os objetivos do futuro sistema, o RUP apresenta uma disciplina denominada Modelagem do Negócio e o XP por sua vez compreende esta tarefa na fase de Exploração, no que se refere à determinação da metáfora e na fase de Planejamento.

O RUP sugere a criação de um Modelo de Negócio para o entendimento geral do sistema que se pretende desenvolver. Tal modelo é um diagrama de Caso de Uso abrangente que representa graficamente o negócio. Utilizando a notação da UML, pode-se obter uma visão dos Atores envolvidos tanto interna/externamente ao sistema. Este modelo inclui diversos outros Use-cases que serão aprofundados na fase de Levantamento de Requisitos. Este artefato gerado norteia as demais disciplinas do processo.

No XP a tarefa da definição do objetivo central do projeto é determinada na fase de exploração, definindo a Metáfora, ela pode ser descrita de forma textual. A metáfora é a forma pela qual toda a equipe entenderá o sentido do projeto como um todo.

Levantamento de Requisitos:

O levantamento de requisitos no RUP é feito a partir da criação de Casos de Uso. No XP é determinado por User Stories durante a fase de Exploração. Tanto os Casos de Uso quanto os User Stories são bem claros e explicitam bem os processos, requisitos e atores envolvidos.

Neste ponto apresenta-se uma importante diferença entre os dois processos de engenharia: No XP, os User Stories, como o nome já sugere, é feito pelo próprio usuário do sistema atual, ao contrário do RUP, deve ser realizado por uma pessoa treinada em UML.

A partir dos User Stories é que os programadores farão a estimativa de tempo para desenvolvimento da funcionalidade. Ao mesmo tempo que o usuário do sistema é a pessoa mais indicada para sugerir as funcionalidades necessárias, este pode errar muito mais do que uma pessoa especialista em levantamento de requisitos. O

nível de detalhe da descrição da História do Usuário depende de pessoa para pessoa e caso ocorra um problema, este será detectado mais adiante no processo.

Análise:

A análise no RUP gera um documento chamado de Modelo de Análise que é composto pelos diagramas da UML. Neste estudo foram utilizados os Diagramas de Casos de Uso, Classe e Seqüência.

No XP, existe a modelagem no momento da Iteração 0 (zero) e a modelagem dentro das demais iterações.

Os artefatos gerados na primeira iteração (zero) são os User Stories, criados pelos clientes na fase de exploração, o Inicial Domain Model e o User Interface Model. O Inicial Domain Model é criado a partir dos CRC cards e geram um modelo CRC model: com aparência e funcionalidade muito próxima de um diagrama de classes.

Ao passar para as demais iterações, ao invés de fazer uso de User Stories, os programadores fazem uso dos Model Storming Sessions: reuniões/entrevistas onde os programadores interagem diretamente com o usuário questionando e aguardando sugestões para problemas ou inovações. Neste momento, pode-se gerar mais User Stories ou uma descrição textual da reunião para efetuar a documentação e posterior consulta, se necessário.

4 ESTUDO DE CASO

A fim de visualizar na prática os dois processos em discussão e buscar um entendimento maior dos pontos fortes e fracos do RUP e XP, bem como a relação que existe entre eles do ponto de vista da modelagem, faz-se uso da aplicação do conhecimento sedimentado neste estudo através de um exemplo de um sistema empresarial.

O estudo foi dividido em três partes: apresentação informal do sistema-exemplo, sua problemática e escopo básico; a aplicação das técnicas do RUP e a aplicação das técnicas do XP.

4.1 SISTEMA – SUB-SISTEMA DE CONTROLE DE LICITAÇÕES

Para realizar o estudo de caso escolheu-se modelar um sistema de gerenciamento de empresas de construção civil. No âmbito geral o sistema deve ter cadastros de todas as pessoas envolvidas com a operação da empresa: clientes, fornecedores, funcionários e etc; sub-sistema de controle de operações financeiras e compatibilidade com sistemas de contabilidade; sub-sistema de controle de Projetos e execução; sub-sistema de controle de licitações e orçamentos entre outros. Dentre as inúmeras funcionalidades que o sistema deve apresentar optou-se, a fim de focar o estudo, pelo processo de controle de licitações e orçamentos.

O processo atual que envolve o controle de Licitações funciona da seguinte maneira:

- Uma empresa especializada em descobrir editais compila um e-mail e envia para a construtora uma lista de licitações;
- Diariamente, confere-se este e-mail e selecionam-se as licitações que interessam para a empresa de acordo com critérios como valor da obra, cidade, know-how, entre outros;
- Caso a licitação interesse procura-se o edital de abertura do processo que contém a documentação para participação da concorrência pública;
- Neste momento o edital pode estar disponível numa página da web ou ainda ser enviado por e-mail para a empresa. Em alguns casos exige-se que os

interessados em participar da concorrência se apresentem pessoalmente para pegar uma cópia deste material;

- Com o edital na empresa, buscam-se informações das datas da concorrência, da organização da proposta e da documentação exigida para participar do processo de licitação;
- Faz-se uma avaliação detalhada da obra: analisa-se se é um bom negócio financeiramente e analisa se o Know-how é compatível com a exigência da obra;
- Agenda-se a data da concorrência;
- Abre-se uma pasta para arquivar a documentação exigida (cópias, etc);
- Busca-se no edital as listas e material, plantas, cronogramas e etc para efetuar o Orçamento da obra;
- Buscam-se os documentos exigidos e coloca-se em uma pasta;
- Compila-se o material para participar da licitação: envelope com a proposta e outro com a documentação exigida;
- Se a empresa vencer a concorrência abre-se um projeto no cronograma de obras da empresa. Caso perca arquiva-se toda documentação no arquivo histórico da empresa;

Deste breve relato do processo de licitação da empresa verificou-se a necessidade de ter um sistema que organize estas informações de uma forma que possibilite a fácil recuperação desta. Este subsistema faz parte do Core Business.

4.2 APLICAÇÃO – RUP

Para aplicar na prática o RUP, tratou-se o sub-sistema de controle de licitações como um projeto integral, então utilizou-se ele como a visão geral do sistema (Modelagem do Negócio), os Casos de Uso foram criados a partir da descrição textual do processo e os diagramas de Classe e Seqüência que constituem o Diagrama de Análise, utilizaram alguns dados pertinentes para elucidação do problema.

Para o desenvolvimento do projeto, foi utilizada uma ferramenta CASE (Computer Aided Software Engineering – Engenharia de Software Apoiada por Computador) Rational Rose 98. Tal ferramenta possibilita a criação dos diagramas e

o armazenamento dos modelos de forma magnética. O projeto obtido como resultado da modelagem está contido no anexo B do trabalho. Abaixo, faz-se comentários a respeito dos modelos obtidos:

Modelo de Negócios e Casos de Uso: o modelo de negócios apresenta uma visão geral de todos Casos de Usos que deverão ser desenvolvidos no sistema (neste caso um Sub-Sistema). Neste exemplo pode-se observar inúmeros Casos de uso que no decorrer do projeto são explorados e observados seus funcionamentos. A Figura 11 apresenta o Modelo de Negócios e os Casos de Uso listados no projeto.

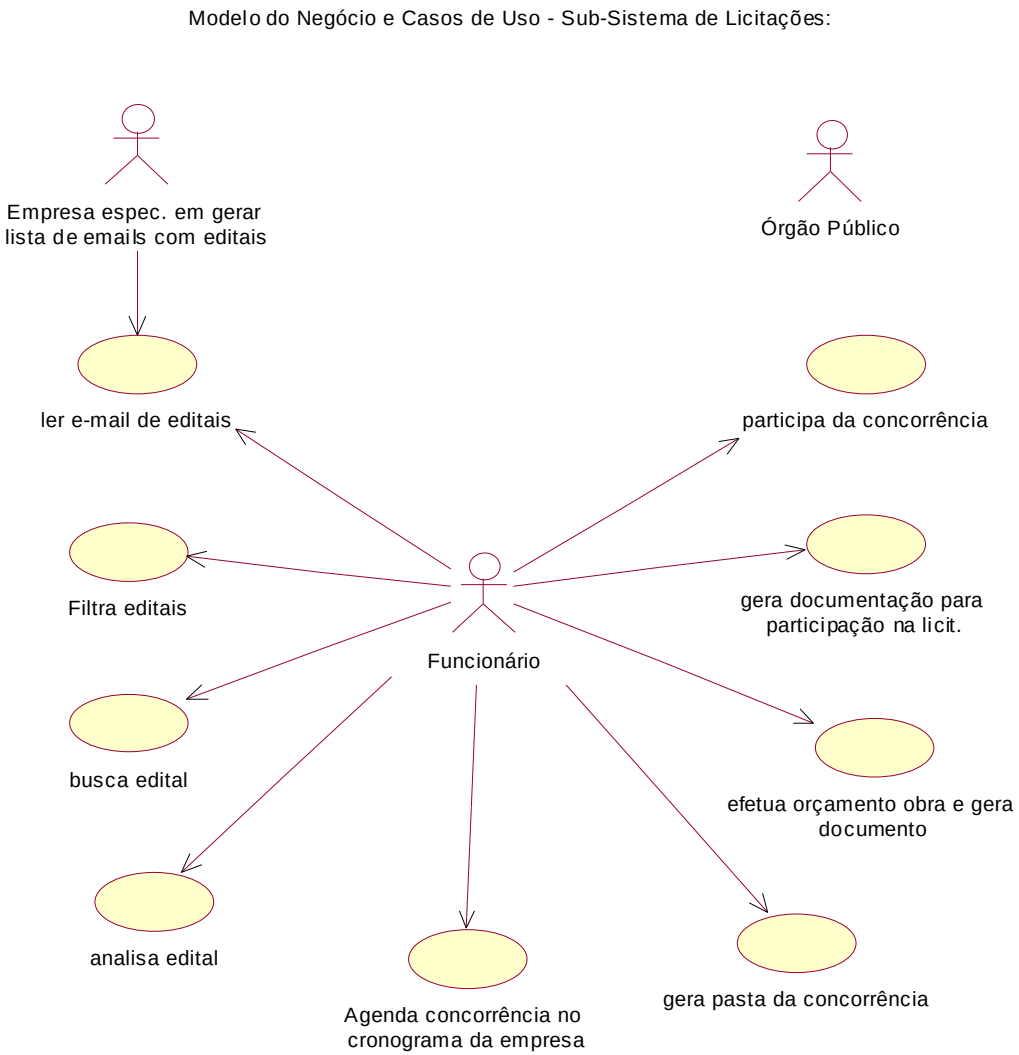


Figura 11 – Modelo de Negócios e os Casos de Uso.

Modelo de Análise – Diagrama de Seqüências: cada Caso de Uso deve ser explorado individualmente na forma de um Diagrama de Seqüência. Na Figura 12 pode-se observar um diagrama de seqüência para o Caso de Uso *Ler E-Mail de Editai*s.

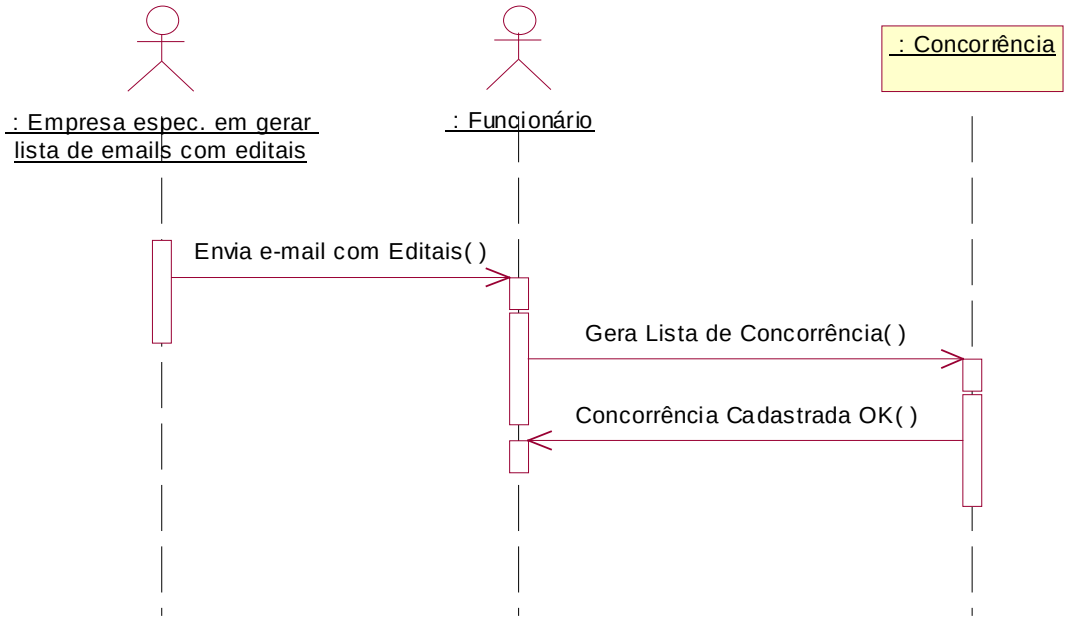


Figura 12 – Diagrama de seqüência para o Caso de Uso *Ler E-Mail de Editais*

Modelo de Análise – Diagrama de Classes: o diagrama de classes apresenta uma visão estática do sistema. Através dele pode-se mapear os atributos e métodos que serão utilizados no projeto. A Figura 13 apresenta o Diagrama de Classes para o Sub-Sistema de Licitações.

Diagrama de Classes do Sub-Sistema de Licitações:

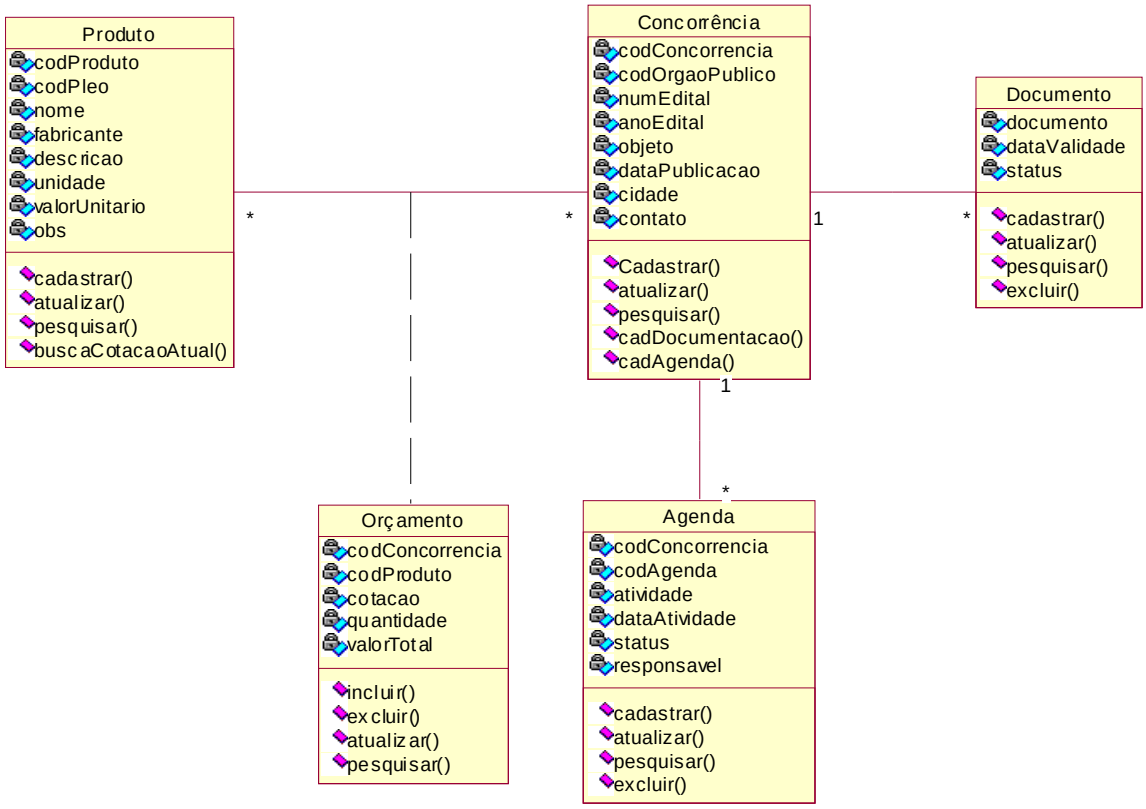


Figura 13 – Diagrama de Classes para o Sub-Sistema de Licitações.

O tempo despendido do processo de modelagem do negócio, levantamento de requisitos e análise do sistema até o momento inicial de programação do sistema – mesmo que isto não esteja sendo desenvolvido – foi um tanto demasiado. Porém, o RUP possibilitou a estruturação do processo de modo que a utilização da UML como linguagem de modelagem fosse substancialmente bem aproveitada. A forma de apresentação e a ordem do processo mantêm a coerência no desenvolvimento dos modelos. Ou seja, com o RUP e UML pode-se considerar uma maior confiança no produto final da modelagem.

Mesmo que o tempo para desenvolver os modelos seja maior, devido ao nível de detalhes, o produto final torna-se mais consistente. O aspecto incremental do RUP tende a ser sob o ponto de vista de novas funcionalidades.

4.3 APLICAÇÃO – XP

Para aplicar na prática o XP, tratou-se o sub-sistema de controle de licitações como um projeto integral. Os User Stories foram criados a partir da descrição textual do processo. Os CRC card e o CRC model foram criados analisando as informações contidas nos User Stories. Alguns esboços de Interfaces complementaram a primeira iteração.

Considerando que este estudo não está focado na implementação do sistema em discussão e sim na modelagem, verificou-se o trabalho transcorrido até o momento limite onde se partiria para a implementação.

Não foi utilizado nenhum sistema de apoio ao desenvolvimento dos modelos. Foram criados modelos de User Stories impressos em folha A4, conforme Anexo A e destes foram criados os CRC cards e model numa ferramenta de desenho padrão do sistema operacional, conforme Anexo C. Abaixo, faz-se comentários a respeito dos modelos obtidos:

User Stories: os User Stories são muito próximos de um Caso de Uso. Porem, são confeccionados pelos próprios clientes como o nome sugere. A partir de um modelo de cartão pré-determinado, o cliente determina um processo e o descreve. A Figura 14 apresenta um User Story desenvolvido no projeto do Sub-Sistema Licitações.

Data: **20** / **11** / **2006** . Tipo da Atividade: Nova: **X** Conserto: ___
Num. Story : **001** Prioridade Usuário: _____ Técnico: _____

Referência Prioridade: _____ Risco: _____ Estimado pelos Técnicos: _____

Descrição da Tarefa: (Ler e-mail de Editais)
Secretária da construtora entra na conta de e-mail da empresa e busca o documento enviado pela empresa que lista concorrências diariamente.

Caminho da Tarefa:

Data:	Status	To do	Comentário

Figura 14 – User Story Ler E-Mail.

CRC cards: Estes artefatos se assemelham a uma classe na UML, contem os atributos e relacionamentos. Na Figura 15 apresenta-se CRC Card que descreve um produto.

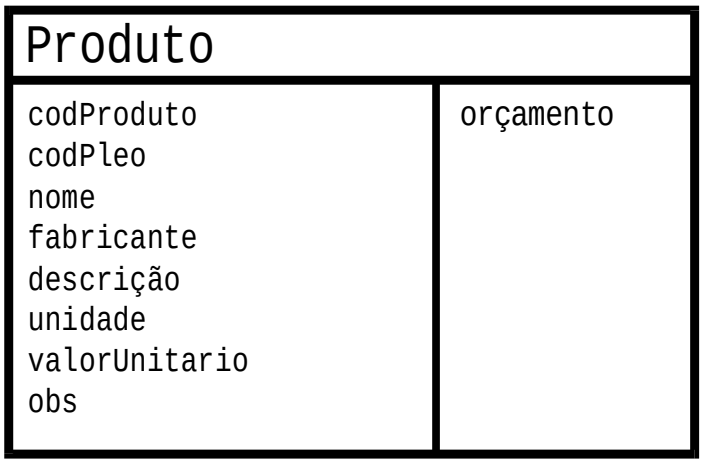


Figura 15 – CRC Card que descreve Produto.

CRC Model: este modelo apresenta todos CRC Cards agrupados apresentando uma visão muito próxima a um Diagrama de Classes da UML. Com este modelo pode-se visualizar os atributos necessários de cada entidade e observar os relacionamentos entre eles. A Figura 16 apresenta o CRC Model para o Sub-Sistema de Licitações.

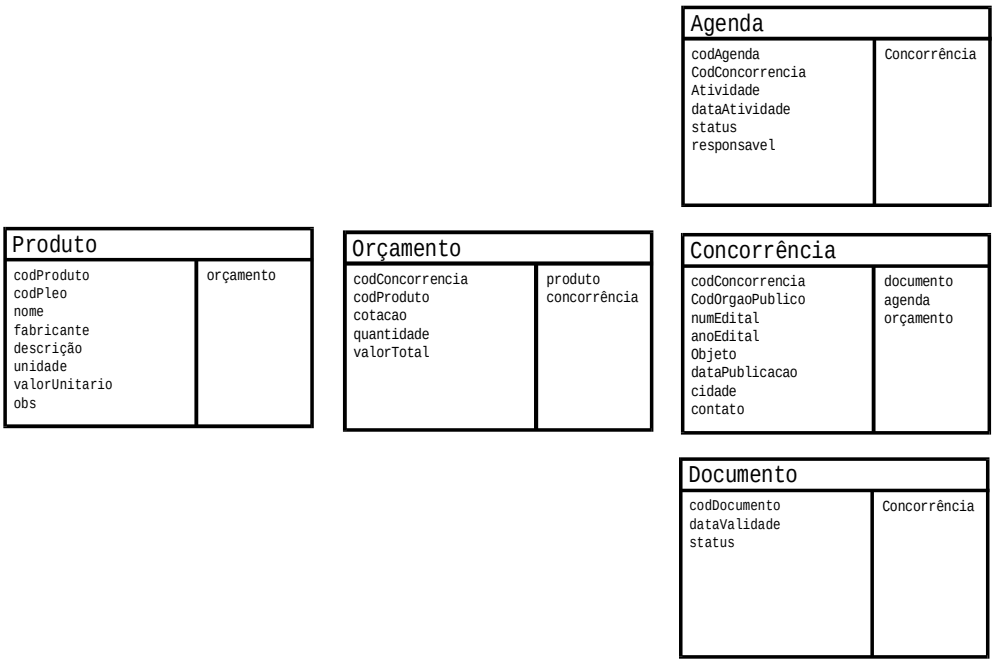


Figura 16 – CRC Model para o Sub-Sistema de Licitações.

Interface Gráfica: através dela, o cliente pode expressar sua vontade para o futuro sistema. Trata-se de um esboço que apresenta as diretrizes de apresentação gráfica do projeto. A Figura 17 apresenta um esboço de uma interface gráfica sugerida pelo cliente.

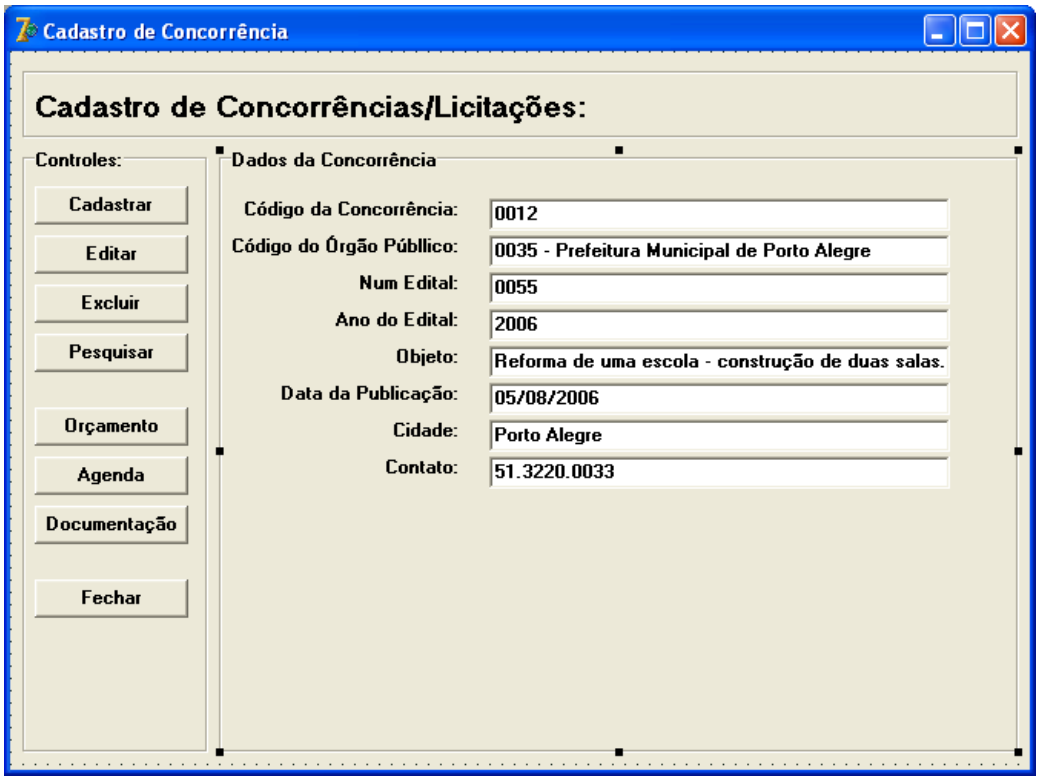


Figura 17 – Interface Gráfica Sugerida pelo cliente.

Durante a modelagem verificou-se a simplicidade dos modelos, mesmo que os User Stories possuam diversos campos e questionamentos, não são de difícil compreensão. No caso de aumentar demasiadamente a complexidade do cartão e este dificultar o entendimento por parte do cliente, pode-se optar por criar cartões híbridos (intermediários entre o proposto por Kent Beck e Scott Ambler). Após a criação dos User Stories torna-se fácil criar os CRC cards e, por conseguinte o CRC model, esse modelo se aproxima muito do diagrama de classes, porém não possui uma riqueza de detalhes que a UML apresenta. As interfaces surgem com facilidade no momento de criação. É fácil para um programador criar alguns esboços de uma interface gráfica que atinja os objetivos.

O ponto positivo foi que se chegou rapidamente ao início da implementação. Leva-se a pensar que é possível entregar um sistema funcionando para o cliente com os procedimentos prioritários definidos no inicio do projeto.

5 CONCLUSÃO

Através do levantamento teórico e das aplicações práticas realizadas no estudo de caso, extraiu-se as seguintes conclusões quanto ao ganho de tempo na modelagem dos processos RUP e XP:

1) Na primeira fase, onde no RUP define-se o modelo de negócios e no XP onde se define a Metáfora e faz-se um planejamento prévio, o XP demandou menos tempo de definição;

Tal fato se deve a simplicidade da criação mental de uma metáfora e publicação aos demais integrantes da equipe, algo notavelmente de baixo tempo em relação à criação de um modelo de Caso de Uso que contemple a funcionalidade como um todo.

2) Na confecção dos Casos de Uso (UML/RUP) em relação aos User Stories (XP) verificou-se que o levantamento de requisitos tende a ser mais eficiente quando se utiliza o RUP.

O fato de o cliente fazer o User Stories (XP) pode vir a tornar o levantamento de requisitos lento. O grau de entendimento do cliente no que se refere passar a informação para o cartão é extremamente relevante. Cabe ressaltar que numa situação real, onde as pressões culturais e ambientais da empresa influem diretamente no poder de atendimento do cliente, faz-se necessário apresentar, explanar e ensinar a usar o material para obter um User Stories como uma boa fonte de informação. Neste caso obtiveram-se boas informações devido à baixa complexidade do módulo do programa e do entendimento por parte da pessoa que descreveu os cartões. Considerando esta discussão pode-se supor que o levantamento de requisitos pode vir a pesar mais no XP, em relação ao tempo, do que um projeto baseado em RUP que, no momento dos levantamentos de requisitos, dispõe de pessoas treinadas para obter o máximo de informação através do cliente.

3) Na confecção do Modelo de Análise (RUP) que neste estudo foi composto pelos diagramas de Classes e Seqüência, verificou-se um maior tempo de modelagem em comparação com os modelos sugeridos pelo XP.

Tal resultado deve-se ao fato da UML possuir uma notação altamente detalhada. A confecção dos CRC cards, CRC model e o esboço das interfaces gráficas foram mais rápidos em relação ao RUP devido à simplicidade do XP.

Estes resultados nos apresentam que a XP é mais veloz que o RUP, de acordo com esta amostra e nas condições apresentadas. Porém, cabe ressaltar que este tipo de resultado não torna um processo melhor que o outro. A XP pode ter sido melhor no quesito tempo, entretanto o RUP, baseado na Unified Modeling Language, tem uma maior detalhamento na sua modelagem.

Isto tudo nos leva a entender que ao escolher um processo para guiar um novo projeto devemos nos questionar quais são nossos reais objetivos. Se o gerente de projeto precisa de Alto nível de detalhamento, escolhe RUP/UML e se precisa de Agilidade na entrega faz-se uso da XP.

6 REFERÊNCIAS BIBLIOGRÁFICAS

[1] LAUDON, Kenneth C.; LAUDON, Jane Price. **Sistemas de Informação**. Rio de Janeiro: LTC, 1999.

[2] RAMOS, Ricardo Argenton. **Treinamento prático em UML**. São Paulo: Digerati Books, 2006.

[3] REZENDE, Denis Alcides. **Engenharia de Software e sistemas de informação**. Rio de Janeiro: Brasport, 2002.

[4] KRUCHTEN, Philippe. **The rational unified process: an introduction**. Addison-Wesley Pub Co., 2000.

[5] BOOCH, Grady; JACOBSON, Ivar; RUMBAUGH, James. **UML, guia do usuário**. Rio de Janeiro: Elsevier, 2000.

[6] SITE **Martin Fowler**. Disponível em http://www.martinfowler.com

[7] SITE **Manifesto for Agile Software Development**. Disponível em http://www.agilemanifesto.org/;

[8] BECK, K. **Programação Extrema Explicada: escolha as mudanças**. Porto Alegre: Bookman, 2004.

[9] SITE **Agile Modeling (AM) Home Page**. Disponível em http://www.agilemodeling.com;

[10] SITE **Processo Unificado e RUP (Rational Unified Process)**. Disponível em http://paginas.terra.com.br/negocios/processos2002/processo_unificado_e_rup.htm

[11] JACOBSON, Ivar; BOOCH, Grady; RUMBAUGH, James. **The unified software development process**. Acm Press Books, 2001. 5ª. Edição.

ANEXO A - Exemplo de User Stories utilizado no estudo de caso

User Story e Cartão de Tarefa

Data: _____ / _____ / _____ . Tipo da Atividade: Nova: ___ Conserto: ___

Num. Story : _____ Prioridade Usuário: _____ Técnico: _____

Referência Prioridade: _____ Risco: _____ Estimado pelos Técnicos: _____

Descrição da Tarefa:

Caminho da Tarefa:

Data:	Status	To do	Comentário

ANEXO B - Estudo de Caso aplicando RUP

PROJETO SUB-SISTEMA DE CONTROLE DE LICITAÇÕES PARA EMPRESA DE ENGENHARIA UTILIZANDO RATIONAL UNIFIED PROCESS.

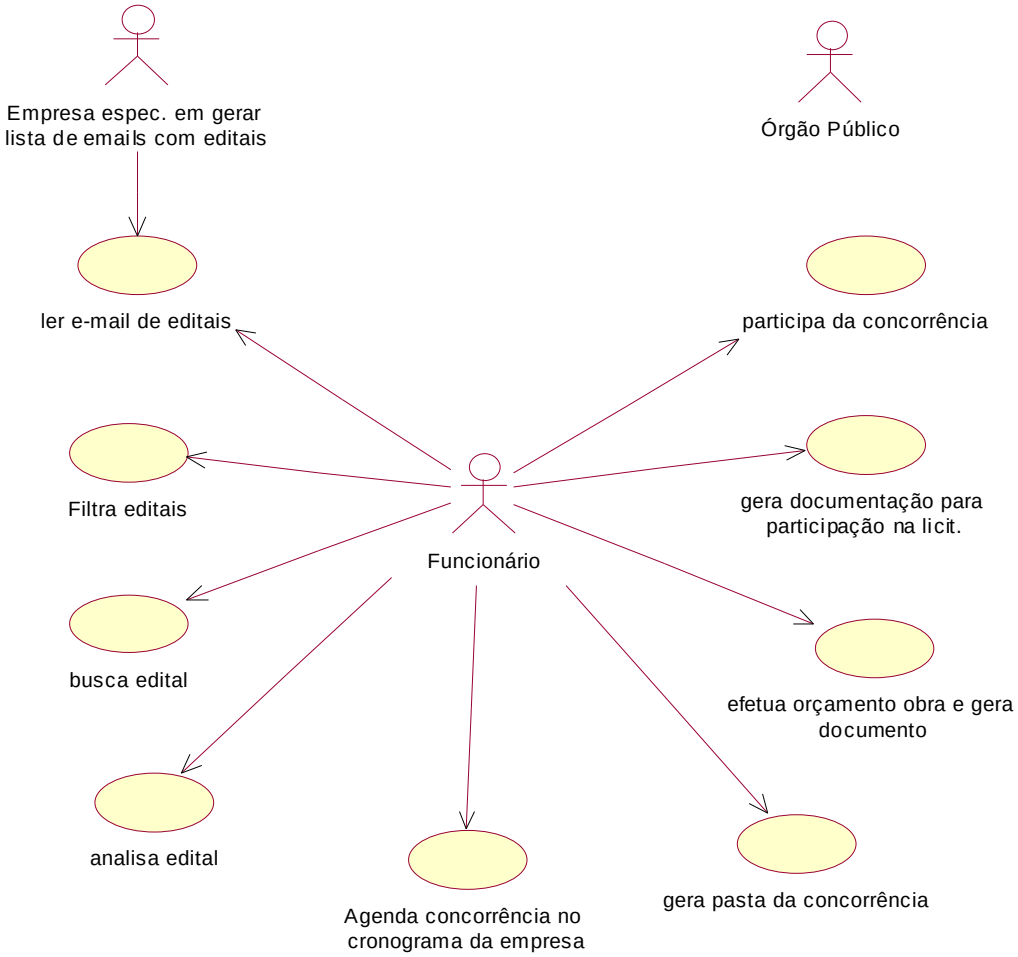
APRESENTAÇÃO:

1) Modelo de Negócio e Casos de Uso:
    a. Ler e-mail de Editais;
    b. Filtra Editais;
    c. Busca Editais;
    d. Analisa Edital;
    e. Agenda Concorrência no cronograma da empresa;
    f. Gera pasta da concorrência;
    g. Efetua orçamento da obra e gera documento;
    h. Gera documento para participar da Licitação;
    i. Participa da concorrência;

2) Modelo de Análise:
    a. Diagramas de Seqüência:
        i. Ler e-mail de Editais
        ii. Busca Editais;
    b. Diagrama de Classes:

Modelo de Negócio e Casos de Uso:

Modelo do Negócio e Casos de Uso - Sub-Sistema de Licitações:



Modelo de Análise:

DIAGRAMA DE SEQÜÊNCIA – *LER E-MAIL DE EDITAIS*

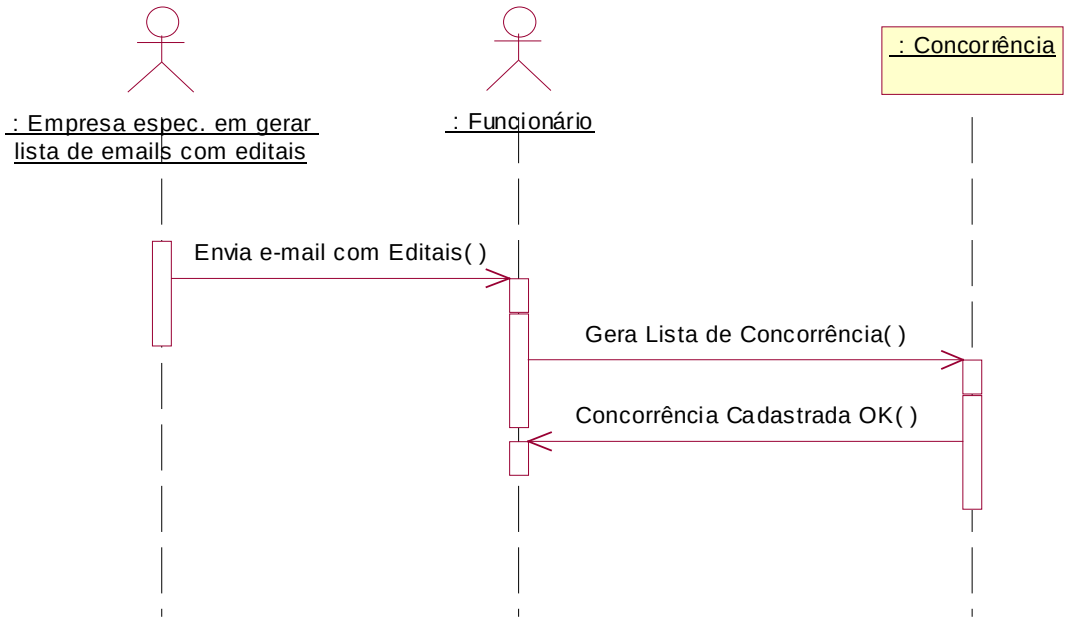


DIAGRAMA DE SEQÜÊNCIA – *BUSCAR EDITAIS*

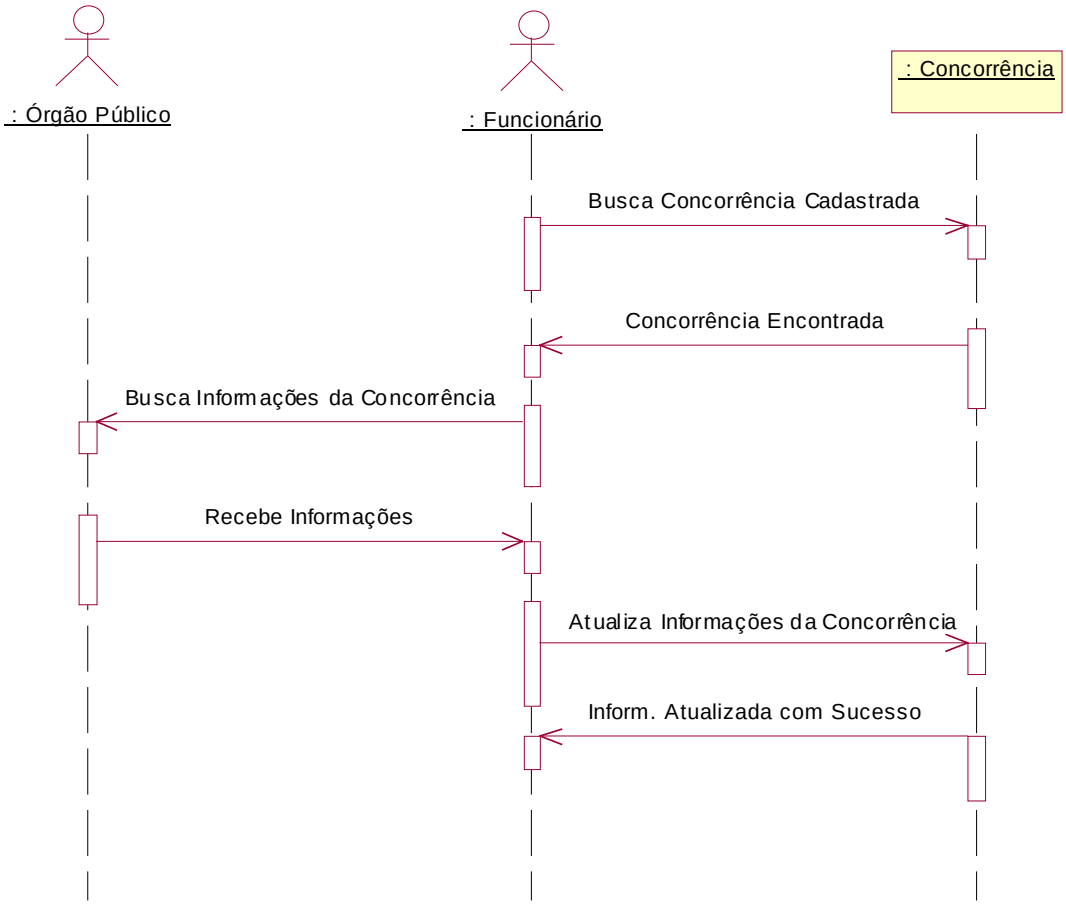
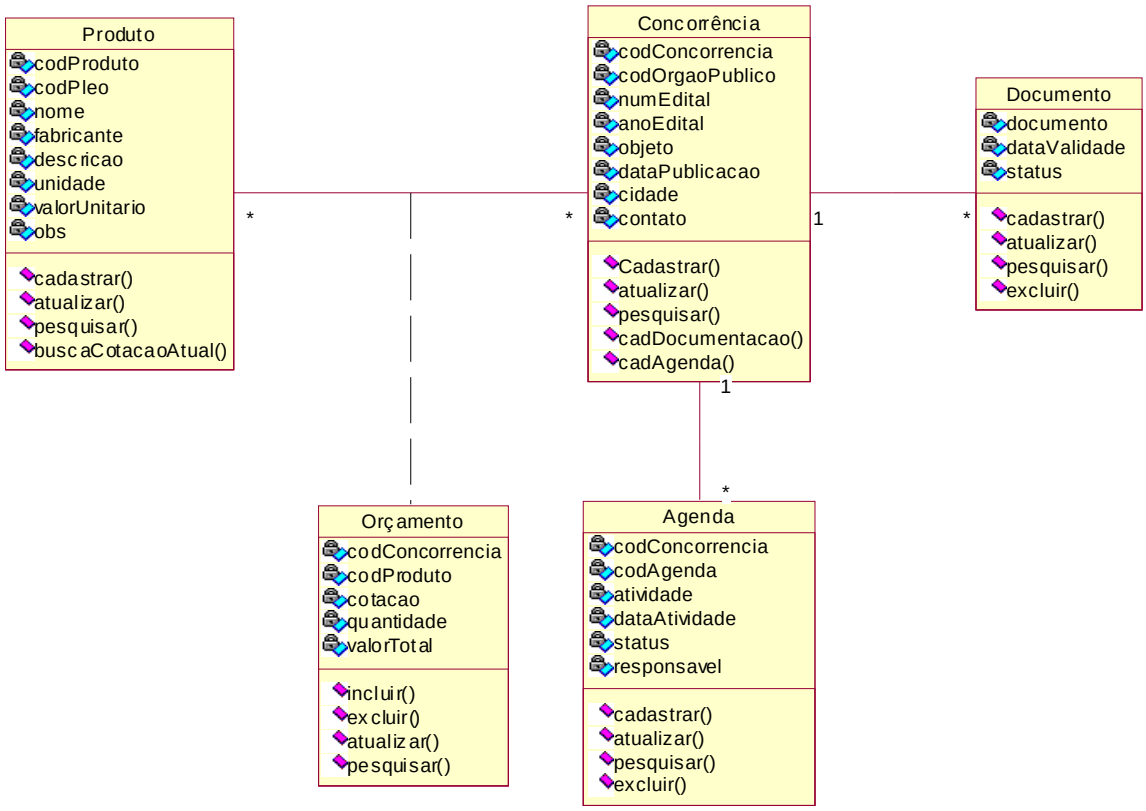


DIAGRAMA DE CLASSES – *SUB-SISTEMA DE LICITAÇÕES*

Diagrama de Classes do Sub-Sistema de Licitações:



ANEXO C - Estudo de Caso aplicando XP

PROJETO SUB-SISTEMA DE CONTROLE DE LICITAÇÕES PARA EMPRESA DE ENGENHARIA UTILIZANDO PROCESSO EXTREME PROGRAMMING.

APRESENTAÇÃO:

3) User Stories:
    a. 001 – Ler e-mail de Editais;
    b. 002 – Filtra Editais;
    c. 003 – Busca Editais;
    d. 004 – Analisa Edital;
    e. 005 – Agenda Concorrência no cronograma da empresa;
    f. 007 – Efetua orçamento da obra e gera documento;
    g. 009 – Participa da concorrência;

4) CRC cards:
    a. Produto;
    b. Orçamento;
    c. Agenda;
    d. Concorrência;
    e. Documento.

5) CRC Model:

6) Interface gráfica:

USER STORIES:

User Story e Cartão de Tarefa

Data: **20** / **11** / **2006** . Tipo da Atividade: Nova: **X** Conserto: ___

Num. Story : **001** Prioridade Usuário: _____ Técnico: _____

Referência Prioridade: _____ Risco: _____ Estimado pelos Técnicos: _____

Descrição da Tarefa: (Ler e-mail de Editais)
Secretária da construtora entra na conta de e-mail da empresa e busca o documento enviado pela empresa que lista concorrências diariamente.

Caminho da Tarefa:

Data:	Status	To do	Comentário

User Story e Cartão de Tarefa

Data: **20** / **11** / **2006** . Tipo da Atividade: Nova: **X** Conserto: ___

Num. Story : **002** Prioridade Usuário: _____ Técnico: _____

Referência Prioridade: _____ Risco: _____ Estimado pelos Técnicos: _____

Descrição da Tarefa: (Filtra de Editais)
Secretária envia e-mail para diretor administrativo que avalia se prossegue o processo para entrar na concorrência

Caminho da Tarefa:

Data:	Status	To do	Comentário

User Story e Cartão de Tarefa

Data: **20** / **11** / **2006** . Tipo da Atividade: Nova: **X** Conserto: ___
Num. Story : **003** Prioridade Usuário: _____ Técnico: _____

Referência Prioridade: _____ Risco: _____ Estimado pelos Técnicos: _____

Descrição da Tarefa: (Busca Edital)
Caso o Dir. Administrativo considere uma licitação viável, a secretária deve buscar informações do processo. Para isto, a pessoa liga para o órgão público em questão e solicita informações complementares bem como o edital completo. Com estas informações, faz-se o cadastro no sistema para conferência posterior.

Caminho da Tarefa:

Data:	Status	To do	Comentário

User Story e Cartão de Tarefa

Data: **20** / **11** / **2006** . Tipo da Atividade: Nova: **X** Conserto: ___
Num. Story : **004** Prioridade Usuário: _____ Técnico: _____

Referência Prioridade: _____ Risco: _____ Estimado pelos Técnicos: _____

Descrição da Tarefa: (Analisa Editais)
O dir. administrativo juntamente com o Engenheiro avaliam em detalhes a viabilidade da concorrência sob o ponto de vista financeiro e de know-how da empresa.

Caminho da Tarefa:

Data:	Status	To do	Comentário

User Story e Cartão de Tarefa

Data: **20** / **11** / **2006** . Tipo da Atividade: Nova: **X** Conserto: ___

Num. Story : **005** Prioridade Usuário: _____ Técnico: _____

Referência Prioridade: _____ Risco: _____ Estimado pelos Técnicos: _____

Descrição da Tarefa: (Agenda Licitação no Cronograma da empresa)
Após aprovação do dir. Adm e Eng. Responsável, a secretária deve colocar as datas pertinentes ao processo de licitação no cronograma geral da empresa, nele estão dados como responsáveis técnicos de fazer orçamento e responsáveis por preparar documentação.

Caminho da Tarefa:

Data:	Status	To do	Comentário

User Story e Cartão de Tarefa

Data: **20** / **11** / **2006** . Tipo da Atividade: Nova: **X** Conserto: ___

Num. Story : **007** Prioridade Usuário: _____ Técnico: _____

Referência Prioridade: _____ Risco: _____ Estimado pelos Técnicos: _____

Descrição da Tarefa: (Efetua Orçamento Obra e Gera Documentação)
Tarefa de gerar orçamento é encaminhada para eng. com prazo determinado para término. A tarefa de preparar documentos para disputar a licitação é encaminhada para secretário executivo.
Datas atualizadas no cronograma da empresa.

Caminho da Tarefa:

Data:	Status	To do	Comentário

User Story e Cartão de Tarefa

Data: **20** / **11** / **2006** . Tipo da Atividade: Nova: **X** Conserto: ___

Num. Story : **009** Prioridade Usuário: _____ Técnico: _____

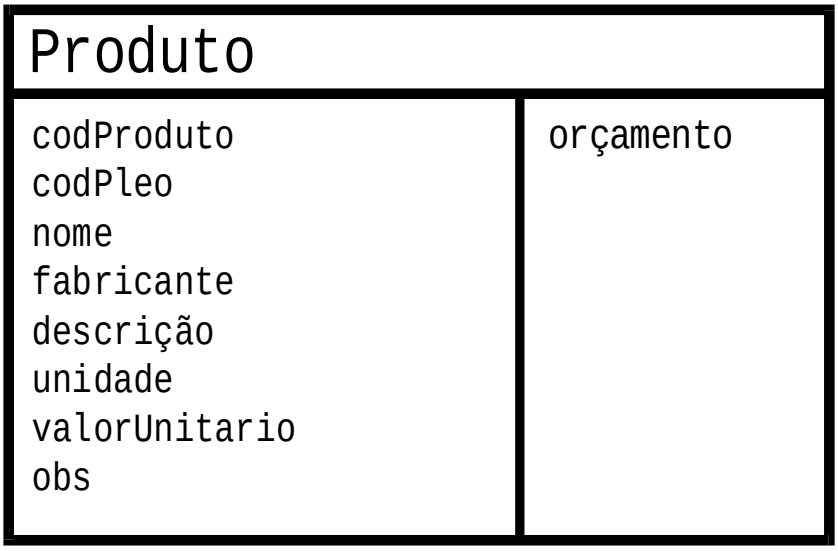
Referência Prioridade: _____ Risco: _____ Estimado pelos Técnicos: _____

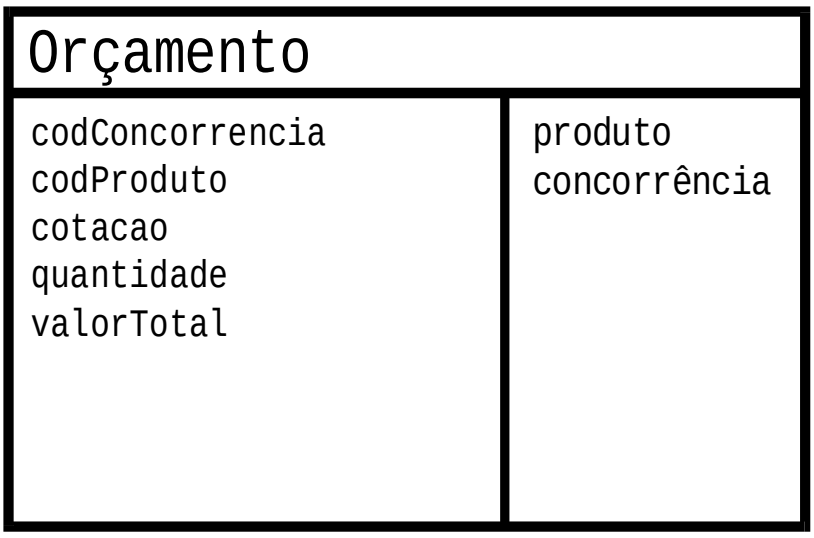
Descrição da Tarefa: (Participação na Licitação) Prioritariamente, destaca-se um Engenheiro para viajar e apresentar documentação exigida para concorrência. Após avaliação das propostas vem o resultado e caso a empresa vença, cadastra-se como uma OBRA (ou projeto) dentro do cronograma da empresa e distribui tarefas.

Caminho da Tarefa:

Data:	Status	To do	Comentário

CLASS RESPONSABILITY COLLABORATOR CARDS(CRC Cards):





Agenda	
codAgenda CodConcorrencia Atividade dataAtividade status responsavel	Concorrência

Concorrência	
codConcorrencia CodOrgaoPublico numEdital anoEdital Objeto dataPublicacao cidade contato	documento agenda orçamento

Documento	
codDocumento dataValidade status	Concorrência

CLASS RESPONSABILITY COLLABORATOR MODEL (CRC-MODEL)

Agenda	
codAgenda CodConcorrencia Atividade dataAtividade status responsavel	Concorrência

Produto	
codProduto codPleo nome fabricante descrição unidade valorUnitario obs	orçamento

Orçamento	
codConcorrencia codProduto cotacao quantidade valorTotal	produto concorrência

Concorrência	
codConcorrencia CodOrgaoPublico numEdital anoEdital Objeto dataPublicacao cidade contato	documento agenda orçamento

Documento	
codDocumento dataValidade status	Concorrência

PLANO DE INTEFACE GRÁFICA: