Centro Estadual de Educação Tecnológica Paula Souza
GOVERNO DO ESTADO DE SÃO PAULO

FACULDADE DE TECNOLOGIA DE AMERICANA

WELLINGTON FÁBIO DE OLIVEIRA MARTINS

PROGRAMAÇÃO ORIENTADA AO OBJETO EM EQUIPES

Americana - SP

Maio – 2009

WELLINGTON FÁBIO DE OLIVEIRA MARTINS

RA: 071210

PROGRAMAÇÃO ORIENTADA AO OBJETO EM EQUIPES

Trabalho de graduação apresentado à Faculdade de Tecnologia de Americana, como parte dos requisitos para obtenção do título de Tecnólogo em Processamento de Dados.

Orientador:

FRANCESCO ARTUR PERROTTI

AMERICANA – SP

Maio – 2009

Dedicatória

A minha esposa Luciene Bispo Custódio Martins que com amor, carinho e dedicação sempre esteve ao meu lado, meus pais Lourdes Luiza de Oliveira Martim e Clovis Romão Martim que sempre me apoiaram incondicionalmente e meu cunhado Samuel César Bispo Custódio que me emprestou o microcomputador.

Agradecimentos

A Deus por sempre iluminar o meu caminho.

Agradeço a todos os que tornaram possível a realização deste trabalho.

A minha família por compartilhar os meus ideais e os alimentarem, por todo o apoio e incentivo, principalmente nos momentos mais difíceis.

Ao meu orientador, Profº Francesco Artur Perrotti pelos conselhos, pelo comprometimento e ajuda na criação deste trabalho.

A meus professores e companheiros de trabalho que sempre me deram oportunidades de novos conhecimentos e crescimento profissional.

Aos amigos que conquistei na FATEC, agradeço pelo companheirismo.

A Luciene, minha esposa, pela companheira que tem sido e por todos os momentos em que tem estado ao meu lado.

Lista de Figuras

Lista de Abreviaturas e Siglas

CPD - Central de Processamento de Dados.

Dot Net - Microsoft Dotnet Ferramenta de desenvolvimento.

IBM - International Business Machines – Empresa da área de TI.

ITSMF - Information Technology Service Management.

OO - Orientado ao Objeto.

OOP - Object Oriented Programming.

POO - Programação Orientada ao Objeto.

RAD - Rational Application Developer.

TI - Tecnologia da Informação.

UML - Unified Modeling Language – Linguagem de Modelagem Unificada.

WEB - Rede Mundial de Computadores.

XP - Extreme Programming.

OMT - Object Modeling Technique – Técnica de Modelagem de Objetos.

OOSE - Object Oriented Software Engineering – Programa para Engenharia Orientada a Objeto.

Resumo

O profissional da área de Tecnologia da Informação que acaba de se graduar e sai à procura uma vaga no mercado, precisa estar preparado para fazer parte de uma equipe de trabalho.

Uma boa forma de programar em equipe é a Programação Orientada ao Objeto, principalmente pelo motivo da popularidade da linguagem Java, da programação Web para servidores, de ferramentas de desenvolvimento como o IBM - RAD e Dot Net da Microsoft entre outras ferramentas voltadas para este tipo de programação que vêm obtendo sucesso e ganhando mercado.

Boas práticas para desenvolvimento de softwares utilizadas em várias corporações como IBM, Ericsson e outras serão apresentadas neste trabalho assim como sugestões para trabalho em equipe, noções de Programação Orientada ao Objeto, alguns diagramas UML e alguns princípios de Extreme Programming.

Palavras-chave: UML – Unified Modeling Language

Extreme Programming – Metodologia de trabalho

Trabalho em Equipe.

Abstract

The professional graduated in Information Technology that has just finished his college and look for a vacancy in the work market, must be prepared to take part of a team work.

A good way to work in a team like a programmer is in the Object Oriented Programming, mainly by reason of the popularity of Java, the programming for Web servers, development tools such as IBM - RAD and Microsoft Dot Net and other tools focused for this type programming that has achieved success and gaining market.

Good practices for software development used in many corporations as IBM, Ericsson and others will be presented in this work as well as suggestions for team work, the concepts of Object Oriented Programming, some UML diagrams and some principles of Extreme Programming.

Keywords: UML – Unified Modeling Language

Extreme Programming – Methodology of work.

Team in work.

Sumário

Introdução

Esta monografia tenta expressar a idéia de que: Uma boa forma de desenvolver sistemas trabalhando com equipes de programadores é utilizando a orientação a objetos.

Com a finalidade de apresentar este tipo de programação em equipes serão apresentados alguns conceitos básicos de trabalho em equipes, conceitos sobre Programação Orientada ao Objeto, alguns princípios do UML e uma crescente metodologia de trabalho chamada Extreme Programming.

Atualmente fala-se muito sobre trabalho em equipes, o profissional ideal e a excelência profissional. A principal qualidade que um profissional precisa ter nesta geração de tecnologias e neste mundo tão pequeno, graças a elas é a capacidade de trabalhar em equipes, com exceção de alguns gênios da computação ou de qualquer setor profissional todos nós precisamos dessa habilidade se nosso objetivo é obter êxito em qualquer atividade.

A computação mais especificamente a programação é uma área ainda em completa metamorfose, enquanto as pesquisas com equipamentos ainda seguem as mesmas regras industriais desde a segunda revolução, que objetiva criar equipamentos mais rápidos, com mais capacidade, conectividade, utilizando para isso conhecimentos sobre a física a química e até biologia. A programação visa criar utilidade para qualquer equipamento eletrônico ou a mais eficaz utilização dos recursos de hardware além da perfeita manipulação da informação. Na realidade programação é a arte de criação, manipulação, ou distorção do mundo virtual que é apenas uma “criança”, ou melhor, a “Virtualização da Realidade” abstração do mundo real.

As profissões que envolvem a Tecnologia da Informação, Análise de Sistemas, Processamento de dados, Ciência ou Engenharia da Computação são muito mais abrangentes do que seus títulos, dentro de empresas ou departamentos de tecnologia as funções dos programadores, analistas, administradores e

operadores de banco de dados, sistemas operacionais (Mainframes), redes, domínios e segurança da informação acabam se confundindo e exigindo que o profissional de qualquer uma destas áreas tenha conhecimento e interação com todas as outras.

Muitos profissionais desta área ou estudiosos quando pensam nas profissões de programadores e analistas, imaginam analistas como gerentes ou “arquitetos” que definem o projeto e passam para os programadores “operários” os problemas que terão de ser resolvidos ou “criados” dentro do projeto, já que a fase de implementação do sistema ou programação propriamente dita é apenas uma das fases do ciclo de vida de sistemas.

Observando o que foi citado acima não parece difícil trabalhar em equipes de programadores. Mas é ai que se iniciam alguns problemas como:

- Um programador experiente prefere trabalhar sozinho ou fazer parte de uma equipe?
- Como as tarefas podem ser compartilhadas dentro de uma equipe de programadores?
- Como um novo integrante de uma equipe de programadores se situa dentro dela?

Para tentar esclarecer estas e outras questões serão apresentadas no capítulo 1 sugestões de trabalho em equipe, organizações de equipes, objetivos, sugestões de comunicação, no capítulo 2 noções de Programação orientada ao objeto, no capítulo 3 uma visão básica sobre UML, e no capítulo 4 equipes XP e no capítulo 5 sugestões de práticas desta metodologia no desenvolvimento em equipes.

1 Trabalhando em Equipes

O mundo empresarial atual funciona com base na integração das atividades e iniciativas de indivíduos e equipes, nunca o tema foi tão abordado e posto em prática, qualquer tipo de atividade pode ser desenvolvida em equipe. Psicólogos ou recrutadores utilizam dinâmicas em qualquer entrevista de emprego observando principalmente o comportamento relacional dos candidatos, criam situações e conflitos que só devem ser resolvidos em conjunto, visto a importância de se saber trabalhar em equipes.

Tendo em vista os profissionais atuais das áreas de TI que é uma profissão nova, mas já estruturada e dividida hierarquicamente dentro dos departamentos das organizações empresariais, apesar de suas nomenclaturas serem modificadas constantemente, suas funções e atribuições como a própria “Tecnologia da Informação” que até à alguns anos ainda não tinha este título, sendo chamado apenas de CPD ou Departamento de Processamento de Dados, hoje já está bem contextualizada não só como título mas como área fundamental e em algumas organizações e não só um departamento, mas até mesmo o único serviço de determinadas empresas.

Figura 1.1 – Tipos de pessoas em uma equipe – Fonte: montagem web.

A figura 1.1 ilustra os tipos de personalidades em uma equipe.

Até a alguns anos quando se via um anúncio de emprego na área de TI eram requeridos os seguintes atributos: "*Conhecimentos em Linguagens de Programação: C, Pascal, Cobol, Conhecimentos avançados em administração de Redes, Gerenciamento de Banco dedados, Aplicativos Gráficos, Necessário três anos de experiência na área etc...*" o profissional precisava ser o extremamente capacitado para dominar todas estas linguagens. Com o passar do tempo o mercado aprendeu a filtrar melhor a procura de profissionais, hoje temos anúncios mais simplificados: "*Precisa-se de Programador Web, Necessário conhecimentos em PHP, MySQL, Servidores Apache, Sistema Operacional Linux e conceitos da WEB 2.0 e Inglês intermediário/avançado.*" ainda é difícil encontrar o candidato ideal, mas pelo menos o indivíduo que deseja trabalhar em uma dessas áreas já sabe em que se especializar.

Segundo (Gestão Estratégica Pública - Turma 2005) Serão apresentadas visões gerais de trabalho em equipes.

1.1 Definição de Equipe:

Segundo (Dicionário Aurélio - 2006), "equipe é um conjunto ou grupo de pessoas que se aplicam a uma tarefa ou trabalho". Definição mais simples, deste modo percebe-se que equipe depende de trabalho e pessoas.

Focando nestes aspectos, a seguir serão apresentados fatores que possibilitam a harmonia desses personagens e podem facilitar a visão geral de trabalho em equipe.

1.2 Comunicação

Se um trabalho é realizado por mais de uma pessoa, o inicio deste está na comunicação, aspecto incansavelmente estudado e debatido em toda a história da

humanidade, o estudo da comunicação humana nunca foi tão explanado, também porque nunca se teve tantos meios e indivíduos conectados como nos dias de hoje.

Por isso a importância da comunicação, dentro de uma organização a necessidade de transmitir aos seus cooperadores, funcionários e todas as pessoas envolvidas os objetivos, cria uma linguagem interna própria.

A visão, a missão e os objetivos de uma instituição determinam à sua personalidade e estes devem ser constantemente lembrados por seus funcionários, dirigentes e todos os que fazem parte da instituição.

Além da comunicação verbal a comunicação visual e escrita são de estrema importância no trabalho em equipe, é preciso um esforço para manter clareza e objetividade em todas as formas de comunicação, principalmente nas que envolvem a TI, pois a informação é atualmente o bem mais precioso de uma instituição e deve ser tratada como tal.

1.3 Confiança e Credibilidade

Tanto dentro como fora da equipe confiança e a credibilidade são status que devem ser conquistados através de atitudes e resultados.

O indivíduo integrante de uma equipe deve conquistar de seus companheiros a confiança para realizar sua parte no trabalho com perfeição e agilidade, isto vem através de seu histórico dentro da equipe. Da mesma forma a equipe deve conquistar este status através de seu histórico dentro da organização.

Estes resultados se favoráveis trazem a equipe maiores responsabilidades e consequentemente maior quantidade de trabalho, delegados a esta. Por isso características que devem estar presentes dentro da equipe são:

- **Sinceridade** – para que haja confiança entre seus integrantes.
- **Competência** – todo trabalho deve ser realizado no prazo estipulado e conforme foi combinado, então devem ser estabelecidos prazos que sejam possíveis de se cumprir.

- **Confiabilidade** – havendo competência consequentemente haverá confiança nas próximas tarefas.

1.4 Escutar

Nada do que foi citado sobre comunicação tem sentido se não for compreendido, antes de comunicar com excelência é preciso ouvir e entender a mensagem ou o problema.

Deve-se dar mais importância a ouvir do que a falar, a ler e entender do que tentar explicar.

Os fatores mais comuns que dificultam a compreensão das mensagens são emocionais, estresse, ansiedade, desinteresse, levar para o lado pessoal e sentimental.

1.4.1 Impessoalidade

Toda e qualquer informação passada dentro de uma equipe em um ambiente de trabalho é completamente profissional e não deve ser recebida como pessoal, é claro que como se trata de seres humanos isso é muito difícil, mas aquele que consegue separar estes ambientes certamente se destaca e tem excelente desempenho como membro de uma equipe.

1.4.2 Contexto

Deve-se prestar atenção no ambiente e no momento em que se está passando uma informação, durante uma conversa, uma reunião ou até mesmo quando se escreve um memorando ou qualquer tipo de comunicado, os fatores emocionais podem influenciar no contexto da mensagem.

Por isso o local, a hora o grau de formalidade e condições em que acontece o diálogo influenciam no escutar.

1.5 Coordenação de Ações

O movimento básico da comunicação no trabalho será aqui expressa em quatro fases:

- **Criação do contexto** – Alguém inicia a ação declarando que lhe falta algo, manifesta uma insatisfação ou realiza um pedido ou aceita uma oferta para que outra pessoa execute uma ação que supra esta falta.
- **Negociação** – Este outro se compromete, ou não, em realizar o pedido feito ou faz uma contra-oferta, procurando redefinir condições, negociando tarefas e prazos, fixando um compromisso e responsabilidades para a execução do que foi solicitado.
- **Realização** – É executada a tarefa a que se comprometeu, com prazos e condições asseguradas, e declara haver cumprido o pedido.
- **Avaliação** – Aquele que iniciou recebe o resultado da ação realizada, verifica o que foi feito e declara que as condições de satisfação foram, ou não, cumpridas. É uma fase em que pode ser forte o componente de aprendizagem sobre erros e acertos, criando condições para novos ciclos de coordenação de ações.

O esquema a seguir apresentado pela figura 1.2 sintetiza o ciclo da comunicação no trabalho o que facilita a visualização:

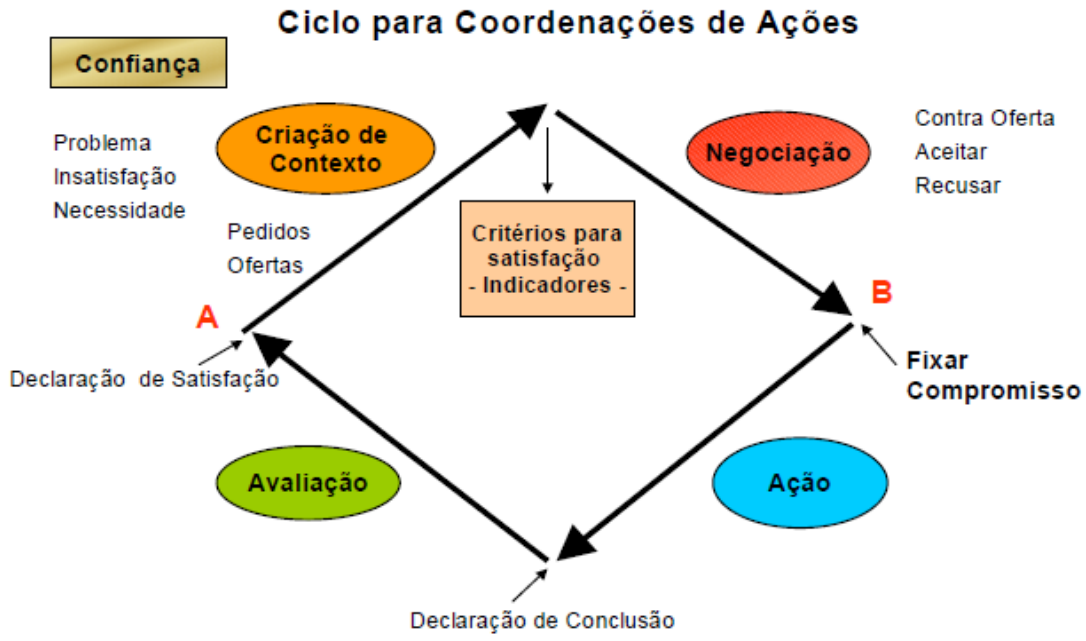

Figura 1.2 – Ciclo para coordenação de ações – Fonte: Gestão Estratégica Pública -2005

1.6 Avaliação de Ações

Todos os resultados obtidos através de ações da equipe são avaliados, por seus integrantes, por seus superiores e até por membros de outras equipes. Para que haja crescimento da organização.

As ações de uma equipe dependem de seus objetivos e tipos de trabalhos realizados por ela, o que foi tentado demonstrar foi uma maneira genérica de visualizar as ações mais comuns a todos os tipos de equipes.

1.7 Motivação

Como se trata de seres humanos que são movidos por emoções, objetivos e alvos pessoais, a motivação da equipe é algo de grande relevância.

Os resultados positivos conquistados já são motivadores naturais, além de boas políticas de salários, comissões, planos de carreira e outros fatores são importantes não só para manter a equipe motivada, mas passam a impressão de solidez e confiabilidade da instituição.

1.8 Documentação

Manter um histórico da equipe é essencial para manter uma personalidade, assim qualquer novo integrante pode conhecer o ambiente em que está se engajando e se adequar com mais facilidade as rotinas de trabalho.

Troféus, fotografias, cartas, são documentos trazem facilmente a memória fatos de êxito de devem ser repetidos ou fracassos que devem ser evitados.

2 Programação Orientada ao Objeto

A Programação Orientada ao Objeto (Object-Oriented Programming) ou (OOP) foi criada no inicio da década de 70 e se origina da linguagem Simula (Simula Language), concebida na Noruega no início da década de 60 que foi criada para fazer simulações, como o próprio nome sugere. Esse tipo de programação possui como principal característica a semelhança com o mundo real a maior facilidade de abstração das rotinas diárias.

A primeira linguagem de programação a utilizar os conceitos de OOP foi a linguagem SIMULA-68, em seguida surgiu a linguagem Smalltalk criada pela Xerox, que popularizou e incentivou o emprego da OOP.

Hoje quando se ouve falar de Objetos, Classes, Métodos, o que vem a mente de um programador é linguagem Java que foi completamente desenvolvida para utilizar esse tipo de programação.

O que a OOP trás realmente de vantagem em relação à programação estruturada é principalmente a reutilização de códigos e a escrita em módulos.

2.1 Definição

Uma definição para objeto seria a de “um ‘ente’ ativo dotado de certas características que o tornam ‘inteligente’, a ponto de tomar certas decisões quando devidamente solicitado.” (Sanra Rahal Júnior, 2009).

2.2 Itens da OOP

A OOP possui então uma hierarquia um pouco diferente da Programação estruturada e algumas características próprias, as principais são: Classes, Objetos, Instâncias, Métodos, Eventos e Atributos.

Classes: são as principais características de um objeto, vários objetos semelhantes que possuem o mesmo tipo de informação e tem o mesmo comportamento são de um certo tipo de classe. Hierarquicamente a classe é o “esqueleto”, o projeto do Objeto. Quando iniciamos um programa OO a primeira coisa que definimos é uma classe.

Objetos: são estruturas prontas para ser utilizadas no programa que pertencem obrigatoriamente a uma classe e estão instanciadas, possuem estados e comportamentos, estados são informações sobre o objeto e comportamento são as coisas que podem ser feitas com o objeto ou pelo objeto.

Instâncias: um objeto em particular de uma dada classe é uma instância desta classe, em Java, por exemplo, criamos uma classe, definimos seus Métodos e Atributos, então para utilizarmos esta classe definimos uma variável que será do tipo desta classe e quando utilizamos no programa esta variável estamos utilizando uma instância da classe criada.

Métodos: São rotinas que representam o que uma classe pode fazer no programa ou o que se pode fazer com a classe, uma classe pode possuir vários métodos que podem ser herdados por outras classes.

Atributos: São características de uma classe, como campos pré-preenchidos ou não de opções desta classe.

Eventos: É um tipo de método especial para tratamento de acontecimentos enquanto o programa está em execução como um clik do mouse, o preenchimento de um campo, o pressionar de uma tecla, a abertura ou fechamento de um módulo do programa dirigido especialmente para a parte gráfica do programa.

2.3 Características

Para que uma linguagem seja considerada Orientada ao Objeto ela precisa suportar quatro conceitos básicos: abstração, encapsulamento, herança e polimorfismo.

Abstração é a habilidade de modelar características do mundo real do problema que o programador esteja tentando resolver. Passar para o programa em forma de entidades ou Classes as características de coisas, pessoas ou situações e interagi-las.

Encapsulamento é a capacidade da linguagem de isolar certos dados ou rotinas de forma que estes só sejam utilizados onde foram criados, um dado ou rotina envolvido por código só visível na rotina onde foi criado, o que possibilita ao programador, maior segurança quanto aos acessos indevidos por parte de outras rotinas, semelhante a variáveis locais definidas em linguagem C, Pascal e outras linguagens quaisquer.

Herança é principal característica que permite um código OO ser re-aproveitável, pois se já foi implementado em ima entidade pai inúmeras características (Atributos) estas podem ser reaproveitadas por outras entidades do mesmo tipo.

Poliformismo é a capacidade que um objeto tem de possuir inúmeros comportamentos, um mesmo código que pode ser aplicado a várias classes de objetos.

Para melhor entendimento de POO, e conhecimento de uma técnica de modelagem, será apresentado a seguir os princípios básicos da Linguagem de Modelação UML.

3 Princípios da UML

Segundo (RODRIGUES DA SILVA e ESCALEIRA VIDEIRA - 2001) O UML é uma "linguagem diagramática, utilizável para especificação, visualização e documentação de sistemas de software".

O UML surgiu em 1997 na seqüência de um esforço de unificação de três das principais linguagens de modelação orientadas a objetos (OMT, Booch e OOSE). Seguidamente, adquiriu o estatuto de norma no âmbito da OMG e da ISO, tendo vindo a ser adotado progressivamente pela indústria e academia em todo o mundo.

Um princípio que causou o surgimento do UML foi à ocorrência da unificação de diferentes elementos existentes em vários métodos, cujos principais são: A linguagem Booch de Grady Booch, a OMT de James Rumbaugh e a OOSE de Ivar Jacobson.

Entre outras inúmeras o UML apresenta as seguintes características:

- É independente do domínio de aplicação, (pode ser usado em projetos de diferentes características, tais como sistemas cliente/servidor tradicionais, sistemas baseados na Web, sistemas de informação Geográficos, sistemas de tempo real),
- É independente do processo ou metodologia de desenvolvimento;
- É independente das ferramentas de modelagem;
- Apresenta mecanismos potentes de extensão;
- Agrega um conjunto muito significativo de diferentes diagramas como diagramas de casos de utilização, de classes, de objetos, de colaboração, de atividades, de estados, de componentes, e de instalação.

Embora o objetivo deste trabalho seja apresentar metodologias de trabalho que envolva equipes e tenha a ver com a modelação de software, não é demais salientar que o UML pode ser usado em outros contextos como: por gestores, para representarem à organização das empresas e respectivos processos de negócios, por juristas, para representarem as relações entre leis entre outras utilizações.

A abrangência da UML é muito extensa para ser expressa em poucas páginas e este não é o objetivo desta monografia, mas sim utilizá-lo como uma melhor visão da Programação orientada ao Objeto, por isso serão apresentados apenas alguns de seus principais diagramas o que facilitará apenas o entendimento deste tipo de programação e sua execução em equipes.

Atualmente a UML se encontra na versão 2.0 com 20 diagramas que representam inúmeras partes e funções dos sistemas de informação, destes serão apresentados somente os que mais são habitualmente utilizados por desenvolvedores em ambientes corporativos.

3.1 Tipos de Elementos Gráficos utilizados nos diagramas UML

Os elementos utilizados nos diagramas UML podem estar presentes em vários diagramas com os mesmos sentidos, mas em contextos diferentes por isso antes de analisar os diagramas serão apresentados na figura 2.1 os elementos principais destes que são: classes, classes ativas, interfaces, casos de utilização, atores, colaborações, componentes e nós.

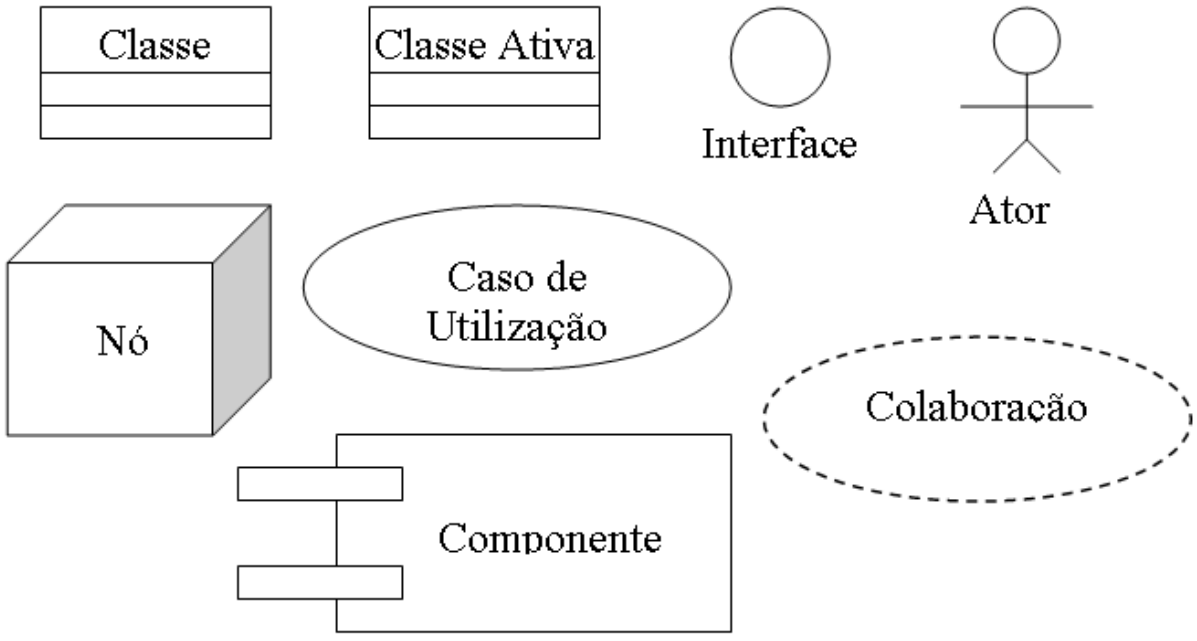

Figura 2.1 – Elementos básicos da UML – Fonte: ilustração criada pelo autor.

Também os elementos básicos de comportamento (Estado, Mensagem), de agrupamento (pacotes) e de anotação (anotações ou notas) são demonstrados na figura 3.2.

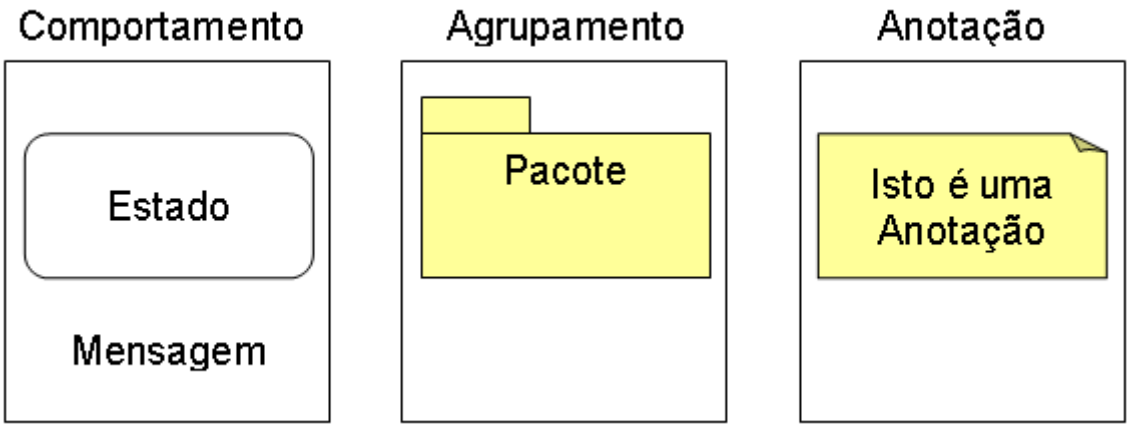

Figura 3.2 – Elementos de comportamento - Fonte: ilustração criada pelo autor.

Completando os diagramas e criando as interdependências dos elementos, as relações entre eles estabelecem o principal entendimento do contexto em que são apresentados e são as seguintes como mostra a figura 2.3:

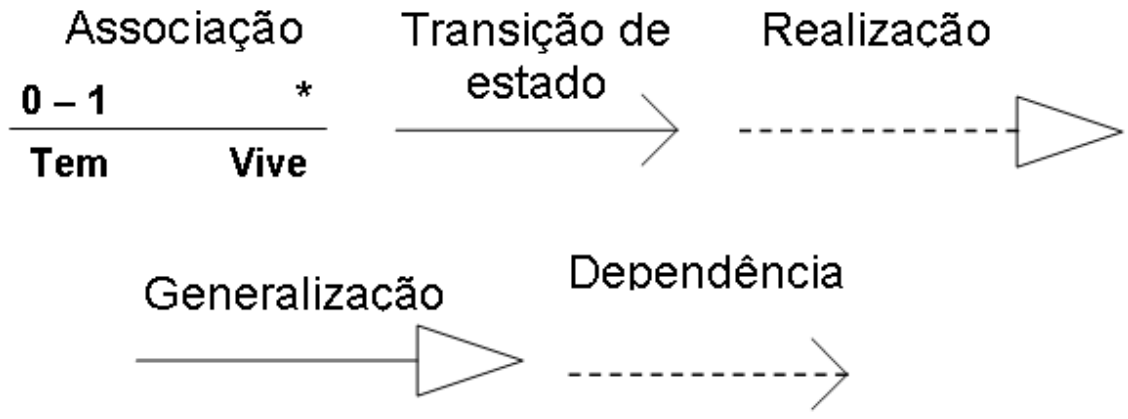

Figura 2.3 – Relações - Fonte: ilustração criada pelo autor.

Os principais tipos de relações do UML são: associação, dependência, realização, generalização e transição de estado, cada um deles é utilizado em seu diagrama com o mesmo significado.

Em seguida serão exemplificados alguns dos diagramas e suas funções dentro de um ambiente de Programação OO.

3.2 Diagrama de Casos de Utilização:

Este diagrama descreve a relação entre os atores e os casos de utilização de um dado sistema, se aplica principalmente para um entendimento dos problemas cotidianos tanto pelos desenvolvedores quanto para o cliente ou usuários deste sistema. É de fácil entendimento é serve como base para os demais.

A figura 2.4 mostra um exemplo de caso de uso em que um ator (Cliente) pode fazer uma reserva em um restaurante:

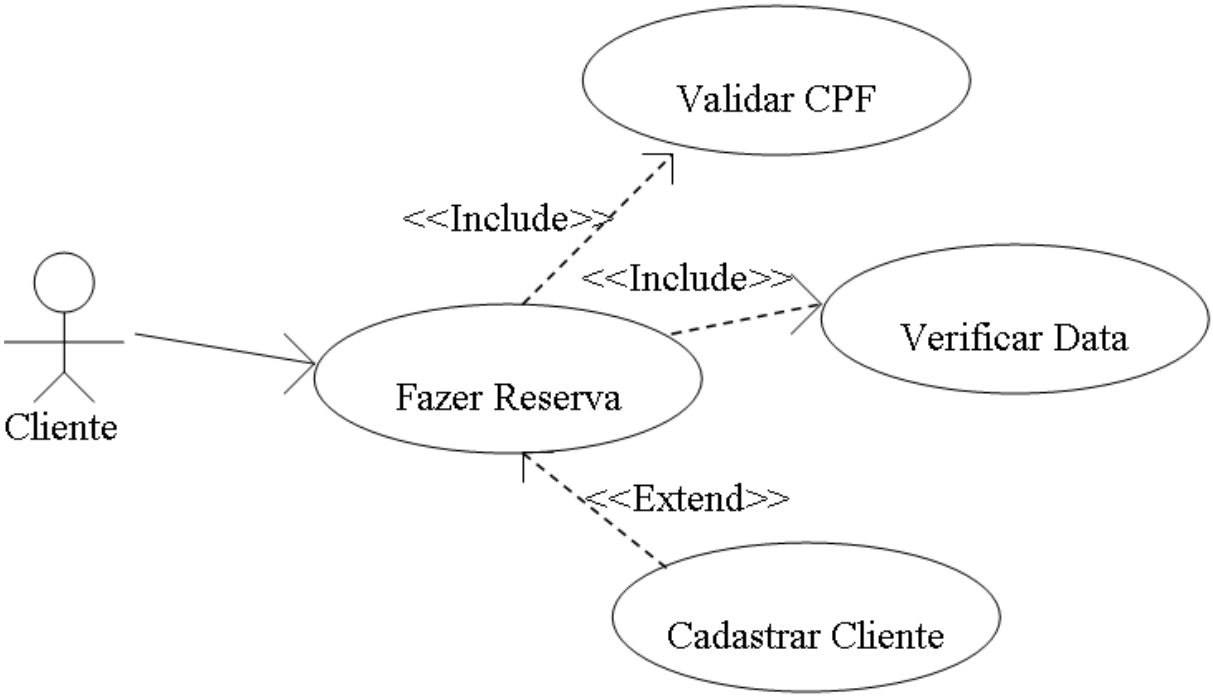

Figura 2.4 – Exemplo de Diagrama de Caso de Utilização - Fonte: ilustração criada pelo autor.

As palavras “Include” e “Extend” indicam o sentido da relação de dependência entre os casos de uso, por exemplo: para fazer a reserva o cliente tem a opção de

se cadastrar caso ainda não tenha o cadastro no restaurante, mas se já for cadastrado não precisa deste caso de utilização por isso a relação entre eles aparece com a palavra “Extend”, pois o caso de uso “Cadastrar Cliente” é uma extensão do caso “Fazer Reserva”. Já os casos “Verificar Data” e “Validar CPF” aparecem com a palavra “Include” nas relações de dependência pelo motivo de serem incluídos no caso “Fazer Reserva” ou como símbolo de obrigatoriedade.

Este é possivelmente o diagrama mais simples de se entender do UML e pode ser utilizado por toda a equipe de desenvolvimento de sistemas.

3.3 Diagrama de Classes

As classes são as estruturas iniciais e fundamentais da POO e por isso para expressar suas relações, seus atributos e métodos, as heranças, cardinalidade das relações este diagrama é utilizado além de muitos outros objetivos.

A seguir estará exemplificado um diagrama de classes que demonstra a herança por outras duas classes dos atributos da superclasse “PessoaFisica”.

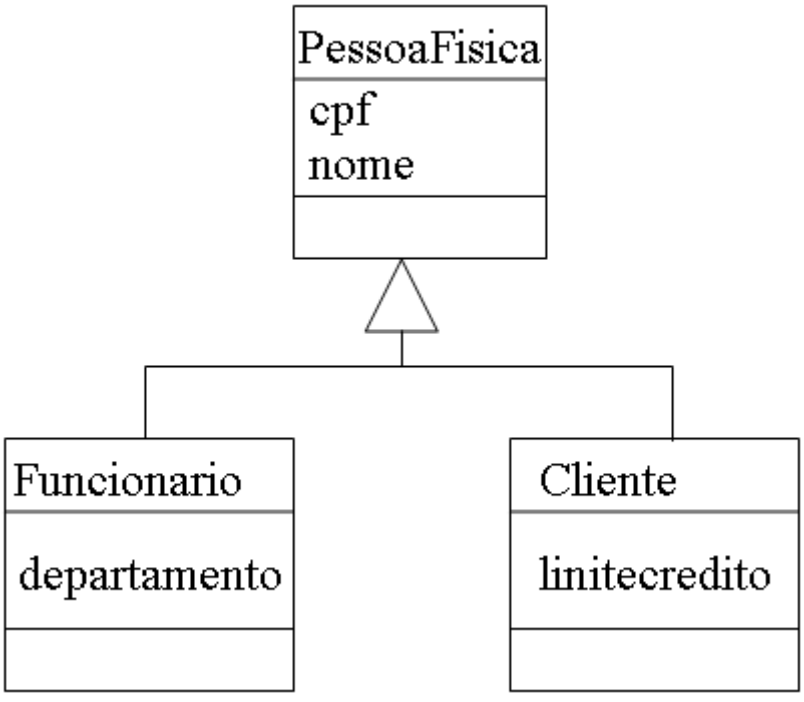

Figura 2.5 – Exemplo de Herança - Fonte: ilustração criada pelo autor.

Além da herança exemplificada pela figura 2.5 o diagrama de classes pode representar inúmeras outras relações entre as classes e dar a base para outros diagramas como o de Objetos.

3.4 Diagrama de Seqüência

Os diagramas de seqüência ilustram interações entre objetos num determinado período de tempo. Em particular, os objetos são representados pelas suas “linhas de vida” e interagem por troca de mensagens ao longo de um determinado período de tempo.

3.5 Diagrama de Atividades

Os diagramas de atividades (ver exemplo da figura 2.6) são um caso particular dos diagramas de transição de estado, no qual a maioria dos estados são substituídos pelos conceitos correspondentes a ações e/ou atividades, e no qual as transições são desencadeadas devido à conclusão de ações nos estados originais.

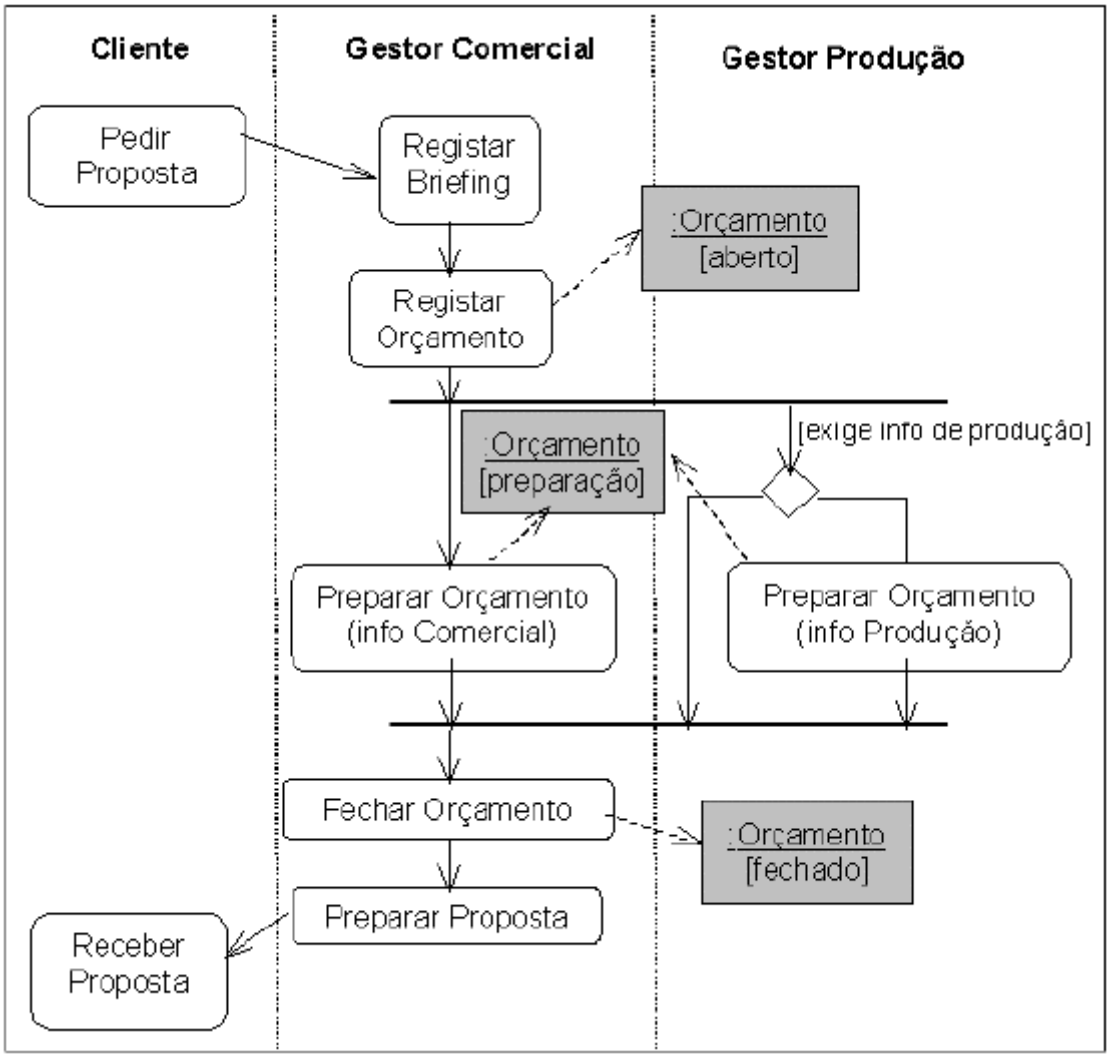

Figura 2.6 – Exemplo de Diagrama de Atividades - Fonte: ilustração criada pelo autor.

Outros diagramas que fazem parte da UML são de relevante importância e demonstram integração entre todas as partes de um sistema de informação como:

- Diagramas de objetos
- Diagramas de comportamento
- Diagramas de estados
- Diagramas de interação (diagramas de seqüência e diagramas de colaboração)
- Diagramas de arquitetura:
- Diagramas de componentes
- Diagramas de instalação

Mas estes não serão mais aprofundados nesta monografia devido à relevância do assunto em questão.

3.6 Considerações sobre o UML

O UML como linguagem de modelagem de sistemas em POO está no inicio, meio e fim deste tipo deste tipo de programação e não somente como documentação, mas como fundamental meio de comunicação e padronização da linguagem.

Já contextualizados os princípios fundamentais da Programação Orientada ao Objeto e a Linguagem de Modelagem UML, deste ponto em diante estarão expressos os conceitos e práticas fundamentais para uma programação em equipe, não só utilizando POO mas qualquer tipo de programação.

4 Princípios Básicos de Extreme Programming

Uma metodologia de trabalho em equipe na área de TI que vem sendo utilizada inclusive por empresas de tecnologia como a IBM Corporation.

Segundo (Teles, 2005) uma definição para Extreme Programming seria: “Uma metodologia de desenvolvimento de software, nascida nos Estados Unidos ao final da década de 90 e que vem fazendo sucesso em diversos países, por ajudar a criar sistemas de melhor qualidade, que são produzidos em menos tempo e de forma mais econômica que o habitual”.

Os principais fundamentos do XP tiveram origem em meados da década de 80 por Kent Beck e Ward Cunningham então funcionários da Tektronixs, Inc, mas só chegaram aos atuais conceitos quase uma década depois como um conjunto de boas praticas para programadores que na época que foram condensadas no padrão de linguagem *Episodes*.

Seus principais fundamentos ou valores são:

- Comunicação
- Coragem
- FeedBack
- Respeito
- Simplicidade

Este capítulo apresenta um breve resumo de cada um desses valores que são explicados de uma forma um pouco mais abrangente no livro citado acima de Vinícius Manhães Teles, Mestre em Informática pela UFRJ.

4.1 Comunicação

Como já citado no capítulo 1.2 também é principal questão no desenvolvimento de sistemas, as ilustrações abaixo retiradas de uma apresentação em PowerPoint, utilizada em uma das palestras sobre XP por (Teles 2005).

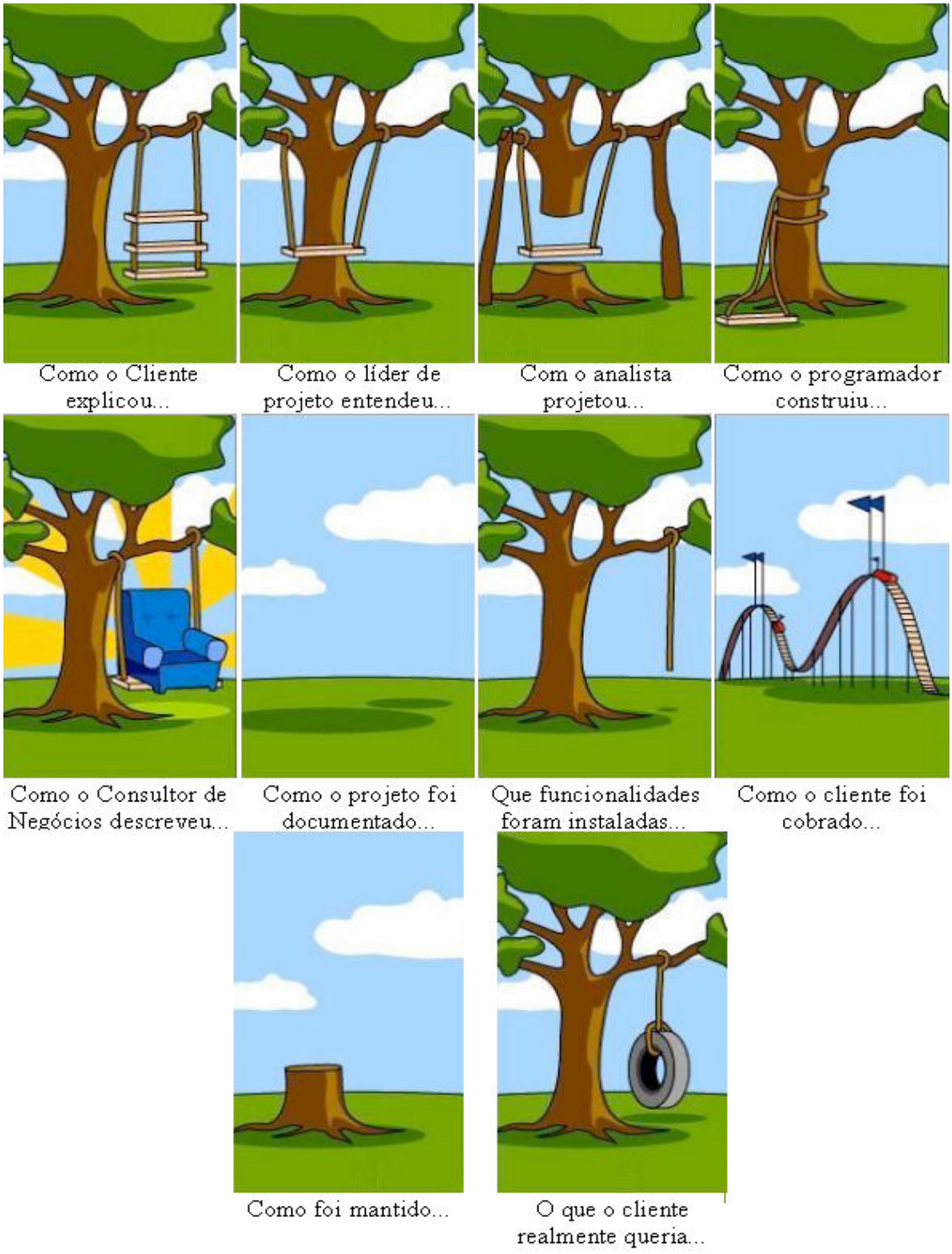

Figura 3.1 – Exemplo de Comunicação – Fonte: Apresentação de Vinícius Teles sobre XP.

A ilustração (Figura 3.1) representa o “Cliente” com uma empresa que precisa de uma solução tecnológica para um determinado problema e para isso contrata os serviços de uma empresa de Tecnologia da Informação.

Para que os desenvolvedores compreendam o que o cliente deseja é preciso que haja comunicação clara e objetiva, para isso, nada melhor do que a comunicação presencial, ao vivo a mais simples forma de comunicação existente. Muito melhor do que videoconferência, telefonemas e outras.

A comunicação é tão importante que se possível até mesmo o desenvolvimento do projeto pode ser levado para dentro da empresa ou departamento do cliente. Quanto mais o cliente estiver envolvido e próximo dos desenvolvedores mais fácil é a compreensão dos problemas.

4.2 Coragem

O que mais ocorre durante um projeto de software são mudanças, mesmo quando são fechados contratos em que são assumidos compromissos de não alteração do que foi especificado, alterações e atualizações são inevitáveis, tanto por parte dos desenvolvedores quanto por parte do cliente e principalmente das alterações constantes no mercado.

Para estas alterações é necessário coragem, porque depois que um sistema já foi implantado no ambiente de produção, aceito e funcionando corretamente é bastante arriscado alterar alguma parte e comprometer todo o resto.

Uma equipe XP confia na eficácia de suas ferramentas e forma de trabalho para proteger o software destas falhas. “Assim ao invés de frear a criatividade do cliente e evitar mudanças, equipes XP as consideram inevitáveis e procuram se adaptar a elas com segurança e com coragem” (Teles, 2005).

4.3 FeedBack

A idéia principal é encurtar ao máximo o período de tempo entre uma ação executara pelos desenvolvedores e a observação deste resultado pelo cliente. Os desenvolvedores procuram entregar novas funcionalidades no menor prazo possível, para que o cliente compreenda rapidamente as conseqüências daquilo que pediu. E por sua vez o cliente procura manter-se próximo da equipe de desenvolvimento para prover informações precisas sobre qualquer dúvida que eles tenham ao longo do projeto.

4.4 Respeito

É o que dá sustentação a todos os demais valores, se não existir respeito pelo projeto, pelas etapas os componentes e pessoas envolvidas, não há nada que possa salvá-lo.

4.5 Simplicidade

O conceito de simplicidade objetiva assegurar que a equipe se concentre em fazer, primeiro, apenas aquilo que é claramente necessário e evite percas de tempo com o que não é essencial.

4.6 Praticas do Extreme programming

Além dos princípios já abordados o que interessa realmente é as praticas que descrevem como é que uma equipe XP trabalha.

Práticas representam aquilo que as equipes XP fazem diariamente, é aqui que os valores são aplicados.

Uma das primeiras práticas observadas é a utilização de um ciclo de vida do sistema iterativo, mas com ciclos um pouco diferentes e mais curtos.

As práticas do XP são divididas em dois tipos: As **Praticas Primárias**, as quais aplicam realmente a filosofia de trabalho e as **Praticas Corolárias** que complementam o exercício deste tipo de desenvolvimento.

4.6.1 Práticas primárias

4.6.1.1 Ambiente Informativo

Para atingir os objetivos são utilizados diversos instrumentos tais como:

- Cartões com histórias colocados em um mural.
- Quadro branco.
- Gráficos com informações relevantes para a equipe.
- Post it sobre as paredes.
- Flip charts.

Cujo o principal objetivo como já percebido é o melhor desempenho da comunicação.

4.6.1.2 Ciclo semanal.

O software é desenvolvido de modo iterativo e incremental, ou seja, uma vez por semana os desenvolvedores se reúnem com o cliente para priorizar um pequeno conjunto de funcionalidades que possam ser implementadas e testadas completamente naquela semana.

Terminando essa iteração, o cliente tem a oportunidade de utilizar e avaliar o que foi produzido. Com base nos resultados, reúne-se novamente com a equipe e estabelece novas prioridades de acordo com o que acabou de aprender com o software e com aquilo que já imaginava ser necessário produzir ao longo do restante do projeto.

Figura 3.2 – Fotografia de uma reunião – Fonte: Apresentação de Vinícius Teles sobre XP

Nesta reunião ilustrada pela figura 3.2 que recebe o nome de **Jogo do Planejamento,** o cliente tem o direito de informar as funcionalidades, também chamadas de **histórias** além de indicar a prioridade das mesmas, Enquanto os desenvolvedores podem estimar e apresentar suas considerações técnicas.

O objetivo é criar um plano para uma semana de trabalho, que seja capaz de gerar (através de funcionalidades) o máximo possível de valor para o negócio do cliente, naquela semana.

4.6.1.3 Histórias

São funcionalidades do sistema geralmente escritas em conjunto ou pelo próprio cliente no Jogo do planejamento é expresso em **cartões** o que o cliente deseja daquele módulo do sistema e também pode ser estimulado um **prazo** e uma **prioridade**.

Toda a iteração semanal é baseada nestes cartões que não precisam necessariamente ser concluídos em uma semana, mas sim de acordo com o prazo estipulado, eles são presos no mural e acompanhados diariamente pela equipe.

Figura 3.3 – Fotografia de uma reunião 2 – Fonte: Apresentação de Vinícius Teles sobre XP

No mural ilustrado pela figura 3.3 os cartões podem ser separados entre não **iniciados, em andamento, finalizados**, e acompanhados facilmente por todos os participantes durante a semana.

4.6.1.4 Programação em par

Todos os códigos produzidos no projeto são desenvolvidos por duas pessoas juntas, diante do mesmo computador, revezando-se no teclado. Uma digita enquanto a outra revisa e corrige, essa dupla ilustrada na figura 3.4 pode e deve sofrer rodízios freqüentemente, para não criar dependência.

Figura 3.4 – Programação em par – Fonte: Apresentação de Vinícius Teles sobre XP

Isso torna possível que qualquer integrante da equipe possa ser substituído em qualquer fase do projeto sem que haja dependência do projeto também facilita a integração e treinamento de novos programadores além da menor incidência de erros nos códigos.

“Quem trabalha continuamente com programação em par se habitua a corrigir e ter seu trabalho corrigido dezenas de vezes ao dia. A incidência de erros identificados pelo colega costuma ser tão elevada que surpreende quem não está acostumado ao uso da técnica” (Teles, 2005).

Este tipo de prática traz maior responsabilidade, simplicidade, pressão do par, menor incidência de erros, confiança, velocidade além de disseminação do conhecimento.

Não é qualquer um que se habitua a esse tipo de programação. Programar em par exige que as pessoas envolvidas sejam receptivas, compreensivas, engajadas e humildes. É necessário aceitar que somos falíveis para programar em par.

4.6.1.5 Trabalho organizado

Para evitar inúmeros problemas como ter que fazer horas-extras para com freqüência para reduzir atrasos e tentar entregar o software em um prazo razoável.

Não se pode esquecer que programadores são seres humanos e se estressam aumentando assim as probabilidades de cometerem erros.

Deve haver muita organização das tarefas para evitar tais desgastes, muita atenção nas prioridades e prazos, por isso o ciclo semanal tem sido muito eficaz.

Figura 3.5 – Bagunça – Fonte: Apresentação de Vinícius Teles sobre XP

Figura 3.6 – Organização – Fonte: Apresentação de Vinícius Teles sobre XP

A figura (Figura 4.5) e (Figura 4.6) ilustram como e onde um ser humano prefere desenvolver seu trabalho.

4.6.2 Práticas Corolárias

As praticas do XP buscam encontrar a raiz do problema, segundo (Teles, 2005) “resolver os sintomas não soluciona o problema”.

4.6.2.1 Código coletivo

Os pares em um projeto XP se revezam na formação, todos têm acesso e autorização para editar qualquer parte do código da aplicação, a qualquer momento. A prioridade do código é coletiva e todos são responsáveis por todas as partes.

4.6.2.2 Envolvimento do Cliente

Há uma divisão de responsabilidades, decisões técnicas são tomadas pelos desenvolvedores e decisões de negócio pelo pessoal de negócios. Para isso projetos XP contam com uma participação ativa de uma ou mais pessoas do negócio, que definem e priorizam as funcionalidades que devem ser implementadas.

4.6.2.3 Reunião em Pé

Uma breve reunião realizada todos os dias, geralmente pela manhã, com a equipe de desenvolvimento que visa compartilhar informações sobre o projeto e priorizar suas atividades. Esta reunião pode contar também com a presença do cliente, por isso deve ser rápida e objetiva sem entrar em muitos detalhes, mas apenas para compartilhar os eventuais problemas ocorridos no dia anterior e as possíveis soluções.

Esta pratica trás um maior entrosamento da equipe e um envolvimento e valorização do trabalho realizado.

É feita em pé por inúmeros motivos apontados por estudos psicológicos, um deles justamente é a objetividade e agilidade para não desperdiçar o tempo dos envolvidos.

5 POO em Equipes

A Programação Orienta ao Objeto é atualmente utilizada em todos os meios comerciais de produção de software, o que acelera e facilita o processo de desenvolvimento de software principalmente em ambientes gráficos e ambientes Web.

A linguagem Java vem ganhando cada vez mais espaço por permitir uma completa integração com este tipo de programação entre outras linguagens.

A dinâmica deste tipo de programação, graças ao conceito de classes faz com que vários programadores possam participar da criação de uma aplicação sem interferir diretamente um no trabalho do outro, permitindo assim um dinamismo maior na distribuição das tarefas em um determinado projeto.

A programação em camadas principalmente quando se fala em cliente/servidor na Web, onde uma parte do projeto é a conexão com o banco de dados, outra a conexão com o servidor de aplicação e outra a interface com o usuário faz com que várias pessoas ou até inúmeras equipes possam trabalhar em camadas diferentes e interagir nestes espaços.

Em alguns casos este tipo de separação de camadas pode acarretar maiores gastos com equipamentos e mais investimentos, mas a qualidade do resultado final compensa estes investimentos.

Os principais benefícios destas práticas são:

A especialização dos funcionários.
A integração de todos da equipe com o projeto que está sendo executado.
Maior comprometimento da equipe.
Melhores resultados e mais qualidade no produto final.

Prova de que este tipo de programação é um sucesso são todos os softwares livres que utilizam deste tipo de conceito de equipe e formam uma comunidade

global de programadores que muitas vezes desenvolvem por puro prazer, sem nenhum fim lucrativo, talvez indireto, mas que é um sucesso mundial.

Muitos dispositivos eletrônicos, graças ao conceito de máquina virtual que a linguagem Java traz, são produzidos com várias funcionalidades e são lançados no mercado abertos a novas funcionalidades que podem ser programadas por seus usuários, e se úteis ou aceitas podem ser compartilhadas entre outros, criando assim uma cooperação entre fabricante e cliente indiretamente.

O conceito de “classes” utilizado na OO é o principal motivo pelo qual o trabalho em equipe é mais eficiente, o fato da estruturação da lógica da classe, da visualização e entendimento por parte do UML e as características de conectividade e herança destes códigos, torna o trabalho mais claro e com conclusão objetiva.

Um programador pode simplesmente se deparar com um problema, criar uma lógica que resolve este, e colocar a solução em uma classe que poderá ser utilizada por outros programadores facilmente sem que estes se preocupem com como foi resolvido o problema, mas somente em utilizar esta solução e continuar o seu trabalho. Com o passar do tempo inúmeras soluções diferentes podem ser implementadas, então é só escolher a que for mais conveniente para cada situação.

6 Conclusão

Um projeto de Tecnologia da Informação geralmente só tem um “Inicio” e raramente um “Fim”, mas sim uma nova versão e outra em seguida e assim por diante. Uma equipe que inicia um projeto mesmo que este seja um sucesso, não será a mesma na próxima e na seguinte versão deste projeto. Muitos novos integrantes podem iniciar nesta equipe e continuar o trabalho.

As praticas do Extreme Programming como programação em par, reuniões diárias aliadas à facilidade de compreensão dos diagramas do UML e do conceito de classes trazidos pela POO, fazem com que a integração de novos membros as equipes de desenvolvedores fique mais fácil e o projeto em si flua mais facilmente.

Esta pesquisa procurou demonstrar que uma boa forma de programação em equipes é a Programação Orientada ao Objeto aliada as práticas do Extreme Programming.

Com uma boa organização, bons princípios de trabalho em equipe, foco no objetivo do projeto, capacitação e bom uso das inúmeras ferramentas que atualmente estão no mercado, às probabilidades de sucesso são bem maiores.

Bibliografia

Gestão Estratégica Pública - Turma 2005 - uma parceria entre a **Escola de Extensão da Unicamp e a Escola de Governo** e Desenvolvimento do Servidor da Prefeitura Municipal de Campinas.
<http://www.preac.unicamp.br/cap_10_trabalho_em_equipes.pdf> Acesso em: 28 de Abril de 2009.

FERREIRA, Aurélio Buarque de Holanda – **Dicionário Aurélio** – 1988.

RODRIGUES DA SILVA, Alberto Manuel e ESCALEIRA VIDEIRA, Carlos Alberto. **UML, Metodologias e ferramentas CASE** – Portugal / 2001.

SANRA RAHAL JÚNIOR, Nelson Abu. **Artigo Científico Programação Orientada ao Objeto** - Doutorando em Ciência da Computação (UFSC)
<http://www.ccuec.unicamp.br/revista/infotec/artigos/leite_rahal.html> Acessado no dia 27 de Abril de 2009.

TELES, Vinícius Manhães. **Extreme Programming: Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade** – 2005.