UNIVERSIDADE EDUARDO MONDLANE

Faculdade de Ciências

Departamento de Matemática e Informática

Trabalho de Licenciatura

TEMA:

USDP aplicada na criação do Modelo Conceptual do Sistema de Processamento de Cadernos Eleitorais

Autora: Rocina Abel Jonaze

Maputo, Julho de 2004

UNIVERSIDADE EDUARDO MONDLANE

Faculdade de Ciências

Departamento de Matemática e Informática

Trabalho de Licenciatura

TEMA:

USDP aplicada na criação do Modelo Conceptual do Sistema de Processamento de Cadernos Eleitorais

Supervisores:

dr. Marcelo Munguanaze

dr. Carlos Cumbana

Autora: Rocina Abel Jonaze

Maputo, Julho de 2004

A minha família em geral, á memória dos meus avós, meu sobrinho e ao Vasco.

Agradecimentos

O presente trabalho resulta da união de esforços humanos, técnico - científicos e materiais.

Muitas pessoas e instituições, directa ou indirectamente contribuíram para a realização do mesmo, para todas elas endereço os meus agradecimentos.

O meu obrigado aos Departamentos de Informática e Sufrágio do STAE- Secretariado Técnico de Administração Eleitoral, em especial ao dr. Orlando d' oliveira Comé e Sr.ª Helena Garrine, pela oportunidade oferecida na elaboração deste trabalho dentro da instituição.

Agradecimentos especiais vão para os meus supervisores dr. Marcelo Munguanaze e dr. Carlos Cumbana, pela orientação metodológica e a valiosa ajuda ao longo da investigação e elaboração deste trabalho.

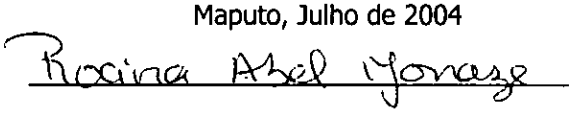
Aos meus colegas Altino Matias, Ovídeo Zavale e Artur Gani, pelo apoio moral e material nas fases críticas e pelas sugestões no decorrer do trabalho.

E por fim, um agradecimento muito especial aos meus familiares, que sempre estiveram do meu lado.

Declaração de Honra

"Declaro por minha honra, que este trabalho é resultado da minha própria investigação, e o mesmo foi realizado apenas para ser submetido como trabalho de licenciatura em informática, na Universidade Eduardo Mondlane."

Maputo, Julho de 2004



(Rocina Abel Jonaze)

Resumo

À medida que os processos eleitorais ocorrem no país, tem-se verificado um conjunto de problemas que dificultam a normal realização das tarefas associadas, como é o caso de:

- Mudanças constantes da legislação Eleitoral;
- Má qualidade das fontes de dados (fichas) devidos ao fraco nível dos brigadistas, envolvido no processo de recenseamento e o tempo reduzido da sua formação.
- A falta de um sistema em on-line com as delegações do STAE Provinciais de modo a permitir a comparação da informação do eleitorado a nível nacional, assim como acesso rápido em tempo real a informação actualizada e o controlo adequado do número de eleitores recenseados até a altura da outra actualização para fins estatísticos.
- A não existência de um historial dos eleitores, considerando que os arquivos em papel são susceptíveis a deterioração ou desaparecimento, dificultando a gestão da informação do eleitor.
- A falta de ligação entre o Sistema de Processamento de Cadernos Eleitorais (SPCE) e todos outros sistemas de gestão de dados do cidadão nomeadamente, Sistema de produção do BI para identificar cidadãos que atingiram a idade eleitoral (Ministério do Interior), Sistema de Gestão da Justiça para identificar os cidadãos que perdem a capacidade eleitoral por razões judiciais ou de óbito (Ministério da Justiça), o que dificulta o processo de actualização dos dados dos eleitores.

Com base nestas constatações surgiu o presente trabalho, tendo como objectivo apresentação de um modelo conceptual para o sistema de processamento dos cadernos eleitorais, usando a metodologia USDP, Metodologia Processo Unificado de Desenvolvimento de Software (*Unified Software Development Process*), como Linguagem de modelação a UML (*Unified Modeling Language*).

Índice

Índice de Figuras

1. Introdução e Objectivos

1.1 Introdução

As Tecnologias de Informação e Comunicação (TICs) têm um papel relevante na definição da competitividade da organização. Tanto ao nível interno quanto ao nível externo, as TICs possibilitam obter uma melhoria na gestão da informação. Internamente, a administração dos recursos-materiais, humanos e financeiros-pode ser bem sucedida com a rapidez e precisão dos diversos recursos disponíveis das TICs. Externamente, estudos e análises do mercado, incluindo seus clientes, concorrentes e fornecedores, podem ser aprimorados mediante recursos das TICs.

Entre os diversos recursos que determinam o papel das TICs, destacam-se os Sistemas de Informação (SI). Um sistema de informação numa organização fornece informação útil aos seus membros e clientes, auxiliando várias operações de forma eficaz e eficiente. Esta informação pode relacionar-se aos clientes, fornecedores, produtos, equipamentos, etc.. Tais SI podem ser modelados pois, a modelação é a parte central de todas as actividades que levam a implantação de um bom sistema [Jacobson et al., 2000].

Somente com auxílio da modelação pode visualizar e controlar o desenvolvimento do sistema de maneira eficaz, identificando e gerindo riscos, estipulando e cumprindo prazos, dentro de estimativas de custo.

Partindo deste princípio várias metodologias para o desenvolvimento de sistemas foram criadas. As primeiras metodologias, classificadas como metodologias estruturadas, caracterizam o desenvolvimento de sistemas em torno de procedimentos e funções.

Muitos sistemas ainda hoje são desenvolvidos com base em metodologias estruturadas, o que os torna instáveis, pelo facto de que a medida em que os requisitos se modificam (o que acontece com muita frequência) e o sistema cresce, o trabalho de manutenção do mesmo se torna cada vez mais difícil.

Com a evolução das metodologias, a visão de desenvolvimento de sistemas passou a adoptar uma perspectiva diferente. Surgiram então as metodologias orientadas a objectos, que caracterizam o desenvolvimento de um sistema em torno de classes e objectos.

O sucesso dos métodos orientados a objectos foi factor primordial para que o número de metodologias criadas sob essa perspectiva crescesse de maneira desordenada num curto espaço

de tempo. Inúmeros métodos surgiram, cada um com suas falhas, e sem nenhum relacionamento com os demais. Desta forma, os usuários destes métodos sentiram dificuldades em encontrar uma linguagem capaz de atender inteiramente as suas necessidades, o que ocasionou a chamada guerra de métodos [Jacobson et al., 1999].

Desta forma, através do empenho de Jacobson, Booch e Rumbaught, surgiu a Linguagem de Modelação Unificada, a UML.

Porém, a UML é apenas parte de uma metodologia, uma linguagem de modelação de sistemas, ela apenas disponibiliza as ferramentas necessárias para se criar e ler modelos, não aponta quais modelos deverão ser criados, nem quando deverão ser criados. Essa tarefa é da responsabilidade de um processo de desenvolvimento de sistemas.

A UML (Unified Modeling Language) foi usada como técnica de modelação por se tratar de uma linguagem-padrão para a elaboração da estrutura de projectos de software orientado a objectos que facilita a visualização, especificação, construção e a documentação de sistemas de software [Fumo, 2001]. Mesmo sendo independente de processo, a UML necessita de interagir com uma metodologia específica para se obter o máximo proveito dos seus recursos, daí que neste trabalho recorremos a USDP. Com a integração do processo unificado à UML, as respostas para "**quem está fazendo o quê**?", "**quando**?" e "**como**?" puderam ser definidos e um padrão no desenvolvimento de sistemas orientados a objectos poderá ser estabelecido.

Neste trabalho descreve-se a metodologia USDP, que tem como ideia central o uso da tecnologia de objectos, incluindo a análise e desenho orientada a objectos, e aplica-se na modelação de algumas funcionalidades do SPCE.

O SPCE será um dos recursos do STAE-Instituição Administrativa do Estado criada para a planificação, organização, execução de processos eleitorais e referendos. Um processo eleitoral compreende, entre outras actividades o recenseamento eleitoral, o sufrágio, o processamento dos cadernos eleitorais, etc. e ainda a actualização de dados dos eleitores para sufrágios subsequentes.

Os resultados deste trabalho têm uma magnitude académica e outra prática já que por um lado, estuda-se e compreende-se a metodologia USDP com o intuito de aplicá-la num caso prático, por outro lado a modelação de um sistema real-SPCE. Este sistema reveste-se de extrema importância na medida em que vai minorar os problemas relacionados com, a gestão dos dados dos eleitores.

1.2 Objectivos gerais e específicos

Constituem objectivos do trabalho os seguintes:

1.2.1 Objectivo geral

- Criar o Modelo conceptual do sistema de processamento de cadernos eleitorais aplicando a fase da concepção e elaboração do USDP.

1.2.2 Objectivos Específicos

Para o alcance do objectivo geral do presente trabalho, foram definidos os seguintes objectivos específicos:

- Analisar a evolução de metodologias de desenvolvimento de sistemas e o enquadramento da USDP;
- Desenvolver o modelo conceptual do SPCE;
- Aplicar as fases da concepção e Elaboração da USDP e a UML no desenvolvimento do modelo do SPCE.

1.3 Resultados Esperados

Com a concepção do modelo conceptual do SPCE, espera-se os seguintes resultados:

1. Um modelo de base de dados dos eleitores;
2. Possibilidade de emissão de cadernos informatizados com o mínimo de erros;
3. Facilidade de actualização que consiste em:
    - Eliminação de eleitores que perderam a capacidade eleitoral activa por várias razões;
    - Emissão de segundas vias;
    - Tratamento de transferências;
    - Inserção de novos eleitores.

1.4 Estrutura do Relatório do Trabalho

Para um melhor entendimento, este relatório foi dividido em seis (6) capítulos, sendo o primeiro capítulo reservado a introdução. Neste apresenta-se os objectivos gerais e específicos, resultados esperados e as anotações e convenções usadas.

O segundo capítulo trata da Metodologia do Trabalho usada no trabalho. Segue-se o terceiro capítulo que apresenta a revisão bibliográfica, para uma melhor compreensão do enquadramento da metodologia USDP, no capítulo 3 é apresentada uma revisão bibliográfica da evolução das metodologias.

O capítulo quatro (4) aborda o Processo Unificado envolvendo a definição de suas características principais, fases e disciplinas.

Procurando aplicar de maneira prática os conceitos do Processo Unificado, destacando as disciplinas de requisito e análise, o capítulo cinco (5) apresenta a aplicação desta metodologia no caso de prático do Sistema de Processamento dos cadernos Eleitorais.

Por fim, o capítulo seis (6) contém as conclusão e recomendações do trabalho.

1.4.1 Notações e convenções usadas

A fonte utilizada para os caracteres no presente trabalho é a Tahoma e o tamanho é 10.

Os títulos dos capítulos apresentam o tamanho 16 e os sub-capítulos 14 e estão em negrito.

Todos os termos na língua inglesa que aparecem em itálico, são palavras cujo significado na língua portuguesa não foi possível encontrar ou elas são frequentemente usadas com tal nome.

2. Metodologia do trabalho

A presente pesquisa será realizada em três etapas básicas:

a) Através de uma revisão bibliográfica na área de metodologias de desenvolvimento de sistemas de informação, apresentar-se-á um resumo de alguns elementos tais como técnicas e ferramentas das metodologias Estruturadas e Orientadas a Objecto. Estes elementos permitirão a autora um melhor entendimento da metodologia USDP.

b) Ainda com base na revisão bibliográfica, apresentar-se-á todas as fases da USDP que permitem construir um sistema, mas salientar que neste trabalho apenas serão aplicadas as fases de Concepção e Elaboração.

c) Usando USDP, modelamos o caso de estudo descrito no ponto 5.1. Para a colheita de dados sobre o sistema recorreu-se à documentação existente nos diversos Departamentos do STAE e entrevistas informais aos funcionários do mesmo, permitindo deste modo uma recolha mais abrangente de dados. Também no âmbito do trabalho fez-se uma observação participativa por forma a inteirar-se do funcionamento do sistema Informático existente no Departamento de Informática do STAE-CENTRAL.

A USDP é uma metodologia iterativa e é usada para o desenho de sistemas orientados a objecto. Na USDP o desenvolvimento é organizado numa série de mini-projectos com o tempo fixo (ex. 4 semanas), chamadas iterações. Cada iteração inclui a sua própria análise de requisitos, projrcto, implementação e testes.

Para além de permitir trabalhar de uma forma iterativa, a USDP permite descrever as actividades a serem executadas tais como escrever um caso de uso dentro de uma disciplina. Para a modelação foi usada a UML.

3. Metodologias de Desenvolvimento de Sistemas

3.1 O Conceito de Metodologia

O termo metodologia, apesar de ser amplamente utilizado, não possui uma definição amplamente aceite. A nível geral, entende-se como metodologia uma série recomendada de passos e procedimentos que devem ser seguidos para obter-se o desenvolvimento de um sistema de informação [Yourdon, 1995].

De acordo com Avison e Fitzgerald em 1997, é o conjunto formado por procedimentos, técnicas, ferramentas e documentação que auxiliará os responsáveis pelo desenvolvimento de sistemas em seus esforços na implementação de um novo sistema de informação. Uma metodologia consistirá de fases, cada uma consistindo de sub-fases, que orientarão estes responsáveis na escolha das técnicas que deverão ser mais apropriadas a cada estágio do projecto e também auxiliá-los a planear, gerir, controlar e avaliar o projecto do sistema de informação.

Avison Maddison Apud e Fitzgerald em 1997, definiram metodologia como sendo um conjunto recomendado de filosofias, fases, procedimentos, técnicas, regras, ferramentas, documentação, gestão e treinamento para o desenvolvimentos de um sistema de informação. Verifica-se neste conceito a inclusão, entre outros, de filosofias que são as teorias e crenças que norteiam os objectivos e procedimentos de uma metodologia.

Flynn em 1992, discordando dos autores anteriores, afirma que o termo metodologia não é apto no contexto de desenvolvimento de sistemas e que o termo método seria perfeitamente adequado para descrever o que vem sendo chamado de metodologia.

No entanto, considerando-se que o termo metodologia inclui certos elementos, como por exemplo a filosofia que orienta todo o processo, que não são considerados pelo conceito de método, pode-se concluir sua abrangência e adequação ao contexto de desenvolvimento de sistemas de informação.

Por outro lado, encontramos o conceito **processo**. Um processo de desenvolvimento de software pode ser visto como um guia para o desenvolvimento e deve estabelecer: as actividades a serem realizadas durante o desenvolvimento de software, sua estrutura e organização (decomposição e precedência de actividades), incluindo a definição de um modelo de ciclo de vida; os artefactos requeridos e produzidos por cada uma das actividades do processo; os procedimentos (métodos, técnicas e normas) a serem adoptados na realização das actividades;

De maneira geral, o ciclo de vida de um software envolve, pelo menos, as actividades de planeamento, levantamento de requisitos, análise, projecto, implementação, testes, implantação, operação e manutenção [Jacobson et al. 1999].

3.2 Técnicas e Ferramentas

Técnicas e ferramentas caracterizam as metodologias. Uma técnica é um modo de fazer uma actividade particular no processo de desenvolvimento de sistemas de informação e que podem vir a ser utilizadas por várias metodologias. Cada técnica pode envolver o uso de uma ou mais ferramentas que representam alguns dos artefactos usados no desenvolvimento de sistemas de informação. Ferramentas geralmente são automatizadas, isto é, são recursos computacionais - softwares que auxiliam neste desenvolvimento [McDermid, 1990].

Verifica-se que muitas técnicas são usadas em várias metodologias apesar deste fato não significar que elas sejam adequadas a todas as metodologias existentes. Geralmente são utilizadas em determinadas metodologias voltando-se para diferentes partes do processo de desenvolvimento, diferentes propósitos ou a diferentes objectos.

Uma das técnica de análise de processos é o ciclo de vida de entidades, comum em muitas metodologias. As técnicas de orientação a objectos buscam representar dados e processos do sistema em estudo. Os diagramas estruturais auxiliam a representar as estruturas hierárquicas de um programa computacional enquanto que matrizes são usadas para mostrar a relação entre duas coisas, como por exemplo, entidades e processos. E, há ainda outras técnicas categorizadas como lógica do processo que incluem técnicas de árvores e tabelas de decisão e diagramas de acção [Mason e Willcocks, 1994].

Muitas das técnicas acima citadas, referem-se a documentação dos processos ou actividades envolvidas no desenvolvimento. Essas técnicas de documentação podem ser utilizadas para comunicar os resultados alcançados a outros analistas, usuários, gerentes e programadores. Também podem ser usadas para auxiliar o processo de análise e projecto como também na verificação de que todos os passos da metodologia foram realizados

3.3 Necessidade de metodologias

Os primeiros sistemas de informação desenvolvidos, na década de 60, foram largamente implementados sem o auxílio de uma metodologia explícita de desenvolvimento de sistemas de informação. Nestes anos, as pessoas que implementavam sistemas de informação eram programadores que não tinham, necessariamente, habilidades adequadas de comunicação ou compreensão das necessidades dos usuários. Na realidade havia a preocupação com as habilidades técnicas dos programadores, deixando-se as demais para um segundo plano. Adicionalmente, os sistemas de informação desenvolvidos geralmente custavam muito mais como também demoravam além do esperado para serem colocados em uso.

Poucos programadores seguiam algum tipo de metodologia baseando-se, na sua maioria, na própria experiência. Modificações no sistema em desenvolvimento, devido a novas necessidades de seus usuários ou a uma deficiência na especificação inicial destas necessidades, levavam a efeitos indesejáveis e inesperados nas demais partes do sistema. Adicionalmente, o uso crescente da TI associado a necessidade gerencial por sistemas apropriados levou a situação onde tornou-se necessário um método capaz de orientar o desenvolvimento dos SI.

Na década seguinte verificou-se as seguintes mudanças:

O reconhecimento crescente de que parte do desenvolvimento de sistemas envolve análise, projecto e construção existindo, portanto, funções distintas de analista de sistema e de programador;

A consciencialização de que as organizações estavam crescendo em tamanho e complexidade, sendo mais desejável sistemas de informação integrados do que soluções específicas para os problemas de cada processo organizacional.

Em resposta ao cenário formado na década de 70, passou a ser utilizado um modelo metodológico para desenvolvimento de sistemas de informação, o modelo Waterfall. Também conhecido como análise de sistemas convencional ou ciclo de vida do desenvolvimento de sistemas, este modelo teve e tem um papel fundamental na área de sistemas de informação, sendo a base sobre a qual foram criadas a maioria das metodologias existentes [Shah e Avison, 1995].

3.4 A evolução das metodologias

3.4.1 O modelo Waterfall

Desenvolvido no final da década de 1960 e começo da década de 1970, o modelo Waterfall é ainda hoje a abordagem mais praticada no desenvolvimento de sistemas de informação. Esta abordagem assume que um sistema de informação tem um ciclo de vida semelhante ao de qualquer produto, com início, meio e fim. Cada etapa do ciclo de vida pressupõe actividades que devem ser completadas antes do início da próxima etapa.

Baseado nesta abordagem, Laudon e Laudon em 1996, apresentaram seis estágios que compõe o ciclo de vida de um sistema de informação apresentado a seguir:

- Fase 1: Definição de Projecto
- Fase 2: Estudo dos Sistemas
- Fase 3: Projecto
- Fase 4: Programação

- Fase 5: Instalação

- Fase 6: Pós - Implementação

Utilização e avaliação do sistema após sua instalação. Inclui actualizações, correcções, etc.

Outros autores, apresentam apenas quatro etapas: análise, projecto, codificação e teste. Nesta situação, algumas etapas englobam actividades que Laudon e Laudon (1996) distribuíram em duas, como por exemplo, a etapa da análise consistiria da definição do projecto e estudo do sistema. Esta diferença encontrada nas actividades relacionadas a cada estágio em nada modifica o modelo Waterfall.

Independentemente das diferenças observadas entre autores, o modelo Waterfall, possui características que o recomenda. Primeiramente este modelo já foi experimentado e testado extensivamente. Verificou-se, ao final de cada fase, a oportunidade dos analistas e usuários avaliarem o progresso obtido até aquele momento. Da mesma forma, a divisão do desenvolvimento de um sistemas em fases permitiu um controle maior sobre este desenvolvimento, facto que pode, teoricamente, representar melhores resultados, melhores sistemas de informação.

Mesmo sendo o modelo referencial da grande maioria das metodologias existentes, o modelo *Waterfall* apresenta algumas limitações que devem ser ressaltadas:

- Uma vez que considera que uma etapa deve ser iniciada após a conclusão das actividades da etapa anterior, é gasto uma quantidade razoável de tempo e esforços levantando informações, especificando e documentando-as a cada etapa para sua posterior utilização. Este facto pode levar a demora na instalação do sistema tornando-o, muitas vezes, obsoleto quando efectivamente colocado em operação.

- Devido a seu grau de formalidade que exige especificações e documentações para cada processo que o sistema de informação executa, alterações são inibidas tornando este processo, muitas vezes, inflexível à mudanças. O próprio processo de detecção de erros no ciclo de vida em cascata clássico é reservado à fase de teste formal do projecto. Neste estágio, a pressão nas actividades finais de desenvolvimento do sistema como detecção de erros de análise e projecto, levam a situações onde torna-se difícil a correcção destes tendo em vista o custo associado a eliminação dos erros.

- Os processos de tomada de decisão exigem, na maioria das vezes, actividades não estruturadas, que não possuem procedimentos bem definidos. Esta realidade, dentro de uma abordagem tradicional – e formal – dificulta a definição das especificações do sistema, dependendo de requisitos correctos e estáveis.

- Uso da implementação *bottom-up*. A implementação inicia seu trabalho testando os módulos do sistema, depois subsistemas e finalmente o sistema. Assim, os erros mais sérios (integridade do sistema) são encontrados no final e não no início da fase de testes.

3.4.2 Metodologias Estruturadas

Nos últimos anos da década de 60, com a introdução da programação estruturada, surgiram as técnicas estruturadas. Nos anos seguintes, já na década de 70, Steven Myers e Constantine propuseram os conceitos do projecto estruturado. Buscou-se então aplicar os conceitos de projecto estruturado à análise de sistemas com o objectivo de desenvolver um método de especificação de requisitos adequado ao projecto estruturado.

Gane e Sarson em 1979, e o Yourdon em 1992 apresentaram metodologias que integravam a análise e o projecto estruturado. As metodologias de Gane e Sarson e de Marco são bastante semelhantes entre si sendo suas principais diferenças encontradas na simbologia das técnicas empregues.

No que se refere ao trabalho de Gane e Sarson, estes descreveram brevemente, em 1979, a metodologia de desenvolvimento de sistemas chamada **STRADIS**-*Structured Analysis, Design and Implementation Systems* - que incorporou a analise, projecto e implementação estruturada de sistemas de informação. É uma metodologia baseada na decomposição funcional e no uso do diagrama de fluxo de dados. Suas fases principais são:

- Fase 1: Estudo inicial
- Fase 2: Estudo detalhado
- Fase 3: Definição e projecto de soluções alternativas
- Fase 4: Projecto Físico

Gene e Sarson acrescentam que para completar o desenvolvimento do sistema outras actividades são necessárias como, por exemplo, desenhar um planeamento incluindo planos de teste e aceitabilidade do sistema.

Muito semelhante à STRADIS, a Metodologia Estruturada Moderna, proposta por Yourdon em 1992, possui como base a decomposição funcional e o projecto *top-down*, através dos quais um problema é sucessivamente decomposto em unidades de fácil gestão.

Entretanto, apesar de basear-se no método estruturado, particularmente em suas técnicas de modelação, versões mais recentes desta metodologia conhecida por **YSM**-*Yourdon Systems Method*, foram descritas por Yourdon em 1993, usando um método conhecido como particionamento de evento.

Este método não é um puro *top-down* nem *bottom-up* mas chamado de *middle-out*. O analista começa desenhando um diagrama de contexto de alto-nível que indica as fronteiras do sistema. Então, após entrevistas com usuário, elabora-se uma lista textual de eventos do ambiente no qual o sistema deverá responder.

Existem três principais fases nesta metodologia que são:

- Fase 1: Estudo de Viabilidade
- Fase 2: Construção do Modelo Essencial
- Fase 3: Construção do Modelo de Implementação

3.4.3 Metodologias Orientadas a Objectos

As técnicas de programação orientadas a objectos foram discutidas pela primeira vez no final da década 60. As primeiras metodologias de projecto orientadas a objecto, ainda que não tivessem este nome, foram descritas por Michael Jackson e Jean-Dominique Warnier na década de 1970 [Avison e Fitzgerald, 1997].

Para Barbieri em 1997, um dos factores críticos encontrados no desenvolvimento de sistemas com orientação a objectos é a falta de definição de uma linha metodológica a ser seguida no projecto. Uma vez que a orientação a objectos nasceu das linguagens orientadas a objectos, eram estas que, inicialmente, desempenhavam o papel de metodologias de sistemas.

Posteriormente, uma grande diversidade de propostas metodológicas foram apresentadas por James Rumbaugh, Grady Booch, Ivar Jacobson, Coad-Yourdon, Shlaer-Mellor, entre outras [Furlan, 1997].

Coad e Yourdon em 1991 citam algumas mudanças ocorridas ao longo da última década que justificam o crescente interesse por abordagem orientada a objectos:

- Os conceitos fundamentais de uma abordagem orientada a objectos tiveram uma década para amadurecer e a atenção voltou-se gradualmente de questões de codificação para questões de projecto e análise;
- A tecnologia fundamental para a construção de sistemas tornou-se muito mais poderosa;
- Os sistemas actualmente construídos são muito mais complexos e estão sujeitos a mudanças constantes, em relação aos sistemas de dez, vinte anos atrás;

As características fundamentais de uma metodologia orientada a objectos incluem a abstracção de dados: Em vez da decomposição funcional, ou da abstracção de procedimentos, que são características predominantes das técnicas estruturadas, as metodologias orientadas a objectos enfatizam a

abstracção de dados. Neste sentido são compatíveis com a modelação de dados e com as metodologias de engenharia da informação. A abstracção de dados também inclui o conceito de super-classe e subclasse que, embora presentes em muitas das metodologias de modelação de dados mais antigas, frequentemente não são tão amplamente desenvolvidas.

Desta abstracção e segundo Shlaer e Mellor em 1990 decorrem os conceitos de objecto:

Objecto - que é uma abstracção de um conjunto de coisas do mundo real, de forma que:

Atributo - é a abstracção de uma única característica possuída pelas entidades (objectos).

Relacionamento - é uma abstracção de um conjunto de associações que existe no mundo real. Existem vários tipos de relacionamentos classificados de acordo com a sua multiplicidade.

Encapsulamento - A essência da abordagem orientada a objectos está relacionada ao conceito de empacotar hermeticamente ou encapsular dados e funções juntos de tal forma que o único meio de obter acesso ou actualizar os dados seja invocando as funções associadas.

Herança - As metodologias orientadas a objectos permitem que um objecto "herde" tanto os atributos de dados como as funções dos objectos a partir dos quais foi criado.

Comunicação por meio de mensagens - Em metodologias de decomposição funcional mais antigas, a arquitectura de comunicação baseia-se em mecanismos de chamada conhecidas por sub-rotinas. Num sistema orientado a objectos, a arquitectura consiste numa rede assíncrona de objectos que se comunicam enviando mensagens.

4. Processo Unificado de Desenvolvimento de Software

"Um processo é um conjunto de passos que definem **quem** faz o **quê, quando** e **como** para alcançar determinado objectivo" [Jacobson et al., 1999]. Na engenharia de software, este objectivo é o de entregar de maneira eficiente e previsível, um produto capaz de atender às necessidades do seu negócio.

A UML é apenas parte de um método para o desenvolvimento de sistemas, e o facto de ser independente do processo permite que possa ser usado por vários processos de engenharia de software.

Entretanto, a adopção do Processo Unificado, como o processo responsável pelo desenvolvimento de sistemas modelados em UML, é mais adequada, tendo em vista que este processo está directamente ligado à UML, utilizando-a como notação de uma série de modelos que compõem os principais resultados das actividades do processo [Jacobson et al., 1999].

Além disso, os principais responsáveis pela criação do Processo Unificado foram Grady Booch, James Rumbaugh e Ivar Jacobson, os mesmos desenvolvedores da UML.

Este capitulo fará uma abordagem sobre o Processo Unificado (USDP), definindo suas características, assim como os seus ciclos de desenvolvimento, as disciplinas e as fases que compõem o ciclo e as disciplinas das iterações que subdividem as fases do ciclo de vida do desenvolvimento do software.

O software é sempre parte de um sistema (ou negócio) maior, o trabalho começa pelo estabelecimento dos requisitos para todos os elementos do sistema e, na sequência, procede-se a alocação para software de algum subconjunto destes requisitos. Esta etapa é a Engenharia de Sistemas e antecede a todas as demais relacionadas. Os desenvolvedores de software necessitam de um guião organizacional que os permita desenvolver sistema de software, como é o caso do Processo Unificado de Desenvolvimento de software. Este processo representa a compilação de metodologias previamente separadas. Não unifica simplesmente o trabalho dos três autores, Ivar Jacobson, Grady Booch e James Rumbaugh, mas incorpora inúmeras contribuições de outros indivíduos e companhias que contribuíram para a UML, assim como um número significativo de contribuintes chaves, tal como a *Rational Software Corporation*, que se serviu da experiência de centenas de organizações utilizadoras, trabalhando com versões anteriores do processo unificado.

O processo unificado de desenvolvimento de software poderá ser usado por qualquer um envolvido no desenvolvimento de software. É primariamente destinado aos membros da equipe

desenvolvedora que lidam com os requisitos das actividades do ciclo de vida, análise, desenho, implementação e teste. Isto é, um trabalho que resulta dos modelos da UML.

A necessidade do processo, torna-se cada vez mais crítica, particularmente em companhias ou organizações em que os sistemas de software são uma tarefa "critica" tais como, financeira, controle do tráfego aéreo, defesa e sistemas de telecomunicações. Isto significa que a condução com êxito dos negócios ou a execução da função pública dependem do software que lhes serve como base.

Estes sistemas de software estão a tornar-se cada vez mais complexos. O tempo de publicação carece de redução e o desenvolvimento está a tornar-se cada vez mais difícil. Por razões como estas a opção por um software, necessita de um processo que guie os desenvolvedores tal como uma orquestra precisa do compositor de notas para guiar a actuação.

A USDP, é um produto final de três décadas de desenvolvimento e uso prático. O seu desenvolvimento segue um caminho, desde a *Objectory Process* (que foi lançada em 1987) através da *Rational Objectory Process* (lançado em 1997) até a *Rational Unified Process* (lançado em 1998). O seu desenvolvimento foi influenciado por muitas fontes; Não serão identificadas todas as fontes, somente iremos nos referir ao impacto das abordagens do Ericson e da *Rational* em relação ao produto.

Em 1987 Ivar Jacobson deixou a Ericson e criou a *Objectory* em Estocolmo. Durante os 8 anos seguintes ele e os seus associados desenvolveram um produto de processos chamado *Objectory*, uma abreviatura de *Object Factory*, expandiram-na para indústrias fora das comunicações e para países fora da Suécia.

O desenvolvimento do processo *Objectory* continuou numa série de lançamento a partir da *Objectory 1.0* em 1988 para a primeira versão *On - Line, Objectory 3.8* em 1995.

A Rational Software Corporation, adquiriu a *Objectory* nos finais de 1995 e a tarefa de unificar os princípios básicos subjacentes aos processos de desenvolvimento de software existentes ganhou uma nova dinâmica. A *Rational* tinha desenvolvido um número de práticas de desenvolvimento de *software* muitas das quais eram complementares a aquelas que eram propriedade da *Objectory*. Por exemplo: "Em 1981 a *Rational* propôs-se a produzir um ambiente interactivo que iria melhorar a produtividade para o desenvolvimento de um largo sistema de *software*".

Em 1990 Mike Deven, escreveu a sua visão sobre a arquitectura centrada (*architecture-driven*) do processo de desenvolvimento iterativo.

Na altura da fusão, entre 1995 -1997 *Rational Objectory Process, Objectory 3.8* tinha mostrado como é que o processo de desenvolvimento de *software*, como produto, podia ser desenvolvido e modelado, desenhou também o processo de desenvolvimento de uma arquitectura de *software*, um grupo de modelos com registo dos resultados do processo. Em áreas como a modelação do caso de uso, a análise e desenho estava verdadeiramente bem desenvolvida. Noutras áreas, a gestão dos requisitos que não sejam os casos de uso, implementação e teste, estavam muito menos desenvolvidos pois continham pouco em relação a gestão do projecto, gestão da configuração, a preparação do desenvolvimento do ambiente (ferramentas e intersecção de processos) e desenvolvimento.

As experiências práticas da *Rational* juntaram-se para formar o processo da *Rational* e *Objectory 4.1*

Nos meados de 1998, o processo *Rational Objectory* tinha se tornado um processo completamente auto-suficiente capaz de suportar todo o ciclo de vida de desenvolvimento de software. Deste modo unificou uma grande variedade de contribuições não só pelos três autores presentes mas por muitas fontes das quais a *Rational* e a UML foram capazes de mobilizar. Em Junho a *Rational* lançou uma nova versão do produto, *Rational Unified Process 5.0* [Philippe Kruchten e Addison-Wesley, 1998]. Muitos elementos proprietários deste processo tornaram-se disponíveis ao público em geral.

4.1 Principais Características da USDP

São três as características que fazem com que o Processo Unificado seja único; ser dirigido por um caso de uso, estar centrada na arquitectura, ser iterativo e incremental. Estas três características relacionam-se e são igualmente importantes. A arquitectura fornece a estrutura que guia o trabalho de cada iteração, os casos de uso definem as metas e dirigem o trabalho de cada iteração [Jacobson et al., 1999]. Remover uma das três características reduziria drasticamente o valor do processo unificado. Estas três características funcionam como o tripé onde o processo é unificado e apoiado, se uma das pernas deste tripé for retirada, o processo cai.

A USDP é um processo de desenvolvimento de *software*. O processo de desenvolvimento de *software* é o conjunto de actividades necessárias para transformar as exigências dos requisitos dos utilizadores num sistema de *software*. Contudo, o processo unificado, criado pela *Rational* (*Rational Unified Process-RUP*) para apoiar o desenvolvimento orientado a objecto, fornecendo uma forma sistemática para se obter vantagens reais no uso da UML, é muito mais do que um processo, é um processo de cenário genérico que pode ser especializado para a classe de sistema de *software* para diferentes áreas de aplicação, diferentes tipos de organizações, níveis de competência diferentes e projectos de tamanhos diferentes.

A USDP é uma base de componentes, o que significa que o sistema de *software* em construção é constituído por componentes de *software*, por via interconectada e interfaces bem definidos. O processo unificado de desenvolvimento de *software* usa a UML durante a preparação dos planos do sistema de *software*. De facto a UML é uma parte integrante da USDP, foram desenvolvidos par a par. Contudo os aspectos realmente distinguidos do processo unificado são visualizáveis nas três palavras chaves.

A seguir para além da descrição das três palavras chaves irá se descrever de forma breve a abordagem genérica sobre o processo, o seu ciclo de vida, fases, iterações, disciplinas e artefactos.

4.1.1 O Processo Unificado é dirigido por caso de uso

Um processo orientado por casos de uso faz com que um sistema seja desenvolvido sob a perspectiva de atender, especificamente, às necessidades de cada usuário que interage com o sistema, evitando, desta forma, que o sistema possa ser desenvolvido a ponto de apresentar funcionalidades desnecessárias. Entretanto, casos de uso não são ferramentas ligadas apenas à especificação de requisitos do sistema, mas também ao projecto, implementação e testes do mesmo, ou seja, o processo de desenvolvimento do sistema é orientado por casos de uso. Um exemplo, da iteração é um ser humano que usa uma máquina multi-banco, ela ou ele introduz o cartão e responde as questões colocadas pela máquina no seu écran e recebe as somas inumerárias em resposta ao cartão e as respostas do utilizador.

Ser orientado por casos de uso significa que o processo de desenvolvimento segue uma disciplina, ou seja, o processo passa por uma série de disciplinas de trabalho que derivam dos casos de uso.

Desta forma, os modelos de análise, projecto e implementação são criados pelos desenvolvedores com base no modelo de casos de uso. Este processo é conhecido como realização de casos de uso e é evidenciado durante todo o ciclo de vida da USDP.

Casos de uso são desenvolvidos de acordo com a arquitectura do sistema. Assim, eles definem a arquitectura do sistema, e esta, por sua vez, influência a selecção dos casos de uso. Consequentemente, tanto a arquitectura do sistema quanto os casos de uso, evoluem durante o ciclo de vida do sistema.

4.1.2 O Processo Unificado é centrado na arquitectura

"A arquitectura é a visão de todos os modelos que juntos representam o sistema como um todo" [Jacobson et al. 1999].

O conceito de arquitectura de *software* engloba os aspectos estáticos e dinâmicos mais significantes do sistema. A arquitectura cresce além das necessidades do empreendimento, como percebido pelos usuários e suporte, e reflectido nos casos de uso.

De qualquer modo, a arquitectura também é influenciada por muitos outros factores, como a plataforma na qual o *software* será implantado, os blocos de construção reutilizáveis, requisitos de desenvolvimento e requisitos não - funcionais.

A arquitectura é uma visão do projecto do sistema como um todo, destacando suas características mais importantes, mas sem entrar em detalhes. Considerando que "o que é significante em um sistema depende do ponto de vista de cada desenvolvedor e que muitas vezes uma opinião sensata esta ligada à experiência, o valor da arquitectura depende das pessoas que estão ligadas ao desenvolvimento do sistema" [Jacobson et al. 1999]. De qualquer modo, o processo ajuda o arquitecto a concentrar-se nas metas correctas, como inteligibilidade, poder de recuperação para mudanças futuras e reutilização.

A relação existente entre casos de uso e a arquitectura é que os casos de uso estão ligados a funcionalidade de um sistema e a arquitectura, por sua vez, está ligada à forma deste. Além disso, funcionalidade e forma devem estar balanceadas para se alcançar um produto final de qualidade, ou seja, casos de uso e arquitectura devem estar ligados a tal ponto que o primeiro seja desenvolvido de acordo com a arquitectura, e esta por sua vez, forneça um ambiente para a realização de todos os requisitos dos casos de uso.

Assim, a arquitectura de um sistema deve ser projectada a ponto de permitir que o sistema evolua, não apenas durante o inicio de seu desenvolvimento, mas também através de gerações futuras. Para alcançar este objectivo, os arquitectos devem trabalhar com as funções chaves de um sistema, ou seja, os casos de uso chaves de um sistema. Estes são apenas cerca de 5% a 10% do total de casos de uso, porém, os mais significantes, aqueles que constituem o núcleo de funções do sistema.

4.1.3 O Processo Unificado é Incremental e Iterativo

Desenvolver um produto de software comercial é uma tarefa ampla que pode estender-se durante muitos meses, possivelmente até um ano ou mais. È prático dividir a tarefa em mini-projectos, cada mini - projecto é uma iteração que resulta em incremento. Iterações referem-se aos passos na disciplina e os incrementos ao crescimento do produto.

Para ser-se mais efectivo as iterações devem ser controladas, isto significa que devem ser seleccionadas e levadas a cabo de uma forma planificada, é por isso que são mini-projectos. Os

desenvolvedores baseiam a selecção do que é para ser implementado numa iteração sobre dois factores:

- A iteração lida com um grupo de casos de uso que em conjunto alarga a utilidade do produto tal como foi desenvolvido;
- A iteração lida com os riscos mais importantes. Iterações sucessivas construídas nos artefactos em desenvolvimento a partir do estado em que ficaram no fim da iteração anterior, é um mini-projecto.

Em toda a iteração os desenvolvedores identificam e especificam os casos de uso relevantes criam o desenho usando a arquitectura seleccionada como guia, implementa o desenho em componentes e certificam se os componentes satisfazem os casos de uso.

Se uma iteração alcança os seus objectivos, o desenvolvimento continua com a iteração seguinte. Quando uma iteração não alcança os objectivos o desenvolvedor deve revisitar as suas decisões anteriores e tentar uma nova abordagem. Para alcançar a melhor economia no desenvolvimento uma equipe de projecto deve tentar seleccionar somente as iterações requeridas para o alcance dos objectivos do projecto. Tentará sequenciar as iterações numa ordem lógica.

O processo de desenvolvimento exigirá maiores esforços e levará mais tempo, minimizar problemas não previstos é um dos objectivos na redução de risco.

De acordo com [Jacobson et al., 1999], há muitos benefícios no processo iterativo controlado:

- Reduz o risco de custo nos gastos para um único incremento. Se os desenvolvedores necessitarem de repetir a iteração a organização perde somente o esforço mal direccionado de uma iteração, não o valor de todo produto;
- Reduz o risco de não lançar-se o produto no mercado de acordo com o tempo planificado. Ao identificar-se os riscos no princípio do desenvolvimento o tempo gasto para a sua solução ocorre cedo quando as pessoas estão sob pouca pressão como ficam numa fase mais avançada. Na abordagem "tradicional" onde problemas difíceis são relevados pelo teste ao sistema, o tempo requerido, para a sua solução, excede o tempo remanescente e quase sempre provoca atrasos da entrega do produto;
- Acelera o tempo de todo esforço de desenvolvimento porque os desenvolvedores trabalham mais eficientemente pelos resultados num enfoque claro e de curta duração em vez de planos longos.

- Reconhece uma realidade muitas vezes ignorada, que as necessidades dos utilizadores e os requisitos correspondentes não podem ser completamente definidos antecipadamente. São tipicamente refinados em iterações sucessivas. Este modo de operar torna mais fácil a adaptação as mudanças necessárias.

Estes conceitos, dirigido por um casos de uso, centrado na arquitectura e desenvolvimento iterativo e incremental, são importantes.

A arquitectura providencia a estrutura que guia o trabalho nas iterações sempre que os casos de uso definem objectivos e conduz o trabalho de cada iteração.

Depois da introdução dos três conceitos chaves, a seguir tratar-se-á de todo o processo, o seu ciclo de vida, artefactos, disciplinas, fases e iterações.

4.2 Ciclo de vida da USDP

A USDP repete uma série de ciclos até que se complete a vida do sistema. Cada ciclo é concluído com o lançamento de um produto para os clientes que consiste em quatro fases (concepção, elaboração, construção e transição). Cada fase é subdividida em iterações em conformidade com a abordagem supracitada.

O produto final inclui também os requisitos, casos de uso, especificações não funcionais e casos de teste. Inclui a arquitectura e os modelos visuais, artefactos modelados pela UML.

Para levarem o ciclo seguinte eficientemente os desenvolvedores precisam de todas as representações do produto de software:

- O modelo do caso de uso com todos os casos de uso e as suas relações com os utilizadores;
- Um modelo de análise que tem dois propósitos:
  - Retirar os casos de uso com mais detalhes;
  - Fazer a alocação inicial do comportamento do sistema, para um conjunto de objectos que providenciam o comportamento.
- Um modelo de desenho que defina a estrutura estática do sistema como subsistema, classes e interfaces;
- O modelo de implementação que inclui componentes (representando o código fonte) e os mapas das classes dos componentes;

- Modelo de teste que descreve os casos de teste que verifica os casos de uso;
- Representação da arquitectura.

O sistema pode ter um modelo do domínio ou modelo de negócios que descreve o contexto de negócios do sistema, todos estes modelos estão relacionados e juntos representam o sistema como um todo.

Todo ciclo realiza-se durante algum tempo. Este tempo está dividido em quatro fases como está ilustrado na fig.4.1, através da sequência de modelos os intervenientes visualizam o que ocorre nessa fase. Dentro de cada fase, os gestores ou desenvolvedores podem repetir o trabalho, em iterações e incrementos seguintes.

Toda a fase termina quando se atinge os objectivos previamente estabelecidos. Define-se todo o objectivo pela disponibilidade de um grupo de artefactos.

Um objectivo serve para muitos propósitos e o mais crítico, é que os gestores tem que tomar algumas decisões cruciais antes que o trabalho prossiga para a fase seguinte. Os objectivos permitem aos gestores assim como os desenvolvedores monitorarem o progresso do trabalho a medida que vai passando por estas quatro fases. Com a conservação de tempo e o esforço despendido em cada fase, desenvolvem-se dados, estes são importantes para estimar o tempo e necessidades de pessoal para outros projectos, projectar as necessidades do pessoal pelo tempo do projecto e o controle da evolução contra estas projecções.

A fig.4.1 alista as disciplinas (requisitos, análise, projecto, implementação e teste) na coluna da esquerda. As curvas estimam, até que ponto as disciplinas são levadas a cabo em cada fase. Cada fase normalmente é subdividida em iterações ou mini-projectos, uma iteração típica passa através de todas as cinco disciplinas tal como está ilustrado na figura.

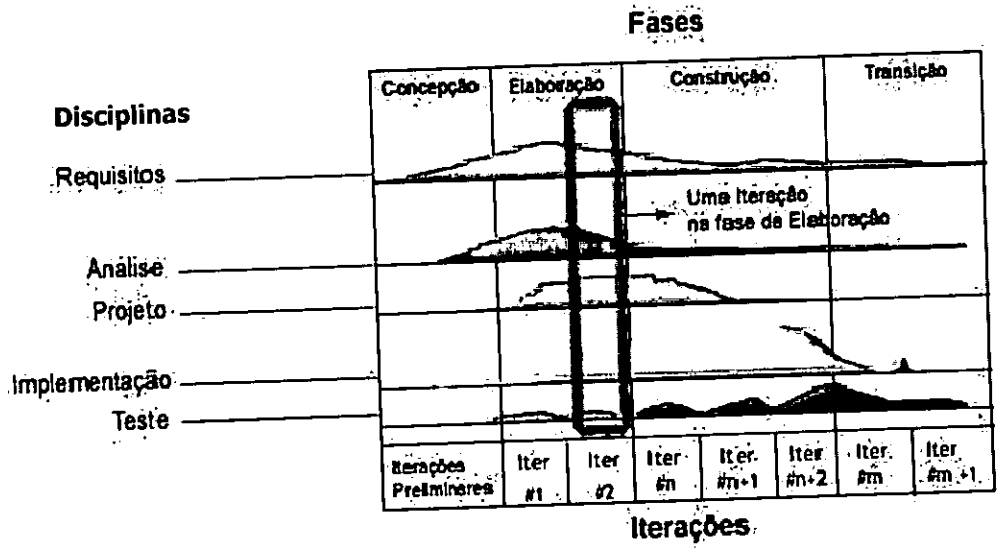


Fig. 4.1 As cinco disciplinas, requisitos, análises, desenho, implementação e teste acima apresentadas e as quatro fases: Concepção, Elaboração, construção e transição (fonte: Júnior, 2001)

4.2.1 Iterações

Antes da sua conclusão, o ciclo de vida a USDP passa por várias e sucessivas iterações. Cada uma destas iterações resulta numa versão de um produto executável que constitui um subconjunto do produto final em desenvolvimento e cresce de modo incremental de uma iteração para outra até se tornar o sistema final, conforme mostra a fig.4.1.

Cada iteração corresponde a uma das quatro fases do ciclo de vida, concepção, elaboração, construção e transição. Em cada uma destas fases, ocorrem varias iterações, sendo que cada iteração passa por todas as cinco disciplinas do processo unificado, requisitos, analise, projecto, implementação e teste. Porém, a importância de cada disciplina para uma iteração depende da fase em que esta iteração se encontra.

Durante a concepção, o foco está na captação de requisitos. Na elaboração, o foco passa a ser a análise e projecto do sistema. A implementação e a actividade central na construção e a transição é caracterizada pela entrega de um produto a comunidade de usuários.

4.2.2 Disciplinas

A USDP descreve as actividades a serem executadas, tais como escrever um caso de uso, dentro das disciplinas (originalmente chamado *Workflows*). Informalmente, a disciplina é um grupo de actividades enquadradas numa área específica tais como actividades que estão dentro da análise

de requisitos. Na USDP o artefacto é um termo geral para qualquer produto resultante de um trabalho: por exemplo programa, esquema da base de dados, documentos em texto, diagramas, modelos, etc.

Na USDP há várias disciplinas de modelação que são descritas em seguida:

4.2.2.1 Requisitos

A captura de Requisitos é o processo para descoberta do que é preciso ser construído. Esta captura geralmente acontece sob difíceis condições porque ela é construída para um usuário, e não para si mesmo, sendo os usuários uma fonte pouco confiável de informações, com tendência das suas necessidades mudarem durante o processo de desenvolvimento.

O propósito da disciplina de Requisitos é o desenvolvimento do sistema para a direcção correcta isto se consegue através da descrição dos requisitos do sistema de uma boa maneira e suficiente de modo que o entendimento possa ser alcançado entre o cliente e os desenvolvedores. Deve-se sempre utilizar a linguagem que o usuário entende para descrever os requisitos obtidos e os seus resultados, auxiliando a gerência do projecto a planear iterações e lançamentos para os usuários.

Apesar das diferenças existentes entre os desenvolvimentos de diferentes sistemas, alguns passos básicos podem ser dados:

- Listar os candidatos a requisitos
- Entender o contexto do sistema
- Capturar requisitos funcionais
- Capturar requisitos não funcionais

Durante o desenvolvimento, todos podem contribuir com boas ideias que poderão se tornar requisitos do sistema.

Deve-se manter uma lista destas boas ideias, esta lista cresce com a adição de novas ideias e decresce com a transformação destas ideias em requisitos

Sugestão de organização da lista:

- Estado (proposto, aprovado, incorporado, validado etc.)
- Custo estimado de implementação

- Prioridade
- Nível de risco associado

Existem duas maneiras para se expressar o contexto de um sistema, que as pessoas chave devem entender perfeitamente no desenvolvimento desses sistemas:

modelo de domínio: descreve os conceitos importantes do contexto através de objectos do domínio;

modelo de negócios: é um super conjunto de modelos de domínio, descrevendo principalmente os processos (existentes ou desejáveis) do contexto.

Os requisitos funcionais, estes são representados através de casos de uso, enquanto que não funcionais, especificam propriedades do sistema, como por exemplo, restrições de ambiente e de implementação, desempenho, dependências relacionadas a plataformas, manutenibilidade, extensibilidade e confiabilidade sendo estes também representados através de casos de uso. A fig.4.2 mostra a importância da disciplina de requisitos em cada fase do ciclo de vida do sistema.

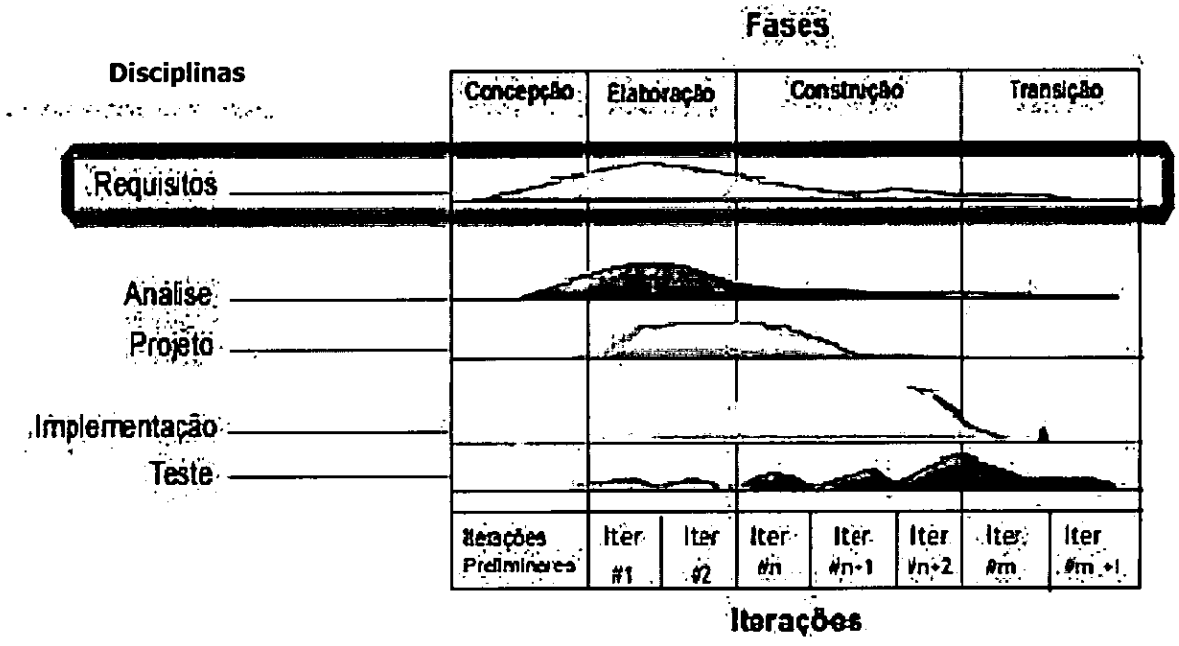


Fig.4.2 O papel dos requisitos no ciclo de vida do software (fonte: Júnior, 2001)

Durante a fase da concepção, o analista identifica a maioria dos casos de uso, isto permite delimitar o sistema e a fronteira do projecto e também detalhar os casos de uso mais críticos (menos de 10%).

Na fase de elaboração, o analista captura a maioria dos requisitos restantes e isto permite que os desenvolvedores estimem o tamanho do esforço de desenvolvimento. No final, o objectivo é ter capturado cerca de 80% dos requisitos e descrito a maioria dos casos de uso.

Durante a fase de construção, os requisitos restantes são capturados e implementados.

Durante a fase de transição, é quase inexistente a captura de requisitos a menos que haja mudanças nos requisitos iniciais.

A descrição da disciplina de requisitos, implica a especificação dos artefactos. Um artefacto pode ser um modelo, um elemento de um modelo ou documento.

Um dos artefactos usados para a captura de requisitos é o modelo de caso de uso que contém actores, casos de uso, seus relacionamentos e permite ao desenvolvedor e ao cliente um entendimento mútuo dos requisitos.

O artefacto glossário de casos de uso é utilizado para definir termos comuns e importantes utilizados na descrição do sistema. É útil para se conseguir consenso entre diferentes participantes no tocante a definição de diferentes noções e conceitos, geralmente deriva dos modelos de domínio e de negócio.

4.2.2.2 Análise

O produto da disciplina de análise é o modelo de análise. Este modelo tem a função de refinar os requisitos especificados na disciplina anterior (disciplina de requisitos) através da construção de diagramas de classes conceptuais, permitindo, desta forma, argumentações a respeito do funcionamento interno do sistema. Além disso, o modelo de análise fornece mais poder expressivo e formalismo, como diagramas de interacções e diagramas de gráfico de estados que representam a dinâmica do sistema.

De facto, a disciplina de análise tem maior importância durante a fase de elaboração, como pode ser visto na fig.4.3. Isso contribui para a definição de uma arquitectura estável e facilita o entendimento detalhado dos requisitos.

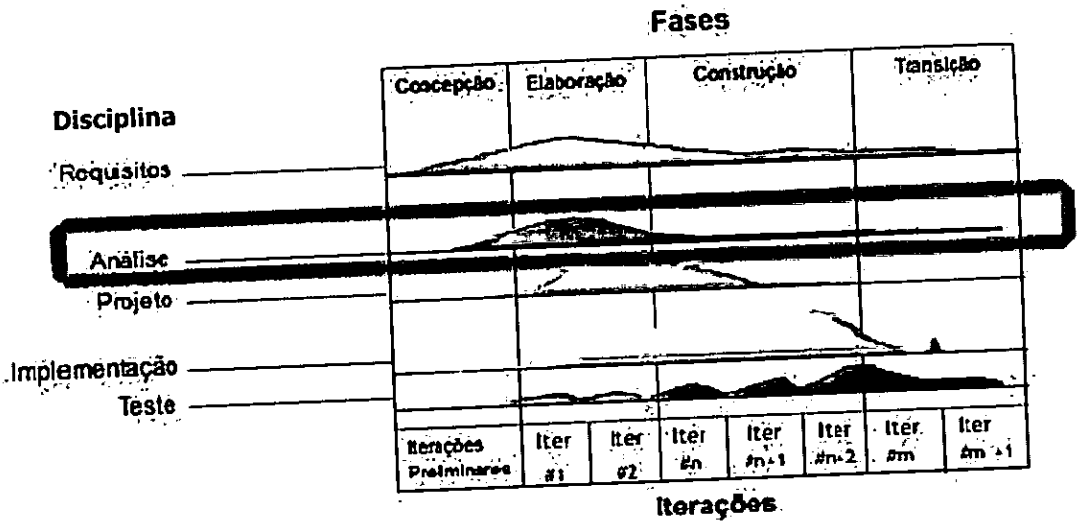


Fig.4.3 O papel da análise no ciclo de vida do software (fonte: Júnior, 2001)

O modelo de análise cresce incrementalmente a medida em que casos de uso são analisados. Para cada iteração, é seleccionado um conjunto de casos de uso que são realizados no modelo de análise. O sistema é construído como uma estrutura de classes de análise e relacionamentos entre estas classes. Assim, na próxima iteração outro conjunto de casos de uso é realizado e adicionado ao produto da iteração prévia.

Uma maneira prática para realizar esta disciplina é primeiro identificar e detalhar os casos de uso para uma iteração, e depois, através da análise da descrição de cada caso de uso, sugerir quais classes e relacionamentos são necessários para realiza-lo. Isto é feito para todos os casos de uso numa iteração.

Dependendo da etapa em que o ciclo de vida do processo se encontra e do tipo de iteração que está sendo trabalhada, a arquitectura do sistema poderá auxiliar a identificação de novas classes e a reutilização de outras já existentes.

Cada classe de análise possui um ou vários papéis numa descrição de um caso de uso. Cada papel especifica as responsabilidades, atributos, e tudo mais que for necessário para detalhar um caso de uso.

Ao estruturar os requisitos do sistema, o modelo de análise fornece uma estrutura com foco na manutenção dos mesmos. Porém, esta estrutura não é útil apenas à manutenção de requisitos, mas também serve de entrada para as disciplinas de projecto e implementação. Desta forma, o modelo de análise pode ser visto com o primeiro passo para o desenvolvimento do modelo de projecto, muito embora, seja considerado um modelo em particular. O facto de se preservar a

estrutura do modelo de análise no projecto, permite que o processo desenvolva um sistema de fácil manutenção, que possua um grande poder de recuperação mediante a mudança de requisitos e que inclua elementos que possam ser reutilizados em sistemas relacionados em desenvolvimento.

De qualquer modo, é importante notar que o modelo de análise não trata problemas que são solucionados em modelos de projecto ou implementação do sistema. A razão pela qual a preservação da estrutura de um modelo de análise é efectuada na prática e evidenciada pelo facto do projecto considerar a plataforma de implementação do sistema como, linguagens de programação, sistemas operacionais pré-fabricados, etc. Portanto, visando a diminuição de custos e o melhor aproveitamento de prazos, a arquitectura deve ser lançada modificando-se o modelo de análise durante a transição para o modelo de projecto.

4.2.2.3 Projecto

Nesta disciplina o sistema é modelado e sua forma é definida de maneira a suprir as necessidades especificadas pelos requisitos. Esta disciplina define um método de projecto que é construído com base no modelo de análise definido na disciplina anterior.

Porém, ao contrário da disciplina de análise, cujo produto é um modelo que descreve características comportamentais e estruturais do sistema num nível conceptual, a disciplina de projecto desenvolve o modelo de projecto que descreve o sistema num nível físico, tendo como principal função obter uma compreensão detalhada dos requisitos do sistema, levando em consideração factores como linguagens de programação, sistemas operacionais, tecnologias de banco de dados, interface com o usuário, etc.

A disciplina de projecto possui seu enfoque entre o fim da fase de elaboração e o início da fase de construção, como pode ser visto na fig.4.4. Isso contribui para a definição de uma arquitectura estável e para a criação do modelo de projecto que servirá de base para a disciplina de

implementação. Desta forma, sendo o modelo de projecto bem ligado a disciplina de implementação, é normal que o modelo de projecto seja mantido através do ciclo de vida completo do sistema.

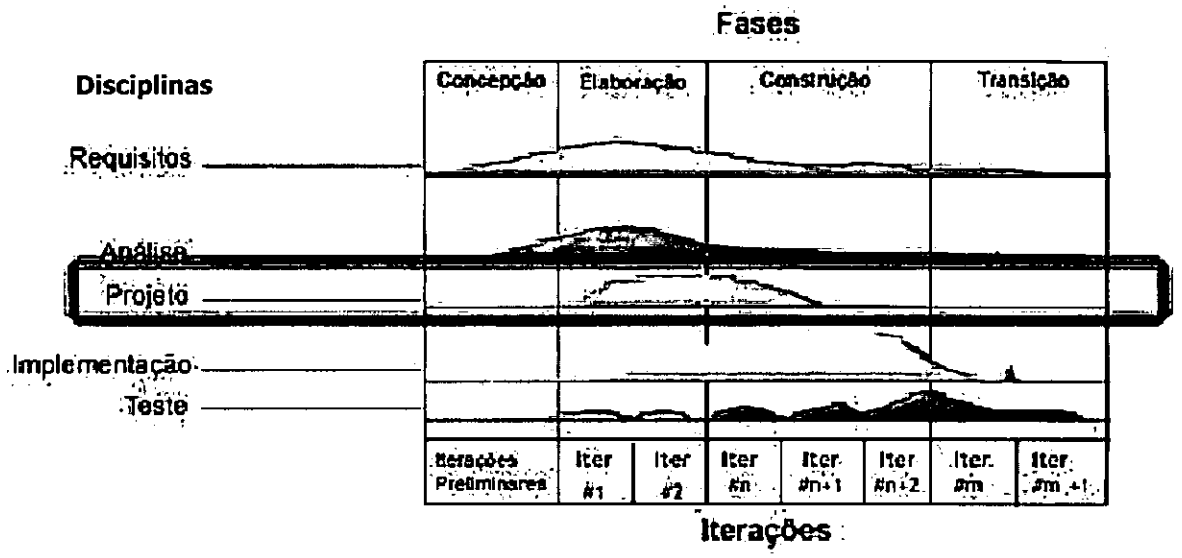


Fig.4.4 O papel do projecto no ciclo de vida do software (fonte: Júnior, 2001)

Através do modelo de projecto, casos de uso são realizados por artefactos de projecto representados por ferramentas de modelação também utilizadas na disciplina de análise como digramas de classes, diagrama de interacção e diagramas de gráfico de estados, agora com intuito de capturar os requisitos de implementação; ou seja, estes mesmos diagramas são construídos num nível mais físico que conceptual. Além disso, a disciplina de projecto também utiliza outras ferramentas, não comuns a disciplina de análise, como diagrama de objectos.

Assim, o modelo de projecto descreve as realizações físicas de casos de uso considerando como requisitos, e outros detalhes relacionados ao ambiente de implementação, causam impacto ao sistema sob consideração. Desta forma, o modelo de projecto funciona como uma abstracção da implementação do sistema. Em suma, enquanto que a disciplina de análise se interessa pelo **o que** o sistema deve fazer, a disciplina de projecto diz respeito a **como** os requisitos serão implementados e, portanto, pressupõe uma infra-estrutura de implementação e é fortemente influenciado por ela.

O projecto pode dividir o sistema em subsistemas visando a organização dos artefactos do modelo de projecto e partes mais gerenciáveis. Um subsistema pode ser composto de classes de projecto, realizações de casos de uso, interfaces, e recursivamente, outros subsistemas. Além disso, um subsistema pode fornecer interfaces que representam a funcionalidade que elas exportam em termos de operações.

Um subsistema deve ser coesivo, ou seja, seu conteúdo deve estar extremamente relacionado. Por outro lado cada subsistema deve depender o mínimo possível de outro subsistema. A grande vantagem de se trabalhar com subsistemas na disciplina de projecto é que estes subsistemas poderão ser projectados separadamente e concorrentemente por diferentes equipes de desenvolvimento em diferentes níveis do projecto.

4.2.2.4 Implementação

A disciplina de implementação é baseado no produto da disciplina de projecto, o modelo de projecto; e implementa o sistema em termos de componentes, ou seja, código fonte, arquivos executáveis, etc.

A maior parte da arquitectura do sistema é definida durante o projecto, como pode ser visto na fig.4.5. Portanto, a disciplina de implementação produz um modelo de implementação que se limita a:

- Planear as integrações do sistema em cada iteração. Neste caso, o resultado é um sistema que é implementado como uma sucessão de etapas pequenas e gerenciáveis.
- Implementar os subsistemas encontrados durante o projecto.
- Testar as implementações e integrá-las, compilando-as em um ou mais arquivos executáveis, antes de enviá-los a disciplina de teste.

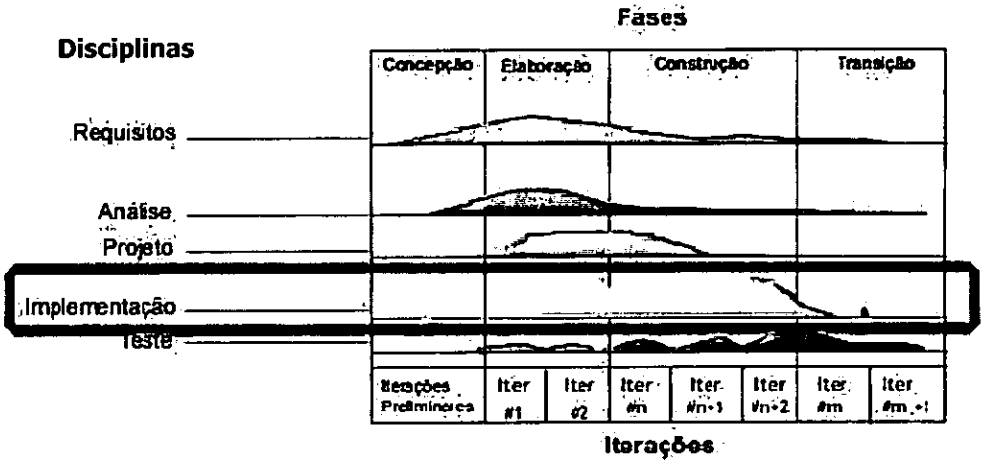


Fig.4.5 O papel da implementação no ciclo de vida do software (fonte: Júnior, 2001)

A disciplina de implementação tem maior importância na fase de construção e apesar de ter suas características próprias, a maior parte de suas actividades é realizada de forma quase mecânica, pelo facto das decisões mais difíceis terem sido tomadas durante a disciplina de projecto. Desta forma, o código gerado durante a implementação, deve ser uma simples tradução das decisões de projecto numa linguagem específica.

4.2.2.5 Teste

A disciplina de teste é desenvolvida com base no produto da disciplina de implementação, ou seja, os componentes executáveis são testados exaustivamente na disciplina de teste para então serem disponibilizados aos usuários finais. Desta forma o principal propósito da disciplina de teste é realizar vários testes e sistematicamente analisar os resultados de cada teste.

Componentes que possuírem defeitos retornarão a disciplinas anteriores como as disciplinas de projecto e implementação, onde os problemas encontrados poderão ser corrigidos.

O teste de um sistema, propriamente dito, é primeiramente empregue durante a fase de elaboração, quando a arquitectura do sistema é definida, e durante a fase de construção quando o sistema é implementado. Porém, um planeamento inicial de testes pode ser feito durante a fase de concepção. Na fase de transição, a disciplina de testes se limita ao conserto de defeitos encontrados durante a utilização inicial do sistema. O papel da disciplina de teste dentro do ciclo de vida do processo unificado pode ser visualizado na figura 4.6

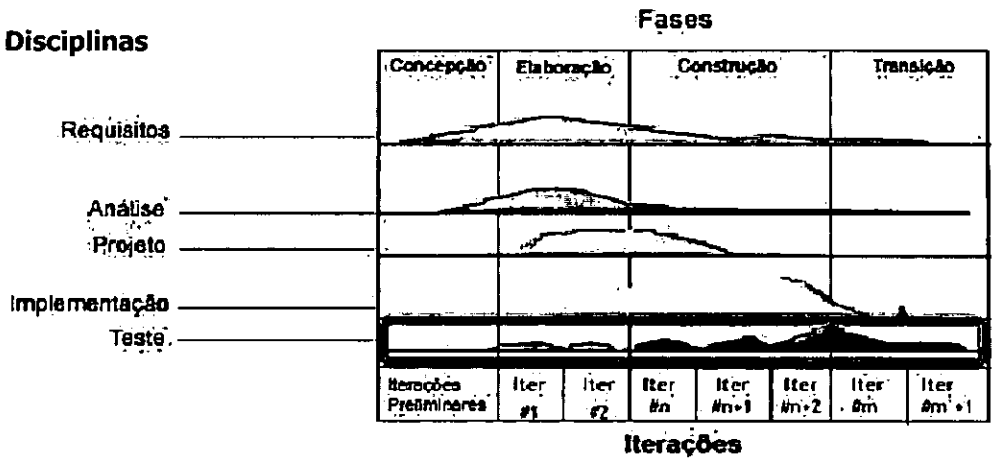


Fig.4.6 O papel do teste no ciclo de vida do software (fonte: Junior, 2001)

O produto da disciplina de teste é o modelo de teste, esse modelo primeiramente descreve como componentes executáveis, provenientes da disciplina de implementação, são testados. O modelo de testes também pode descrever como aspectos específicos do sistema são testados, como por exemplo, se a interface com o usuário é útil e consistente ou se o manual do usuário cumpre seu objectivo.

Em suma, o papel da disciplina de teste é verificar se os resultados da disciplina de implementação cumprem os requisitos estipulados por clientes e usuários, para que possa ser decidido se o sistema necessita de revisões ou se o processo de desenvolvimento pode continuar.

4.2.3 As fases de desenvolvimento de um sistema em USDP

Um ciclo de vida está dividido em fases, cada uma podendo ser subdividida em iterações e consequentes incrementos. O final de uma fase é marcado por um ponto de verificação, isto é, pela disponibilidade de um conjunto de artefactos que possibilitem a avaliação do projecto, tais como modelos e outros documentos.

Os pontos de verificação servem a diversos propósitos:

Gerentes devem tomar certas decisões cruciais antes do trabalho continuar.

Permitem a monitoração do progresso dos trabalhos.

Observando o tempo e o esforço despendidos em cada fase, é possível reunir dados úteis para estimar os requisitos de tempo de outros projectos.

Conforme mostra a fig.4.1, cada fase é subdividida em iterações e cada iteração executa as cinco disciplinas listadas na coluna à esquerda (requisitos, análise, projecto, implementação e testes). As curvas não devem ser interpretadas literalmente, mas representam uma aproximação do esforço despendido com cada actividade em cada fase.

São quatro as fases que compõem o ciclo de vida da USDP:

4.2.3.1 Concepção

O objectivo principal da fase de concepção é delimitar a fronteira do projecto, definindo como o sistema será utilizado por cada usuário, através da criação dos casos de uso mais relevantes. A partir desta fronteira, poderá ser definido o que o desenvolvimento do projecto deverá cobrir, no que diz respeito a custos, prazos, retorno de investimento, etc.

Além disso, o esforço empenhado na fase de concepção poderá evitar o fracasso do projecto através da identificação prévia de riscos. A causa do fracasso de vários projectos é o facto de riscos críticos serem encontrados tarde demais, geralmente durante a integração do sistema ou durante a etapa de testes do mesmo, quando não se há mais tempo nem orçamento para revisões.

A maior parte do trabalho da fase de concepção está concentrado na disciplina de requisitos, porém cada disciplina possui seu papel dentro desta fase.

No final da fase de concepção, os objectivos do ciclo de vida do projecto devem ser analisados para se decidir se o desenvolvimento deve prosseguir em plena escala.

4.2.3.1.1 Requisitos

A disciplina de requisitos da fase de concepção é caracterizado pela identificação dos requisitos mais relevantes para a definição da fronteira do sistema e para a compreensão preliminar da arquitectura do mesmo.

Ao capturar os requisitos nesta fase, o analista de sistemas identifica os casos de uso e actores que definem a fronteira do sistema, priorizando e detalhando aqueles que serão, eventualmente, mais importantes para a descrição inicial da arquitectura do mesmo. O resultado deste trabalho é o primeiro modelo de casos de uso.

4.2.3.1.2 Análise

Na disciplina de análise da fase de concepção, o modelo de análise inicial é criado a partir do detalhamento, em termos de classes e relacionamentos, de boa parte dos casos de uso definidos durante a disciplina de requisitos anterior.

Este modelo será importante para se obter uma compreensão clara dos casos de uso e para o desenvolvimento da arquitectura do sistema.

Porém, durante a fase de concepção muito pouco do modelo completo de análise é construído, assim o modelo de análise nesta fase é apenas o primeiro passo para a obtenção da visão arquitectural do sistema.

4.2.3.1.3 Projecto

A meta da disciplina de projecto nesta fase é fazer um esboço do modelo de projecto que contribuirá para a descrição inicial da arquitectura do sistema.

Isto é feito através da identificação de classes de projecto e seus relacionamentos a partir do estudo dos casos de uso detalhados na disciplina anterior, a disciplina de análise.

4.2.3.1.4 Implementação e Teste

Para reduzir ao mínimo custos e tempo despendidos na fase de concepção as tarefas relacionadas as disciplinas de implementação e teste podem não ser executadas.

Esta decisão não acarreta nenhum prejuízo ao processo de desenvolvimento do sistema, tendo em vista que as principais metas da fase de concepção, que são a definição da fronteira do sistema e

a descrição inicial de sua arquitectura, são alcançadas entre as disciplinas de requisitos e projecto realizados anteriormente.

4.2.3.2 Elaboração

Durante esta fase muitos dos produtos de casos de uso são especificados em detalhes e a arquitectura do sistema é desenhado, há maior relacionamento entre a arquitectura e o sistema.

A Elaboração, consiste em duas ou quatro iterações, é recomendado que cada iteração dure entre duas a seis semanas a menos que a equipe julgue que seja pesado. Cada iteração está dimensionada no tempo, isto significa que a sua data limite está fixada.

Uma simples maneira de colocar a questão é que a arquitectura é análoga a um esqueleto coberto de pele mas com muito pouco músculo (o software) entre os ossos e a pele, precisa-se o músculo suficiente para permitir o esqueleto a fazer movimentos básicos. O sistema é todo corpo com esqueleto, a pele e os músculos.

Por isso a arquitectura é expressa com pontos de vista de todos os modelos do sistema que em conjunto representam o sistema num todo. Isto implica que existem pontos de vista arquitecturais para o modelo caso de uso, análise do modelo, modelo de desenho, implementação do modelo e o desenvolvimento do modelo. Sob ponto de vista do modelo de implementação inclui componentes que provam que a arquitectura é executável.

Durante esta fase de desenvolvimento os casos de uso mais críticos identificados durante a fase da elaboração são realizados. O resultado desta fase é uma linha base da arquitectura.

No fim da fase da elaboração o gestor do projecto está em condições de planificar as actividades fazer uma estimativas dos recursos necessários para completar o projecto. Aqui a questão principal será os casos de uso, arquitectura e planos suficientemente estáveis e estarão os riscos sobre controlo para que se possa assumir todo trabalho de desenvolvimento num contracto.

4.2.3.2.1 Requisitos

A primeira tarefa da disciplina de requisitos da fase de elaboração é identificar os casos de uso adicionais, ou seja, aqueles que não foram identificados na fase de concepção. Estes casos de uso juntos formam cerca de 80% do total de casos de uso referentes ao sistema, porém, só uma parte deste conjunto deve ser detalhada nesta fase, mais especificamente, a parte que corresponde aos casos de uso que contribuem para a compreensão plena dos requisitos e para a criação da base da arquitectura do sistema.

Assim como na fase anterior, os casos de uso capturados nesta disciplina deverão ser detalhados e priorizados, porém, a quantidade dos casos de uso identificados e detalhados neste primeiro momento depende das circunstâncias nas quais o sistema estará sendo desenvolvido. Por exemplo, se for estipulado um preço fixo ao cliente, consequentemente, a maior parte dos casos de uso deverá ser detalhada, provavelmente mais que 80 %. Se o empreendimento for financiado pelos próprios desenvolvedores, então, a percentagem dos casos de uso detalhados poderá ser menor. Neste último caso, o risco a ser considerado no desenvolvimento é maior, em compensação, o tempo gasto e o esforço empregado na fase de elaboração são menores. Esta estratégia é adoptada quando se espera um retorno imediato do empreendimento.

A partir da identificação e detalhamento dos casos de uso nesta primeira etapa, o analista de sistemas poderá estruturar o modelo de casos de uso. Isto é feito a partir da identificação de similaridades e eliminação de eventuais redundâncias existentes no modelo de casos de uso, de forma a deixá-lo mais compreensivo e de fácil manutenção.

4.2.3.2.2 Análise

A disciplina de análise da fase de elaboração se concentra na análise dos casos de uso significantes à arquitectura do sistema. Estes casos de uso, que representam um pouco menos de 10% do total de casos de uso definidos, são analisados visando o complemento do trabalho de análise arquitectural feito na fase de concepção. Assim, tendo como ponto de partida a arquitectura definida superficialmente durante a fase de concepção, esta disciplina de análise poderá definir uma base sólida para a arquitectura do sistema de forma a se conseguir uma arquitectura executável.

Através deste trabalho de análise, os casos de uso significantes à arquitectura do sistema são analisados em termos de classes e a estas classes são aloucadas responsabilidades específicas. Os relacionamentos existentes entre classes e os atributos de cada classe também são definidos durante esta etapa do processo. Desta forma, as classes que são relevantes à arquitectura são seleccionadas e reunidas através da análise de suas responsabilidades, formando pacotes que contribuirão para a definição de uma base para a visão arquitectural do modelo de análise.

Outros casos de uso que não estão directamente ligados à arquitectura do sistema e que não são relevantes para a compreensão preliminar dos requisitos são analisados posteriormente na fase de construção.

4.2.3.2.3 Projecto

Nesta fase menos de 10% do total de casos de uso são projectados e implementados. Esta pequena percentagem é apenas uma fracção dos casos de uso identificados nesta fase. O projecto da fase de elaboração é realizado a nível arquitectural, ou seja, o projecto envolve classes, subsistemas e casos de uso significantes à arquitectura.

Tanto pacotes na análise quanto subsistemas no projecto são importantes para a definição de uma visão arquitectural. Apesar de várias classes poderem não ser significantes à arquitectura, pacotes e subsistemas geralmente são.

O arquitecto é o responsável pelo projecto dos aspectos significantes à arquitectura, ele continua o trabalho iniciado na fase de concepção e projecta a arquitectura em camadas.

As camadas mais baixas da arquitectura são caracterizadas por representarem mecanismos de projecto, ou seja, mecanismos do sistema operacional no qual o sistema proposto irá operar, linguagens de programação, sistemas de banco de dados, etc.

As camadas mais altas estão próximas às camadas de aplicação da arquitectura. Desta forma, baseado nos pacotes definidos no modelo de análise, o arquitecto identifica os subsistemas correspondentes que serão incluídos no modelo de projecto. Geralmente, cada pacote do modelo de análise se torna um subsistema no modelo de projecto, porém nem sempre isto acontece, algumas vezes um pacote pode se referir a vários subsistemas do modelo de projecto, e em outros casos, um subsistema pode ser criado com base em vários pacotes do modelo de análise.

Assim como na fase anterior, classes de projecto são identificadas através da "tradução" de classes de análise, porém, durante a fase de elaboração, só são seleccionadas as classes de projecto interessantes à arquitectura, visando o enriquecimento da descrição arquitectural do projecto.

Durante a disciplina de projecto, a realização de cada caso de uso é feita num nível mais físico do que conceptual, ou seja, classes de projecto, subsistemas, interfaces, a linguagem de programação e o banco de dados são levados em consideração neste momento. O resultado desta disciplina é um conjunto de realizações de casos de uso de projecto, onde são criados artefactos para cada caso de uso significante à arquitectura.

4.2.3.2.4 Implementação

A disciplina de implementação na fase de elaboração implementa e testa os elementos significantes à arquitectura com base no modelo de projecto resultante da disciplina anterior.

Desta forma, são identificados os componentes necessários para a implementação dos subsistemas predefinidos. Este trabalho prossegue de maneira incremental até que se obtenha a primeira versão executável do sistema proposto.

4.2.3.2.4 Teste

Durante esta disciplina de trabalho são testados os subsistemas executáveis produzidos no fluxo de implementação.

O teste se inicia pela camada mais baixa da arquitectura, ou seja, o banco de dados. No caso das camadas mais altas, o mais importante é avaliar como estas utilizam as mais baixas.

Os testes realizados não se limitam a avaliar apenas a funcionalidade do sistema, mas também sua performance.

4.2.3.3 Construção

Durante a fase de construção o produto é constituído o software é completado e junta-se a arquitectura. Nesta fase a linha base da arquitectura cresce para tornar-se num sistema completamente acabado. A visão desenvolve-se para um produto pronto a ser transferido para a comunidade utilizadora.

Durante esta fase de desenvolvimento o grosso dos recursos requeridos é gasto. A arquitectura do sistema é estável contudo porque os desenvolvedores podem descobrir melhores maneiras de estruturarem o sistema. Podem seguir pequenas mudanças arquitectónicas ao arquitecto. No fim desta fase o produto contém todos os casos de uso que os gestores e clientes acordaram desenvolver para este lançamento. Esta poderá não estar completamente livre de defeitos, mas estes defeitos serão descobertos e corrigidos durante a fase de transição.

A pergunta importante será que o produto satisfará as necessidades dos utilizadores o suficiente para que alguns clientes possam receber as primeiras entregas?

4.2.3.3.1 Requisitos

Durante a disciplina de requisitos da fase de elaboração, foram já identificados todos os casos de uso e actores, porém, apenas uma parcela destes casos de uso, aqueles necessários para a descrição da base da arquitectura, foram detalhados. Agora na fase de construção, deverão ser detalhados os casos de uso remanescentes.

Outra tarefa da disciplina de requisitos da fase de construção consiste na construção de protótipos de interfaces com usuários, isto é necessário quando a interface em questão for muito complexa.

De qualquer forma o trabalho da disciplina de requisitos exercido nesta fase é pequeno em comparação ao trabalho de implementação, que é onde esta fase foca suas actividades.

4.2.3.3.2 Análise

Na prática, o modelo de análise dificilmente será preservado até esta etapa do processo, transferindo a disciplina de projecto a responsabilidade da análise dos casos de uso remanescentes, de forma a proporcionar mais objectividade ao processo de desenvolvimento. Porém, esta decisão fica a critério do analista de sistemas.

Se o modelo de análise for preservado até esta fase do processo, o mesmo será completado através da análise de classes e casos de uso com base no produto da disciplina anterior, ou seja, aquelas classes e casos de uso remanescentes, que não foram analisados na fase de elaboração.

O resultado da disciplina de análise nesta fase é um modelo de análise mais completo que aquele desenvolvido durante a fase de elaboração, fazendo com que a visão arquitectural definida anteriormente seja apenas um subconjunto do modelo de análise actual.

4.2.3.3.3 Projecto

Esta disciplina é responsável pelo projecto de aproximadamente 90% dos casos de uso, aqueles que não foram utilizados para desenvolver a base da arquitectura durante a fase anterior.

Porém, o projecto de subsistemas só é necessário se for verificado que os subsistemas a serem adicionados são similares ou alternativos aqueles já existentes.

4.2.3.3.4 Implementação

Esta é a principal disciplina da fase de construção, onde o maior esforço da fase é empregue.

O trabalho de implementação é feito com base no modelo de projecto, revisto e actualizado na disciplina anterior, onde a arquitectura do sistema deve estar estabelecida, porém algumas actualizações podem ser necessárias, o que será de responsabilidade do arquitecto. A partir da arquitectura preestabelecida são implementados e testados os subsistemas.

O resultado deste trabalho, depois de algumas iterações, junto com a integração dos subsistemas e teste, é a versão operacional inicial, representado 100% do total de casos de uso.

4.2.3.3.5 Teste

O objectivo desta disciplina é a realização de testes na primeira versão operacional do sistema, proveniente da disciplina de implementação anterior.

Os testes realizados devem seguir procedimentos preestabelecidos com o intuito de alcançar metas estipuladas num plano de testes.

Se um teste não alcançar seus objectivos os procedimentos devem ser modificados até que os testes sejam realizados de forma satisfatória.

4.2.3.4 Transição

A fase de transição cobre o período durante o qual o produto passa para o lançamento. Durante esta fase um pequeno número de utilizadores experimenta o produto e reporta os defeitos e deficiências.

Os desenvolvedores então corrigem os problemas reportados e incorporam algumas das sugestões de melhoramento para o lançamento geral para a comunidade de utilizadores.

A fase de transição envolve actividades tais como a manufacturação, treinamento dos clientes, providencia de uma linha de assistência e a correcção de defeitos descobertos depois da entrega.

A equipe de manutenção divide estes defeitos em duas categorias:

- Aqueles com efeitos suficientes na operacionalidade que justifique um lançamento imediato;
- Aqueles que podem ser corrigidos no seguinte lançamento regular.

4.2.3.4.1 Actividades da fase de transição

São poucas as actividades exercidas pelas cinco disciplinas nesta fase, pelo facto da maior parte do trabalho já ter sido realizado durante a fase de construção. Assim, o foco das disciplinas de trabalho está na correcção de defeitos visando à eliminação de falhas que possam ocorrer na utilização inicial do produto, e na realização de testes para assegurar que estas correcções não provocaram novos defeitos.

Porém, o maior esforço da fase de transição está na realização de actividades que não estão directamente relacionadas as cinco disciplinas do processo unificado.

Em geral, as actividades da fase de transição são:

- Preparar a versão beta do produto;
- Instalar a versão beta para que seja testada pelos usuários;
- Gerir o resultado dos testes,
- Adaptar o produto corrigido às circunstâncias definidas pelo usuário;
- Completar os artefactos do projecto;
- Determinar quando o projecto chegará a sua conclusão.

Esta sequência de actividades varia de acordo com a natureza de cada projecto, ou seja, se o produto está sendo desenvolvido para o mercado ou para um cliente específico. No primeiro caso, haverá muitos usuários em potencial, onde cada um utilizará o produto de forma específica, sem cumprir uma rotina de testes preestabelecida.

No segundo caso, o cliente provavelmente escolherá um local para a instalação inicial do produto, e seguirá uma rotina sistemática para a realização dos testes.

O esquema de actividades varia também dependendo do facto do sistema ter sido desenvolvido visando à substituição de um já existente ou não. No caso de substituição, provavelmente a tarefa de migração ou conversão de dados, do sistema substituído para o novo sistema, deverá ser realizada.

O principal factor que determinará a conclusão da fase de transição, e por conseguinte, do projecto, é a "satisfação" do cliente, o que também deve ser avaliada sob algumas considerações.

No caso do produto ser lançado no mercado, o gerente do projecto considera que a maior parte de usuários estará satisfeita quando o projecto disponibilizar uma versão do produto que apresente soluções para os problemas encontrados durante os testes da versão beta. Na maioria dos casos, o produto continua evoluindo. Sendo assim, a fase de transição terminará quando o projecto passar a responsabilidade de manutenção contínua para a equipe de suporte.

No caso do produto ser desenvolvido para um cliente em particular, o gerente de projecto considera que o cliente estará satisfeito quando o sistema concluir seus testes de forma satisfatória. Isto depende da interpretação dos requisitos estipulados no contrato original e nas eventuais mudanças feitas nos requisitos durante fases posteriores. O cliente pode contratar o serviço de suporte do vendedor do sistema, assumir a responsabilidade pelo suporte ou delega-la a terceiros.

Os detalhes que evidenciam a satisfação do cliente podem variar, porém, uma vez que o projecto alcança seus objectivos de forma satisfatória, a fase de transição estará concluída. Caso contrário, deverá se iniciar um novo ciclo de desenvolvimento.

5. Aplicação de USDP no Modelo Conceptual de SPCE

O caso de estudo e as disciplinas descritas neste capítulo, contribuíram para a criação do modelo conceptual através de algumas das fases do processo unificado.

Dada a necessidade de armazenamento dos dados do eleitor mantendo de forma consistente, de modo a facilitar a gestão da informação referente a esse mesmo eleitor, a nível do STAE e outros interessados, achou-se pertinente idealizar o Sistema de Processamento de Cadernos Eleitorais (SPCE). Para tal houve necessidade de se fazer um estudo no que diz respeito as actividades executadas no Departamento de Informática de modo a se alcançar os objectivos da modelação do SPCE.

Estas actividades foram estudadas durante a fase da concepção na disciplina de requisitos e representadas através do diagrama de caso de uso, delimitando a fronteira do sistema, criando deste modo o diagrama de caso de uso. A partir deste diagrama, o diagrama de caso de uso incrementa através do refinamento dos casos de uso, até a criação do primeiro modelo de análise. Esta fase é apresentada numa única iteração.

A disciplina de análise na fase da elaboração foi caracterizado pelo estudo detalhado do modelo de caso de uso, como produto da disciplina de requisitos. A disciplina de análise permitiu que fosse criado o modelo de análise, usando o diagrama de classes conceptuais.

Este modelo foi elaborado em duas iterações:

- Na primeira iteração, seleccionou-se um conjunto de casos de uso, que não tinham sido identificados durante a fase da concepção, incrementando o modelo de caso de uso, o qual serviu de base para o complemento dos modelos de análise seguintes;

- Na segunda iteração definiram-se os atributos das classes do modelo de análise.

As fases de construção e transição do processo unificado não foram apresentadas neste trabalho, atendendo que neste caso apresentou-se o modelo conceptual. Desta forma, o caso de estudo apresentado neste capítulo, considera apenas o ciclo de desenvolvimento, composto pelas disciplinas de requisitos e análise, tendo em conta que os objectivos do trabalho é a criação do modelo conceptual, pois o mesmo ilustra os conceitos importantes do domínio do problema, suas associações e atributos, e não representa entidades de software.

Na fase da elaboração para além da disciplina de análise, existe a disciplina de projecto que engloba os diagramas de interacção e de estados, que não foram abordados neste trabalho atendendo o objectivo anteriormente traçado que se limita só na criação do modelo conceptual, e

não na implementação do modelo que seria a fase ideal para a construção dos diagramas da disciplina de projecto e subsequentes.

Nesta ordem de ideias, a criação do modelo conceptual no SPCE é importante sendo este criado, poderá ser implementado em qualquer sistema integrado de gestão que a instituição optar.

5.1 Caso de Estudo: SPCE

O Sistema de Processamento de Cadernos Eleitorais (SPCE) trata de toda informação dos Eleitores do País. Esta secção descreve os principais fluxos de informação e os requisitos que serão modelados aplicando a metodologia USDP.

Um caderno eleitoral é um conjunto de 1000 ou 500 eleitores inscritos num determinado posto de recenseamento distribuídos pelo território nacional para recolher a informação dos potenciais eleitores. No final de cada recenseamento, por mecanismos adequados, cada posto de recenseamento envia as "**fichas por informatizar**" ao STAE central obedecendo a hierarquia administrativa do País.

Já no STAE central as fichas são armazenadas num local especialmente preparado; de onde são retiradas para a informatização. Os sectores responsáveis pelo manuseamento dos dados das fichas são o Departamento de Sufrágio e Departamento de Informática.

Obedecendo a um plano de trabalho, um funcionário do Departamento do Sufrágio retira do armazém os cadernos (fichas) por informatizar, confere e faz a entrega ao Departamento de Informática. Um **arquivista** da Informática recebe os cadernos e regista toda a informação referente a quantidades, local de recenseamento (identificado por Província, Distrito, Posto Administrativo, Localidade e Local de Recenseamento) e outro tipo de controle (por exemplo, o estado das fichas).

De seguida os cadernos são, entregues à secção de *operadores de scanner,* para procederem à leitura das fichas e como produto, obtém-se uma base de dados alfanumérica. De seguida estes dados são importados para uma base de dados relacional.

No processo de leitura, algumas fichas são rejeitadas. Estas fichas são entregues a **secção dos operadores de registo de dados** para a introdução manual dos dados. Para além da edição dos dados, os operadores fazem correcção de dados gravados.

O administrador de base de dados trata da informação sobre o processamento dos cadernos eleitorais e actualização.

O processo de actualização de recenseamento eleitoral consiste na inserção de novos eleitores, eliminação de eleitores que perderam a capacidade eleitoral no sistema, tratar das transferências.

Pretende-se um SPCE que garanta a integridade e fiabilidade dos dados referentes ao recenseamento eleitoral e fornecer informação capaz de ser usada nos diferentes processos eleitorais e referendos.

A fig.5.1 ilustra alguns intervenientes na recolha e processamento de dados eleitorais e evidencia a fronteira do presente estudo.

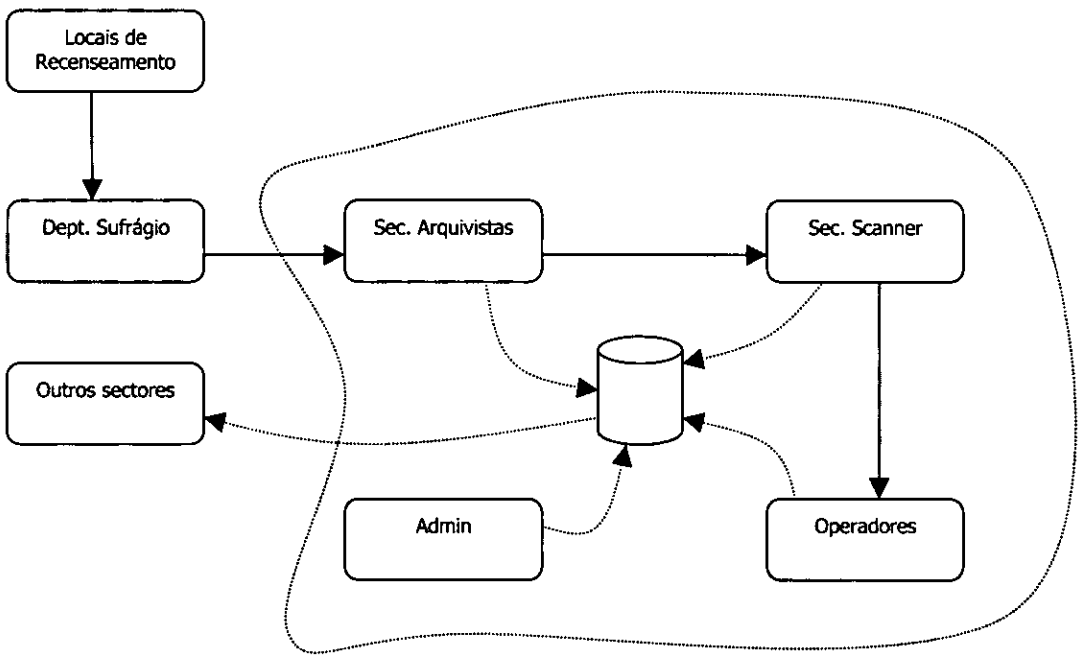


Fig. 5.1 A circulação da informação dos cadernos pelos sectores do STAE

Requisitos do SPCE.

Neste trabalho foi feita a modelação de alguns casos de uso, i.e., o sistema deverá permitir:

1. Controlar entrada de cadernos no Departamento de Informática;
2. Controlar a gravação de dados (fichas) pelos scanner;
3. Controlar o processo de correcção de fichas de eleitor;
4. Controlar o processo de actualização;
5. Consultar Dados.

5.2 Fase da Concepção

Durante esta fase do processo unificado, deve-se delimitar a fronteira do sistema, definindo como o sistema será utilizado por cada usuário, através da criação dos casos de uso mais relevantes.

Com base no modelo de casos de uso inicial pode-se desenvolver o primeiro modelo de análise, a partir da criação de diagrama de classes conceptuais. Esta fase é apresentada numa única iteração.

5.2.1 Iteração 1

5.2.1.1 Disciplina de Requisitos

A partir das considerações feitas anteriormente podemos chegar ao diagrama principal de casos de uso apresentado na fig.5.2.

Este diagrama é o primeiro passo no desenvolvimento do modelo de caso de uso. A partir deste diagrama inicial, o modelo de casos de uso crescerá incrementalmente, através do refinamento dos casos de uso, até alcançar o objectivo desta fase, delimitar a fronteira do sistema sob perspectiva de seus usuários.

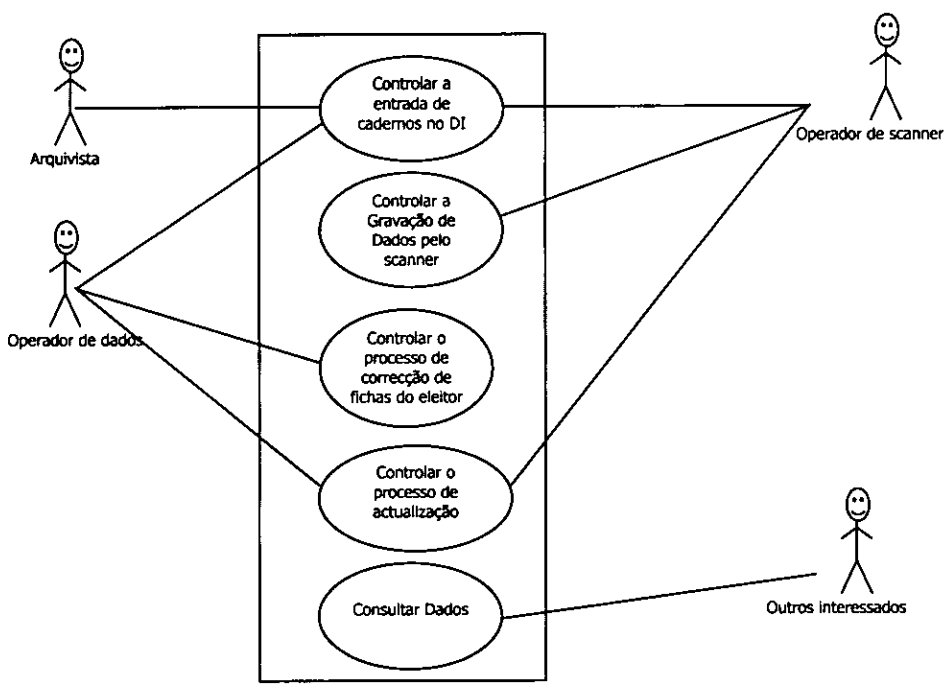


Fig.5.2 Diagrama de Caso de Uso do Sistema

O primeiro refinamento do diagrama inicial de casos de uso é realizado com objectivo de detalhar o caso de uso "Controlar Entrada de Cadernos no DI"

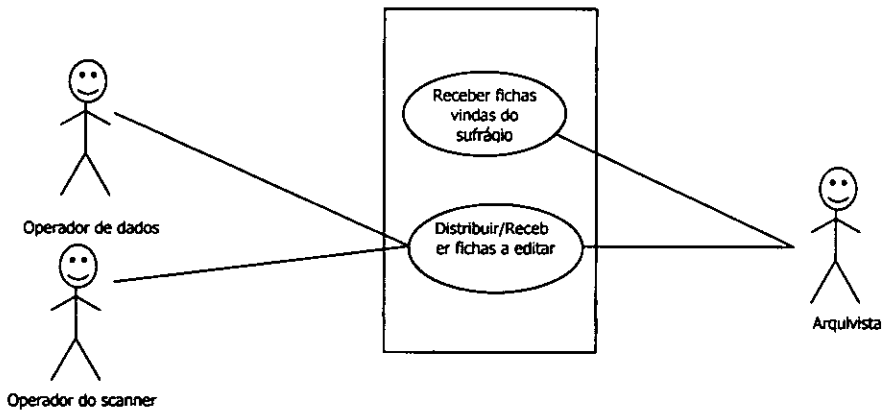


Fig.5.3 Diagrama de Caso de Uso "Refinamento do caso de uso "Controlar Entrada de Cadernos no DI"

Se neste momento, algum caso de uso apresentado no diagrama da fig. 5.3 precisasse de ser refinado, visando o objectivo da fase de concepção, a construção de diagramas de casos de uso para o refinamento poderia ser realizada imediatamente ou em outra iteração. Porém, nesta fase, não temos necessidade de refinar o diagrama da fig. 5.3, o que nos permite voltar ao diagrama principal da fig. 5.2 e detalharmos o caso de uso "Controlar a gravação de dados pelo scanner", conforme mostra a fig. 5.4.

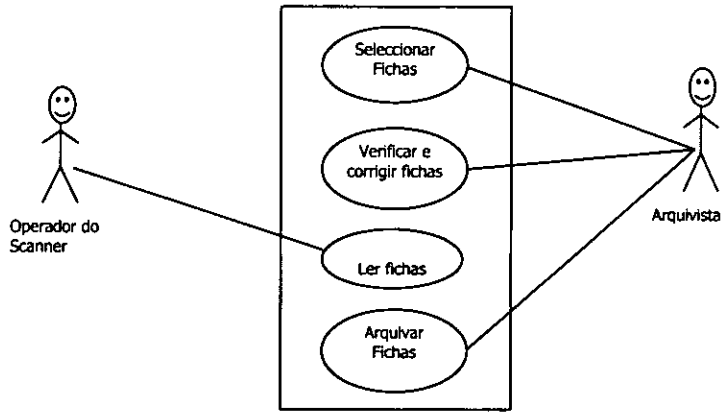


Fig.5.4 Diagrama de Caso de Uso "Refinamento do caso de uso Controlar a Gravação de dados (fichas) pelo scanner"

O diagrama da fig. 5.4 também não apresenta a necessidade de refinamento de seus casos de uso dentro da fase de concepção. Qualquer trabalho de refinamento deste diagrama seria desnecessário, tendo em vista que detalhes minuciosos na modelação dos requisitos não são importantes neste momento.

A disciplina de requisitos da fase de concepção termina com a criação do diagrama 5.4. Desta forma, obtém-se o primeiro modelo de casos de uso que servirá de base para a criação do modelo de análise nas disciplinas seguintes.

5.2.1.2 Disciplina de Análise

O papel da disciplina de análise nesta etapa do processo é identificar classes e seus relacionamentos através do estudo do modelo de casos de uso desenvolvido na disciplina anterior. Desta forma, o diagrama de classes desenvolvido neste momento deve ser o mais simples possível, visando apenas à realização em termos de classes e relacionamentos dos casos de uso definidos anteriormente.

O primeiro diagrama de classes do modelo de análise é mostrado na fig.5.5

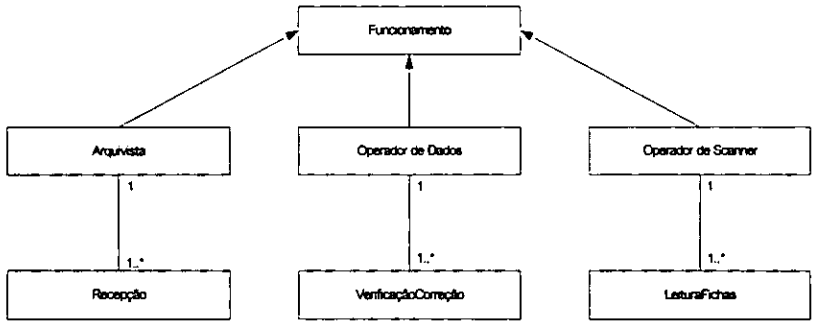


Fig. 5.5 – Diagrama de classes do modelo de análise. Fase da concepção

Podemos perceber no diagrama da fig.5.5 o relacionamento entre classes. As multiplicidade existentes nas extremidades de cada associação nos permite tirar conclusões de que, por exemplo, cada "Operador de Scanner" faz a leitura de uma ou várias "fichas" e as "ficha" no mínimo devem pertencer a um "Operador de Scanner".

O diagrama da disciplina de análise é construído de forma conceptual, não se preocupando, portanto, se a linguagem de programação e o sistema de gestão de banco de dados escolhidos disponibilizarão recursos para que os conceitos de orientação a objectos possam ser implementados, tais considerações são de responsabilidade da disciplina de projecto.

5.3 Fase de Elaboração

O principal objectivo da fase de elaboração é estabelecer a base da arquitectura que irá guiar os trabalhos nas fases de construção e transição, isto é feito com base nos requisitos que ainda não foram capturados na fase de concepção.

A fase de elaboração deste estudo de caso é desenvolvida através de duas iterações.

5.3.1 Iteração 1

5.3.1.1 Disciplina de requisitos

A disciplina de requisitos da fase de elaboração tem como objectivo a criação de diagramas de casos de uso com base nos requisitos remanescentes, ou seja, aqueles que ainda não foram identificados durante a fase de concepção. A criação destes diagramas incrementará o modelo de casos de uso, o qual servirá de base para o trabalho de complemento dos modelos de análise e projecto nas disciplinas seguintes.

O primeiro diagrama de casos de uso criado nesta etapa do processo é apresentado pela fig.5.6.

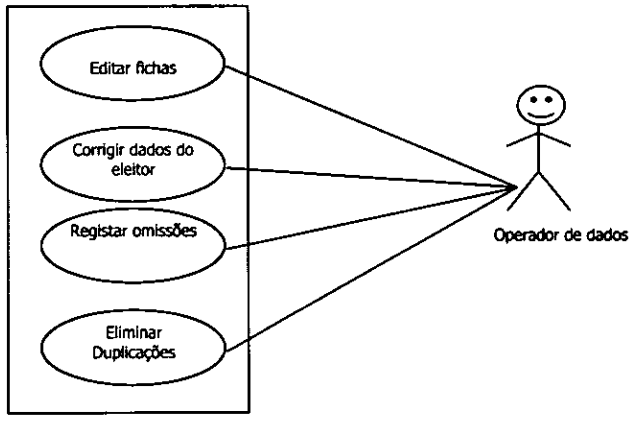


Fig.5.6 Diagrama do caso de uso "Controlar o processo de correcção de dados do eleitor"

O diagrama da fig.5.6 é um refinamento do caso de uso "Controlar o processo de correcção de fichas de eleitor" existente no diagrama de casos de uso da fig.5.2.

Outro diagrama de casos de uso desenvolvido nesta fase é apresentado na fig.5.7.

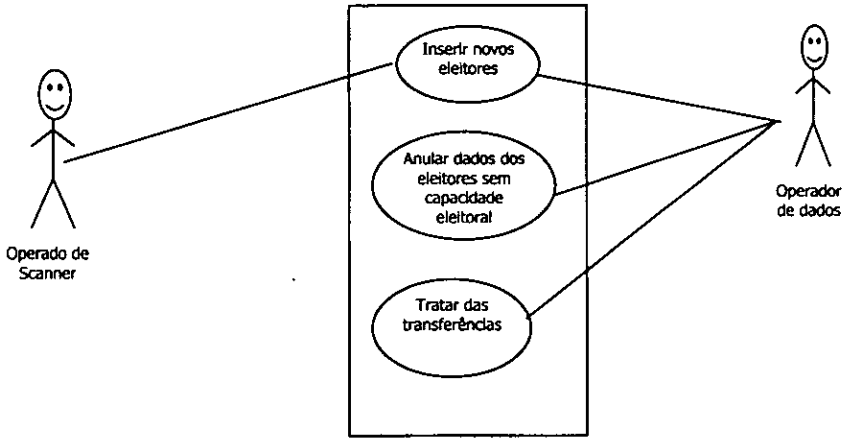


Fig.5.7 Diagrama do caso de uso "Refinamento do caso de uso Controlar o processo de actualização"

Este diagrama é um refinamento do caso de uso " Controlar o processo de actualização" existente no diagrama de casos de uso da fig. 5.2.

Observando o diagrama da fig. 5.7, percebe-se que o mesmo foi desenvolvido em um nível de abstracção diferente daquele expresso pelos diagramas de casos de uso da fase de concepção.

Na fase de concepção deste caso de estudo, os diagramas de caso de uso mostram como os actores interagem com os casos de uso que representam procedimentos existentes para a execução de determinadas actividades dentro do Departamento de Informática do STAE. Na fase de elaboração, os diagramas de casos de uso são desenvolvidos dentro do contexto de utilização do sistema em questão.

Com a criação deste diagrama concluímos o desenvolvimento do modelo de casos de uso na fase de elaboração.

5.3.1.2 Disciplina de Análise

A disciplina de análise da fase de elaboração tem como objectivo complementar o modelo de análise, inicialmente desenvolvido na fase de concepção, através da identificação de classes e seus respectivos relacionamentos mediante o estudo dos requisitos remanescentes capturados e representados através de casos de uso na disciplina anterior.

A análise de tais requisitos resultou na actualização do modelo de análise como pode ser visto no diagrama de classes apresentado na fig. 5.8.

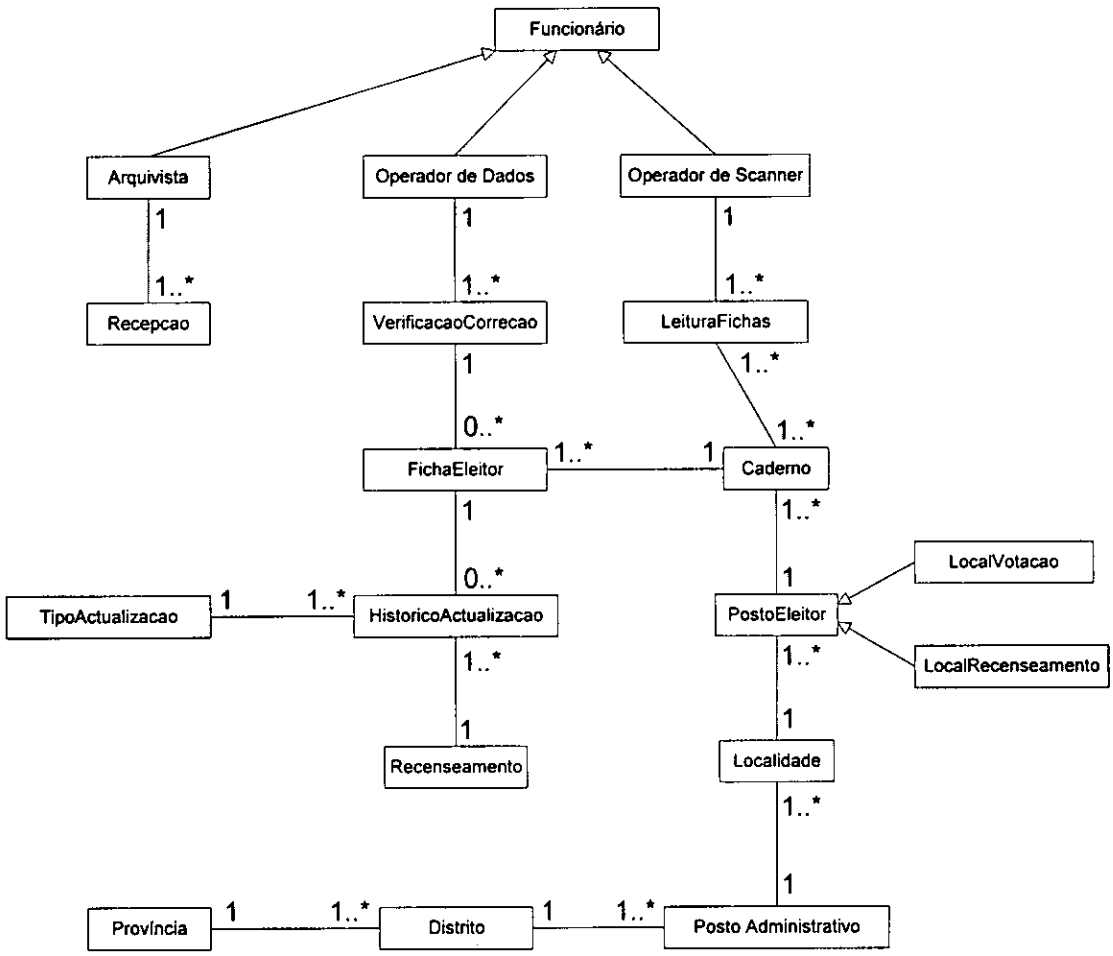


Fig. 5.8 Diagrama de classes do modelo de análise. Fase de Elaboração – Iteração 1

A partir da actualização do modelo de análise, o diagrama de classes poderá ser analisado visando à identificação de pacotes que serviriam de base para a criação de subsistemas na disciplina de projecto, na segunda iteração da fase de elaboração.

5.2.2 Iteração 2

5.2.2.1 Disciplina de Requisitos

A disciplina de requisitos durante a segunda iteração da fase de elaboração não sofrerá alterações tendo em vista que o seu trabalho durante a primeira iteração alcançou o objectivo da fase, no que diz respeito à identificação dos requisitos remanescentes.

5.2.2.2 Disciplina de Análise

O papel da disciplina de análise durante esta segunda iteração consiste, entre outras tarefas, na análise dos requisitos do modelo de casos de uso objectivando a colecta de informação necessária

para a definição dos atributos das classes do modelo de análise. O resultado deste trabalho pode ser visto na fig. 5.9.

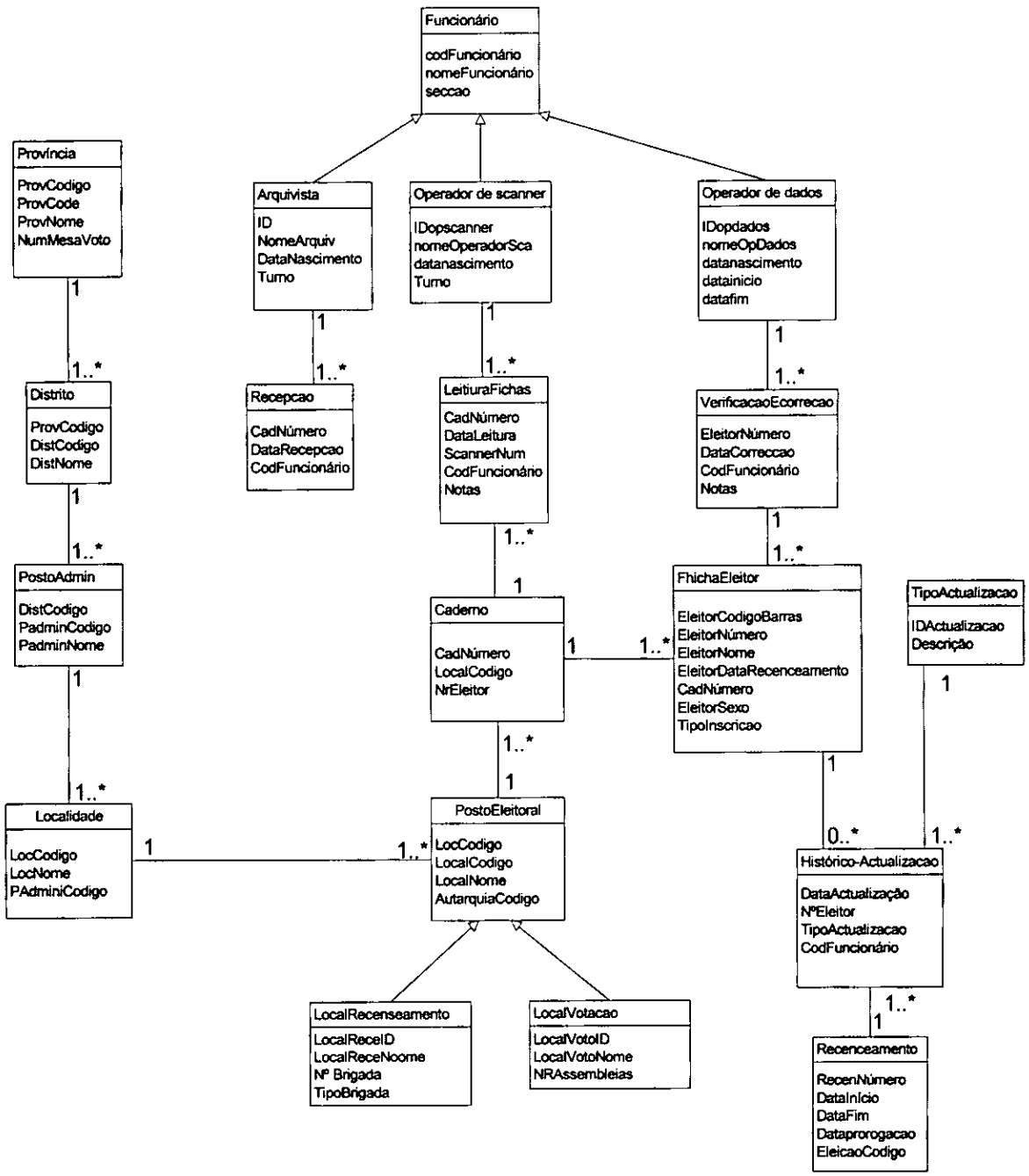


Fig. 5.9 Diagrama de classes do modelo de análise. Fase de Elaboração – Iteração 2

Outra tarefa da disciplina de análise durante esta segunda iteração é a identificação de pacotes que serviriam de base para a identificação dos subsistemas na disciplina de projecto.

Uma das tarefas desta disciplina de projecto consiste na modelação de um diagrama de classes lógico a partir do diagrama de classes conceptual resultado da disciplina de análise.

Após a criação de um diagrama de classes lógico e da definição dos atributos de cada classe, tarefas como a criação de diagramas de interacção, diagramas de gráficos e de diagramas de actividades podem ser realizadas.

A criação de diagramas de interacção foi designada a fase de construção, com o propósito de modelar o comportamento do usuário diante das interfaces gráficas do sistema e como estas interfaces interagem com o banco de dados.

O objectivo da fase de construção é produzir a versão beta do produto que está sendo desenvolvido, com base na arquitectura executável resultante da fase de elaboração. O maior trabalho da fase de construção está centrado, de facto, na disciplina de implementação.

6. Conclusões e Recomendações

A crescente complexidade dos sistemas de informação vem tornando cada vez mais relevante o papel das metodologias de desenvolvimento de sistemas de informação. A grande variedade de metodologias, associada a oferta de várias técnicas e ferramentas, leva a necessidade de se fazer uma revisão bibliográfica das metodologias antecedentes, para melhor compreender a metodologia USDP aplicada no presente trabalho.

O modelo *Waterfall* mesmo sendo o referencial da grande maioria das metodologias existentes, este apresenta algumas limitações, que são resolvidas com o Processo Unificado.

As abordagens realizadas sobre a utilização da UML dentro do Processo Unificado foram suficientes para o desenvolvimento do caso de estudo do capítulo 5. Porém, muito ainda se pode aprender com a Metodologia Unificada, tendo em vista que a mesma possibilita o desenvolvimento de sistemas de pequeno à grande porte, em todos os tipos de domínios de problemas.

É óbvio que a escolha de uma linguagem orientada a objectos possibilita o melhor aproveitamento dos recursos da Metodologia Unificada, entretanto, considerando factores que envolvem o desenvolvimento de sistemas como investimentos, custos, prazos, recursos humanos e materiais, nem sempre isto será possível. De qualquer forma, a escolha de uma ou outra linguagem de programação não impedirá que Processo Unificado alcance o seu objectivo, desenvolver um sistema obedecendo a estimativas de custos e prazos, de forma eficaz e principalmente, padronizada.

A aplicação do USDP e a UML para a modelação de sistemas permite produzir um modelo que vá de encontro aos requisitos do utilizador dada a sua flexibilidade apoiada nas suas características descritas no ponto 4.1.

O caso de estudo deste trabalho abordou a modelação conceptual do USDP, concentrando-se nas disciplinas de requisitos e análise do Processo Unificado. Por este motivo, uma sugestão para um trabalho futuro seria a implementação de um sistema usando uma linguagem orientada a objectos com o intuito de obter o máximo proveito dos recursos da Metodologia Unificada.

As fontes e os beneficiários dos dados dos eleitores são vários. Recomenda-se que se encontrem mecanismos adequados de articulação entre sectores do Ministério de Interior, Ministério da Saúde, Ministério da Justiça, Serviços da Migração entre outros ao melhoramento do processo de actualização dos dados dos eleitores.

É altura de se pensar num sistema em on-line com uma Base de Dados centralizada no STAE central que esteja ligada aos gabinetes Provinciais e ainda na introdução do cartão magnético ao

invés do cartão actualmente usado, pois o cartão magnético facilitaria a consulta dos dados do eleitor no sistema.

7. Bibliografia

- Avison, D.E., Wood-Happer, A.T. (1990). An Exploration in Information System Development Multiview, Maidenhead, McGraw-Hill.

- Avison, D.E., Fitzgerald, G. (1997). Information Systems Development Methodologies Techniques and tool, London, McGraw-Hill 2 ed.

- Alhir, S. S. (1998). UML in a natshell. A desktop quick reference, O' Reilly & Associates, inc..

- Barbieri, Carlos. (1997). A Unificação dos Métodos OO, Rio de Janeiro, Computerworrld.

- Coad, P, Yourdon, E. (1991). Object Oriented Design, New Jersey, Prentice-Hall.

- Correia, A., Valigy, I., Frank, M., Salimo, G., Ernesto, M., Archetti A., Conceição, R. (1994). Dados Estatísticos do Processo Eleitoral, Maputo, M L Graphics Lda.

- Flynn, D.J. (1992). Information Systems Requirements-Determination and Analysis, Maidenhad McGraw-Hill.

- Fumo, T. (2001). A Metodologia Orientada a Objectos e a Unified Modeling Language Aplicadas na Construção de um Sistema de Apoio à Gestão de uma Unidade Sanitária. Tese de Licenciatura Maputo, Universidade Eduardo Mondlane.

- Furlan, J. D. (1997). Modelagem de Negócios, São Paulo, Makron Books.

- Furlan, J. D. (1998). Modelagem de Objectos através da UML, Brasil, Makron Books.

- Fowler, Martin. (1999). UML Distilled, Kendall Scott.

- Gane, C., Sarson, T. (1979). Strutured Systems Analysis Tools and Techiniques, New Jersey, Prentice Hall.

- Jacobson, I. Booch, G. Rumbaugh, J. (1999). The Unified Software Development Process, Massachusetts, Addison-Wesley.

- Jacobson, I., Booch, G., Rumbaugh, J. (2000). UML Guia do Usuário, Rio de Janeiro, Campus.

- Júnior, R. (2001). Análise e Projecto Orientado a Objectos Usando UML e Processo Unificado. Tese de Licenciatura. Belém, Universidade Federal do Pará.

- Kruchten, F., Wesley, A. (1998). The Rational Unified Process An Introduction, Reading, MA..

- Laudon, K.C., Laudon, J.P. (1996). Management Information Systems Organization and Technology, New Jesey, Prentice Hall, 4.edição.

- Larman,Craig.(1997). APPLYING UML AND PATTERNS An Introduction to Object-Analysis and Design, UP per Saddle River, New Jersey, Prentice Hall.

- Mason, D., Willcocks, L. (1994). Systems Analysis, Systems Design Maidenhead, McGraw-Hill.

- Mcdermid, E.M. (1990). Software Engineering for Information Systems, Maidenhead, McGraw-Hill.

- Parkin, Adrew. (1980). Analise de Sistemas, Lisboa, Presença.

- Shah, H.U., Avison, D.E. (1995). The Information Systems Development Cicle A first curse in Information systems, Maidenhead, McGraw-Hill.

- Shlaer, S. Mellor, S. (1990). Análise de Sistemas Orientada à Objectos, São Paulo, McGraw-Hill.

- Yourdon, E. (1991). Modern Structered Anaysis, New Jersey Prentice-Hall.

- Yourdon, E. (1992). Analise Estruturada Moderna, Rio de Janeiro, Campos.

- Yourdon, E. (1993). Model – Driven Systems Development Yourdon Systems Method, New Jersey, Prentice-Hall.

- Yourdon, E. (1995). Declínio e Queda dos Analistas e Programadores, São Paulo, Makron Books.

- Yourdon, E. e Argila, C. (1996). Case studies in Object-Oriented Analysis, Prentice Hall.

- The Unified Process. [On-line]. Consultado em 20/08/2003.Disponível em: http://www.enterpriseunifiedProcess.info

- http://www.Rational.com/ http://www.ArgoUML.org.

- http://www.Rational.com/ProductsRup.

- UML. [On-line]. Consultado em 20/07/2003. Disponível em: http://uml.systemhouse.mci.com.

Anexo A

«Acrónimos e Glossário de Termos»

8. Anexo A: Acrónimos e Glossário

8.1 Abreviaturas (Acrónimos)

Símbolo/Abreviatura	Significado
RUP	*Rational Unified Process*
SI	Sistemas de Informação
SPCE	Sistema de Processamento de Cadernos Eleitorais
TI	Tecnologias de Informação
TICs	Tecnologias de Informação e Comunicações
UML	*Unified Modeling Language*
USDP	*Unified Software Development Process*

8.2 Glossário de Termos usados

Termo	Descrição
Arquitectura	Visão do projecto do sistema como um todo, destacando suas características mais importantes, mas sem entrar em detalhes.
Artefacto	A tangível parte da informação que, é criado, mudado e usado pelos trabalhadores ao desempenhar suas funções ou actividades; Representa uma área de responsabilidade.
Disciplina	é um grupo de actividades (relata artefactos) enquadradas numa área específica tais como actividades que estão dentro da análise de requisitos.
Ficha	Instrumento para recolha de dados de identificação do eleitor
Incremento	é uma pequena e manejável parte do sistema, geralmente é o delta ou diferença entre dois sucessivos desenvolvimentos. Cada iteração irá resultar em pelo menos um novo desenvolvimento, e irá deste modo adicionar um incremento para o sistema. Todavia a sequência do desenvolvimento será criado dentro de uma iteração, cada uma aumentando um pequeno incremento para o sistema.
Iteração	a distinta colocação das actividades conduzidas de acordo com o destinado (iteração), projecta e avalia critério que resulta num lançamento interna ou externa.
Requisitos	(exigência; necessidades) uma condição ou capacidade para o qual o sistema tende a confirmar.
Software	Conjunto de programas e instruções que operam o computador.
Sistema de informação	é um sistema que armazena, pesquisa, edita e visualiza informação para os utilizadores, manter grandes quantidades de dados com relacionamento complexo, que são guardados em bases de dados relacionais ou orientados a objectos.
Tecnologias Informação	É um Conjunto de conhecimentos, reflectidos quer em equipamentos e programas, quer na sua criação e utilização a nível pessoal e empresarial.
UP	**(Unified Process)** um processo de desenvolvimento do software, baseado na UML que é iterativo, centrado na arquitectura, dirigido por um caso de uso.

Anexo B

« Ficha de Inscrição »

REPÚBLICA DE MOÇAMBIQUE — RECENSEAMENTO ELEITORAL

Ficha a Informatizar

Nome Completo

JAQUELINA MOIANE

INSTRUÇÕES:

POR FAVOR MARCAR ASSIM ●
NÃO USE CANETA VERMELHA
MARQUE
APENAS UMA
VEZ EM CADA
COLUNA

EXEMPLO:

Nome Com

C	A	B	D
A	●	A	A
B	B	●	B
●	C	C	C
D	D	D	●

Número de Inscrição

0 A 1 3 8 4 5 9 / 0 8 1 9

(Transferência /2ª via)
Código anterior do eleitor

Tipo de inscrição

Nova inscrição ●
2ª via ○
Transferência ○

Sexo

M ○ F ●

Local de Recenseamento

Código do Local

Pr.	Dis.	P.A.	Loc.
0	1	0	1

Data de Inscrição

Dia		Mês		Ano	
0	0	Janeiro	○	1999	●
1	1	Fevereiro	○	2000	○
●	2	Março	○	2001	○
3	3	Abril	○	2002	○
	4	Maio	○	2003	○
	5	Junho	○		
	6	Julho	●		
	7	Agosto	○		
	8	Setembro	○		
	9	Outubro	○		
		Novembro	○		
		Dezembro	○		

Data de Nascimento

0 5 0 4 1 9 6 4

Número do Novo B.I.

0007392898