Documentação de um Produto de Software

Versão 3.0

Autora: Profª Ana Paula Gonçalves Serra

Revisor: Prof. Fernando Giorno

2005

ÍNDICE DETALHADO

PREFÁCIO **4**

1. INTRODUÇÃO AO DOCUMENTO **6**

1.1. TEMA 6
1.2. OBJETIVO DO PROJETO 6
1.3. DELIMITAÇÃO DO PROBLEMA 6
1.4. JUSTIFICATIVA DA ESCOLHA DO TEMA 6
1.5. MÉTODO DE TRABALHO 6
1.6. ORGANIZAÇÃO DO TRABALHO 7
1.7. GLOSSÁRIO 7

2. DESCRIÇÃO GERAL DO SISTEMA **8**

2.1. DESCRIÇÃO DO PROBLEMA 8
2.2. PRINCIPAIS ENVOLVIDOS E SUAS CARACTERÍSTICAS 8
2.3. REGRAS DE NEGÓCIO[G] 8

3. REQUISITOS[G] DO SISTEMA **9**

3.1. REQUISITOS FUNCIONAIS 9
3.2. REQUISITOS NÃO-FUNCIONAIS 10
3.3. PROTÓTIPO 10
3.4. MÉTRICAS E CRONOGRAMA 11

4. ANÁLISE E *DESIGN* **12**

4.1. ARQUITETURA DO SISTEMA 12
4.2. MODELO DO DOMÍNIO 12
4.3. DIAGRAMAS DE INTERAÇÃO 13
4.4. DIAGRAMA DE CLASSES 14
4.5. DIAGRAMA DE ATIVIDADES 14
4.6. DIAGRAMA DE ESTADOS 14
4.7. DIAGRAMA DE COMPONENTES 15
4.8. MODELO DE DADOS 16
4.8.1. Modelo Lógico da Base de Dados 16
4.8.2. Criação Física do Modelo de Dados 16
4.8.3. Dicionário de Dados 16
4.9. AMBIENTE DE DESENVOLVIMENTO 16
4.10. SISTEMAS E COMPONENTES EXTERNOS UTILIZADOS 16

5. IMPLEMENTAÇÃO **17**

6. TESTES **18**

6.1. PLANO DE TESTES 18
6.2. EXECUÇÃO DO PLANO DE TESTES 18

7. IMPLANTAÇÃO **19**

7.1. DIAGRAMA DE IMPLANTAÇÃO 19
7.2. MANUAL DE IMPLANTAÇÃO 19

8. MANUAL DO USUÁRIO **20**

9. CONCLUSÕES E CONSIDERAÇÕES FINAIS 21

BIBLIOGRAFIA 22

COMENTÁRIOS SOBRE A DOCUMENTAÇÃO 24

1. COMO UTILIZAR O PRODUTO DE DOCUMENTAÇÃO NO TG? 24

2. NO MEU TG IREI UTILIZAR MODELAGEM ESTRUTURADA, QUAL É A DIFERENÇA? ..24

3. NO MEU TG HAVERÁ UM ESTUDO TEÓRICO ALÉM DO SISTEMA, QUAL É A DIFERENÇA? 25

4. QUAL É O MATERIAL QUE POSSO CONSULTAR CASO TENHA DÚVIDAS? 25

5. QUAIS SÃO AS FERRAMENTAS CASE QUE PODEM SER UTILIZADAS? 25

GLOSSÁRIO 28

Prefácio

O objetivo deste documento é fornecer um roteiro para o desenvolvimento de sistemas de software utilizando os princípios da engenharia de software orientada a objetos com notação UML (*Unified Modeling Language*). É destinado a todos os alunos da Universidade São Judas Tadeu dos cursos de Ciência da Computação, Sistemas de Informação e Processamento de Dados, apoiando as disciplinas de Metodologia de Desenvolvimento de Sistemas, Engenharia de Software I, Engenharia de Software II, entre outras, além do Trabalho de Graduação (TG).

Esta é a versão 3.0 do documento, totalmente revisada para utilizar a notação UML e modelos do RUP (*Rational Unified Process* – Processo Unificado *Rational*). Neste documento são citados alguns modelos do RUP que podem ser utilizados e consultados na ferramenta *Rational Unified Process* (que faz parte da ferramenta *Rational Suite Enterprise*) ou o site da IBM. A escolha da orientação a objetos é devido à tendência de mercado, mas nada impede que o roteiro seja seguido no caso de opção pela Modelagem Estruturada (ver item Comentários sobre a Documentação).

No final deste documento há um glossário, os termos que constam no glossário são representados no documento pela letra [G] em azul.

Sugestões e Comentários podem ser enviados para prof.anapaula@usjt.br.

Modelo da Documentação

Esta é a parte mais importante do texto pois apresenta um roteiro de documentação orientado a objetos de sistemas de software utilizando notação UML, desde a fase inicial do projeto de software até a sua implantação.
Para a criação dos diagramas aconselha-se a utilização de alguma ferramenta CASE [G].

1. Introdução ao Documento

O objetivo deste capítulo é apresentar o projeto. Para tal, deve-se desenvolver um texto, com as seguintes características: impessoalidade, objetividade, clareza, precisão, coerência e concisão. A introdução deve abrange os itens a seguir.

1.1. Tema

Neste item deve-se apresentar o tema do projeto, de forma clara e objetiva.

1.2. Objetivo do Projeto

Neste item devem ser descritos os objetos gerais e específicos do projeto como um todo. Independente do que será implementado, este item visa o entendimento global do projeto.

1.3. Delimitação do Problema

Neste item deve ser descrita a delimitação do problema, que define o ponto central do projeto. Isso quer dizer que, dentro de uma idéia geral do projeto, deve-se ressaltar a idéia específica efetivamente a ser desenvolvida. É neste item que a amplitude do projeto tem sua delimitação perfeitamente definida.

1.4. Justificativa da Escolha do Tema

Neste item deve-se expor a motivação acadêmica para a elaboração do projeto em questão, detalhando os motivos de ordem teórica ou de ordem prática para a sua realização.

1.5. Método de Trabalho

Neste item deve-se descrever o método a ser utilizado para realização do projeto, o tipo de processo de desenvolvimento de software[1], a modelagem a ser utilizada (orientada a objeto, estruturada, outras).

[1] Para maiores detalhes dos tipos de processos de desenvolvimento de software consultar o livro Engenharia de Software – Roger Pressman – 5ª edição - Capítulo 2.

1.6. Organização do Trabalho

Neste item deve-se descrever como o documento estará organizado.

1.7. Glossário

Neste item deve-se definir os termos importantes utilizados no projeto, facilitando o seu entendimento. Caso exista um número extenso de termos no projeto consultar e utilizar o modelo rup_gloss.dot – artefato do RUP.

2. Descrição Geral do Sistema

Este capítulo tem como objetivo descrever de forma geral o sistema, o escopo e as principais funções. A descrição geral do sistema deve abrange os itens a seguir. Como referência pode-se consultar e utilizar o modelo rup_vision_sp.dot – artefato do RUP.

2.1. Descrição do Problema

Neste item deve ser descrito o problema que será resolvido com o desenvolvimento do sistema. As questões a seguir devem ser respondidas.

- ✓ Quem é afetado pelo sistema?
- ✓ Qual é o impacto do sistema?
- ✓ Qual seria uma boa solução para o problema?

2.2. Principais Envolvidos e suas Características

2.1.1. Usuários do Sistema

Neste item deve ser descrito para qual tipo de empresa se destina o sistema, os tipos de usuários que utilizarão o sistema.

Estas informações são importantes para a definição de usabilidade [G] do sistema.

2.1.2. Desenvolvedores do Sistema

Neste item deve ser descrito os tipos de pessoas envolvidas em todo o desenvolvimento do sistema direta ou indiretamente.

Estas informações são importantes para a distribuição de responsabilidades e pontos-focais de desenvolvimento.

2.3. Regras de Negócio[G]

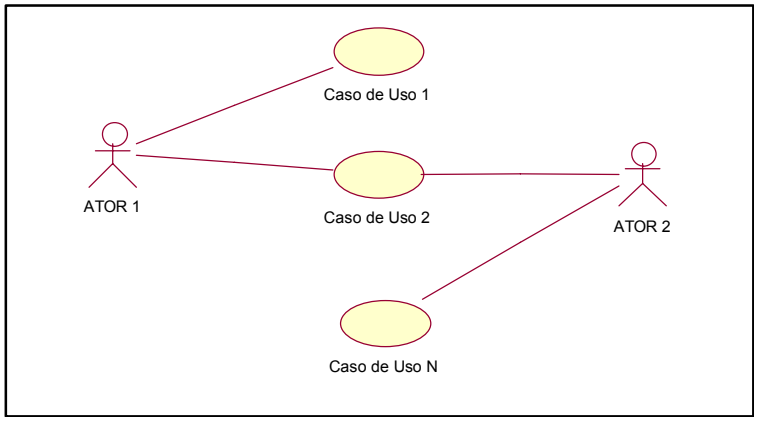
Neste item devem ser descritas as regras de negócio relevantes para o sistema, como por exemplo, restrições de negócio, restrições de desempenho, tolerância à falhas, volume de informação a ser armazenada, estimativa de crescimento de volume, ferramentas de apoio, etc.

3. Requisitos [G] do Sistema

Este capítulo tem como objetivo descrever os requisitos do sistema. No caso de sistemas que possuam usuários / solicitantes reais para o levantamento de requisitos, pode-se utilizar o modelo de documento de entrevista com usuários do RUP de Solicitações dos Principais Envolvidos (rup_stkreq.dot).

3.1. Requisitos Funcionais

Neste item devem ser apresentados os requisitos funcionais que especificam ações que um sistema deve ser capaz de executar, ou seja, as funções do sistema. Os requisitos funcionais geralmente são melhor descritos em diagramas de caso de uso, juntamente com o detalhamento dos atores e de cada caso de uso.
A seguir é apresentada a notação básica de um diagrama de caso de uso.



Notação básica do diagrama de caso de uso.

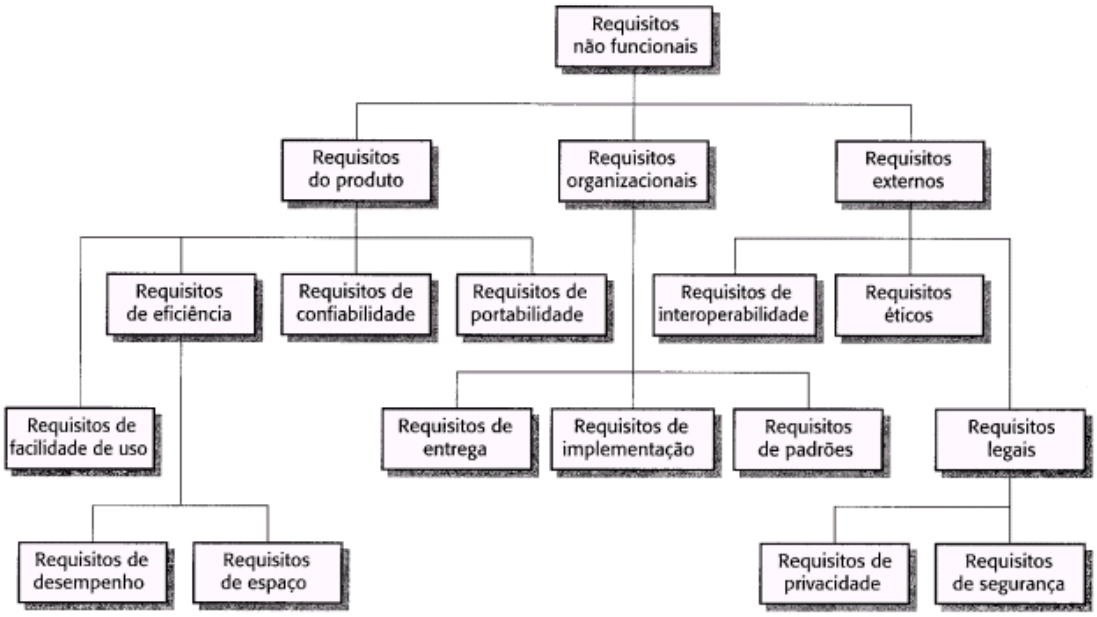
Cada ator do diagrama de caso de uso deve ser descrito de forma sucinta (2 linhas) e cada caso de uso deve ser especificado. A seguir são apresentados itens básicos para a especificação dos casos de uso do diagrama.

- ✓ Nome do Caso de Uso
- ✓ Breve descrição
- ✓ Atores envolvidos
- ✓ Pré-condições
- ✓ Seqüência de Eventos ou Fluxo Principal de Eventos
- ✓ Pós-condições;
- ✓ Exceções ou Fluxo Secundário de Eventos
- ✓ Observações

Para maiores detalhes de especificação de casos de uso consultar e utilizar o modelo rup_ucspec.dot – artefato do RUP.

3.2. Requisitos Não-Funcionais

Neste item devem ser apresentados os requisitos não funcionais, que especificam restrições sobre os serviços ou funções providas pelo sistema. A seguir são apresentados alguns tipos de requisitos não funcionais. Para maiores detalhes de requisitos não-funcionais consultar e utilizar o modelo de documento rup_ucspec.dot – artefato do RUP.



Sommerville, Ian. Engenharia de Software, 6ª edição.

3.3. Protótipo

Neste item deve ser apresentado o protótipo do sistema que consiste na interface preliminar contendo um subconjunto de funcionalidades e telas. O protótipo deve ser incrementalmente evoluído até a concordância completa dos requisitos previstos para o sistema, de comum acordo com o usuário. O protótipo é um recurso que deve ser adotado como estratégia para levantamento, detalhamento, validação de requisitos e modelagem de interface com o usuário (usabilidade).

As telas do sistema podem ser criadas na própria linguagem de desenvolvimento ou em qualquer outra ferramenta de desenho. Cada tela deve possuir uma descrição detalhada do seu funcionamento. Alguns itens importantes na descrição são:

- Objetivo da tela;
- De onde é chamada e que outras telas pode chamar;
- Regras:
  - ✓ Domínio (tamanho de campo, tipo de dados que aceita valor default);
  - ✓ Tipo de usuários que podem acessar;

- ✓ Lógica de negócio (campos obrigatórios, validade entre datas, preenchimento anterior de um campo para efetuar uma operação, etc).

A descrição detalhada das telas deve registrar informações que possam ser consultadas na implementação do sistema, facilitando, agilizando e minimizando erros de implementação e na execução de testes.

3.3.1. Diagrama de Navegação

Neste item deve ser apresentada a seqüência de navegação das telas.

3.4. Métricas e Cronograma

Neste item devem ser estimados os esforços necessários em termos de recursos alocados [G] e tempo para a obtenção do sistema. Para realizar a estimativa, indicam-se o uso de alguma técnica de métrica, como Pontos de Função ou Pontos de Caso de Uso.

Após os cálculos de métricas deve-se elaborar o cronograma detalhado do sistema, que contempla todas as tarefas descritas e os recursos alocados para cada tarefa, com datas para início e término de cada atividade. A seqüência das tarefas e a divisão entre os recursos devem ser realizadas de acordo com o processo de desenvolvimento de software escolhido para o desenvolvimento do sistema, descrito no item 1.5.

Para elaboração do cronograma pode-se utilizar uma ferramenta como o Microsoft Project.

4. Análise e *Design*

Este capítulo tem como objetivo analisar e detalhar a solução do sistema de acordo com os requisitos levantados e validados no capítulo 3. Para isso, deve-se ter uma visão geral da arquitetura do sistema e a modelagem da solução do sistema através de diagramas. Para maiores detalhes pode-se consultar artefatos do RUP da fase de análise e *design*.

4.1. Arquitetura do Sistema

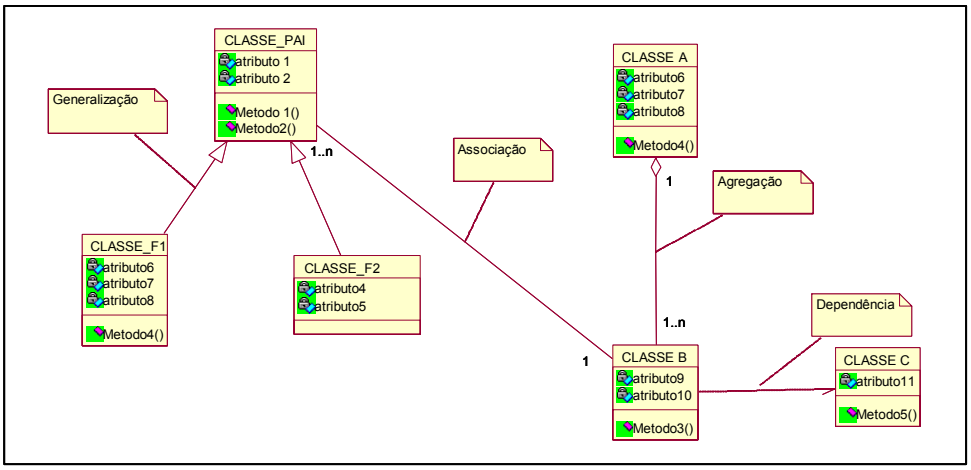
Neste item deve ser apresentada a arquitetura de infra-estrutura do sistema, demonstrando o tipo de arquitetura que será utilizada (por exemplo, cliente/servidor de n-camadas), a configuração de hardware, de rede e de software a serem utilizados, bem como o dimensionamento mínimo de conexões.

4.2. Modelo do Domínio

Neste item deve ser apresentado o modelo do domínio, que representa um primeiro modelo conceitual do diagrama de classes. Posteriormente, esse diagrama deve ser validado e complementado para compor o diagrama de classes final.

O diagrama de classes deve possuir todas as classes identificadas do sistema, deve conter os atributos e métodos de cada classe, e os relacionamento entre elas.

A seguir é apresentada a notação básica de um diagrama de classes



Notação básica do diagrama de classes.

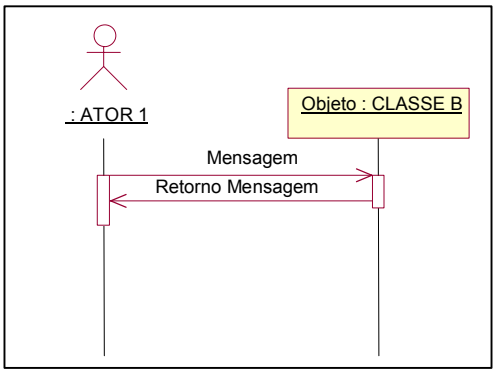
4.3. Diagramas de Interação

O diagrama de interação é composto pelos diagramas de seqüência e colaboração (comunicação, versão 2.0 UML) e modela os aspectos dinâmicos do sistema, mostrando a interação formada por um conjunto de objetos permitindo identificar mensagens que poderão ser enviadas entre eles.

4.3.1. Diagrama de Seqüência

Neste item devem ser apresentados os diagramas de seqüência essenciais ao sistema. Um diagrama de seqüência representa interações de objetos organizadas em uma seqüência temporal, apresentando os objetos que participam da interação e a seqüência das mensagens trocadas. O diagrama de seqüência deve validar o diagrama de classes e vice-versa.

A seguir é apresentada a notação básica de um diagrama de seqüência.



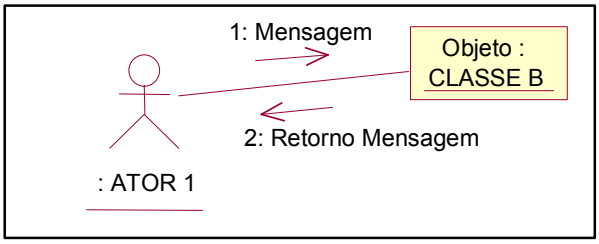
Notação básica do diagrama de seqüência.

4.3.2. Diagrama de Colaboração / Comunicação

Esse diagrama é uma alternativa para o diagrama de seqüência (item 4.3.1). Neste item devem ser apresentados os diagramas de colaboração/comunicação essenciais ao sistema. Um diagrama de colaboração descreve um padrão de interação entre objetos, apresentando os objetos que participam da interação bem como os seus links e mensagens trocadas.

Geralmente as ferramentas CASE geram automaticamente o diagrama de colaboração/comunicação a partir do diagrama de seqüência.

A seguir é apresentada a notação básica de um diagrama de colaboração/comunicação.



Notação básica do diagrama de colaboração.

4.4. Diagrama de Classes

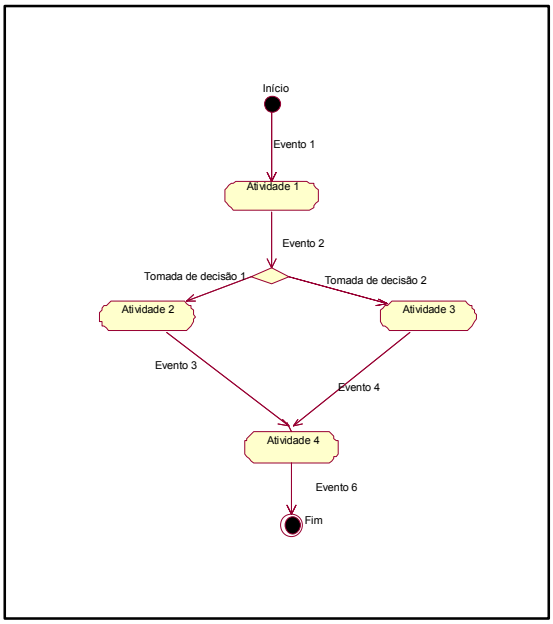
Neste item deve ser apresentado o diagrama de classes completo e validado.

4.5. Diagrama de Atividades

Neste item deve ser apresentado o diagrama de atividades, que representa o detalhamento de tarefas e o fluxo de uma atividade para outra de um sistema.

Nem todos os sistemas necessitam da elaboração do diagrama de atividades, pois nem todas as tarefas do sistema necessitam de um detalhamento. Com isso, deve-se analisar a real necessidade e no que este diagrama irá auxiliar na implementação do sistema, como: detalhamento de *workflow*, de métodos, entre outros.

A seguir é apresentada a notação básica de um diagrama de atividades.



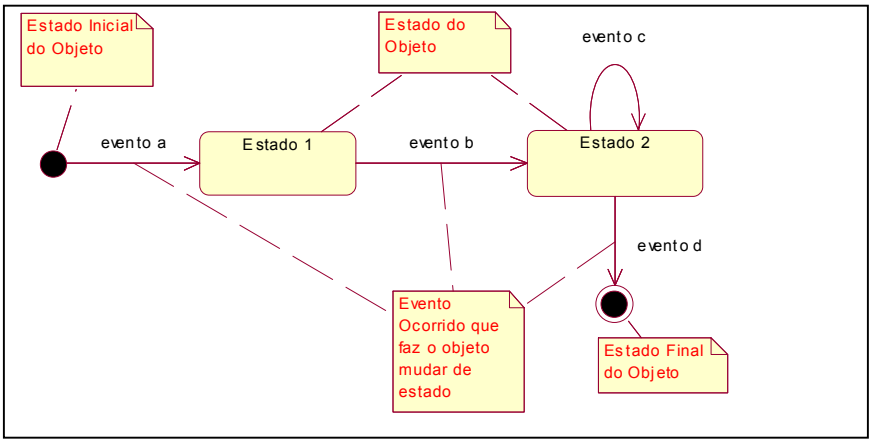
Notação básica do diagrama de atividades.

4.6. Diagrama de Estados

Neste item deve ser apresentado o diagrama de estados, que especifica as seqüências de estados pelas quais o objeto pode passar durante seu ciclo de vida em resposta a eventos.

Nem todos as classes necessitam da elaboração do diagrama de estados, pois nem todas as classes mudam muito de estado no seu ciclo de vida. Com isso, deve-se analisar a real necessidade desse diagrama para o desenvolvimento do sistema.

A seguir é apresentada a notação básica de um diagrama de estados.

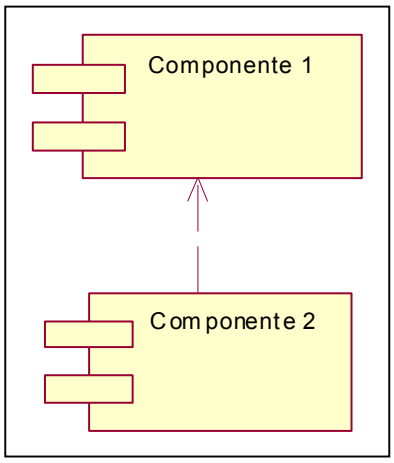


Notação básica do diagrama de estados.

4.7. Diagrama de Componentes

Neste item deve ser apresentado o diagrama de componentes que apresenta a organização e as dependências entre os componentes [G].

A seguir é apresentada a notação básica de um diagrama de componentes.



Notação básica do diagrama de componentes.

4.8. Modelo de Dados

4.8.1. Modelo Lógico da Base de Dados

Neste item deve ser apresentado o modelo lógico da base de dados, que pode ser o modelo entidade-relacionamento ou objeto da base de dados. No caso do modelo entidade-relacionamento o modelo lógico deve passar por todas as regras de normalização.

Como base para geração do modelo lógico pode-se utilizar o diagrama de classes. Geralmente ferramentas CASE geram automaticamente o modelo lógico da base de dados a partir do diagrama de classes.

4.8.2. Criação Física do Modelo de Dados

Neste item deve ser realizada a criação física do banco de dados, ou seja, a criação de *scripts*.

4.8.3. Dicionário de Dados

Neste item deve ser criado o dicionário de dados do banco de dados, com o obejtivo de documentar todas as tabelas, atributos, *stored procedures* [G].

4.9. Ambiente de Desenvolvimento

Neste item devem ser apresentados os softwares de desenvolvimento (linguagem de programação, banco de dados, ferramentas, etc.), equipamentos de hardware e redes que sejam essenciais para o desenvolvimento do sistema.

4.10. Sistemas e componentes externos utilizados

Neste item devem ser descritos os sistemas e componentes externos que serão utilizados no sistema. Por exemplo, sistemas que serão integrados ao sistema desenvolvido, componentes comprados ou livre que estão sendo utilizados para facilitar ou complementar o desenvolvimento do sistema.

5. Implementação

Este capítulo tem como objetivo a implementação das classes em termos de componentes, ou seja, toda a implementação deve ser realizada de acordo com as definições das fases anteriores e todos os recursos da programação orientada a objetos que a linguagem escolhida oferece.

Geralmente ferramentas CASE geram automaticamente pseudocódigos fontes (dependendo da linguagem utilizada) baseados no diagrama de classes.

Algumas boas práticas de programação devem ser seguidas para um maior entendimento do código. Algumas delas são:

- Cabeçalho de funções contendo campos como descrição, data de criação, autor, etc;
- Comentários no código;
- Padronização de nomes de variáveis, parâmetros, funções, tabelas, *stored procedures*, etc;
- Verificação de declaração das variáveis;
- Tratamento de erros;
- Criação de *stored procedures* para acesso aos dados da base;
- Utilização de padrões de projeto [G];
- Otimização do código, utilizando os "melhores" algoritmos e funções de recursividade.

6. Testes

Este capítulo tem como objetivo identificar defeitos no sistema, validar as funções do sistema, verificar se os requisitos foram implementados de forma adequada e avaliar a qualidade do software.

Para maiores detalhes pode-se consultar artefatos do RUP da fase de Testes.

6.1. Plano de Testes

Neste item deve ser criado o plano de testes do sistema, permitindo a validação do sistema por parte do desenvolvedor, através da verificação dos requisitos do sistema desenvolvido. Inicialmente, identificam-se os requisitos técnicos e funcionais do sistema, e listam-se todas as situações que podem ocorrer com o sistema (essas situações podem ser elaboradas através do diagrama de caso de uso e dos diagramas de seqüência). Deve-se realizar testes de consistência de campos, funcionalidades, desempenho, etc. O Plano de Testes do Sistema deverá conter, no mínimo.

- ✓ Nº do Teste;
- ✓ Descrição do Teste;
- ✓ Resultado Esperado**.**

Por conter todos os testes do sistema, este plano poderá ser um anexo na documentação do sistema. Alguns tipos de testes a serem realizados são: teste de funcionalidades, teste de usabilidade, teste de desempenho, teste de carga, teste de *stress*, teste de volume, teste de segurança e controle de acesso, teste de tolerância a falhas e recuperação, teste de configuração, teste de instalação, etc..

Para maiores detalhes consultar o modelo de documento de plano de testes do RUP rup_tstpln.dot.

6.2. Execução do Plano de Testes

Neste item devem ser registrados os testes realizados no sistema tendo como base o Plano de Testes do Sistema.

O registro dos testes deve conter a identificação do sistema, o nome do realizador dos testes e a configuração do ambiente onde foi realizado o teste. Além disso, para cada teste, deve-se ter os seguintes dados:

- ✓ Nº do teste;
- ✓ Resultado Obtido; e
- ✓ Comentários (se necessário).

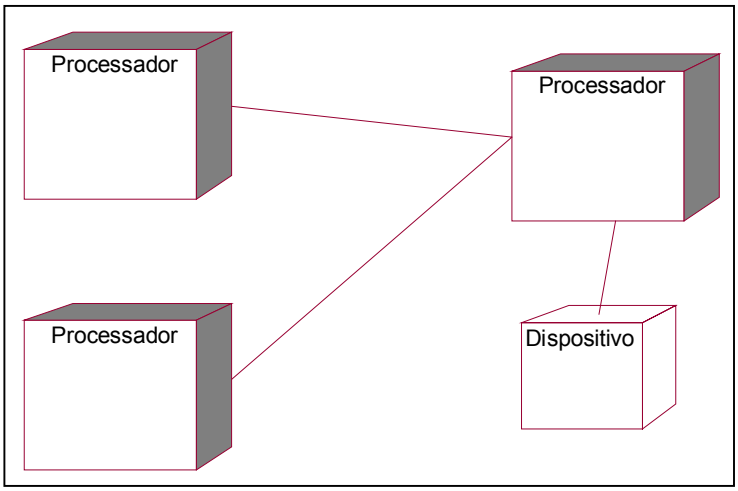
7. Implantação

Este capítulo tem como objetivo apresentar informações relevantes para a implantação e funcionamento do sistema.

7.1. Diagrama de Implantação

Neste item deve ser apresentado o diagrama de implantação que representa a parte física do sistema, exibindo os dispositivos, as máquinas de processamento em tempo de execução e os componentes que nelas serão instalados.

A seguir é apresentada a notação básica de um diagrama de implantação.



Notação básica do diagrama de implantação.

7.2. Manual de Implantação

Neste item deve ser elaborado o manual de instalação. Este manual deve conter a descrição passo a passo de como deve ser realizada a instalação do sistema.

Para maiores detalhes pode-se consultar artefatos do RUP da fase de Instalação.

8. Manual do Usuário

Este capítulo tem como objetivo a elaboração de um manual do usuário. Este manual deve conter a descrição passo a passo de como utilizar o sistema.

Para maiores detalhes pode-se consultar artefatos do RUP da fase de Instalação.

9. Conclusões e Considerações Finais

Este capítulo tem como objetivo apresentar e demonstrar a aplicabilidade dos resultados obtidos, suas limitações, inovação, possíveis integrações com outros projetos e continuação do sistema em trabalhos futuros.

Bibliografia

Neste item devem-se apresentar todas as obras (livros, artigos, Internet, revistas, etc...) utilizadas na elaboração da documentação e na implementação do projeto.

Comentários sobre a Documentação

Esta é uma parte que responde a algumas dúvidas freqüentes referente à documentação. Caso o leitor esteja utilizando esse documento para elaboração do TG, leia com a atenção o item 1, 2 e 3 desta seção.

Comentários sobre a documentação

1. Como utilizar o produto de documentação no TG?

Esse documento tem como objetivo fornecer um roteiro que auxilie os alunos no desenvolvimento de software orientado a objetos, utilizando notação UML. Com isso, o roteiro deve ser algo flexível e adaptável a cada projeto. Cada grupo deve analisar e decidir em conjunto com o seu professor orientador os itens do documento que se aplicam para o projeto escolhido e se não há necessidade de criar ou substituir itens, informações e diagramas complementares ao roteiro.

Além da aplicabilidade dos itens do documento, cada grupo deve decidir com o seu orientador o processo a ser utilizado ao longo do desenvolvimento, as etapas que devem ser cumpridas em cada reunião de orientação e o tipo de modelagem e implementação a ser utilizada. O desenvolvimento orientado a objetos é sugerido pela tendência de mercado, caso a escolha seja por um desenvolvimento estruturado ver item 2 desta seção.

2. No meu TG irei utilizar Modelagem Estruturada, qual é a diferença?

A diferença básica está nos diagramas e na implementação. A seguir são apresentados os itens que podem ser substituídos no roteiro e qual seria um item correspondente para a modelagem estruturada.

- ✓ **Item 3.1. Requisitos Funcionais:** apesar do diagrama de caso de uso ser da modelagem orientada a objetos, este pode ser utilizado na modelagem estruturada para identificar os usuários, os sistemas externos e as funções do sistema. Caso o diagrama de caso de uso não seja utilizado deve-se realizar uma descrição textual e detalhada de cada requisito;
- ✓ **Item 4.2 e 4.4. Diagrama de Classes:** deve-se substituir o diagrama de classes pelo diagrama de fluxo de dados (DFD) – nível 0 (diagrama de contexto), nível 1, nível 2,...nível n se necessário.
- ✓ **Item 4.3. Diagrama de Interação:** devem-se substituir os diagramas de interação pelo dicionário de dados dos diagramas de fluxos de dados;
- ✓ **Item 4.5. Diagrama de Atividades:** apesar do diagrama de atividades ser da modelagem orientada a objetos, este pode ser utilizado na modelagem estruturada para o detalhamento de funções complexas do sistema. Isso também pode ser realizado através de fluxogramas, diagramas de Nassi-Schneiderman, portugol, etc;
- ✓ **Item 4.7. Diagrama de Componentes:** deve-se substituir o diagrama de componentes pelo diagrama hierárquico de funções (DHF)
- ✓ **Item 5. Implementação:** deve-se utilizar toda a modelagem criada para implementação do sistema utilizando os recursos da programação estruturada;
- ✓ **Item 6.1. Plano de Teste:** o plano de teste deve seguir o item 6.1, mas as situações de teste podem ser extraídas dos diagramas de fluxo de dados (DFD).
- ✓ **Item 7.1. Diagrama de Implantação:** apesar do diagrama de implantação ser da modelagem orientada a objetos, este pode ser utilizado na modelagem estruturada para representar a parte física do sistema.

Para maiores detalhes sobre Modelagem Estruturada consultar o livro: Análise Estruturada Moderna. Edward Yourdon. Editora Campus, 3ª edição.

3. No meu TG haverá um estudo teórico além do sistema, qual é a diferença?

A principal diferença é criação de um capítulo intermediário entre o Capítulo 1 e 2, que deve apresentar todo o estudo teórico da pesquisa. Esse capítulo deve conter basicamente os itens a seguir, mas cabe ao professor orientador definir os itens a serem criados no estudo teórico.

- ✓ Apresentação do Trabalho Teórico;
- ✓ Justificativa do Trabalho Teórico e a relação com o projeto a ser desenvolvido;
- ✓ Dissertação do Trabalho Teórico;
- ✓ Análise dos resultados/Conclusão do Trabalho Teórico.

4. Qual é o material que posso consultar caso tenha dúvidas?

- ✓ FOWLER, M.; SCOTT.K. **UML Essencial**. Editora Bookman, 3ª edição.
- ✓ BOOCH, G.; RUMBAUGH, J.; JACOBSON. I. **UML – Guia do Usuário**. Editora Campus, 2000.
- ✓ BOGGS, W.; BOGGS, M. **Mastering UML com Rational Rose 2002**. Editora Alta Books, 2002.
- ✓ QUATRANI. T. **Modelagem Visual com Rational Rose 2000 e UML**. Editora Ciência Moderna, 2001.
- ✓ PRESSMAN, R. Software Engineering. A Practitioner´s Approach. 5ª edição, 2003. McGrawHill.
- ✓ SOMERVILLE, I. **Engenharia de Software**. Addison Wesley, 6ª edição.
- ✓ YOURDON, E **Análise Estruturada Moderna**. Editora Campus, 3ª edição.
- ✓ Ferramenta *Rational Unified Process*.
- ✓ Site: http://prof.usjt.br/anapaula (disciplinas de Metodologia de Desenvolvimento de Software e Engenharia de Software)

5. Quais são as Ferramentas CASE que podem ser utilizadas?

- ✓ *Rational Suite Enterprise*
  - *Rational Rose* para modelagem de sistemas orientados a objetos;
  - *Rational Unified Process* - RUP para utilização do processo de desenvolvimento e para consulta de artefatos, definições, etc;
  - *Requisite Pro* para criação e gerenciamento de requisitos.
  - Além de outras ferramentas da Suíte...

- ✓ *System Architect* para modelagem de sistemas orientados a objetos e estruturados.
- ✓ *Microsoft Project* para elaboração de cronograma, planejamento e gerenciamento do projeto.
- ✓ *Erwin* para elaboração do modelo lógico e físico do banco de dados;
- ✓ *ArgoUML* para modelagem de sistemas orientados a objetos.
- ✓ *Dome* para modelagem de sistemas orientados a objetos.
- ✓ Outras....

Glossário

Glossário

- ✓ **Componente:** representa uma parte física da implementação de um sistema, que inclui código de software, com o objetivo de criar código de software coeso para sua reutilização e facilidade de manutenção.

- ✓ **Ferramenta CASE (*Computer Aided Software Engineering*):** é uma ferramenta que auxilia no processo de desenvolvimento de software, ajudando a garantir a qualidade do projeto e facilitando a criação de modelos, documentos.

- ✓ **Padrões de Projeto (*design patterns*):** são soluções simples para problemas específicos no projeto de software orientado a objetos. Padrões de projeto capturam soluções que foram desenvolvidas e aperfeiçoadas ao longo do tempo.

- ✓ **Recursos alocados:** pessoas que irão trabalhar no projeto.

- ✓ **Regras de negócio**: declarações e regras da política ou condição que deve ser satisfeita no âmbito do negócio.

- ✓ **Requisito:** um requisito descreve uma condição ou capacidade à qual um sistema deve se adaptar, sejam necessidades dos usuários, um padrão ou uma especificação.

- ✓ ***Stored Procedures*:** é uma rotina escrita através de comandos SQL, que tem como objetivo encapsular o processo de negócio e sua reutilização. As *stored procedures* ficam armazenadas no gerenciador de banco de dados.

- ✓ **Usabilidade:** é a qualidade da interface homem-máquina, que permite que o usuário realize com eficiência e conforto as atividades a que o sistema se destina.