Capítulo 1

Projeto

Neste capítulo do projeto de engenharia veremos questões associadas ao projeto do sistema, incluindo protocolos, recursos, plataformas suportadas, inplicações nos diagramas feitos anteriormente, diagramas de componentes e implantação. Na segunda parte revisamos os diagramas levando em conta as decisões do projeto do sistema.

1.1 Projeto do sistema

Depois da análise orientada a objeto desenvolve-se o projeto do sistema, qual envolve etapas como a definição dos protocolos, da interface API, o uso de recursos, a subdivisão do sistema em subsistemas, a alocação dos subsistemas ao hardware e a seleção das estruturas de controle, a seleção das plataformas do sistema, das bibliotecas externas, dos padrões de projeto, além da tomada de decisões conceituais e políticas que formam a infraestrutura do projeto.

Deve-se definir padrões de documentação, padrões para o nome das classes, padrões de retorno e de parâmetros em métodos, características da interface do usuário e características de desempenho.

Segundo [**?**, **?**], o projeto do sistema é a estratégia de alto nível para resolver o problema e elaborar uma solução. Você deve se preocupar com itens como:

1. Protocolos

    - Definição dos protocolos de comunicação entre os diversos elementos externos (como dispositivos). Por exemplo: se o sistema envolve o uso dos nós de um cluster, devem ser considerados aspectos como o protocolo de comunicação entre os nós do cluster.
        - Neste projeto blablabla
    - Definição dos protocolos de comunicação entre os diversos elementos internos (como objetos).
        - Neste projeto blablabla

- Definição da interface API de suas bibliotecas e sistemas.
  - Neste projeto blablabla
- Definição do formato dos arquivos gerados pelo software. Por exemplo: prefira formatos abertos, como arquivos txt e xml.
  - Neste projeto blablabla

2. Recursos

- Identificação e alocação dos recursos globais, como os recursos do sistema serão alocados, utilizados, compartilhados e liberados. Implicam modificações no diagrama de componentes.
  - Neste projeto blablabla
- Identificação da necessidade do uso de banco de dados. Implicam em modificações nos diagramas de atividades e de componentes.
  - Neste projeto blablabla
- Identificação da necessidade de sistemas de armazenamento de massa. Por exemplo: um *storage* em um sistema de cluster ou sistemas de backup.
  - Neste projeto blablabla

3. Controle

- Identificação e seleção da implementação de controle, seqüencial ou concorrente, baseado em procedimentos ou eventos. Implicam modificações no diagrama de execução.
  - Neste projeto blablabla
- Identificação das condições extremas e de prioridades.
  - Neste projeto blablabla
- Identificação da necessidade de otimização. Por exemplo: prefira sistemas com grande capacidade de memória; prefira vários hds pequenos a um grande.
  - Neste projeto blablabla
- Identificação e definição de *loops* de controle e das escalas de tempo.
  - Neste projeto blablabla
- Identificação de concorrências – quais algoritmos podem ser implementados usando processamento paralelo.
  - Neste projeto blablabla

4. Plataformas

   - Identificação das estruturas arquitetônicas comuns.
     - Neste projeto blablabla
   - Identificação de subsistemas relacionados à plataforma selecionada. Podem implicar em modificações no diagrama de pacotes e no diagrama de componentes.
     - Neste projeto blablabla
   - Identificação e definição das plataformas a serem suportadas: hardware, sistema operacional e linguagem de software.
     - Neste projeto blablabla
   - Seleção das bibliotecas externas a serem utilizadas.
     - Neste projeto blablabla
   - Seleção da biblioteca utilizada para montar a interface gráfica do software – GDI.
     - Neste projeto blablabla
   - Seleção do ambiente de desenvolvimento para montar a interface de desenvolvimento – IDE.
     - Neste projeto blablabla

5. Padrões de projeto

   - Normalmente os padrões de projeto são identificados e passam a fazer parte de uma biblioteca de padrões da empresa. Mas isto só ocorre após a realização de diversos projetos.
     - Neste projeto blablabla

1.2 Projeto orientado a objeto – POO

O projeto orientado a objeto é a etapa posterior ao projeto do sistema. Baseia-se na análise, mas considera as decisões do projeto do sistema. Acrescenta a análise desenvolvida e as características da plataforma escolhida (hardware, sistema operacional e linguagem de softwareção). Passa pelo maior detalhamento do funcionamento do software, acrescentando atributos e métodos que envolvem a solução de problemas específicos não identificados durante a análise.

Envolve a otimização da estrutura de dados e dos algoritmos, a minimização do tempo de execução, de memória e de custos. Existe um desvio de ênfase para os conceitos da plataforma selecionada.

Exemplo: na análise você define que existe um método para salvar um arquivo em disco, define um atributo nomeDoArquivo, mas não se preocupa com detalhes específicos da linguagem. Já no projeto, você inclui as bibliotecas necessárias para acesso ao disco, cria um objeto específico para acessar o disco, podendo, portanto, acrescentar novas classes àquelas desenvolvidas na análise.

Efeitos do projeto no modelo estrutural

- Adicionar nos diagramas de pacotes as bibliotecas e subsistemas selecionados no projeto do sistema (exemplo: a biblioteca gráfica selecionada).
  - Neste projeto blablabla
- Novas classes e associações oriundas das bibliotecas selecionadas e da linguagem escolhida devem ser acrescentadas ao modelo.
  - Neste projeto blablabla
- Estabelecer as dependências e restrições associadas à plataforma escolhida.
  - Neste projeto blablabla

Efeitos do projeto no modelo dinâmico

- Revisar os diagramas de seqüência e de comunicação considerando a plataforma escolhida.
  - Neste projeto blablabla
- Verificar a necessidade de se revisar, ampliar e adicionar novos diagramas de máquinas de estado e de atividades.
  - Neste projeto blablabla

Efeitos do projeto nos atributos

- Atributos novos podem ser adicionados a uma classe, como, por exemplo, atributos específicos de uma determinada linguagem de softwareção (acesso a disco, ponteiros, constantes e informações correlacionadas).
  - Neste projeto blablabla

Efeitos do projeto nos métodos

- Em função da plataforma escolhida, verifique as possíveis alterações nos métodos. O projeto do sistema costuma afetar os métodos de acesso aos diversos dispositivos (exemplo: hd, rede).
  - Neste projeto blablabla
- De maneira geral os métodos devem ser divididos em dois tipos: i) tomada de decisões, métodos políticos ou de controle; devem ser claros, legíveis, flexíveis e usam polimorfismo. ii) realização de processamentos, podem ser otimizados e em alguns casos o polimorfismo deve ser evitado.
  - Neste projeto blablabla
- Algoritmos complexos podem ser subdivididos. Verifique quais métodos podem ser otimizados. Pense em utilizar algoritmos prontos como os da STL (algoritmos genéricos).
  - Neste projeto blablabla
- Responda a pergunta: os métodos da classes estão dando resposta às responsabilidades da classe?
  - Neste projeto blablabla
- Revise os diagramas de classes, de seqüência e de máquina de estado.
  - Neste projeto blablabla

Efeitos do projeto nas heranças

- Reorganização das classes e dos métodos (criar métodos genéricos com parâmetros que nem sempre são necessários e englobam métodos existentes).
  - Neste projeto blablabla
- Abstração do comportamento comum (duas classes podem ter uma superclasse em comum).
  - Neste projeto blablabla
- Utilização de delegação para compartilhar a implementação (quando você cria uma herança irreal para reaproveitar código). Usar com cuidado.
  - Neste projeto blablabla

- Revise as heranças no diagrama de classes.
  - Neste projeto blablabla

Efeitos do projeto nas associações

- Deve-se definir na fase de projeto como as associações serão implementadas, se obedecerão um determinado padrão ou não.
  - Neste projeto blablabla
- Se existe uma relação de "muitos", pode-se implementar a associação com a utilização de um dicionário, que é um mapa das associações entre objetos. Assim, o objeto A acessa o dicionário fornecendo uma chave (um nome para o objeto que deseja acessar) e o dicionário retorna um valor (um ponteiro) para o objeto correto.
  - Neste projeto blablabla
- Evite percorrer várias associações para acessar dados de classes distantes. Pense em adicionar associações diretas.
  - Neste projeto blablabla

Efeitos do projeto nas otimizações

- Faça uma análise de aspectos relativos à otimização do sistema. Lembrando que a otimização deve ser desenvolvida por analistas/desenvolvedores experientes.
  - Neste projeto blablabla
- Identifique pontos a serem otimizados em que podem ser utilizados processos concorrentes.
  - Neste projeto blablabla
- Pense em incluir bibliotecas otimizadas.
- Se o acesso a determinados objetos (atributos/métodos) requer um caminho longo (exemplo: A->B->C->D.atributo), pense em incluir associações extras (exemplo: A-D.atributo).
  - Neste projeto blablabla
- Atributos auxiliares podem ser incluídos.
  - Neste projeto blablabla

- A ordem de execução pode ser alterada.
  - Neste projeto blablabla
- Revise as associações nos diagramas de classes.
  - Neste projeto blablabla

Depois de revisados os diagramas da análise você pode montar dois diagramas relacionados à infraestrutura do sistema. As dependências dos arquivos e bibliotecas podem ser descritos pelo diagrama de componentes, e as relações e dependências entre o sistema e o hardware podem ser ilustradas com o diagrama de implantação.

1.3 Diagrama de componentes

O diagrama de componentes mostra a forma como os componentes do software se relacionam, suas dependências. Inclui itens como: componentes, subsistemas, executáveis, nós, associações, dependências, generalizações, restrições e notas. Exemplos de componentes são bibliotecas estáticas, bibliotecas dinâmicas, dlls, componentes Java, executáveis, arquivos de disco, código-fonte.

Veja na Figura 1.1 um exemplo de diagrama de componentes. Dessa forma, um diagrama de componentes é uma ferramenta valiosa para o seu projeto de desenvolvimento de software para simulação de curvas IPR na engenharia de petróleo por várias razões:

- Visualiza a estrutura do sistema;
- Mostra as dependências entre os elementos;
- Organiza o código em módulos;
- Facilita a comunicação na equipe;
- Identifica áreas críticas do sistema;
- Serve como documentação.

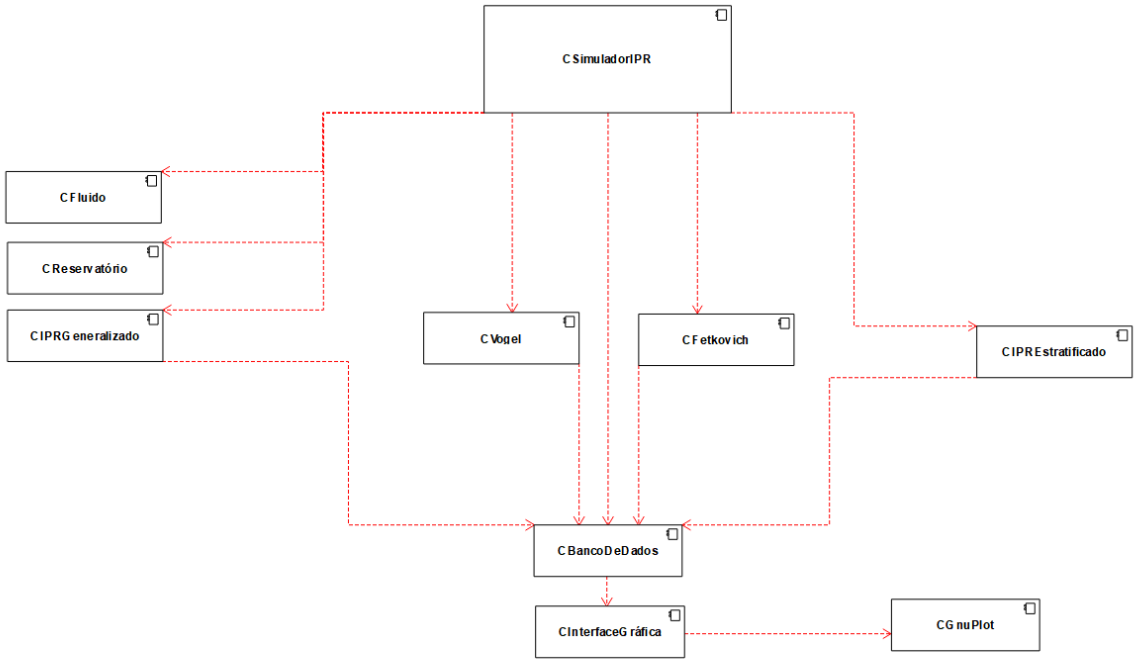

Figura 1.1: Diagrama de componentes

1.4 Diagrama de implantação

O diagrama de implantação é um diagrama de alto nível que inclui relações entre o sistema e o hardware e que se preocupa com os aspectos da arquitetura computacional escolhida. Seu enfoque é o hardware, a configuração dos nós em tempo de execução.

O diagrama de implantação deve incluir os elementos necessários para que o sistema seja colocado em funcionamento: computador, periféricos, processadores, dispositivos, nós, relacionamentos de dependência, associação, componentes, subsistemas, restrições e notas. O diagrama de implantação desempenha um papel crucial no projeto de engenharia de petróleo com simulações de curvas IPR para Poços Verticais. Ele se aplica de forma significativa ao ajudar a visualizar como os diversos componentes de software e hardware interagem em seu ambiente de execução. Neste contexto, o diagrama de implantação permite a:

- Visualização da Infraestrutura;
- Modelagem de Conexões;
- Planejamento de Recursos;
- Identificação de Pontos de Falha;
- Escalabilidade.

Veja na Figura 1.2 um exemplo de diagrama de implantação.

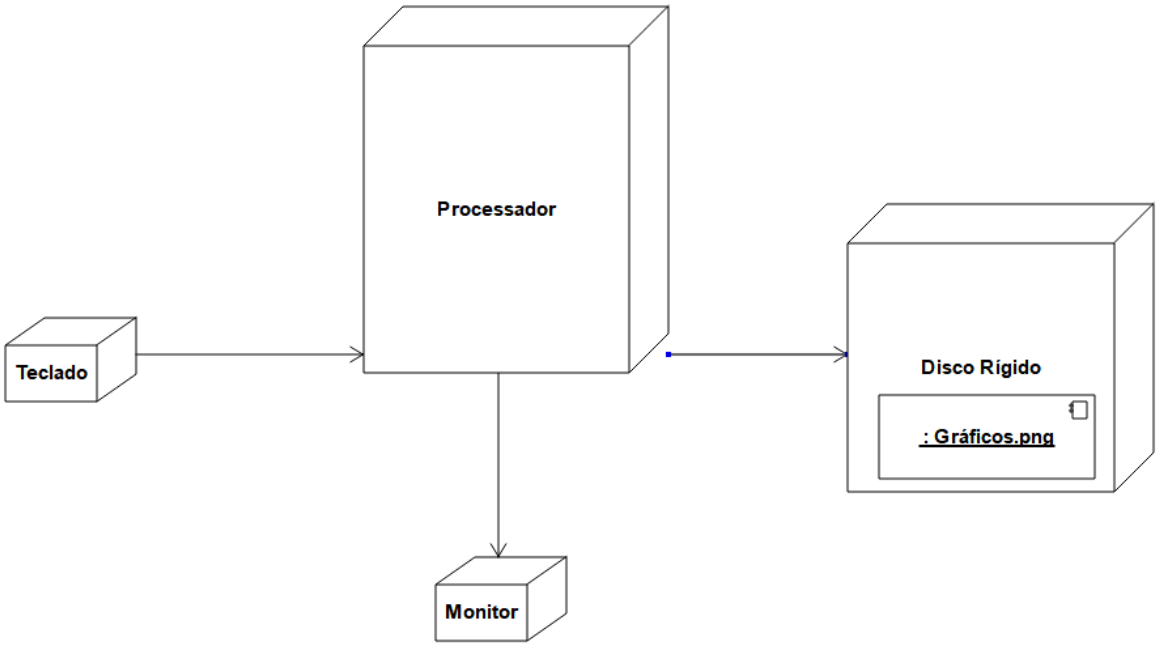

Figura 1.2: Diagrama de implantação

1.4.1 Lista de características <<features>>

No final do ciclo de concepção e análise chegamos a uma lista de características *<<features>>* que teremos de implementar.

Após a análises desenvolvidas e considerando o requisito de que este material deve ter um formato didático, chegamos a seguinte lista:

- v0.1
  - Lista de classes a serem implementadas
  - Testes
- v0.3
  - Lista de classes a serem implementadas
  - Testes

- v0.7
  - Lista de classes a serem implementadas
  - Testes

1.4.2 Tabela classificação sistema

A Tabela a seguir é utilizada para classificação do sistema desenvolvido. Deve ser preenchida na etapa de projeto e revisada no final, quando o software for entregue na sua versão final.

Licença:	[X] livre GPL-v3 [ ] proprietária
Engenharia de software:	[ ] tradicional [X] ágil [ ] outras
Paradigma de programação:	[ ] estruturada [X] orientado a objeto - POO [ ] funcional
Modelagem UML:	[X] básica [X] intermediária [ ] avançada
Algoritmos:	[X] alto nível [X] baixo nível
	implementação: [ ] recursivo ou [X] iterativo; [X] determinístico ou [ ] não-determinístico; [ ] exato ou [X] aproximado
	concorrências: [X] serial [X] concorrente [X] paralelo
	paradigma: [X] dividir para conquistar [ ] programação linear [ ] transformação/ redução [ ] busca e enumeração [ ] heurístico e probabilístico [ ] baseados em pilhas
Software:	[ ] de base [X] aplicados [ ] de cunho geral [X] específicos para determinada área [X] educativo [X] científico
	instruções: [X] alto nível [ ] baixo nível
	otimização: [X] serial não otimizado [X] serial otimizado [X] concorrente [X] paralelo [ ] vetorial
	interface do usuário: [ ] kernel numérico [ ] linha de comando [ ] modo texto [X] híbrida (texto e saídas gráficas) [ ] modo gráfico (ex: Qt) [ ] navegador
Recursos de C++:	[X] C++ básico (FCC): variáveis padrões da linguagem, estruturas de controle e repetição, estruturas de dados, struct, classes(objetos, atributos, métodos), funções; entrada e saída de dados (*streams*), funções de cmath

	[X] C++ intermediário: funções lambda. Ponteiros, referências, herança, herança múltipla, polimorfismo, sobrecarga de funções e de operadores, tipos genéricos (templates), *smarth pointers*. Diretrizes de pré-processador, classes de armazenamento e modificadores de acesso. Estruturas de dados: enum, uniões. Bibliotecas: entrada e saída acesso com arquivos de disco, redirecionamento. Bibliotecas: *filesystem*
	[X] C++ intermediário 2: A biblioteca de gabaritos de C++ (a STL), containers, iteradores, objetos funções e funções genéricas. Noções de processamento paralelo (múltiplas threads, uso de *thread, join* e *mutex*). Bibliotecas: *random, threads*
	[ ] C++ avançado: Conversão de tipos do usuário, especializações de templates, excessões. Cluster de computadores, processamento paralelo e concorrente, múltiplos processos (pipes, memória compartilhada, sinais). Bibliotecas: *expressões regulares, múltiplos processos*
Bibliotecas de C++:	[X] Entrada e saída de dados (*streams*) [X] cmath [X] *filesystem* [ ] *random* [X] *threads* [ ] *expressões regulares* [ ] *múltiplos processos*
Bibliotecas externas:	[X] `CGnuplot` [X] `QCustomPlot` [ ] `Qt diálogos` [ ] QT Janelas/menus/BT_____
Ferramentas auxiliares:	Montador: [X] make [] cmake [X] qmake
IDE:	[X] Editor simples: kate/gedit/emacs [ ] kdevelop [ ] QT-Creator [ ] _____
SCV:	[ ] cvs [ ] svn [X] git
Disciplinas correlacionadas	[ ] estatística [ ] cálculo numérico [ ] modelamento numérico [X] análise e processamento de imagens