Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Disciplina: Qualidade e Teste de Software

Profª Drª Marcia Cassitas Hino

2º Semestre/2023

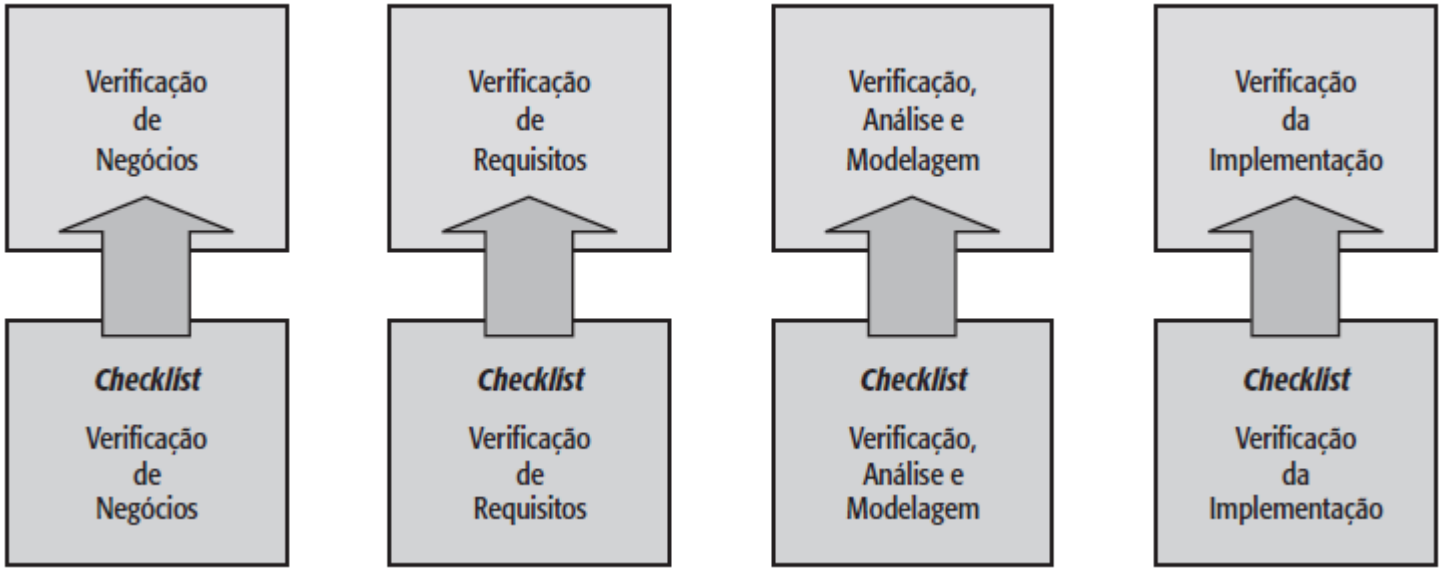
Universidade Tuiuti do Paraná

Checklist Verificação

- *Checklist* é um poderoso instrumento a ser aplicado nas revisões de documentos (VER) ao criar uma abordagem alinhada de como avaliar a qualidade de um documento.

- Esse direcionamento é realizado através de “guias” que orientam as revisões dos documentos, reduzindo o grau de subjetividade em relação ao que deve ser avaliado ou não.

- Orienta o profissional a investigar diversos aspectos em relação ao documento ou atividade.



Checklist *aplicado às diversas fases dos testes de verificação*

- As etapas de um projeto de software produzem documentos sem que estes sofram uma avaliação direta, o que impede que determinados problemas sejam avaliados por diferentes ângulos, aumentando os riscos das decisões formalizadas serem inconsistentes na sua prática.
- É nesse ponto que o processo de verificação ganha força e torna-se um importante mecanismo de prevenção de defeitos, pois atua diretamente na fonte das decisões e avalia se determinadas atividades críticas do processo de software estão sendo realizadas.

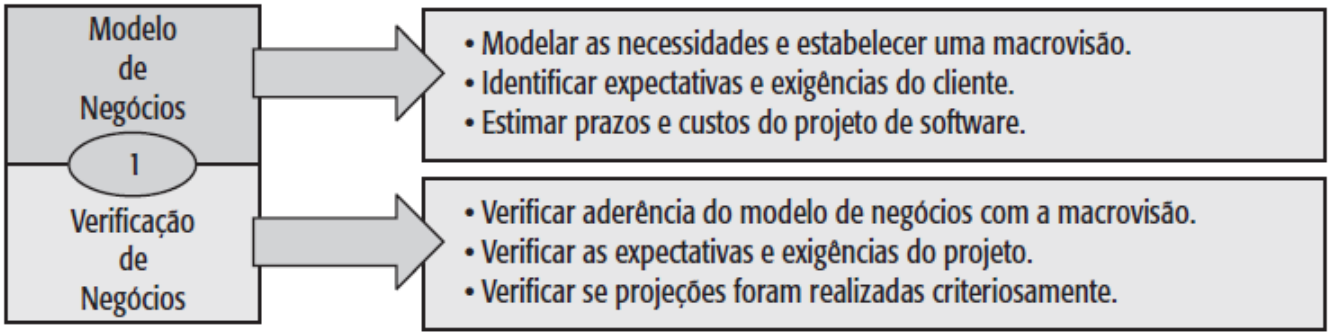
Verificação de Negócios

Principais produtos:

- Modelo de Negócios
- Análise de Riscos
- Árvore de Decisão
- Estudo de Viabilidade

Principais atividades:

- Revisar contexto do mercado e necessidades do cliente
- Revisar riscos do projeto
- Auditar alternativas de execução do projeto
- Revisar estudo de viabilidade do Projeto



- É nessa fase que se estabelece o primeiro contato com as necessidades, expectativas e exigências do cliente.
- Atender a essas expectativas é crucial para o sucesso de um projeto de software.
- Gera-se uma visão geral do projeto a qual possibilita dimensionar recursos humanos, físicos e financeiros necessários para a construção do software e para sua adequada operacionalização.

Pontos de Verificação

- Garantir que os diversos documentos produzidos tenham total aderência às necessidades apontadas pelos clientes.

- Cuidado para não criar um "canhão para matar uma única mosca".

Universidade Tuiuti do Paraná

Pontos críticos:

- Avaliar se todas as necessidades, metas e exigências foram listadas.
- Verificar se a modelagem de negócios cobre todas as necessidades.
- Conferir se as projeções realizadas são baseadas em métricas e indicadores confiáveis.
- Avaliar a existência de alternativas a essa solução.
- Avaliar o retorno sobre o investimento em cada alternativa existente (ROI).
- Validar as opções de investimento (árvore de decisão).

Exemplos

Checklist do Modelo de Negócios		
Levantamento das Necessidades do Cliente		
– Todas as necessidades foram devidamente registradas.	☐ Sim	☐ Não
– Cada necessidade apontada possui uma descrição.	☐ Sim	☐ Não
Definição das Características do Software		
– Cada característica atende ao menos a uma necessidade identificada.	☐ Sim	☐ Não
– Cada característica possui uma descrição clara.	☐ Sim	☐ Não
– Cada característica possui exemplos que auxiliam seu entendimento.	☐ Sim	☐ Não
– Existe uma rastreabilidade entre características e necessidades.	☐ Sim	☐ Não

Checklist da Proposta de Projeto de Software		
Definição dos Objetivos do Projeto		
– Todos os objetivos foram apontados e claramente descritos.	☐ Sim	☐ Não
– Todos os objetivos podem ser quantificáveis.	☐ Sim	☐ Não
– Todos os objetivos possuem data-limite para ocorrer.	☐ Sim	☐ Não
– Existe rastreabilidade entre objetivos e necessidades.	☐ Sim	☐ Não
Definição dos Riscos		
– Todos os riscos foram identificados e adequadamente descritos.	☐ Sim	☐ Não
– Existe um plano de ação para cada risco definido.	☐ Sim	☐ Não
– Foram definidos "impacto" e "probabilidade" para cada risco apontado.	☐ Sim	☐ Não

Critérios de Finalização

- Trata de uma fase extremamente crítica do projeto de software
- É recomendável estabelecer um critério de finalização apoiado no aceite de toda a documentação
- Sugestões:
  - Modelagem de negócios assinada pelas diretorias e gerências envolvidas.
  - Proposta de projeto de software assinada pelas diretorias e gerências envolvidas.

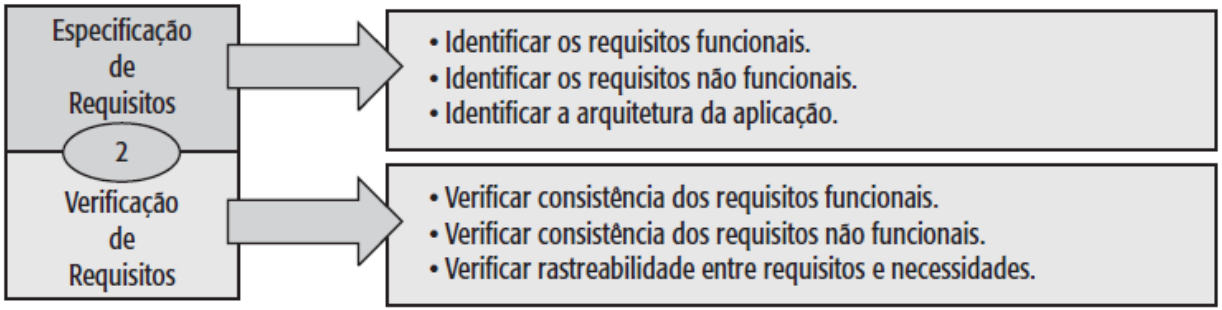
Verificação de Requisitos

Principais produtos:

- Especificação dos Requisitos
- Rastreabilidade

Principais atividades:

- Revisar especificação de requisitos funcionais
- Revisar especificação de requisitos não funcionais
- Revisar priorização de requisitos
- Auditar rastreabilidade de requisitos



- A missão é detalhar todos os aspectos funcionais e não funcionais relativos, estabelecendo um conjunto de especificações de negócio com o máximo detalhamento.
- Refina-se a expectativa do cliente em relação ao produto, uma vez que o nível de detalhamento possibilita a descoberta de aspectos que não tinham sido previstos.

Pontos de Verificação

- Concentrar-se na identificação de todos os requisitos funcionais e não funcionais de um software.
- Cada requisito deve ser claro o suficiente para que não produza uma interpretação errada.
- Investigar os requisitos e garantir sua adequada definição.

Exemplos de pontos críticos:

- **Completo:** Todos os requisitos devem estar documentados.
- **Claro:** Cada requisito deve expressar seu propósito em relação ao projeto.
- **Simples:** Cada requisito ter sua essência com uma simples definição.
- **Preciso:** Cada requisito deve ser exato, não dar margens a dúvidas.
- **Consistente:** Cada requisito não deve possuir conflitos com outros requisitos.
- **Relevante:** Cada requisito deve pertencer ao escopo do projeto.
- **Testável:** Cada requisito deverá fornecer informações que possibilitem quantificar se um determinado item foi adequadamente implementado.
- **Factível:** Cada requisito deve ser viável em sua implementação, avaliando as condições financeiras, humanas e tecnológicas disponíveis no projeto.

Exemplos

Checklist de Requisitos		
Diagrama de Casos de Uso		
– Existe um modelo de casos de uso para cada subsistema identificado.	☐ Sim	☐ Não
– Todos os casos de usos estão adequadamente descritos.	☐ Sim	☐ Não
– Todos os atores estão adequadamente representados.	☐ Sim	☐ Não
Levantamento de Requisitos		
– Cada caso de uso representa um requisito funcional.	☐ Sim	☐ Não
– Existe rastreabilidade entre requisitos identificados e necessidades.	☐ Sim	☐ Não
– Requisitos foram avaliados por importância, volatilidade e criticidade.	☐ Sim	☐ Não

Checklist de Requisitos		
Especificações Funcionais		
– Cada requisito funcional possui uma especificação detalhada.	☐ Sim	☐ Não
– As especificações contemplam os fluxos básicos, alternativos e exceção.	☐ Sim	☐ Não
– As especificações contemplam pré-requisitos e pós-condições.	☐ Sim	☐ Não
Especificações Não Funcionais		
– Todas as categorias de requisitos não funcionais foram levantadas.	☐ Sim	☐ Não
– Cada requisito não funcional possui uma especificação detalhada.	☐ Sim	☐ Não
– Todas as dependências dos componentes foram estabelecidas.	☐ Sim	☐ Não

Critérios de Finalização

- O critério de finalização deve garantir que todos os requisitos foram identificados.
- Os requisitos devem refletir as características funcionais e não funcionais que o cliente espera receber.
- Sugestões:
  - Especificações funcionais criadas e revisadas.
  - Especificações não funcionais criadas e revisadas.
  - Rastreabilidade entre requisitos e entre necessidades.

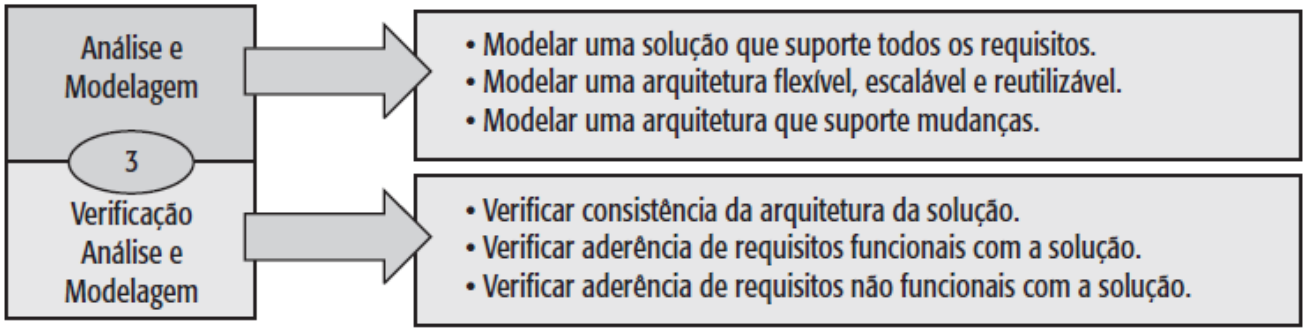
Verificação, Análise e Modelagem

Principais produtos:

- Arquitetura da aplicação
- Modelos estáticos
- Modelos dinâmicos
- Modelos de distribuição

Principais atividades:

- Revisar arquitetura da aplicação
- Revisar o modelo estático do projeto de software
- Revisar o modelo dinâmico do projeto de software
- Revisar nível de componentização
- Revisar nível de reutilização



- Nessa etapa o objetivo é definir uma solução tecnológica que suporte os requisitos do cliente, e requisitos de qualidade.
- Muitas vezes, as soluções modeladas nem sempre são adequadas ao longo prazo, evidenciando falhas no processo de definição da arquitetura.
- Validar determinados aspectos críticos de uma solução, avaliando a existência de mecanismos que suportem um alto nível de parametrização, possibilitando assimilar mudanças nos negócios e de tecnologias.

Pontos de Verificação

- Avaliar aderência da solução tecnológica aos requisitos funcionais e não funcionais.
- Avaliar a modelagem como um todo.
- Avaliar quanto a solução consegue absorver as modificações que poderão ocorrer ao longo do tempo.

Pontos críticos:

- Avaliar se os requisitos funcionais e não funcionais são atendidos pela solução.
- Avaliar se a arquitetura suporta crescimento e segurança.
- Avaliar se a arquitetura suporta futuras mudanças de negócios e de tecnologia.
- Avaliar se a arquitetura pode ser operacionalizada em vários ambientes.
- Avaliar as restrições e problemas conhecidos da arquitetura a ser adotada.

Exemplos

Checklist do Diagramas UML		
Diagramas de Classes		
– Todas as classes possuem nome e descrição adequados.	☐ Sim	☐ Não
– Todos os atributos da classe possuem nome e descrição adequados.	☐ Sim	☐ Não
– Todos os serviços da classe possuem nome e descrição adequados.	☐ Sim	☐ Não
Diagrama de Estado		
– Todas as transições de estado possuem um serviço ou evento associado.	☐ Sim	☐ Não
– Todos os estados possuem nome e descrição adequados.	☐ Sim	☐ Não
– Todas as transições de estado refletem o real ciclo de vida da classe.	☐ Sim	☐ Não
Diagramas de Componentes		
– Os *packages* agrupam componentes com as mesmas características.	☐ Sim	☐ Não
– Cada componente agrupa classes de única camada: *user*, *business*, *data*.	☐ Sim	☐ Não
– Todas as dependências dos componentes foram estabelecidas.	☐ Sim	☐ Não

Checklist da Arquitetura		
Suportar Mudanças nos Negócios		
– Existem parametrizações que modificam a funcionalidade da aplicação.	☐ Sim	☐ Não
Suportar Mudanças Tecnológicas		
– O software possui independência do banco de dados.	☐ Sim	☐ Não
– O software possui independência do sistema operacional.	☐ Sim	☐ Não
– O software possui independência de *browsers*.	☐ Sim	☐ Não

Critérios de Finalização

- Deve garantir que a solução tecnológica que foi modelada atende adequadamente a todos os requisitos dos clientes, além de incorporar características que deixam a arquitetura flexível e aderente a futuras mudanças.
- Sugestões:
  - Diagramas estáticos (classes e objetos) criados e revisados.
  - Diagramas dinâmicos (estados, sequência e atividades) criados e revisados.
  - Diagramas de distribuição (componentes e implantação) criados e revisados.

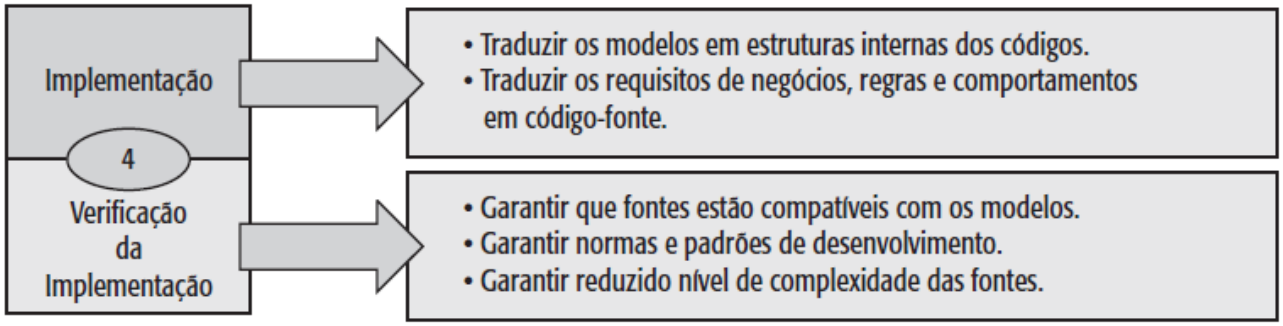
Verificação de Implementação

Principais produtos:

- Código-fonte
- Componentes
- Manual do Usuário

Principais atividades:

- Revisar o código-fonte
- Avaliar complexidade do código-fonte
- Auditar rastreabilidade entre componentes
- Revisar manual do usuário



- Essa fase encerra o ciclo de verificação de testes.
- A maioria das organizações começa seu ciclo de verificação nessa fase.
- Algumas empresas de software realizam um processo formal de verificação do código produzido, onde um programador revisa o código de outro.

Pontos de Verificação

- Garantir a qualidade do código-fonte gerado pela equipe de desenvolvimento.
- Verificar o uso de “regras da boa programação”, que podem contemplar diversas normas e padrões corporativos.

Pontos críticos:

- Comparar os modelos de arquitetura com os códigos-fontes.
- Utilizar ferramenta de análise estática para avaliar a complexidade dos fontes.
- Avaliar as mensagens apresentadas ao usuário final.
- Inspecionar a existência de rotinas de tratamentos de erros nos processos críticos.
- Inspecionar se o volume de comentários é suficiente.
- Inspecionar a legibilidade do código.
- Inspecionar o padrão de nomenclaturas existentes.

Exemplos

Checklist do Banco de Dados		
Comparação do Modelo de Dados com o Banco de Dados		
– Todas as tabelas do modelo de dados foram implementadas.	☐ Sim	☐ Não
– Todos os campos de cada tabela foram implementados.	☐ Sim	☐ Não
– Todos os índices de cada tabela foram implementados.	☐ Sim	☐ Não
– Todos os *stored procedures* de cada tabela foram implementados.	☐ Sim	☐ Não
– Todas as visões do modelo de dados foram implementadas.	☐ Sim	☐ Não
– Todos os campos de cada visão foram implementados.	☐ Sim	☐ Não

Checklist do Código-fonte		
Comparação do Modelo de Arquitetura do Software com o Código-fonte		
– Todas as classes do modelo foram implementadas.	☐ Sim	☐ Não
– Todos os métodos de cada classe foram implementados.	☐ Sim	☐ Não
– Todos os atributos de cada classe foram implementados.	☐ Sim	☐ Não
Mensagens Apresentadas ao Usuário Final		
– Nenhuma mensagem apresenta erros gramaticais.	☐ Sim	☐ Não
– Todas as mensagens são claras e bem objetivas.	☐ Sim	☐ Não
– Todas as mensagens apresentam ícones adequados ao contexto.	☐ Sim	☐ Não
Legibilidade do Código		
– Todas as estruturas estão adequadamente identadas.	☐ Sim	☐ Não
– Não existem linhas agrupadas com IF, SELECT, FOR NEXT e FOR EACH.	☐ Sim	☐ Não
– Tratamentos de erros e desvios sempre estão no final das rotinas.	☐ Sim	☐ Não
– Todas as declarações de variáveis e constantes estão no início da rotina.	☐ Sim	☐ Não
– Não existem vários comandos em uma única linha.	☐ Sim	☐ Não
Volume de Comentários		
– Todas as rotinas possuem descrição sobre seu comportamento.	☐ Sim	☐ Não
– Todos os desvios de rotinas possuem um comentário.	☐ Sim	☐ Não

Critérios de Finalização

- Uso de ferramenta para realizar, de forma automática, inspeções nos códigos gerados analisando o uso de boas práticas de programação e uma análise de complexidade do código-fonte.

- O resultado dessa análise é uma lista de não-conformidades que deverá ser analisada pela equipe de desenvolvimento até que os critérios de finalização sejam alcançados.

Critérios analisados durante a inspeção do código	Tolerância a Erros por Severidade		
	Alta	**Média**	**Baixa**
Banco de Dados	*Nenhuma*	*Nenhuma*	*Nenhuma*
Internacionalização	*Nenhuma*	*Nenhuma*	*Nenhuma*
Lógica	*Nenhuma*	*Nenhuma*	*Nenhuma*
Performance	*Nenhuma*	**10 erros**	**10 erros**
Portabilidade	*Nenhuma*	*Nenhuma*	*Nenhuma*
Usabilidade	**5 erros**	**10 erros**	**20 erros**
APIs do Windows	*Nenhuma*	*Nenhuma*	*Nenhuma*

Complexidade Ciclomática	Avaliação da Complexidade	Esforço de Manutenção e Teste	Probabilidade de Inserção de Erros	Percentual Máximo Permitido
< 5	Simples	Baixo esforço	1%	100%
5-10	Moderada	Médio esforço	5%	20%
11-20	Difícil	Grande esforço	10%	5%
21-50	Muito difícil	Muito complexo	30%	Não permitido
> 50	Impossível testar	Refazer	–	Não permitido

Planilhas de apoio:

01-PlanoTestes
02-TesteUnitario-Integracao
03-ReferenciaCTI
04-CasosTeste
05-RelatorioTestes
06-VER
07-PPQA

Obrigada!