

Requisitos – Aula 23

Professores: Milene Serrano e Maurício Serrano

Agenda

- › Considerações Iniciais
- › Análise
 - Identificação de Partes
 - Verificação
 - Validação
 - Teste
- › Em resumo...
- › Considerações Finais

Considerações Iniciais

Análise

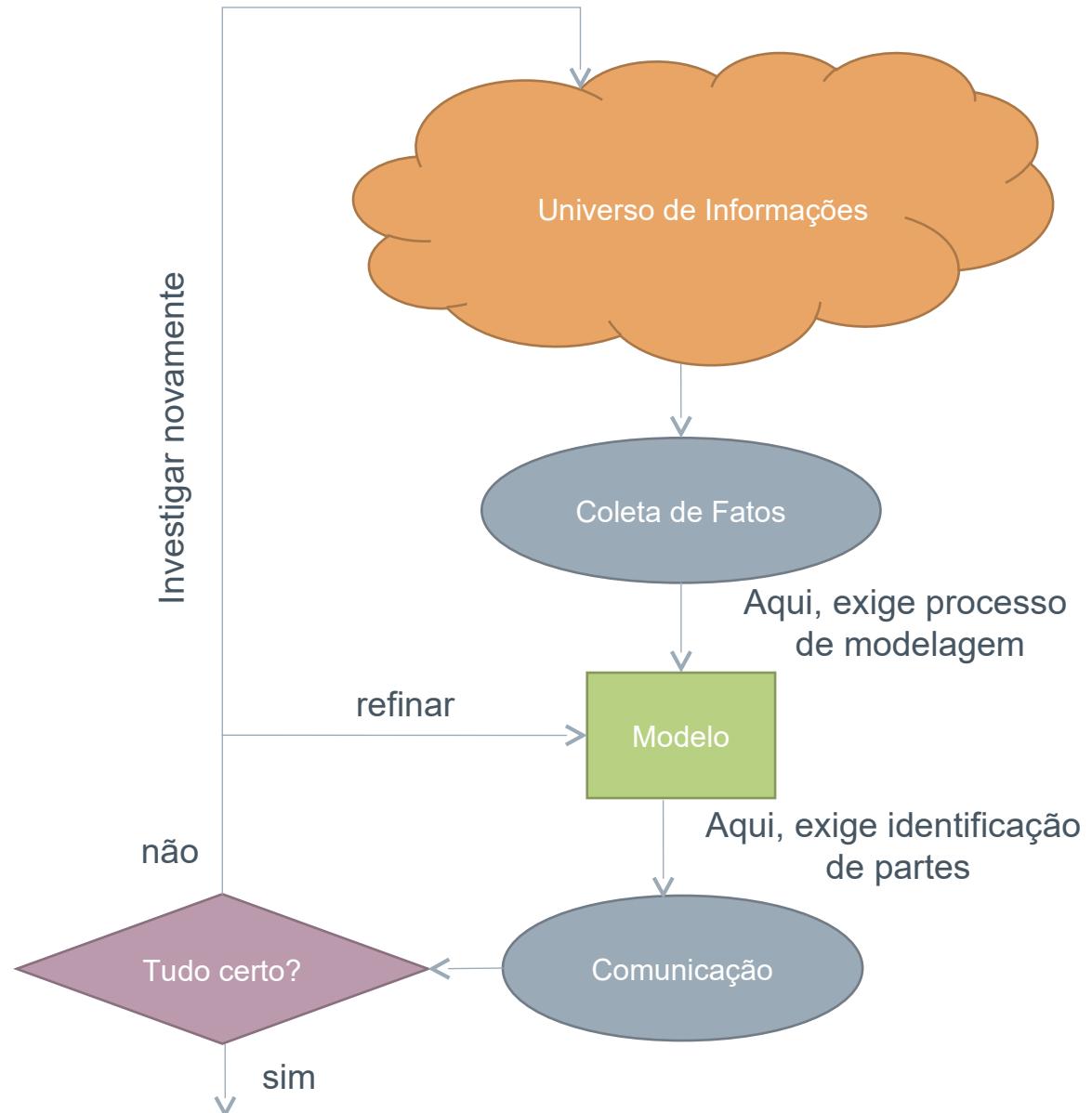
A atividade de Análise de Requisitos atua fundamentalmente em verificação e validação. Só faz sentido analisar algo, quando temos uma representação.



Loop da Análise

Para que tudo seja bem sucedido, reparem que é fundamental uma boa coleta de fatos e uma boa comunicação.
Isso nos remete a um bom processo de elicitação.

Observem o *loop* da análise. A identificação de partes ocorre entre o MODELO e a COMUNICAÇÃO.
Se tudo estiver bem (SEM PROBLEMAS), o modelo nos server.
Caso contrário, teremos que retornar ao Universo de Informação ou, então, refinar o modelo.



Em coleta de fatos...

- 
- **Leitura de Documentos** – permite ao Engenheiro de Requisitos ter acesso ao que é documentado em linguagem natural, escrito; conhecendo o Universo de Informações mesmo antes de aplicar outras técnicas;
 - **Observação** – em posição passiva no Universo de Informações, o Engenheiro de Requisitos observa o ambiente, no qual o software irá atuar;
 - **Entrevistas** – muito usual na coleta de fatos. Demanda que o Engenheiro de Requisitos disponha de conhecimento sobre o problema, visando elaborar um bom conjunto de perguntas;
 - **Questionários** – quando necessário apurar um grande número de clientes;
 - **Análise de Protocolos** – analisando o trabalho de alguém através de verbalizações desse alguém;

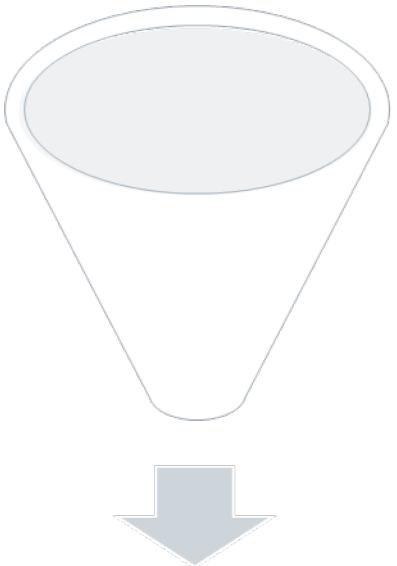
Em coleta de fatos...

- **Participação Ativa dos Atores** – aqui, os clientes são convidados a participar do processo de modelagem, criticando os modelos, primeiramente modelados por Engenheiros de Requisitos, e ajudando a evoluir esses modelos.
- **Enfoque Antropológico** – aqui, o Engenheiro de Requisitos é quem se integra ao Universo de Informações para obter um conhecimento mais amplo possível do problema.
- **Reuniões** – existem várias formas de estruturar/organizar uma reunião. Formatos que proporcionem cooperação entre os participantes são mais desejados.
- **Reutilização** – na definição dos requisitos de um software, é sempre desejado que sejam reutilizados fatos já coletados anteriormente, com a construção de produtos de software já desenvolvidos.
- **Recuperação do Desenho de Software** – estratégia baseada no estudo de produtos de software disponíveis na própria organização. Como, muitas vezes, a documentação de legados é pouca ou ausente, tem-se a necessidade de se recuperar essa documentação. Trata-se de um processo chamado Engenharia Reversa, parte-se de um nível mais baixo de abstração, recuperando artefatos de níveis mais altos de abstração.

Em comunicação...

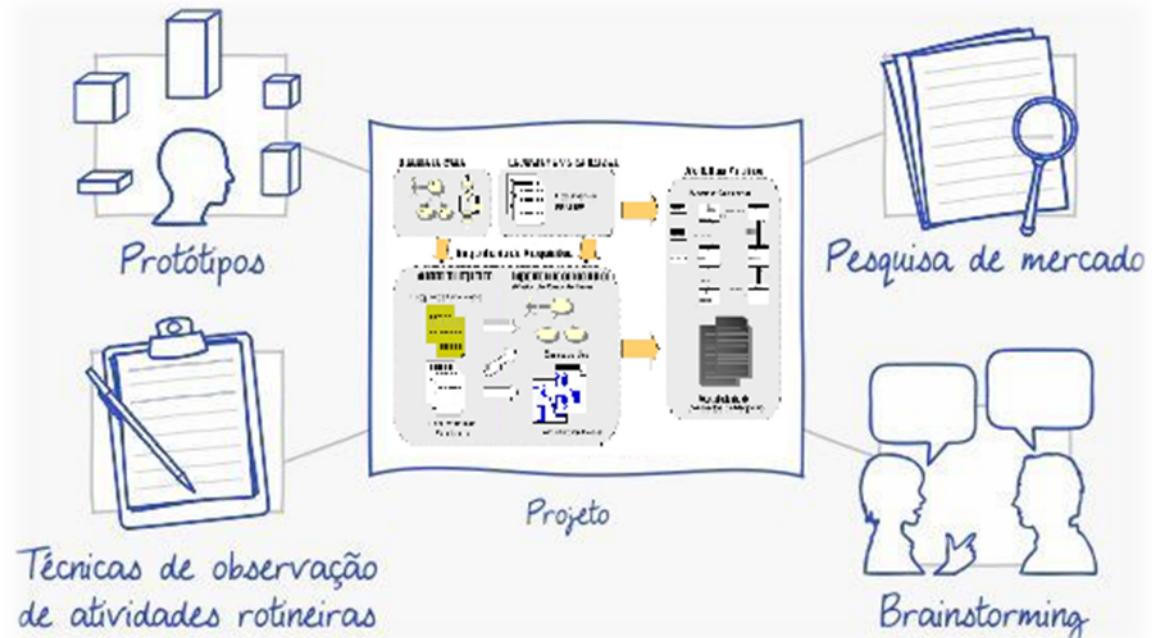
- 
- **Apresentação** – procurar, na apresentação de informações, ser o mais claro e compreensível possível, facilitando a comunicação entre clientes e engenheiros; engenheiros e clientes; apenas os clientes, e apenas os engenheiros. Apresentar o contexto é de fundamental relevância.
 - **Linguagem** – entender a linguagem de seus clientes é papel do Engenheiro de Requisitos. O conhecimento da linguagem do cliente facilita a comunicação.
 - **Nível de Abstração** – usar um nível de abstração mais alto, quando os clientes não possuem conhecimento técnico. Adequar esse nível também é papel do Engenheiro de Requisitos. O cliente deve se sentir confortável com o nível de abstração usado na comunicação.
 - **Retroalimentação** – uma maneira de melhorar a passagem da informação do emissor ao receptor consiste em estimular o receptor recolocar o que foi comunicado até que o emissor responda positivamente à recolocação.

Em resumo...



Sabemos que:

- Coleta de Fatos é algo fundamental para a atividade de elicitação. Essa atividade já foi tratada no começo do curso.
- Comunicação também foi algo discutido desde o começo do curso, influenciando as atividades de pré-rastreabilidade, elicitação bem como modelagem.



O que se preocupar no momento?

Dessa forma, temos três tarefas fundamentais a serem conhecidas com maior profundidade:

Identificação de Partes

Verificação

Validação



Fonte da Figura: <<internet, em material aberto>>

Análise de Requisitos

Identificação de Partes

Identificação de Partes



Primeiramente, para analisar, é necessário saber como está organizado e armazenado o modelo objeto de nossa análise.

Identificar partes do modelo é relevante nesse momento.

Essa tarefa tem estreita relação com a modelagem e a elicitação, nas quais se torna clara a identificação das fontes de informação.

Analizar sistemas complexos inteiros não é uma tarefa sensata. Portanto, particionar essa análise torna-se adequado.



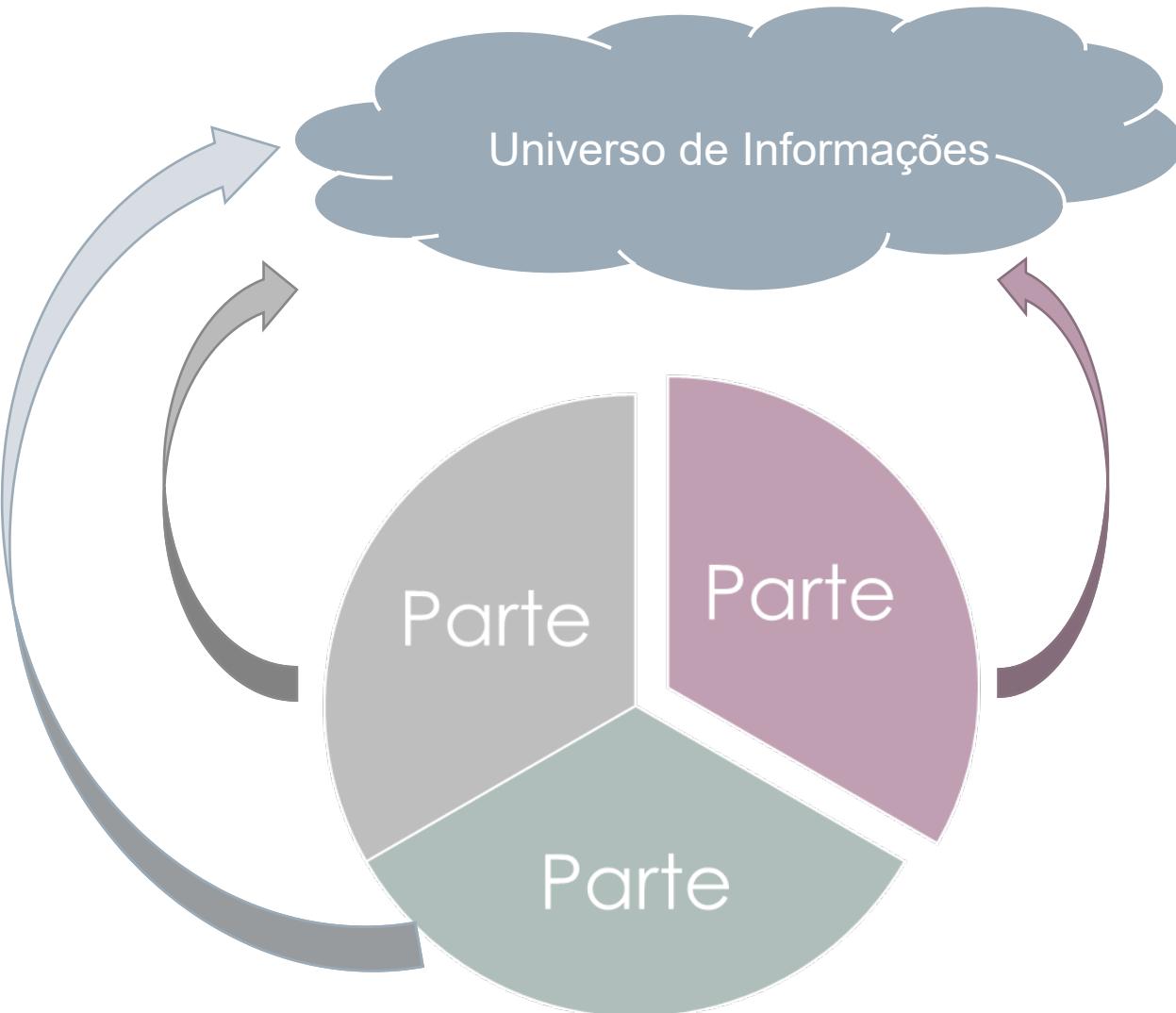
Rastreabilidade

Mas, não basta identificar as partes. É necessário conhecer os elos de ligação entre essas partes e as fontes de informação do Universo de Informações.

Aqui, temos um aspecto que ainda iremos conhecer no curso: um olhar mais profundo em Rastreabilidade de Requisitos. Falamos um pouco sobre pré-rastreabilidade. Mas, ainda precisaremos conhecer pós-rastreabilidade...

Uma vez identificadas as partes e conhecidos os elos de ligação dessas com as fontes de informação, podem ser estabelecidas as políticas de análise, i.e. quais as partes que terão prioridade, que tipo de varredura será utilizada, ou mesmo se será utilizado enfoque estatístico. A área de testes coloca em evidência esses aspectos. Umas das máximas utilizadas na área de testes é que **BOA PARTE DOS PROBLEMAS ENCONTRA-SE EM UMA PEQUENA PARTE DO SISTEMA.**

Pré-rastreabilidade



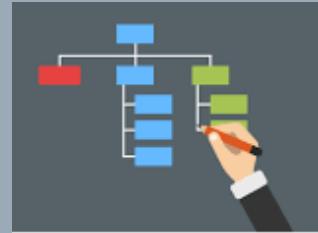
Manter o rastro entre as partes identificadas e as fontes de informação é de fundamental relevância!

PRÉ-RASTREABILIDADE

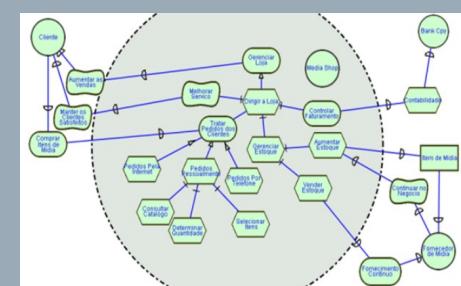
Fazendo um paralelo com i*

é uma notação muito interessante, pois ela permite análise de requisitos.

Mais especificamente, com propagação orientada aos impactos e demais relacionamentos especificados ANDs e ORs.



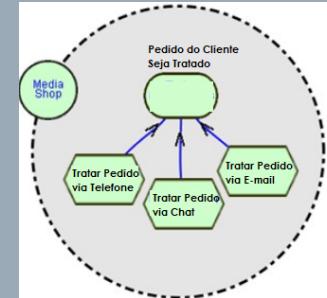
Modela



forma mais geral



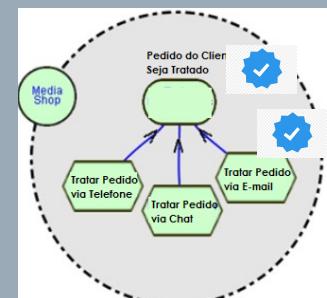
Particiona



Preocupa-se com **means-end**, pensando em **VARIABILIDADES...**



Analisa



Propaga-se com base nas relações especificadas...

Análise de Requisitos

Verificação

Verificação



Se optarmos por uma definição mais rígida, verificação entre modelos NÃO demanda intervenção humana, sendo feita via software.

No entanto, vamos considerar verificação como uma análise de modelos SEM que haja direta comparação com o Universo de Informações, sendo essa análise uma tarefa desempenhada tanto por seres humanos como por software, segundo regras bem definidas.

EM RESUMO: tem algo errado no nosso modelo em termos de notação, processo, procedimentos? O modelo está de acordo com o que se espera dele? Portanto, **SÓ INTERESSA AOS ENGENHEIROS DE SOFTWARE! O CLIENTE NÃO PRECISA SE ENVOLVER!**

Verificação



Existem várias estratégias de verificação de software.

Se considerarmos que algumas dessas estratégias estão mais associadas à elicitação de requisitos, teremos um conjunto, do qual sobressaem as seguintes estratégias:

- Inspeções;
- Uso de estratégias formais, e
- Reutilização de domínios.

Análise de Requisitos

Verificação com **INSPEÇÕES**



Inspeções

Inspeções podem ser aplicados na definição de requisitos para a verificação em documentos de requisitos, quer eles sejam produzidos por clientes, e sem uma estrutura própria, ou produzidos por Engenheiros de Software/Requisitos, sendo, nesse caso, uma lista de requisitos.

Trata-se de um **MÉTODO GERENCIAL DE REUNIÕES**, as quais objetivam o descobrimento de **DEFEITOS** em documentos.

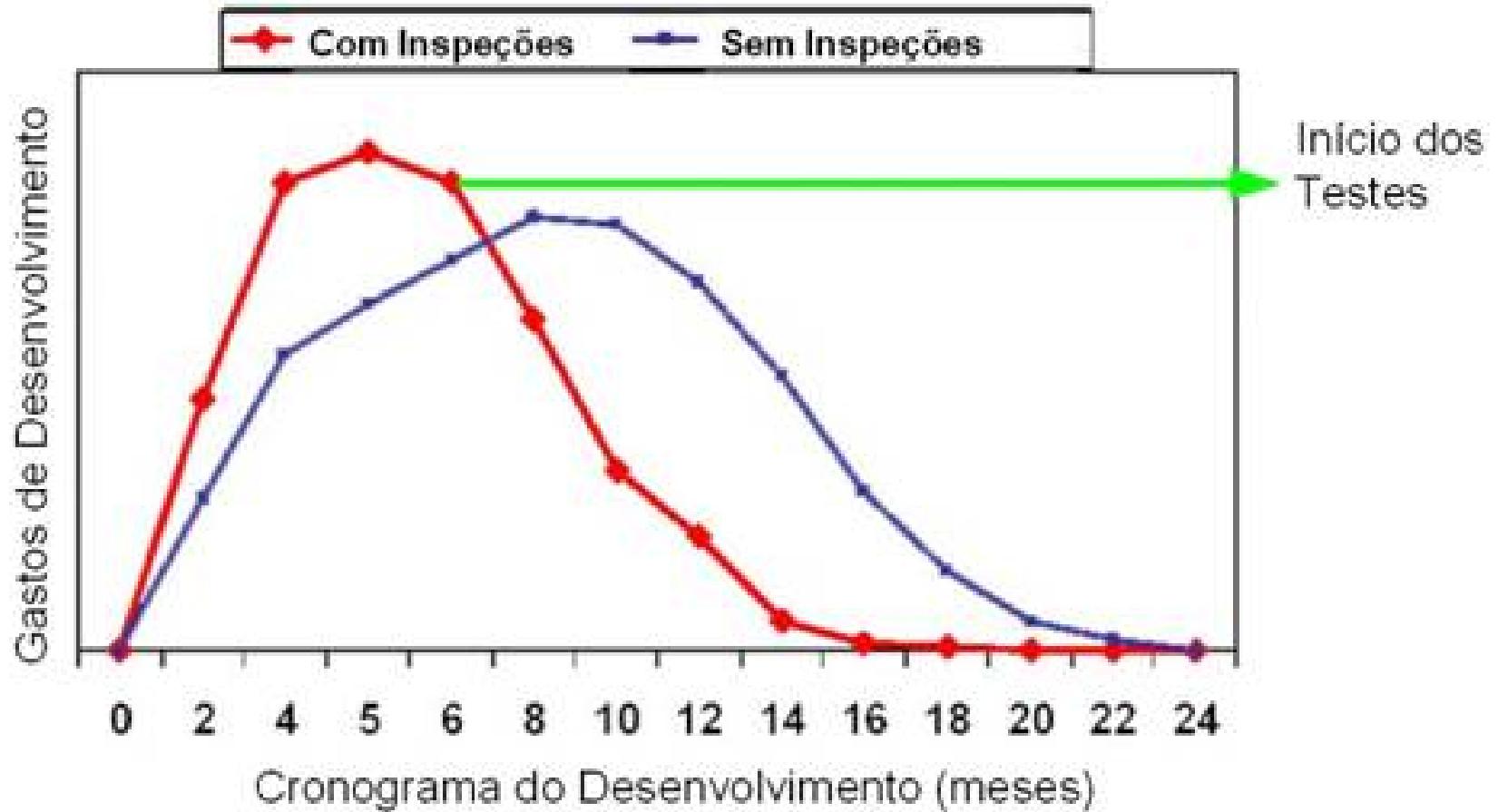
Há definições claras sobre os papéis que cada membro da equipe tem que desempenhar bem como quais resultados devem ser produzidos.

A produtividade do uso de Inspeções na revisão de desenhos e de código, segundo relatos da indústria, tem sido excelente. Portanto, consegue-se, efetivamente, descobrir defeitos antes do sistema ser testado.

Inspeções

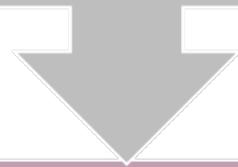
* Dessa data para frente, essa tendência é mais significativa ainda...

Adaptado de (WHEELER et al., 1996)



Inspeções

O objetivo básico das Inspeções é verificar se o modelo de requisitos está de acordo com a notação do modelo e com o que se espera desse modelo.



Segundo Fagan, tem-se um método de inspeção que estabelece um processo com seis passos principais. A saber:

Planejamento;

Visão Geral;

Preparação;

Inspeção;

Correção, e

Acompanhamento.



Análise de Requisitos

Verificação com **USO DE ESTRATÉGIAS FORMAIS**

Uso de Estratégias Formais



No uso de formalismos...

- Aqui, o Engenheiro de Software faz o papel de um provador de teoremas. Portanto, a verificação ocorre dada a possibilidade de se identificar inconsistências.

FOCO:

- $\Delta = \text{inconsistências (fatos, informação)}$

Opa! Interessante uma comparação com o Paradigma Lógico. PROLOG

Uso de Estratégias Formais

Em lógica e matemática, uma lógica proposicional é um sistema formal, no qual as fórmulas representam proposições que podem ser formadas pela combinação de proposições atômicas usando conectivos lógicos e um sistema de regras de derivação, que permite que certas fórmulas sejam estabelecidas como "teoremas" do sistema formal.

As proposições são sentença declarativa com valores verdadeiro ou falso. Por exemplo: "Maria gosta de João e de Pedro"; "Todos os seres humanos têm uma mãe"; "Cinco é maior do que quatro".

A lógica proposicional estuda como raciocinar com afirmações que podem ser verdadeiras ou falsas, deduzindo a partir de um certo conjunto de hipóteses (proposições verdadeiras), e considerando um dado contexto.

Análise de Requisitos

Verificação com REUTILIZAÇÃO DE DOMÍNIOS

Lembram-se da Falácia
da Página em Branco?

Reutilização de Domínio

Reutilização de Domínio...

- Técnica, na qual estratégias e heurísticas de Inteligência Artificial são usadas. Pretende-se criticar os requisitos, comparando-os com base em um domínio previamente codificado.

Portanto, a formação desse domínio seria baseada em fatos de sistemas similares que já tenham sido elicitados.

Assistentes inteligentes fornecem críticas sobre os requisitos preparados pelo Engenheiro de Software/Requisitos.

Dada a disponibilidade de um domínio, é possível determinar fatos errados e fatos faltantes.

FOCO:

- $\Delta = \{ \begin{array}{l} \text{fatos errados (fatos do domínio, fatos)} \\ \text{fatos faltantes (fatos do domínio, fatos)} \end{array} \}$



Análise de Requisitos

Validação

Validação

A validação de software, ou seja, a **confirmação de que o produto é aquele desejado pelo usuário**, ocorre, normalmente, no fim do ciclo de vida.

O teste do sistema, como é comumente conhecida esta validação, é o teste integrado dos programas do sistema pelo usuário.

No nosso caso, estamos preocupados e focados na validação feita no próprio processo de elicitação de requisitos, sendo essa anterior à própria especificação.

EM RESUMO: você fez o modelo orientando-se pela notação corretamente. O modelo está de acordo com o que se espera dele. Mas, **NÃO ATENDE O CLIENTE!** **NÃO É O QUE ELE DESEJA/ESPERA!**

Validação

Várias estratégias de validação de software têm sido propostas pela literatura.

Se por hipótese, definirmos que algumas dessas estratégias também podem ser aplicadas à elucidação de requisitos, teremos um conjunto, do qual sobressaem as seguintes estratégias:

- Comprovação informal;
- Prototipagem, e
- Baseada em Ponto de Vista.

Análise de Requisitos

Validação com **COMPROVAÇÃO INFORMAL**

Comprovação Informal

Comprovação Informal:

- A validação ou a revisão dos requisitos, nesse caso, é basicamente uma tarefa de leitura de descrições em **linguagem natural** e do uso dos clientes para identificar problemas na expressão dos requisitos.
- As estratégias para comprovação informal são várias, mas elas têm em comum a **falta de um apoio automatizado**, e a **excessiva dependência das habilidades analíticas dos leitores**.

FOCO:

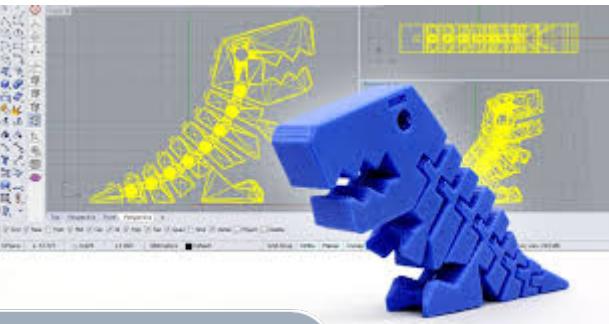
- $\Delta = \text{lista de erros}(\text{informação}, \text{fatos})$



Análise de Requisitos

Validação com PROTOTIPAÇÃO

Prototipação



Prototipação OU Prototipagem

- Em prototipação, vários tipos de protótipos têm sido propostos a fim de obter uma retroalimentação do Universo de Informações.
- Alguns deles usam linguagem de alto nível (linguagens do tipo geradores de aplicação), enquanto outros usam linguagens de especificação executáveis.
- A ideia básica é que pela prototipação é possível validar os requisitos/especificação com base nas expectativas do usuário.

FOCO:

- $\Delta =$ comportamento dos fatos(expectativas do usuário, fatos)



Análise de Requisitos

Validação com **ANÁLISE DE PONTOS DE VISTA**



Análise de Pontos de Vista

Análise de Pontos de Vista

- Aqui, tem-se foco em três tipos de problemas: inconsistências, fatos errados e fatos faltantes.
- São levados em conta diferentes pontos de vista.

FOCO:

- $\Delta = \begin{cases} \text{inconsistências}(fatos}_{pv1}, fatos_{pv2}) \\ \text{fatos errados}(fatos}_{pv1}, fatos_{pv2}) \\ \text{fatos faltantes}(fatos}_{pv1}, fatos_{pv2}) \end{cases}$



Análise de Pontos de Vista

Na tarefa de modelar as expectativas dos usuários, dentro de um Universo de Informações, o Engenheiro de Software/Requisitos pode encontrar, e frequentemente encontra, **opiniões diferentes sobre o problema em questão.**

Diferentes Engenheiros de Software/Requisitos, quando modelando as expectativas de usuários num mesmo Universo de Informações, **produzem diferentes modelos.**

O **mesmo Engenheiro de Software/Requisitos**, quando modelando o mesmo Universo de Informações, pode fazê-lo usando **diferentes perspectivas.**

Tudo isso é conhecido!

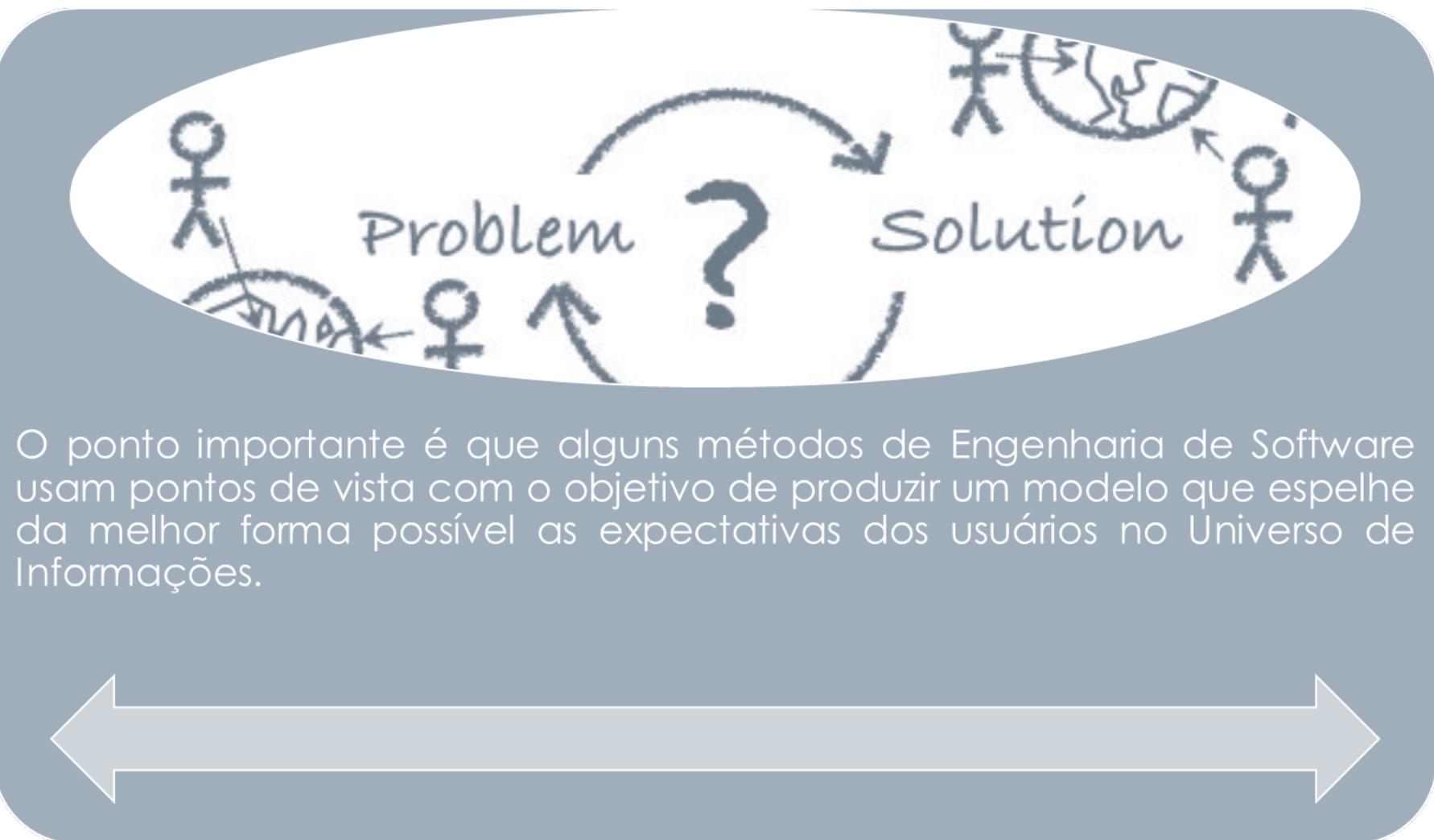
Análise de Pontos de Vista

Apenas uma
brincadeira para
melhor
entendimento
quanto aos
pontos de vista...

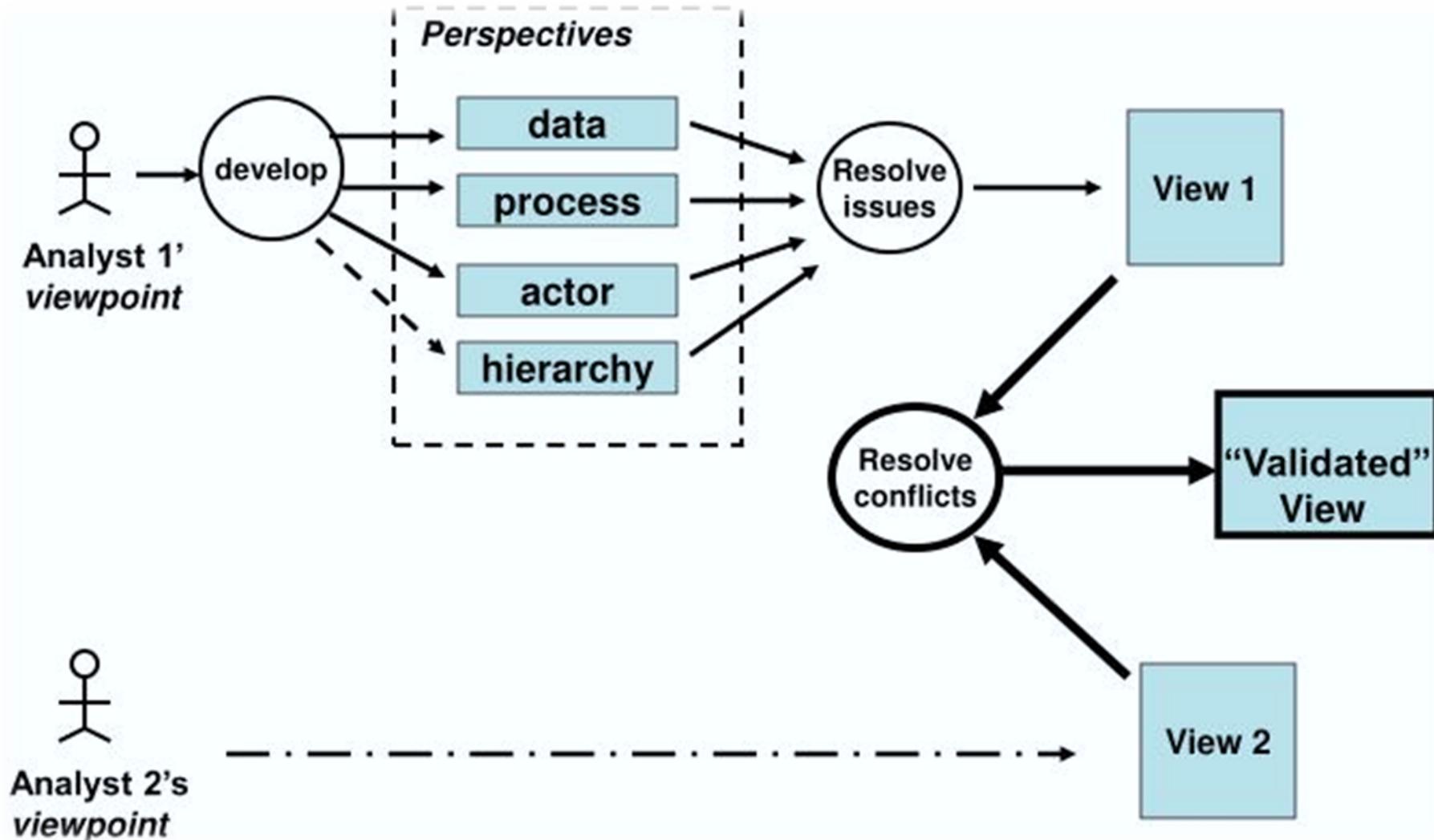


Fonte da Figura: <<internet, em material aberto>>

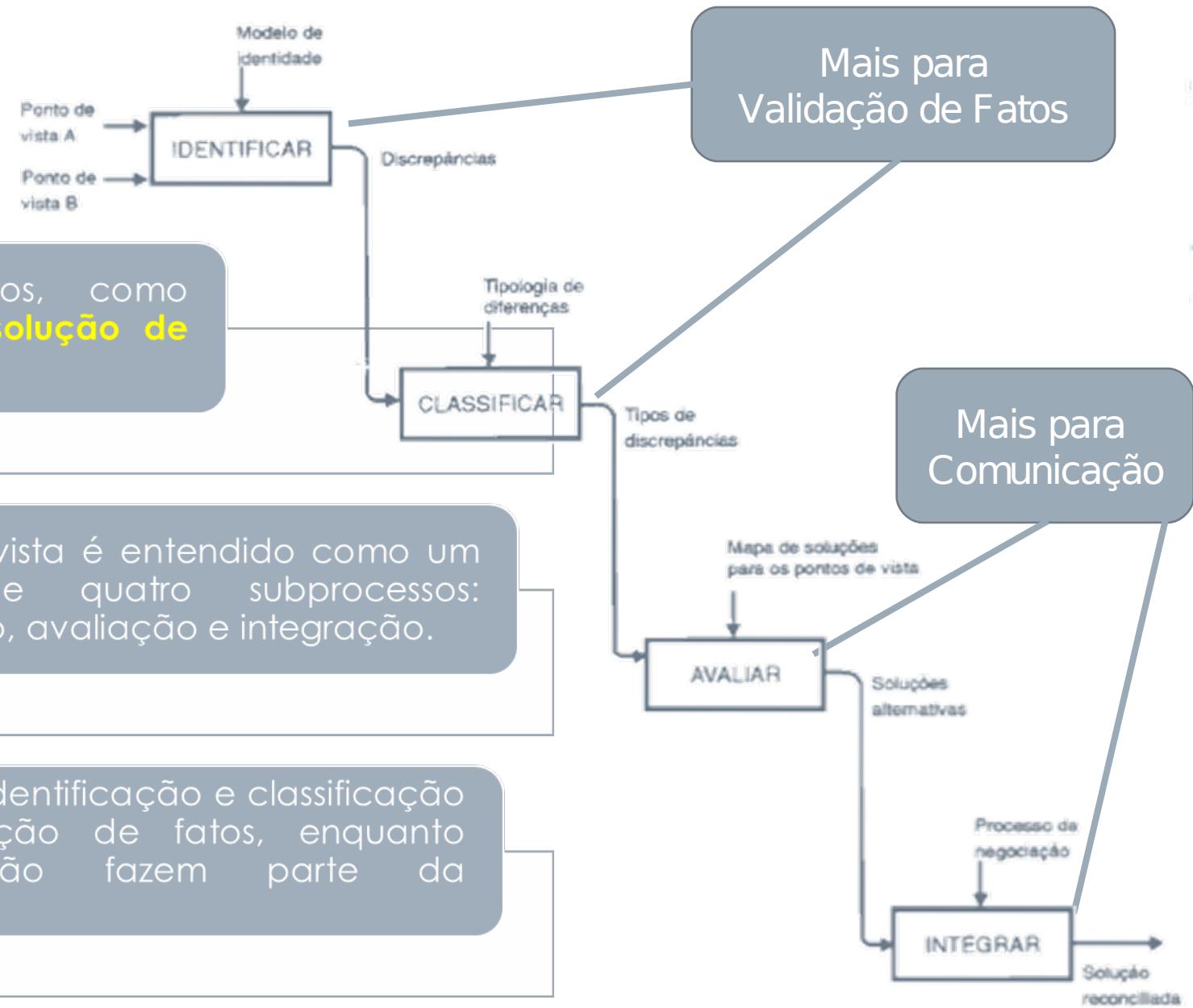
Análise de Pontos de Vista



Análise de Pontos de Vista



Resolução



Análise de Requisitos

Teste de Software

Teste de Software

Com as metodologias ágeis, tem-se incremento de software desde as primeiras iterações do desenvolvimento de software.

Portanto, é comum realizarmos verificação e validação com apoio do que chamamos de teste de software, executando o código.

Podemos entendê-los – os Testes de Software - como sendo o processo de executar programas usando casos de teste, com o objetivo de encontrar defeitos, em um ambiente controlado.

Dado de
Teste

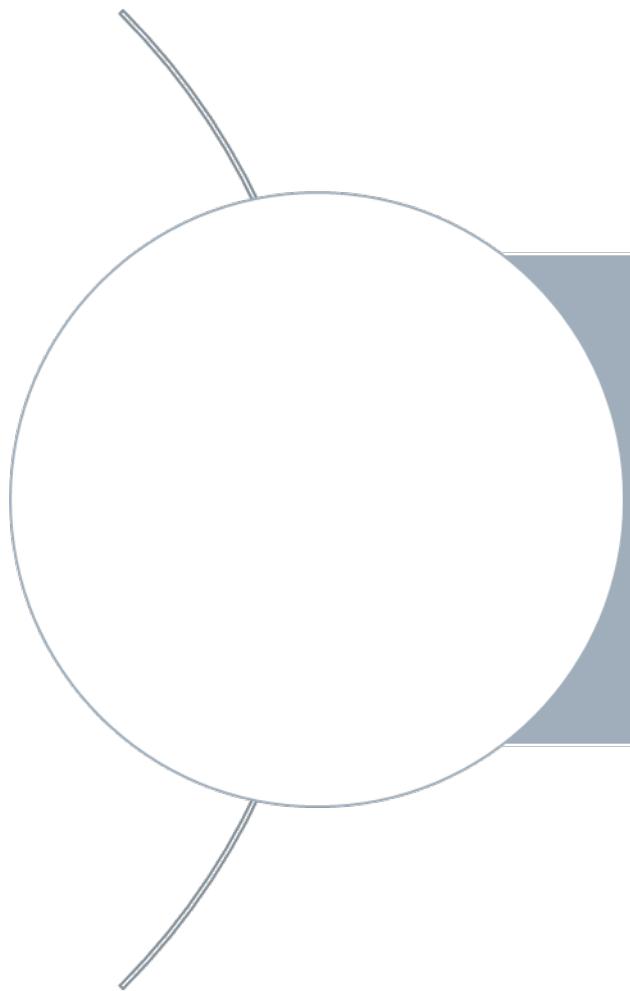


Saída
Esperada



Caso de
Teste

Teste de Software

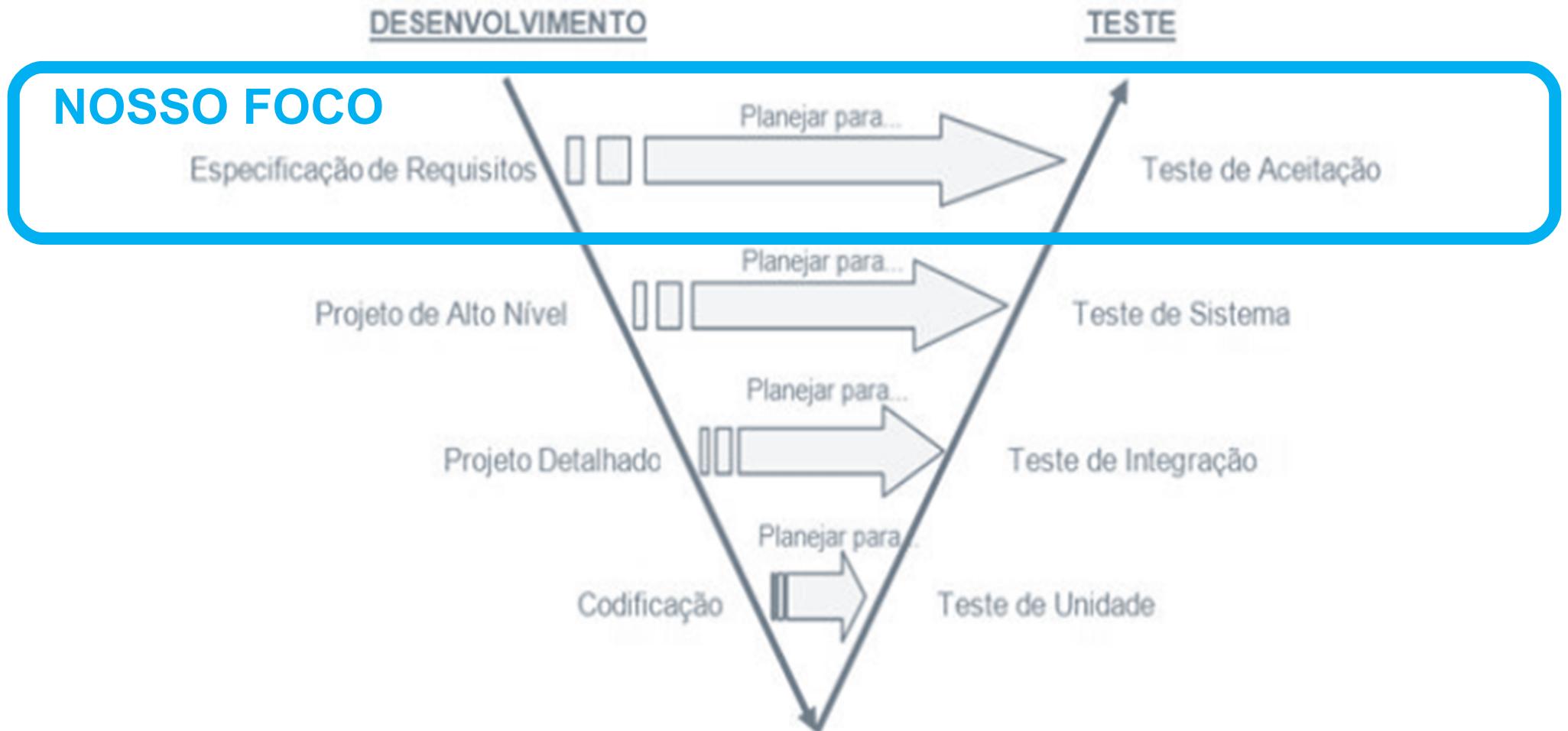


Um bom teste:

- Tem alta probabilidade de encontrar um erro.
- Não é redundante.
- Não deve ser muito simples, nem muito complexo.

* No caso da redundância, têm exceções, dependendo da criticidade do sistema.

Teste de Software



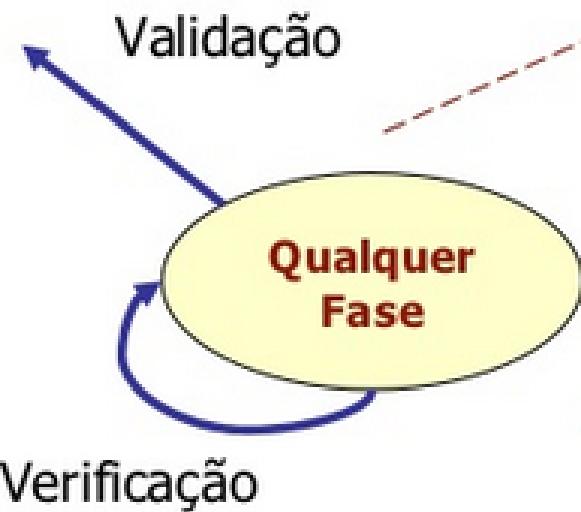
Não nos aprofundaremos em testes que dependem de execução de código.
Mas, existe uma disciplina dedicada a esse tipo de conhecimento.

Em resumo...

Em resumo...

O software deve atender às necessidades dos usuários.

Estamos construindo o produto certo?



Os artefatos construídos devem estar de acordo com a especificação do software.
Estamos construindo certo o produto?



Processo de exercitar um sistema ou componente para verificar que este satisfaz as especificações e para identificar falhas.

Considerações Finais

Considerações Finais

- › Nessa aula, foi apresentada a atividade de análise de requisitos.
- › No caso, focou-se nas tarefas:
 - *Identificação de Partes;*
 - *Verificação, e*
 - *Validação.*
- › Adicionalmente, comentou-se sobre Testes de Software.
- › Continuem os estudos!



Referências

Referências

Bibliografia Básica

1. [Ebrary] Young, Ralph. Requirements Engineering Handbook. Norwood, US: Artech House Books, 2003.
 2. [Open Access] Leite, Julio Cesar Sampaio do Prado. Livro Vivo - Engenharia de Requisitos. <http://livrodeengenhariaderequisitos.blogspot.com.br/> (último acesso: 2017)
 3. [Ebrary] Chemuturi, Murali. Mastering Software Quality Assurance : Best Practices, Tools and Technique for Software Developers. Ft. Lauderdale, US: J. Ross Publishing Inc., 2010.
 4. Software & Systems Requirements Engineering: In Practice - Brian Berenbach, Daniel Paulish, Juergen Kazmeier, Arnold Rudorfer (Livro bem completo mas, não tem exemplar físico na biblioteca, nem mesmo consta na Ebrary)
 5. Requirements Engineering and Management for Software Development Projects - Murali Chemuturi (Livro bem completo mas, não tem exemplar físico na biblioteca, nem mesmo consta na Ebrary)
-

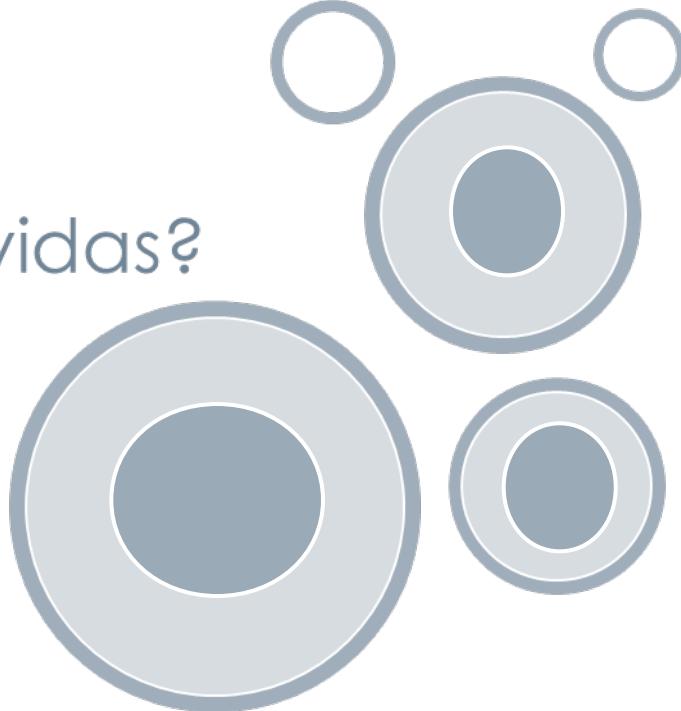
Referências

Bibliografia Complementar

1. [BIBLIOTECA – 15 exemplares] Pfleeger, Shari Lawrence. Engenharia de Software: Teoria e Prática. 2ª. Edição. São Paulo: Prentice Hall, c2004. xix, 535 p. ISBN 978858791831
 2. [BIBLIOTECA – 3 exemplares] Withall, Stephen. Software Requirement Patterns. Redmond: Microsoft Press, c2007. xvi, 366 p. ISBN 978735623989.
 3. [BIBLIOTECA - vários exemplares] Leffingwell, 2011, Agile Software Requirements, <http://www.scaledagileframework.com/> (último acesso: 2017)
 4. [Elibrary] Evans, Isabel. Achieving Software Quality Through Teamwork. Norwood, US: Artech House Books, 2004.
 5. [Elibrary] Yu, Eric, Giorgini, Paolo, and Maiden, Neil, eds. Cooperative Information Systems: Social Modeling for Requirements Engineering. Cambridge, US: MIT Press, 2010.
 6. [Open Access] Slides disponíveis em: <https://www.wou.edu/~eltonm/Marketing/PP%20Slides/> (último acesso: 2017)
-

FIM

Dúvidas?



Orientações?

Sugestões?

mileneserrano@unb.br ou mileneserrano@gmail.com
serrano@unb.br ou serr.mau@gmail.com