PONTIFÍCIA UNIVERSIDADE CATÓLICA DE SÃO PAULO

Engenharia de Software Processo

Introdução a Requisitos de Software

Prof. Carlos Eduardo de B. Paes
Departamento de Computação
Pontifícia Universidade Católica de São Paulo
carlosp@pucsp.br

O Problema da Pedra

- Clientes normalmente nos dão a missão que pode ser descrita como “Traga-me uma **pedra**”

O Problema da Pedra

- Quando você lhe entrega uma pedra, …

- … o cliente diz: "Sim, mas …, na verdade …, o que eu queria era uma **pequena pedra azul**".

O Problema da Pedra

- Quando você lhe entrega uma pequena pedra azul, …

- … o cliente diz: "Sim, mas …, na verdade …, o que eu realmente queria era uma pequena pedra esférica e azul".

O Problema da Pedra

- Quando você lhe entrega uma pequena pedra esférica e azul, …

- … o cliente diz: "Sim, mas …, na verdade …, o que eu realmente queria era uma **pequena pedra esférica de mármore azul**".

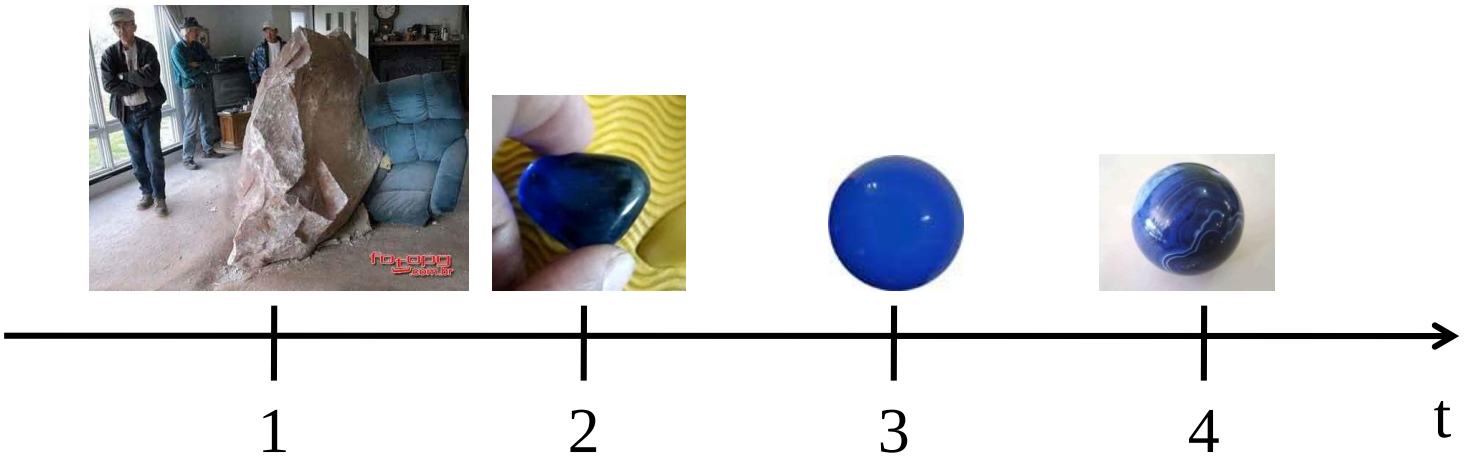
O Problema da Pedra

- Quando você lhe entrega uma pequena pedra esférica de mármore azul, …

- … o cliente finalmente diz: “Finalmente! Era isso que eu queria!”.

O Problema da Pedra

- O que será que aconteceu entre a entrega da primeira e a última pedra?



Cliente?

Desenvolvedor?

O Problema da Pedra

- Talvez o cliente tenha mudado o seu desejo sobre o que queria

- Certamente o desenvolvedor ficou frustrado

- Afinal, foi ele que carregou as pedras!
- O cliente também ficou frustrado
- Embora possa ter encontrado dificuldades para dizer o que queria, ele está convencido de que expressou seus desejos claramente.
- Foi o desenvolvedor não entendeu!

O Problema da Pedra

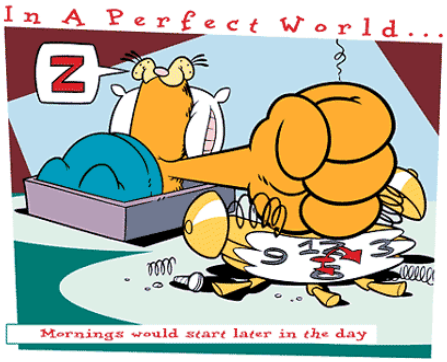
- Para complicar, em projetos reais existem, normalmente, mais pessoas envolvidas. Por exemplo:
  - Cliente
  - Desenvolvedor
  - Pessoal de marketing
  - Pessoal de testes e garantia de qualidade
  - Gerentes de produtos
  - Gerente geral, e
  - Vários outros stakeholders.

O Problema da Pedra

- Além disso, clientes articulam requisitos de software vagos supondo que depois poderão:
  - Esclarecer melhor
  - Alterar ou
  - Adicionar novos requisitos
  - Por acreditarem que o software, por natureza, são flexíveis!
- A flexibilidade do software frente ao hardware existe, porém, o software é:
  - Intangível
  - Abstrato
  - Complexo
  - Mutável

O Problema da Pedra

- Não é por isso que vamos ficar fazendo e refazendo até acertar!
  - Não podemos gastar, por exemplo, 2 anos no "projeto da pedra" e, no final, ter que refazê-lo tudo novamente!
- Além disso, o "mundo ideal" não existe!
  - Não dá para especificar, desenvolver, testar e implantar um "sistema de pedras" em tempo zero e custo zero!
  - Também não temos investimento e tempo infinitos!



O Problema da Pedra

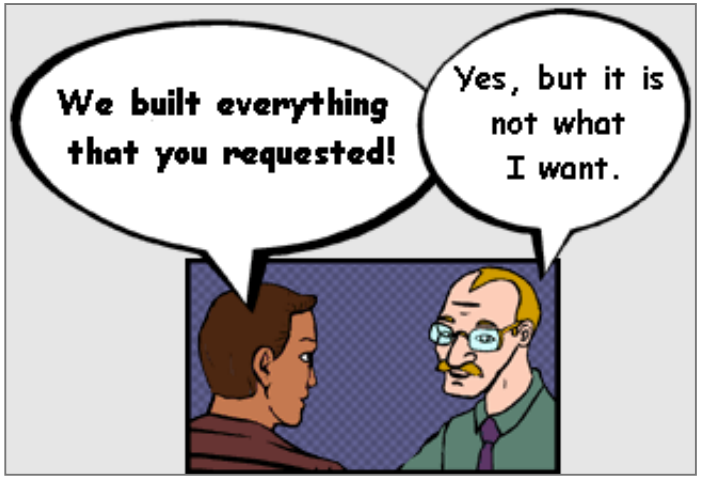
- Mais da metade dos projetos de software que estão atualmente em andamento, já ultrapassaram o custo e o cronograma.
- 25% a 33% desses projetos serão cancelados antes que estejam finalizados, normalmente após consumirem grandes somas de dinheiro.
- Prevenir tais falhas e fornecer uma abordagem racional para construir sistemas que o cliente deseja é o nosso objetivo.

O Problema da Pedra

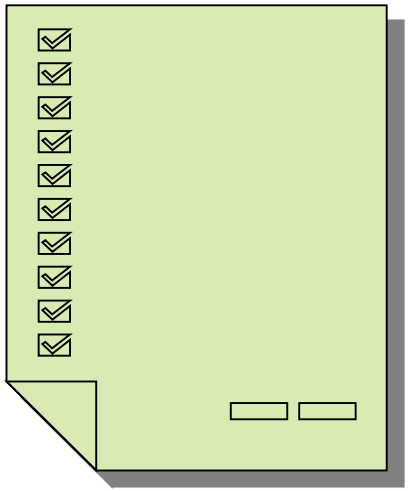
- Para tanto, é crucial que clientes, mas sobretudo desenvolvedores:
  - Tenham habilidades para definir e gerenciar o processo de requisitos, pois são eles que criam os requisitos iniciais e que, no final, determinam o sucesso ou a falha do sistema.
- Programadores heróis e solitários deveriam ser figuras do passado!

- Eles deveriam descansar em paz!

Situação familiar?



O que é Requisito?



Definição

- Segundo Dorfman e Thayer * (1990), um REQUISITO é:
  - Uma capacidade que o software precisa ter para atender às necessidades do cliente de resolver um problema e atingir seu objetivo.
  - Uma capacidade que o software do sistema ou de componentes do sistema deve ter para satisfazer um contrato, padrão, especificação, ou outros documentos formalmente impostos (IEEE, 1990).
- Com base nesta definição, podemos entender que o nosso objetivo é:
  - Levantar REQUISITOS de software que permita a construção de um sistema que atenda as NECESSIDADES do CLIENTE e, assim, resolva o seu PROBLEMA.

* Software Requirements - a management perspective - System and Software Requirements Engineering - Dorfman, M. and Thaye, IEEE, 1990

Requisito

- **Requisitos do usuário**
  - Declarações, em linguagem natural e diagramas, sobre os serviços que o sistema oferece e as restrições para a sua operação. Escrito para os clientes
- **Requisitos do sistema**
  - Estabelecem detalhadamente as funções e restrições do sistema. O documento de requisitos, chamado de especificação funcional, pode servir como um contrato entre cliente e desenvolvedor
- **Especificação de software**
  - Especificação abstrata e precisa do software, indicando o que ele deve fazer (sem dizer como) que serve de base para o projeto e para a implementação
  - Acrescenta mais detalhes à especificação funcional e é escrito para a equipe de desenvolvimento

Requisito

- Requisitos servem como especificação do que deve ser implementado
- Requisitos são descrições de como o sistema deve se comportar, de uma propriedade ou atributo do sistema
- Um requisito pode descrever:
  - uma facilidade encontrada no nível do usuário
  - uma propriedade geral do sistema
  - uma restrição do sistema
  - uma restrição ao desenvolvimento do sistema

(Sommerville, 2003)

Exemplos

- “O sistema deve rodar em microcomputadores da linha PC que possuam microprocessador Pentium ou superior”
- “A interface do sistema deve ser gráfica, de acordo com um padrão de interface dirigida a menu”
- “Alternativamente, o sistema deve possibilitar o seu uso através de linhas de comando, para usuários avançados”
- “O gerente da padaria deve consultar quanto vendeu em um dia”

Algumas Diretrizes

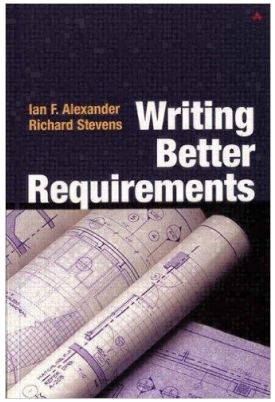
- Definir um formato padrão e usá-lo para todos os requisitos
- Utilizar o idioma de forma consistente. Usar “deve” para requisitos obrigatórios, “deveria” (ou é recomendável) para requisitos desejáveis
- Evitar o uso de jargões de computação
- Empregar termos característicos do problema

Algumas Diretrizes

- Use sentenças diretas e objetivas
- Use vocabulário limitado
- Defina requisitos verificáveis (eu consigo testar?)
- Evite ambiguidades
- Evite sentenças muito longas
- Evite uso de conjunções como ou, e, com, também
- Evite termos vagos ou indefinidos

Referência Importante

- Ian Alexander & Richard Stevens, **Writing Better Requirements**, Addison-Wesley, 2012.



- Ivy Hooks, **Writing Good Requirements**, Published in the Proceedings of the Third International Symposium of the INCOSE - Volume 2, 1993.

Gerenciamento de Requisitos

- Uma abordagem sistemática de fazer levantamento, organizar e documentar os requisitos do sistema e
- Um processo que estabelece e mantém concordância entre o cliente e a equipe do projeto na alteração de requisitos do sistema.

Gerenciamento de Requisitos

- Essa definição é similar a de Dorfman e Thayer e a definição do IEEE de "engenharia de requisitos de software".
- A engenharia de requisitos inclui levantamento, análise, especificação, verificação e gerenciamento dos requisitos de software, em que o "gerenciamento de requisitos de software" é o planejamento e o controle de todas essas atividades relacionadas

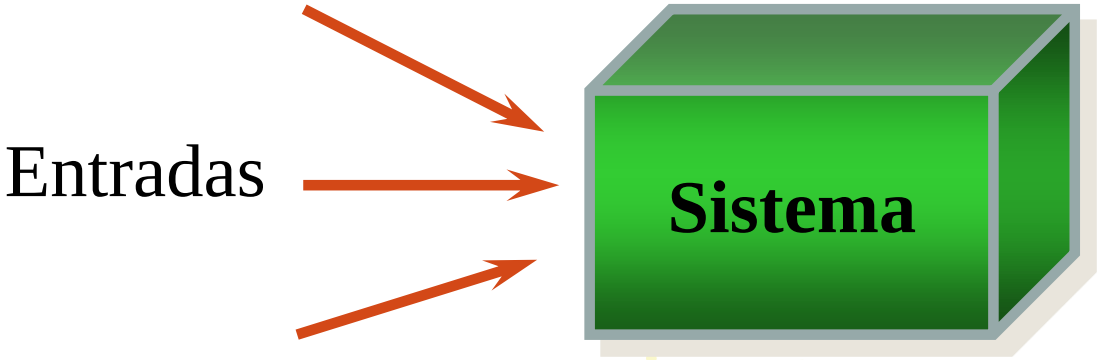
Problemas Comuns de Requisitos

- Pesquisa 1996
  - Falta de rastreabilidade de mudanças (71%)
  - Dificuldades de escrita (70%)
  - Características fora de controle (67%)
  - Falta de organização (54%)
- Outros
  - Os requisitos nem sempre são óbvios e têm muitas fontes.
  - Nem sempre é fácil expressar os requisitos claramente em palavras.
  - Existem diversos tipos de requisitos em diferentes níveis de detalhe.

Problemas Comuns de Requisitos

- Outros:
  - O número de requisitos poderá impossibilitar a gerência se não for controlado.
  - Os requisitos estão relacionados uns com os outros e com outros distribuíveis do processo de várias maneiras.
  - Os requisitos têm propriedades exclusivas ou valores de propriedade. Por exemplo, eles não são igualmente importantes nem igualmente fáceis de atender.
  - Há vários envolvidos e responsáveis, o que significa que os requisitos precisam ser gerenciados por grupos de pessoas de diferentes funções.
  - Os requisitos são alterados.

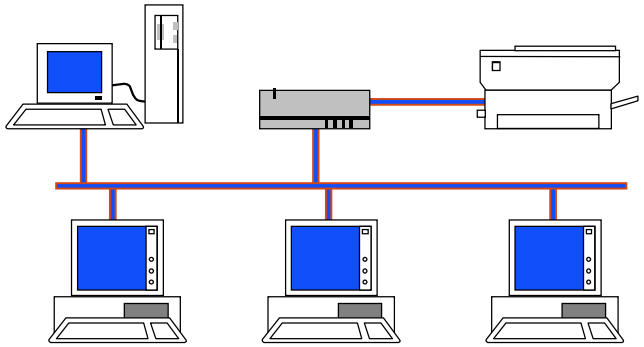
O que requisito de software especifica?



Funções

Requisitos
Não-Funcionais
(Ex: Performance)

Restrições de Projeto



(Ex: Ambiente)

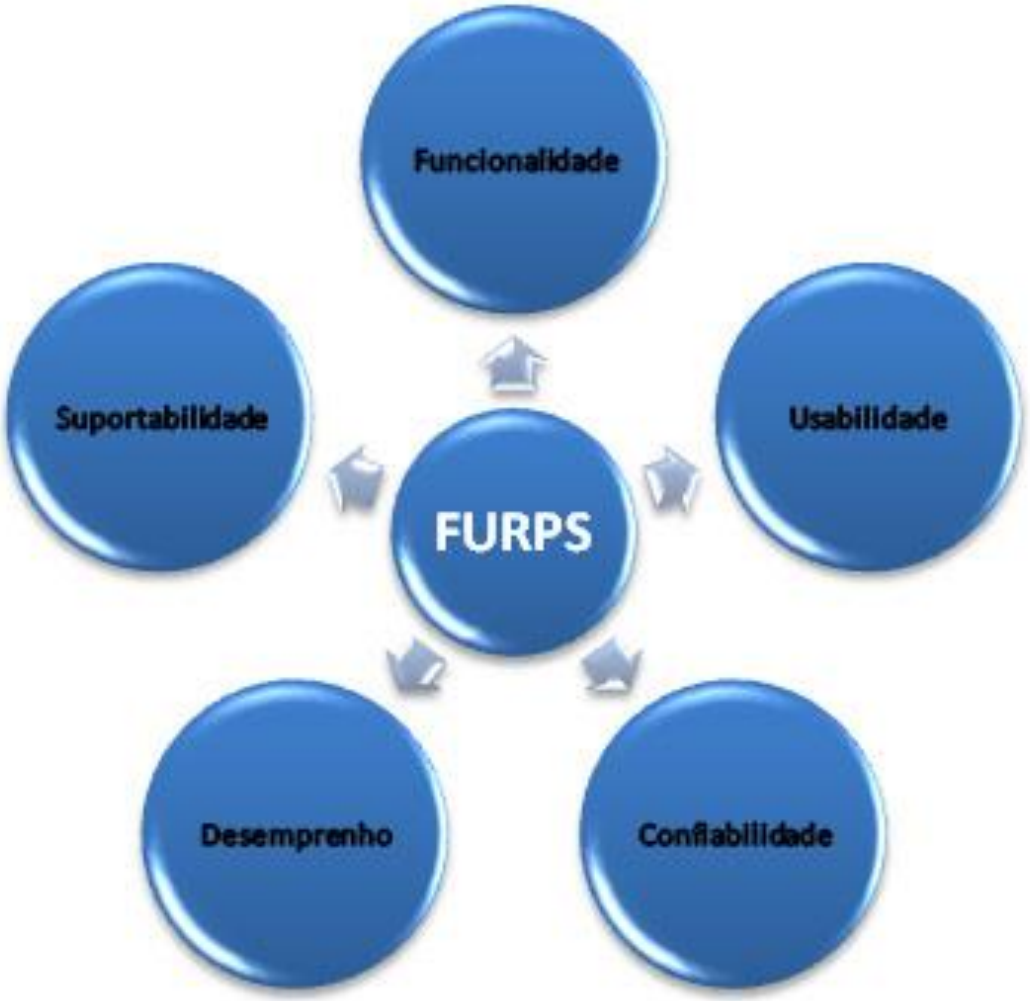
Tipos e Categorias de Requisitos

- Modelo FURPS+ (Grady, 1992) (Functionality, Usability, Reliability, Performance, Supportability)
  - **Funcional**: características, capacidade e segurança
  - **Usabilidade**: fatores humanos, recursos de ajuda, documentação
  - **Confiabilidade**: frequência de falhas, capacidade de recuperação, previsibilidade
  - **Desempenho**: tempos de resposta, fluxo de vazão (throughput), disponibilidade, uso de recursos
  - **Facilidade de suporte**: facilidade de adaptação e de manutenção, internacionalização, configurabilidade

Tipos e Categorias de Requisitos

- O + em FURPS+
  - **Implementação**: limitações de recursos, linguagens e ferramentas, hardware e etc.
  - **Interface**: restrições impostas pelas interfaces com sistemas externos
  - **Operações**: gerenciamento do sistema no ambiente operacional
  - **Empacotamento**: por exemplo, uma caixa física
  - **Questões** Legais: licenças de uso, etc.

Tipos e Categorias de Requisitos



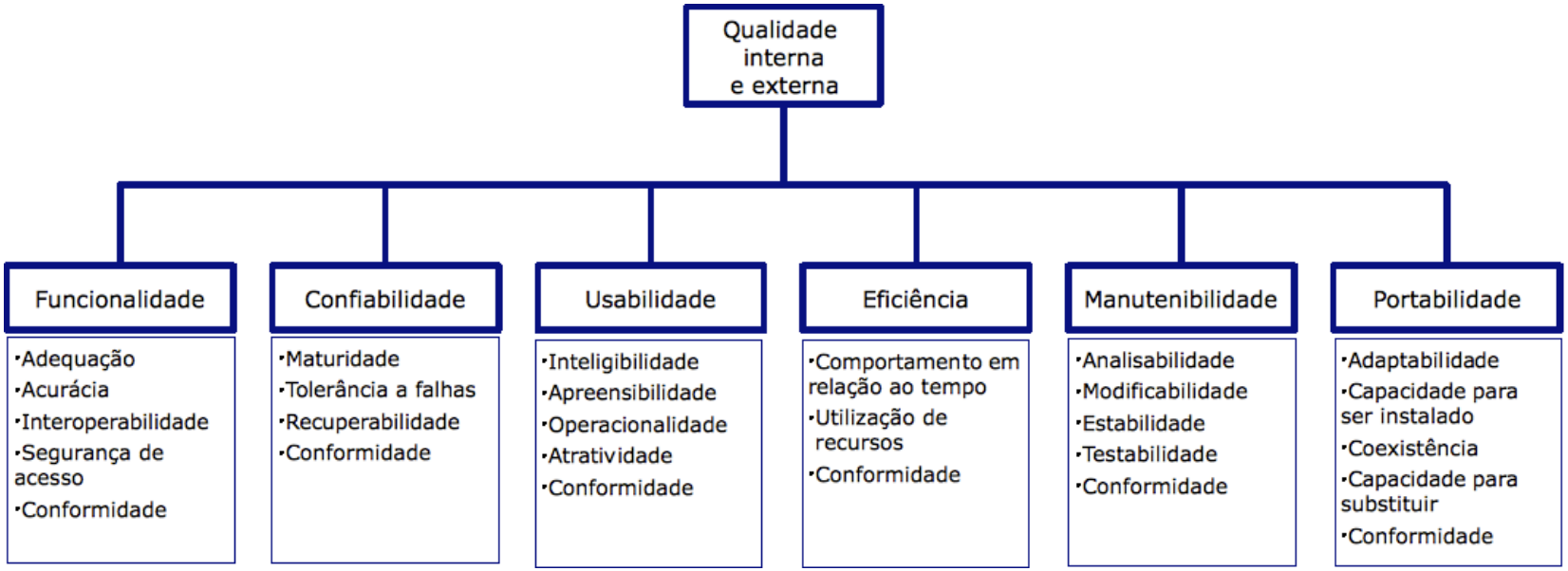
Tipos e Categorias de Requisitos

- Outros sistemas de categorização
  - ISO/IEC 9126 (norma para qualidade de produto de software)
  - SEI
- Também chamados de atributos de qualidade
- Normalmente requisitos são categorizados em
  - Funcionais (comportamentais)
  - Não-Funcionais (todos os outros)

ISO : https://pt.wikipedia.org/wiki/Organiza%C3%A7%C3%A3o_Internacional_para_Padroniza%C3%A7%C3%A3o

ISO/IEC 9126

- Foco na qualidade do produto de software



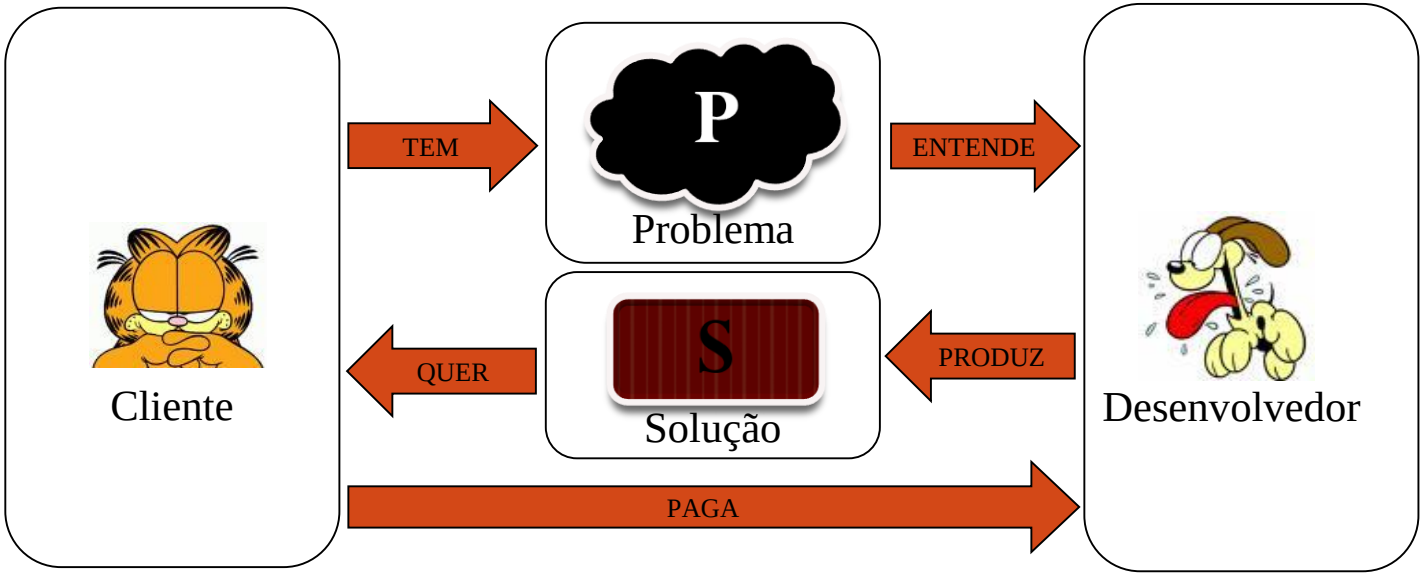
A Pirâmide de Requisitos

- Para nos ajudar nesta Engenharia de Requisitos, usaremos a Pirâmide de Requisitos como um Mapa Mental para:
  - Obter os principais passos a seguir
  - Entender a lógica desses passos

A Pirâmide de Requisitos

Desenvolvemos software porque:

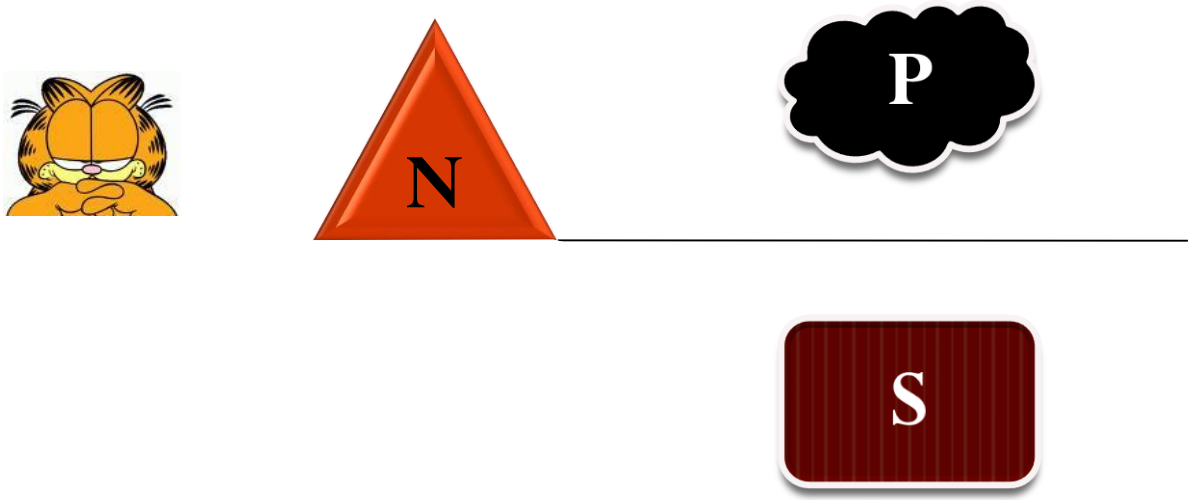
- Existe alguém que acredita que:
  - O software é a SOLUÇÃO do seu PROBLEMA
  - O software é a SOLUÇÃO para ganhar vantagem competitiva (OPORTUNIDADE de negócio)
- Existe alguém disposto a pagar por essa SOLUÇÃO



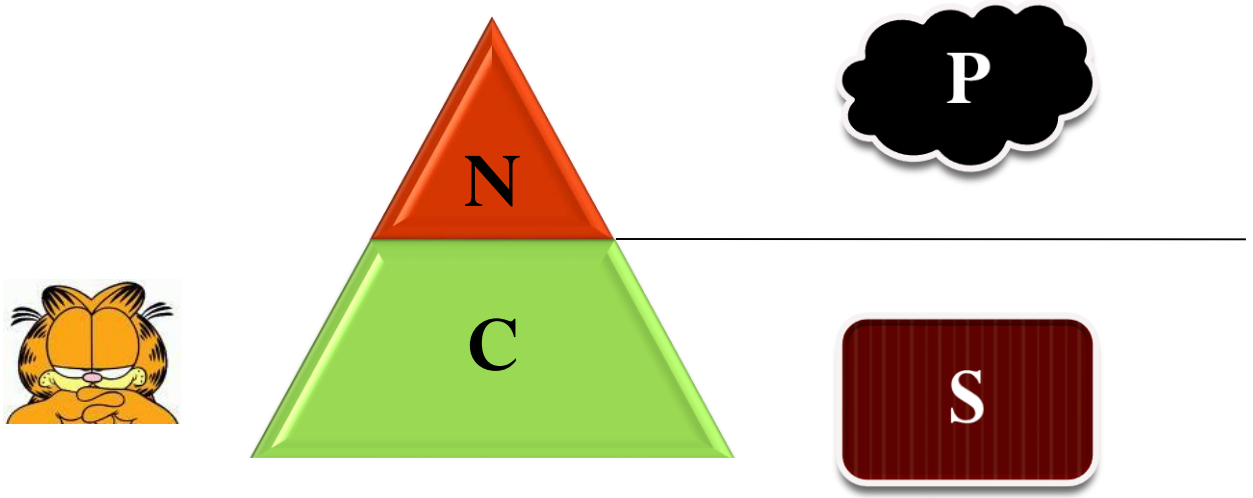
A Pirâmide de Requisitos

- Um PROBLEMA pode possuir várias alternativas de SOLUÇÃO
- Para que o Cliente possa selecionar uma dessas alternativas, ele precisa conhecer as CARACTERÍSTICAS de cada SOLUÇÃO

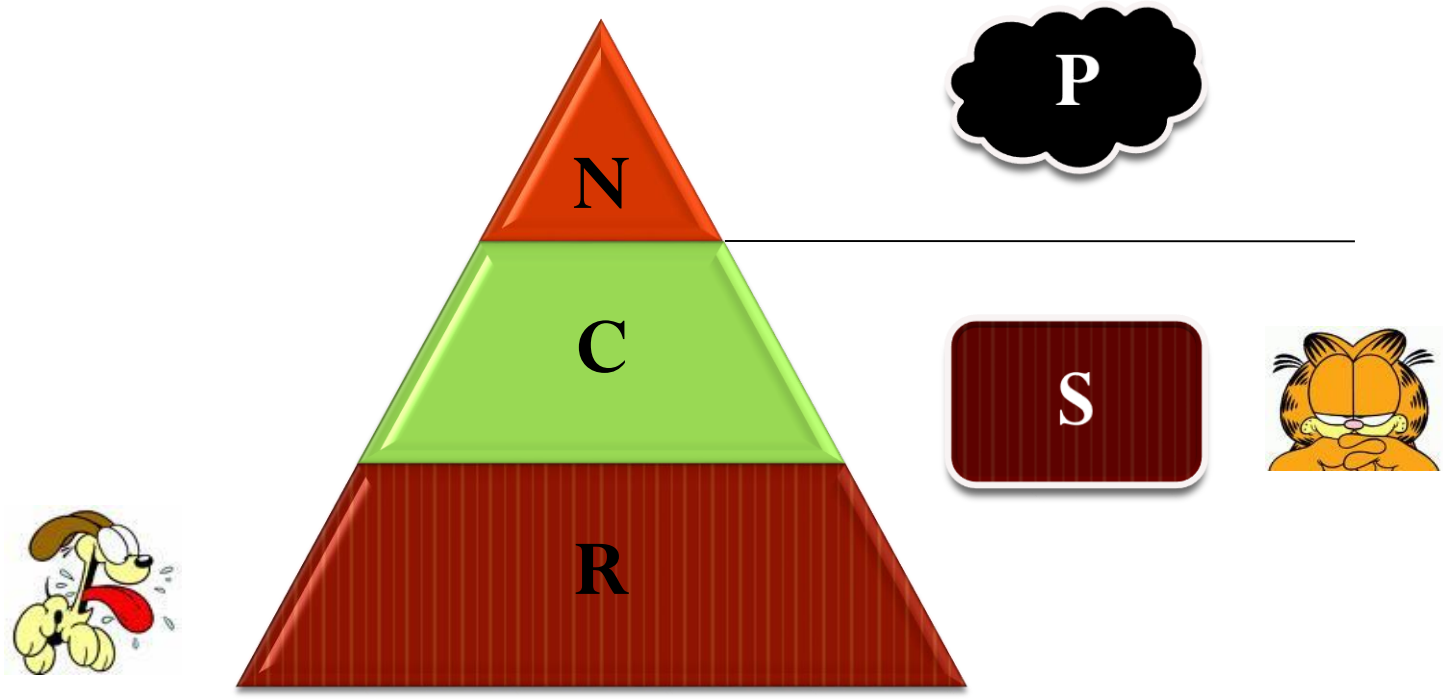
- Para produzirmos uma SOLUÇÃO, precisamos, antes, entender o PROBLEMA e as NECESSIDADES do Cliente



- Um **P**ROBLEMA pode possuir várias alternativas de **S**OLUÇÃO
- Para que o **Cliente** possa selecionar uma dessas alternativas, ele precisa conhecer as **C**ARACTERÍSTICAS de cada **S**OLUÇÃO



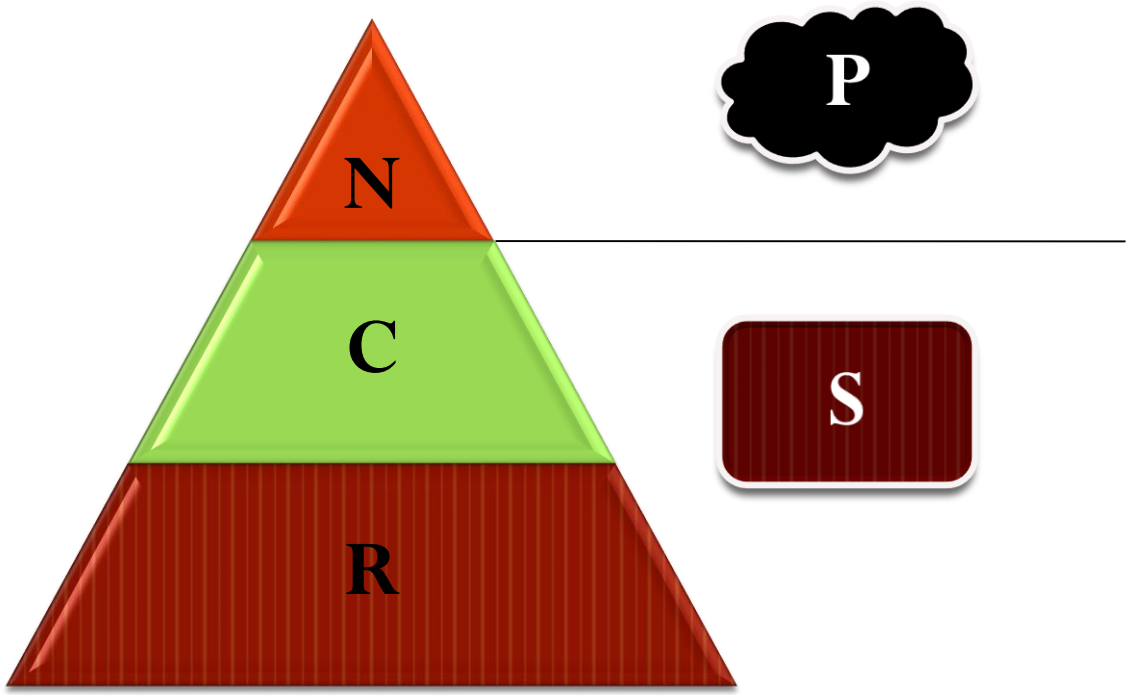
- Para que o **Desenvolvedor** possa produzir a **S**OLUÇÃO selecionada pelo **Cliente**, ele deve atender a um conjunto de **R**EQUISITOS da **S**OLUÇÃO



- É difícil para alguém racional ir contra a ideia de realizar um bom trabalho de Engenharia de Requisitos
- No entanto pesquisas demonstram que, como uma indústria, frequentemente falhamos devido:
  - A falta de retorno dos usuários
  - Requisitos e especificações incompletas
  - Mudanças nos requisitos e especificações

- Um pensamento comum entre desenvolvedores e clientes é:
  - É melhor iniciar logo a implementação, porque estamos atrasados. Podemos determinar os requisitos mais tarde
- Embora bem intencionada esta abordagem degenera-se para um esforço de desenvolvimento caótico
- Alguns crêem que protótipos podem ser evoluídos (*refactoring*) sempre que Clientes solicitarem alterações
  - Mas normalmente não temos **tempo** e **recursos** ilimitados para fazer o que o Cliente deseja a qualquer momento
  - Por isso definimos um contrato onde **Custo**, **Tempo** e **Escopo** são fixados

- Assim, fazer um bom trabalho de Engenharia de Requisitos é essencial para construirmos uma **S**OLUÇÃO que atenda às **N**ECESSIDADES do Cliente e resolva o seu **P**ROBLEMA dentro do **Custo**, **Tempo** e **Escopo** estabelecidos em contrato



Requisitos e o RUP

- O RUP é dirigido por casos de uso
- Técnica criação por Jacobson
- Casos de uso → requisitos funcionais do sistema
- Temos dois tipos de requisitos: funcionais e não funcionais
- Os requisitos são evolutivos, ou seja, são identificados e detalhados ao logo do ciclo de desenvolvimento
- Procuramos atacar os requisitos mais críticos e importantes o mais cedo possível
  - Ex: na fase de Concepção e Elaboração

Requisitos e o RUP

- Como os requisitos são organizados no RUP?
  - Modelo de Casos de Uso
  - Especificação Suplementar
  - Glossário
  - Visão do Produto
  - Regras de Negócio

Mapa do território