Requisitos de Software

Engenharia de requisitos

- Estabelece os serviços que o cliente requer de um sistema e as restrições sob as quais tal sistema operará e será desenvolvido.
  - Tais serviços e restrições são chamados de **requisitos**

O que é um requisito?

- Pode ser uma descrição abstrata de alto nível de um serviço, uma restrição de sistema ou até uma especificação matemática, entre outras coisas
- Define o problema cujo desenvolvimento do sistema deve resolver
  - O sistema tem que ser construído de modo a satisfazer **todos** os seus requisitos

Requisitos funcionais e não-funcionais

- **Requisitos funcionais**
  - Serviços que o sistema deve fornecer
  - Como o sistema deve reagir a entradas específicas
  - Como o sistema deve se comportar em determinadas situações.
- **Requisitos não-funcionais ou de qualidade**
  - Restrições sobre serviços ou funções oferecidos pelo sistema, tais como restrições de tempo, restrições sobre o processo de desenvolvimento, padrões, linguagens de programação, etc.

Sistema exemplo: LIBSYS

- Um sistema de biblioteca que fornece uma interface única para uma série de banco de dados de artigos em bibliotecas diferentes.
- Os usuários podem pesquisar, baixar e imprimir estes artigos para estudo pessoal.

Exemplos de requisitos funcionais

- O usuário deve ser capaz de pesquisar em todo o conjunto do banco de dados ou selecionar um subconjunto a partir dele.
- Para todo pedido deve ser alocado um identificador único (ORDER_ID).
- O sistema deve fornecer **telas apropriadas** para o usuário ler os documentos no repositório de documentos.

Imprecisão de requisitos

- Problemas surgem quando os requisitos não são precisamente definidos.
- Requisitos ambíguos podem ser interpretados de maneiras diferentes pelos desenvolvedores e usuários.
- Considere o termo “telas apropriadas”
  - Intenção do usuário – tela de propósito especial para cada tipo diferente de documento;
  - Interpretação do desenvolvedor – fornece uma tela de texto que mostra o conteúdo do documento.

Requisitos completos e consistentes

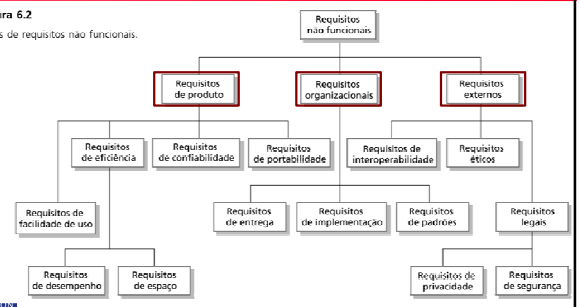
- Em princípio, requisitos devem ser completos e consistentes.
- Completude
  - Eles devem incluir descrições de **todos** os recursos requeridos.
- Consistência
  - Não deve haver **conflitos** ou **contradições** nas descrições dos recursos de sistema.
- Na prática, é impossível produzir um documento de requisitos completo e consistente.

Requisitos não-funcionais

- Definem propriedades e restrições de sistema
- Exemplos incluem confiabilidade, tempo de resposta e requisitos de armazenamento.
- Restrições são capacidade de dispositivos de E/S, representações de sistema, etc.
- Requisitos de processo podem também ser especificados, impondo uma linguagem de programação, IDE ou método de desenvolvimento particular.
- Requisitos não-funcionais podem ser mais críticos do que os requisitos funcionais. Quando não satisfeitos podem tornar o sistema inútil.

Tipos de requisitos não-funcionais

Figura 6.2
Tipos de requisitos não funcionais.



Exemplos de requisitos não-funcionais

Quadro 6.2
Exemplos de requisitos não funcionais.

Requisito de produto
8.1 A interface de usuário para o LIBSYS deve ser implementada como simples HTML, sem frames ou applets de Java.

Requisito organizacional
9.3.2 O processo de desenvolvimento do sistema e os documentos a serem entregues devem estar em conformidade com o processo e produtos a serem entregues definidos em XYZCo-SP-STAN-95.

Requisito externo
10.6 O sistema não deve revelar quaisquer informações pessoais sobre os usuários do sistema ao pessoal da biblioteca que usa o sistema, com exceção do nome e número de referência da biblioteca.

Metas e R.N.F

- Requisitos não-funcionais podem ser difíceis de definir precisamente
  - Requisitos imprecisos podem ser difíceis de verificar.
- Meta
  - Uma intenção geral (geralemente abstrata) do usuário (ex: facilidade de uso).
  - São úteis para desenvolvedores quando exprimem as intenções dos usuários do sistema.
- Requisito não-funcional verificável
  - Uma declaração usando alguma medida que pode ser objetivamente testada.

Exemplo de Meta e R.N.F.

Quadro 6.3

Metas do sistema e requisitos verificáveis

Meta do sistema

O sistema deve ser fácil de ser usado pelos controladores experientes e ser organizado de modo que os erros dos usuários sejam minimizados.

Requisito não funcional verificável

Os controladores experientes devem ser capazes de usar todas as funções do sistema depois de um treinamento no total de duas horas. Após esse treinamento, o número médio de erros cometidos pelos usuários experientes não deve exceder dois por dia.

Medidas para R.N.F.

- Devido à sua própria definição, requisitos não-funcionais são esperados serem mensuráveis
- Assim, deve-se associar uma forma de medida/referência a cada requisito não-funcional elicitado

Medidas para R.N.F.

Tabela 6.1 Métricas para especificar requisitos não funcionais

Propriedade	Medida
Velocidade	Transações processadas/segundo Tempo de resposta de usuário/evento Tempo de atualização da tela
Tamanho	Kbytes Número de chips de RAM
Facilidade de uso	Tempo de treinamento Número de frames de ajuda
Confiabilidade	Tempo médio de falha Probabilidade de indisponibilidade Taxa de ocorrência de falhas Disponibilidade
Robustez	Tempo para reiniciar após falha Porcentagem de eventos que causam falhas Probabilidade de corrupção de dados por falhas
Portabilidade	Porcentagem de declarações dependentes do sistema-alvo Número de sistemas-alvo

Conflitos entre R.N.F.

- Conflitos entre os diferentes requisitos não-funcionais são comuns em sistemas complexos.
- Em um sistema de Hardware
  - Para minimizar o peso, o número de *chips* separados no sistema deve ser minimizado.
  - Para minimizar o consumo de energia, *chips* de baixa potência devem ser usados.
    - E o desempenho pode ser impactado!
  - Contudo, o uso de *chips* de baixa potência pode significar que mais chips devem ser usados . Qual é o requisito mais crítico?

Requisitos de usuário

- Requisitos funcionais e não-funcionais descritos de modo a serem compreensíveis por usuários que não têm conhecimento técnico detalhado.
  - Declarações de alto nível escritas em linguagem natural
  - São definidos usando uma linguagem simples, tabelas e diagramas quando estes podem ser compreendidos por todos os usuários
  - Histórias de usuários (comuns em metodologias ágeis) são similares a requisitos de usuários

Requisitos de sistema

- Um documento estruturado estabelecendo descrições detalhadas das funções, serviços e restrições operacionais do sistema.
  - Define o que deve ser implementado e pode até ser parte de um contrato entre o cliente e o desenvolvedor.
  - Fornecem uma base para o desenvolvimento do sistema
    - Em XP: **histórias de usuário** + **tarefas de desenvolvimento**
    - Em OpenUP: **casos de uso** + **especificação detalhada**
- Eles podem ser incorporados no contrato de sistema.
- Podem ser definidos ou ilustrados usando **notações gráficas**

Ex: Requisitos de usuário x de sistema

Quadro 6.1 Requisitos de usuário e de sistema.

Definição de requisitos de usuário

1. LIBSYS deve manter o acompanhamento de todos os dados exigidos pelas agências de licenciamento de direitos autorais no Reino Unido e em outros lugares.

Especificação dos requisitos de sistema

1.1 Ao solicitar um documento ao LIBSYS, deve ser apresentado ao solicitante um formulário que registra os detalhes do usuário e da solicitação feita.
1.2 Os formulários de solicitação do LIBSYS devem ser armazenados no sistema durante cinco anos, a partir da data da solicitação.
1.3 Todos os formulários do LIBSYS devem ser indexados por usuário, nome do material solicitado e fornecedor da solicitação.
1.4 O LIBSYS deve manter um registro de todas as solicitações feitas ao sistema.
1.5 Para materiais aos quais se aplicam os direitos de empréstimo dos autores, os detalhes do empréstimo devem ser enviados mensalmente às agências de licenciamento de direitos autorais que se registraram no LIBSYS.

Diretrizes para escrever requisitos

- Usar um formato padrão para todos os requisitos.
- Usar a linguagem de uma forma consistente.
  - 'deve' para requisitos obrigatórios, e
  - 'deveria' para requisitos desejáveis.
- *Realçar* o texto para identificar as partes principais do requisito.
- Evitar o uso de jargões de computação.

Requisitos e Projeto

- Requisitos devem definir o que o sistema deve fazer e o projeto deve descrever como ele faz isto.
  - Requisitos => problema
  - Projeto => solução
- Na prática, requisitos e projeto são inseparáveis
  - Uma arquitetura de sistema pode ser projetada para estruturar os requisitos;
  - O sistema pode ter que interoperar com outros sistemas que geram novos requisitos;
  - O uso de uma solução de projeto específica pode ser um requisito de domínio.

Problemas com linguagem natural

- Falta de clareza
  - É difícil atingir uma precisão sem tornar o documento difícil de ler e ambíguo
- Confusão de requisitos
  - Requisitos funcionais e não-funcionais tendem a estar misturados.
- Fusão de requisitos
  - Vários requisitos diferentes podem ser expressos juntos
- Dificuldade de estruturar a especificação

Alternativas à especificação em linguagem natural

Tabela 6.2 Notações para especificação de requisitos

Notação	Descrição
Linguagem natural estruturada	Esta abordagem depende da definição de formulários ou templates-padrão para expressar a especificação de requisitos.
Linguagens de descrição de projeto	Esta abordagem usa uma linguagem semelhante à linguagem de programação, porém com mais características abstratas, para especificar os requisitos por meio da definição de um modelo operacional do sistema. Essa abordagem não é amplamente usada hoje em dia, embora possa ser útil para especificações de interfaces.
Notações gráficas	Uma linguagem gráfica, complementada com anotações de texto é usada para definir os requisitos funcionais do sistema. Um antigo exemplo dessa linguagem gráfica é SADT (Ross, 1977) (Schoman e Ross, 1977). Atualmente, as descrições de casos de uso (Jacobsen, et al., 1993) e os diagramas de seqüência são comumente usados (Stevens e Pooley, 1999).
Especificações matemáticas	São notações baseadas em conceitos matemáticos, como máquinas de estados finitos ou conjuntos. Essas especificações não ambíguas reduzem discussões entre cliente e fornecedor. No entanto, a maioria dos clientes não compreende as especificações formais e são relutantes em aceitá-las no momento da contratação.

Especificações em linguagem estruturada

- A liberdade do elaborador de requisitos é limitada por um *template* pré-definido para requisitos.
- Todos os requisitos são escritos de maneira padronizada.
- A terminologia usada na descrição pode ser limitada.
- A vantagem é que a maior parte da expressividade da linguagem natural é mantida
  - Mas o grau de uniformidade é imposto na especificação.

Especificação em linguagem estruturada

Quadro 6.7

Definição de um recurso de grade de editor.

2.6.1 Recursos de grade

O editor deve fornecer um recurso de grade no qual uma matriz de linhas horizontais e verticais fornece um fundo para a janela do editor. Essa grade deve ser passiva e o alinhamento das entidades é de responsabilidade do usuário.

Justificativa lógica: Uma grade ajuda o usuário a criar um diagrama bem organizado com entidades bem espaçadas. Embora uma grade ativa, na qual as entidades 'saltam' as linhas de grade, possa ser útil, o posicionamento é impreciso. O usuário é a melhor pessoa para decidir onde as entidades devem ser posicionadas.

Especificação: ECLIPSE/WS/Tools/DE/FS Seção 5.6
Fonte: Ray Wilson, escritório de Glasgow

Especificação baseada em formulário

Tabela 6.3 Especificação de requisitos do sistema com utilização de um formulário-padrão

Bomba de insulina/Software de controle/SRS/3.3.2	
Função	Calcular dose de insulina: nível seguro de açúcar
Descrição	Calcula a dose de insulina a ser liberada quando o nível medido de açúcar atual está na zona segura entre 3 e 7 unidades
Entradas	Leitura atual de açúcar (r2), as duas leituras anteriores (r0 e r1)
Origem	Leitura atual de açúcar do sensor. Outras leituras da memória
Saídas	CompDose — a dose de insulina a ser liberada
Destino	Loop de controle principal
Ação	CompDose será zero se o nível de açúcar estiver estável ou em queda, ou se o nível estiver aumentando, mas a taxa de aumento estiver diminuindo. Se o nível estiver aumentando e a taxa de aumento estiver aumentando, então CompDose será calculado dividindo-se a diferença entre o nível atual de açúcar e o nível anterior por 4, e arredondando o resultado. Se o resultado do arredondamento for zero, então CompDose será definido como a dose mínima que pode ser liberada.
Requer	Duas leituras anteriores de modo que a taxa de mudança do nível de açúcar possa ser calculada.
Precondição	O reservatório de insulina conter, pelo menos, o máximo de dose única permitida de insulina
Pós-condição	r0 é substituído por r1, portanto r1 é substituído por r2
Efeitos colaterais	Nenhum

Especificação tabular

- Usada para suplementar a linguagem natural.
- Particularmente útil quando você tem de definir uma série de possíveis cursos alternativos de ação.

Especificação tabular

Tabela 6.4 Especificação tabular de cálculo

Condição	Ação
Nível de açúcar em queda (r2 < r1)	CompDose = 0
Nível de açúcar estável (r2 = r1)	CompDose = 0
Nível de açúcar aumentando e taxa de aumento diminuindo ((r2 r1) < (r1 r0))	CompDose = 0
Nível de açúcar aumentando e taxa de aumento estável ou aumentando ((r2 – r1) > (r1 – r0))	CompDose = arredondar ((r2 – r1)/4) Se o resultado do arredondamento for = 0 então CompDose = DoseMínima

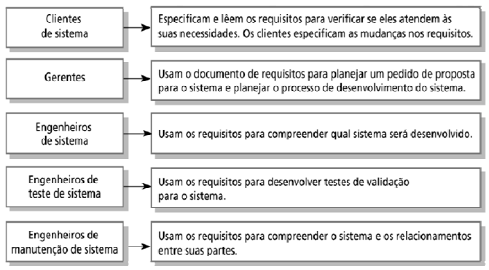
O documento de requisitos

- O documento de requisitos é a declaração oficial do que é requisitado pelos desenvolvedores do sistema.
- Deve incluir ambos, uma definição dos requisitos de usuário e uma especificação dos requisitos de sistema.
- **NÃO É** um documento de projeto.
  - Logo que possível, será preciso definir **como** o sistema deve fazer, ao invés de **o que** deve ser feito.

Usuários de um documento de requisitos

Figura 6.5

Usuários de um documento de requisitos.



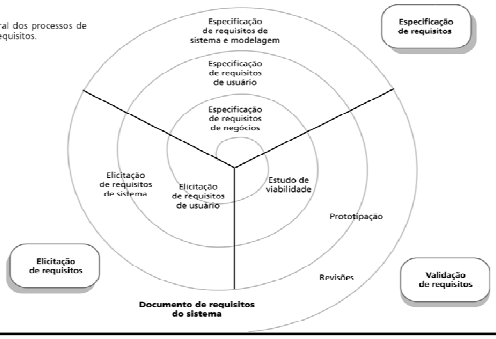
Processo de Engenharia de Requisitos

Processos de engenharia de requisitos

- Os requisitos e as formas de obtê-los e documentá-los variam drasticamente de um projeto para o outro
- Contudo, existe uma série de atividades genéricas comuns a todos os processos
  - Elicitação de requisitos;
  - Análise de requisitos;
  - Validação de requisitos;
  - Gerenciamento de requisitos.

Engenharia de requisitos

Figura 7.2
Modelo em espiral dos processos de engenharia de requisitos.

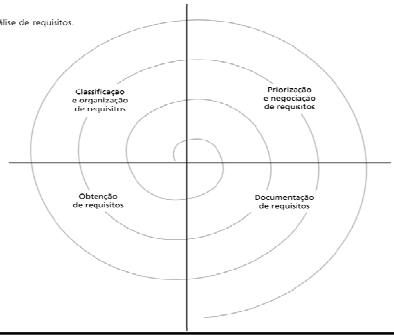


Elicitação e análise

- Também denominada de descoberta de requisitos
- Envolve pessoal técnico trabalhando com os clientes para descobrir sobre o domínio da aplicação, os serviços que o sistema deve fornecer e sobre as restrições operacionais.
- Pode envolver
  - Usuários finais
  - Gerentes
  - Engenheiros envolvidos na manutenção
  - especialistas de domínio
  - representantes de sindicato, etc.
- Estes são chamandos stakeholders *(partes interessadas)*

A espiral da elicitação e análise de requisitos

Figura 7.3
Processo de elicitação e análise de requisitos.



Atividades do processo de elicitação e análise

- Identificação (ou Elicitação) de requisitos
  - Interação com os stakeholders para coletar seus requisitos. Os requisitos de domínio são também descobertos neste estágio.
- Análise e Negociação de requisitos
  - Agrupa requisitos relacionados e organiza-os em conjuntos coerentes.
  - Priorização de requisitos e resolução de conflitos de requisitos.
- Documentação de requisitos
  - Os requisitos são documentados e colocados na próxima volta da espiral.

Identificação de requisitos

- Processo de reunir informações sobre os sistemas propostos e existentes
  - Obter requisitos de usuário e de sistema a partir dessas informações.
- As fontes de informação incluem documentação, stakeholders e as especificações de sistemas similares.
- Protótipos também podem ser usados tanto para descobrir quanto para validar requisitos.

Algumas Técnicas de Elicitação

- Entrevistas
- Questionários
- Brainstorming
- Casos de Uso

Problemas da análise de requisitos

- Stakeholders não sabem o que eles realmente querem.
- Stakeholders expressam requisitos em seus próprios termos.
- Diferentes stakeholders podem ter requisitos conflitantes.
- Fatores organizacionais e políticos podem influenciar os requisitos de sistema.
- Mudanças de requisitos durante o processo de análise

Resolução de Conflitos

- É normal que ocorram requisitos conflitantes
- Por exemplo
  - R-23: O sistema deve ...
  - R-45: O sistema não deve ...
- Cliente/usuário deve ser consultado para resolver conflitos (ambiguidades)

Atribuição de Prioridade

- Alguns requisitos são mais urgentes que outros
- É essencial determinar a prioridade dos requisitos junto ao cliente
- Requisitos de maior prioridade são considerados em primeiro lugar
- Requisitos podem ser vistos em três classes distintas
  - Essenciais
  - Importantes
  - Desejáveis
- Em princípio, sistema deve resolver todos os requisitos de essenciais para desejáveis

Exemplo de Prioridade

- [RF001] Consulta X ao B.D. deve retornar dados A, B, C
  - Prioridade: Essencial
- [RNF001] Consulta X ao B.D. deve visualizar dados segundo padrão Y
  - Prioridade: Importante
- [RNF010] Consulta X ao B.D. deve usar cores azuis nos resultados
  - Prioridade: Desejável

Validação de requisitos

- Dedica-se a mostrar que os requisitos definem o sistema que o cliente realmente deseja.
- Custos de erros de requisitos são altos e, desse modo, a validação é muito importante
  - O custo da reparação de um erro de requisitos depois da entrega pode equivaler a muitas vezes o custo de reparação de um erro de implementação

Técnicas de validação de requisitos

- Revisões de requisitos
  - Análise manual sistemática dos requisitos.
  - Potencialmente acompanhada por *stakeholders*
- Prototipação
  - Uso de um modelo executável do sistema para verificar requisitos
- Geração de casos de teste.
  - Desenvolvimento de testes para requisitos a fim de verificar a testabilidade
  - Testes de aceitação

Verificação de requisitos

- Verificação de validade. O sistema fornece as funções que melhor apóiam as necessidades do cliente?
- Verificação de consistência. Existe algum tipo de conflito de requisitos? Para um mesmo requisito não pode haver contradição
- Verificação de completude. Todas as funções requisitadas pelo cliente foram incluídas?
- Verificação de exequibilidade. Os requisitos podem ser implementados com o orçamento e a tecnologia disponíveis
- Facilidade de verificação. Os requisitos podem ser verificados? Usar conjunto de testes para demonstrar que a funcionalidade entregue atende o requisito

Gerenciamento de requisitos

- Gerenciamento de requisitos é um processo para compreender e controlar as mudanças de requisitos
- Requisitos são, inevitavelmente, incompletos e inconsistentes
  - Novos requisitos surgem durante o processo inteiro
  - Diferentes *stakeholders* têm requisitos diferentes e estes são freqüentemente contraditórios.

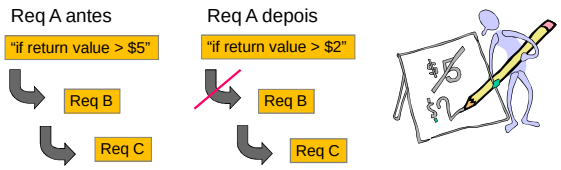
Gerenciamento de requisitos

- Deve ser aplicado a todas as mudanças propostas aos requisitos
  - Especialmente importante para sistemas já prontos ou em estágios avançados de desenvolvimento
- Estágios principais
  - Análise de problema: discutir problemas e mudanças de requisitos;
  - Análise de mudança e estimativa de custo: avaliar os efeitos das mudanças sobre outros requisitos;
  - Implementação de mudança: Modificar vários artefatos para refletir as mudanças.
- O impacto da mudança tem que ser avaliado para TODO O SISTEMA!

Rastreabilidade

- A rastreabilidade tem a ver com relacionamentos entre os requisitos, suas fontes e o projeto do sistema
  - É necessário manter essa informação registrada nos locais apropriados
- Rastreabilidade da fonte
  - Ligam requisitos aos stakeholders que os propuseram ou aos elementos externos que o criaram;
- Rastreabilidade de requisitos
  - É a ligação dos requisitos dependentes;
- Rastreabilidade de projeto
  - Ligações entre os requisitos e os módulos de projeto.

Rastreabilidade: Análise de Impacto



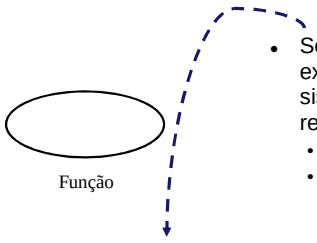
- Links dos requisitos modificados devem ser marcados para "revisar"
- Links marcados para "revisar" devem ser analisados

Diagramas de Casos de Uso

Objetivos

- Introduzir conceitos de *use case*, ator e fluxo de eventos
- Apresentar sub-fluxos de eventos
- Discutir sobre identificação, evolução e organização de use cases
- Apresentar notação UML para reusar atores e use cases

Use Case



- Sequência de ações, executada pelo sistema, que gera um resultado
  - De valor observável
  - E para ator particular

Use Case e Ator

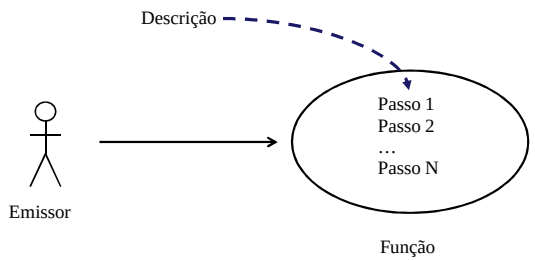

Emissor/Receptor

- Alguém ou alguma coisa (**fora do sistema**) que interage com o sistema

Use Case e Ator

- A descrição de um *use case* define o que o sistema faz quando o *use case* é realizado
- A funcionalidade do sistema é definida por um conjunto de *use cases*, cada um representando um fluxo de eventos específico

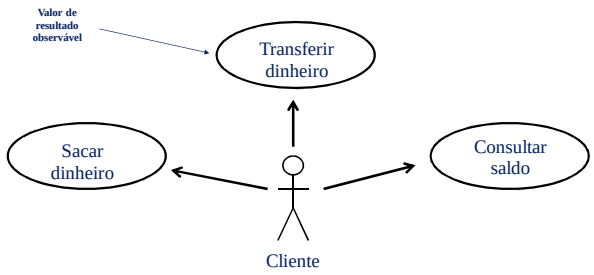
Use Case e Ator



Exemplo de *Use Case* e Ator

- Cliente de banco pode usar um caixa automático para
  - sacar dinheiro, transferir dinheiro ou consultar o saldo da conta
- Ator: Cliente
- *Use cases*: Sacar dinheiro, transferir dinheiro e consultar saldo

Exemplo de *Use Case* e Ator



Identificando *Use Cases*

- Em geral, é difícil decidir entre um ou vários use cases. Por exemplo, seriam use cases
  - Inserir cartão em um Caixa Automático?
  - Entrar com a senha?
  - Receber o cartão de volta?
- Use cases representam valor observável para ator. Pode-se determinar
  - A partir de interações (sequencia de ações) com o sistema que resultam valores para atores
  - Quando satisfazem um objetivo particular de um ator que o sistema deve prover

Reuso em *Use Cases*

- Comportamento comum a mais de dois *use cases* (ou que constitui parte independente)
  - Pode-se modelar como *use case* para ser reusado
- Há três possibilidades
  - Inclusão
  - Extensão
  - Generalização/Especialização

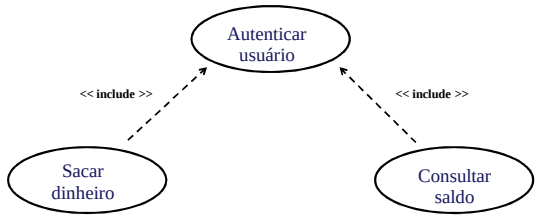
Inclusão

- Vários use cases possuem texto de fluxo de eventos
  - Comum/idêntico
  - Sempre usado
- Equivalente a fatoração feita em programação através de sub-programas
  - #include da linguagem C

Inclusão

- Como exemplo, tanto “Sacar dinheiro” quanto “Consultar saldo” necessitam da senha
  - Pode-se criar novo use case “Autenticar usuário” e incluí-lo
- Mas atenção
  - NÃO SE DEVE CRIAR USE CASES MÍNIMOS
  - Deve haver ganho no reuso

Inclusão



Inclusão

- Descrição de Consultar saldo
  - Fluxo de Eventos Principal:
    - include (Autenticar usuário). Sistema pede a Cliente que selecione tipo de conta (corrente, etc.). ...

Extensão

- Use case pode ser estendido por outro
  - Extensão de funcionalidade/Caso excepcional
- Extensão ocorre em pontos específicos
  - Pontos de extensão

Extensão

- Há também extensão de texto (fluxo de eventos)
  - Porém sob condições particulares
- Pode ser usada para
  - Simplificar fluxos de eventos complexos
  - Representar comportamentos opcionais
  - Lidar com exceções

Extensão

Pontos de extensão urgente

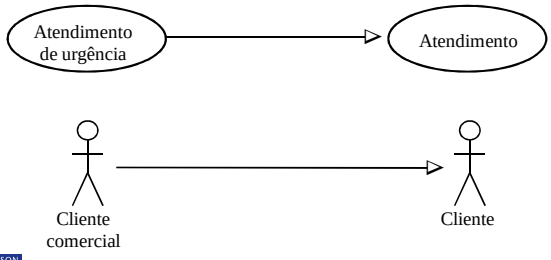
Extensão

- Descrição de Atendimento
  - Fluxo de Eventos Principal:
    - Colete os itens do pedido. (urgente). Submeta pedido para processamento.

Especialização

- *Use case* pode especializar outro
  - Adição/refinamento do fluxo de eventos original
- Especialização permite modelar comportamento de estruturas de aplicação em comum

Especialização



Fluxo de Eventos

- Parte mais importante de um use case
- Define a sequencia de ações entre o ator e o sistema
- Geralmente escrito em linguagem natural
  - Uso preciso de termos definidos no glossário de acordo com o domínio do problema
- Também pode ser expresso formalmente
  - Pré e pós-condições (ou pseudo-código)

Exemplo de Fluxo de Eventos

- Um esboço inicial sobre Sacar dinheiro seria
  1. O use case inicia quando o Cliente insere um cartão no CA. Sistema lê e valida informação do cartão
  2. Sistema pede a senha. Cliente entra com a senha. Sistema valida a senha.
  3. Sistema pede seleção do serviço. Cliente escolhe “Sacar dinheiro”
  4. Sistema pede a quantia a sacar. Cliente informa.
  5. Sistema pede seleção da conta (corrente, etc). Cliente informa.
  6. Sistema comunica com a rede para validar a conta, senha e o valor a sacar.
  7. Sistema pede remoção do cartão. Cliente remove.
  8. Sistema entrega quantia solicitada.

Fluxo de Eventos

- Na descrição do que o sistema faz através de fluxos de eventos completos
  - Surgem caminhos alternativos
  - Casos diferentes a considerar
  - Efeitos/valores diferentes a produzir
- Eventualmente descreve todos esses caminhos possíveis

Sub-fluxos de Eventos

- Fluxo de eventos visto como
  - Vários sub-fluxos de eventos
- Sub-fluxos são descritos como
  - Principal
  - Alternativos/excepcionais
- Abordagem visa reuso de fluxos de eventos (sub-fluxos)

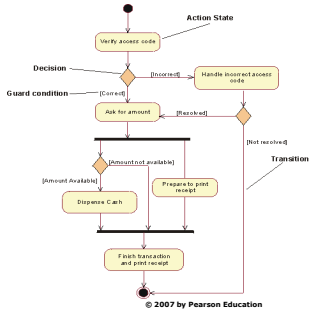
Exemplo de Sub-fluxos

- Seja o *use case* Validar usuário
  - Fluxo principal:
    - O *use case* inicia quando o sistema pede ao Cliente a senha. Cliente entra com senha. Sistema verifica se a senha é válida. Se a senha é válida, sistema confirma e termina o *use case*.
  - Fluxo excepcional:
    - Cliente pode cancelar a transação a qualquer momento pressionando a tecla ESC, reiniciando o *use case*. Nenhuma modificação é feita na conta do Cliente.
  - Fluxo excepcional:
    - Se Cliente entra com senha inválida, o *use case* reinicia.

Diagrama de Atividades

- Descreve aspectos dinâmicos de um sistema através de fluxo de tarefas
- Alternativa para modelar fluxos de eventos de casos de uso

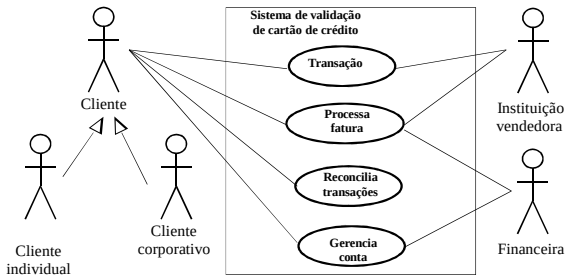
Diagrama de atividades: exemplo



Diagramas de *Use Cases*

- *Use cases* são organizados dentro de um diagrama que caracteriza limites da funcionalidade do sistema
- Em diagramas de *use cases*
  - Atores são relacionados por generalização/especialização

Exemplo de Diagrama



Alguns casos de uso do LIBSYS

Figura 7.6

Casos de uso para o sistema de biblioteca.