Feature-Oriented Domain Analysis (FODA) Method

Modelagem de Domínio - Feature Models

Luciano Marchezan | Fernando Lima

Roteiro

- Introdução
  - Análise de Domínio
  - Feature
  - FODA
    - Análise de Contexto
    - Modelagem de Domínio
- Modelagem de Domínio
  - Análise de *Feature*
  - Modelagem Entidade-Relacionamento
  - Análise Funcional
- Conclusão

Introdução

Análise de Dominio

- A descoberta sistemática de semelhanças entre softwares é um **requisito chave** para se obter reuso de software.
- Uma técnica que pode ser usada para se alcançar esse requisito é a **Análise de domínio.**

Análise de Dominio

"A análise de domínio é a **exploração sistemática** de softwares que busca definir e explorar **pontos em comum** entre eles." [Kang et al. 1990].

"Um processo pelo qual a informação utilizada no desenvolvimento de sistemas de software é **identificada, capturada, e organizada** com o objetivo de torná-la **reutilizável** durante a criação de novos sistemas." [Prieto-Díaz 1990]

A análise de domínio define as ***features*** e as **capacidades** de uma classe de sistemas de softwares relacionados.

Feature

- Um **aspecto**, **qualidade**, ou **característica** de um ou mais sistemas de software que é **visível ao usuário** [American 85].

- São as **propriedades** de um sistema que **afetam diretamente** os **usuários** finais [Czarnecki and Eisenecker 2000].

Análise de Domínio orientada a *Feature* (FODA)

- A partir da análise de domínio são produzidos **produtos de domínio.**
- Estes, representam **funcionalidades e arquiteturas** que aplicações em um domínio possuem em **comum.**
- O desenvolvimento de produtos de domínio que sejam genéricos e amplamente aplicáveis dentro de um domínio é o principal objetivo do método FODA.

FODA é um método utilizado para descobrir e representar pontos em comum entre sistemas de software relacionados.

Análise de Domínio orientada a *Feature* (FODA)

Para apoiar o desenvolvimento e reutilização de "abstrações", este método baseia-se num conjunto de conceitos de modelagem:

- agregação / decomposição
- generalização / especialização
- parametrização

Agregação - Decomposição

- Agregar - Abstrair uma coleção de unidades em uma nova unidade. Exemplo: escola é uma agregação de alunos, professores, etc;
- Decompor - Refinar uma agregação em suas unidades constituintes.

Essa modelagem primitiva permite composição de componentes em um novo componente agregado ou decomposição de um componente abstrato em suas partes

Generalização - Especialização

- Generalizar - Abstrair os pontos em comum entre uma coleção de unidades em uma nova unidade conceitual suprimindo diferenças.
- Especializar - Refinar uma unidade generalizada em uma nova unidade incorporando detalhes. Por exemplo, a entidade "empregado" é uma abstração de secretários, gerentes, etc.

Essa modelagem primitiva permite o desenvolvimento de componentes genéricos que podem ser refinados de muitas maneiras diferentes.

Parametrização

É uma técnica de desenvolvimento de componentes em que os componentes são adaptados de maneiras diferentes, substituindo os valores de parâmetros que são incorporados no componente.

Permite a codificação de como a adaptação é feita dentro de um componente.

Método FODA

O método FODA consiste de duas fases:

- Análise de contexto: definir os limites do domínio
- Modelagem de domínio: produzir o modelo de domínio

FODA - Análise de Contexto

- Define o escopo de um domínio que provavelmente produzirá produtos de domínio exploráveis.

- Ao final da análise é gerado um modelo de contexto. Este modelo define o **limite do domínio**.

FODA - Modelagem de Domínio

- A partir do domínio definido, **semelhanças e diferenças** são analisadas.

- São produzidos modelos que representam **diferentes aspectos** dos problemas do domínio.

- A modelagem de domínio consiste em 3 atividades principais: análise de *feature*, modelagem entidade-relacionamento e análise funcional.

Modelagem de Domínio – Análise de Feature

Análise de *Feature*

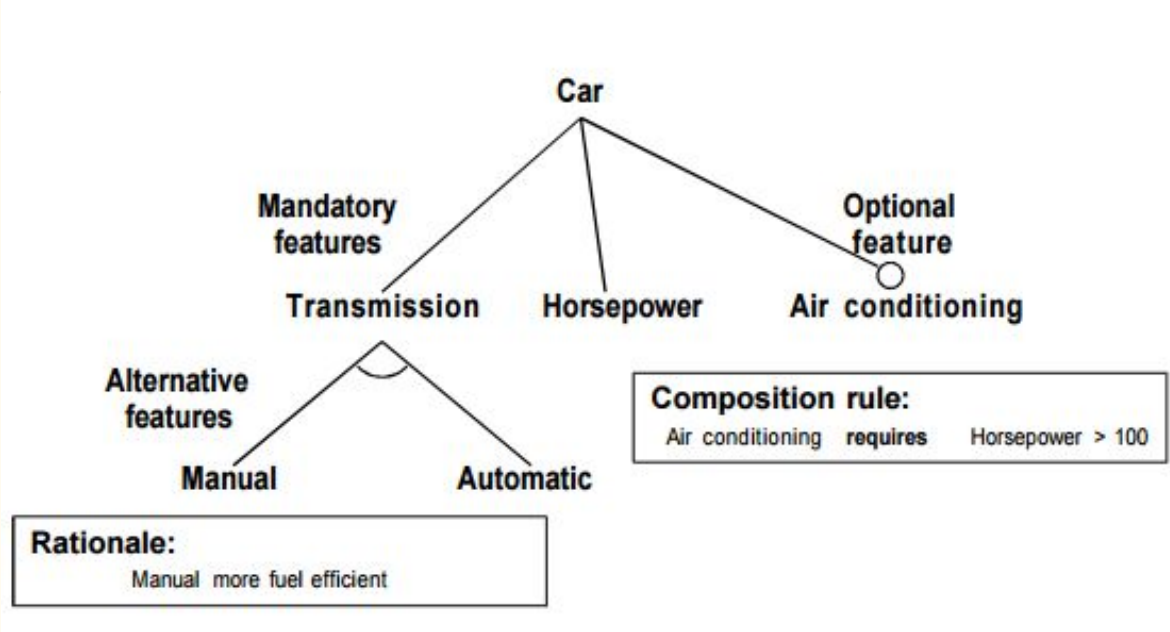
- Objetiva **capturar** em um **modelo** o entendimento dos usuários-finais sobre as capacidades gerais de aplicações em um domínio.
- Estas capacidades podem incluir:
  - 1 - serviços fornecidos pela aplicação; 2 - desempenho da aplicação; 3 - plataforma de hardware necessária; 4 - custos.
- O *feature model* deve capturar as **características comuns** e as **diferenças** das aplicações no domínio.

Análise de *Feature* - Processos e Guidelines

- O processo de Análise de *Feature* consiste em:
    - Reunião dos documentos fontes
    - Identifição das *Features*
    - Abstração e Classificação das *Features* identificadas como um modelo
    - Definição das *Features*
    - Validação do modelo.

Descrição do modelo

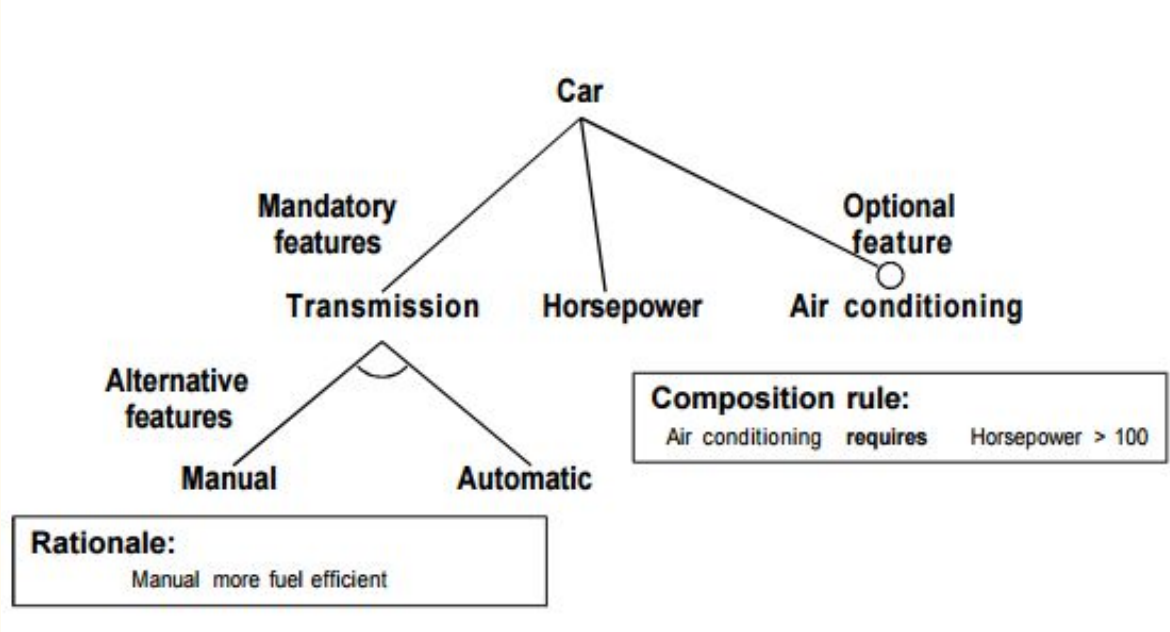
- Lembrando: *Features* são os atributos ou propriedades de um sistema que afetam diretamente os usuários finais.
- O diagrama possui a forma de árvore, onde a raiz representa o conceito sendo descrito os os demais nós são as *features*.



Example Showing Features of a Car

Descrição do modelo - Tipos de *features:*

- *Features* mandatórias: são as que todas as aplicações daquele domínio **devem** ter.
- *Features* alternativas: a aplicação só pode possuir **uma por vez**.
- *Features* opcionais: a aplicação **pode ou não** possui-lás.

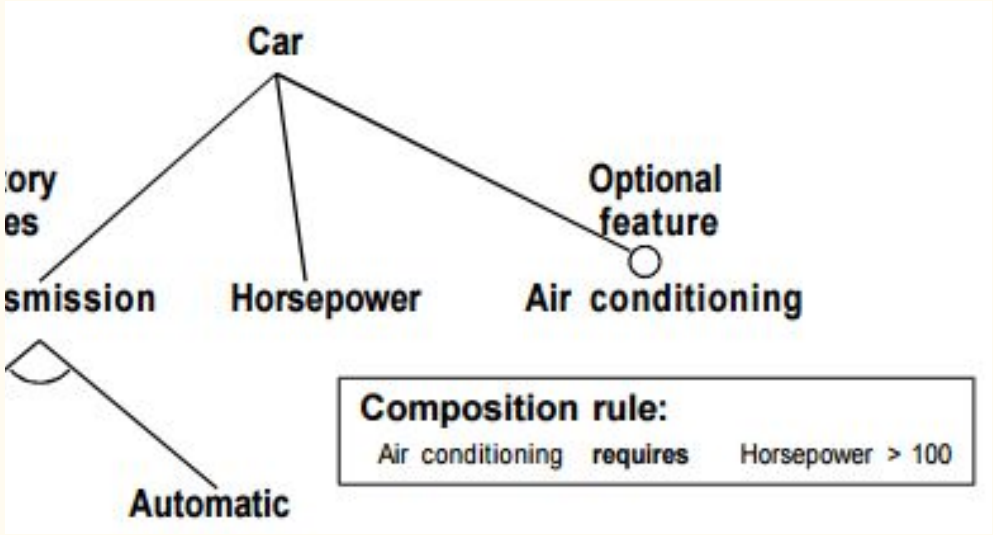


Descrição do modelo - Interdependências

As interdependências das *features* são capturada usando regras de composição:

- Regras obrigatórias: capturam **implicações** entre duas ou mais *features*.

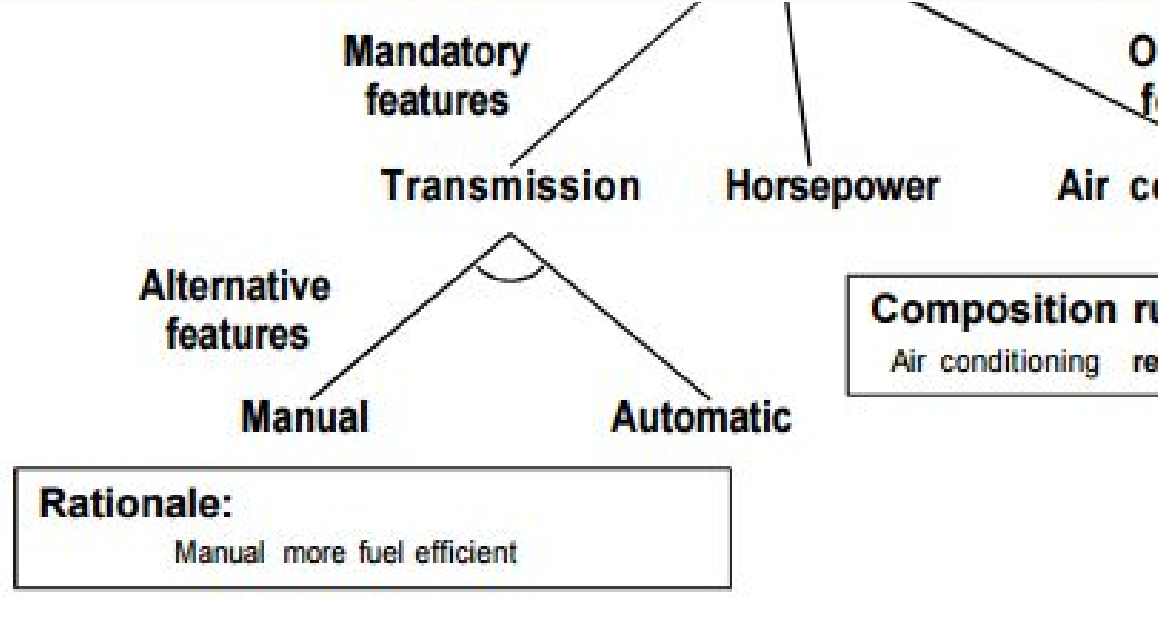
- Regras mutuamente exclusivas: modelam **restrições** em **combinações** de *features*.



Descrição do modelo - Rationales

- Uma rationale demonstra os motivos por selecionar ou não uma *feature*.

- Geralmente são usadas para representar uma *feature* de forma **informal**.



Descrição do modelo - Propósito das *features*

- *Features* de contexto: descrevem o **objetivo geral** de uma aplicação. Representam questões como os requisitos de **desempenho**, **precisão** e **sincronização de tempo** que afetam as operações da aplicação.

- *Features* de representação: descrevem como a informação é **visualizada** pelo o **usuário** ou **produzida** para outra **aplicação**.

- *Features* operacionais: descrevem as **funções** ativas realizadas, ou seja, o que a aplicação **faz**.

Descrição do modelo - Classificação das *features*

- *Features* em tempo de compilação: resultam de diferentes pacotes do software e são processadas em tempo de compilação. Por exemplo: *features* de diferentes aplicações, mas da mesma família.

- *Features* em tempo de ativação (tempo de carregamento): são aquelas selecionadas no início da execução e permanecem estáveis durante a execução. A instanciação é feita fornecendo valores no início de cada execução.

- *Features* em tempo de execução: podem ser modificadas manualmente ou automaticamente durante a execução.

Resumindo...

Modelos de *features* do método FODA consistem de 4 elementos chave:

1. Diagrama de *feature (Feature Model)* - representação de uma decomposição hierárquica de *features* incluindo a indicação de quando uma *feature* é ou não obrigatória, alternativa, ou opcional.

2. Definição de *feature* para todas as *features*, incluindo a indicação de quando uma *feature* é em tempo de compilação, ativação ou execução.

3. Regras de composição.

4. Rationale para *features* mostrando indicadores.

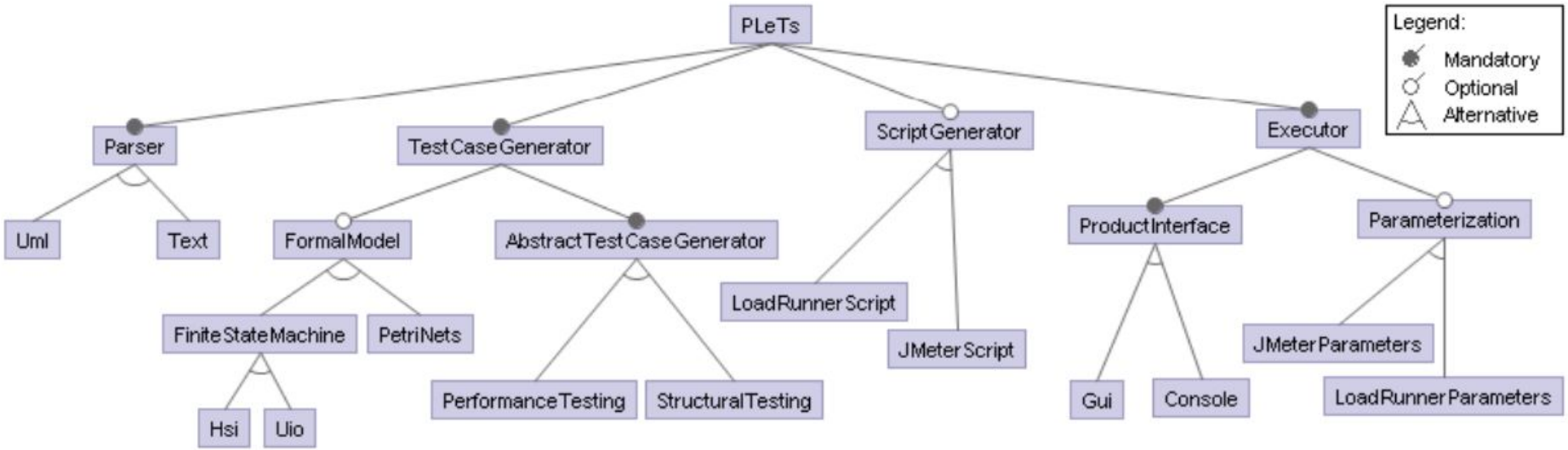


Figure 2: The PLeTs Feature Model [7].

Uso do Modelo

- *Feature Model* serve de mediador entre o usuário e o desenvolvedor

    - Para usuário: apresenta o que são padrões de *Features*, quais outras *Features* eles podem escolher e quando podem escolhê-las.

    - Para desenvolvedor: o modelo de *feature* indica quais as necessidades para ser parametrizadas nos outros modelos e arquiteturas de software, e como a parametrização deve ser feita.

Validação do Modelo

- O modelo precisa ser validado por especialistas do domínio.

- Os especialistas que foram consultados durante a análise **não devem** ser selecionados para a validação.

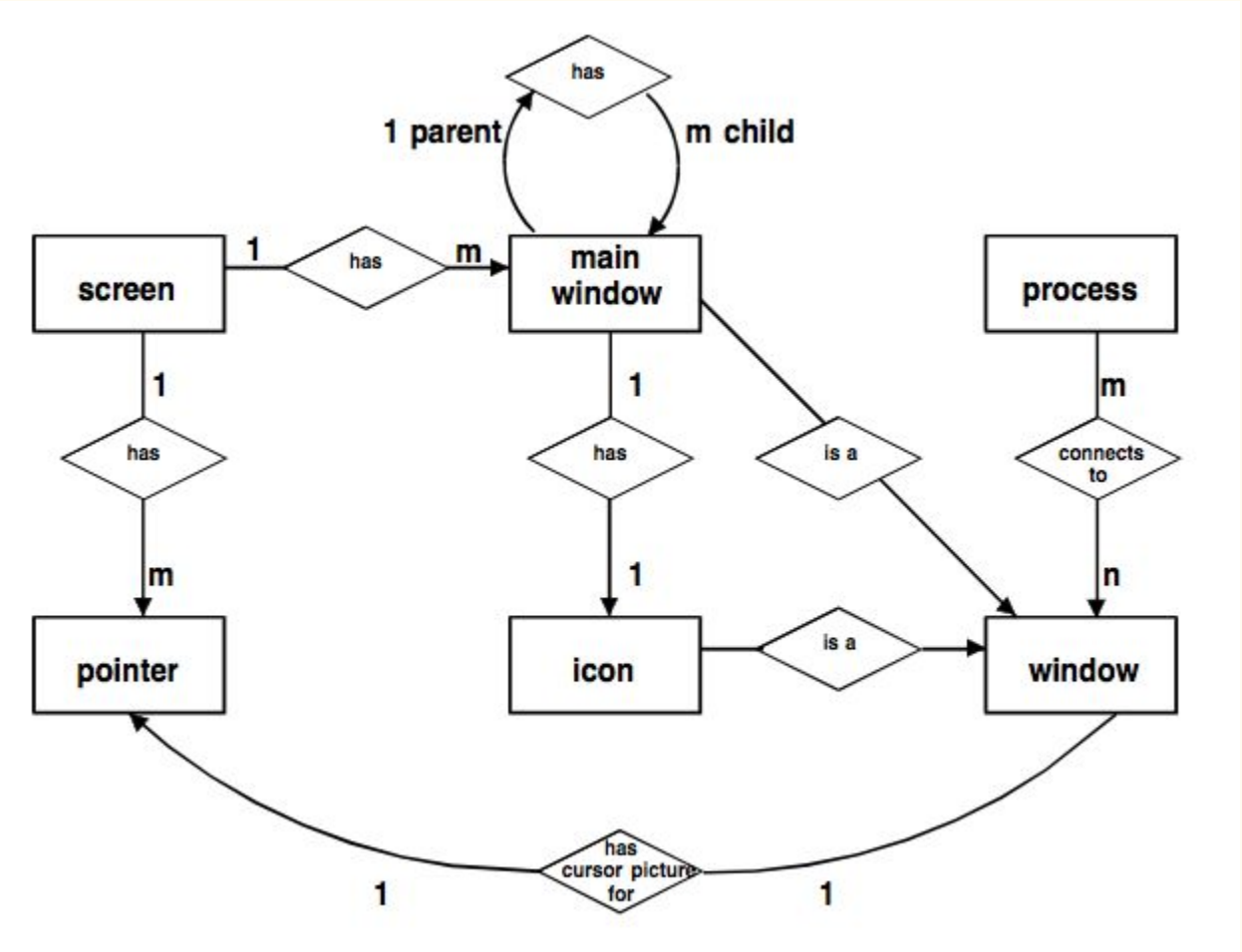
Modelagem de Domínio - Modelagem Entidade-Relacionamento

Modelagem Entidade-Relacionamento

- A modelagem entidade-relacionamento captura e define o conhecimento do domínio que é essencial para a implementação de aplicações no domínio
- Tem como propósito:
  - representar o conhecimento do domínio explicitamente em termos de entidades do domínio e seus relacionamentos
  - fazê-los disponíveis para a derivação de objetos e definição de dados durante a análise funcional e modelagem da arquitetura

Descrição do Modelo

- É uma adaptação do método de Chen, *Semantic Data Modelling*.
- São usados notações e métodos do Chen, porém, há o acréscimo da generalização e agregação
- Os blocos básicos de construção são as classes entidades e os relacionamentos *is-a* e *consist-of*
  - Classes entidades representam abstrações dos objetos do domínio da aplicação;
  - *Is-a* (generalização) especifica as semelhanças e diferenças entre entidades;
  - *Consist-of* (agregação) especifica a estrutura de composição entre entidades;



- Exemplo de notação do diagrama de entidade-relacionamento

- Outras informações pertinentes para cada entidade podem ser descritas de forma textual

A.3. Attribute Description Form

```
Attribute: <attribute name>

Synonyms: <synonyms>

Description:
<a textual description of the attribute>

Value range:
<value range specification> [ <unit name>]

Source:
<information source>
```

The source of information (e.g., standards, text
systems) from which the feature is derived is in

A value range can be any combination of:
(1) value types such as integer, string, r
(2) range of values (e.g., 10 through 10
(3) strings (e.g., South, North)

Examples of the <unit name> are days, p

A.4.1. Relationship Type *is-a* Form

```
Relationship Type is-a ;

Description:
        To describe generalization/specialization relationships
        between entities. An entity in a generalization/
        specialization hierarchy inherits all of the attributes
        of its generalization entity. The value of an inherited
        attribute may be modified as long as the modified value
        is within the range of its generalization entity's value.
        A specialization entity may have attributes that are not
        defined for its generalization entity.;

Parts:  generalization  ANY
        specialization  ANY;
```

A.4.2. Relationship Type *consists-of* Form

```
Relationship Type consists-of;

Description:
To specify an aggregation relationship between an
entity and its constituent parts.;

Parts:  whole ANY
        parts ANY
        how-many INTEGER;

Connectivity:
        whole MANY
        parts MANY
        how-many ONE;

Syntax:
        whole CONSISTS OF [how-many] parts;
        parts IS A PART OF whole;
```

```
eneralization;
ENERALIZATION-OF specialization;
```

Uso do Modelo

- Usos para o Modelo de Entidade-Relacionamento:
  - Dar suporte a análise e entendimento de problemas do domínio
  - Derivar e estruturar objetos do domínio usados no desenvolvimento da aplicação
- As entidades são unidades de informação de domínio que devem ser processadas e/ou mantidas pelo sistema
- Os relacionamentos são usados para identificar as composições de objetos do domínio, semelhanças e diferenças entre objetos de domínio

Modelagem ER - Processos e Guidelines

- A construção de um sistema:
  - Identificando as entidade e seus relacionamentos
  - Nomear as entidade e seus relacionamentos
  - Cada entidade é caracterizada por propriedades, das quais alguma pode ser usada como identificador
- Na nomeação, costuma-se usar substantivos, verbos e adjetivos em Inglês.
- As entidade dentro do modelo são classificadas dentro de conjuntos homogêneos, ou seja, as entidades na mesma classe possuem propriedade em comum

Modelagem de Domínio
-
Análise Funcional

Análise Funcional

- Identifica as semelhanças e diferenças funcionais das aplicações no domínio
- O modelo de *feature* e modelo entidade-relacionamento são usados como *guidelines* no desenvolvimento do modelo funcional
- As *features* mandatórias e as entidades são as bases para definir modelo funcional abstrato
- As *features* alternativas ou opcionais são incorporadas no modelo durante o refinamento

Descrição do Modelo

As especificações de um modelo funcional podem ser classificadas em duas categorias:

- Especificações de funções: descrevem o aspecto estrutural de uma aplicação em termos de entradas, saídas, atividades, dados internos, estruturas lógicas e relações de fluxo de dados entre eles.

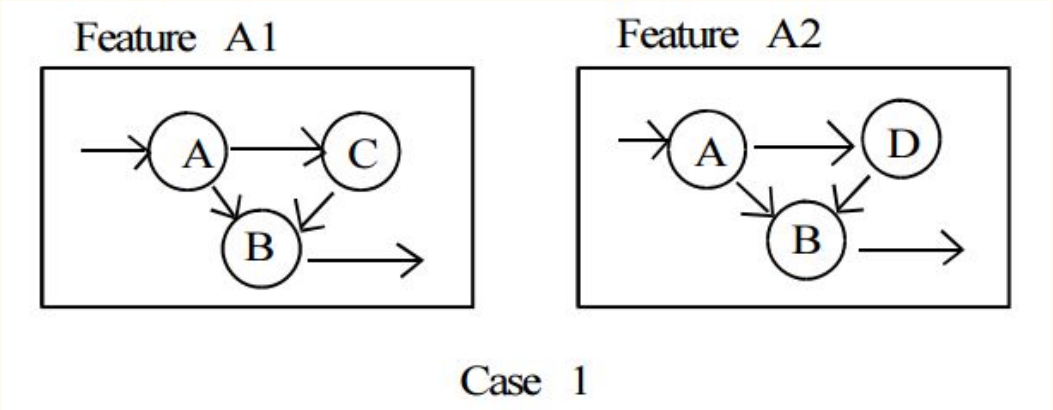
- Especificações de comportamentos: descrevem como a aplicação se comporta em termos de eventos, entradas, estados, condições e transições entre estados.

Um diferencial do método FODA em comparação com outros métodos de análise de domínio é a **parametrização** através do uso de *features* e *issues/decisions*.

Conforme um modelo abstrato é refinado, *features* alternativas e opcionais são incorporadas no modelo. Quaisquer *issues/decisions* levantadas durante a análise também são incorporadas no modelo.

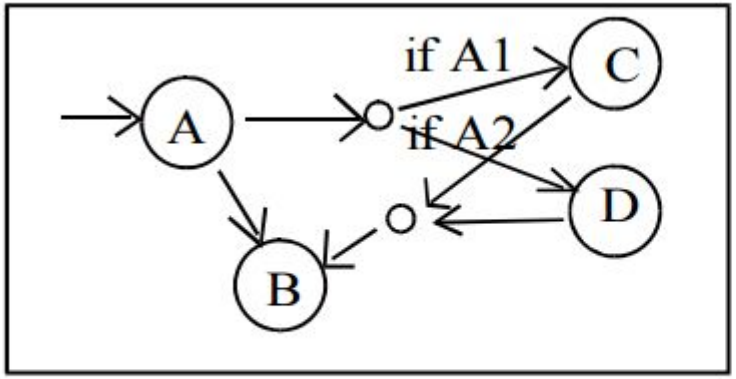
Há 3 maneiras de incorporar as *features* e *issues*/ no modelo:

1. Através do desenvolvimento de componentes separados (refinamentos) para cada alternativa,



Há 3 maneiras de incorporar as *features* e *issues*/ no modelo:

2. Através do desenvolvimento de um componente, mas com parametrização para se adaptar a cada alternativa



or

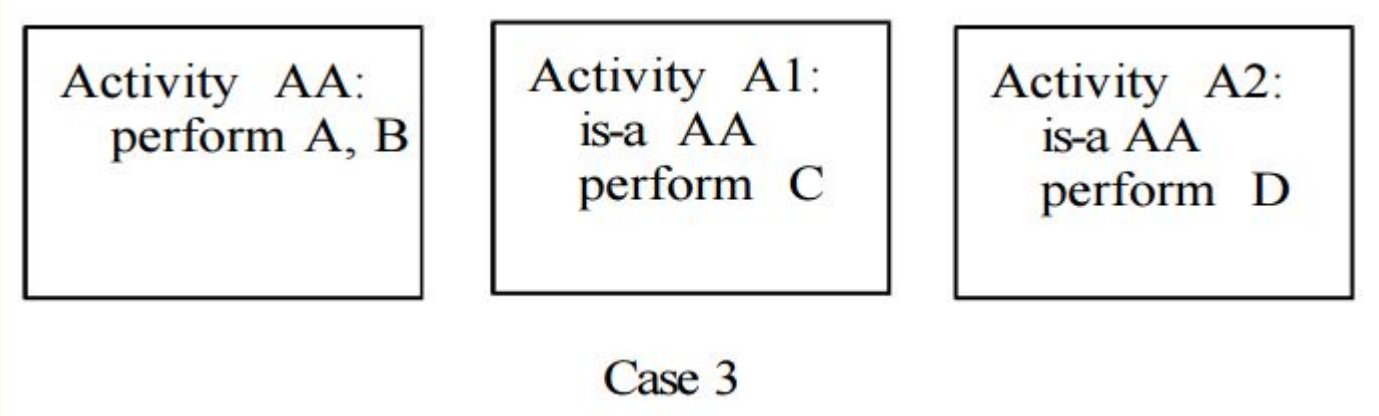
Activity AA:

perform A,B
if A1 perform C
if A2 perform D

Case 2

Há 3 maneiras de incorporar as *features* e *issues*/ no modelo:

3. Ao definir um componente geral e desenvolver cada alternativa como uma instanciação deste componente.



Uso do modelo

- Entender os problemas de domínio
- Reusar modelos na análise de requisitos

Concluindo

Pontos importantes

- A descoberta sistemática de semelhanças entre softwares é um **requisito chave** para se obter reuso de software. Ela pode ser feita através da análise de domínio.

- Uma *feature* é um **aspecto**, **qualidade**, ou **característica** de um ou mais sistemas de software que é **visível ao usuário**.

- FODA é um método utilizado para **descobrir** e **representar** pontos em comum entre sistemas de software relacionados.

- O FODA é baseado em agregação / decomposição; generalização / especialização; e parametrização.

Pontos importantes

- O *feature model* deve capturar as **características comuns** e as **diferenças** das aplicações no domínio.
- *Feature models* representam uma **decomposição hierárquica** de *features* incluindo a indicação de quando uma *feature* é ou **não obrigatória**, **alternativa**, ou **opcional**.
- Uma *feature* pode ser classificada como: em tempo de **compilação**, **ativação** ou **execução**.
- As interdependências das *features* são capturada usando **regras de composição**.

- Uma rationale demonstra os motivos por **selecionar ou não** uma *feature*.

Próxima missão

- Atlas

Referências

Kang, Kyo C., et al. *Feature-oriented domain analysis (FODA) feasibility study*. No. CMU/SEI-90-TR-21. Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, 1990.

Czarnecki, Krzysztof. "Generative programming: Principles and techniques of software engineering based on automated configuration and fragment-based component models." (1998).

Prieto-Díaz, Rubén. "Domain analysis: An introduction." *ACM SIGSOFT Software Engineering Notes* 15.2 (1990): 47-54.