Feature Model

Czarnecki Notation - FeatuRESB

Luciano Marchezan | Fernando Lima

Lembrando...

- Análise de domínio
  - Técnica para descoberta sistemática de semelhanças entre softwares
- *Feature*
  - Um aspecto, qualidade, ou característica de um ou mais sistemas de software que é visível ao usuário [American 85].
- Método FODA
  - Método utilizado para descobrir e representar pontos em comum entre sistemas de software relacionados.

Roteiro

- FeatuRSEB
- Notação de Czanercki [2000]
- Notação de Czanercki [2005]

FeatuRSEB

Reuse-Driven Software Engineering Business - RSEB

- É uma abordagem Model-Driven e sistemática para reuso em software de larga escala
- É usado Unified Modeling Language (UML), com extensões para:
  - Modelar e especificar sistemas de aplicação
  - Modelar e especificar componentes reusáveis do sistema e arquiteturas em camadas
  - Para expressar a variabilidade em termos de pontos de variação e variantes anexadas
- RSEB define vários processos de desenvolvimento de software model-driven:
  - Engenharia de Família de Arquiteturas (desenvolve uma arquitetura em camadas)
  - Engenharia de Sistema de Componentes (desenvolve um sistema de componentes reusáveis)
  - Engenharia de Sistema de Aplicações (desenvolve as aplicações selecionadas)

FODAcom

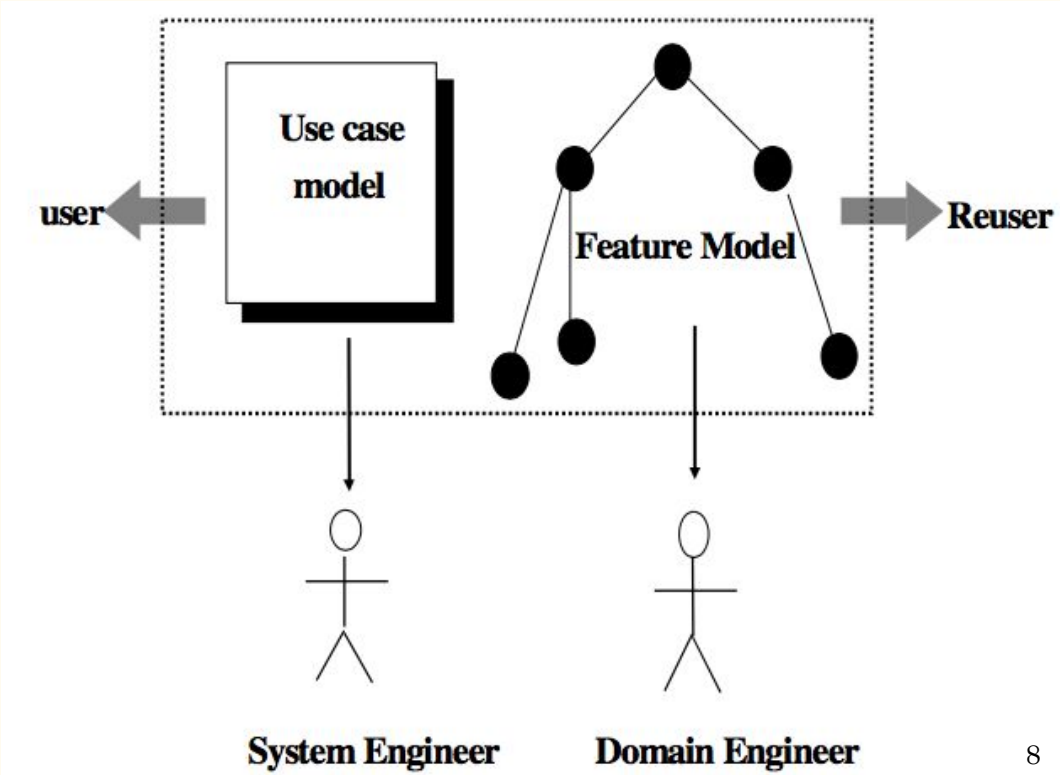
- Telecom Italia lançou o projeto FODAcom nos anos 90 sobre a liderança da Telesoft S.p.A. para criar uma versão customizada do FODA para aplicações no domínio da telecomunicação
- Contatos com Hewlett-Packard levaram a uma introdução precoce de vários conceitos chave do RSEB no FODAcom
- O que resultou no uso principal do modelo de feature como uma síntese concisa da variabilidade e semelhanças representadas em outros modelos RSEB

Featuring RSEB

- FODA é bem compatível com a RSEB,pois é uma abordagem model-driven, com as informações de domínio capturadas em diferentes modelos que refletem diferentes pontos de vista do domínio
- Experiências com FODAcom mostraram como o RSEB é incompleto com respeito a análise de domínio
- Diferente dos métodos de engenharia de domínio como FODA, o RSEB não fornece modelos de features explícitos ou um guia de como construir e transformar tais modelos de features

Features vs. Casos de Uso

- O modelo de caso de uso e modelo de feature tem propósitos distintos
- Modelos de caso de uso:
  - É orientado a usuário
  - Fornece o "o que" do domínio: uma descrição completa do o que o sistema faz no domínio
- O modelo de feature:
  - É orientado a reusuário
  - Fornece "quais" do domínio: quais funcionalidades podem ser selecionadas no momento da engenharia do novo sistema do

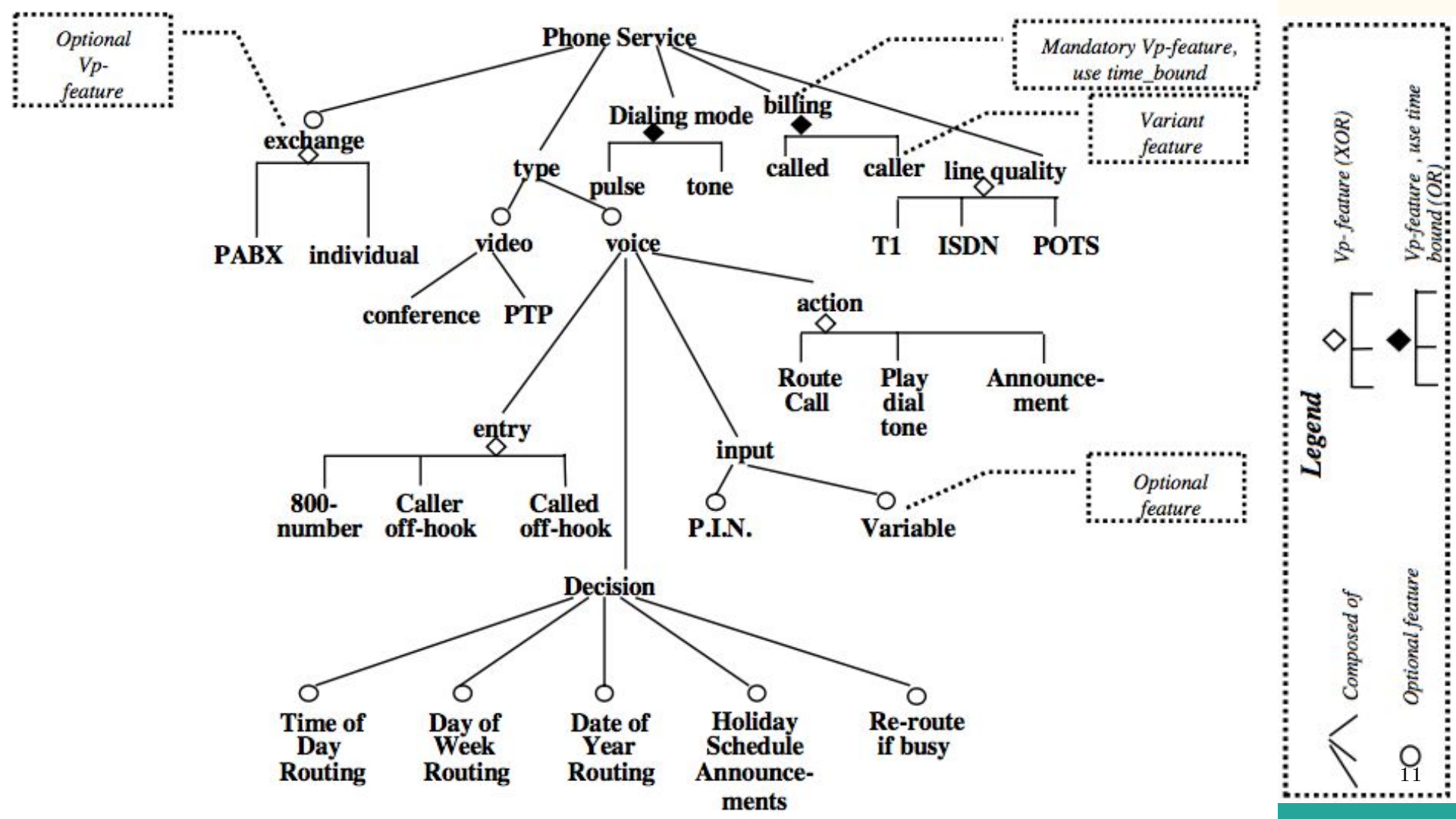


Features vs. Casos de Uso

- Os Casos de uso agora reúnem e descrevem requisitos do usuário sobre as funcionalidades do sistema, feito pelo engenheiro do sistema
- Os modelos de feature organizam os resultados das análises de semelhanças e variabilidades para a preparação para o engenheiro e acesso ao reuso, realizado pelo engenheiro de domínio

Notação Para Features

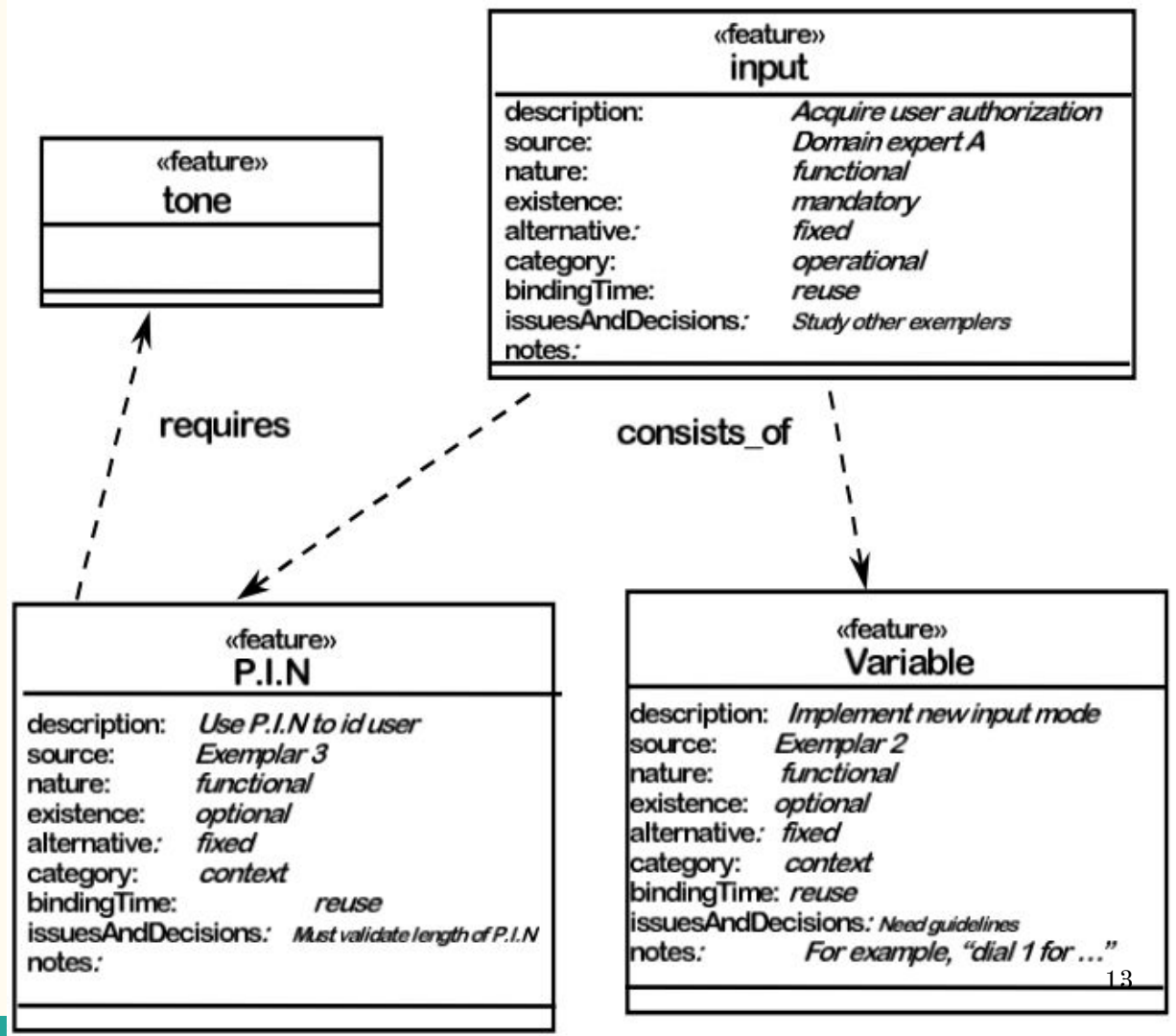
- Tipos de Feature:
  - **Mandatória**: features correspondentes a um centro de capacidades incorporadas as características do domínio principal no nível do problema e constitui a “infraestrutura” do domínio
  - **Opcional**: features correspondentes a identificação de algumas capacidades ou características as quais podem ser desnecessárias em alguns sistemas do domínio
  - **Variante**: features correspondentes a caminhos alternativos para configurar uma feature mandatória ou alternativa


Optional Vp-feature
Phone Service
Mandatory Vp-feature, use time_bound
Variant feature
exchange
Dialing mode
billing
type
pulse
tone
called
caller
line quality
PABX
individual
video
voice
T1
ISDN
POTS
conference
PTP
action
Route Call
Play dial tone
Announce-ment
entry
input
800-number
Caller off-hook
Called off-hook
P.I.N.
Variable
Optional feature
Decision
Time of Day Routing
Day of Week Routing
Date of Year Routing
Holiday Schedule Announce-ments
Re-route if busy
Legend
Vp-feature (XOR)
Vp-feature , use time bound (OR)
Composed of
Optional feature

Notação Para Features

- O relacionamento **composed_of**: Uma feature pode ser modelada como composta por várias sub-features, seguindo uma abstração do mecanismo de decomposição /agregação
- O atributo de existência: Uma feature pode ser **mandatória** ou **opcional.**
- O relacionamento de alternativa: **variação** e **variantes de features.** Uma feature pode agir como um ponto de variação (chamada feature de ponto de variação ou vp-feature) no modelo, enquanto outras features desempenham o papel de suas possíveis variantes (chamadas variantes de feature)

- Os requeridos e restrições mutual_exclusion. Essas regras definem as restrições semânticas nas features opcionais e variantes, para dar consistência para o modelo. É possível especificar quais as features devem ser selecionadas juntas e quais não podem ser selecionadas juntas



Processo Para a Construção do Modelo de Feature

- Analogia do processo de construção do modelo de objeto OOSE (Object-Oriented Software Engineering) ou RSEB :
  - Identificar o objeto: Dado um modelo de caso de uso, aplique o padrão Boundary-Entity-Control para mapear cada caso de uso em uma colaboração de objetos
  - Homogeneizar o conjunto de objetos identificados: Decidir se os objetos com mesmo nome são, de fato, os mesmos objetos, diferentes instâncias do mesmo objeto ou objetos diferentes. Se necessário renomear ou mesclar os objetos
  - Realizar análise de robustez: Agrupar objetos em sub-sistemas estáveis, possivelmente reagrupando ou mesclando os objetos. Aplicar princípios de coesão e acoplamento para aumentar a robustez de clusters de objetos
  - Completar o conjunto de operações: Examinar os casos de uso e atribuir responsabilidades e operações aos objetos correspondentes. No final do processo o modelo de caso de uso pode ser re-escrito em termos dos objetos para obter o modelo de caso de uso de análise.

- O próximo passo é mapear os objetos de análise para objetos de projeto, adicionando ou modificando o objeto para considerar restrições de implementação. O UML «trace» conecta os casos de uso aos objetos de análise correspondentes e objetos de análise aos objetos de projeto correspondentes.
- Considerando o processo em FeatuRSEB, a primeira extensão para o processo RSEB é explicitar o conjunto de modelos de uso de caso usados para construir o Modelo de Casos de Uso Familiar(Modelo de Caso de Uso do Domínio), cada um associada com sistema exemplar selecionado para a análise de domínio.
- Em seguida, análogo ao processo para a RSEB, no FeatuRSEB é executada a uniformização / variabilidade e análise/síntese do valor agregado, em primeiro lugar sobre os modelos de caso de uso e, em seguida, diretamente no modelo de features na fase de análise de robustez.

- A construção do modelo de features pode ser esboçado das seguintes formas:
  - Mesclar modelos de casos de uso exemplares individuais em um modelo de caso de uso de domínio, usando a captura de pontos de variação e expressando as diferenças. Usando o «trace» para manter o rastro do originário
  - Criar um modelo de features inicial com funcionalidades derivadas do modelo de caso de uso do domínio (normalmente usando nomes de casos de uso como ponto de partida para nomes das features)
  - Criar o modelo de objeto de análise RSEB, aumentando o modelo de feature com features arquiteturais. Esses recursos referem-se à estrutura e configuração do sistema e não a funções específicas
  - Criar o modelo de projeto RSEB, aumentando o modelo de feature com features de implementação.

Construção do Modelo de Caso de Uso de Domínio

- Construir o modelo de atores do domínio
- Construir o modelo de caso de uso do domínio
- Realizar a análise de robustez no modelo de caso de uso do domínio

Extraindo Features Funcionais do Modelo de caso de Uso do Domínio

- Identificar as features mandatórias e opcionais
- Decompor as features em sub-features
- Identificar vp-features e suas variantes
- Realizar a análise de robustez no modelo de features

Notação de Czanercki [2000]

Notação de Czanercki [2000]

- A raiz do diagrama representa um conceito, aqui chamado de *concept node*. Os demais elementos são chamados de *features nodes*.
- Assim como no FODA, as *features* são separadas em mandatórias, alternativas e opcionais. A diferença é a adição das *or-features*.
- Além disso, é possível combinar as *features* opcionais com as alternativas, gerando o tipo de *features* alternativas opcionais.

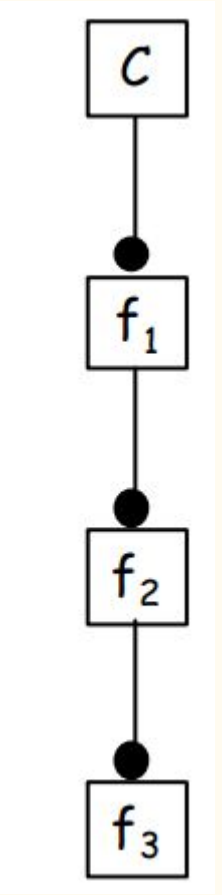


Figura 1: Diagrama de *features*

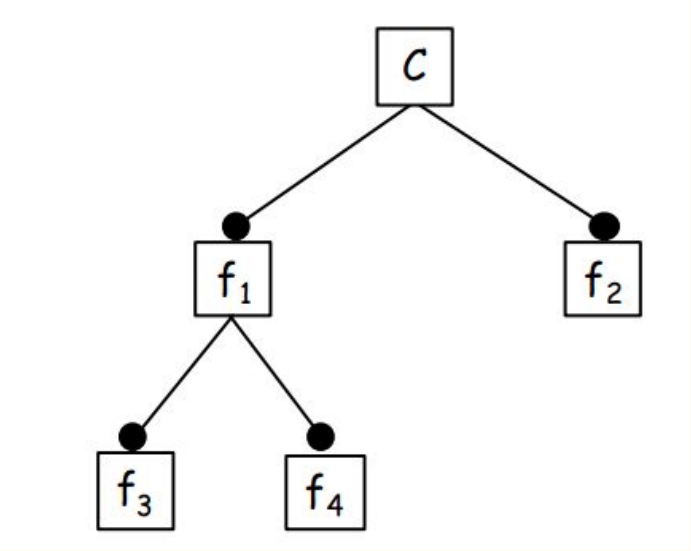


Figura 2: Exemplo de *features* mandatórias

Feature set {C, f1, f2, f3, f4}.

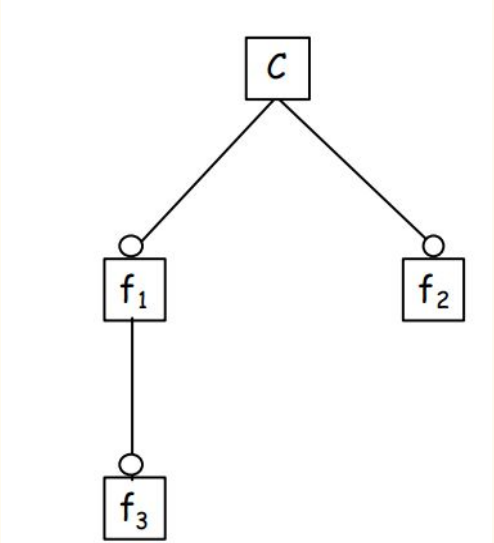


Figura 3: Exemplo de *features* opcionais

Feature set

1. {C};
2. {C, f1};
3. {C, f1, f3};
4. {C, f2};
5. {C, f2, f3};
6. {C, f1, f2};
7. {C, f1, f3, f2}.

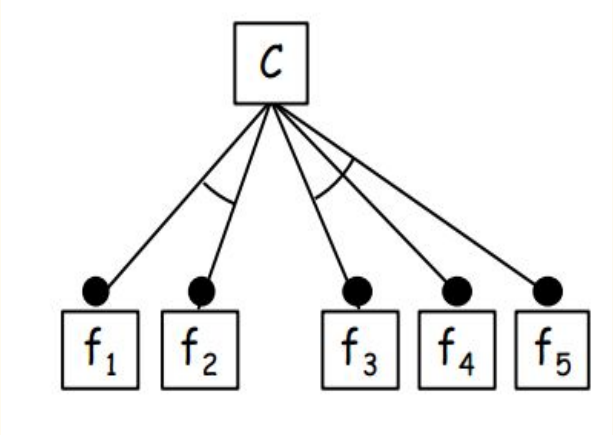


Figura 4: Exemplo de *features* alternativas

Feature set

1. {C, f1, f3};
2. {C, f1, f4};
3. {C, f1, f5};
4. {C, f2, f3};
5. {C, f2, f4};
6. {C, f3, f4};
7. {C, f2, f5}.

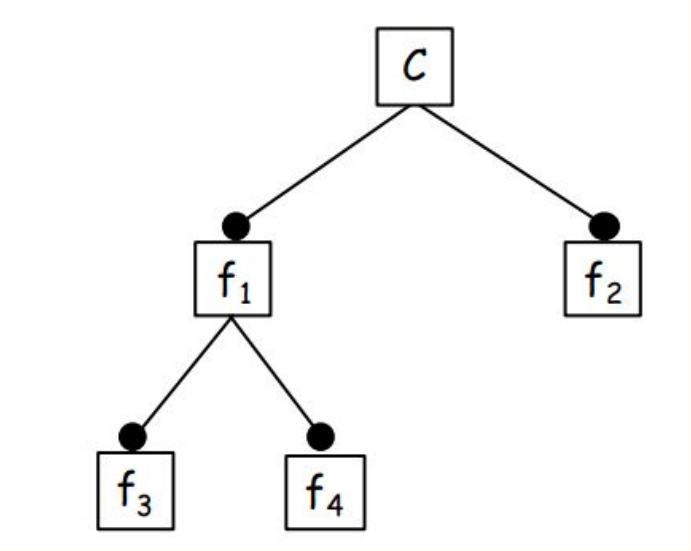


Figura 2: Exemplo de *features* mandatórias

Feature set {C, f1, f2, f3, f4}.

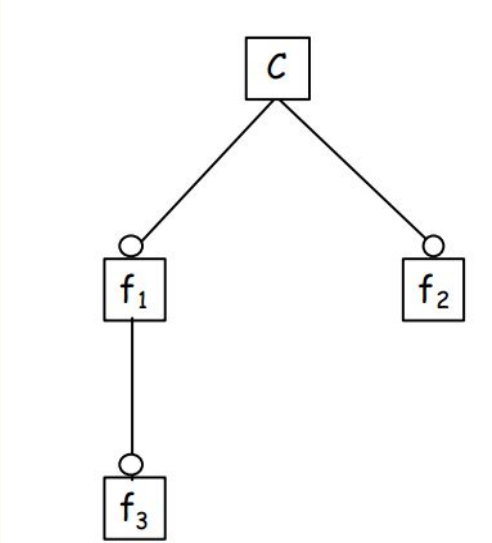


Figura 3: Exemplo de *features* opcionais

Feature set

1. {C};
2. {C, f1};
3. {C, f1, f3};
4. {C, f2};
5. {C, f2, f3};**X inválido**
6. {C, f1, f2};
7. {C, f1, f3, f2}.

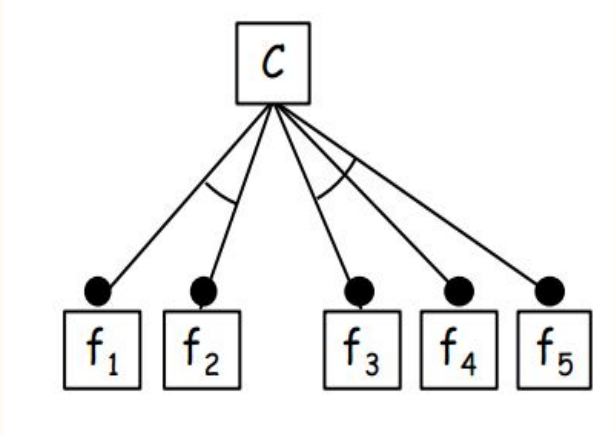


Figura 4: Exemplo de *features* alternativas

Feature set

1. {C, f1, f3};
2. {C, f1, f4};
3. {C, f1, f5};
4. {C, f2, f3};
5. {C, f2, f4};
6. {C, f3, f4};**X inválido**
7. {C, f2, f5}.

Dimensões

- Uma feature com apenas um conjunto de *subfeatures* alternativas e nenhuma outra *subfeature* é denominado “dimensão”.

- Além disso, um conjunto de *subfeatures* alternativas junto com uma ou mais features mandatórias também é considerado uma dimensão.

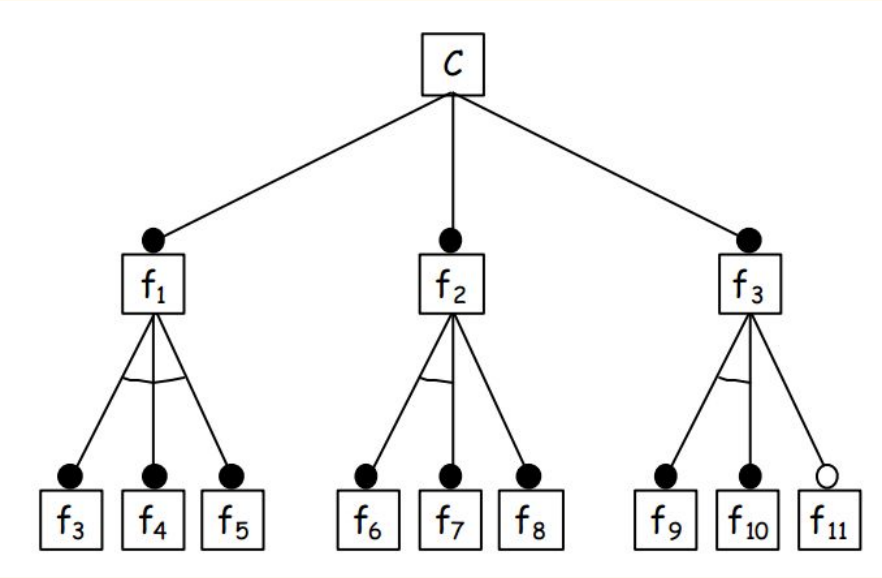


Figura 5: Exemplo de diagrama de *features* com duas dimensões

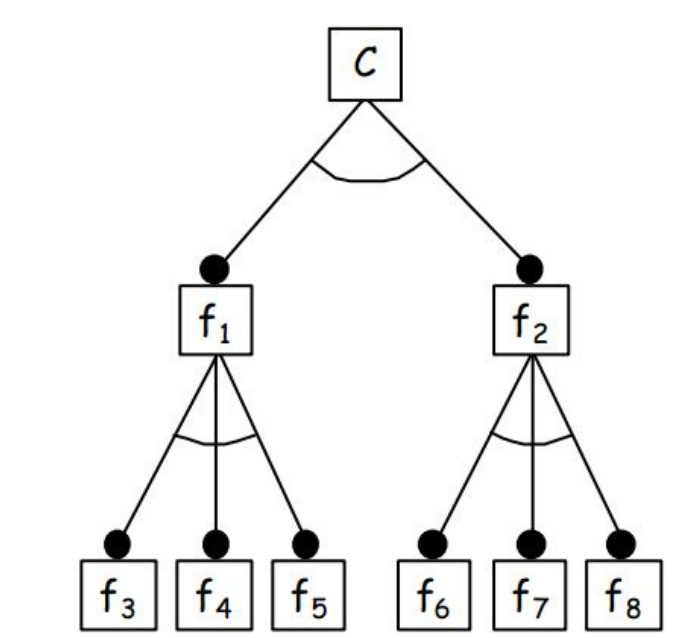


Figura 6: Exemplo de diagrama de *features* dimensões alternativas

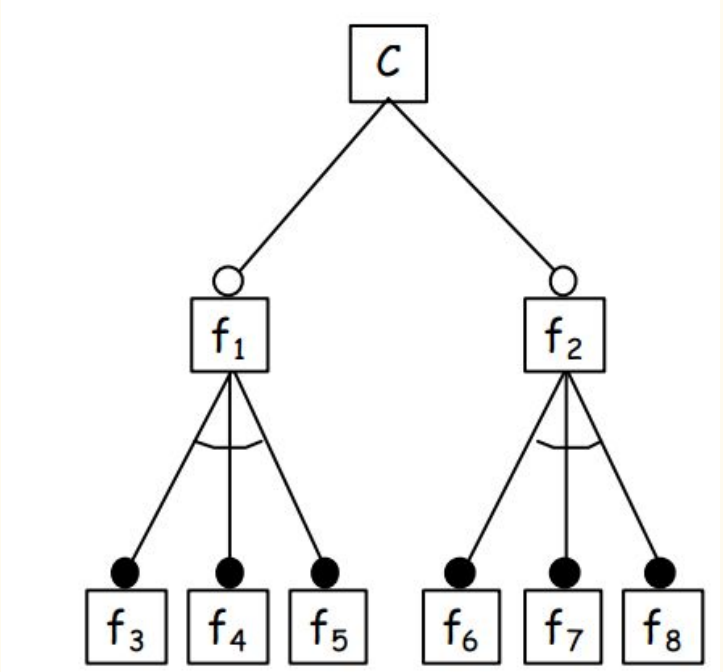


Figura 7: Exemplo de diagrama de *features* dimensões opcionais

Features alternativas opcionais

- Uma *feature* alternativa também pode ser opcional.

- Porém, durante a normalização (veremos a seguir), todo o conjunto de *features* alternativas contendo uma ou mais features opcionais será transformado em features alternativas opcionais.

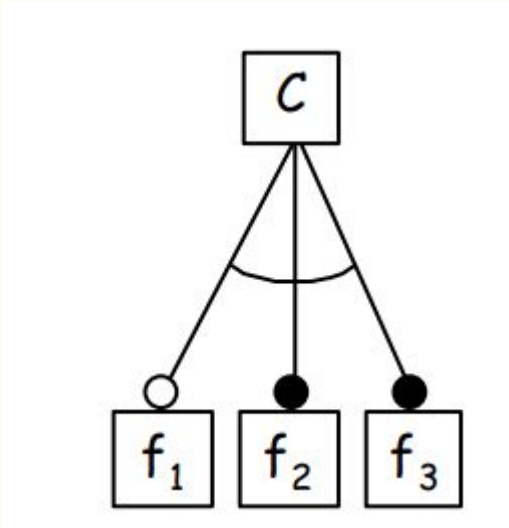


Figura 8: Exemplo de diagrama com uma *feature* alternativa opcional

Or-Features

- Um conceito pode ter um ou mais conjuntos de *or-features*. Uma *features*, também pode ter um ou mais conjuntos de *or-features*.
- Nesse caso, pelo menos **uma** *or-feature* deve ser selecionada de cada conjunto.

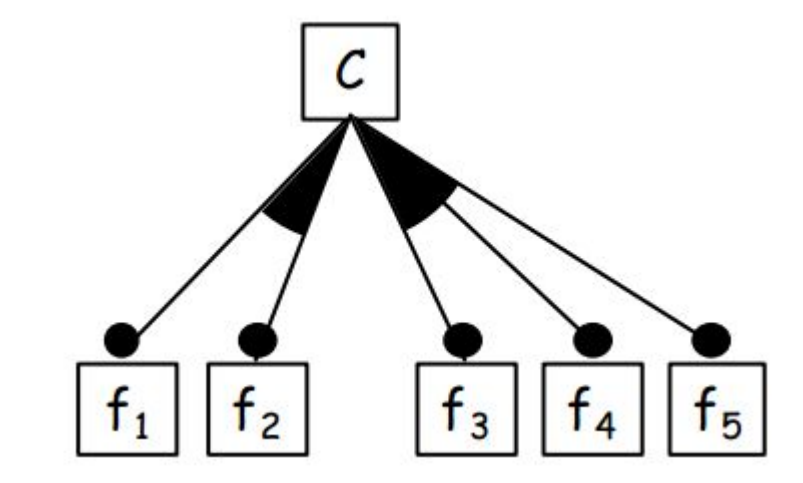


Figura 9: Exemplo de um diagrama de *feature* com dois conjuntos de *or-features*.

Optional Or-Features

- Uma *or-feature* também pode ser opcional.
- Porém, durante a normalização, todo o conjunto de *or-features* que contenha uma ou mais *or-features* opcionais são transformados em *or-features* opcionais.

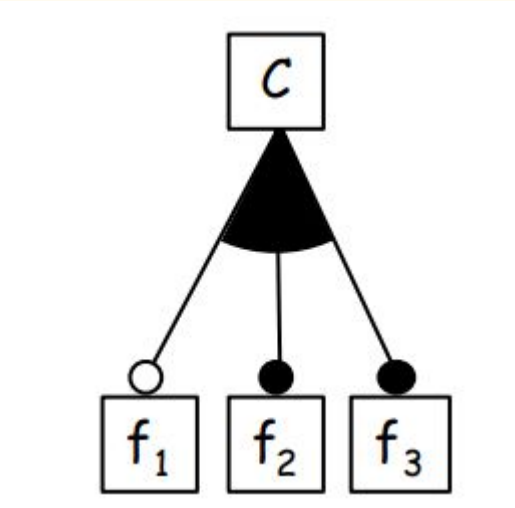


Figura 10: Exemplo de um diagrama de *features* com *optional or-feature*

Normalização

- Ter uma ou mais *features* opcionais, em um conjunto de *features* alternativas, é o mesmo que todas as *features* desse conjunto serem opcionais.

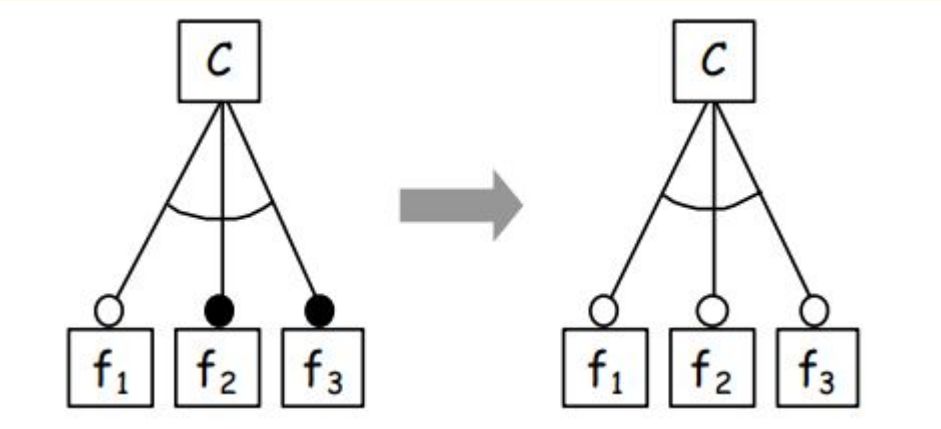


Figura 11: Exemplo de normalização de *feature* alternativa opcional

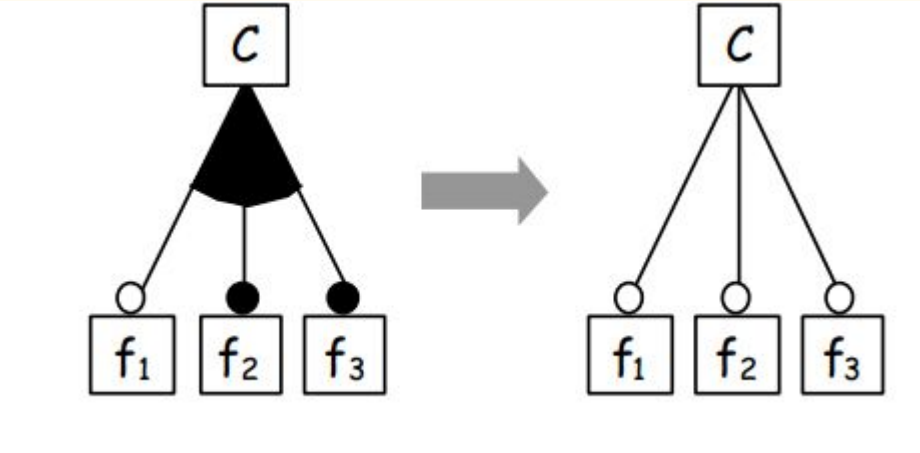
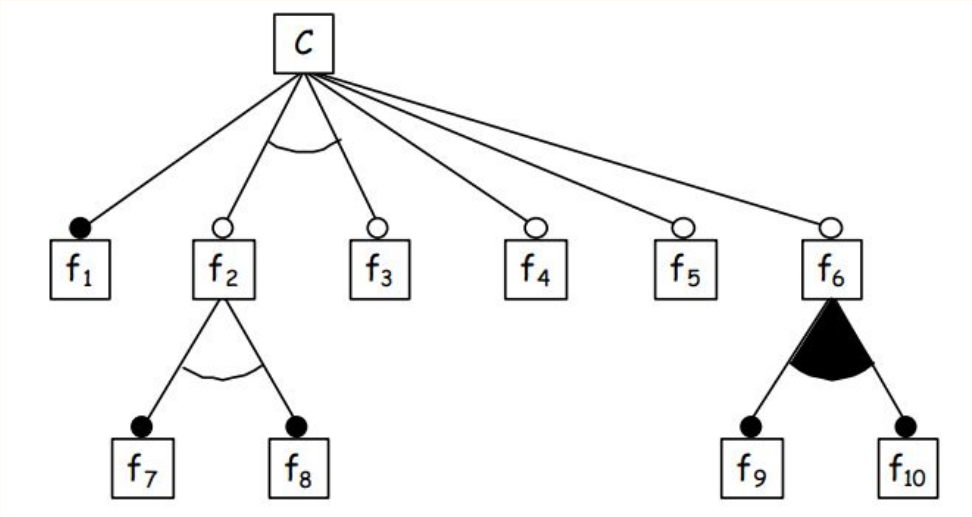


Figura 12: Exemplo de normalização de *or-feature* opcional

Diagramas de *features* sem bordas



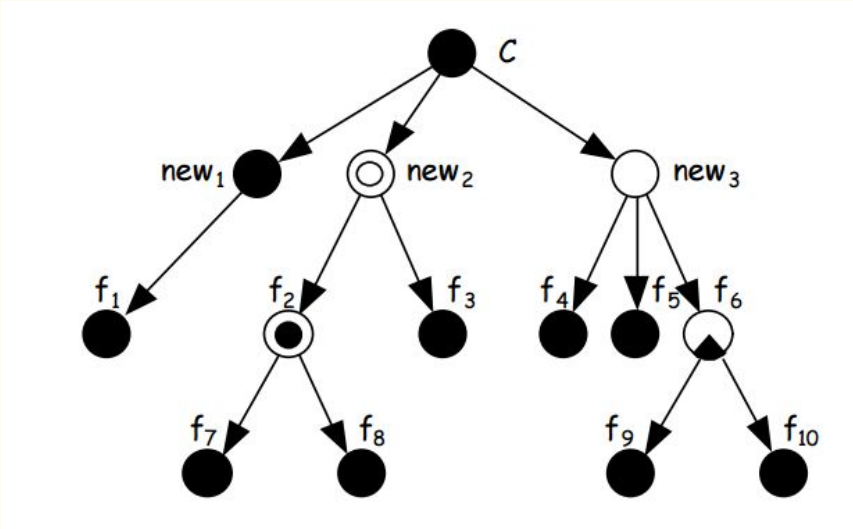


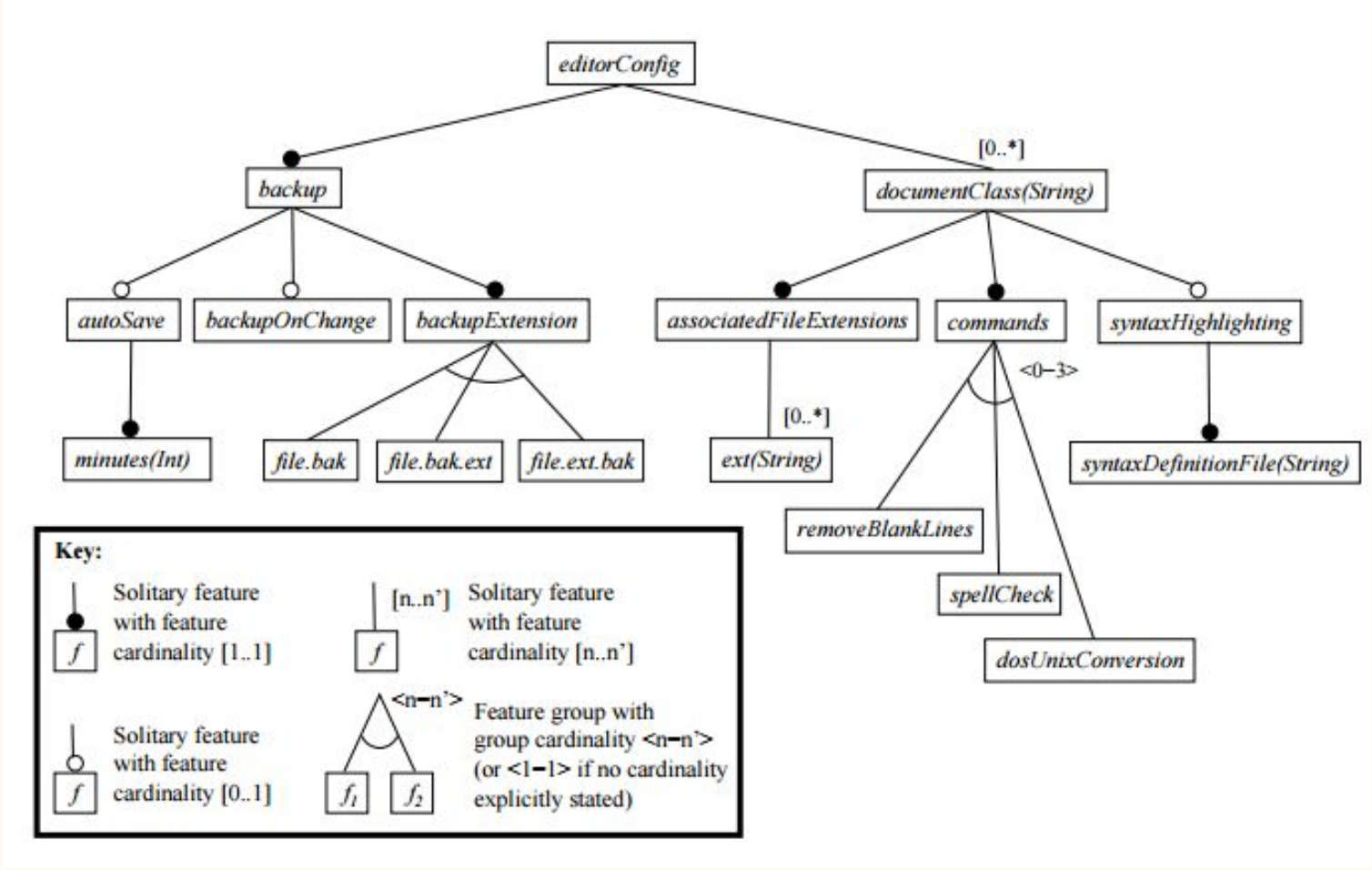
Figura 13: Exemplo de diagrama sem bordas

Notação de Czanercki [2005]

Tipos de *features*

- *Features* agrupadas: fazem parte de um grupo de *features*

- *Features* solitárias: não fazem parte de nenhum grupo. São mais comuns.

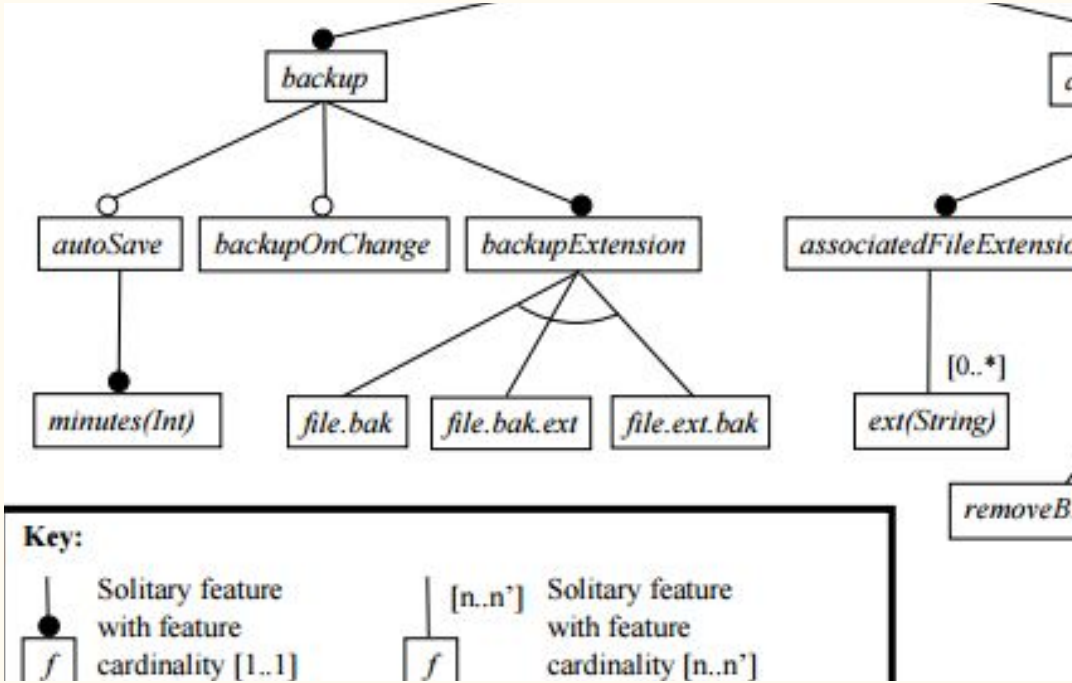


Exemplo de diagrama de *features* de um editor de texto

Atributos

- Uma *feature* pode ter um atributo.

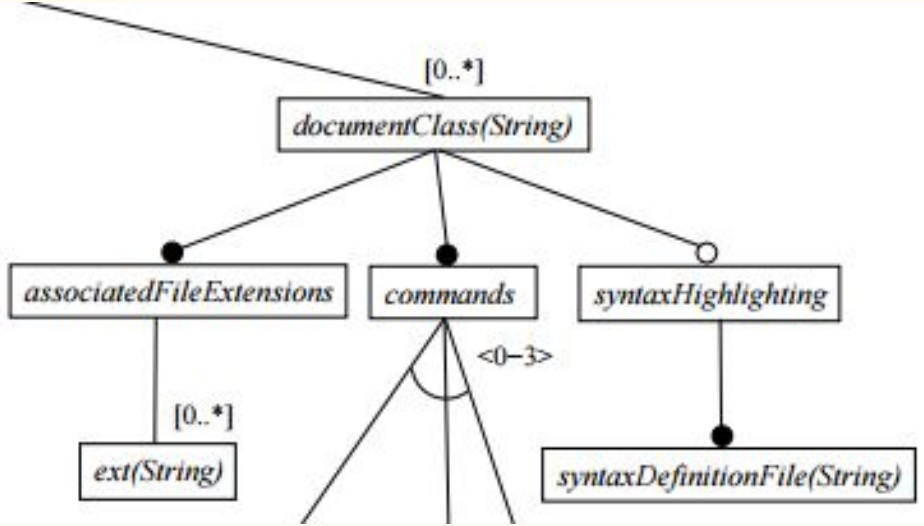
- O atributo é associado com um tipo, mas também pode especificar um valor.



Cardinalidades

- Toda *feature* solitária é qualificada por uma cardinalidade. Esta, indica quantas vezes a sub-árvore pode ser replicada(especializada).

- *Features* sem cardinalidade explícita podem ser [1..1] (*features* mandatórias) ou [0..1] (*features* opcionais).

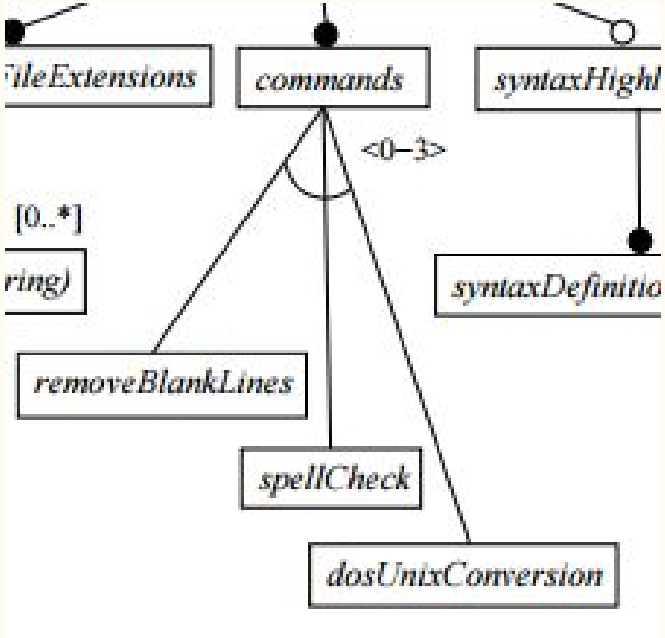


Cardinalidade

- Uma cardinalidade pode ter mais de um intervalo;
- Ex 1 : [0..2][6..6]. Aqui, podemos especializar a *feature* 0, 1, 2 ou 6 vezes.
- Ex 2 : [1..2][5..*]. Aqui, podemos especializar a *feature* 0, 1, 2 ,5 ou **n** vezes, tal que **n**$>$5.

Cardinalidade de Grupo

- Restringe as opções de seleção de uma *feature* dentro de um grupo.
- Ex: [1-5]. Aqui Aqui, deve se selecionar **pelo menos** uma feature e no **máximo** 5.
- Se nenhuma cardinalidade é explicita, então a cardinalidade do grupo é [1-1];

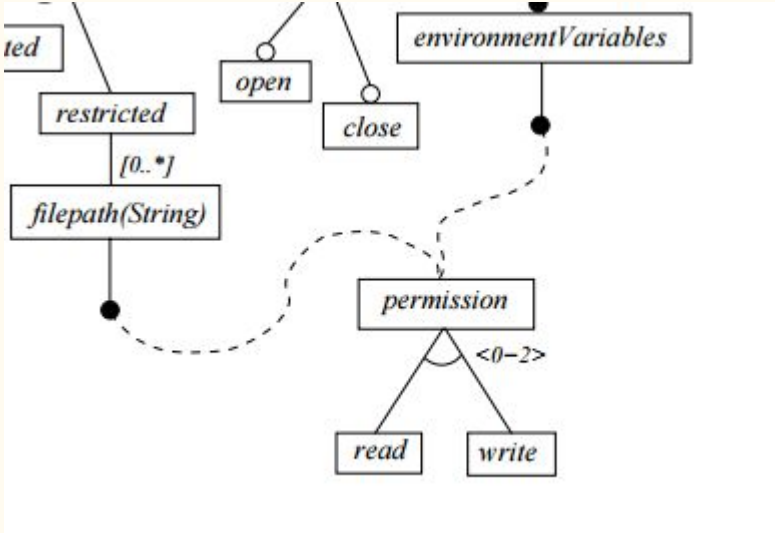


Cardinalidade de Grupo

Definição formal: Cardinalidade de grupo $[n^1\text{-}n']$ é o intervalo com **n** como limite inferior e n' como limite superior. Dado que $k > 0$ é o número de *features* do grupo, então: $0 \leq n \leq n' \leq k$.

Referenciar diagrama

- Um diagrama de *feature* pode referenciar outro se necessário;

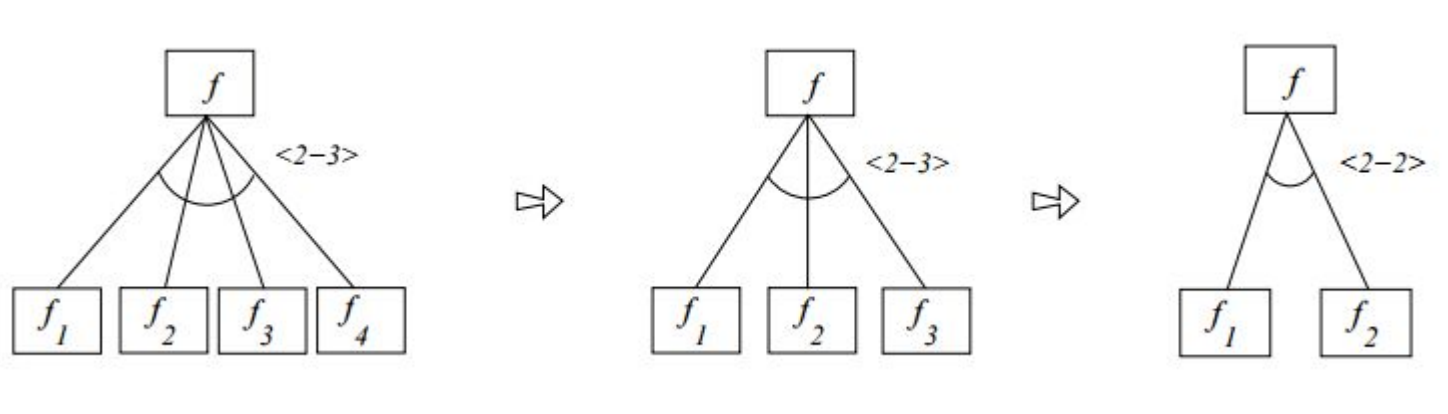


Configuração vs Especialização

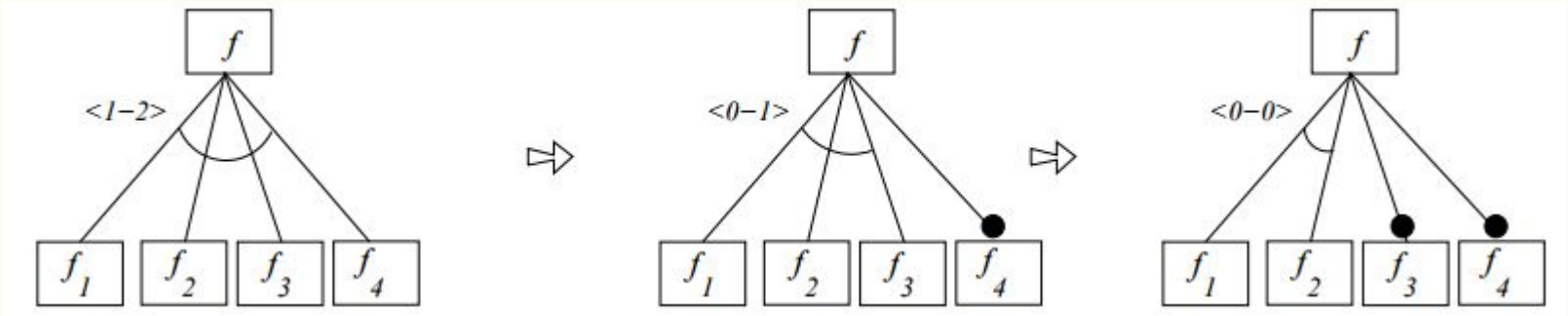
- Configuração: consiste nas *features* selecionadas de acordo com o diagrama.
  - A relação entre um diagrama de *features* e uma configuração é semelhante a relação de uma classe e sua instanciação em uma linguagem de programação orientada a objetos.
- Especialização: transformar um diagrama de *features* e gerar outro diagrama de *features*. Um diagrama completamente especializado significa uma configuração.
- Existem duas formas de realizar a configuração:
  - Derivar uma configuração de um diagrama diretamente;
  - Especializar o diagrama completamente e então derivar a configuração (trivial).

Especializando

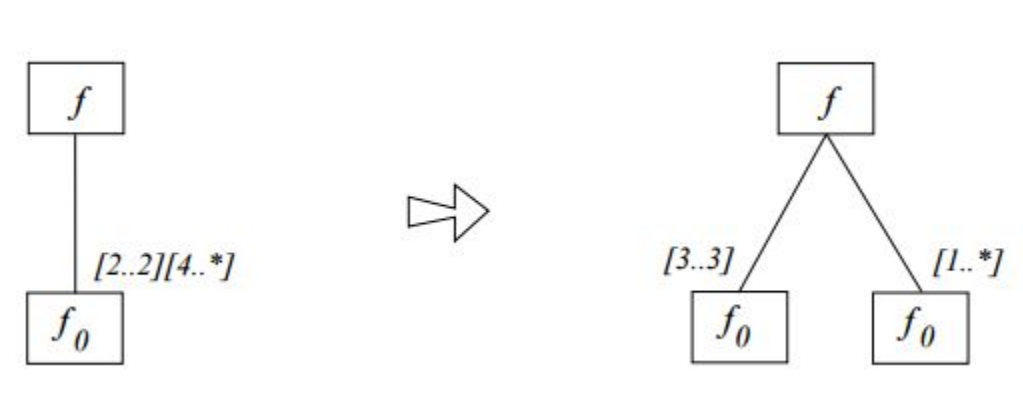
1. Refinar uma cardinalidade de *feature*;
2. Refinar uma cardinalidade de grupo;
3. Remover *subfeatures* de um grupo;
4. Selecionar uma *subfeature* de um grupo;
5. Atribuir um valor a um atributo;
6. Clonar uma *subfeature* solitária;



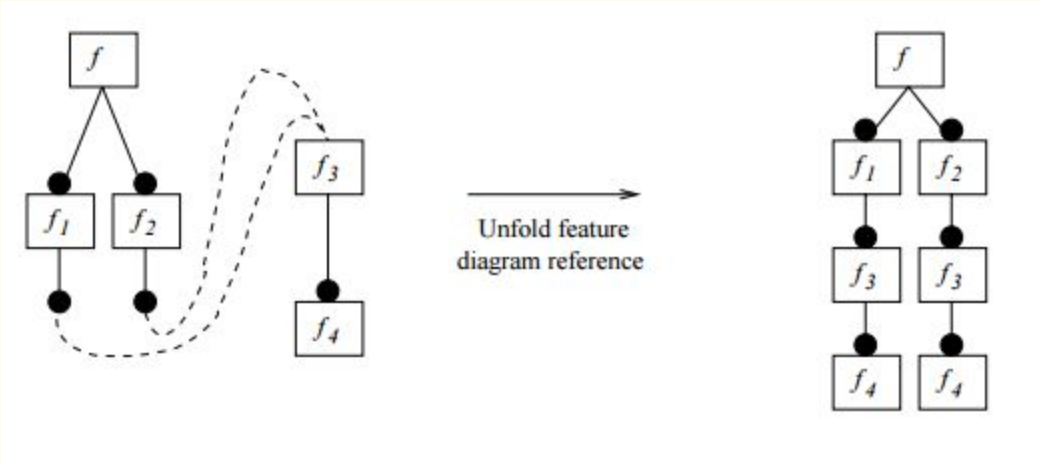
Exemplo de especialização (removendo *subfeatures* do grupo)



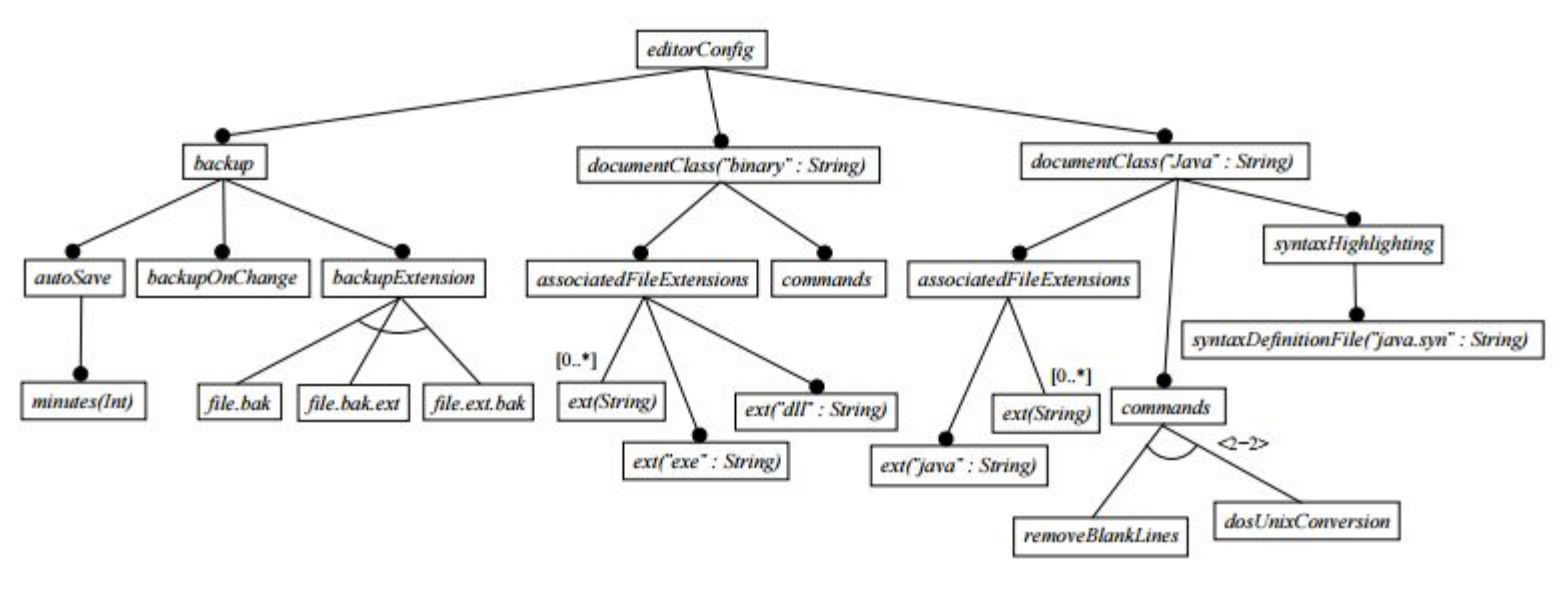
Exemplo de especialização (selecionando *subfeatures* de um grupo)



Exemplo de especialização (clonando *feature* solitária)



Exemplo de especialização de uma referência



Exemplo de diagrama especializado de *features* de um editor de texto

Concluindo

Pontos importantes

- Essa notação permite aproveitar o que a FODA e a RSEB descrevem melhor, modelo de feature e casos de usos
- Há a vp-feature, que sendo mandatória ou opcional, nos limita e escolher somente uma das features disponibilizadas(XOR)
- Há a vp-feature use time bound, que sendo mandatória ou opcional, nos permite escolher um ou mais features disponibilizadas(OR)
- Há os requeridos, que definem quais features são necessárias para ser permitido a seleção de outra feature

- Há as restrições mutual_exclusion, que limitam as escolhas, quando quiser uma feature, outra não pode ser selecionada

Pontos importantes

- É possível combinar as *features* opcionais com as alternativas, gerando o tipo de *features* alternativas opcionais.
- Uma *or-feature* representa que pelo menos **uma** *feature* deve ser selecionada de cada conjunto.
- Possuir uma ou mais *features* opcionais, em um conjunto de *features* alternativas, é o mesmo que todas as *features* desse conjunto serem opcionais.
- Uma *feature* pode ter um atributo associado com um tipo, que também pode especificar um valor.
- Toda *feature* solitária é qualificada por uma cardinalidade que indica quantas vezes a sub-árvore pode ser replicada(especializada).

Pontos Importantes

- Cardinalidade de grupo restringe as opções de seleção de uma *feature* dentro de um grupo.
- Um diagrama de *feature* pode referenciar outro.
- Configuração: consiste nas *features* selecionadas de acordo com o diagrama.
- Especialização: transformar um diagrama de *features* e gerar outro diagrama de *features*.

Próxima semana

- Notação de Gurp-Bosch-Svahnberg Notation [van Gurp et al. 2001].

Referências

Kang, K., Cohen, S., Hess, J., Nowak, W. and Peterson, S. (1990). Feature-oriented domain analysis (FODA) feasibility study, Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

Griss, Martin L., John Favaro, and Massimo d'Alessandro. "Integrating feature modeling with the RSEB." Software Reuse, 1998. Proceedings. Fifth International Conference on. IEEE, 1998.

Czarnecki, K. and Eisenecker, U. W. (2000). Generative Programming: Methods, Tools, and Applications, AddisonWesley, Boston, MA.

Czarnecki, K., Helsen, S. and Eisenecker, U. (2004). Staged configuration using feature models, in R. L. Nord (ed.), Software Product Lines: Third International Conference, SPLC 2004, Boston, MA, USA, August 30- September 2, 2004. Proceedings, Vol. 3154 of Lecture Notes in Computer Science, Springer-Verlag, Heidelberg, Germany, pp. 266–283.

Czarnecki, K., Helsen, S. and Eisenecker, U. (2005). Formalizing cardinality-based feature models and their specialization, Software Process Improvement and Practice 10(1).