PL-AOVgraph: Uma extensão de AOV-Graph para Linha de Produto de Software

Lidiane Santos
diane_lid@hotmail.com

Thais Batista
thaisbatista@gmail.com

Lyrene Silva
lyrene@gmail.com

Roteiro

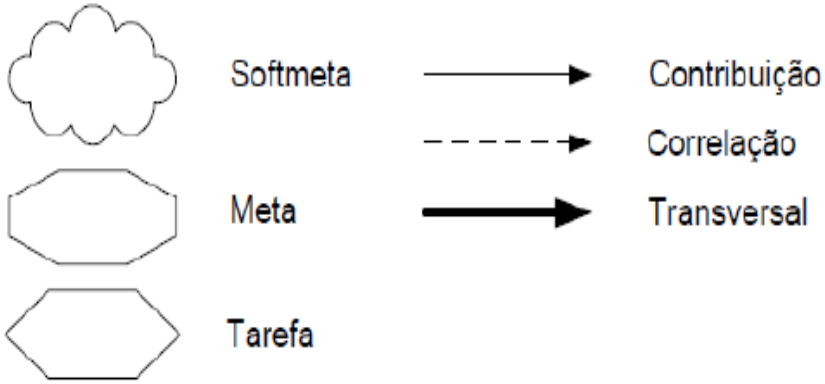
- Introdução
  - Motivação
  - Objetivos
- AOV-Graph
- PL-AOVgraph
- Mecanismo de Mapeamento
- Estudo de Caso
- Conclusão

Introdução

- Motivação
  - Linhas de Produto de Software
    - Modelo de *Features*
    - Limitações
  - Características Transversais
    - AOV-Graph
    - Limitações
- Objetivos
  - PL-AOVgraph
  - Mapeamento bi-direcional entre PL-AOVgraph e Modelo de *Features* automatizado pela ferramenta ReqSys

AOV-Graph

- Linguagem de Modelagem de Requisitos Orientada a Aspectos
- Extensão de V-Graph
- Separação e composição de características transversais
- Componentes
  - Tarefa (*Task)*
  - *Softmeta (Softgoal)*
  - Meta (*Goal)*
- Relacionamentos
  - Contribuição (*and, or, xor*)
  - Correlação (*make, break, help, hurt, unknown*)
  - Transversal



PL-AOVgraph

- AOV-Graph não é suficiente para representar variabilidades
- Limitações do Modelo de *Features*
- Extensão de AOV-Graph para Linha de Produto
- Relacionamento de Contribuição *inc-or*
- Propriedades
  - *cardinalityMin*
  - *cardinalityMax*
  - *groupFeature*
  - *cardinalityGroupMin*
  - *cardinalityGroupMax*
  - *isFeature*
- Mapeamento bi-direcional entre PL-AOVgraph e Modelo de *Features*

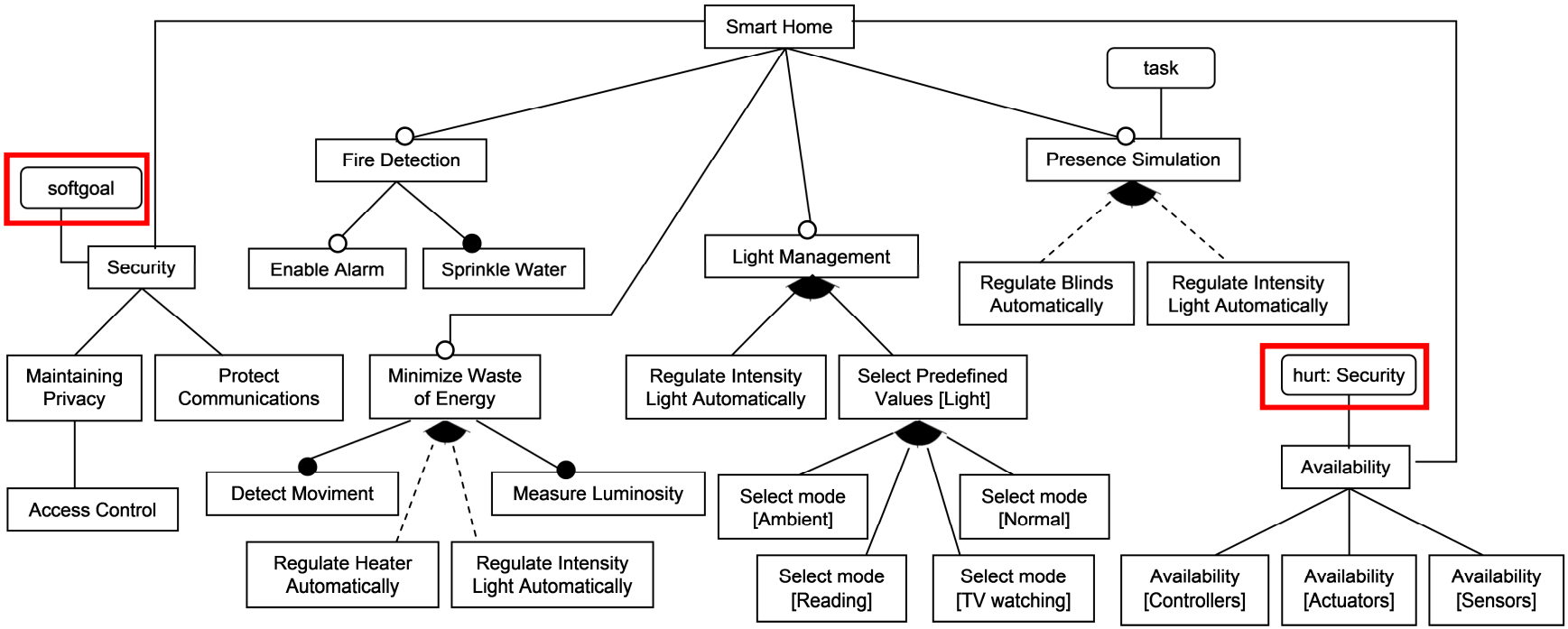
Especificação PL-AOVgraph de Smart Home

```
1.  goal_model (Smart Home; GM1){
2.          task Fire Detection (or; T2;){
3.                  task Enable Alarm (or; T2.1;){
4.                          task Activate Siren (inc-or; T2.1.1;){ property{isFeature=no} }
5.                          task Activate Lights (inc-or; T2.1.3;){ property{isFeature=no} } }
6.                  task Sprinkle Water (and; T2.2;){} }
7.          task Light Management (or; T3;){
8.                  task Regulate Intensity Light Automatically (inc-or; T3.1;){}
9.                  task Select Predefined Values [Light] (inc-or; T3.2;){
10.                         task Select mode [TV watching] (inc-or; T3.2.1;){}
11.                         task Select mode [Reading] (inc-or; T3.2.2;){}
12.                         task Select mode [Normal] (inc-or; T3.2.3;){}
13.                         task Select mode [Ambient] (inc-or; T3.2.4;){} } }
14.         task Presence Simulation (or; T5;){
15.                 task_ref = (Regulate Blinds Automatically; T1.2; inc-or;) }
16.         task Minimize Waste of Energy (or; T6;){
17.                 task Measure Luminosity (and; T6.1;){}
18.                 task Detect Movement (and; T6.2;){}
19.                 task_ref = (Regulate Heater Automatically; T4.2.1; inc-or;) }
20.         softgoal Security (S1;){
21.                 softgoal Maintaining Privacy (S1.1;){
22.                         softgoal Access Control (S1.1.1;){} }
23.                 softgoal Protect Communications (S1.2;){} }
24          softgoal Availability (S2;){
25.                 softgoal Availability [Controllers] (S2.1;){}
26.                 softgoal Availability [Sensors] (S2.2;){}
27.                 softgoal Availability [Actuators] (S2.3;){} }
28          correlation (hurt){
29.                 source = softgoal_ref = (Availability; S2;)
30.                 target = softgoal_ref = (Security; S1;) }
31.         crosscutting (source = Light Management (T3)){
32.                 pointcut (PC1): include(Presence Simulation; T5;) and include(Minimize Waste of Energy; T6;)
33.                 advice (around): PC1{
34.                         task_ref = (Regulate Intensity Light Automatically; T3.1; inc-or;) } }
35. }
```

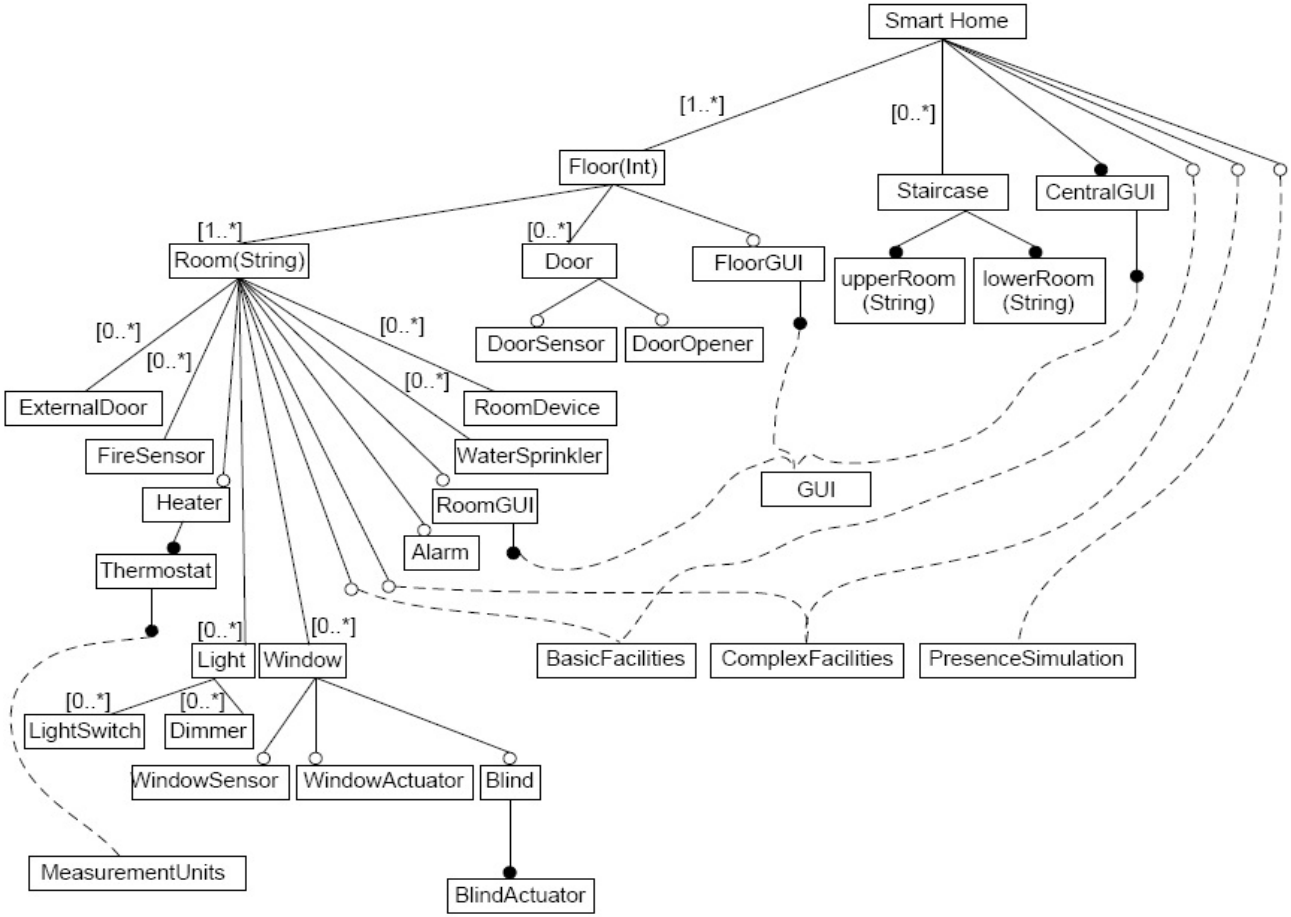
Mecanismo de Mapeamento
(De PL-AOVgraph para Modelo de *Features*)

Regra	Descrição
1	Para cada *goal model* será criada uma nova raiz, ou seja, um novo Modelo de *Features*.
2	O Modelo de *Features* segue a mesma hierarquia da árvore de PL-AOVgraph, ou seja, um componente pai em PL-AOVgraph se tornará uma *feature* pai, bem como um componente filho, uma *feature* filha.
3	Os relacionamentos de contribuição dos tipos *and*, *or*, *xor* e *inc-or* correspondem a *features* obrigatórias, opcionais, alternativas e ou-inclusivo, respectivamente.
4	Para os componentes com as propriedades *cardinalityMin* e *cardinalityMax*, se a cardinalidade mínima for igual a 0 será gerada uma *feature* opcional com cardinalidade [0..m], porém se a cardinalidade mínima for diferente de 0, será criada uma *feature* obrigatória com cardinalidade [n..m], onde *n* está especificado em *cardinalityMin* e *m* em *cardinalityMax*.
5	Os elementos agrupados por meio da propriedade *groupFeature* se tornam *features* agrupadas, através do agrupamento de *features* com cardinalidade [i..j], onde *i* é especificado em *cardinalityGroupMin* e *j* em *cardinalityGroupMax*.
6	Os componentes que são referenciados aparecem novamente como *features*, porém com uma marcação específica, para indicar referência.
7	O relacionamento de correlação é representado por meio de uma anotação, indicando o tipo da correlação e a *feature* correspondente ao *target*, que é adicionada a *feature* que representa o *source* deste relacionamento.
8	No relacionamento transversal, os *advices* se tornam *features* de referência, que são chamadas pelas *features* que representam os *poincuts*.
9	Os componentes que possuírem a propriedade *isFeature* configurada como *"no"* não serão considerados *features*.
10	O tipo de cada componente PL-AOVgraph é indicado no conteúdo de uma anotação, que é adicionada a cada feature.

Modelo de Features gerado



Modelo de Features de Smart Home



Mecanismo de Mapeamento
(De Modelo de *Features* para PL-AOVgraph)

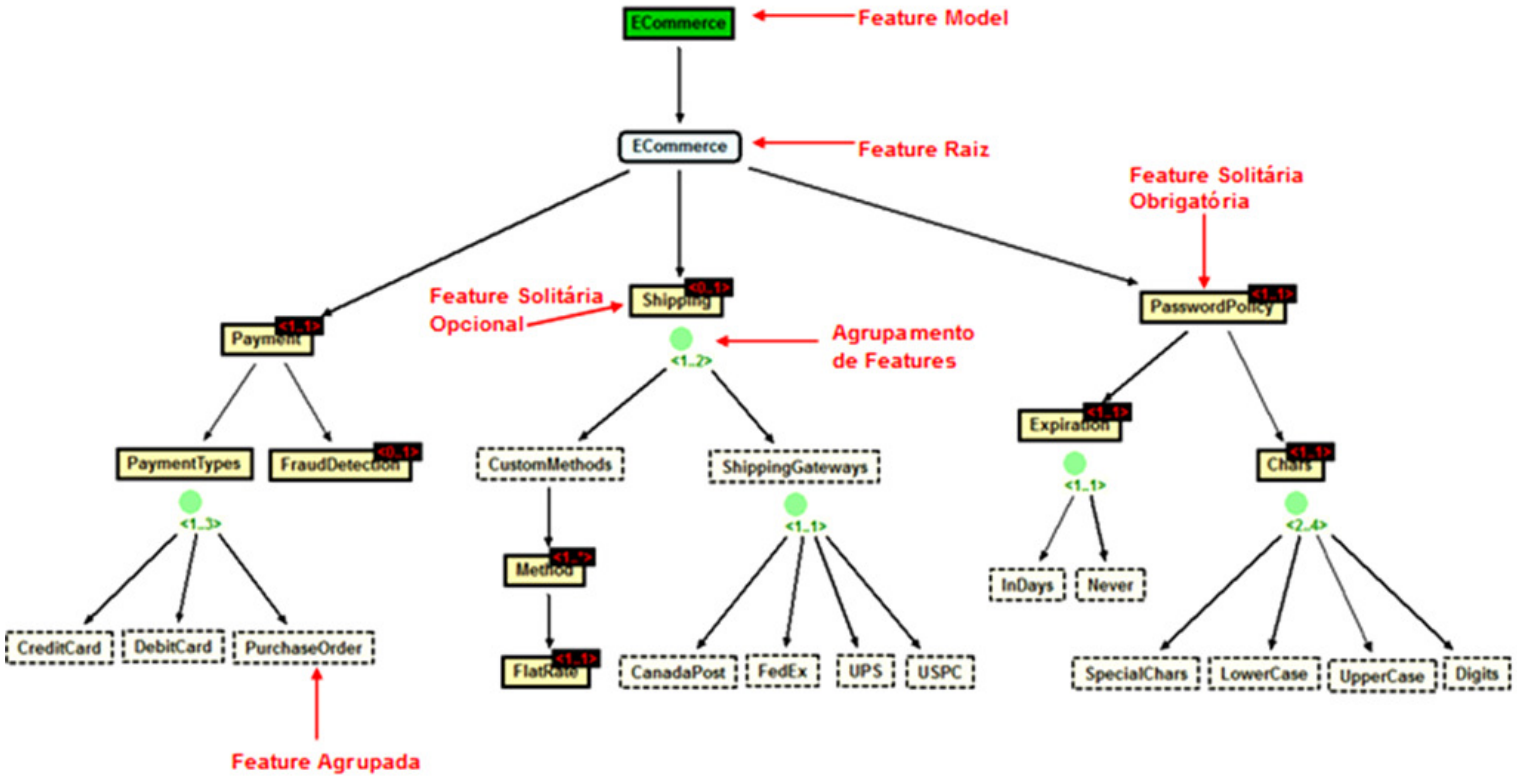
Regra	Descrição
1	Para cada raiz, ou seja, para cada Modelo de Features, será criado um *goal model*.
2	PL-AOVgraph segue a mesma hierarquia da árvore do Modelo de Features, ou seja, uma feature pai no Modelo de Features se tornará uma componente pai, bem como uma feature filha, um componente filho.
3	Features obrigatórias, opcionais, alternativas e ou-inclusivo são representadas por relacionamentos de contribuição dos tipos *and*, *or*, *xor* e *inc-or,* respectivamente.
4	Features com cardinalidade específica se tornam componentes com as propriedades *cardinalityMin* e *cardinalityMax*.
5	Agrupamento de features com cardinalidade específica, passa a ser representado através da propriedade *groupFeature* juntamente com as propriedades *cardinalityGroupMin* e *cardinalityGroupMax*.
6	As features marcadas como referência passam a ser componentes referenciados, representados pela palavra reservada *task_ref*.
7	Se uma feature possuir anotação e no conteúdo da anotação houver um dos tipos de relacionamento de correlação (*hurt, break, make, help, unknown*), será gerada uma correlação de forma que o source do relacionamento será a feature que possui a anotação, enquanto o tipo da correlação e o *target* estão descritos na anotação.
8	Cada feature de referência que aparecer mais de uma vez no modelo será adicionada a um relacionamento transversal, de forma que a feature de referência se torna o componente *advice*, as features que chamam a referência passam a ser os componentes do (s) *pointcut (s)* e a *feature* pai daquela referenciada se torna o componente *source*.
9	Se uma feature possuir anotação e no conteúdo da anotação houver o tipo do componente PL-AOVgraph (task, goal, softgoal), será gerado um componente conforme o especificado na anotação.

PL-AOVgraph gerado

```
1.  goal_model (Smart Home; GM1){
2.          task Floor(int) (T1;){property{cardinalityMin=1;cardinalityMax=n;}
3.                  task FloorGUI (or; T2;){
4.                          task_ref = (GUI; T26; and;) }
5.                  task Door (T3;){property{cardinalityMin=0;cardinalityMax=n;}
6.                          task DoorSensor (or; T4;){}
7.                          task DoorOpener (or; T5;){} }
8                   task Room(String) (T6;){property{cardinalityMin=1;cardinalityMax=n;}
9.                          task RoomDevice (T7;){property{cardinalityMin=0;cardinalityMax=n;}}
10.                         task WaterSprinkler (T8;){property{cardinalityMin=0;cardinalityMax=n;}}
11.                         task RoomGUI (or; T9;){
12.                                 task_ref = (GUI; T26; and;) }
13.                         task Alarm (or; T10;){}
14.                         task Window (T11;){property{cardinalityMin=0;cardinalityMax=n;}
15.                                 task Blind (or; T12;){
16.                                         task BlindActuador (and; T13;){} }
17                                  task WindowActuator (or; T14;){}
18.                                 task WindowSensor (or; T15;){} }
19.                         task Light (T16;){property{cardinalityMin=0;cardinalityMax=n;}
20.                                 task Dimmer (T17;){property{cardinalityMin=0;cardinalityMax=n;}}
21.                                 task LightSwitch (T18;){property{cardinalityMin=0;cardinalityMax=n;}} }
22                          task ExternalDoor (T19;){property{cardinalityMin=0;cardinalityMax=n;}}
23.                         task FireSensor (T20;){property{cardinalityMin=0;cardinalityMax=n;}}
24.                         task Heater (or; T21;){
25.                                 task Thermostat (and; T22;){
26.                                         task_ref = (MeasurementUnits; T30; and;) } }
27.                         task_ref = (BasicFacilities; T31; or;)
28.                         task_ref = (ComplexFacilities; T51; or;) } }
29.         task Staircase (T23;){property{cardinalityMin=0;cardinalityMax=n;}
30.                 task upperRoom(String) (and; T24;){}
31                  task lowerRoom(String) (and; T25;){} }
32.         task CentralGUI (and; T26;){
33.                 task_ref = (GUI; T54; and;) }
34.         task_ref = (BasicFacilities; T54; or;)
35.         task_ref = (ComplexFacilities; T54; or;)
36.         task_ref = (PresenceSimulation; T54; or;)
37.  }
```

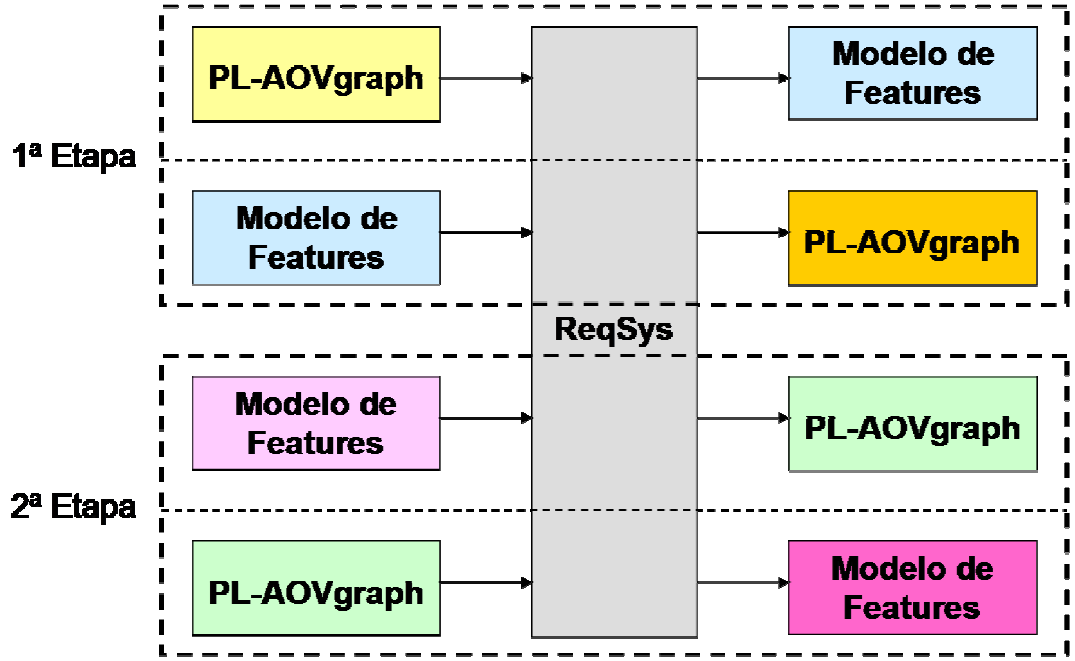
ReqSys

- Automatização do mapeamento bi-direcional
- Ferramenta XFeature para visualização dos Modelos de *Features*



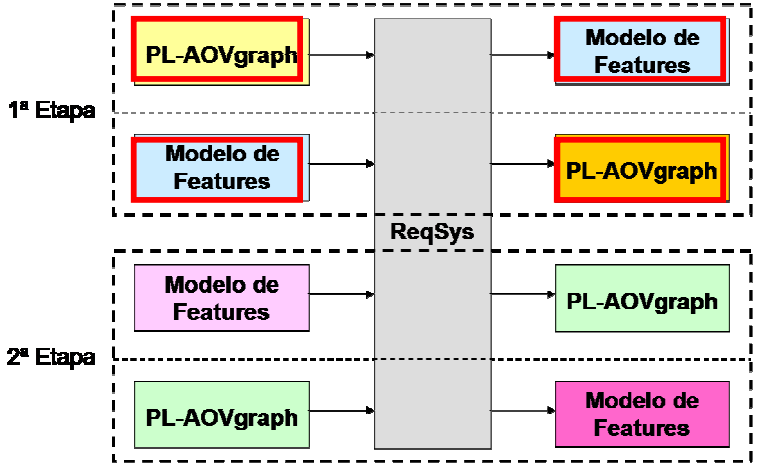
Estudo de Caso

- *Smart Home* – Sistema para automação residencial
- Elaboração da especificação PL-AOVgraph de *Smart Home*
- Etapas do estudo de caso



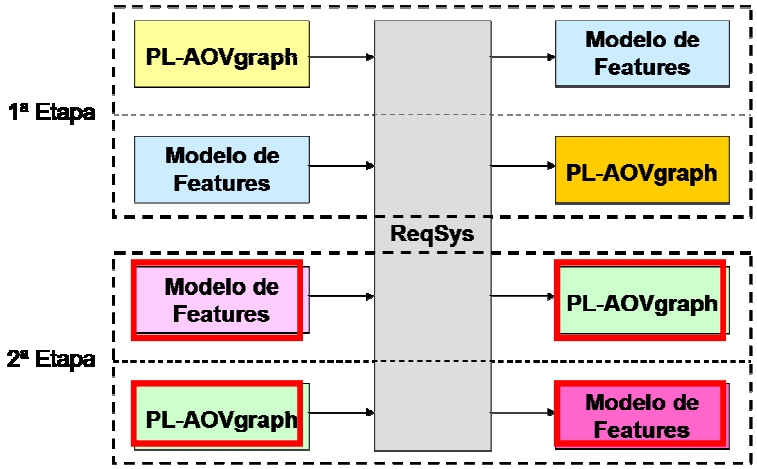
1ª Etapa

- Transformações
  - De PL-AOVgraph para Modelo de *Features*
  - De Modelo de *Features* para PL-AOVgraph
- Análise
  - Especificações PL-AOVgraph
    - Propriedade *isFeature* impede a criação de algumas *tasks*
  - Modelo de *Features*
    - Extenso



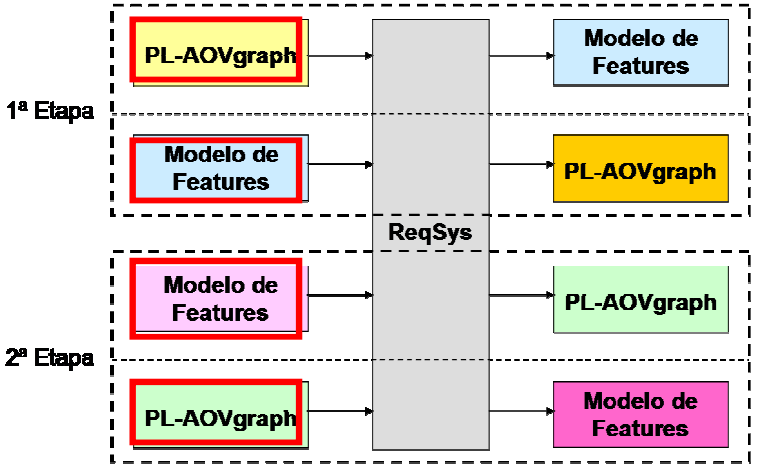
2ª Etapa

- Transformações
  - De Modelo de *Features* para
    - PL-AOVgraph
  - De PL-AOVgraph para Modelo de *Features*
- Análise
  - Modelos de *Features*
    - Equivalentes
  - Especificação PL-AOVgraph
    - *Tasks* não nomeadas adequadamente



Análise Geral

- Modelos de *Features*
  - Diferentes visões do sistema
    - Elementos físicos
    - Requisitos



  - Modelos de Features gerados a partir de especificações PL-AOVgraph representam o sistema sob uma visão detalhada
  - Modelo mais completo, porém extenso

- Especificações PL-AOVgraph
  - Imprecisão na nomeação dos requisitos

Conclusão

- Limitações do Modelo de *Features*
- Características Transversais
- Limitações de AOV-Graph
- Contribuições
  - PL-AOVgraph
  - Definição de regras de mapeamento entre PL-AOVgraph e Modelo de *Features*
  - Implementação do mapeamento bi-direcional na ferramenta ReqSys
  - Estudo de caso

Trabalhos Futuros

- Analisar se *goals* e *softgoals* podem ser transformadas em *features*
- Implementar a criação de *Intertype Declarations* e de *advices* dos tipos *before* e *after* a partir de *features*
- Utilizar semântica para melhorar a nomeação das *features* e dos componentes de PL-AOVgraph
- Elaborar um mecanismo de rastreabilidade entre PL-AOVgraph e Modelo de *Features* e implementá-lo

Referências

- BERG, K.; BISHOP, J. **Tracing Software Product Line Variability – From Problem to Solution Space**. Proceedings of SAICSIT, 2005.
- CLEMENTS, P.; NORTHROP, L. **Software Product Lines: Practices and Patterns.** Addison-Wesley, 2001.
- CZARNECKI, K.; HELSEN, S.; EISENECKER, U. **Formalizing Cardinalitybased Feature Models and their Specialization**. Software Process: Improvement and Practice, 10(1):7_29, 2005.
- FILMAN, R. E. et al. **Aspect-Oriented Software Development**. Addison-Wesley, 2005.
- GOMAA, H. **Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures.** Addison-Wesley, 2005.
- MCILROY, M. D. **Mass Produced Components**. In: The NATO Engineering Conference, 1968.
- ROCHA, D. K. F. **Visualização de Requisitos a partir de modelos AOV-Graph**. Natal, 2009. 84p. Monografia. Universidade do Estado do Rio Grande do Norte – UERN.
- SILVA, L. F. **Uma Estratégia Orientada a Aspectos para Modelagem de Requisitos**. Rio de Janeiro, 2006. 222p. Tese de Doutorado em Engenharia de Software. Pontifica Universidade Católica do Rio de Janeiro - PUC-Rio.
- SILVA, L.; BATISTA, T.; SOARES, S.; SANTOS, L. **On the Role of Features and Goals Models in the Development of a Software Product Line.** In: Internacional Workshop on Early Aspects at AOSD'10, 2010.