



# Problem-Oriented Modelling and Verification of Software Product Lines

Andreas Classen

Mémoire présenté en vue de l'obtention  
du grade de maître en informatique

---

22 Juin 2007

## Outline



- 1 Smart Home
- 2 Feature Interaction Detection
- 3 Automation
- 4 Conclusion



# Outline

- 1 Smart Home
- 2 Feature Interaction Detection
- 3 Automation
- 4 Conclusion



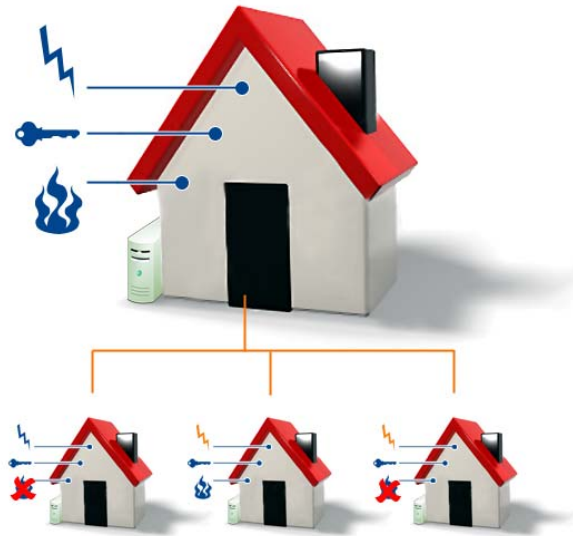
# The smart home example





# Different versions

Localised versions, changes upon client request, ...

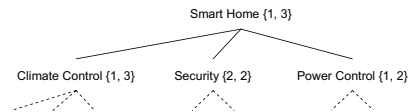


# Perspective 1: Product line



## Product Line

- Focus on feature combinations
- Supporting language: feature diagrams (FDs)



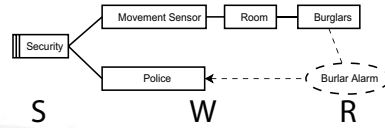
- Which are the products the manufacturer can build/offer ?
- Software product lines approach
- Economies of scale
- Requirements as features in a feature diagram

# Perspective 2: Problem frames



## Problem Frames (PF)

- Focus on environment
- Supporting language: problem diagrams



- R: What is the requirement ?
  - alert police if a burglar breaks in "
- W: How does the world behave ?
  - a burglar causes movement in the room "
  - the movement sensor captures movement in the room "
- S: Derived specification
  - alert police if the sensor captures movement "

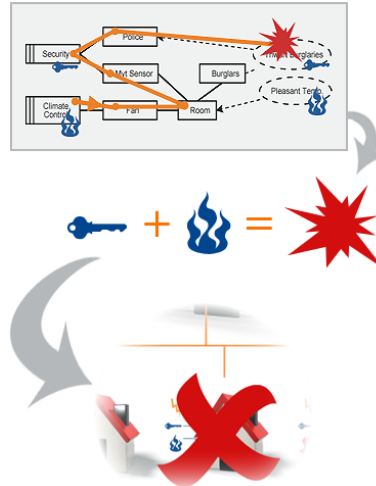
# Outline

- 1 Smart Home
- 2 Feature Interaction Detection
- 3 Automation
- 4 Conclusion

# Feature interaction detection

Goal: Detect **feature interactions**, i.e. incompatibilities between features.

- They may occur
  - solely in the software
  - or in environment
- Complex causal chains involving software and environment
- Idea: detect interactions through formal verification



# Feature interaction detection

Combine previous perspectives



## Problem Frames

- Focus on environment



- More precise than features
- Main correctness argument:  
 $S, W \vdash R$
- Verification possible
- But: no information on products!



## Product Line

- Focus on feature combinations



- "o detailed feature descriptions
- But: information about products

⇒ **Combine both**

# Feature interaction detection Procedure



$$\left. \begin{array}{l} S_{\text{key}}, W_{\text{key}} \vdash R_{\text{key}} \quad \checkmark \\ S_{\text{flame}}, W_{\text{flame}} \vdash R_{\text{flame}} \quad \checkmark \end{array} \right\} S_{\text{flame}}, S_{\text{key}}, W_{\text{flame}}, W_{\text{key}} \vdash R_{\text{flame}}, R_{\text{key}} \quad \times$$

Note: Additional proofs are required, but were omitted for conciseness.

# Outline

- 1 Smart Home
- 2 Feature Interaction Detection
- 3 Automation
- 4 Conclusion

# Automated verification



Next step: Automate the interaction detection approach.

- Approach expressed in four different algorithms
  - Automating the approach means automating these algorithms
- ⇒ Need for an automatable formalism:
- we chose the **event calculus**
  - with its **Decreasoner** implementation

# The event calculus

Illustration on the smart home example



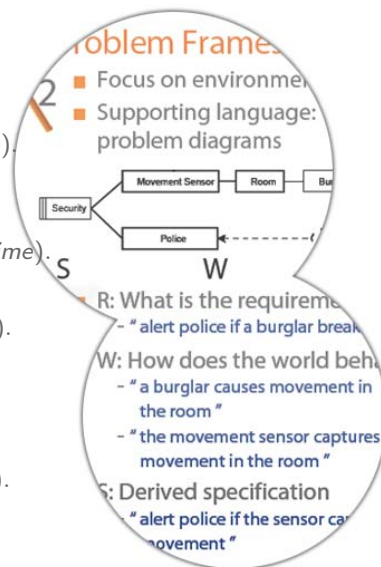
## ■ Requirement

$$\begin{aligned} & Happens(BreakInReq(), time) \\ & \Rightarrow \exists time' . time' \geq time \\ & \quad \wedge HoldsAt(PoliceAlerted(), time') \end{aligned}$$

## ■ Domain description

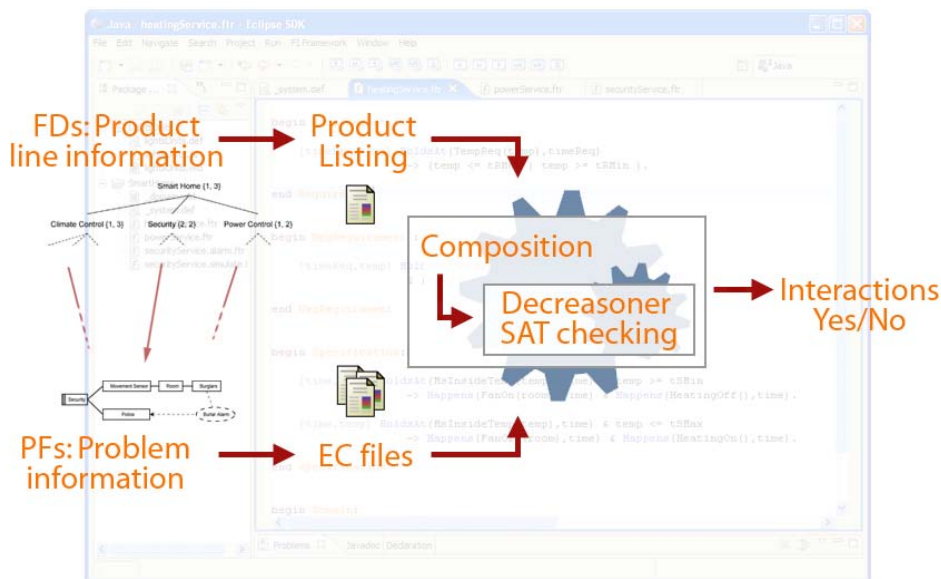
- $Initiates(BreakInReq(), Movement(), time)$ .
- $HoldsAt(Movement(), time)$   
 $\Leftrightarrow HoldsAt(MsMovement(), time)$ .

## ■ Specification

$$\begin{aligned} & HoldsAt(MsMovement(), time) \\ & \Leftrightarrow Happens(AlertPolice(), time) \end{aligned}$$


# Tool Support: FIFramework

A proof-of-concept implementation of the approach



# Application on smart home

Overview of results



- Classic feature analysis followed by a problem analysis
  - A system of four features verified for interactions
  - 27 events and fluents based on shared phenomena in PFs
  - 30 event calculus formulae expressing domain assumptions, requirements and specifications
  - Five files as input for FIFramework
- ⇒ Two interactions detected:
- Burglar alarm vs. climate control
  - Simulating occupancy vs. efficient energy control



# Outline

- 1 Smart Home
- 2 Feature Interaction Detection
- 3 Automation
- 4 Conclusion



# Perspectives

## Opportunities and Challenges

### Opportunities

- Stronger satisfiability notion for feature diagrams
- Analysis of problem variability
- Support for systematic modularisation
- Enriched feature diagrams

### Challenges

- Existence of a mapping between both
- Impact of pre-existing environments
- Scalability to large systems



The contribution of this work is an approach that

- combines FDs and PFs;
- enriches a FD by a problem analysis;
- allows for a general approach to feature interaction detection in SPLs;
- can be largely automated using the event calculus;
- is feasible, as demonstrated by a proof-of-concept tool implementation;
- is illustrated in detail on a smart home example case;



The End - Thank you for your attention.