Achieving Run-time Evolution of Dynamic Software Product Lines through a Variability Modeling Approach

Lorena Arcega

FOSD 2015
Index

- Introduction
- Background
- Evolution challenges
- Evolution strategy
- Case Study
- Conclusions
Dynamic Software Product Lines (DSPLs) extend existing product line engineering approaches by moving their capabilities at run-time.
Software evolution has emerged as a key research field in software engineering.
However, Dynamic Software Product Lines (DSPLs) evolution has not yet deserved enough attention.
This work addresses the use of DSPLs to achieve the evolution by means of integrating new developed components.
Background

PervML DSL
DSL for describing pervasive systems using high-level abstraction concepts.

Background

PervML Model

Diagram: A diagram illustrating systems labeled as "Illumination Service," "Lights," "Alarm," "Security Service," "Siren," and "Blinking Lights." The connections are labeled as 'a,' 'c,' 'd,' and 'e.' Nodes are marked as 'Service,' 'Device,' and 'Channel.'
Background

Feature Modelling

A widely used formalism for modeling and reasoning about commonality and variability of a system.

Background

Feature Modelling

- **Smart Hotel Room**
  - **Automated Illumination** (1,a,2)
  - **Security** (3,c)
    - **Alarm** (5)
      - **Siren** (d,6)
      - **Blinking Light** (e,7)

Mandatory
Optional
Background

Weavig Model
Background

Variability Model
- Active configuration
- Inactive configurations

Active configuration (Features)
- C1
  - F1, F2
- C2
  - F1, F3
- C3
  - F1, F3, F5

Active Architecture Components (Devices, services and channels)

Architecture Model
- Active components
- Inactive components

Inactive configurations

Context Model + Rules
- Active condition
- Inactive conditions

Active configuration
- C1
- C2
- C3
Background

Model-based Reconfiguration Engine (MoRE)  
Translate context changes into changes in the activation/deactivation of features. Then these changes are translated into reconfiguration actions that modify the system components accordingly.

Background

Model-based Reconfiguration Engine (MoRE)
Evolution Challenges

Challenge 1

Co-evolution as in SPLs but maintaining more models such as the context model and the reconfigurations.
Evolution Challenges

Challenge 2

Keep the models partially connected to the running system and keep the interaction between the running system and the context throughout the evolution.
Evolution Challenges

Challenge 1

Co-evolution: If the assets evolve the variability specification must to evolve and vice versa.
Challenge 2

Keep the models partially connected to the system and keep the interaction between the system and the context.
Evolution Strategy
Evolution Strategy

DSPL Models V2

C2 → C4 → C3

M

A

P

E

Running Configuration

DSPL Models V1

C1 → C0, C2, C3

Partially connected

Running Configuration

Interaction

Devices - Services - People

www.usj.es
Evolution Strategy

• Configuration 2 of DSPL Models V2 is the same that Configuration 2 of DSPL Models V1.
There are transitions from the running configuration to some configurations involved in the new models.
Case Study

• Smart Hotel case study:
  – Simulated environment.
  – Average occupancy of about 18 simultaneous clients.
  – MoRE reconfigures the system following the context changes triggered by the clients.
Case Study

• Evolution in the Smart Hotel:
  – Eight versions were developed at design-time.
  – Initial derivation from version 1 to version 8.
  – MoRE reconfigured the architecture of the system.
  – An evolution to one of the next versions was performed.
This work address the evolution of a DSPL by integrating newly developed components.
Conclusions

The evaluation of our strategy in the Smart Hotel DSPL has shown that the models were evolved while the current configuration of the Smart Hotel kept running.
Lorena Arcega
FOSD 2015

larcega@usj.es

http://eps.usj.es/svit/