Configurator-as-a-Service: Tool Support for Deriving Software Architectures at Runtime

Varvana Myllärniemi, Mikko Ylikangas, Mikko Raatikainen, Jari Pääkkö, Tomi Männistö

Aalto University, Finland

Timo Aaltonen

Nokia Research Center, Finland
Background

• Software variability is often managed with a model-based approach
  – Variability is captured in models that represent, e.g., product features or components
  – Individual variants are then derived by binding variability in such models

• Certain situations require dynamic variability
  – Examples: mobile device context changes, customization of web services, newly discovered devices in smart spaces...
  – Dynamic variability benefits from model-based approach, since cannot assume human involvement in derivation
  – Tool support with reasonable performance is necessary
Contribution

• Concepts and implementation of a Configurator-as-a-Service (CaaS), a tool for deriving product variants at runtime
  – Operates on configuration models that describe dynamically varying product architectures from feature and component viewpoint
  – Given a configuration model and requirements for a particular product variant, finds a complete and consistent model of a product variant
  – Implemented as a stateless web service
  – Utilizes efficient *smodels* inference engine for configuration finding, which is a computationally complex task
Modelling architecture and its variability in CaaS

• Firstly, CaaS separates a configuration model and configuration
  – Configuration model: a model with variability that describes all possible product variants
  – Configuration: a model of one product variant, derived from the configuration model by binding variability

• Secondly, CaaS uses Kumbang language to capture both configuration models and configurations
  – Kumbang synthesizes concepts from many feature modelling methods, Koala and other ADLs, and traditional product configuration
  – Kumbang models products from two viewpoints
    • Features
    • Components (and required / provided interfaces)
Configuration model & configuration
Features & components

Configuration model

MapFeatures

show[0…1]

ShowTaxis
ShowRestaurants

Implementation constraints

MapViewerApplication

main

mapSource

infoSource[0…1]

MainModule

MapSource

infoSource[0…1]

TaxiInfoSource

RestaurantInfoSource

Configuration

:MapFeatures

show[1]

:ShowTaxis

:MapViewerApplication

:MapSource

:MainModule

:TaxiInfoSource
Overall CaaS design

Runtime derivation engine
- Utilize a predefined configuration model / Generate a configuration model
- Identify requirements for the configuration and variants that must be present
- Instantiate configuration

CaaS
- Translate to WCRL; Ground; Check model consistency
- Check consistency; Find consequences; Find complete configuration

Legend
- System
- Artifact
- Activity

(1): Model upload
- Configuration model

(2): Derivation
- Configuration selections
- Consistent and complete configuration
Separation of CaaS and runtime derivation engine

- **Runtime derivation engine**
  - Utilize a predefined configuration model / Generate a configuration model
  - Identify requirements for the configuration and variants that must be present
  - Instantiate configuration

- **CaaS**
  - Translate to WCRL; Ground; Check model consistency
  - Check consistency; Find consequences; Find complete configuration;

Legend:
- System
- Artifact
- Activity

Specific to the dynamic setting
Computationally complex, generic
Phase 1: configuration model is created and uploaded
Phase 2: deriving a product variant

Runtime derivation engine
- Utilize a predefined configuration model / Generate a configuration model
- Identify requirements for the configuration and variants that must be present
- Instantiate configuration

CaaS
- Translate to WCRL; Ground; Check model consistency
- Check consistency; Find consequences; Find complete configuration

Legend
- System
- Artifact
- Activity

(1): Model upload
- Configuration model

(2): Derivation
- Configuration selections
- Consistent and complete configuration
Deriving a product variant, example

Configuration model

MapFeatures

ShowTaxis  ShowRestaurants

Implementation constraints

MapViewerApplication

MainModule  MapSource

infoSource[0…1]

TaxiInfoSource  RestaurantInfoSource

Configuration selections

:ShowTaxis

Resulting complete and consistent configuration

:MapFeatures

show[1]

:MapViewerApplication

:MapSource

:MainModule

:TaxiInfoSource
Finding a complete and consistent configuration

- A consistent configuration ≈ no rules of the configuration model are violated
- A complete configuration ≈ all necessary selections have been made
- In the worst case, this task requires at least an exponential amount of time in relation to the size of the problem
- CaaS utilizes smodels, which is a general-purpose inference engine based on stable model semantics of logic programs
  - Configuration models are translated into weight constraints, and consistent and complete configurations are calculated as stable models of such logic programs
Application of CaaS

• Our Configurator-as-a-Service can be utilized for runtime derivation if
  – The varying product architecture can be modelled with Kumbang, that is, via features and component with interfaces
  – A separate runtime derivation engine exists or can be developed

• We have evaluated the concepts and implementation of CaaS with the case of Social Devices
  – Mobile devices that are able to perform interactive user-visible actions whenever other social devices are near by
  – CaaS is utilized for finding actions and device sets that can perform that action
Conclusions

• CaaS enables deriving product variats at runtime
  – Models variability and software architectures from feature and component viewpoint
  – Derivation utilizes inference engine *smodels*

• Benefits
  – Separates computationally complex yet generic functionality of runtime derivation from the functionality specific to the particular dynamic setting
  – As a stateless web service, easy to integrate with any runtime derivation framework, and easy to call at runtime

• Current implementation returns only one consistent and complete configuration out of potentially many configurations
  – We are currently working on utilizing recommendation techniques to select the best or good enough configuration among all possible consistent and complete configurations