Kumbang Modeler: A Prototype Tool for Modeling Variability

Hanna Koivu, Mikko Raatikainen, Marko Nieminen, Tomi Männistö

Helsinki University of Technology (TKK) Finland
Content

- Background: Kumbang, software product family, and feature-component modeling
- Method: Design Science & User centered design
- Result: A prototype tools to model variability
- Lessons learned: General, usability, variability modeling
Background: Kumbang

- Kumbang is a *conceptualisation* (domain ontology), a *language*, and *tools* for configurable applications developed at TKK.
- Kumbang provides concepts for modeling variability from two viewpoints adhering to IEEE 1471-2000 standard:
  - The user-visible characteristics of individual products, i.e., *features*.
  - The *architecture* of the products in terms of components etc.
  - In addition, interrelations between the views can be specified.
- Differentiates between family and instance.
- Kumbang is provided a formal semantics by defining a mapping from the ontology to weight constraint rule language.
- Tool support: *Kumbang Configurator* for resolving the variability in a product family to meet the specific set of requirements at hand.
- Timo Asikainen *et al.*: SPLC’06 and Advanced Engineering Informatics, 21(1), 2007 (http://www.soberit.hut.fi/svamp/)
Software product family

Product family architecture

Shared assets

PF development

derivation

Product individuals

© Mikko Raatikainen, 2007
Feature model and component structure

- Text editor
  - Equation editor
    - EqEdit
    - MathPal
  - Clipboard
    - Single-item clipboard
    - Multi-item clipboard
      - capacity: \{3, 5, 9\}
Method

▪ Design science methodology
  ▪ A research method common in IS research to construct new and innovative artifacts

▪ User-centered design
  ▪ Goal-directed design, especially Personas
  ▪ Feasibility test and two light-weight usability test
Kumbang Modeler: Eclipse plug-in

- Eclipse is a popular development environment
  - Many developers are familiar with Eclipse
- Eclipse plug-ins
  - Eclipse plug-ins are currently very popular
  - Easy to install
  - Java based; relatively easy to develop
Example dialogs

New Feature Type

- Feature Type Name:
- Comment (optional):
- Supertypes:
- Add...
- Remove
- Type is abstract

Add Subfeature Definition

- Subfeature Name:
- Comment (optional):
- Possible Feature Types:
- SubFeature
- Add...
- Remove
- Cardiinality:
  - Minimum: 1
  - Maximum: 1
- Similarity:
  - No further restrictions on chosen types
  - All chosen types must be of the same type
  - All chosen types must be of different types

© Mikko Raatikainen, 2007
Usability tests

- Follow a web store scenario to construct a model
- The first user, who knew Kumbang very well, had very little trouble making a model according to the scenario
  - Some suggestions for improving the user interface
- The second, who had no previous knowledge of Kumbang, had trouble understanding the need for relation between types and definitions used for compositional structure
  - Inconveniently repetitive information
  - This led to user interface simplification
Lessons Learned: General

- Eclipse feasible platform for plug-ins
  - Familiar to use, easy to install, and easy to distribute
- Modeler makes easier to construct model
  - New features such as advanced checks for consistency and component diagram are under development
Lessons Learned: Usability

- User-centered design was relatively successful approach
  - New point of view to tool development
  - Not much additional work
  - Difficulties in application such as information for personas could not be directly found
- Strict adherence to all user centered guidelines was not reasonable
  - Most of them valuable although at first seemed a bit awkward such as personas
  - For example, goal differentiation was not feasible for a prototype tool
- The usability tests were relatively light-weight ones
  - More usability tests are needed
Lessons Learned: Variability Modeling

- Balancing between conceptual clarity and easy to use
  - Difficulties in usability tests were mainly because of overly complex modeling constructs for representing simple variability
  - For example, simple optional features should be easy to add
- More empirical studies needed of the nature of variability
Questions?

mikko.raatikainen@tkk.fi