17-708 SOFTWARE PRODUCT LINES: CONCEPTS AND IMPLEMENTATION

FEATURE AND DECISION MODELING

CHRISTIAN KAESTNER
CARNEGIE MELLON UNIVERSITY
INSTITUTE OF SOFTWARE RESEARCH
LEARNING GOALS

define the relevant terms: product line, feature, concern, option, feature selection, feature dependency, product, domain, variant

understand why a product line targets a specific domain,
model features and feature dependencies by means of feature models,

tradeoffs among representations
WHAT IS A FEATURE?

Feature
Concern
Configuration Option
Configuration
Configuration Space
Constraint
Variant
Product
IN-CLASS EXERCISE

List 10 features in domain X
SOFTWARE PRODUCT LINES

A software product line (SPL) is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way.

Software Engineering Institute
Carnegie Mellon University
WHAT IS A DOMAIN?
DEScribing Configuration Spaces

List of configurations
List of options and textual constraints
Formula
Feature models
Decision models
Tradeoffs
TOOL DEMO: FEATUREIDE
IN-CLASS EXERCISE

Organize your features and identify constraints

Create a feature model
Create a corresponding formula
Estimate the number of configurations
DOCUMENTING FEATURES

Description of a feature and its corresponding (set of) requirements
Relationship to other features, especially hierarchy, order, and grouping
External dependencies, such as required hardware resources
Interested stakeholders
Estimated or measured cost of realizing a feature
Potentially interested customers and estimated revenue
Configuration knowledge, such as ‘activated by default’
Configuration questions asked during the requirements analysis step
Constraints, such as “requires feature X and excludes feature Y”
All kinds of behavioral specifications, including invariants and pre- and post-conditions
Known effects on non-functional properties, such as “improves performance and increases energy consumption”
Rationale for including a feature in the scope of the product line
Additional attributes, such as numbers and textual parameters, used for further customization during product generation
Potential feature interactions
CASE STUDIES:
KCONFIG, PURE::VARIANTS
DECISION MODELING

<table>
<thead>
<tr>
<th>dimension</th>
<th>decision modeling</th>
<th>feature modeling</th>
<th>Kconfig</th>
<th>CDL</th>
<th>CVL initial</th>
</tr>
</thead>
<tbody>
<tr>
<td>applications</td>
<td>variability modeling; derivation support</td>
<td>diverse applications: concept modeling (e.g., domain modeling), variability and commonality modeling; derivation support</td>
<td>modeling variability in the kernel; derivation support</td>
<td>modeling variability in eCos; derivation support</td>
<td>variability modeling; derivation support</td>
</tr>
<tr>
<td>unit of variability</td>
<td>decisions to be made in derivation</td>
<td>features are properties of concepts, e.g., systems</td>
<td>drivers, subsystems, kernel options, build option</td>
<td>drivers, subsystems, kernel options, build option</td>
<td>VSpect: essentially decisions in derivation; pre-made decisions (mandatory features)</td>
</tr>
<tr>
<td>orthogonality</td>
<td>orthogonal</td>
<td>mostly used in orthogonal fashion</td>
<td>orthogonal (added configuration UI concepts, e.g., packages, components)</td>
<td>orthogonal (but admitting non-orthogonal uses is discussed)</td>
<td></td>
</tr>
<tr>
<td>data types</td>
<td>comprehensive set of basic types: composites: sets, records, arrays</td>
<td>comprehensive set of basic types: references; composites: via hierarchy, group and feature cardinalities</td>
<td>Boolean, tristate, numbers and strings; choices</td>
<td>choices; parameters with comprehensive set of basic types; classifiers</td>
<td></td>
</tr>
<tr>
<td>hierarchy</td>
<td>secondary concept; diverse approaches, e.g., visibility or relevance hierarchy (no decomposition)</td>
<td>essential concept; single approach tree hierarchy modeling, parent-child configuration constraints and decomposition</td>
<td>characteristics of FM&amp;DM: essential organization means (FM), visibility induced, driven by UI concepts (DM)</td>
<td>like in FM (essential organization means; decomposition hierarchy)</td>
<td>essential concept; vspec tree, like in FM</td>
</tr>
<tr>
<td>dependencies and constraints</td>
<td>no standard constraint language but similar range of approaches (Boolean, numeric, sets)</td>
<td>no standard constraint language but similar range of approaches</td>
<td>propositional three-valued logics with comparisons</td>
<td>propositional Boolean logics with expressions on data</td>
<td>propositional and predicate logic with expressions on data</td>
</tr>
<tr>
<td>mapping to artifacts</td>
<td>essential concept; no standard mechanism</td>
<td>optional concept; no standard mechanism</td>
<td>mapping to C preprocessor via a custom build system (no explicit mapping model)</td>
<td>explicit mapping in the variability model; variability symbols available to C preprocessor</td>
<td>essential concept; mapping model, base-model independent</td>
</tr>
<tr>
<td>binding time and mode</td>
<td>not standardized, occasionally supported</td>
<td>not standardized, occasionally supported</td>
<td>static or dynamic binding decided at compile time</td>
<td>static or dynamic binding decided at compile time</td>
<td>not included in CVL (dependent on application)</td>
</tr>
<tr>
<td>modularity</td>
<td>no standard mechanism; decision groups play partly this role</td>
<td>no standard mechanism; feature hierarchy plays partly this role</td>
<td>model is split into files; no modularization beyond hierarchy in the language</td>
<td>loadable packages, reparenting</td>
<td>explicit support (packages, configurable units)</td>
</tr>
<tr>
<td>tool aspects</td>
<td>representation of models as lists, tables, trees, and graphs; configuration UI: an (ordered) list of questions</td>
<td>representation of models as lists, tables, trees, and graphs; configuration UI: a (unordered) list of questions</td>
<td>modeling in textual configuration UI; syntax; configuration UI; context; variables; vis</td>
<td>modeling in textual configuration UI; syntax; configuration UI; context; variables; vis</td>
<td>user interfaces are the domain of vendors; basic aspects of variability and configuration</td>
</tr>
</tbody>
</table>
ADOPTION
PATHS


