three cases of feature-based variability modeling in industry

thorsten berger, divya nair, ralf rublack, joanne m. atlee, krzysztof czarnecki, andrzej wasowski
variability is everywhere
variability adds complexity

code

```c
76    check_range(unsigned long vstart, unsigned long vend,
77        unsigned long kstart, unsigned long kend)
78    {
79        unsigned long vaddr, kaddr;
80
81    #ifdef DEBUG_CHECK_RANGE
82        sm_printf("check_range: \%lx \%lx \%lx \%lx \n", 
83                vstart, vend, kstart, kend);
84    #endif
85    /* do some range checking for detecting an overlap... */
86    for (vaddr = vstart; vaddr <= vend; vaddr += PAGE_SIZE)
87    {
88        kaddr = (find_pa(vaddr) | PAGE_OFFSET);
89        if (kaddr > kstart && kaddr <= kend)
90            {
91    #ifdef DEBUG_CHECK_RANGE
92        sm_printf("OVERLAP: vaddr \%lx kaddr \%lx\n", 
93                vaddr, kaddr);
94    #endif
95            return 1;
96    }
97    }
98    return 0;
99    }
```

requirements

![requirements diagram]

architecture

![architecture diagram]

tests

![tests diagram]
variability modeling

(toy) feature model
survey findings

large diversity of tools

industry lacks guidance

modeling challenges

quantitative → qualitative

among 42 survey participants

conducted 8 semi-structured interviews (1-1.5h)

this paper: 3 described/analyzed in-depth
research questions

practices?

benefits?

challenges?
subject selection

Development scales:
- Small (2 developers)
- Large (60 developers)
- Ultra-large (100 teams)

Domains:
- eCommerce
- Industrial applications/energy
- Automotive

Product line adoption:
- Reactive
- Extractive
- Proactive

Consulting company (≤50 employees)
Component producer (≤25,000 employees)
Car manufacturer (≤150,000 employees)
MODELING CONTEXT
A: consulting company

~40 features

CaptainFeature

home-grown generator/preprocessor

web shop
B: component producer

C/C++ code

requirements

test cases

pure::variants

~1,100 features

power electronics firmware
C: car manufacturer

- **C/C++ code**
- **logical design blocks**
- **Simulink models**
- **components**

**semi-structured feature lists**

- **top level**
  - 300-500 features
- **intermediate level**
  - <800 features
- **low level**
  - <3000 features

TeamCenter / Excel

car model
BENEFITS
configuration / code generation?

not primarily configuration!

organization of knowledge!

resembles perceived benefits of MDD*

B: The first one is that it’s visible, you see the features that you had in the code before.

B: Actually, you see the features of the whole product line. Before, they saw features of the specific products.

C: To agree between the R&D organization and with the product planning organization over the content of each product.

B: The same functionality was implemented twice [...] They implemented the same features.
PRACTICES
who edits the models?

centralized model governance

B: We have a colleague who [...] really has the domain knowledge.

B: Whenever we have an issue, we try to organize a workshop or a meeting.

C: On the top level, it's centralized, [maintained by] a central group.

bad news for distributed modeling
how to build the hierarchy?

result of domain analysis (top-down) and evolution (bottom-up)

A: We always looked at it from the perspective of what we can sell.

exploit domain knowledge

A: We tried to come up with logical relationships between the features.

limits possible automation

CatalogSystem

ShoppingCart
subjects avoid modeling constraints

configuration?

A: done by consultant

B: maintain set of tested configurations

C: constraints checked at manufacturing

A: You need a consultant to tell [the customer] what he needs.
constraints?
yet, all have constraints!

B: We started adding them. But it’s very few.

B: We started adding relationships like “recommended”, because we have defect features.
evolution?

primarily addition / rare removal

stable model hierarchies

versioning of the model, not individual features
CHALLENGES
short-term versus long-term benefit

organizational pushback in a matrix organization

B: It’s some kind of a strained situation [between] product development [and] the technology people.
developer motivation and organization

B: Developers are used to working for a long time on the same abstraction level.

C: We have a lot of dependencies between teams, so it’s quite difficult for the teams to work autonomously.
SUMMARY
key take-aways

benefits
  organization of knowledge
  collaboration
  configuration

pragmatic practices
  centralized governance
  versioning of the model
  limited constraint modeling

challenges
  acceptance of abstraction layer
  organizational pushback
  dependencies between teams
future work

static analysis infrastructure (FarCE) to recover constraints (*)

incremental adoption of product lines (**) study feature identification and coordination dynamics

investigate other units of variability

study failed attempts

*) Nadi, Berger, Kästner, Czarnecki: Mining Configuration Constraints: Static Analyses and Empirical Results. ICSE. 2014
thanks for your attention

three cases of feature-based variability modeling in industry

thorsten berger, divya nair, ralf rublack, joanne m. atlee, krzysztof czarnecki, andrzej wasowski