Feature Interactions: the Good, the Bad, and the Ugly

Jo Atlee • U Waterloo • Dec 2013

Sandy Beidu
Shoham Ben-David
Cecylia Bocovich
Jonathan Hay
Pourya Shaker

WATFORM
David R. Cheriton School of Computer Science
University of Waterloo
feature-oriented software development

**feature**: a unit of *functionality* or *added value* in the product

stakeholders’ mental model of system

feature-oriented software system
product lines

feature model = valid configurations

reusable implementations

products

[ Example from Sven Apel ]
product lines
feature interactions

feature interaction: features influence each other in defining overall system behaviour [Zave]

› conflicts over shared context
› violations of global correctness properties
› emergent behaviours

feature interaction problem: the number of potential interactions is exponential in the number of features
what this talk is about

modelling feature requirements
  › feature modularity
  › modelling intended interactions

analyzing feature combinations
  › to detect interactions

resolution strategies
  › strategies that avoid classes of interactions
the good
not all interactions are bad!

intended interactions
  › advanced cruise-control variants override basic cruise control
  › prohibit navigation overrides navigation
  › prohibit-navigation override overrides prohibit-navigation

unintended but harmless interactions
  › call screening prevents activation of caller id

(planned) resolutions to conflicts
  › brake override overrides (acceleration ⊕ braking)
feature-oriented requirements modelling language (FORML)
Shaker, Atlee, Wang, RE’12

a notation for modelling the requirements of a product line (PL)

› supports feature modularity
› provides language constructs for expressing intended interactions explicitly
› composes features into a product line
req models decomposed by feature

**world model**
a conceptual model of the problem world
- defines possible world states
- including feature phenomena

**behaviour model**
state-machine models (of features)
- whose events, conditions and actions are expressions over world phenomena
- and over feature phenomena
modelling features

features are modelled as hierarchical state machines that sense and control the world

transition labels:
id: e [c] / id_1: [c_1] a_1, ..., id_n: [c_n] a_n
• triggering event: a change in the world
• guard condition: predicate over the world
• action: a prescribed change to the world
• transition or action name
a new feature may...

---

**introduce behaviours**
- via: new machines

**eliminate behaviours**
- via: new or stronger enabling conditions on existing actions or transitions

**substitute behaviours**
- via: new *pre-empting* actions or transitions

---

*can also be expressed as extensions to existing features:*
- new regions, new states, new transitions, weakened enabling conditions

---

**intended interactions:**
modelled as structural extensions at extension points in existing features
adding behaviours

Cruise Control (CC)

extends BDS state

new region

state-machine extension

transition BDS(t3): [strengthen with c: not inState(main.enabled.main.engaged.main.active) or driverOverride()]

BDS{main.on}
replacing behaviours

Headway Control (HC)

new region includes pre-empting transition: models HC intentionally prohibiting CC

extends CC state
composition is a product line

transitions, actions, clauses are guarded by presence conditions (of their declaring feature)

product line = \{BDS, BDS + CC, BDS + CC + HC\}
summary of FORML

• precise modular modelling of features
• new features extend existing features
  › with added, removed, and replaced behaviours
• explicit modelling of intended feature interactions
• result of feature composition is a product line
the bad
hybrid brakes ⊕ anti-lock breaking

2010 Toyota Prius

hybrid brake system
  › (normal) hydraulic brake system
  › regenerative braking system
    – converts loss of vehicle momentum into electrical energy
    – stored in on-board batteries

anti-lock brake system (ABS)
  › maintains stability, steerability during panic braking

interaction
  › braking force after ABS actuation reduced
  › vehicle stopping distance is increased
  › 62 reported crashes, 12 injuries
cruise control ⊕ traction control

cruise control
  › vehicle set to maintain driver-specified speed

traction control
  › brake fluid applied when wheels slip

interaction
  › engine power is increased (to maintain speed)
  › driver senses “sudden acceleration”
    - vehicle becomes difficult to control

resolution
  › advise drivers not to use cruise control on slippery roads
feature interaction

\[ F_1 \models \Phi_1 \]
\[ F_2 \models \Phi_2 \]
\[ \vdots \]
\[ F_n \models \Phi_n \]

\[ F_1 \oplus F_2 \oplus \cdots \oplus F_n \not\models \Phi_1 \land \Phi_2 \land \cdots \land \Phi_n \]

feature composition (= product)
model checking
Clarke, Emerson ‘81, Queille, Sifakis ‘82
detecting feature interactions

\[ F_1 \oplus F_2 \oplus \cdots \oplus F_n \]

Software Model

Model Checker

Property holds?

YES

Stop

NO

Counter Example

Property

\[ \phi_1, \phi_2, \ldots \]
product-line model checking
Classen, Heymans, Schobbens, Legay, Raskin, ICSE’10

1. Property
2. Product Line Model
3. PL Model Checker
4. Property holds?
5. Invalid Configurations + Counter Example
6. Stop
properties should...

- reflect each feature’s desired behaviour
- be conditional on whether a feature is present
- accommodate intended interactions
  › which affect whether a transition executes

\[
t : \text{ev} \left[ f \land \text{cond} \land ( g \Rightarrow \text{cond2} ) \right] / x := \text{val}
\]
properties

A property for each transition in the PL model:
› if transition executes, the effects of its actions are realized
› can be generated automatically from PL model

\[ AG \left( t_{\text{execute}} \rightarrow AX( x = \text{val} ) \right) \]
FORML PL model → Translator → Model & properties to be analyzed → Model checker

- Rich data types
- Complex multi-step execution semantics

- Boolean data types
- Simple execution semantics
summary of interaction detection

• properties can be generated automatically from the PL model

• analyzer checks every property in all behaviours of all products in product line

• analyzer identifies, for each property, all products in which the property can be violated

• only unintended interactions will be reported
the ugly: scalability
lots of features

e.g., telephony has 1000+ features per system

a system of feature-rich systems

› features from multiple providers
› multiple active versions of the same feature
lots of interactions

results of the second feature interaction contest
lots of types of interactions

control-flow
one feature affects the flow of control in another feature

data-flow
one feature affects (deletes, alters) a message destined for another feature

data modification
shared data read by one feature is modified by another feature

data conflict
two features modify the same data

control conflicts
two features issue conflicting actions

assertion violation
one feature violates another feature's assertions or invariants

resource contention
the supply of resources is inadequate, given the set of competing features
introduced in several phases
Bowen, SETSS’89

[req] understanding / specifying how features ought to interact

[req] the number of interactions (and resolutions) to consider grows exponentially with the number of features

[design] more interactions introduced during design due to sharing of resources, I/O devices, protocol signals, etc.

[imp] near-commonalities among features leads to questions about how to effectively reuse software components

[test] the sheer number of interactions and resolutions to be tested lengthens the testing phase
wicked problem

- lots of features
- lots of interactions
- multiple types of interaction
- lots of resolutions
- introduced in several phases

resolve interactions through feature composition

- compose features into products (or product lines)
- composition algorithm resolves entire classes of interactions
conflict-free composition
Hay, Atlee, FSE’00

resolution strategy: maximal subset of enabled transitions with nonconflicting actions
› uses feature priority to resolve conflicts

\[
F_1 \oplus F_2 \approx F_1 \oplus F_2 \\
A, X \\
A, Y \\
B, Y \\
B, X \\
t_1 \quad t_{1\&t2} \quad t_2
\]
violation-free composition
Hay, Atlee, FSE’00

**resolution strategy:** maximal subset of enabled transitions with *nonconflicting* and *nonviolating* actions

4 classes of interactions
- actions conflict
- actions violate assertions
- new assertions not satisfied
- new assertions conflict

resolution
- resolve by priority
- apply transition
feature coordination

- fixed set of features
- pre-determined selection of features
- static integration
- perfect coordination possible
- unlimited features
- semi-configurable selection of features
- set of static integrations
- perfect coordination possible, but impractical
- user-defined selection of features
- dynamic integration
- loose coordination
model features modularly with intended interactions

resolve classes of undesired interactions through composition

detect remaining unintended interactions