Motivation

These deltas require changes outside the feature model (different artifacts coevolve)

$\Delta_1$: YRS-M1 build rules and related source files are deleted; YRS-M2 source code incorporates the prediction code from YRS-M1 (no overall functionality is lost)

$\Delta_2$: Similar to the previous change, but the merge occurs towards the parent feature, and a rename also takes place

The Importance of Coevolution (Example Scenario)

Does the bug exist in t_1? Does the bug exist in t_2? Bug found in YS!

If coevolution among different artifacts is not taken into account, one cannot trace YS back to YRS-M2 at t_2, nor to YRS-M1 and YRS-M2 at t_1

From the feature model alone, one is led to conclude that YRS-M1 is removed at t_2, and that YRS-M2 is removed at t_3

Coevolution Patterns

A high-level representation of a change in place, and how different artifacts evolve as a result

Case Study from a Large and Complex System

Manual analysis over 500 commits relative to the addition and removal of features, while keeping track of how the Linux feature model, mapping (Makefiles) and implementation assets (C code) coevolve

Approach

Pattern Example

CTC

$M = \{f_1, f_2, \ldots\}$

$M' = \{f_1, f_2, \ldots\}$

I

$\{f_1, \ldots, f_2, \ldots\}$

with $f_2 > f_1$

Explains the merge from $\Delta_1$

Future Work

Verify the coverage of our patterns in systems other than Linux

Publications
