Workshop on Configuration

Vienna, Aug. 29th-30th, 2013

Towards Anomaly Explanation in Feature Models

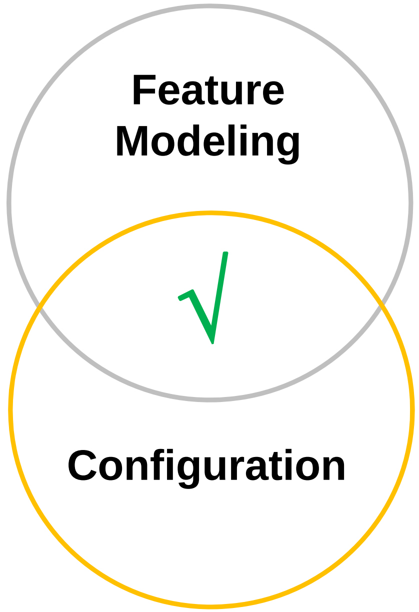
Alexander Felfernig[1], David Benavides[2], José Galindo[2], and Florian Reinfrank[1]

[1]Graz University of Technology, Austria

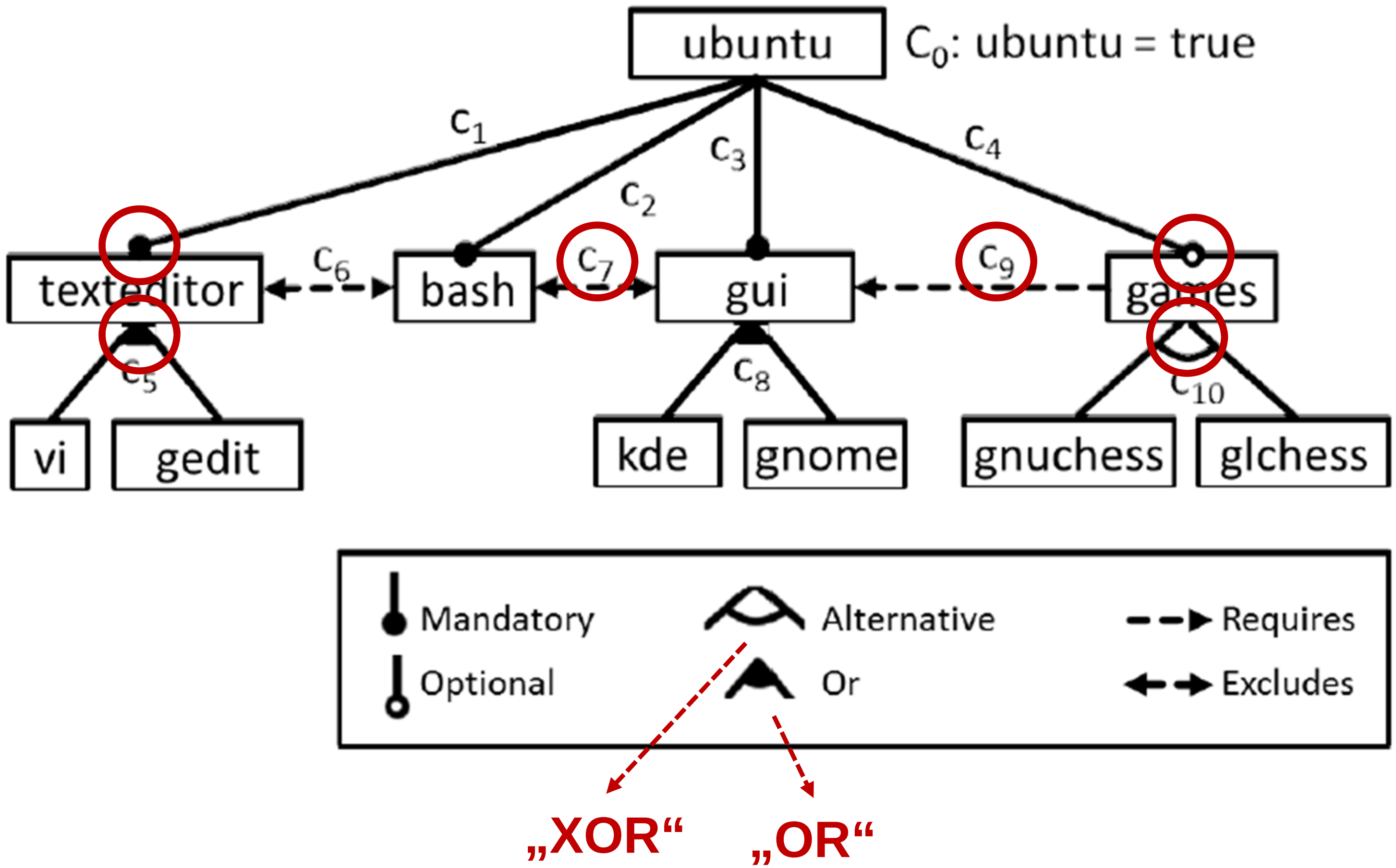
[2]University of Seville, Spain

Overview

- Introduction
  1. Feature Models (FMs): Modeling Concepts
  2. FMs: Configuration Task Definition
  3. FMs: Analysis Operations
- Testing & Debugging
  4. Configuration Models: Testing & Debugging
  5. FM Analysis Operations as Test Cases
  6. FM Analysis Operations & Explanations
- Ongoing & Future Work



Feature Models (FMs): Modeling Concepts



FMs: Configuration Task Definition

Definition 1 (FM Configuration Task). A feature model (FM) configuration task is defined by the triple (F,D,C) where $F = \{f_1, f_2, ..., f_n\}$ is a set of features f_i, $D = \{dom(f_1), dom(f_2), ..., dom(f_n)\}$ ($dom(f_i) = \{true, false\}$) is the set of corresponding feature domains, and $C = CR \cup CF$ is a set of constraints restricting the possible configurations which can be derived from the feature model. In this context, $CR = \{c_1, c_2, ..., c_k\}$ represents a set of requirements (of a specific user) and $CF = \{c_{k+1}, c_{k+2}, ..., c_m\}$ a set of feature model constraints.

Configuration Task: Example

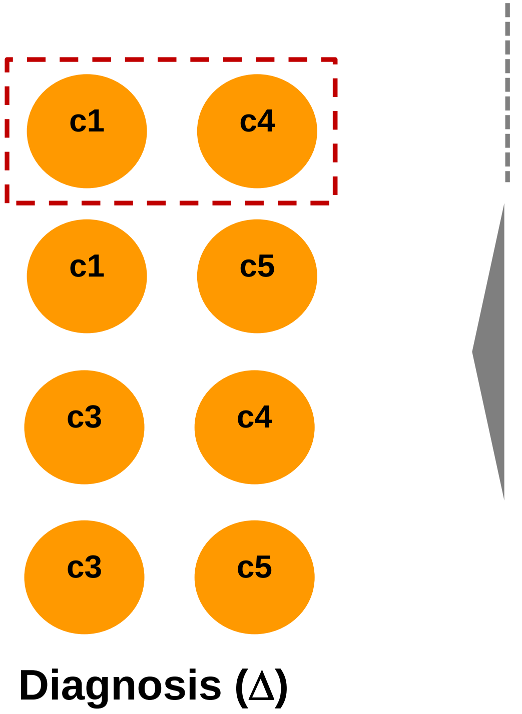
- $F = \{ubuntu, texteditor, bash, gui, games, gedit, vi, kde, gnome, gnuchess, glchess\}$
- $D = \{dom(ubuntu) = \{true, false\}, dom(text-editor) = \{true, false\}, dom(bash) = \{true, false\}, dom(gui) = \{true, false\}, dom(games) = \{true, false\}, dom(gedit) = \{true, false\}, dom(vi) = \{true, false\}, dom(kde) = \{true, false\}, dom(gnome) = \{true, false\}, dom(gnuchess) = \{true, false\}, dom(glchess) = \{true, false\}$
- $CR = \{c_0\text{: } ubuntu = true\}$
- $CF = \{ c_1 : ubuntu \leftrightarrow texteditor, c_2 : ubuntu \leftrightarrow bash, c_3\text{: } ubuntu \leftrightarrow gui, c_4\text{: } games \rightarrow ubuntu, c_5\text{: } texteditor \leftrightarrow gedit \vee vi, c_6\text{: } \neg texteditor \vee \neg bash, c_7\text{: } \neg bash \vee \neg gui, c_8\text{: } gui \leftrightarrow kde \vee gnome, c_9\text{: } games \rightarrow gui, c_{10}\text{: } (gnuchess \leftrightarrow \neg glchess \wedge games) \wedge (glchess \leftrightarrow \neg gnuchess \wedge games)\}$

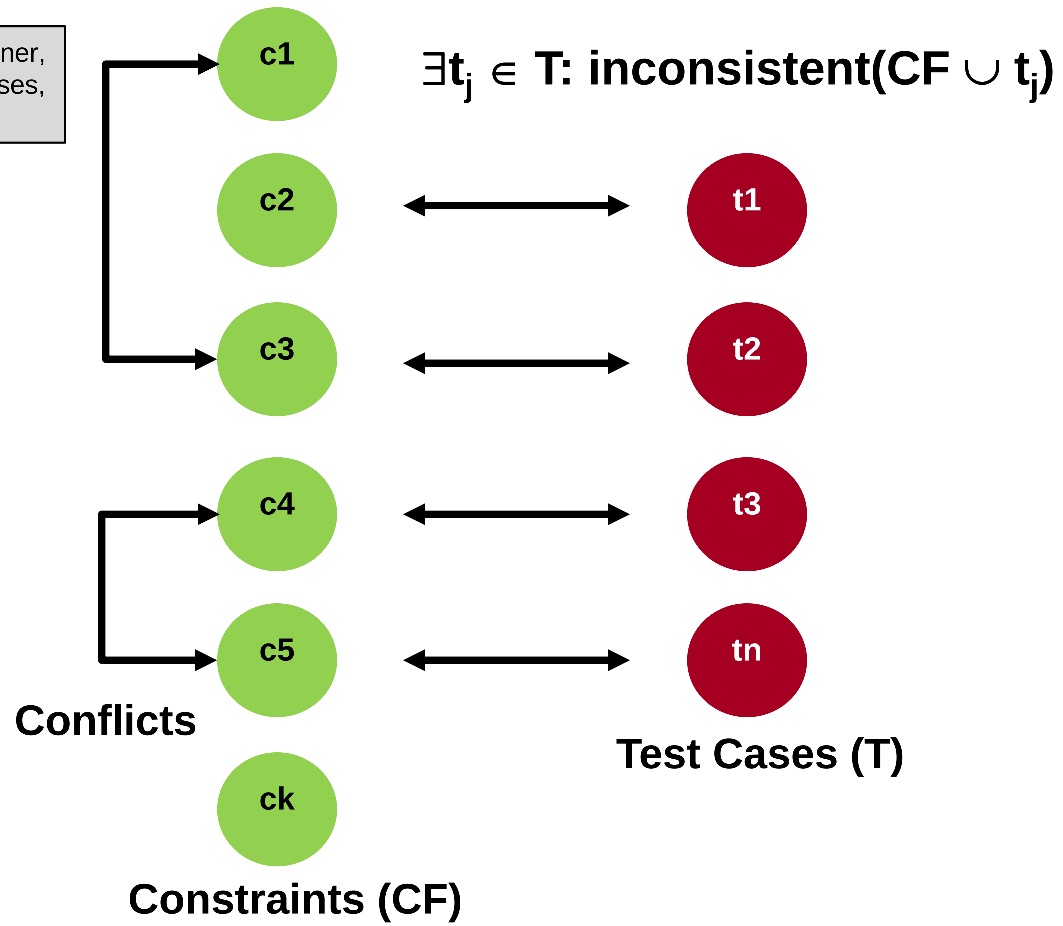
FMs: Analysis Operations

Analysis operation	Property Check
Void feature model	inconsistent(CF $\cup$ $\{c_0\}$)?
Dead (f_i)	inconsistent(CF $\cup$ $\{c_0\}$ $\cup$ $\{f_i$=true$\}$)?
Conditionally dead (f_i)	consistent(CF $\cup$ $\{c_0\}$ $\cup$ $\{f_i$=false$\}$) and consistent(CF $\cup$ $\{c_0\}$ $\cup$ $\{f_i$=true$\}$)?
Full mandatory (f_i)	inconsistent(CF $\cup$ $\{c_0\}$ $\cup$ $\{f_i$=false$\}$)?
False optional (f_{opt})	inconsistent(CF $\cup$ $\{c_0\}$ $\cup$ $\{f_{par}$=true $\wedge$ f_{opt}=false$\}$)?
Redundant (c_i)	inconsistent((CF $\cup$ $\{c_0\}$ - $\{c_i\}$) $\cup$ $\neg$(CF $\cup$ c_0))?

Configuration Models: Testing & Debugging

A. Felfernig, G. Friedrich, D. Jannach, and M. Stumptner, Consistency-based Diagnosis of configuration knowledge bases, in Artificial Intelligence, 152(2), 2004, pp. 213–234.





Explanation $\Delta \subseteq CF$: consistent $(CF - \Delta \cup t_j)\ \forall t_j \in T$

FM Analysis Operations as Test Cases

Example analysis operation:
„Dead feature" $f_i \in F$?

inconsistent $(CF \cup \{f_i = true\} \cup \{c_0\})$

Test Case: $t_j \in T$

t_j: $f_i = true$

Explanation $\Delta \subseteq CF$: consistent $(CF - \Delta \cup \{f_i = true\})$

FM Analysis Operations & Explanations

Analysis operation	Explanation (Diagnosis Task)
Void feature model	$\text{FASTDIAG}(CF, CF \cup \{c_0\})$
Dead (f_i)	$\text{FASTDIAG}(CF, CF \cup \{c_0\} \cup \{f_i = \text{true}\})$
Conditionally dead (f_i)	$CF \leftarrow CF \cup \{f_i = \text{true}\}$
Full mandatory (f_i)	$\text{FASTDIAG}(CF, CF \cup \{c_0\} \cup \{f_i = false\})$
False optional (f_{opt})	$\text{FASTDIAG}(CF, CF \cup \{c_0\} \cup \{f_{par} = true \wedge f_{opt} = false\})$
Redundant (c_i)	$c_i \notin \text{FMCORE}(CF \cup \{c_0\})$

Explanations: Used Algorithms

- Preferred conflicts (minimal)

U. Junker. QuickXplain: Preferred explanations and relaxations for over-constrained problems. AAAI'04, pp. 167–172, 2004.

- HSDAG with test cases

A. Felfernig, G. Friedrich, D. Jannach, and M. Stumptner, Consistency-based Diagnosis of configuration knowledge bases, in Artificial Intelligence, 152(2), 2004, pp. 213–234.

- Preferred diagnoses (minimal): FastDiag

A. Felfernig, M. Schubert, and C. Zehentner. An efficient diagnosis algorithm for inconsistent constraint sets. AIEDAM, 26(1):53–62, 2012.

- Redundant constraints: FMCore

Alexander Felfernig , D. Benavides, J. Galindo, F. Reinfrank. Towards Anomaly Explanation in Feature Models, Workshop on Configuration, pp. 117-124, Vienna, Austria, 2013.

Evaluation

Feature Model: Xerox	#Variables: 172		#Constraints:205			
# Diagnoses	Inconsistency Rate					
	2% (140 diagnoses)		5% (84 diagnoses)		7% (55 diagnoses)	
	FASTDIAG	HSDAG	FASTDIAG	HSDAG	FASTDIAG	HSDAG
1	1638	3354	1260	2996	1740	3023
2	2013	6646	1710	3167	2050	3203
3	2262	12106	1970	9454	2330	9544
4	2434	12355	2180	9536	2580	9654
5	2637	28111	2341	12044	2790	12165
10	3417	69950	2921	64631	3330	65240
20	4758	75317	3911	90715	5010	91726
all	46785	>100000	17301	>100000	10541	>100000

A. Felfernig, M. Schubert, and C. Zehentner. An efficient diagnosis algorithm for inconsistent constraint sets. AIEDAM, 26(1):53–62, 2012.

R. Reiter. A theory of diagnosis from first principles. Artificial Intelligence, 32(1):57–95, 1987.

$$Inconsistency\ Rate = \frac{\#conflicts\ in\ FM}{\#constraints\ in\ FM}$$

Ongoing & Future Work

- Further evaluation of algorithms (ongoing work with University of Seville)
- Additional analysis operations (e.g., taking into account multiplicity bounds)
- Improved prediction of the sources of faulty behavior (e.g., exploitation of eye tracking „confusion patterns“)
- Algorithms for intra-constraint redundancies

Conclusions

- Approach to integrate contributions of “Feature Modeling” and “Configuration” communities
- Diagnosis & redundancy detection as a basis for the explanation of “well-formedness” violations
- Generation of test cases on the basis of feature model analysis operations
- No additional management overheads for the generated test cases
- Not a substitute for “conventional” KB testing!

Thank You!