软件配置的交互式修复

北京大学 熊英飞

合作者

王治

王波、张汉生、王杰、赵海燕、张伟

University of Waterloo University of Namur

北京大学

Leonardo Passos, Steven She, Krzysztof Czarnecki
 Arnaud Hubaux

2014 NASAC

报告人介绍-熊英飞

- 2000~2004, 电子科技大学本科
- 2004~2006,北京大学研究生
 导师:梅宏、杨芙清
- 2006~2009, 日本东京大学博士 - 导师: 胡振江、武市正人
- 2009~2011,加拿大滑铁卢大学博士后
 –导师: Krzysztof Czarnecki
- 2012~,北京大学"百人计划"研究员 研究方向:软件分析、编程语言设计

Variability Models & Configurators



Variability Models

eCos Configurator - Errors

| unnamed3* - eCos Configurati | ion Too | I | | x |
|------------------------------------|--------------|--------------|----------------------------------|---|
| <u>File Edit View Build T</u> ools | <u>H</u> elp | | | |
| 🗋 🖻 🖬 👗 🐚 📾 🐪 👗 | | ? ? | | |
| Configuration | | Item | Property | |
| 🖃 😫 Object Pool | v3_0 | PreloadSize | Requires PreloadSize <= PoolSize | |
| buffer Size (KB) | 4 | | | |
| (Byte) | 512 | | | |
| مع) Pool Size | 8 | Property | Value | |
| 🗆 🔽 Preload | | N L | 10 | |
| ab) Preload Size | 10 | Value | 10 | _ |
| Allocation_Time | | Default | 10 | |
| Startup | | Flavor | data | - |
| ✓ First Access | | Requires | PreloadSize <= PoolSize | - |
| 🗖 Idle | | DefaultValue | 10 | - |

eCos Configurator - Inactive Options

| unnamed3* - eCos Configurati | on Too | I | | x |
|------------------------------------|--------------|---|-----------------------------|-----|
| <u>File Edit View Build T</u> ools | <u>H</u> elp | | | |
| 🗅 🖙 🔚 👗 🖻 💼 🦦 👗 | | ? ? | | |
| Configuration | | Item | Property | |
| 🖃 🚼 Object Pool | v3_0 | ⁰ PreloadSize Requires PreloadSize <= PoolSize | | |
| ab Buffer Size (KB) | 4 | | | |
| ab) Object Size (Byte) | 512 | | | |
| ab Pool Size | 8 | Promonto Value | | |
| 🖃 🔽 Preload | | roperty | Value | - 1 |
| ab) Preload Size | 10 | Macro | Startup | |
| □ □ Allocation Time | | Enabled | False | |
| | | Flavor | bool | = |
| lisabled First Access | | Implements | Allocation_Time | |
| ☐ Idle | | ActiveIf | PreloadSize <= PoolSize / 2 | |

Error resolution and option activation both need to resolve violation of constraint.

Survey

- 97 Linux users and 9 eCos users
- Resolving a violation is hard
 - 20% Linux users need "a few dozen minutes" to activate an option in average
 - 56% eCos users consider activation to be a problem

eCos Configurator

| unnamed3* - eCos Configurati | on Too | B Resolve conflicts | | ? × |
|------------------------------|--------------|---------------------|-----------------|--------------------|
| <u> </u> | <u>H</u> elp | | | Continue Cancel |
| 🗅 😅 🖬 👗 🖬 📾 🐪 👗 | | | | |
| Configuration | | Item | Property | |
| 🖃 🚼 Object Pool | v3_0 | PreloadSize | Requires Preloa | adSize <= PoolSize |
| Buffer Size (KB) | 4 | | | |
| ab) Object Size (Byte) | 512 | | | |
| Pool Size | 8 | | | |
| 🖃 🔽 Preload | | | | |
| ab Preload Size | 10 | | | None All |
| 🖃 🚞 Allocation_Time | | Proposed Solutions: | | |
| Startup | | Item | Value | |
| First Access | | | 8 | |
| 🗖 Idle | | | Ū. | |
| | | | | |
| | | | | |
| | | | | |

Essentially, fixes work for both resolving errors and activating options

Fix Incompleteness

| unnamed3* - eCos Configurati | on Too | 🔒 Reso | lve conflicts | ? × |
|------------------------------------|--------------|---------|--------------------|-------------------------------|
| <u>File Edit View Build T</u> ools | <u>H</u> elp | | Increase | <u>C</u> ontinue Cancel |
| 🗅 🚅 🔚 X 🖻 📾 🐜 👗 | | / | to any value >= 10 | |
| Configuration | | Item | Prop | perty |
| 🖃 🚼 Object Pool | v3_0 | Preload | dSize Req | uires PreloadSize <= PoolSize |
| ab Buffer Size (KB) | 4 | | | |
| علم) Object Size (Byte) | 51/2 | | | |
| Pool Size | 8 | | | |
| R, 🗹 Preload | | | | |
| ab Preload Size | 10 | _ | | None |
| 🗆 📄 Allocation_Time | | Propose | d Solutions: | |
| Startup | | Item | Valu | e |
| ✓ First Access | | Prel | oadSize | |
| 🗖 Idle | | | | |
| | | | | Further decrease |
| | | | | |
| Disable | | | | |
| | | | | |

78% eCos users have ecountered situations where the proposed fix is not useful

How to complete fixes



Our Solution – Range Fixes

| 🔐 unnamed3* - eCos Configurati | on Too | I | | | | |
|---|-----------------------------|-------------------|----------------------------------|---|--|--|
| <u>F</u> ile <u>E</u> dit <u>V</u> iew <u>B</u> uild <u>T</u> ools <u>H</u> elp | | | | | | |
| 🗅 😅 🖬 👗 🛍 💼 🐜 👗 | | ? 🤋 | | | | |
| Configuration | | Item | Property | | | |
| 🖃 🚰 Object Pool | v3_0 | PreloadSize | Requires PreloadSize <= PoolSize | | | |
| Buffer Size (KB) | 4 | | | | | |
| ab) Object Size (Byte) | 512 [PreloadSize ≤ 8] | | | | | |
| ab) Pool Size | 8 | [DealSize > -10] | | | | |
| 🖃 🔽 Preload | | [POOISIZE >= 10] | | | | |
| ab Preload Size | 10 | [Preload = false] | | | | |
| 🖃 🧰 Allocation_Time | | | | | | |
| Startup | | Flavor | data | = | | |
| First Access | | Requires | PreloadSize <= PoolSize | | | |
| 🗖 Idle | | DefaultValue | 10 | Ŧ | | |

Our Contributions

- Defining the range fix generation problem
 Three desirable properties of range fixes
- Proposing a range fix generation algorithm
- Exploring the constraint interaction problem
 - Summarizing and adapting three strategies used in existing work
 - Comparing the strategies empirically



Desired Properties of Fixes

| Correctness | Minimality of variables | Maximality of ranges | | | |
|---|--|--|--|--|--|
| Any change represented by a range fix will satisfy the constraint | There is no way to change a subset of variables to satisfy the constraint | A range fix represents the maximal ranges over the variables | | | |
| A de | esirable one: [PreloadSize · | <=8] | | | |
| Undesirable ones | | | | | |
| [PreloadSize <= 9] | [PreloadSize <=8, Preload = false] | [PreloadSize <=7] | | | |

Algorithm Outline

- Step 1: find the variables to change
 - Basic idea: translating to an SMT problem
 - ① treat configurations also as soft constraints
 - 1. [soft] Preload = true
 - 2. [soft] PreloadSize = 10
 - 3. [soft] PoolSize = 8
 - 4. [hard] Preload \rightarrow PreloadSize <= PoolSize
 - ② ask an SMT solver for unsatisfiable cores

- (1, 2, 3)

- ③ pick one variable from each core
 - {Preload}, {PreloadSize}, {PoolSize}

Algorithm Outline

- Step 2: find the range of the variables
 - Basic idea: simplify the constraint
 - Example: {PreloadSize}
 - replace unchangeable variables with their current values
 - true \rightarrow PreloadSize <= 8
 - ② simplify the constraint and convert to CNF
 - [PreloadSize <= 8]</p>

Constraint Interaction

| 🖫 unnamed3* - eCos Configurati | ion Too | I | | | | |
|------------------------------------|---|--------------|---------------------------|---------|--|--|
| <u>File Edit View Build T</u> ools | <u>F</u> ile <u>E</u> dit <u>V</u> iew <u>B</u> uild <u>T</u> ools <u>H</u> elp | | | | | |
| 🗅 🖻 🖬 🐰 🖪 📾 🖌 👗 | | ∖? 🦻 | | | | |
| Configuration | | Item | Property | | | |
| 🖃 🚰 Object Pool | v3_0 | PreloadSize | Requires PreloadSize <= P | oolSize | | |
| ab Buffer Size (KB) | 4 | | | | | |
| ab Object Size (Byte) | 512 | | | _ | | |
| ab Pool Size | 8 | | dCi=0 | | | |
| 🖃 🔽 Preload | | l [Preioa | usize <= 8] | | | |
| ab) Preload Size | 10 | [PoolSi | ze >= 10] | | | |
| 🖃 🧰 Allocation_Time | | | | | | |
| Startup | | [Preloa | d = falsej | = | | |
| First Access | | Requires | | , | | |
| 🗖 Idle | | DefaultValue | 10 | - | | |

Constraint Interaction

| | | Causing another error | |
|---|--------------------------|---|----------|
| unnamed3* - eCos Configuration | n Tool | | <u> </u> |
| <u>File E</u> dit <u>V</u> iew <u>B</u> uild <u>T</u> ools <u>H</u> | <u>H</u> elp | | |
| 🗋 🗅 😅 🔚 🕹 🕷 📾 📾 🐜 🖬 🏜 | 🛍 💦 🧣 | | |
| Configuration | Item | Property | |
| 🖃 🚼 Object Pool 🛛 🗸 🗸 | ^{/3_0} PoolSize | Requires PoolSize == BufferSize * 1024 / ObjectSize | - |
| ab Buffer Size (KB) 4 | 4 | | |
| (Byte) 5 | 512 | | |
| Pool Size | 12 Property | Value | |
| 🖃 🗹 Preload | File | unnamed3_install/include/pkgconf\hal.h | |
| ab Preload Size / 1 | 10 Macro | PoolSize | |
| 🗆 🧰 Allocation_Time | Value | 12 | - |
| 🗖 Startup | Default | 0 | - |
| ✓ First Access | Flavor | data | |
| 🗖 Idle | Requires | PoolSize == BufferSize * 1024 / ObjectSize | |
| | | | Ť |
| Increase PoolSize | | Interacting constraint | |

Propagation Strategy

- Conjoin all satisfied constraints with the unsatisfied one into a single constraints
- Example
 - Original Constraints:
 - Preload → PreloadSize <= PoolSize
 - PoolSize == BufferSize * 1024 / ObjectSize
 - New Constraints:
 - Preload → PreloadSize <= PoolSize /\ PoolSize == BufferSize * 1024 / ObjectSize

Fixes from Propagation Strategy

| unnamed3* - eCos Configuration Tool | | | | | | |
|---|-------|-------------|-----------------------------|--------|--|--|
| <u>File Edit View Build T</u> ools <u>H</u> elp | | | | | | |
| 🗅 😅 🖬 🗼 🛍 🛍 👗 | | ? 🤋 | | | | |
| Configuration | | Item | Property | | | |
| 🖃 🚰 Object Pool | v3_0 | PreloadSize | Requires PreloadSize <= Poo | olSize | | |
| ab Buffer Size (KB) | 4 | | | | | |
| (Byte) Object Size | 512 | | | | | |
| ab Pool Size | 8 | Droporty | Value | | | |
| Preload [Preload | dSize | <= 8] | | | | |
| | | 10 0 Duff | Greizo - Dooleizo / 2 | 1 | | |
| □ □ Allocati [POOISIZE >= 10 & BUTTERSIZE = POOISIZE / 2] | | | | | | |
| Star [PoolSize >= 10 & ObjectSize = 4096 / PoolSize] | | | | | | |
| ✓ First [Preload = false] | | | | | | |
| 🗖 Idle | | ···] | | | | |

Experiments

- Source
 - Version histories from 5 open source projects
 - 535~933 variables, 85~330 constraints
- Steps
 - Compare each pair of consecutive versions
 - Replay the user changes in different orders
 - Generate fixes for the violations and compare with user changes

Results

- Generation Time
 - Average: 50ms
 - Maximum: 250ms
- Complexity of Fix
 - Measured by the number of variables per fix
 - Max:58
 - Median:2
 - 83% of the fix lists contain less than 10 variables

How to guide the users to identify their desirable fixes?

Our Solution

• Use the idea of priority

 The priority of a variable represents the likelihood of its current value being desirable to the user.

- Two Basic ideas:
 - Show one fix each time that only change variables with lower priorities
 - Dynamically adjust the priority of variables through implicit translation of user feedback



Provide feedback for each variable

- Accept the change (and pick a value)
- Reject the change
 - Fix duration
 - Current range is incorrect, future fixes can propose changes for this variable
 - Error duration
 - Current value is correct when fixing this error
 - Permanent duration
 - Current value is correct in the whole configuration process







The user feedbacks are stored so that later fixes will be smarter.

Evaluation Results – virtex4



(b) Number of variables

Fig. 6: Experimental results for virtex4

The number of variables is decreased by 23% in average, with a maximum reduction of 98%

Takeaway Messages

- Error Resolution is difficult in configuring large systems
- Solution space can be represented completely and concisely via range fixes
- Large fix list could be controlled by priorities
- Constraint solvers can be used in different ways to solve different problems

Thank you for your attention!

References:

- Yingfei Xiong, Hansheng Zhang, Arnaud Hubaux, Steven She, Jie Wang, Krzysztof Czarnecki. Range Fixes: Interactive Error Resolution for Software Configuration. Submitted to IEEE Transactions on Software Engineering.
- Yingfei Xiong, Arnaud Hubaux, Steven She, Krzysztof Czarnecki. Generating Range Fixes for Software Configuration. In ICSE'12: Proceedings of 34th International Conference on Software Engineering, pages 89-99, June 2012.
- Bo Wang, Leonardo Passos, Yingfei Xiong, Krzysztof Czarnecki, Haiyan Zhao, Wei Zhang. SmartFixer: Fixing Software Configurations based on Self-adaptive Priorities. In SPLC'13: Proceedings of 17th International Software Product Line Conference, August 2013.
- Arnaud Hubaux, Yingfei Xiong, Krzysztof Czarnecki. A User Survey of Configuration Challenges in Linux and eCos. VaMoS'12: Sixth International Workshop on Variability Modelling of Software-intensive Systems, January 2012.
- Leonardo Passos, Marko Novakovic, Yingfei Xiong, Thorsten Berger, Krzysztof Czarnecki, Andrzej Wasowski. A Study of Non-Boolean Constraints in Variability Models of an Embedded Operating System. FOSD'11: 3rd International Workshop on Feature-Oriented Software Development, June 2011.