



Towards Incremental Model Slicing for SPL Regression Testing

Sascha Lity, Hauke Baller, Ina Schaefer, May 14th, 2015

Technische Universität Braunschweig

Motivation

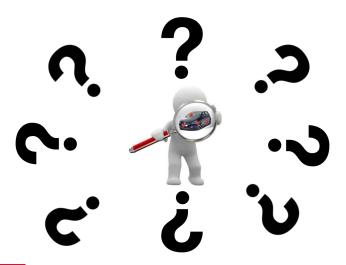




May 14th, 2015 Sascha Lity, Hauke Baller, Ina Schaefer Page 2 Towards Incremental Model Slicing for SPL Regression Testing



Motivation





May 14th, 2015 Sascha Lity, Hauke Baller, Ina Schaefer Page 2 Towards Incremental Model Slicing for SPL Regression Testing



Motivation





May 14th, 2015 Sascha Lity, Hauke Baller, Ina Schaefer Page 2 Towards Incremental Model Slicing for SPL Regression Testing



Product-by-Product Testing









May 14th, 2015 Sascha Lity, Hauke Baller, Ina Schaefer Page 3 Towards Incremental Model Slicing for SPL Regression Testing



Product-by-Product Testing

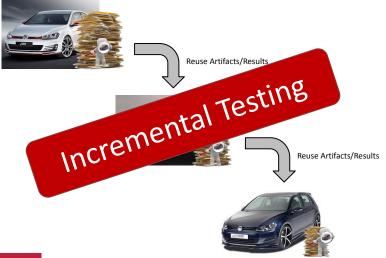




May 14th, 2015 Sascha Lity, Hauke Baller, Ina Schaefer Page 3 Towards Incremental Model Slicing for SPL Regression Testing



Product-by-Product Testing





May 14th, 2015 Sascha Lity, Hauke Baller, Ina Schaefer Page 3 Towards Incremental Model Slicing for SPL Regression Testing



Applying Regression Testing for SPLs

Regression testing allows for incremental SPL testing [TTK04, Eng10]

Regression Testing validates whether modified software behaves as intended, and modifications have not adversely impacted the software's quality. [Rot96]

- Software version testing vs. software variant testing?
- Reuse of test artifacts?
- Incremental SPL testing?
- Retest decision?





Applying Regression Testing for SPLs

Regression testing allows for incremental SPL testing [TTK04, Eng10]

Regression Testing validates whether modified software behaves as intended, and modifications have not adversely impacted the software's quality. [Rot96]

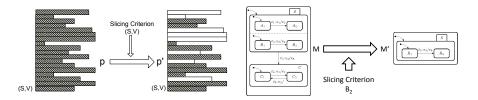
- Software version testing vs. software variant testing?
- *Reuse* of test artifacts? ⇒ Adaption of Delta Modeling [CHS15]
- Incremental SPL testing? \Rightarrow Delta-oriented [LSKL12, Loc12, LLL+14]
- Retest decision? \Rightarrow Slicing [AHKL93, BH93, GHS92]





Program/Model Slicing

- Static program projection technique by Weiser [Wei81]
- Preserves syntax and semantics w.r.t. slicing criterion
- Based on control/data dependencies
- Model slicing [JWZC02, KLB12, ACH+13] adapts concept for state-based models





May 14th, 2015 Sascha Lity, Hauke Baller, Ina Schaefer Page 5 Towards Incremental Model Slicing for SPL Regression Testing



Variability-aware Slicing [ACLF11, KLB12, KS14]

Conditioned Model Slicing of Feature-Annotated State Machines Malte Lochau

Jochen Kamischke Institut für Programmiers und Reaktive System Mühlenplondtetr. 23 Braunschweig, Germany j.kamischke@tu-bs.dt

ABSTRACT

Categories and Subject Descriptors

D.2.4 [Advance Engineering]: Subware/Program Veti Souther, D.2.13 [Suffware Engineering]: Rescalib Sub

General Terms

Keywards

1 INTRODUCTION

Model have a sich roller

presiduine and/or a low. POSDV 12: Representers 24, 29, 2013, Devules, Germany. Converseler 2012, ACM 978, 1, 4905, 1309, 451209, -812030.



Hauke Baller

models integrate any potential behaviors of all product not entrol on SPL within a visitual, so called 100% model. This

Slicing Feature Models

Mathieu Acher, Philippe Collet, Philippe Lahieu 135 - CNRS UMR 6070

Advect-Finitee models (20h) are a papelar fermidien for most be provided to miliou error-proce and todicon tacks describing the community and variability of urbrary points. anocidand with manufily analysing DM(12), Nanadity, with any regular feator manage their comparison of the structure are regular feator manage their comparison of a BN, profiles of exposure explaintations, tradingies that processing the provided structures. a merit samp ensamp tas primer a popular a a X, including constraints. The dring there SPE, postitioners taking constraintly meningful decomposition of MA and has been targetaid has the statistica language.

Aryuerde Feature Models, Suffware Product Lines, Slining. and to provide mechanisms for syntheticing large FMs from

I. INTRODUCTION

of the family. A common way is to describe variabilities of allows SPL developers to automatically obtain semantically an SPL in terms of features which are domain abstractions meaningful decompositions of PMs. relevant to makeholders [2]. A Feature Model (PM) is used U. D. Stature and Statures and Stat

>Mis are becoming accountyly complex. There are a num-ioon provided a graphical representation through Fourier Edu-ber of factors that contribute to their geording complexity. A growth. The two eventual components of an FM are hierarchy contributing factor is that FMs are being used not only to and variability. A FM Airwords (typically a mus) emerants. the software platform. Another combining factor is the use of multiple FMs to describe SPLs. It has been observed that or Ov-groups. In addition, implies or excludes constraints that maintaining a single large PM for the entire cotten may not be cut across the hierarchy can be specified to express mon feasible [8], [5], [1], [9]. Following a model-based approach, several PMs in usually designed to describe arthreast features. An PM is composed of a feature diagram (see Definition 1) various levels of abstraction and to manage variability of plus a set of constraints expressed in propositional logic (see SPLs (also called moduly populations) makes developing SPLs $\bullet G = (-, E)$ is a monod true where is a finite set

Managine the complexity of heilding DMc with a luras-

978-1-4577-1638-34114576-00-(2) 2011 HEEE ASE 2011 Lawrence KS 1158

smaller composable FMs. In mexicous work [17], we designed The goal of software product line (SPL) engineering is to aggregation) and defined semantic properties that must be

II. BACKOROUND: FRATURE MODELS.

FMs were first introduced in the FODA method 181, which In combinations [3], [4]. FMs are becoming incrusingly complex. There are a num-also provided a graphical representation through Feature Dis-G.v. EMAND: XOD, OD. Jupil, East is defined as fol-

> of features and E features, i.e., paren-child relations between them) ;

K_MAND is to a set to sugge the address with their parsets;
XOR () and OR () define former

Program Slicing in the Presence of Preprocessor Variability

United-Program clicing is a common means to support

developers in examining the source rath with respect in de-longing, programs comprehension, or expression index). While a vasi assumed of irelasingue rabit, firsy are musily indexed in single officence options. Reserve, with the increasing imperiance of variable and highly configurable systems, such as the Linear laword, the number of uniforme varianty, subject in andysis, increases downsitionally. Conceptuarily, a highest law andysis, increases downsitionally, Conceptuarily, it is highest law andysis,

shring an early variant in industry. It is interested in appry shring an early variant in industry. To even our this position,

L INTRODUCTION

While existing techniques are very matters and applicable even at large scale for single (monolithic) programs, they are

In this paper we bridge the gap between clicing and more code variability by proposing a technique that copports

only of limited are for unriable software system. Such type smally constitute a whole family of milated programs, can be deviced from a common code hour [4]. To this

Program slicing is a common and well-established method this more security up to present (1), instance, program desing above, in documpose a program with speculation compatibility appendixment. At a send, 2 is possible to specific the program in some detail. (2) is instance, to an extend the specific the program is more detail. (3) is instance, to an extend indicated indicates of a set of statements (the program short) on a particular point in the program (the decing certained). Meanwhile, maneness toolsagees have been proposed that ra-tion the original (solits) desing become of Wessen and thes, Sandro Schulz

Research problem: Obviously, statisticy analyzing bil-loss of configurations urparately is infamilie. On the sthe-hand, current closing techniques mainly work on proprocesso programs (i.e., a particular configuration) that is, the CPP developes are evaluated and reserved before analysis, and thus, variability is not supported. In other words, such kind of

Contribution: The main contribution of this name is a

While the current implementation exhibits still a stations, we argue that it provides an important foundation for langue vanishility and thus, enzythe efficient and waldely

In the followine, we entropy information about source in the interfering of prevant memory action where only variability using CPP dentities. Moreover, we key the fromthiose of our during tachnique by introducing the basis concerts of moreaux dicase.

(Line VA) transition variance in the mean hope, then in other one (Line 13-15) anight a constant value to 1. Buth forgenerate are reprinted and their in-trachologies (a controlled by configuration optimes (a.k.a. feature), by anighing (boolean)

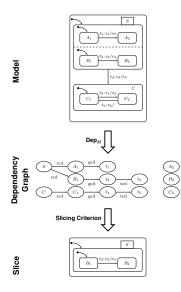
a allow

E is a set of edges that defines mandatory



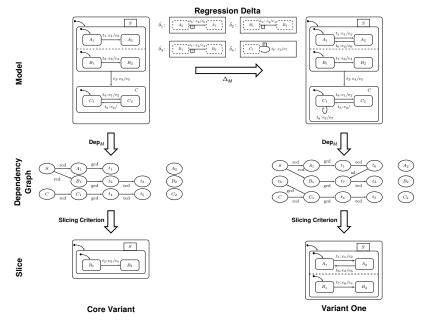
May 14th, 2015 Sascha Lity, Hauke Baller, Ina Schaefer Page 6 Towards Incremental Model Slicing for SPL Regression Testing



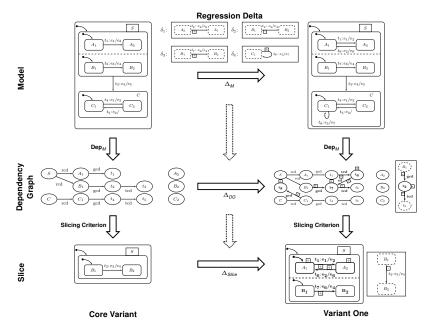


Core Variant

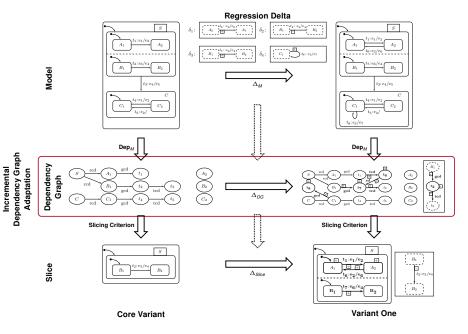
Lity, Baller, Schaefer: Towards Incremental Model Slicing for Delta-oriented SPLs, SANER'15 [LBS15]



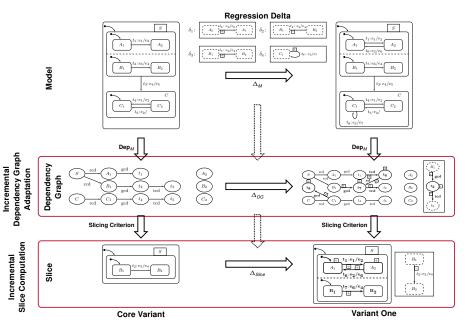
Lity, Baller, Schaefer: Towards Incremental Model Slicing for Delta-oriented SPLs, SANER'15 [LBS15]



Lity, Baller, Schaefer: Towards Incremental Model Slicing for Delta-oriented SPLs, SANER'15 [LBS15]



Lity, Baller, Schaefer: Towards Incremental Model Slicing for Delta-oriented SPLs, SANER'15 [LBS15]



Lity, Baller, Schaefer: Towards Incremental Model Slicing for Delta-oriented SPLs, SANER'15 [LBS15]

Discussion

Dependence on Differences

- Number of model deltas
- Impact/Distributability of changes

Worst Case

- Distributed changes
- No effort reduction

Average Case

- Local changes
- Exploit commonality

Benefits

- 1. Effort reduction for dependency graph generation
- 2. Direct derivation of slice changes

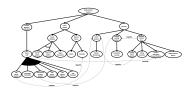


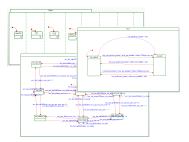


Evaluation – BCS Case Study [LLLS12]

Body Comfort System

- 27 features \Rightarrow 11616 variants
- Sampling subset: 18 Products (incl. core) [OLZG11, LSKL12]
- Sampling result \Rightarrow Order
- Model size:
 - Ø Elements: 106
 - Ø States: 40
 - Ø Transition: 66
 - Ø Hierarchy Depth: 3



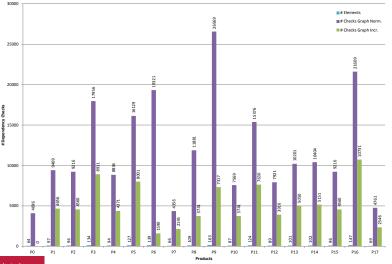




May 14th, 2015 Sascha Lity, Hauke Baller, Ina Schaefer Page 9 Towards Incremental Model Slicing for SPL Regression Testing



Evaluation – Dependency Graph (First Results)

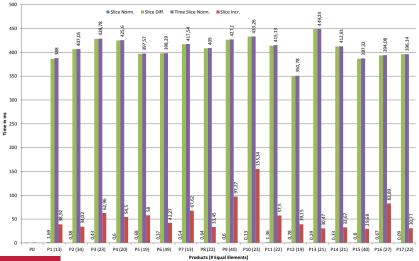




May 14th, 2015 Sascha Lity, Hauke Baller, Ina Schaefer Page 10 Towards Incremental Model Slicing for SPL Regression Testing



Evaluation – Slicing and Model Diff (First Results)





May 14th, 2015 | Sascha Lity, Hauke Baller, Ina Schaefer | Page 11 Towards Incremental Model Slicing for SPL Regression Testing



First Idea – Retest Coverage Criterion

Slice differences indicate *retest potentials* based on the impact of model changes

- What are retest requirements/goals?
 - Elements corresponding to slice differences
 - Elements connected to those elements
- Applicable for selection and prioritization
 - (Pairwise) coverage of retest requirements/goals
 - Number of occurrences of a difference usable as weighting factor
 - Potential retest test case generation
- Retest requirements/goals refer only to retestable behavior
- Model parts not influenced by changes indicate behavior not to be retested





Conclusion & Future Work

Conclusion

- Regression Testing \Rightarrow Incremental SPL testing
- Incremental model slicing \Rightarrow Change impact analysis
- Slice differences \Rightarrow Retest potentials
- Retest decision \Rightarrow Retest coverage criterion

Future Work

- Comprehensive evaluation
- Extension of control and data dependency
- Extension & optimization of implementation
- Application for (evolution-aware) change impact analysis





Thank You for Your Attention! Any Questions?





Integrated Model-based Testing of Continuously Evolving Software Product Lines

FUTURE DFG priority programme 1593 Design For Future - Managed Software Evolution

TU Braunschweig, Institute for Programming and Reactive Systems Sascha Lity *Email*: lity@ips.cs.tu-bs.de



May 14th, 2015 | Sascha Lity, Hauke Baller, Ina Schaefer | Page 14 Towards Incremental Model Slicing for SPL Regression Testing



References I

Kelly Androutsopoulos, David Clark, Mark Harman, Jens Krinke, and Laurence Tratt. State-based Model Slicing: A Survey. ACM Comput. Surv., 45(4):53:1–53:36, August 2013.
Mathieu Acher, Philippe Collet, Philippe Lahire, and Robert B. France. Slicing feature models. In <i>ASE'11</i> , pages 424–427, 2011.
Hiralal Agrawal, Joseph R. Horgan, Edward W. Krauser, and Saul London. Incremental regression testing. In <i>Proceedings of the Conference on Software Maintenance</i> , ICSM '93, pages 348–357, Washington, DC, USA, 1993. IEEE Computer Society.
Samuel Bates and Susan Horwitz. Incremental program testing using program dependence graphs. In Proceedings of the 20th ACM SIGPLAN-SIGACT Symposium on Principles of

Programming Languages, POPL '93, pages 384–396, New York, NY, USA, 1993. ACM.





References II



Dave Clarke, Michiel Helvensteijn, and Ina Schaefer. Abstract Delta Modeling.

Mathematical Structures in Computer Science, 25(3):482-527, 2015.



Emelie Engström.

Exploring Regression testing and software product line testing - research and state of practice.

Lic dissertation, Lund University, May 2010.



R. Gupta, M.J. Harrold, and M.L. Soffa. An approach to regression testing using slicing. In *Software Maintenance, 1992. Proceerdings., Conference on*, pages 299–308, Nov 1992.



Wang Ji, Dong Wei, and Qi Zhi-Chang. Slicing Hierarchical Automata for Model Checking UML Statecharts. In *Formal Methods and Software Engineering*, pages 435–446. Springer, 2002.



Jochen Kamischke, Malte Lochau, and Hauke Baller. Conditioned Model Slicing of Feature-Annotated State Machines. In *FOSD'12*, pages 9–16, 2012.





References III



Frederik Kanning and Sandro Schulze. Program Slicing in the Presence of Variability. In *ICSME'14 - ERA Track*, 2014.



Sascha Lity, Hauke Baller, and Ina Schaefer. Towards incremental model slicing for delta-oriented software product lines. In *Software Analysis, Evolution and Reengineering (SANER), 2015 IEEE 22nd International Conference on*, pages 530–534, March 2015.



Malte Lochau, Sascha Lity, Remo Lachmann, Ina Schaefer, and Ursula Goltz. Delta-oriented Model-based Integration Testing of Large-scale Systems. *Journal of Systems and Software*, 91:63–84, 2014.



Sascha Lity, Remo Lachmann, Malte Lochau, and Ina Schaefer. Delta-oriented Software Product Line Test Models - The Body Comfort System Case Study.

Technical Report 2012-07, Technische Universität Braunschweig, 2012.





References IV



Malte Lochau.

Model-Based Conformance Testing of Software Product Lines. PhD thesis, Technische Universität Braunschweig, 2012.



Malte Lochau, Ina Schaefer, Jochen Kamischke, and Sascha Lity. Incremental Model-Based Testing of Delta-Oriented Software Product Lines. In *TAP'12*, pages 67–82. Springer, 2012.



S. Oster, M. Lochau, M. Zink, and M. Grechanik. Pairwise Feature-Interaction Testing for SPLs: Potentials and Limitations. In *FOSD'11*, 2011.



Alessandro Orso, Saurabh Sinha, and Mary Jean Harrold. Incremental Slicing Based on Data-Dependences Types. In *ICSM'01*, page 158, 2001.



Gregg Rothermel.

Efficient, Effective Regression Testing Using Safe Test Selection Techniques. PhD thesis, Clemson University, May 1996.





References V

Antti Tevanlinna, Juha Taina, and Raine Kauppinen. Product Family Testing: A Survey. SIGSOFT Softw. Eng. Notes, 29(2):12–12, March 2004.



Heike Wehrheim.

Incremental Slicing.

In Formal Methods and Software Engineering, ICFEM'06, pages 514-528, 2006.



Mark Weiser.

Program slicing.

In ICSE'81, pages 439-449, 1981.



