West Virginia University

An empirical study of pre-release software faults in an industrial product line

Tom Devine, Katerina Goseva-Popstajanova
Lane Department of Computer Science and Electrical Engineering, West Virginia University
Sandeep Krishnan, Robyn R. Lutz
Department of Computer Science, Iowa State University
J. Jenny Li
Avaya Labs, US

West Virginia University

Acknowledgements

This material is based upon work supported in part by the National Science Foundation under grants number 0916275 and 0916284, with funds from the American Recovery and Reinvestment Act of 2009.

West Virginia University

Software Product Lines

- A Software Product Line (SPL) is a family of products designed to take advantage of their common features and predicted variabilities.
  - E.g., cruise control software designed to have a common core and several variations to accommodate different vehicle models.
- Traditionally, SPL components are classified as:
  - Commonalities: components that are shared, or reused, among all the products in a product line.
  - Variabilities: components that are not present in all members of the SPL.

Motivation

- Does systematic reuse in Software Product Lines (SPLs) provide measurable benefits?
  - Studies exist on the benefits of reuse in software development.
  - Software Product Lines (SPLs) rigorously employ systematic reuse.
  - However, few empirical studies conducted on SPLs exist in literature.
- We perform an evidence-based, empirical case study of an industrial SPL.

West Virginia University

Research Approach

We explore research questions divided into the following categories:

1. Component Level Analysis
   - Measures correlations of different metrics collected at the class level, then aggregated into packages.
   - Replicates prior work in an SPL environment.
2. Degree of Reuse Analysis
   - Examines the fault-proneness and change-proneness of components with different degrees of reuse.
   - Provides novel results unique to SPLs.
3. Longitudinal Analysis
   - Examines the impact of reuse in an SPL on future products.
   - Evaluates predictive models for new family members.
   - Provides novel results unique to SPLs.

CASE STUDY

West Virginia University

PolyFlow

- A product line of software testing tools developed in Java by Avaya Corporation.
- Formerly known as “eXVantage”
- Allows developers to:
  – generate and execute test cases
  – calculate associated coverage measures
  – other developmental testing tasks
- Utilizes a modular architecture
  – related classes are grouped into packages that serve as components

West Virginia University

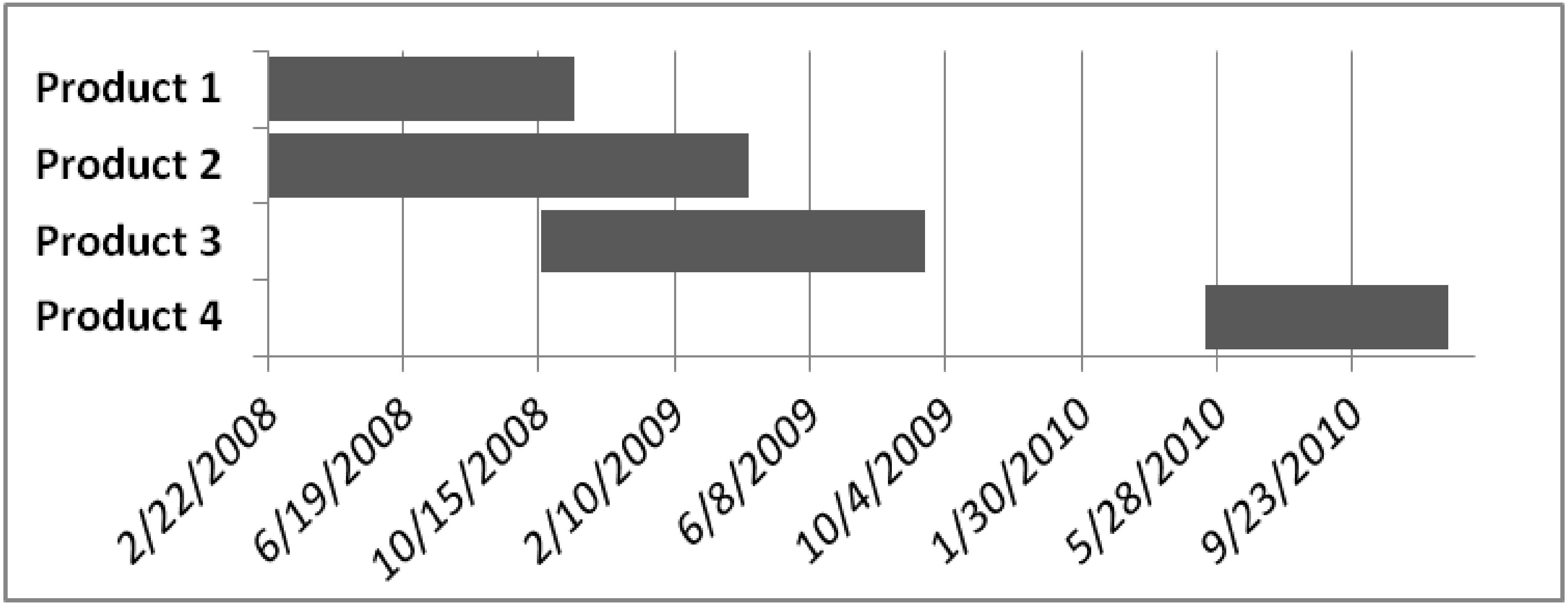
PolyFlow

- Variabilities include support for various
  - operating systems
  - target programming languages
  - user interfaces
- We consider four members of the SPL.

Product	Components	LoC
P_1	23	47,138
P_2	29	35,238
P_3	37	49,676
P_4	22	36,852

West Virginia University

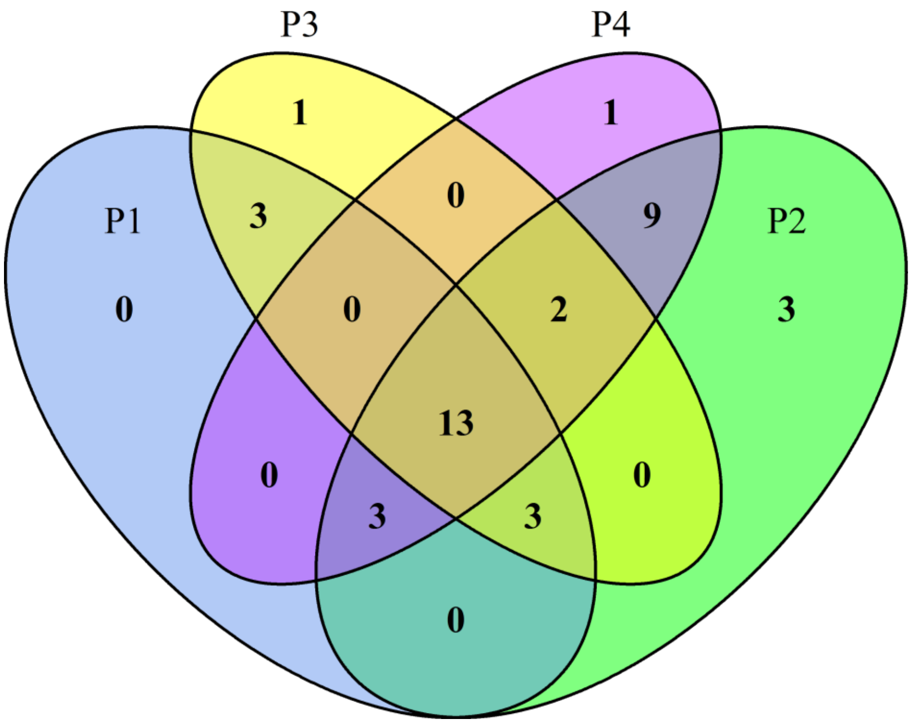
PolyFlow



Pre-release timeline of development and testing efforts for PolyFlow products.

PolyFlow

- In practice, we found that not all components easily fit into the traditional paradigm.
- We observed much more diversity in degrees of reuse:
  - Common Components are used in all products.
  - High Reuse Variations are used in 3 products.
  - Low Reuse Variations are used in 2 products.
  - Single-Use Variations are currently used in only one product.



DATA EXTRACTION AND METRIC DESCRIPTIONS

Data Sources

- CVS Subversion (SVN) Repository
  - Repository for all components' code
  - Logs automatically maintained for all changes made
- Modification Request (MR) Database
  - Record of requests for code changes
  - Created during software development and testing
- The data were acquired through a painstaking, iterative process of review.
  - Each step was verified and validated by a domain expert for the product line.

West Virginia University

Type of Metrics

- Source Code Metrics
  - Represent information about the source code of the components, i.e. LoC, number of files
  - Gathered from static analysis of the Java code
- Change Metrics
  - Represent the modifications made to a particular component, i.e. total LoC added or number of files added
  - Gathered from MR Database and analysis of the SVN log files

West Virginia University

Metrics

Metric	Description
Lines of Code (LoC)	total number of non-comment lines of Java source code in a component.
Number of Files (NoF)	number of files comprising the component.
Maximum Complexity	maximum complexity of any method in a component.
Average Complexity	average complexity of a component's methods.
CodeChurn	total LoC added and deleted from a component.
Average CodeChurn	CodeChurn divided by total LoC.
FileChurn	NoF added to or deleted from the repository.
Average FileChurn	FileChurn divided number of files.
Improvements	number of MRs for improvements to the code.
New Features	number of MRs for new code to implement new features.
Number of Faults	pre-release faults detected during testing

COMPONENT LEVEL ANALYSIS

West Virginia University

Component Level: Research Questions

1. Is Number of Faults correlated with any of the gathered metrics?
2. Are any of the gathered metrics correlated to each other?

West Virginia University

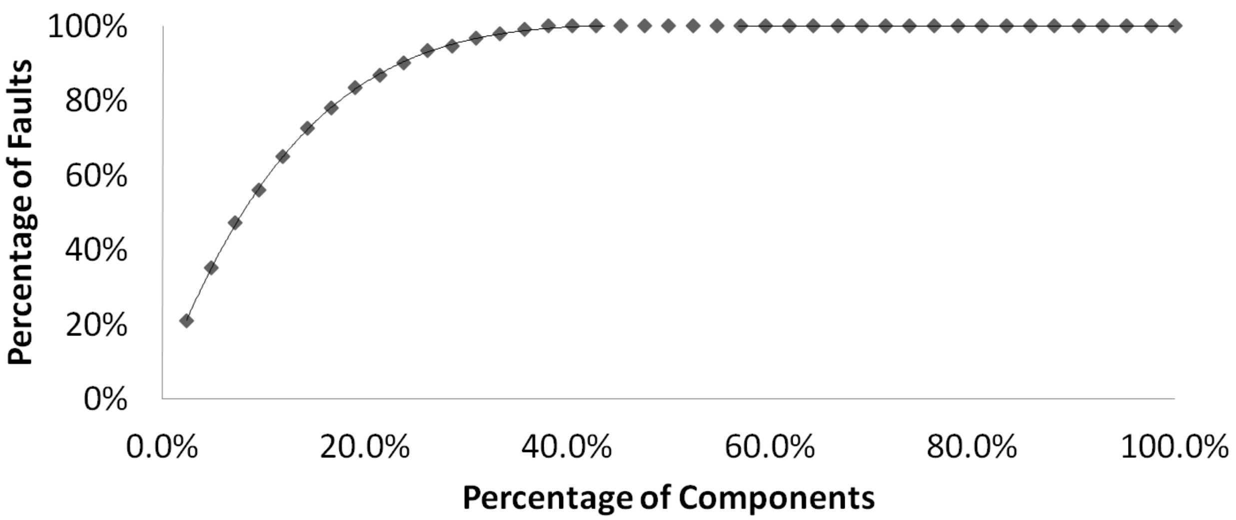
Component Level: Correlations

	Faults	Improvements	NewFeatures	CodeChurn	AvgCodeChurn	FileChurn	AvgFileChurn	LoC	NumFiles	MaxComplexity	AvgComplexity
Faults	–	0.597 (0.0001)	0.760 (0.0000)	0.702 (0.0000)	0.612 (0.0000)	0.435 (0.0056)		0.490 (0.0015)	0.469 (0.0026)	0.321 (0.0461)	
Improvements		–	0.676 (0.0000)	0.586 (0.0001)	0.597 (0.0001)		-0.388 (0.0146)	0.418 (0.0082)	0.359 (0.0247)	0.299 (0.0645)	0.352 (0.0281)
NewFeatures			–	0.734 (0.0000)	0.674 (0.0000)	0.359 (0.0247)				0.398 (0.0122)	
CodeChurn				–				0.548 (0.0003)		0.497 (0.0013)	
AvgCodeChurn					–	0.398 (0.0121)				0.417 (0.0083)	
FileChurn						–				0.344 (0.0320)	
AvgFileChurn							–	-0.343 (0.0327)			-0.344 (0.0319)
LoC								–		0.599 (0.0001)	
NumFiles									–	0.374 (0.0192)	
Max Complexity										–	
Avg. Complexity											–

Spearman correlation ρ values for non-trivial, statistically significant associations, accompanied by the p-value in parentheses.

Component Level Research Question

3. Does a small set of components contain the majority of faults?
   - 80% of all pre-release faults occur in 20% of components.
   - Confirms related findings in an SPL environment.



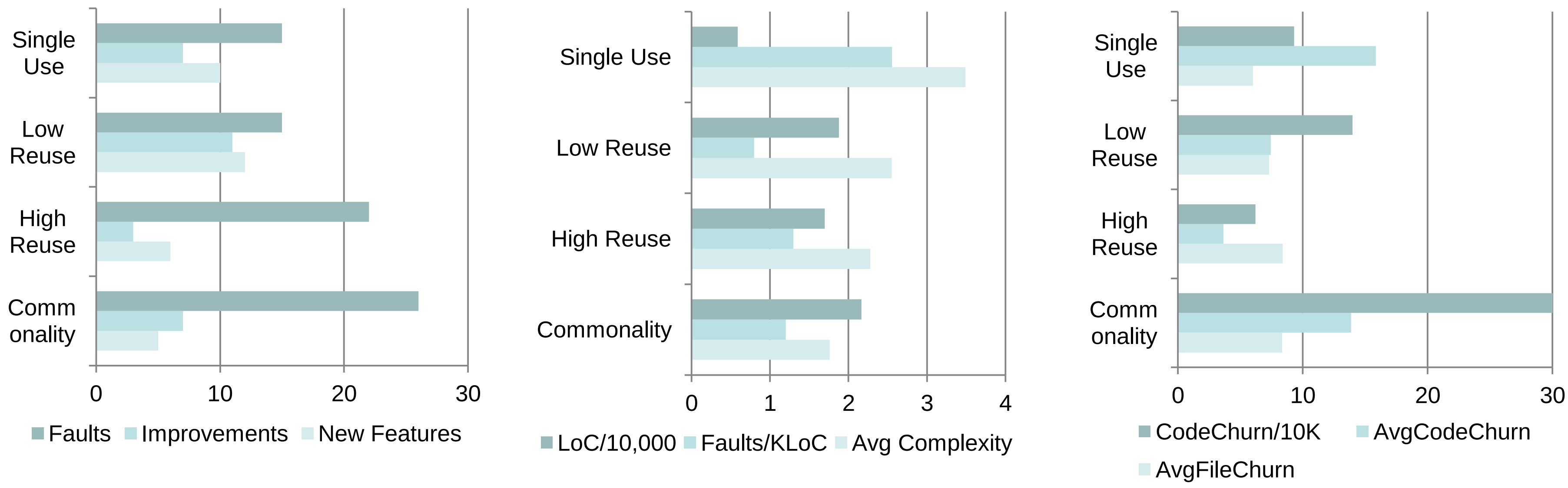
DEGREE OF REUSE ANALYSIS

West Virginia University

Degree of Reuse: Research Questions

4. Do the Number of Faults and/or Fault Density vary in components by degree of reuse?
5. Do the number of New Features and Improvements vary in components by degree of reuse?
6. Does the change-proneness of the code vary by degree of reuse?

Degree of Reuse: Observations



– Single reuse components are the most likely to change and have the highest fault densities.
– Common components exhibit high Average Code Churn.
– Results are consistent with our previous work on an open-source SPL [Krishnan et al., MSR 2011].

LONGITUDINAL ANALYSIS

West Virginia
University

Longitudinal Analysis: Research Questions

7. Do products developed later benefit from the reuse inherent in the product line?
8. Can the Number of Faults in a new product be predicted from previously existing products' data?

West Virginia University

RQ 7: Benefits of SPL Reuse on New Products

P_3 had 37 total faults:

- 8 found and fixed before P_3 was created
- 11 in low reuse variation components
- 18 in high-reuse variation components

P_4 had 69 total faults:

- 67 found and fixed before P_4 was created
- 1 in high-reuse variation components
- 1 in single use variation component

- This suggests that later products do benefit from the faults fixed in the components they share with other concurrently or previously developed products.

West Virginia University

RQ 8: Predictive Modeling for New Products

We used data from P_1 and P_2 to create a linear model via stepwise regression, then used the model to predict the number of faults in P_3 and P_4.

- At least in this SPL, the data from more mature products can be used to predict the number of faults in subsequently developed products.
- Such predictions could be useful for allocating testing effort to the most fault prone components.

Product	Actual Faults	Predicted Faults	Absolute Error
P_3	0	0.18	0.18
P_4	1	1.08	0.08

West Virginia University

Current Work

- Explore the external validity by performing an empirical study on another SPL.
- PolyFlow
  - Medium sized, i.e. 4 products of 35k-50k LoC
  - 2 new components
  - Industrial product line
- New Empirical Study
  - Larger scale, i.e. products with 1+ million LoC
  - Multiple new components
  - Open source product line

West Virginia University

Lessons Learned

- Change metrics are more highly correlated to number of faults than are static code metrics.
  - Rigorous change control is central to the quality of the members of an SPL.

West Virginia University

Lessons Learned

- There is a spectrum of component reuse with significant, measurable differences among their fault profiles.
  - Planned reuse in SPLs enhances the quality of products.
  - Despite systematic reuse, common components still require changes as the SPL evolves.
  - The sustainability of a product line over time seems to depend on consistent, ongoing reuse with a few, cohesive variations.

Lessons Learned

- The systematic reuse in Software Product Lines (SPLs) provides measurable benefits
  - In quality - New products benefitted from faults fixed in reused components.
  - In fault proneness prediction - Linear models based on prior products' data can accurately predict the Number of Faults in future products.
    - More empirical studies are needed to generalize to other SPLs.

THANK YOU!!!