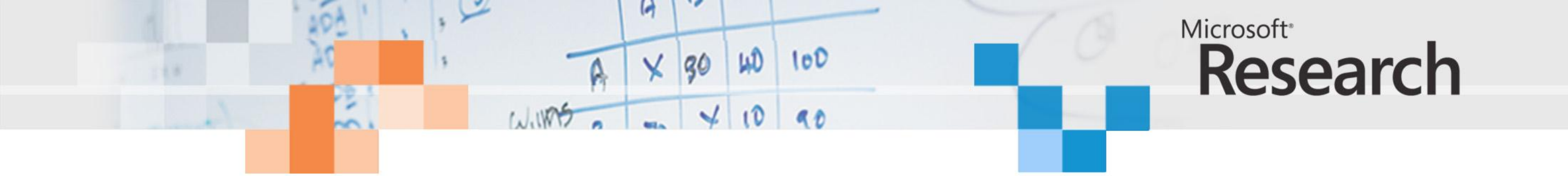


Cross-project defect prediction

Thomas Zimmermann
Microsoft Research

Empirical Software Engineering and Measurement Group (ESM)

The Empirical Software Engineering and Measurement research area activities at Microsoft Research focus on understanding various software development issues from an empirical perspective. We are involved in doing practical studies on large software systems. All our work is done in conjunction with Microsoft product teams such as Windows and Visual Studio.

Our current interests are in the areas of:

- Software Reliability: Predicting Failures/Failure-proneness, Test Prioritization, Failure Analysis.
- Software Process: Organizational impact on quality, Agile software development, Global software development, Effort estimation
- Empirical Studies: Unit Testing, Inspections, Assertions, Test Driven Development

Publications

Silvia Breu, Rahul Premraj, Jonathan Sillito, and Thomas Zimmermann, Information Needs in Bug Reports: Improving Cooperation Between Developers and Users, in *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, Association for Computing Machinery, Inc., February 2010

Christian Bird, Nachiappan Nagappan, Harald Gall, Premkumar Devanbu, and Brendan Murphy, Using Socio-Technical Networks to Predict Failures, in *Proceedings of the IEEE International Symposium on Software Reliability Engineering (ISSRE)*, IEEE, November 2009

Laurie Williams, Gunnar Kudrjavets, and Nachiappan Nagappan, On the Effectiveness of Unit Test Automation at Microsoft, in *Proceedings of the IEEE International Symposium on Software Reliability Engineering (ISSRE)*, IEEE

People

Brendan Murphy (MSR Cambridge)
Nachi Nagappan (MSR Redmond)
Thomas Zimmermann (MSR Redmond)

Upcoming events

ICSE 2010 NIER Track – New and Emerging Results, Cape Town, South Africa. *Submit by 7 January, 2010*

MSR 2010 - Mining Software Repositories, Cape Town, South Africa. *Submit by 11/14 January, 2010 (abstracts/papers)*

ESEM 2010 – Empirical Software Engineering and Measurement, Bolzano, Italy.

Visitors

Professors

Harald Gall (2008, 2009)
Victor R. Basili (2007)
Neeraj Suri (2007)

Christian Bird, Nachiappan Nagappan, Harald Gall, Premkumar Devanbu, and Brendan Murphy, Using Socio-Technical Networks to Predict Failures, in *Proceedings of the IEEE International Symposium on Software Reliability Engineering (ISSRE)*, IEEE, November 2009

Laurie Williams, Gunnar Kudrjavets, and Nachiappan Nagappan, On the Effectiveness of Unit Test Automation at Microsoft, in *Proceedings of the IEEE International Symposium on Software Reliability Engineering (ISSRE)*, IEEE, November 2009

Md. Mainur Rahman, Guenther Ruhe, and Thomas Zimmermann, Optimized Assignment of Developers for Fixing Bugs – An Initial Evaluation for Eclipse Projects (Short Paper), in *Proceedings of the Third International Symposium on Empirical Software Engineering and Measurement (ESEM)*, IEEE Computer Society, October 2009

Audris Mockus, Nachiappan Nagappan, and Trung T. Dinh-Trong, Test Coverage and Post-Verification Defects: A Multiple Case Study, in *Proceedings of the ACM-IEEE Empirical Software Engineering and Measurement Conference (ESEM)*, IEEE Computer Society, October 2009

Thomas Zimmermann and Nachiappan Nagappan, Predicting Defects with Program Dependencies (Short Paper), in *Proceedings of the Third International Symposium on Empirical Software Engineering and Measurement (ESEM)*, IEEE Computer Society, October 2009

David Ma, David Schuler, Thomas Zimmermann, and Jonathan Sillito, Expert Recommendation with Usage Expertise (Short Paper), in *Proceedings of the 25th IEEE International Conference on Software Maintenance (ICSM)*, IEEE Computer Society, September 2009

Gaeul Jeong, Sunghun Kim, and Thomas Zimmermann, Improving Bug Triage with Bug Tossing Graphs, in *Proceedings of the 7th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/ FSE)*, Association for Computing Machinery, Inc., August 2009

Thomas Zimmermann, Nachiappan Nagappan, Harald Gall, Emanuel Giger, and Brendan Murphy, Cross-project Defect Prediction, in *Proceedings of the 7th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/ FSE)*, Association for Computing Machinery, Inc., August 2009

Visitors

Professors

Harald Gall (2008, 2009)
Victor R. Basili (2007)
Neeraj Suri (2007)
Laurie Williams (2009)
Andreas Zeller (2005, 2009)

Post-docs

Martin Pinzger (2007)

Interns

Christian Bird (2008, 2009)
Philip Guo (2009)
Ayse Tosun (2009)
Andreas Johansson (2007)
Lucas Layman (2007)
Thomas Zimmermann (2006)

Recruiting

We are always looking for exceptional PhD candidates to join us as interns, any time of the year, though summer is the typical time interns visit. We have the possibility of students spending an internship either in Cambridge, UK or Redmond, USA. For more information about becoming an intern, please visit our internship website. After applying via the internship site please do send us a note to let us know to watch out for your application.

Related Groups

Human Interaction in Programming (HIP)
Software Reliability Research (SRR)
Rigorous Software Engineering (RSE)
Foundations in Software

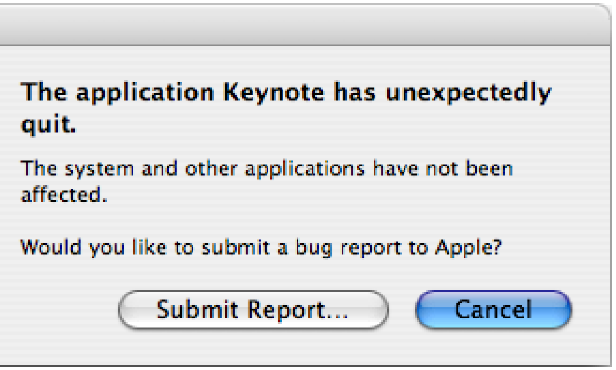
VOODOO ON A MOUNTAIN OF DATA

Upcoming Events

- ICSE 2010: http://www.sbs.co.za/icse2010/
  - New Ideas and Emerging Results
  - ACM Student Research Competition (SRC) sponsored by Microsoft Research
- MSR 2010: Mining Software Repositories http://msrconf.org/
  - Mining Challenge: will be announced next week!
- ESEM 2010: Empirical Software Engineering http://esem2010.case.unibz.it/

DEFECT PREDICTION

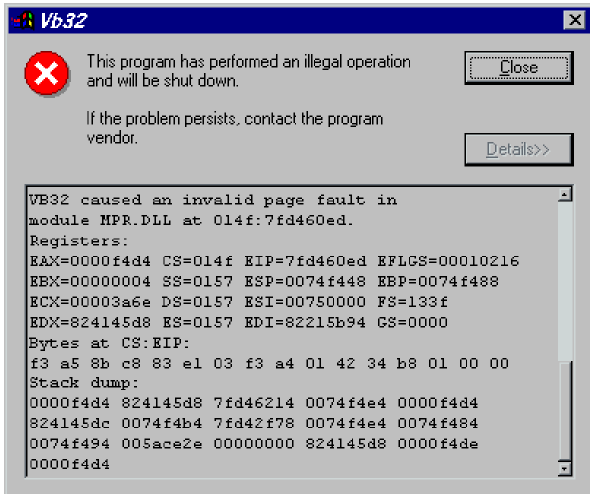
Bugs are everywhere



The application Keynote has unexpectedly quit.

The system and other applications have not been affected.

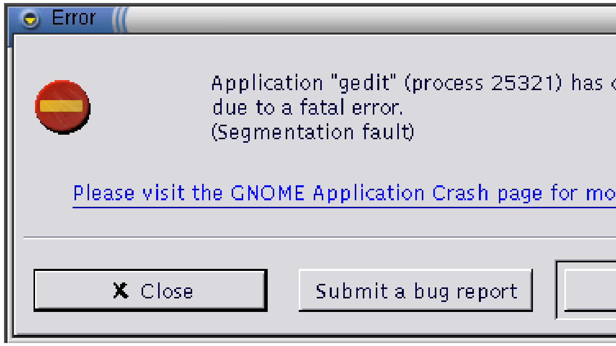
Would you like to submit a bug report to Apple?



This program has performed an illegal operation and will be shut down.

If the problem persists, contact the program vendor.

```
VB32 caused an invalid page fault in
module MPR.DLL at 014f:7fd460ed.
Registers:
EAX=0000f4d4 CS=014f EIP=7fd460ed EFLGS=00010216
EBX=00000004 SS=0157 ESP=0074f448 EBP=0074f488
ECX=00003a6e DS=0157 ESI=00750000 FS=133f
EDX=824145d8 ES=0157 EDI=82215b94 GS=0000
Bytes at CS:EIP:
f3 a5 8b c8 83 e1 03 f3 a4 01 42 34 b8 01 00 00
Stack dump:
0000f4d4 824145d8 7fd46214 0074f4e4 0000f4d4
824145dc 0074f4b4 7fd42f78 0074f4e4 0074f484
0074f494 005ace2e 00000000 824145d8 0000f4de
0000f4d4
```



Application "gedit" (process 25321) has
due to a fatal error.
(Segmentation fault)

Please visit the GNOME Application Crash page for mo

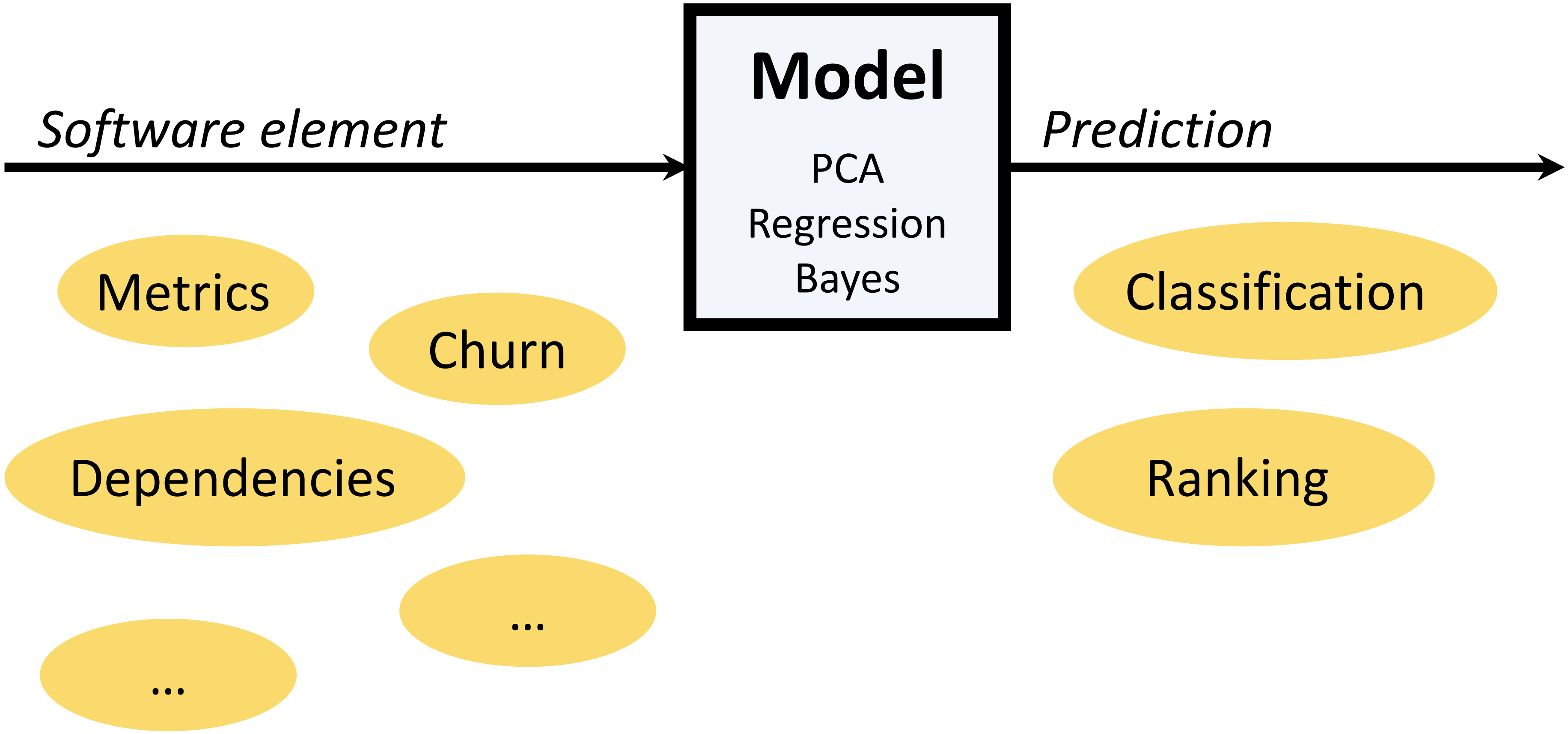
Quality assurance is limited...

...by time...

...and by money.

Spent QA resources on the components/files that need it most, i.e., are most likely to fail.

Defect prediction



Software element

Model

PCA

Regression

Bayes

Prediction

Metrics

Churn

Dependencies

…

…

Classification

Ranking

Classification

Has a binary a defect or not?

Ranking

Which binaries have the most defects?

Defect prediction

- Learn a prediction model from historic data
- Predict defects for the same project
- Hundreds of prediction models exist
- Models work fairly well with precision and recall of up to 80%.

Predictor	Precision	Recall
Pre-Release Bugs	73.80%	62.90%
Test Coverage	83.80%	54.40%
Dependencies	74.40%	69.90%
Code Complexity	79.30%	66.00%
Code Churn	78.60%	79.90%
Org. Structure	86.20%	84.00%

From: N. Nagappan, B. Murphy, and V. Basili. The influence of organizational structure on software quality. ICSE 2008.

Nachiappan Nagappan, Brendan Murphy, Victor R. Basili

[ICSE 2008]

INFLUENCE OF ORGANIZATIONAL STRUCTURE ON SOFTWARE QUALITY

Motivation

- **Conway's Law:** "Organizations that design systems are constrained to produce systems which are copies of the communication structures of these organizations."
- **Brooks** argues in the Mythical Man Month that the product quality is strongly affected by that structure.
- Little empirical evidence for relationship between organizational structure and direct measures of software quality like failures

Organization metrics

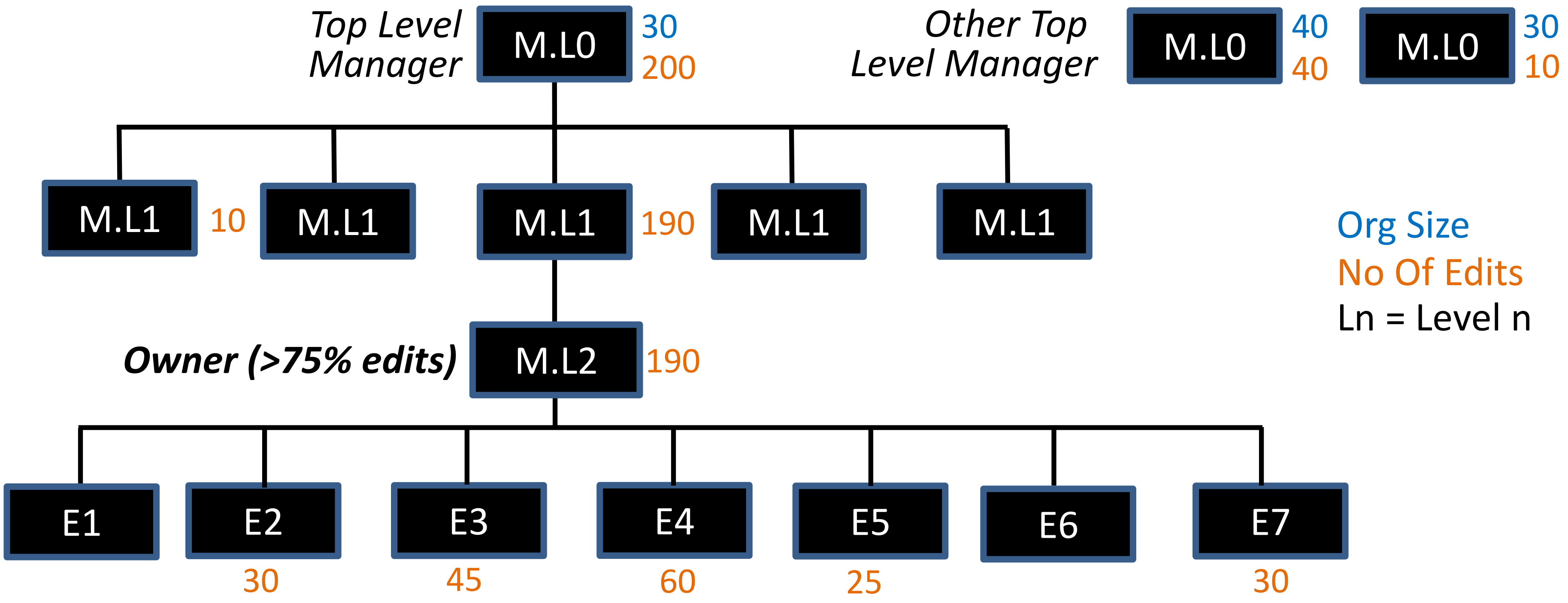
- The more people who touch the code, the lower the quality (NOE)
- A large loss of team members affects the knowledge retention and lowers the quality (NOEE)
- The more edits to components, the higher the instability and the lower the quality (EF)
- The lower the level the ownership, the better the quality (DMO)

Organization metrics

- The more cohesive the contributors (organizationally), the higher the quality (PO)
- The more cohesive the contributions (edits), the higher the quality (OCO)
- The more the diffuse the contribution to a binary, the lower the quality (OOW)
- The more diffuse the different organizations contributing code, the lower the quality (OIF)

Examples of organizational metrics

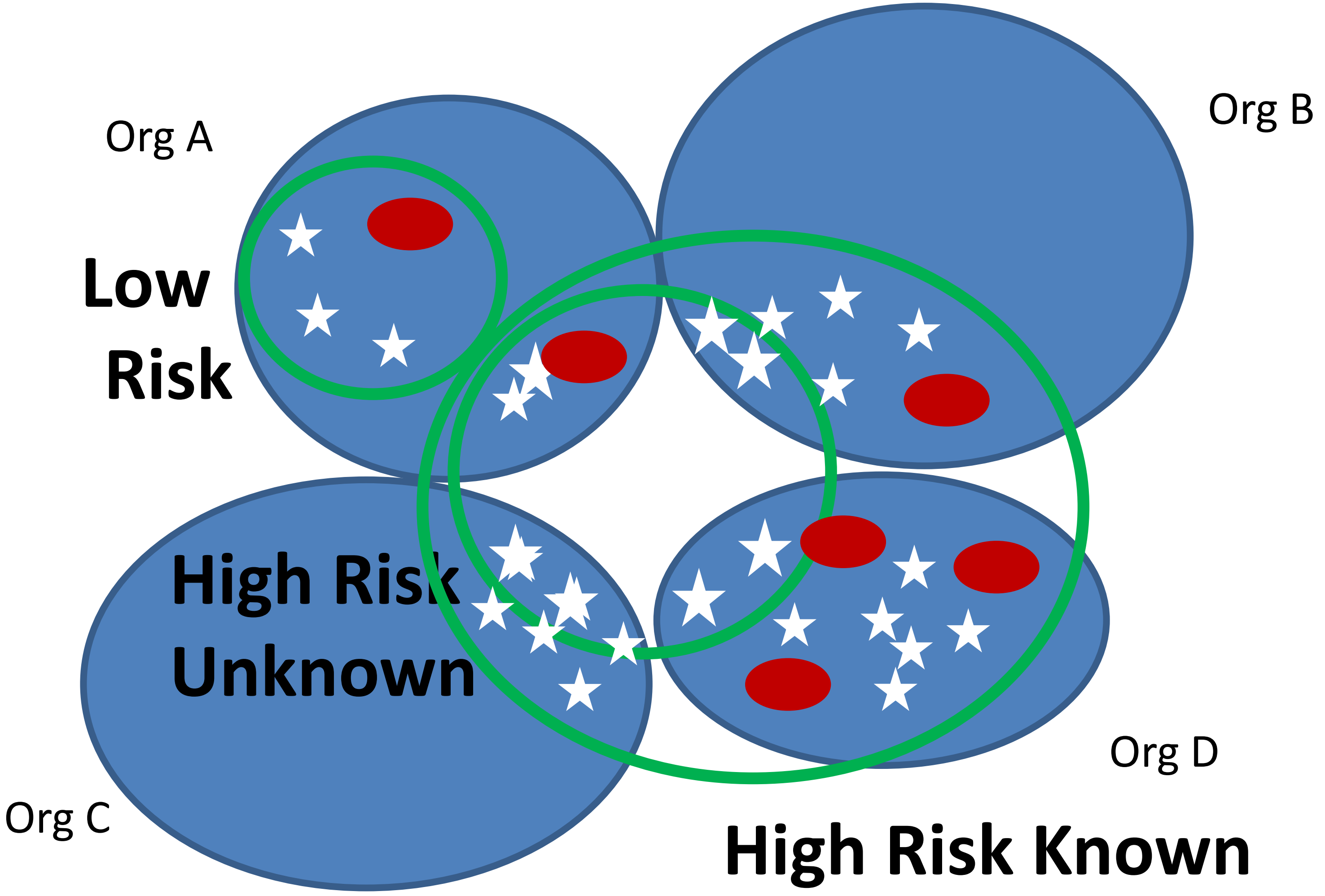
Total Edits = 250 Total Engineers Editing = 32 Total Ex-Engineers = 0



NOE = 32 NOEE = 0 EF = 250 DMO = 2 PO = 7/30

OCO = 190/250 OOW = 5/32 OIF (>10% edits) = 2

Interpretation



Org A

Org B

Low Risk

High Risk Unknown

Org C

Org D

High Risk Known

Binary

New Feature

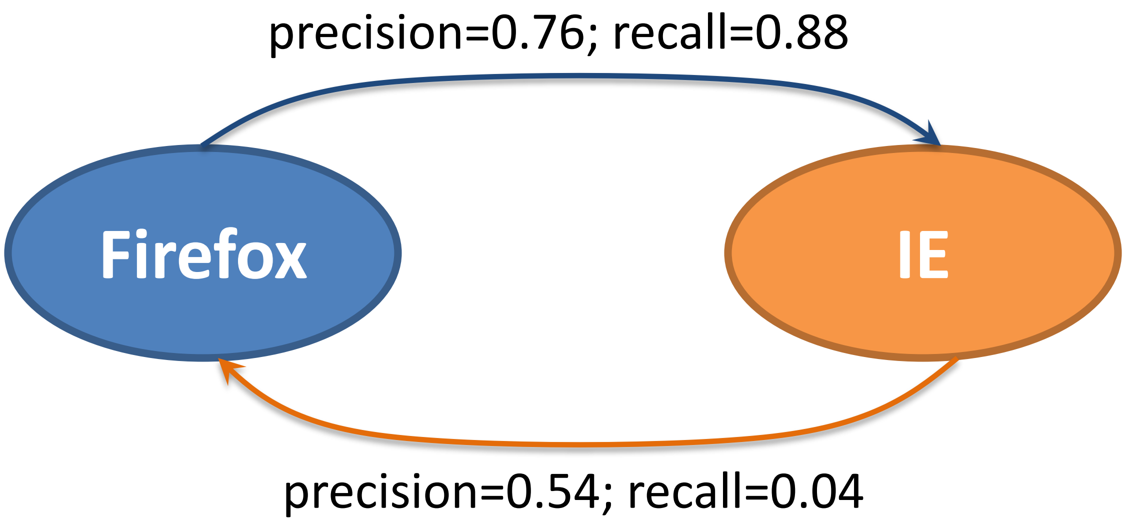
Thomas Zimmermann, Nachiappan Nagappan, Harald Gall,
Emanuel Giger, Brendan Murphy [ESEC/FSE 2009]

CROSS-PROJECT DEFECT PREDICTION

Why cross-project prediction?

- Some projects do have not enough data to train prediction models or the data is of poor quality
- New projects do have no data yet
- Can such projects use models from other projects? (=cross-project prediction)

A first experiment: Firefox and IE



precision=0.76; recall=0.88

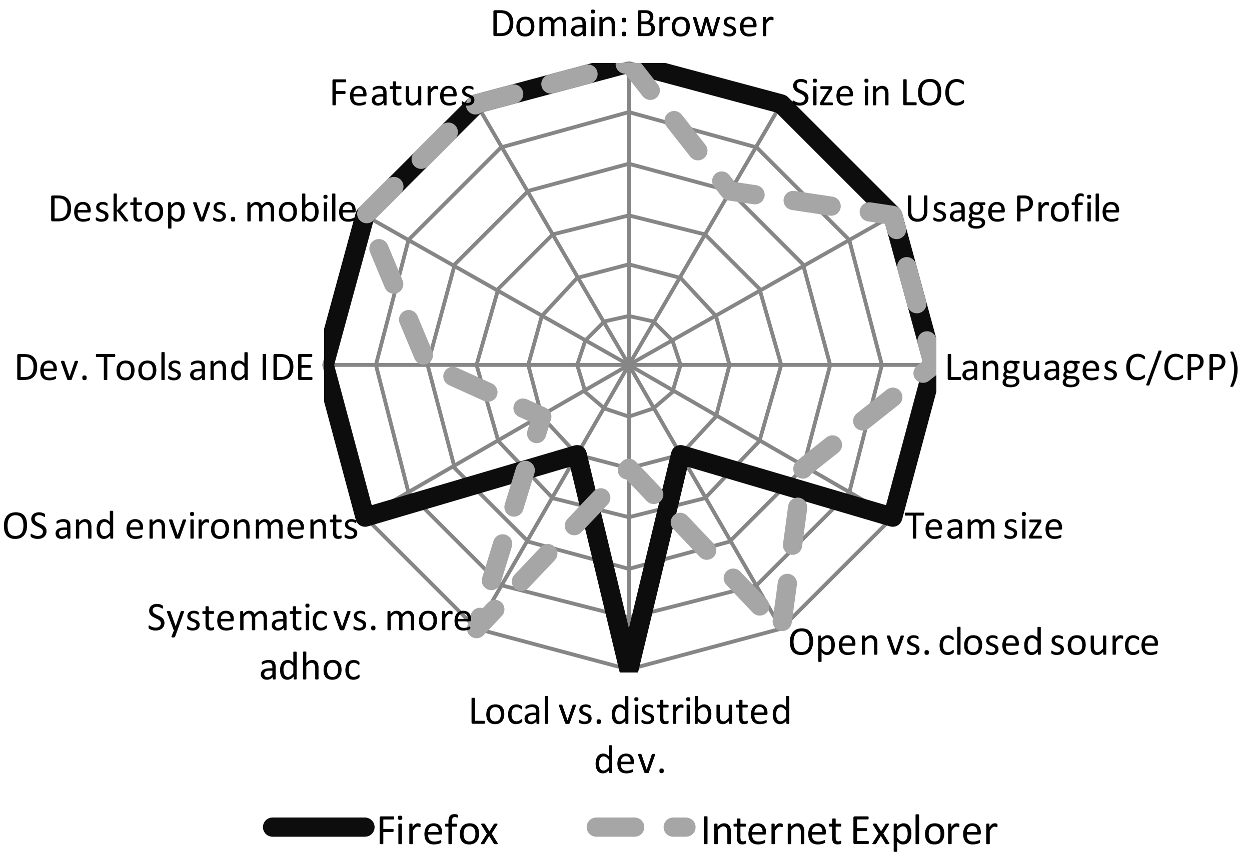
Firefox

IE

precision=0.54; recall=0.04

Firefox can predict defects in IE.
But IE cannot predict Firefox. WHY?

Comparing Firefox and IE



Research questions

- RQ 1. To which extent is it possible to use cross-project data for defect prediction?
- RQ 2. Which kinds of systems are good predictors? What is the influence of data, domain, company, and process?

THE EXPERIMENT

Experiment outline

- 12 systems with 28 datasets
  - different versions
  - different levels of analysis (components, files)
- Run all 622 cross-project combinations
  - for example Firefox and IE is one combination
  - then *train* model from Firefox data, *test* on IE
  - ignore invalid combinations, e.g., do not train from Eclipse 3.0 and test on 2.1

Experiment outline

- For each combination, we record
  - – similarities between projects
  - – precision, recall, and accuracy values
  - – success, i.e., are all of precision, recall, and accuracy > 0.75

Systems studied

Project	No. of versions	Total LOC	Total Churn
Firefox	2	3.2 – 3.3 MLOC	0.64 – 0.95 MLOC
Internet Explorer	1	2.30 MLOC	2.20 MLOC
Direct-X	1	1.50 MLOC	1.00 MLOC
Internet Information Services (IIS)	1	2.00 MLOC	1.20 MLOC
Clustering	1	0.65 MLOC	0.84 MLOC
Printing	1	2.40 MLOC	2.20 MLOC
File system	1	2.00 MLOC	2.20 MLOC
Kernel	1	1.90 MLOC	3.20 MLOC
SQL Server 2005	1	4.6 MLOC	7.2 MLOC
Eclipse	3	0.79 – 1.3 MLOC	1.0 - 2.1 MLOC
Apache Derby	4	0.49 – 0.53 MLOC	4 – 23 KLOC
Apache Tomcat	6	0.25 – 0.26 MLOC	8 – 98 KLOC

Data used in prediction models

Relative code measures on churn, complexity and pre-release bugs

- Added LOC / Total LOC
- Deleted LOC / Total LOC
- Modified LOC / Total LOC
- (Added + deleted + modified LOC) / (Commits + 1)
- Cyclomatic Complexity / Total LOC
- Pre-release bugs / Total LOC

RESULTS

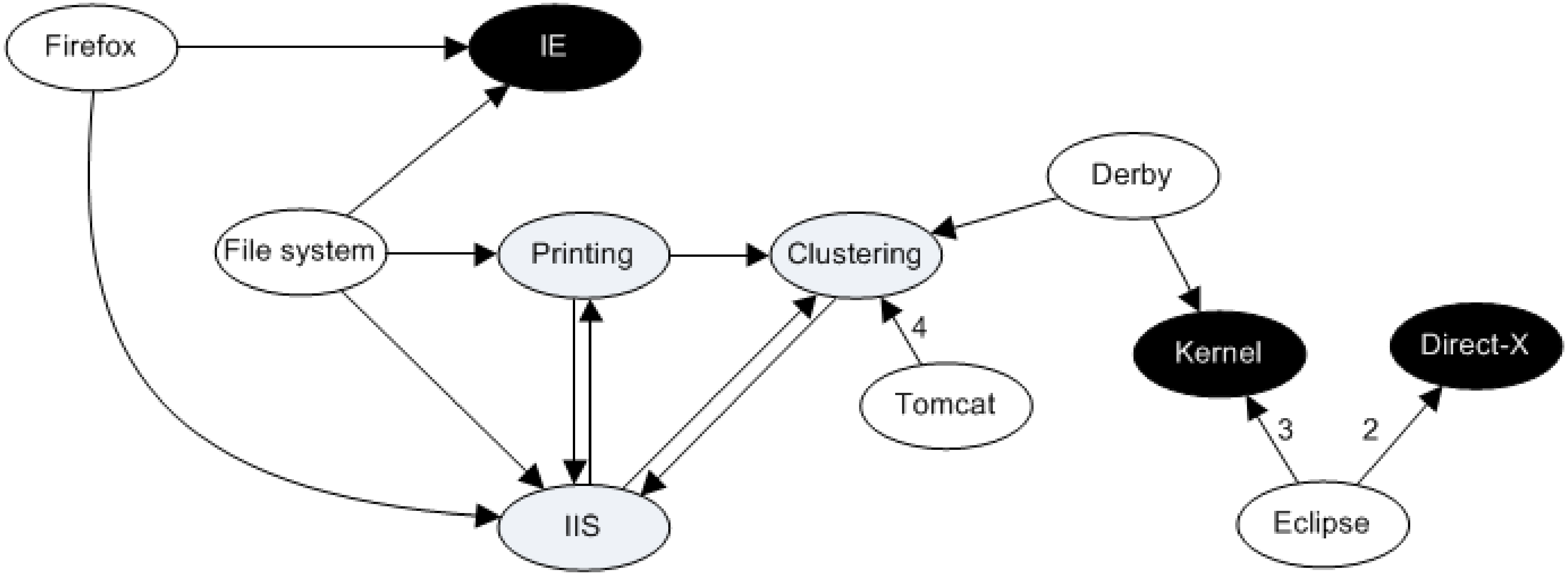
Success rate

Any guesses?

3.4%

(21 experiments)

Successful cross-project predictions

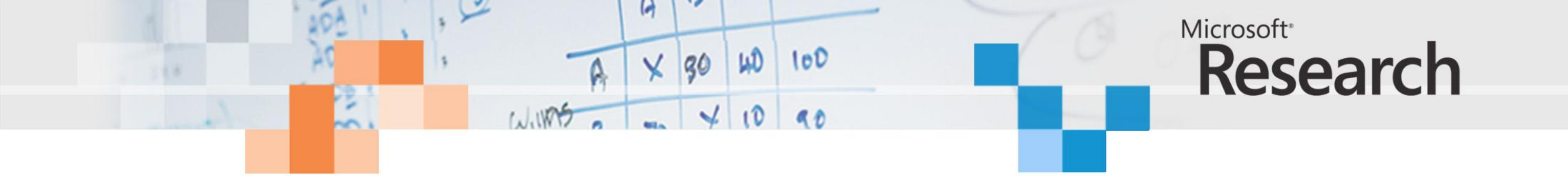


Characterizing projects

Domain
Domain name
Product uses database
Product is localized
Type of user interface

Characterizing projects

Domain	Product
Domain name	Name
Product uses database	Company
Product is localized	Intended audience
Type of user interface	Operating system
	Programming language
	Single prog. language
	Project uses C/C++
	Project uses C#
	Project uses Java
	First version
	Total lines of code



Characterizing projects

Domain	Product	Process
Domain name	Name	Open source
Product uses database	Company	Global development
Product is localized	Intended audience	Code reviews
Type of user interface	Operating system	Static checkers
	Programming language	Number of developers
	Single prog. language	
	Project uses C/C++	
	Project uses C#	
	Project uses Java	
	First version	
	Total lines of code	

Characterizing projects

Domain	Product	Process	Data
Domain name	Name	Open source	Level of analysis
Product uses database	Company	Global development	Number of observations
Product is localized	Intended audience	Code reviews	Median, maxium, and standard deviation of the metrics (18 metrics)
Type of user interface	Operating system	Static checkers	
	Programming language	Number of developers	
	Single prog. language		
	Project uses C/C++		
	Project uses C#		
	Project uses Java		
	First version		
	Total lines of code		

Comparing projects

- If characteristic is “Domain”, “Product”, “Prog. languages”, or “Level of analysis”
  - Same, Different
- If nominal (for example “Open Source”)
  - Both X, Both Y, Both Z, …, Different
- If numeric (for example “LOC”)
  - Less, Same, More

Comparing projects

	Characteristics			
Project	**Domain**	**Open source**	**Code reviews**	**LOC**
Train: Firefox	Browser	Yes	Yes	3.2M
Test: IE	Browser	No	Yes	2.3M
Similarity				

Comparing projects

	Characteristics			
Project	**Domain**	**Open source**	**Code reviews**	**LOC**
Train: Firefox	Browser	Yes	Yes	3.2M
Test: IE	Browser	No	Yes	2.3M
Similarity	Same			

Comparing projects

	Characteristics			
Project	**Domain**	**Open source**	**Code reviews**	**LOC**
Train: Firefox	Browser	Yes	Yes	3.2M
Test: IE	Browser	No	Yes	2.3M
Similarity	Same	Different		

Comparing projects

	Characteristics			
Project	**Domain**	**Open source**	**Code reviews**	**LOC**
Train: Firefox	Browser	Yes	Yes	3.2M
Test: IE	Browser	No	Yes	2.3M
Similarity	Same	Different	Both Yes	

Comparing projects

	Characteristics			
Project	**Domain**	**Open source**	**Code reviews**	**LOC**
Train: Firefox	Browser	Yes	Yes	3.2M
Test: IE	Browser	No	Yes	2.3M
Similarity	Same	Different	Both Yes	Less

Comparing projects

	Characteristics			
Project	**Domain**	**Open source**	**Code reviews**	**LOC**
Train: Firefox	Browser	Yes	Yes	3.2M
Test: IE	Browser	No	Yes	2.3M
Similarity	Same	Different	Both Yes	Less

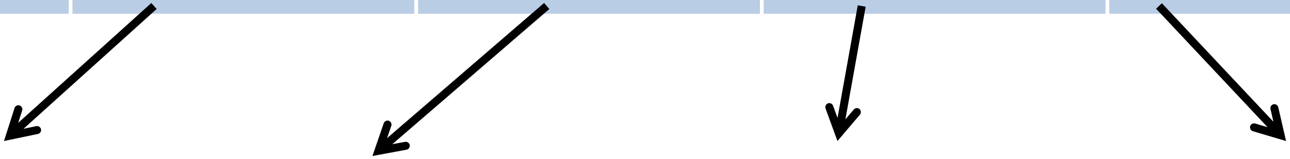
How are these similarities related to precision, recall, and accuracy?

Influence of individual factors

- For each characteristic and level
  - check with t-tests whether they influence precision, recall, accuracy
  - in total 375 tests; account for multiple hypothesis testing with Bonferroni correction
- Possible effects on precision, recall, accuracy
  - Increase
  - Decrease
  - No significant effect

Influence of individual factors

	Characteristics			
Project	**Domain**	**Open source**	**Code reviews**	**LOC**
Train: Firefox	Browser	Yes	Yes	3.2M
Test: IE	Browser	No	Yes	2.3M
Similarity	Same	Different	Both Yes	Less



Accuracy UP

Recall UP
Accuracy DOWN

Precision UP
Recall UP
Accuracy DOWN

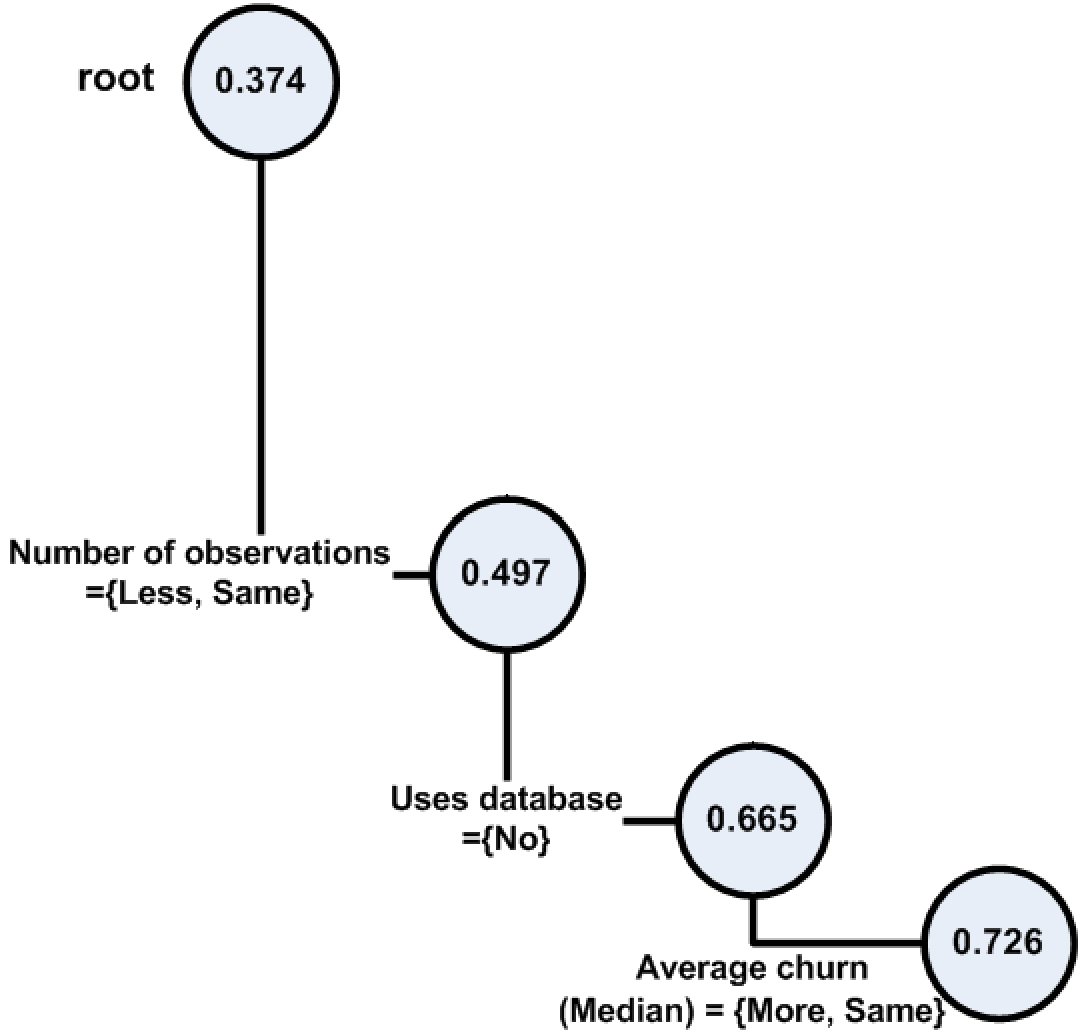
Precision DOWN

Influence of individual factors

Table 2. Nominal characteristics and how they influence precision, recall, and accuracy.

Factor	**Both**	Precision	Recall	Accuracy		Precision	Recall	Accuracy
Domain	Same:	—	—	UP	Different:	—	—	DOWN
Company	Apache: Microsoft:	— UP	DOWN —	— DOWN	Different:	DOWN	—	—
Product								
Open source	Yes: No:	— UP	DOWN —	UP DOWN	Different:	—	UP	DOWN
Global development	Yes: No:	— UP	DOWN —	UP —	Different:	—	UP	DOWN
Code reviews	Yes: No:	UP —	UP DOWN	DOWN UP				
Static checkers	Yes: No:	UP —	— DOWN	DOWN UP	Different:	—	UP	DOWN
Intended audience	Developer: End-user	DOWN UP	DOWN UP	— —				
Operating system	Multi: Windows:	— UP	DOWN —	UP DOWN	Different:	—	UP	DOWN
Type of user interface								
Product uses database	No:	UP	UP	—	Different:	DOWN	—	—
Product is localized	Yes:	UP	—	DOWN				

Decision tree for precision


root
0.374
Number of observations ={Less, Same}
0.497
Uses database ={No}
0.665
Average churn (Median) = {More, Same}
0.726

Project B

Project C

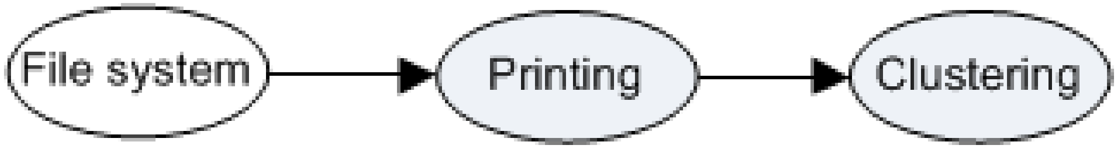
Project A

Additional decision trees

- Recall
  - *highest observed value 0.728 for*
  - global development (differs or both no),
  - median of relative number of pre-release bugs (more for test project), and
  - intended audience (different or both end-user).
- Accuracy
  - *highest observed value 0.843 for*
  - median of relative number of pre-release bugs (same),
  - operating system (both support multiple systems), and
  - median of relative added LOC (fewer or same in the test project).

Future work

- Replicate: more projects + characteristics
- Address further research questions:
  - Why are cross-project predictions sometimes not transitive?



  - How does the set of metrics influence the predictions? Does IE predict Firefox when different metrics are used?

Summary

- Out of the box, only 3.4% of cross-project defect predictions worked
- But we can do better! Precision + recall > 0.70
  - Identify factors that influence the success of cross-project predictions
  - Decision trees help to select the right projects

- http://research.microsoft.com/projects/esm

Recommended ESM reading

- Thomas Zimmermann, Nachiappan Nagappan, Harald Gall, Emanuel Giger, Brendan Murphy: Cross-project defect prediction. ESEC/SIGSOFT FSE 2009: 91-100
- Christian Bird, Nachiappan Nagappan, Premkumar T. Devanbu, Harald Gall, Brendan Murphy: Does distributed development affect software quality? ICSE 2009: 518-528
- Nicolas Bettenburg, Sascha Just, Adrian Schröter, Cathrin Weiss, Rahul Premraj, Thomas Zimmermann: What makes a good bug report? FSE 2008: 308-318
- Nachiappan Nagappan, Brendan Murphy, Victor R. Basili: The influence of organizational structure on software quality. ICSE 2008: 521-530
- Marc Eaddy, Thomas Zimmermann, Kaitin D. Sherwood, Vibhav Garg, Gail C. Murphy, Nachiappan Nagappan, Alfred V. Aho: Do Crosscutting Concerns Cause Defects? IEEE Trans. Software Eng. 34(4): 497-515 (2008)
- Thomas Zimmermann, Nachiappan Nagappan, Andreas Zeller: Predicting Bugs from History. Software Evolution 2008: 69-88
- Nachiappan Nagappan, Thomas Ball, Andreas Zeller: Mining metrics to predict component failure. ICSE 2006: 452-461