Microsoft

Theo: Test Effectiveness Optimization from History

Kim Herzig[£], Michaela Greiler[$], Jacek Czerwonka[$], Brendan Murphy[£]

[£]Microsoft Research, Cambridge
[$]Microsoft Corporation, Tools for Software Engineers (TSE)

Data extracted by the CODEMINE process managed by the TSE group (Redmond)

Improving Development Processes

Microsoft aims for shorter release cycles

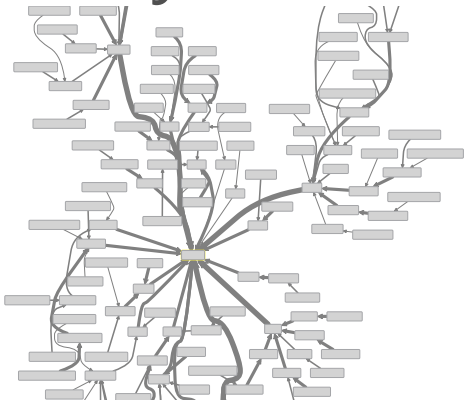
- Speed up development processes (e.g. code velocity)
- Maintaining / increasing product quality

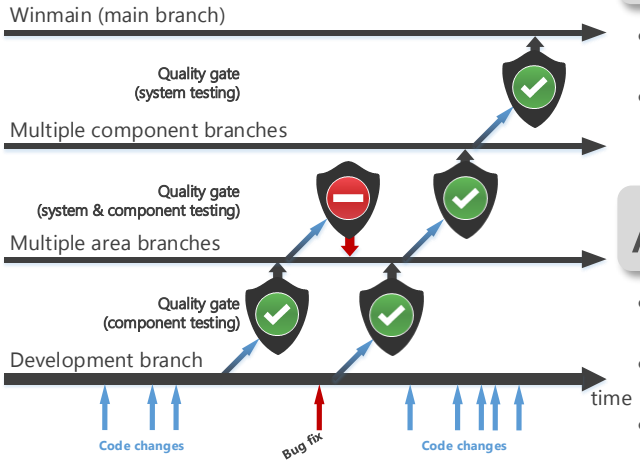
Empirical data to support & drive decisions

Joint effort by MSR & product teams

- **MSR Cambridge**: Brendan Murphy, Kim Herzig
- **MSR Redmond**: Tom Zimmermann, Chris Bird, Nachi Nagappan
- **TSE** Redmond: Jacek Czerwonka, Michaela Greiler
- **Windows, Windows Phone, Office, Dynamics product teams**

Why Software Testing?





Software testing is very expensive

- Thousands of gates executed, millions of test cases executed
- Different branches, architectures, languages, etc.

Current process aims for maximal protection

- Aims to find code issues as early as possible
- Slows down product development

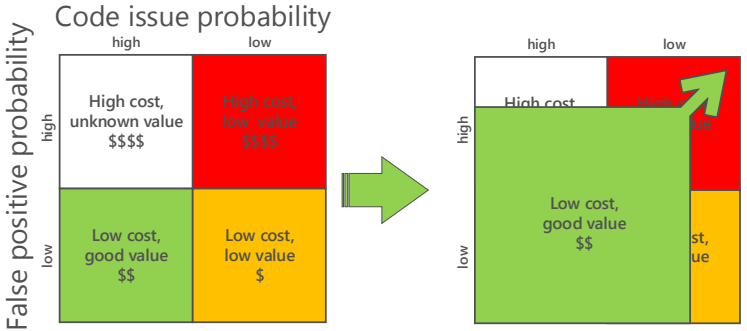
Actual problem

- We tend to repeat the same tests over and over again
- Too many false alarms (failures due to test and infrastructure issues)
- Each test failure further slows down product development

Project Goal

Reduce the number of test executions…

- **Identify tests** not failing at all or that are **more likely to produce false alarms**



…without sacrificing code quality

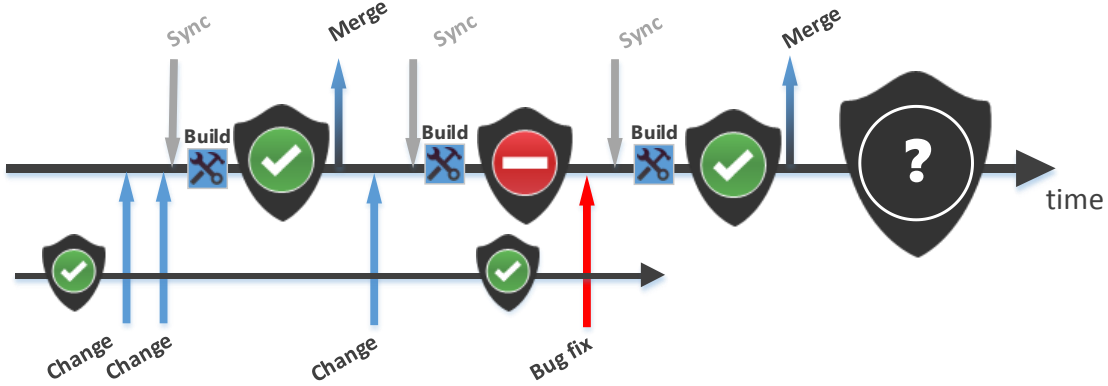
- **Run** every **test at least once before** integrating code change into **trunk branch** (winmain).
- We **eventually find all code issues** but **we take the risk of finding them later**.

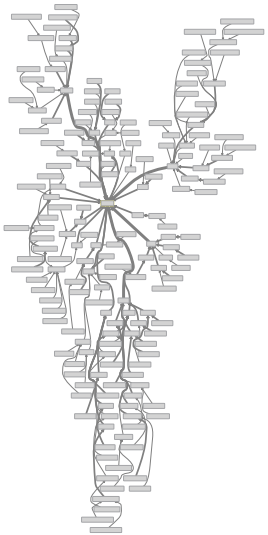
Dynamic, self-adaptive optimization model

Solution

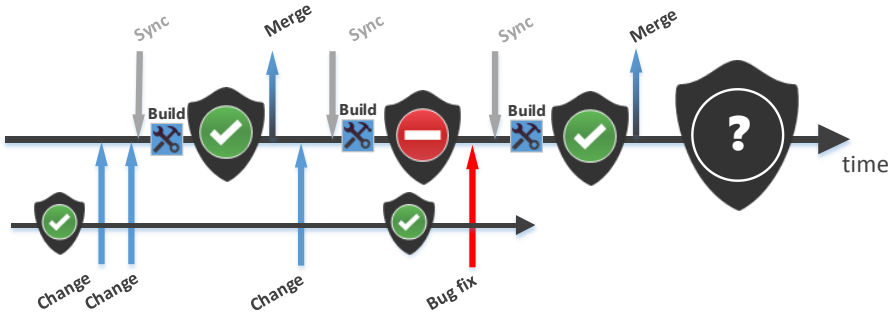
Dynamically reducing execution frequency of quality gate tests

- **Analyze historic events on branch**
  - Builds
  - Quality gate executions
- **Analyze past code changes**
  - From where do these changes come from?
- **Analyze past test results**
  - Passing tests, false alarms, detected code issues





Solution



Cost function decides when to skip tests.

$$Cost_{Execution} > Cost_{Skip} \,?\, \texttt{suspend} : \texttt{execute test}$$

$$Cost_{Execution} = Cost_{Machine/Time} * Time_{Execution} + \text{"Cost of potential false alarm"}$$

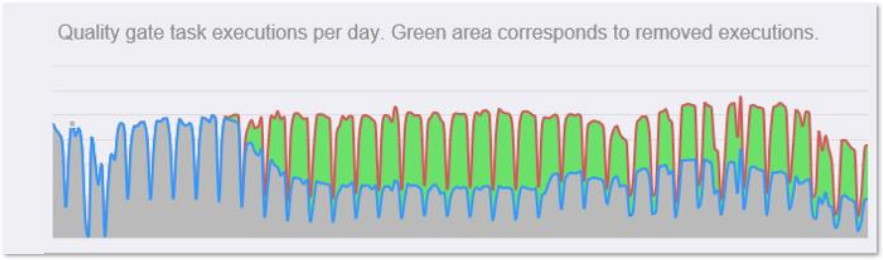
$$= Cost_{Machine/Time} * Time_{Execution} + (Prob_{FP} * Cost_{Developer/Time} * Time_{Triage})$$

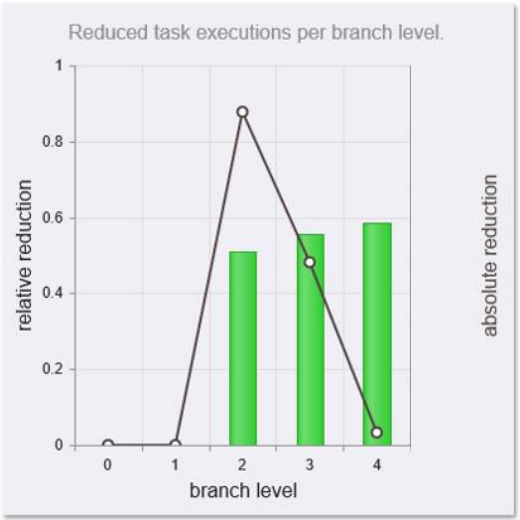
$$Cost_{Skip} = \text{"Potential cost of elapsing a bug to next higher branch level"}$$

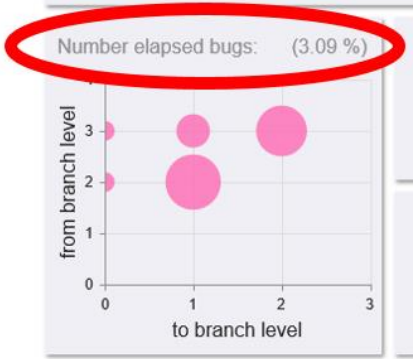
$$= Prob_{TP} * Cost_{Developer/Time} * Time_{Freeze\ branch} * \#Developers_{Branch}$$

Current Results

Simulated on Windows 8.1 development period
(BVT only)







Elapsed bug statistic

Number of bugs detected in time period:	
Number of elapsed bugs:	
Average level elapsing:	
Elapsed into winmain:	

Machine Cost:	-$1,567,607.76
Cost by False Positives:	-$59,852.80
Cost by Elapsed Bugs:	$78,848.00
Cost Model Balance:	-$1,548,612.56

Simulation Configuration

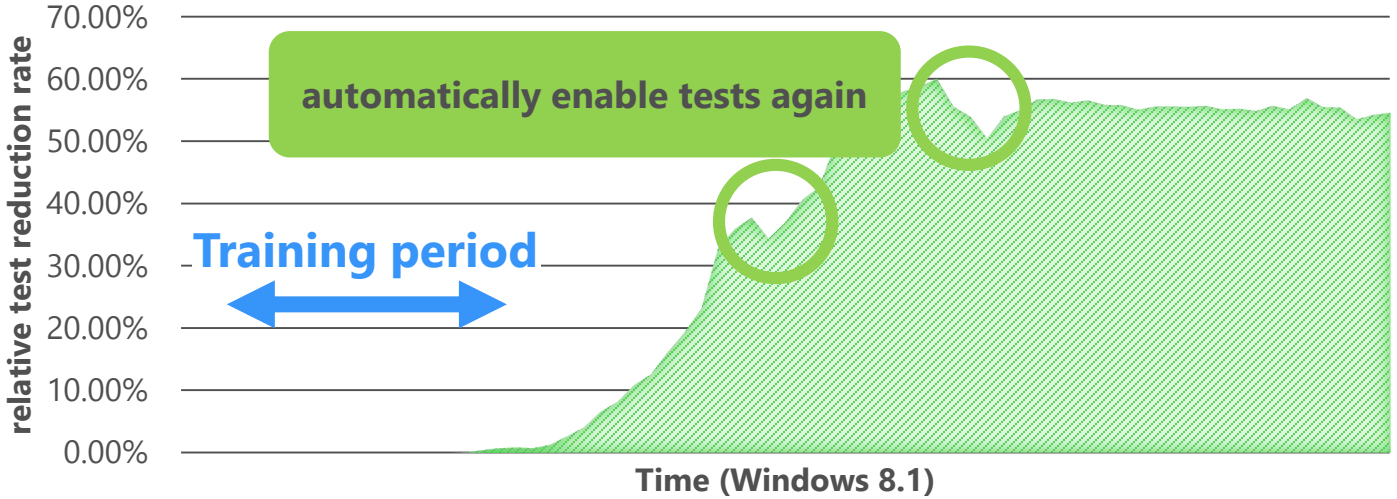
Cost machine / minute:	
Cost engineer / minute:	
Minutes per triage:	
Minutes blocked per bug:	

Windows

Dynamic, Self-Adaptive

Decision points are connected to each other

- **Skipping tests influences the risk factors** of higher level branches
- We **re-enable tests** if code quality drops (e.g. different milestone)

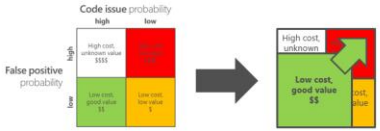


Microsoft Research

Project Goal

Reduce the number of test executions ...

- **Identify tests** not failing at all or that are **more likely to produce false failures**



... without sacrificing code quality

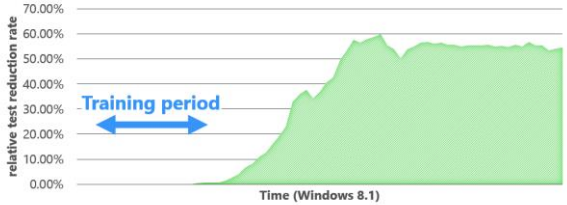
- **Run** every **test at least once before** integrating code change into **trunk branch** (winmain).
- We **eventually find all code issues but take risk** of finding them later (on higher level branches).

Dynamic, self-adaptive optimization model

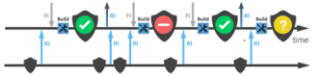
Dynamic, Self-Adaptive

Decision points are connected to each other

- **Skipping tests influences the risk factors** of higher level branches
- We **re-enable tests** if code quality drops (e.g. different milestone)



Solution



Skipping tests is risky. Using cost function to model risk

- **Comparing expected costs of running and not running test**

$$Cost_{Execution} > Cost_{Skip\ Test}\ ?\ \texttt{suspend}\ :\ \texttt{execute test}$$

$$Cost_{Execution} = Cost_{Machine/Time} * Time_{Execution} + \text{"Cost of potential false failure"}$$
$$= Cost_{Machine/Time} * Time_{Execution} + (Prob_{FP} * Cost_{Developer/Time} * Time_{Triage})$$

$$Cost_{Skip\ test} = \text{"Potential cost of elapsing a bug to next higher branch level"}$$
$$= Prob_{TP} * Cost_{Developer/Time} * Time_{Freeze\ branch} * \#Developers_{Branch}$$

Current Results

Simulated on Windows 8.1 development period (BVT only)

Windows

Microsoft Dynamics

Office

Windows Phone