

Industrial Strength Software Measurement

Audris Mockus, David Weiss {audris,weiss}@avaya.com

Topics

- Why measure?
 - On industrial scale?
 - On project scale?
 - On individual scale?
 - On country scale?
- The GQM model for measurement?
 - Goals, Questions, Measures
 - Evolution of goals
 - »The cost, quality, time to market rotation
 - Characteristics of industrial measurement
- Some of our goals
- Available data
- Some examples
 - Interval Quality
 - Registration Refactoring
 - Introduction of Test Coverage Tools

Measurement Approach: GQM

- Identify goals of software development process
 - Example: Produce more new features, fewer defects with fewer, more distributed, resources.
- Propose questions whose answers establish progress towards goals
 - Example: What is the ratio of new features to bug fixes by product? By site?
- Define measures that can be used to answer questions and that can be practically obtained for the software project
 - Example: Ratio of new feature MRs to bug fix MRs by product and site, normalized.
- Validate measures internally and externally
 - Example: remove tool generated artifacts and ensure the measure represents the phenomena it is intended to measure
- Establish infrastructure for data collection and analysis
 - Dashboards
 - Automated data collection and analysis



Software Changes: A Fabric of Measurement

MR = Modification Request

- For every change
 - Why was it made?
 - Who requested it?
 - Who made the change?
 - What was changed?
 - •When was it changed?
 - ••••
- States of an MR
 - Created (Developer, Tester, Support)
 - Assigned (MR Review Board)
 - Submitted (Developer)
 - Verified (Tester)
 - Completed (MR Review Board)
 - Accepted





- Software is created incrementally, via changes recorded by a VCS
- A delta is an addition and deletion of lines in a file

before:	after:
	// print N integers
int i=N;	int i=N;
while (i)	while (N > 0 && i > 0)
printf ("%d\n",i);	printf ("%d\n",i);

- one line deleted
- two lines added
- two lines unchanged

Change Hierarchy



labs

Characteristics of Industrial Measurement

- Meaningful
 - Show progress towards meeting goals
 - Trends, snapshots, figures of merit
- Nonintrusive
 - Don't add to developers' burden
 - Use data (already) collected for development purposes
- Automatable
 - Handle large amounts of data over long periods of time
 - 10s of thousands of records over decades
 - Automatically produce dashboards (website)
- Customizable
 - Each project can customize for its own version of goals
- Feasible
 - Data can be collected in an automated way
 - Verification possible



Some Key Feasible Measures

- Diffusion (# of subsystems, modules, files, developers)
- Size (# of lines added, deleted, and in the touched files)
- Diffusion & Size (# of deltas, MRs)
- Lead time (interval from start to completion)
- Purpose (Fix/New)
- Identity and experience (# of delta done in the past/recently/on a relevant part of the product) of creators

Some Benefits of Change Measures

+ Availability and cost

+ obtainable for all projects using CM

- + nonintrusive use existing data
- + Detail and coverage
 - + fine grained information at MR/delta level
 - + complete all parts of software are recorded
 - + massive larger than surveys/project measures
- + Stability and bias
 - +uniform slowly change over time
 - +unbiased no observer effect

Some Drawbacks of Change measures

- Require validation and careful interpretation
 - Data recorded for other purposes
 - Often need nontrivial datamining techniques
 - Different project support systems contain different attributes
 - Different projects may use the same system in different ways

Some Current Avaya Goals (1)

Significantly improve predictability

- Is predictability improving?
- What fraction of projects are on time?
- What are the factors associated with late projects?
- Significantly improve quality
 - Is quality improving?
 - What is the customers' perceptions of software quality?
 - What is the in-process quality?
- Rapidly produce new products (days and weeks instead of months and years)
 - Use a modular, family architecture
 - Take advantage of commonalities to compose and generate rather than hand code
 - Make production predictable
 - Continually predict, trial, and leverage expected future needs
 - Develop infrastructure for composing products from modules

Some Current Avaya Goals (2)

- Keep production within limits of resources, which are becoming more distributed
 - How distributed are resources? What's the trend?
 - Are there differences in productivity, quality among sites?
- Make globally distributed development (independent component development at different sites) an advantage
 - Are there differences in productivity, quality among sites?
- Introduce new software development processes
 - Agile development using automated test tools
- Ensure minimum of 60% test coverage for all new code

AVAYA Iabs

(A Few) Proposed Questions

- 1. What is the time and effort to create a new product? How predictable is product creation (time, effort, resources)?
 - Snapshot and trends
- What is the ratio of new modules to reused modules in a product?
 - Snapshot and trends
- For each (new?) product, which modules are new, which are reused unchanged, and which are reused wit adaptation or configuration?
- What is the ratio of new features to bug fixes by product? By site?
 - 1. Snapshot and trends
- What is the time and effort to create a new version of a new module? By site? How predictable is module creation (time, effort, resources)?
 - Snapshot and trends
- Is the architecture modular? Are interfaces suitable for use in many products and well-defined?
- Does the architecture match the organization (one site per module)?
- Is iterative development possible?
- What is the quality of products? What is the quality of modules?
 - Snapshot and trends

Plan



- Iterate on goals, questions
- Define data collection needs and resources
 - Who is responsible for assuring (accurate) data are collected?
- Trial data collection and analysis
- Iterate, revise, scale-up: create dashboards





R&D Locations with > 10 Staff

AVA

lahs

The number of International R&D locations has increased while the number of US locations has decreased between 2001 and 2005.





AVAYA

Growth of the Code Base

Growth of Avaya Code Base (C/C++/Java)

AVA

Yinhs





Predictability

AVAYA Predicting Software Development (50 sampled projects)



Example Staffing Profile





AVAYA

Copyright© 2006 Avaya Inc. All rights reserved

Distributed development, innovation, new features, legacy adaptation all contribute to delays







Interval Quality

STATISTICS NO.

Context of quality measurement

- Primary question:
 - Is the quality (reliability/availability) experienced by customers increasing/decreasing?
- Data sources
 - Customer inventory
 - Service calls, system alarms
 - Software changes
- Primary challenges
 - Storing, cleaning, and linking data sources
 - Designing a simple to understand and use quality measure



The probability of a customer observing a failure

- Is affected by:
 - Major/minor release
 - How soon after launch the system was installed
 - How long the system was running
 - The size and utilization of the system

The graph shows two factors:

- time after launch
- release



AVAYA Iab

Interval Quality

- Probability that a customer observes a failure within one, three, and six months after installation
 - 1 month
 - more noisy
 - allows seeing trends earlier
 - 6 months
 - more stable
 - have to wait for results

Drawback

- does not account for the proximity to launch
- Significant differences are marked with *, **, and ***
- Priorities changed from time-to-market to quality



Release



Interval Quality and Defect Density





Introducing New Technology

Context for Refactoring a Telecommunications Domain

- One domain of Avaya's IP telephony software
- 30 KLOC C++, ASN.1 generated code, 3rd party protocol stack within 7 MLOC system
- 40 different developers over 5 years
- Design degradation
- Constant change
 - inflow of defects from 5+ deployed releases
 - changes to implement new functionality for 2+ future releases

Software Refactoring



- Improve code structure without changing external behavior
- Sequence of simple behavior preserving code transformation steps
- For instance: "Extract Method": Turn a code fragment into a method whose name explains the purpose of the method

Refactoring Hypotheses

- H1: The customer reported defect rate will improve
 - Better ("collaboration"-based) design
 - Refactoring exposed pre-existing issues
- H2: The refactoring reduces the effort required to make changes
 - Information hiding
 - If design is good changes will be confined

Measures

- H1: The number of field MRs found and the root cause of these problems
- H2: Change effort and the amount of code that needs to be inspected to make the change

Defect Density



- The number of defects depends on release size
- Reported defects and submitted changes in registration domain
- Four pre- and one post-refactoring release

	Release Size	Field defects
pre-Refactoring	526	41
post-Refactoring	80	0

- Adjust for the shorter exposure of the last release: assume only 50% of defects in the first 7 months (41/2=20)
- Fisher's exact test p-value 0.06

Change Effort

Stage	#changes	avg(log(PersonMonths))
Pre-Ref.	292	1.12
Post-Ref.	151	1.23

Άν

- two-sample t-test of log(effort) p-value=.06
- Mann-Whitney of log(effort) p-value=.06
- The LOC in the refactored area decreased by 50%

Validation

- Reality
 - Verified the process
 - Verified selection of relevant changes (MRs)
 - Manually inspected all field MRs
 - Several operationalizations
- Modeling
 - Distribution: take logs or use nonparametric tests
 - Normalize by size where needed
 - Apply relevant models
- A case study precludes causal inference

Automated Test Coverage: Goals, Questions

- Estimate impact of introducing new tools, techniques
- Test coverage: Move detection of defects earlier



- Do we see the expected impact?
- What is the effect on effort, quality, schedule?

Coverage Report (Batch) AVAYA

🕘 New Tab 🛇 Coverage Report					×
Files	Coverage Report - All Files				^
All /usr/add-on/pavlov/defty-ws1/ /usr/add-on/pavlov/defty-ws1/	File 🖉	# Classes	Line Coverage	Branch Coverage	Priority
/usr/add-on/pavlov/defty-ws1/	/usr/add-on/pavlov/defty-ws1/rene/lip.pj/gmtce.ss/lip.p/assoc.d/act_assoc.c	1 (0%	0%	16
/usr/add-on/payloy/defty-ws1/	/usr/add-on/pavlov/deftv-ws1/rene/lip.pj/gmtce.ss/lip.p/assoc.d/add_assoc.c	14	46%	12%	7 =
/usr/add-on/pavlov/defty-ws1/	/usr/add-on/pavlov/defty-ws1/rene/lip.pj/gmtce.ss/lip.p/assoc.d/chk_assoc.c	1 (0%	0%	15
/usr/add-on/payloy/defty-ws1/	/usr/add-on/pavlov/defty-ws1/rene/lip.pi/gmtce.ss/lip.p/assoc.d/del_assoc.c	1 5	54%	27%	6
	/usr/add-on/pavlov/defty-ws1/rene/lip.pj/qmtce.ss/lip.p/assoc.d/qet_assoc.c	12	24%	7%	7
	/usr/add-on/pavlov/defty-ws1/rene/lip.pj/gmtce.ss/lip.p/assoc.d/init_assoc.c	11	100%	100%	0
All Files	/usr/add-on/pavlov/defty-ws1/rene/lip.pj/gmtce.ss/lip.p/link.d/al_loop.c	15	52%	50%	1
	/usr/add-on/pavlov/detty-ws1/rene/lip.pj/gmtce.ss/lip.p/link.d/alloc_link.c	17	77%	32%	3
Classes	/usr/add-on/pavlov/defty-ws1/rene/lip.pj/gmtce.ss/lip.p/link.d/alloc_map.c	13	30%	18%	11
defeultelese (201)	/usr/add-on/pavlov/defty-ws1/rene/lip.pj/gmtce.ss/lip.p/link.d/chk_map.c	10	U%	0%	10
defaultelass (0%)	/usr/add-on/pavlov/defty-wst/rene/lip.pl/dmtce.ss/lip.p/link.d/cr_loop.c	10	50%	40%	1
defaultclass (46%)	/usr/add-on/pavlov/defty-ws1/rene/lip.p/gmtce.ss/lip.p/link.d/des_loop.c	1 0	32%	92.76	
defaultclass (0%)	/usr/add-on/pavlov/defty-ws1/rene/lip.pj/gmtce.ss/lip.p/link.d/endpt_cap.c	14	48%	25%	2
defaultclass (54%)	/usr/add-on/pavlov/defty-wst/rene/lip.pj/gmtce.ss/lip.p/link.d/free_lipk_c	17	70%	40%	10
defaultclass (24%)	/usr/add-op/pavlov/defty-ws1/rene/lin pi/gmtce.cs/lin p/link.d/free_map.c	16	62%	46%	1
defaultclass (100%)	/usr/add-on/pavlov/defty-ws1/rene/lip.pi/omtce.ss/lip.p/link.d/hdl_rmve_n.c	12	20%	10%	2
defaultclass (52%)	/usr/add-on/pavlov/defty-ws1/rene/lip.pi/amtce.ss/lip.p/link.d/link_step.c	13	30%	10%	12
defaultclass (77%)	/usr/add-on/payloy/defty-ws1/rene/lip.pi/gmtce.ss/lip.p/link.d/move_link.c	10	0%	0%	17
defaultclass (30%)	/usr/add-on/pavlov/defty-ws1/rene/lip.pj/gmtce.ss/lip.p/link.d/pnc_check.c	12	27%	4%	6
defaultclass (0%)	/usr/add-on/pavlov/defty-ws1/rene/lip.pj/gmtce.ss/lip.p/link.d/pref_score.c	11	18%	5%	21
defaultclass (58%)	/usr/add-on/pavlov/defty-ws1/rene/lip.pj/gmtce.ss/lip.p/link.d/put_stim.c	14	41%	35%	4
defaultclass (52%)	/usr/add-on/pavlov/defty-ws1/rene/lip.pj/gmtce.ss/lip.p/link.d/ref_map.c	1 (0%	0%	5
defaultclass (33%)	/usr/add-on/pavlov/deftv-ws1/rene/lip.pi/gmtce.ss/lip.p/link.d/reg_map.c	1 7	70%	43%	1
defaultclass (48%)	/usr/add-on/pavlov/defty-ws1/rene/lip.pj/gmtce.ss/lip.p/link.d/upd_map.c	1 5	56%	16%	4
defaultclass (70%)	/usr/add-on/pavlov/defty-ws1/rene/lip.pj/gmtce.ss/lip.p/lip_map.dl/lip_l2p.c	16	64%	20%	3
defaultclass (62%)	/usr/add-on/pavlov/defty-ws1/rene/lip.pj/gmtce.ss/lip.p/lip_map.dl/lip_l2p_reg.c	<u> </u>	0%	0%	3
defaultclass (20%)	/usr/add-on/pavlov/defty-ws1/rene/lip.pi/gmtce.ss/lip.p/lip_map.dl/lip_p2l.c	14	48%	23%	3
defaultclass (30%)	/usr/add-on/pavlov/defty-ws1/rene/lip.pj/gmtce.ss/lip.p/lip_map.dl/lip_p21_reg.c	<u> </u>	0%	0%	3
N N 8 OF 1 file://X:/report/fra	User/add-on/naviow/defty-wet/cene/lin_ni/omtoe_ce/lin_n/lin_man_dl/lin_refrach_c me-summary-usr.add-on.paviow/defty-wet/cene/lin_pi.omtoe.ss.lin.p.lin_man.dl/lin_20.c.html	11	n.%.	0%	



Source Code View (GUI - Initial)

jexvantage v. ile Tools Vie	2.0 build 1003 sualization					Summary	Test Case	Properties	Undate
All-Nodes	O All-Edges							E derente	
0		3	0		6	ī.			10
* 0046 */ : * 0047 */ : * 0048 */ * 0049 */ · * 0050 */ * 0051 */	EXTERN_C s EXTERN_C s void audit. <mark>(</mark>	truct For a truct Which be co AccScl ⁱⁿ re incre appli	given node, is computed vered if thi d with a val ase the all- es to edges.	its co l based is node Lue X. 1 nodes c	olor is d on(at la is cover This imp. coverage	determine east) how red. Supp lies cove by at le	d by its w many node ose a node ring this ast X node	weight, s will is highli node will s. The sam	ghted c
* 0052 */ * 0053 */	LkLstEnd: D SOCK I	s_Idx lk_li DX ds;	ist; ,	/* link /* inde:	ed list x to D s	to search ocket tal	h */ ble */		
• 0054 */ • 0055 */	char int rc;	ifno;	:	/* boar	d interf	ace numb	== +/		
* 0056 */	for (ifno = 0; :	lfno < MAX_C	LANPT+1	; ifno++	• •			
0050 */	1 	_list = SII r (de = Sc	2(bd_x, ifn)	o); k listl	. Li bd:				
0061 */		ds != 3 ds = D	SOCK_END_LIS	T; next)					
0063 */	C		D socket[ds]].sock :	state ==	SOCK T	AREQ)		
0065 */ 0066 */		{	L						
0067 */			Send RSCLR for all ac	esetReq cept so	uest wit ckets on	h RSCL_D the boa	ETAIL cd.		
0069 */			:/ : = g_rsclre:	set_req	(bd_x, R	SCL_DETA	IL,		
eXVantage:	coverage	File: auditAc	cScktDlci.cpp	Line: 4	46 of 82	Coverage	: All-Nodes	Highlightin	g: All Prior

Automated Test Coverage: Feasibility

- Measured introduction of a test coverage/slicing tool
 - Usage logged: date, IP, login, invocation options
 - Changes to the codebase: login, file, date, size
 - Changes to the test code (JUnit) base
 - MRs: date, origin, developer
- Expected outcome
 - Logins with higher test tool usage have fewer MRs raised in testing and post-launch
- Complications
 - The coverage tool was run as a part of build process to create reports, so it was impossible to determine who used the reports
 - There was limited understanding about potential uses of the tool among developer population, some important functions were not utilized

Summary



- Why measure?
 - Estimate parameters important to business
 - Customer satisfaction, predictability, time and resources needed to create products
 - Evaluate progress on particular projects
 - When will it be ready? How many architects, developers, testers will we need?
 - Estimate capabilities and needs to understand areas for improvement
 - What problems do we need to solve to improve?
 - What is the impact of introducing new technology, methods?
 - Personal, Business, Country, World
- What's a good model for measurement?
 - Define goals first, then ask questions needed to evaluate progress towards achieving goals
 - Goals change over time interval, quality, cost
 - What are characteristics of industrial measurement?
 - Change data as a key information source
 - Automatibility, nonintrusiveness
- Some examples
 - Interval Quality
 - Registration Refactoring
 - Introduction of Test Coverage Tools