

eWorkshops: Testing Defect Reduction Heuristics against Expert Knowledge

Forrest Shull
Fraunhofer Center – Maryland / CeBASE

and

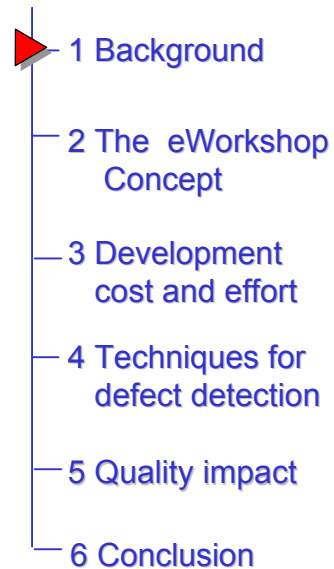
Vic Basili^{†‡}, Barry Boehm^{*}, A. Winsor Brown^{*}, Patricia Costa[†],
Mikael Lindvall[†], Dan Port^{*}, Ioana Rus[†], Roseanne Tesoriero[†], and
Marvin Zelkowitz^{†‡}

[†]Fraunhofer Center for Experimental Software Engineering, Maryland

^{*}University of Southern California, Center for Software Engineering

[‡]University of Maryland Empirical Software Engineering Group

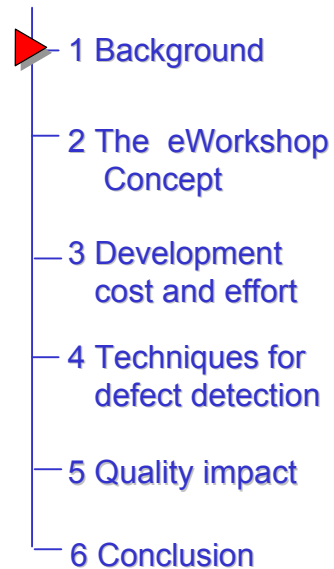
Contents



1	Background
2	The eWorkshop Concept
3	Development cost and effort
4	Techniques for defect detection
5	Quality impact
6	Conclusion

- Motivation for the eWorkshops to collect empirical info
- What is an eWorkshop?
- Some results of eWorkshops on defect reduction topics
 - Effect of defects on cost and effort
 - Defect reduction techniques
 - Quality impact of defects
- Conclusions: Why should researchers / practitioners care?

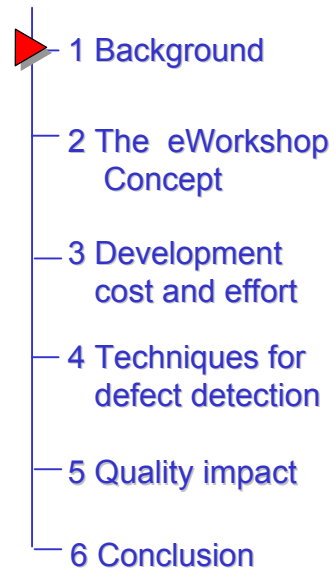
Contents



- 1 Background
- 2 The eWorkshop Concept
- 3 Development cost and effort
- 4 Techniques for defect detection
- 5 Quality impact
- 6 Conclusion

- Motivation
 - There has been at least 25 years of empirical studies and published measurements of software phenomena (that don't always get used by practitioners)
 - Individual developers have many years' worth of experience with software, specific to their context
 - Based on all this information, can we say we really understand something about the principles of software development?
- Overall goal
 - Formulate our understanding of some essential phenomena
 - ...reflecting a degree of consensus of the “experts”
 - ...useful for extrapolating some useful ideas for practice
- Goal is NOT to formulate “one size fits all” dogma
- Goal IS to study how & why things vary between environments

Contents



- 1 Background
- 2 The eWorkshop Concept
- 3 Development cost and effort
- 4 Techniques for defect detection
- 5 Quality impact
- 6 Conclusion

- First topic area was defect reduction
- History
 - Started from top-10 list of Boehm & Basili
 - Subjected heuristics to scrutiny by experts
 - Based on discussion, heuristics could be
 - Refined
 - Added
 - Restated
 - ... or a meta-statement could be made about the state of the knowledge
- Ongoing goal:
 - Revised / edited list of heuristics...
 - ... that summarizes the state of the knowledge and indicates the level of confidence
 - ... which can be compared to other data and continually expanded

Contents

1 Background

2 The eWorkshop
Concept3 Development
cost and effort4 Techniques for
defect detection

5 Quality impact

6 Conclusion

- Meetings among experts is a classical way of creating and disseminating knowledge.
 - Understand where consensus exists, level of confidence in existing results
- But:
 - Experts are spread all over the world
 - Workshops are generally not captured for further analysis
 - Certain personalities often dominate a discussion
- To overcome these problems, we designed the concept of the **eWorkshop**
 - An on-line meeting, which replaces *some* of the usual face-to-face workshop
 - Uses simple collaboration tools, supported by...

The eWorkshop

Process:

1. Choose a topic of discussion
2. Invite participants
3. Distribute Pre-meeting information sheet
4. Establish meeting codes – for meeting analysis
5. Publish synthesized info from pre-meeting sheets
6. Schedule pre-meeting training on tools
7. Set up control room
8. Conduct meeting
9. Post-meeting analysis and synthesis and storage
10. Dissemination of packaged knowledge

Roles:

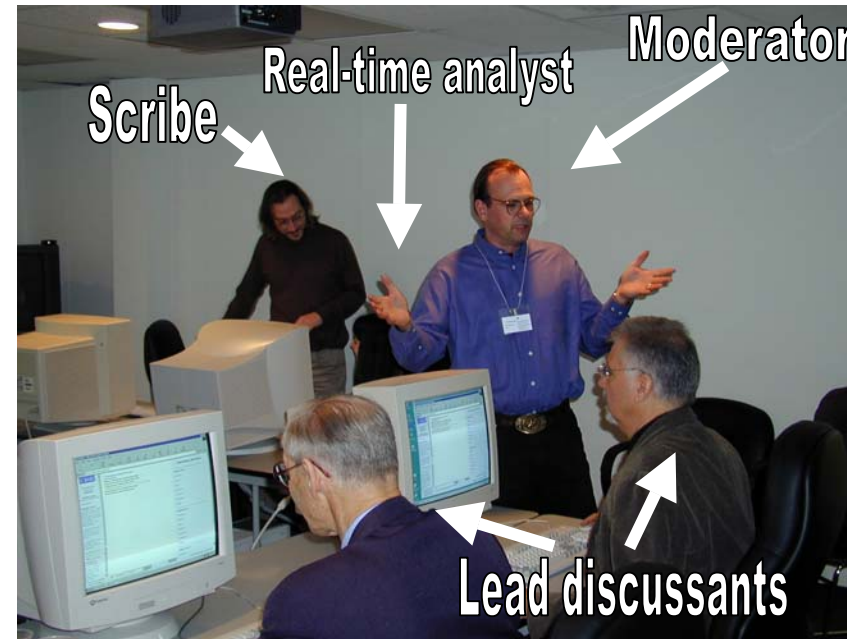
On the Scene:

- **Lead discussants** - leads the technical discussion
- **Participants** – 3 -15 experts in their respective domain
- **Moderator** - monitors and focuses the discussion (e.g., proposing items on which to vote) and maintains the agenda

Behind the Scene: Support team operating from “control room”

- **Director** - assesses and sets the pace of the discussion
- **Scribe** - summarizes the discussion on whiteboard
- **Analyst** - analyzes responses by type
- **Tech support**

Control Room:



Contents

1 Background

2 The eWorkshop
Concept

3 Development
cost and effort

4 Techniques for
defect detection

5 Quality impact

6 Conclusion

- 3 online eWorkshops run in 2001/2002
- 1 traditional workshop at 2002 Software Metrics Symposium
- Over 30 participants (developers, consultants, academics)
 - **Ed Allen** (MSU), **Frank Anger** (NSF), **Vic Basili** (UMD), **Barry Boehm** (USC), **Winsor Brown** (USC), **Sunita Chulani** (IBM), **Noopur Davis** (Davis Systems), **Michael Dyer** (Lockheed Martin), **Christof Ebert** (Alcatel), **Bill Elliott** (Harris Corp.), **Eileen Fagan** (Michael Fagan Associates), **Martin Feather** (JPL), **Liz Green** (Harris Corp.), **Ira Forman** (IBM), **Scott Henninger** (UNL), **Ross Jeffery** (U. New South Wales), **Philip Johnson** (U. Hawaii), **Oliver Laitenberger** (IESE), **Ray Madachy** (USC), **Audris Mockus** (Avaya), **Yoshihiro Matsumoto** (Toshiba), **Tom McGibbon** (ITT Industries), **James Miller** (U. Alberta), **James Moore** (MITRE), **Don O'Neill** (Don O'Neill Consulting), **Dan Port** (USC), **Stan Rifkin** (Masters Systems), **Dieter Rombach** (IESE), **Dan Roy** (STTP, Inc.), **Hossein Saiedian** (U. Kansas), **George Stark** (IBM Global Services), **Giancarlo Succi** (U. Alberta), **Gary Thomas** (Raytheon), **Otto Vinter** (independent software engineering mentor)

Contents

- 1 Background
- 2 The eWorkshop Concept
- ▶ 3 Development cost and effort
- 4 Techniques for defect detection
- 5 Quality impact
- 6 Conclusion

1: Finding and fixing a software problem after delivery is often 100 times more expensive than finding and fixing it during the requirements and design phase

Contents

1 Background

2 The eWorkshop
Concept3 Development
cost and effort4 Techniques for
defect detection

5 Quality impact

6 Conclusion

- Participants agreed that 100x was a useful heuristic for **severe** defects.
 - **117:1** (O'Neill); **137:1** (Matsumoto); Also agreement from Allen, Boehm, Chulani, Davis, French
- Effort multiplier was much less for **nonsevere** defects
 - **2:1** (Vinter, Boehm)
- Tolerance for paying that cost varies with business model:
 - Often this problem is addressed by *not* fixing defects after delivery, for certain types of systems. (Vinter; Brown)
 - In some domains, severe delivered defects, fixed quickly, can increase customer satisfaction. (Stark)
 - In some domains, severe delivered defects can have infinite cost
 - so when does development become cost-prohibitive? (Graham)
- We have no idea whether this is true for non-waterfall types of lifecycles, where early & late development phases get muddled.
 - Johnson

Contents

1 Background

2 The eWorkshop
Concept3 Development
cost and effort4 Techniques for
defect detection

5 Quality impact

6 Conclusion

- **1'**: Finding and fixing a **severe** software problem after delivery is often **100 times more expensive** than finding and fixing it during the requirements and design phase
- **1.1**: Finding and fixing **non-severe** software defects after delivery is about **twice as expensive** as finding these defects pre-delivery

Implication: Worthwhile to find defects early (especially the right types!)

Contents

- 1 Background
- 2 The eWorkshop Concept
- ▶ 3 Development cost and effort
- 4 Techniques for defect detection
- 5 Quality impact
- 6 Conclusion

2: About 40-50% of the effort on current software projects is spent on avoidable rework.

Contents

1 Background

2 The eWorkshop
Concept3 Development
cost and effort4 Techniques for
defect detection

5 Quality impact

6 Conclusion

- Significant effort is spent, but rates vary.
 - **7%** (Brothers); **40-50%** (Basili); **<= 60%** (Boehm); **20-80%** (O'Neill)
- For **higher-maturity projects**, the rate is less.
 - Thomas, Boehm, Clark suggested around **10-20%**;
 - Brothers disagreed
- For **higher-maturity products**, the rate is less.
 - French suggested **30%**
- Comparing rework costs is dangerous because different measures can be used, and certain aspects are hard to quantify.
- Demonstrates the benefits of metrics collection because rework costs are easy to see.
 - Rifkin, Basili, Davis

Contents

- 1 Background
- 2 The eWorkshop Concept
- ▶ 3 Development cost and effort
- 4 Techniques for defect detection
- 5 Quality impact
- 6 Conclusion

- **2': A significant percentage** of the effort on current software projects is typically spent on avoidable rework
- **2.1: The amount of effort spent on avoidable rework decreases as process maturity increases**
- **2.2: The amount of effort spent on avoidable rework decreases over time, as product maturity increases**

Implications:

- *We need to invest more in defect prevention*
- *Avoid streams of avoidable changes*
- *Don't discourage unavoidable changes*

Contents

- 1 Background
- 2 The eWorkshop Concept
- 3 Development cost and effort
- ▶ 4 Techniques for defect detection
- 5 Quality impact
- 6 Conclusion

6: Peer reviews catch 60% of the defects.

Contents

- 1 Background
- 2 The eWorkshop Concept
- 3 Development cost and effort
- 4 Techniques for defect detection
- 5 Quality impact
- 6 Conclusion

- Lots of data and expert consensus for 60% effectiveness
 - **50-70%** on average across all phases (Laitenberger),
 - **64%** early lifecycle (Elliott),
 - **70-80%** for experienced orgs (Rifkin),
 - **60%** in design and code reviews (Roy),
 - **60%** of requirements defects (Vinter),
 - **50%** (Miller),
 - **57%** (Jeffery),
 - **> 50%** (Nikora),
 - “**95%** of defects found before testing” (Fagan)
- **Regardless of domain or lifecycle phase**

Contents

- 1 Background
- 2 The eWorkshop Concept
- 3 Development cost and effort
- 4 Techniques for defect detection
- 5 Quality impact
- 6 Conclusion

- **Increased process maturity => increased effectiveness**
 - Across many companies, **50-65%** for less mature organizations, **70-80%** for structured software engineering, **85-95%** for disciplined practices (O'Neill)
- Keeping reviews in place as an effective practice is difficult (Nikora, Hantos, Graham). Can be mitigated by:
 - Introducing new people, fresh perspectives (Graham)
 - Protocols for making sure time well-spent (Graham, Hantos)
 - Providing guidelines for tailoring (Hantos)
 - Targeting documents with a high expected ROI (Mockus)
 - Targeting reviewers with backgrounds suited to the document under review (Jeffery)

Contents

- 1 Background
- 2 The eWorkshop Concept
- 3 Development cost and effort
- 4 Techniques for defect detection
- 5 Quality impact
- 6 Conclusion

- **6'**: Reviews catch **more than half** of a product's defects **regardless of the domain or lifecycle phase** during which they were applied

- **6a**: It is difficult to keep reviews in place as an organizational practice.

Implication: Peer reviews are a proven, effective defect reduction method, worth the effort required to keep them in place.

Contents

- 1 Background
- 2 The eWorkshop Concept
- 3 Development cost and effort
- 4 Techniques for defect detection
- **5 Quality impact**
- 6 Conclusion

4: About 80% of the defects come from 20% of the modules and about half the modules are defect free.

Contents

- 1 Background
- 2 The eWorkshop Concept
- 3 Development cost and effort
- 4 Techniques for defect detection
- **5 Quality impact**
- 6 Conclusion

- Supporting data and expert consensus for the 80/20 rule:
 - Typically only **10% of modules** have defects after system test (Roy);
 - only **10% of changed telecomm modules** contributed defects (Allen);
 - **20% of changed modules** contributed 80% of defects (Rifkin);
 - **20% of modules** contribute 40-80% of defects, depending on product line (Ebert);
 - **19% of modules** contributed 70% of defects (Vinter);
 - **40% of files** contributed 100% of faults, early release; **4% of files** contributed 100% of faults, late release (Weyuker);
 - Relationship varies with development processes, quality goals, maturity of software...
- But can those 20% of modules be targeted?
 - 20% of modules often usually contain most of the system code (Mockus)
 - **Most-changed modules** often appear to be most defect-prone (Mockus, French)

Contents

- 1 Background
- 2 The eWorkshop Concept
- 3 Development cost and effort
- 4 Techniques for defect detection
- ▶ 5 Quality impact
- 6 Conclusion

- Almost **no modules** from many systems were defect-free **during development** (O'Neill)
- **40%** of all modules in an embedded system were defect-free **after delivery**

Contents

- 1 Background
- 2 The eWorkshop Concept
- 3 Development cost and effort
- 4 Techniques for defect detection
- ▶ 5 Quality impact
- 6 Conclusion

- **4'**: As a general rule of thumb, **80% of a system's defects come from 20% of its modules**. However, the relationship varies based on environment characteristics such as **processes used and quality goals**
- **4''**: **During development**, almost **no modules** are defect-free as implemented
- **4'''**: **Post-release**, about **40% of modules** may be defect-free

Implication: Worthwhile to identify classes of error-prone modules – for deciding where to put extra attention, not for limiting testing.

Contents

- 1 Background
- 2 The eWorkshop Concept
- 3 Development cost and effort
- 4 Techniques for defect detection
- 5 Quality impact
- 6 Conclusion

- Implications for researchers
 - Identified areas where little or no data being collected (downtime resulting from defects, effect of disciplined personal practices)
 - How many of these heuristics hold for “non-waterfall lifecycles,” e.g. XP?
 - What are the root causes for rework?
 - What are the root causes of defects?
- Implications for practitioners
 - Useful for decision support (?)
 - Is your environment represented?
 - Are your results similar?

Contents

1 Background

2 The eWorkshop
Concept

3 Development
cost and effort

4 Techniques for
defect detection

5 Quality impact

6 Conclusion

- For full description of previous discussions:
 - www.cebase.org/www/researchActivities/defectReduction/index.htm
- Or for any other comments, suggestions, questions:
 - fshull@fc-md.umd.edu