Building an Experience Base for Software Engineering: A Report on the First eWorkshop

Victor Basili,

Roseanne Tesoriero, Patricia Costa,

Mikael Lindvall, Ioana Rus,

Forrest Shull, Marvin Zelkowitz

Fraunhofer Center for Experimental Software Engineering Maryland

CeBASE

Need for Empirical Software Engineering

- Software is too fragile, unpredictable (Presidential Commission,PITAC)
- "No surprise" software development (NSF Workshop Report)
- Industry needs a quantitative basis for
  - choosing among life cycle models and development approaches
    - Rapid/evolutionary/spiral/adaptive development
    - COTS/legacy/agent/portfolio-based systems
    - Open-source; extreme programming; architecture-based development
  - tailoring them for specific needs
    - testing for detecting a specific defect class
    - designing for evolving a particular feature

CeBASE

Need for Empirical Software Engineering

- Software development teams need to understand the right models and techniques to achieve high dependability in their project
- For example:
  - When is technique X more effective than technique Y?
  - What is the cost of applying technique X in this environment?
  - How should you tailor technique X for your environment?
- Researchers need feedback to better understand how to evolve their techniques for practice.

CeBASE

CeBASE Project

The goal of the Center for empirically-Based Software Engineering (CeBASE) is to accumulate **empirical models** to provide validated guidelines for selecting techniques and models, recommend areas for research, and support education

A first step is to build an empirical **experience base**
continuously evolving with empirical evidence
to help us identify what affects cost, reliability, schedule,...

To achieve this we are
Integrating existing data and models
Initially focusing on new results in two high-leverage areas
Defect Reduction, e. g. **reading techniques** (see top ten issues)
COTS Based Development (see top ten issues)

CeBASE

Examples of Using Empirical Results for development, research, education Technique Tailoring

Is tailoring the reading process associated with an inspection worth the effort?

- Procedural inspections, based upon **specific goals**, will find defects related to those goals, so inspections can be customized. (UMD)

Implications for empirically based **software development process**:

- The better you can articulate your goals, the more effectively you can choose and tailor process.

Implications for **software engineering research**:

- It is important to empirically study the effects of processes on product

Implications for **software engineering education**:

- Don't teach that there is a one size fits all process; teach how to tailor processes

CeBASE

Examples of Using Empirical Results for development, research, education Technique Selection Guidance

When should you use a procedural approach to code reviewing?

- For a reviewer with an average experience level, a **procedural approach** to defect detection is more effective than a less procedural one (UMD, USC)

Implications for empirically based **software development process**:

- Experts might be more effective working on their own but most people should apply a procedural approach. Novices need training.

Implications for **software engineering research**:

- How can we improve document reading procedures based upon how experts analyze documents?

Implications for **software engineering education**:

- Effective procedures that can be taught for reviewing documents

CeBASE

Examples of Using Empirical Results for development, research, education Technique Selection Guidance

When are peer reviews more effective than functional testing?

- **Peer reviews** are more effective than functional testing for faults of **omission** and **incorrect specification** (UMD, USC)

Implications for empirically based **software development process**:

- If, for a given project set, there is an expectation of a larger number of faults of omission or incorrect facts than use peer reviews

Implications for **software engineering research**:

- How can peer reviews be improved with better reading techniques for faults of omission and incorrect fact?

Implications for **software engineering education**:

- Teach how to experiment with and choose the appropriate analytic techniques

CeBASE

Motivation for eWorkshops

- To build an empirical **experience base** of continuously evolving empirical evidence we need to elicit, share, and integrate knowledge and data from experts in the field

- Meetings among experts are a classical way of creating and disseminating knowledge. By analyzing these discussions, knowledge can be created and knowledge can be shared
- But:
  - Experts are spread all over the world and hard to get people to travel to meet
  - Workshops are usually oral presentations and discussions that generally are not captured for further analysis
  - Certain personalities often dominate a discussion

CeBASE

Motivation for eWorkshops

- To overcome these problems we designed the concept of the *eWorkshop*, using the facilities of the Internet
- This presentation describes the process of running an eWorkshop and the results from the first eWorkshop, which was held March 16, 2001

CeBASE

The eWorkshop

- An on-line meeting, which replaces the usual face-to-face workshop
- Uses a Web-based chat-application, structured to accommodate a workshop needs without becoming an unconstrained on-line chat discussion
- The goal is to synthesize new knowledge from a group of experts as an efficient and inexpensive method in order to populate the CeBASE experience base
- It uses simple collaboration tools, minimizing potential technical problems and decreasing the time it takes to learn the tools
- It requires a defined process, a set of roles and a control room

CeBASE

The eWorkshop Process (Organizing Team View)

Organization of the workshop follows a protocol :

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

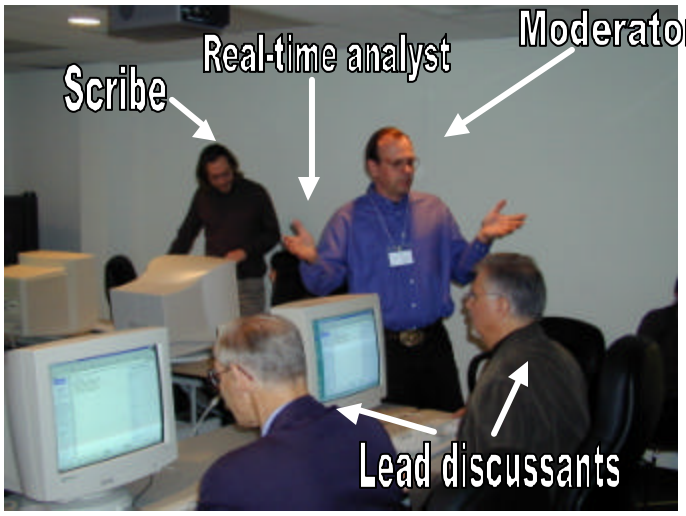
CeBASE

The eWorkshop Roles

- **Lead discussants** - leads the technical discussion
- **Participants** - experts in their respective domain
- Support team operating from a single **control room**
  - **moderator** - monitors and focuses the discussion (e.g., proposing items on which to vote) and maintains the agenda
  - **director** - assesses and sets the pace of the discussion
  - **scribe** - highlights agenda items and summarizes the discussion and updates the whiteboard
  - **tech support** - handles problems that might occur with the tools
  - **analyst** - codes the responses based upon a predefined taxonomy

CeBASE

Example of eWorkshop Control Room



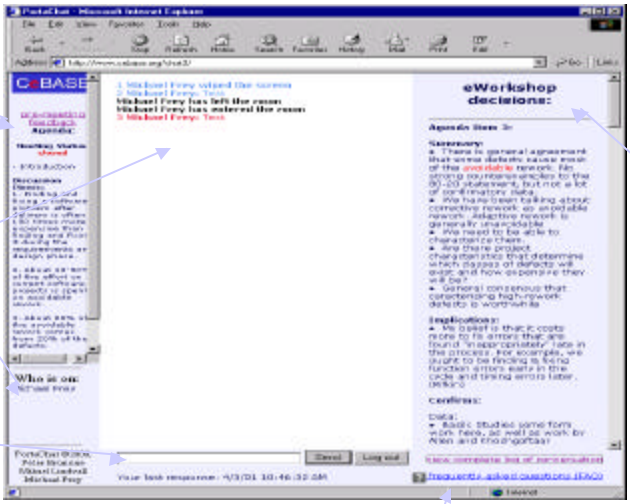
CeBASE

The eWorkshop Tools

- The main tool is the web-based chat-application, adapted from some open source software
- The **chat tool** allows participants to
  - be located remotely from the control room
  - create identities based upon their names
  - submit statements to a message board
  - carry on a discussion online by responding to statements on the message board following a set of conventions
  - vote and measure level of consensus
- All statements are automatically captured in real-time, allowing them to be analyzed in real-time and afterwards in more depth

CeBASE

Chat Tool



The eWorkshop Tools

- The chat tool has six main areas:
  - **Agenda**: indicates the status of the meeting
  - **Input panel**: enables participants to type statements during the discussion
  - **Message board**: forms the meeting discussion area
  - **Whiteboard**: synthesizes a summary of the discussion and it is controlled by the scribe
  - **Attendee list:** Indicates who is currently participating
  - **FAQ:** a list of questions and answers regarding the tool and the process.

The First CeBASE eWorkshop

- Held March 16, 2001, (11AM - 1PM US EST)
- 1. Choose Topic: Defect Reduction (Top Ten List - Boehm/Basili, IEEE Computer)
- Subtopic Goal: Evolve a proposed set of empirical models/heuris tics related to software **defect reduction cost and effort**
- Discussion Items (Items 1, 2, 3 of the Defect Reduction Top Ten):
  - Finding & fixing defects after delivery is 100x more expensive than finding & fixing during requirement and design phase
  - About 40-50% of development effort is spent on avoidable rework
  - 80% of rework comes from 20% of defects

CeBASE

The First CeBASE eWorkshop Process

2. Invite Participants: mix of industry and research

3. Solicit pre-meeting feedback from participants:
   - Do you have data that confirms/refutes the model?
   - Can you help refine the model? (e.g. by suggesting more accurate guidelines or specifying to what types of systems it applies)
   - Can you suggest other references or citations relevant to the model?
   - Can you state the implications for practice, if this model is true?

4. Establish Meeting Codes for Analysis, e.g.,

CeBASE

The First CeBASE eWorkshop Process

5. Aggregate & disseminate positions regarding the discussion items

6. Tool Training
   Test the chat tool, demonstrate the FAQ

7. Set Up Control Room

8. Conduct Meeting

9. Analysis and Synthesis
   Used VQI data mining tool

10. Dissemination of the results

CeBASE

The First CeBASE eWorkshop eWorkshop Evaluation

- Goal: Assess the eWorkshop's effectiveness in strengthening empirical software engineering knowledge
- Questions:
  - Q1: Was the chat an effective way to discuss the topics?
  - Q2: Did the meeting result in new information on the topic of defect reduction?
- Sources available for analysis:
  - the transcript of the actual text from the meeting
  - the scribe summary approved by the participants
  - the analyst's coding of each captured response
  - the users by session

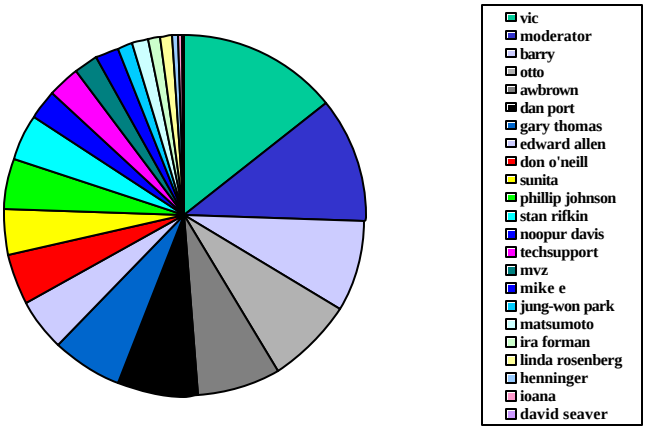
CeBASE

The First CeBASE eWorkshop
eWorkshop Evaluation

- 19 “real” participants (+visitors) – 11 contributed significantly to discussion
- 11 different references to citations
- Description of discussion:
  - 13% of responses related to data (73 responses)
  - 19% on voting
  - 17% on rework effort
  - 13% discussing definitions
  - 10% on overhead
- So…
  - There was no monopolizing voice among participants
  - Most responses were content-related
- Most of participants reported it was a good experiences and that they would like to do it again

CeBASE

Chat Participant Distribution



11 participants contributed (4% - 15% each)

CeBASE

Analysis of 1st eWorkshop Defect Reduction – Item 1

- Finding & fixing defects after delivery is 100x more expensive than finding & fixing during requirement and design phase
  - Participants agreed that 100x was a useful heuristic for *severe* defects.
    - 117:1 (O'Neill); 137:1 (Matsumoto); Allen; Davis; Boehm; Chulani
  - Effort multiplier was much less for non-severe defects
    - 2:1 (Vinter)
  - Often this problem is addressed by *not* fixing defects after delivery, for certain types of systems.
    - Vinter; Brown
  - We have no idea whether this is true for non-waterfall types of lifecycles, where early & late development phases get muddled.
    - Johnson

CeBASE

Analysis of 1st eWorkshop Defect Reduction – Item 2

- About 40-50% of development effort is spent on avoidable rework
  - Significant effort is spent, but rates vary.
    - 40-50% (Basili); <= 60% (Boehm); 20-80% (O'Neill)
  - For higher-maturity projects, the rate is around 10-20%.
    - Thomas, Boehm, Clark (?)
  - 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
  - The implication is that we need to invest more in defect prevention.
    - Vinter, Davis, Thomas, Boehm (5:1 or 10:1 payoff)

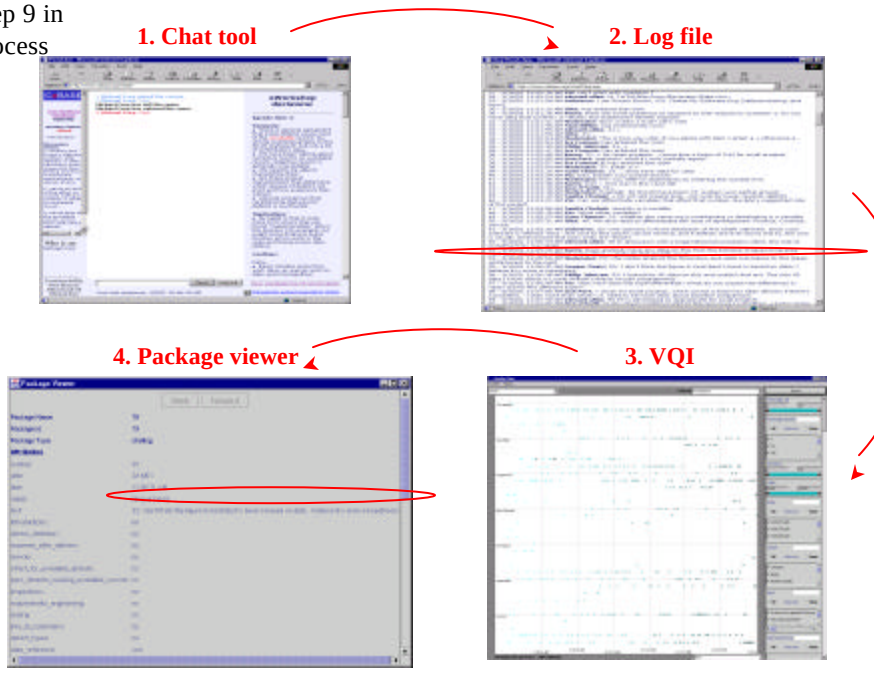
CeBASE

Analysis of 1st eWorkshop Defect Reduction – Item 3

- 80% of rework comes from 20% of defects
  - *Most* rework comes from *relatively few* defects.
    - Thomas, O'Neill, Rifkin, Allen, Basili
  - Rework Definition?
    - Broad definition: changes in OS, DB, customer base…
  - Defect Definition?
    - Any change made to software (Brown); corrective and performance-related changes only (Rifkin)
  - Certain defects are more likely to cause massive rework.
    - Architecture-breakers (Boehm); defects found "inappropriately" late in the lifecycle (Rifkin)

CeBASE

Step 9 in process



The First CeBASE eWorkshop Participants' Feedback

- The majority of the participants
  - liked the eWorkshop
  - "good way to discuss," "worthwhile and stimulating" and "a relatively easy way for a group to get a discussion going."
  - would participate again and would recommend this discussion vehicle to others
  - thought that more pre-meeting preparation would be a great benefit
  - there would be a benefit in sharing their positions relative to the topics on the agenda, together with arguments to support or refute them, such as data and references, prior to the meeting.
- Difficulties people reported were related to
  - the tool
  - their lack of preparation in using the technology before the meeting.

CeBASE

The Second CeBASE eWorkshop

- Held July 16, 2001 (11:30AM to 1:30PM, US EST)
- Subtopic Goal: Evolve a proposed set of empirical models/heuristics related to **impact defects have on software**
- Topic: Defect Reduction (Top Ten List Items 4, 5, 9, 10)
  - About 80 percent of the defects come from 20 percent of the modules, and about half the modules are defect free
  - About 90 percent of the downtime comes from, at most, 10 percent of the defects
  - All other things being equal, it costs 50 percent more per source instruction to develop high-dependability software products than to develop low-dependability software products
  - About 40 to 50 percent of user programs contain nontrivial defects
  - Models/Heuristics supported
  - Results on CeBASE web site (cebase.org)
- What we changed

CeBASE

The Second CeBASE eWorkshop

- Subtopic Goal: Evolve a proposed set of empirical models/heuris tics related to **impact defects have on software**
- Results
  - Models/Heuristics supported and refined
  - Results on CeBASE web site (cebase.org)
- What we changed
  - Tool Training - added more
  - Context Preparation - made pre-meeting response requests clearer
  - Analysis - Simplified the categories
  - Chat Tool -
    - Easier to see who is currently online
    - easier to log in
    - easier to analyze discussion

CeBASE

Conclusions

- CeBASE has an ambitious goal: collecting relevant empirically-based software engineering knowledge
- The results from the first two eWorkshop generally
  - refined the relevant items on the top ten Defect Reduction Issue List
  - provided additional references and data that seek to classify when specific defect reduction heuristics are applicable.
- The eWorkshop is one mechanism for supporting CeBASE goals
  - provides short term analysis and synthesis
  - provides initial information for the experience base
  - allows us to start long term analysis and synthesis process of
    - generating the empirical models
    - make recommendations to industry, research, education

CeBASE

Next Steps

- The Third CeBASE eWorkshop
  - Nov 12, 2001 @ noon
  - Topic: Effective methods for finding defects
    - (Defect Reduction Top 10 issues 6,7, and 8)
  - Apply to participate
- Continue to evolve the eWorkshop and supporting tools
- Next Workshops will begin top ten COTS Issues
- Face to face Workshop in Spring of 2002

CeBASE

Questions

- Can I participate in eWorkshops?
  - Please apply
- Where can I find the knowledge?
  - cebase.org
- Can I comment on the knowledge?
  - Yes
- Can I join CeBASE?
  - Find application at cebase.org
- Can I use eWorkshops ?
  - Yes,

CeBASE

Questions
Can I use eWorkshops?

- Some Advice
  - pick a focused and small topic area
  - make sure participants are trained
  - real-time white board summary is critical
  - two hours is about as much as you can do
  - control room roles are important

CeBASE