A Light-Weight Process for Capturing and Evolving Defect Reduction Experience

Victor Basili,

Mikael Lindvall, Forrest Shull

Fraunhofer Center for Experimental Software Engineering Maryland

CeBASE

Software Engineering Needs

- Industry needs a 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

Software Engineering Needs

- 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 Center for empirically-Based Software Engineering (CeBASE) accumulates **empirical models** to provide validated guidelines for selecting techniques and models

CeBASE works on:

- Integrating existing data and models
- Initially focusing on new results in two high-leverage areas
  - **Defect Reduction**
  - **COTS Based Development**

The CeBASE strategy is to build an empirical **experience base**

- continuously evolving with empirical evidence
- to help us identify what affects cost, reliability, schedule,...

CeBASE

Examples of Using Empirical Results: 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 then use peer reviews

Implications for **software engineering research**:

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

Building an Experience Base: Dust to Pearls Approach

Main principle: Capture the knowledge "dust" that developers use and exchange on a daily basis, and make it available throughout the community

- *Immediately* with minimal modification
- *Long-term* after analysis and synthesis.

The mechanisms used are based on

- Dust: The AnswerGarden Approach (Short-term needs based, Ad Hoc!, organic growth, fine-granular items)
- Pearls: The Experience Factory Approach (Long term needs, analysis&synthesis, feedback loops, separate groups)

Analyze and synthesize the dust and turn it into knowledge pearls

CeBASE

Building an Experience Base: Dust to Pearls (Light-Weight) Approach

How do you seed an empty Experience Base?

- Build a warehouse of knowledge
- Build the base and evolve in real-time (light-weight)

Light-weight approach based on short cycles in the QIP:

- Invest less now, harvest some now,
- Evaluate, improve
- Invest more later, harvest more later

An approach that

- Encourages experts to contribute to the Experience Base by quickly adding value and feeding back results
- Is light-weight enough not to add any significant work to already busy experts
- Allows the Experience Factory Group (CeBASE team) to analyze and evolve the content of the Experience Base over time

CeBASE

Building an Experience Base: Dust-to-Pearl Examples

In CeBASE we use the approach in several different ways:

Question -> Answer in E-mail -> FAQ -> set of FAQs -> Process Description

Incident -> Captured Lessons Learned -> set of LLs -> Best Practices

Defect -> Bug Report -> set of Bug Reports -> Problem areas ->
Solutions -> Design Rules -> Development Practices

eWorkshop Chat statements -> Real time analysis -> Summary ->
Best Practices, Lessons Learned (the following presentation)

CeBASE

Motivation for eWorkshops

- Goal is to learn from the experience of others in terms of problems and solutions and build an empirical **experience base** of continuously evolving empirical evidence we can 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 it is 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

- eWorkshop:
  - an on-line meeting using a Web-based chat-application, that uses simple collaboration tools, minimizing potential technical problems and decreasing the time it takes to learn the tools
  - 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

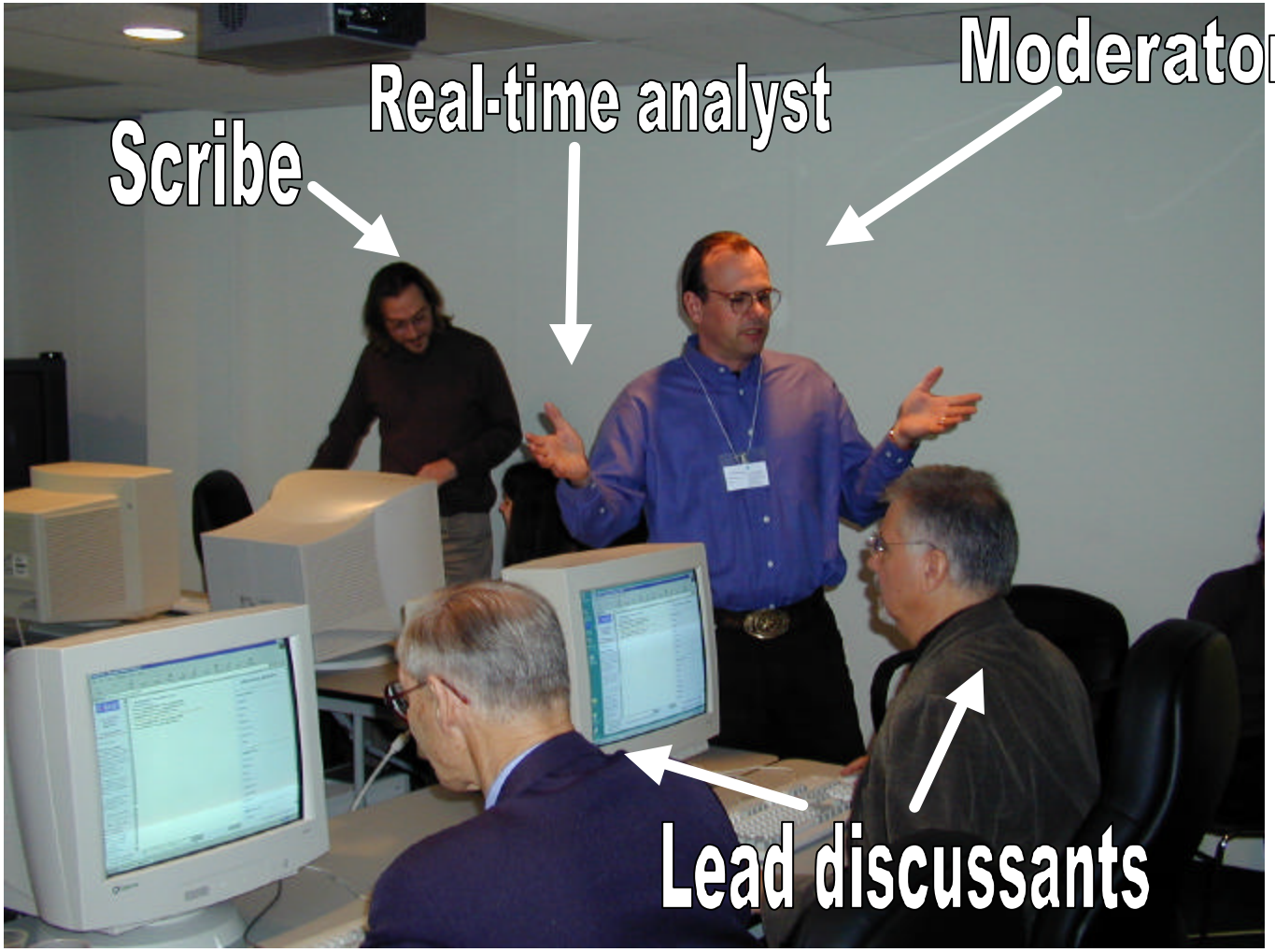
CeBASE

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, 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



The eWorkshop Tools

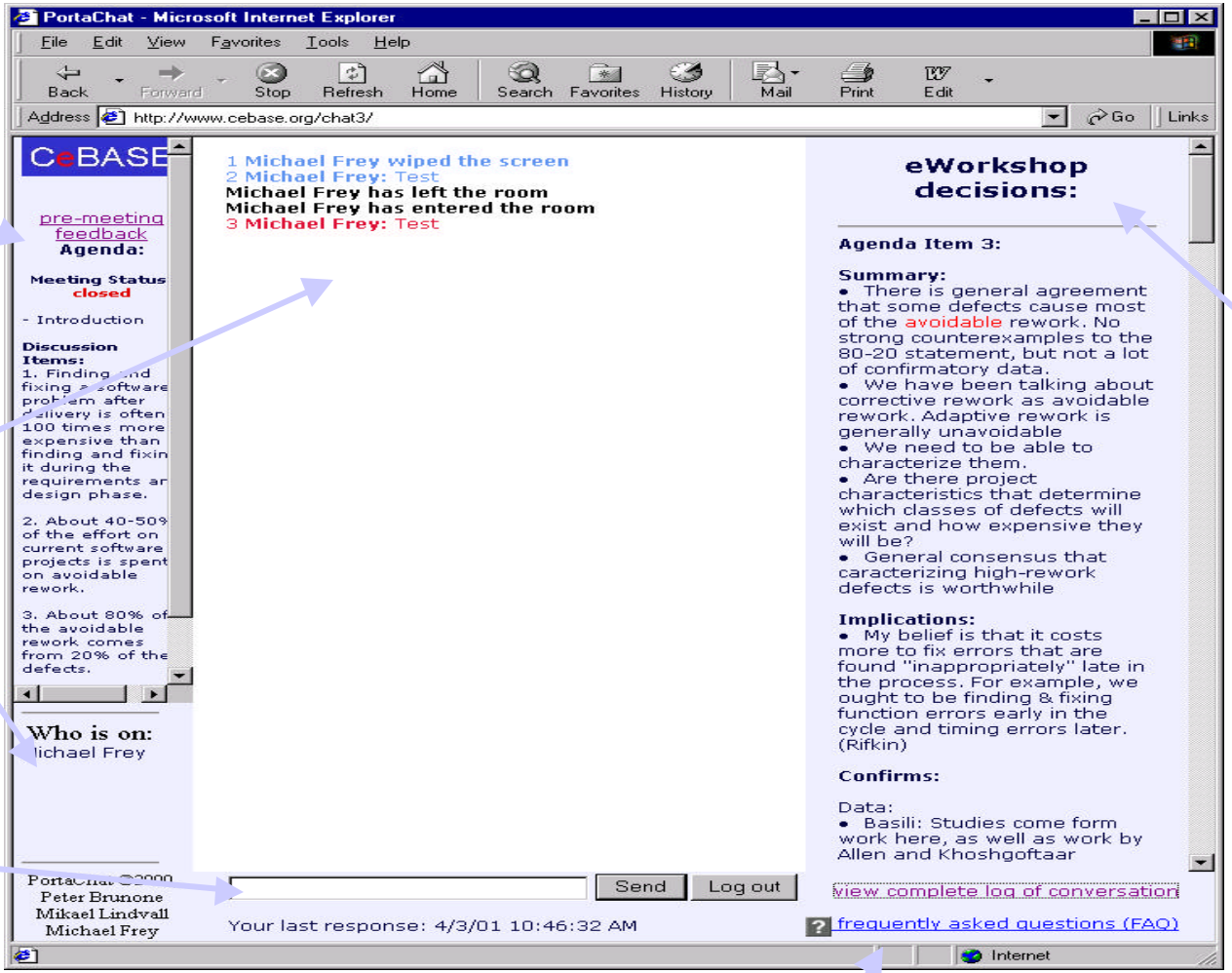
- The main tool is the web-based chat-application, adapted from some open source software

- 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



Agenda

Message board

Attendee list

Input Panel

Whiteboard

FAQ

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.

Defect Reduction eWorkshops

- History
  - Started from Top-10 list of Boehm & Basili in *IEEE Computer*, 2001
  - 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

CeBASE

Defect Reduction eWorkshops

- 3 online eWorkshops run in 2001/2002
- 1 traditional workshop at 2002 Software Metrics Symposium
- Over 80 participants (developers, consultants, academics), including:
  - **Ed Allen** (MSU), **Frank Anger** (NSF), **Vic Basili** (UMD), **Barry Boehm** (USC), **Carol Brothers** (Alcatel), **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), **Vern French** (xwave), **Megan Graham** (Guidant Corp.), **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)

CeBASE

Defect Reduction eWorkshops: Process

1. Choose Topic: Defect Reduction
   Seed the process: Top Ten List - Boehm/Basili, IEEE Computer

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. Examples of codes are:
   - VT – voting, RWK - Rework effort due to defects, DD - Finding/fixing Defects after Delivery

CeBASE

Defect Reduction eWorkshops: 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: *Collect the dust*

9. Analysis and Synthesis: *Create the pearls*
   Used VQI data mining tool

10. Dissemination of the results

CeBASE

Effect on cost and effort: Original seed

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

Effect on cost and effort: Collecting dust

- 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 ***non-severe*** 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

CeBASE

Effect on cost and effort: Current pearls

- **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!)*

CeBASE

Effect on cost and effort: Original seed

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

CeBASE

Effect on cost and effort: Collecting dust

- 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

CeBASE

Effect on cost and effort: Current pearls

- **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

CeBASE

Effectiveness of techniques for defect detection: Original seed

6: Peer reviews catch 60% of the defects.

Effectiveness of techniques for defect detection: Collecting dust

- 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**

CeBASE

Effectiveness of techniques for defect detection: Collecting dust

- **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)

CeBASE

Effectiveness of techniques for defect detection: Current pearls

- **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.*

CeBASE

Impact of defects on quality: Original seed

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

Impact of defects on quality: Collecting dust

- 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)

CeBASE

Impact of defects on quality: Collecting dust

- 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**

CeBASE

Impact of defects on quality: Current pearls

- **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.*

CeBASE

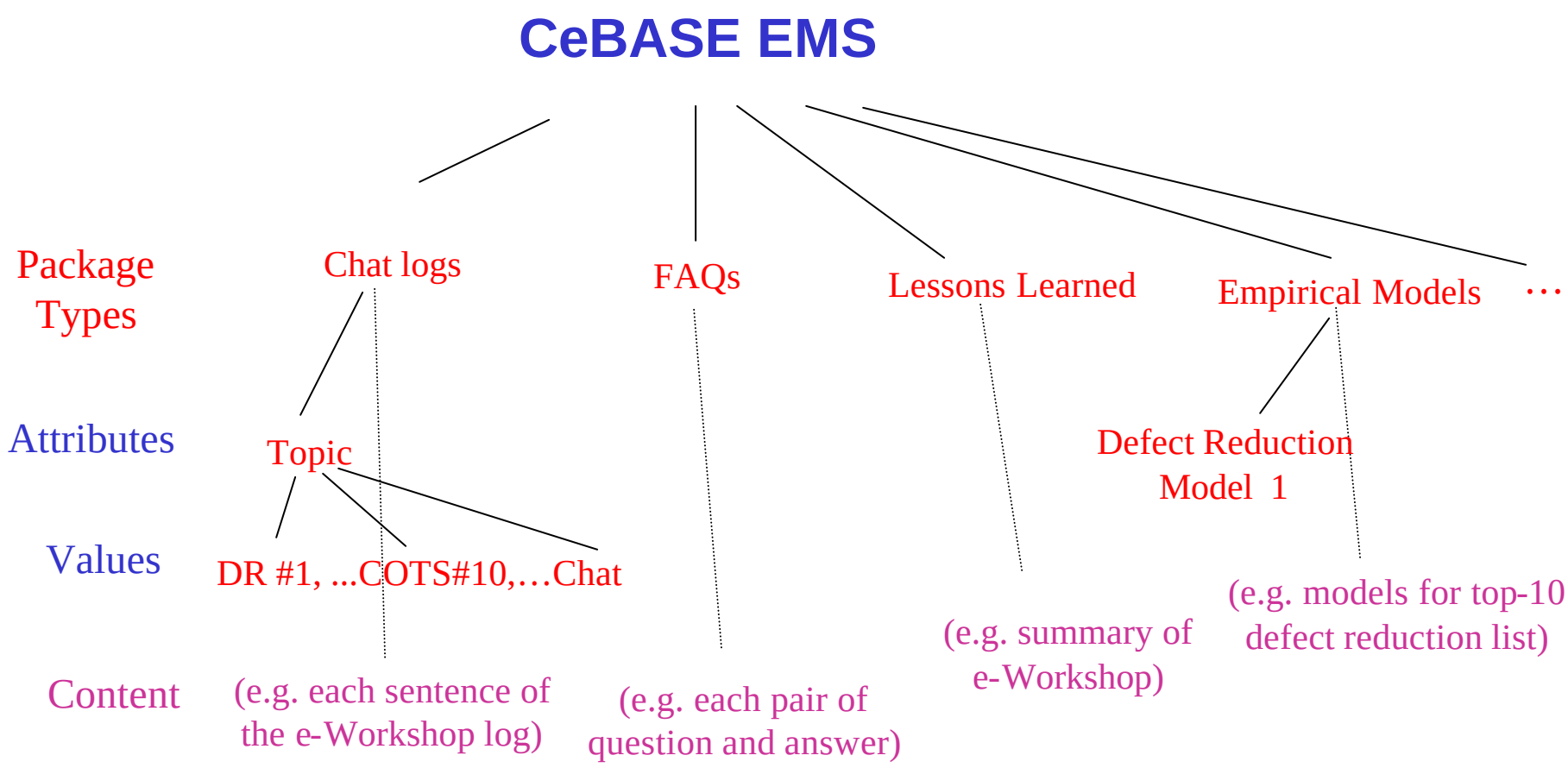
Conclusions on Defect Reduction

- 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?
  - …answering the above generates the pearls for the next round.

CeBASE

CeBASE Experience Management System (EMS) Logical/Physical Structure



eWorkshop Evaluation:
An Effective Way of Moving Dust to Pearls?

- 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

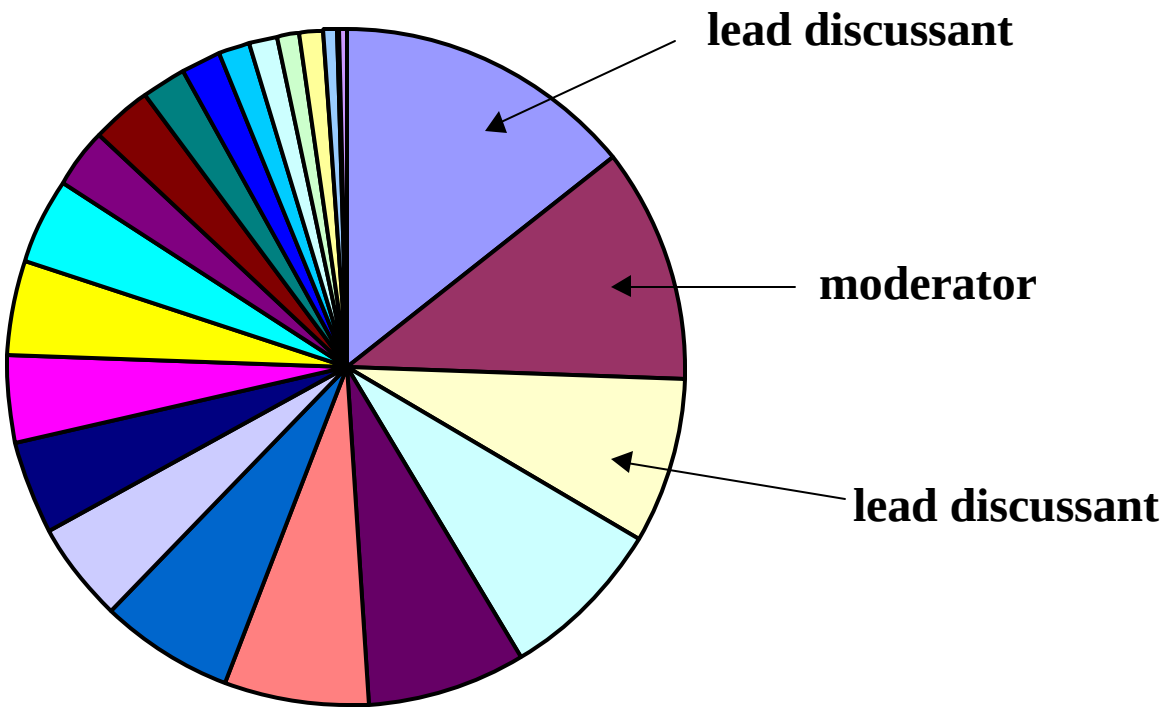
CeBASE

eWorkshop Evaluation (example of first eWorkshop)

- 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)

Conclusion

The results of these eWorkshops

- Refined the relevant items about Defect Reduction knowledge
- Provided additional references and data that seek to classify when specific defect reduction heuristics are applicable

We've run 8 eWorkshops covering

- Defect reduction (3)
- COTS (2)
- Agile methods (3)

The majority of participants have found the eWorkshops "a good way to discuss," "worthwhile and stimulating" and "a relatively easy way for a group to get a discussion going."

CeBASE

Wrap-Up

- For full description of previous discussions:
  - www.cebase.org/www/researchActivities/defectReduction/index.htm

CeBASE