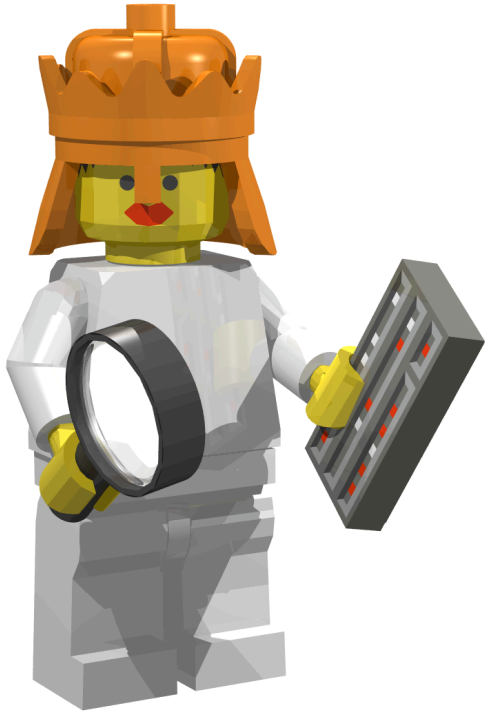
Benchmarking: The Way Forward for Software Evolution



Susan Elliott Sim
University of California, Irvine
ses@ics.uci.edu

Background

- Developed a theory of benchmarking based on own experience and historical research
- Successful benchmarks examined for commonalities:
  - TREC Ad Hoc Task
  - TPC-A™
  - SPEC CPU2000
  - Calgary Corpus and Canterbury Corpus
  - Penn treebank
  - *xfig* benchmark for program comprehension tools
  - C++ Extractor Test Suite (CppETS)

Susan Elliott Sim, Steve Easterbrook, and Richard C. Holt. Using Benchmarking to Advance Research: A Challenge to Software Engineering, Proceedings of the Twenty-fifth International Conference on Software Engineering, Portland, Oregon, pp. 74-83, 3-10 May, 2003.

Overview

- What is a benchmark?
- Why benchmark?
- What to benchmark?
- When to benchmark?
- How to benchmark?

- Talk will interleave theory with implications for software evolution

The Way Forward…

- Start with an exemplar.
  - Motivating Comparison + Task Sample
- Use the exemplar within the network to learn about each other's research
  - Comparison, discussions, relative strengths and weaknesses
  - Cross-fertilization, codification of knowledge
  - Hold meetings, workshops, symposia
- Add Performance Measures
- Use the exemplar (or benchmark) in publications
  - Common validation
- Promote use of exemplar (or benchmark) in broader research community

What is a benchmark?

- A benchmark is a standard test or set of tests used to compare alternatives. It consists of a motivating comparison, a task sample, and a set of performance measures.
  - Becomes a standard through acceptance by a community
  - Primarily concerned with technical benchmarks in computer science research communities.

Benchmark Components

1. Motivating Comparison
   - Comparison to be made
   - Motivation for research area and benchmark
2. Task Sample
   - Representative sample of problems from a problem domain
   - Most controversial part of benchmark design
3. Performance Measures
   - Performance = fitness for purpose; a relationship between technology and task
   - Can be qualitative or quantitative, measured by human, machine, or both

What is not a benchmark?

- Not an evaluation designed by an individual or single laboratory
  - Potential as starting point, but not a standard
- Not a baseline or fixed point
  - Needed for comparative evaluation, but not sufficient
- Not a case study that is used repeatedly
  - Possibly a proto-benchmark or exemplar
- Not an experiment (nor trial and error)
  - Usually no hypothesis testing, key factors not controlled

Benchmarking as an Empirical Method

Characteristics from Experiments	Characteristics from Case Studies
Features ? Use of control factors ? Replication ? Direct comparison of results	**Features** ? Little control over the evaluation setting, (e.g. choice of technology and user subjects) ? No tests of statistical significance ? Some open-ended questions possible
Advantages ? Direct comparison of results	**Advantages** ? Method is flexible and robust
Disadvantages ? Not suitable for building explanatory theories	**Disadvantages** ? Limited control reduces generalizability of results

Overview

- What is a benchmark?
- Why benchmark?
- What to benchmark?
- When to benchmark?
- How to benchmark?

Impact of Benchmarking

"…benchmarks cause an area to blossom suddenly because they make it easy to identify promising approaches and to discard poor ones." -Walter Tichy

"Using common databases, competing models are evaluated within operational systems. The successful ideas then seem to appear magically in other systems within a few months, leading to a validation or refutation of specific mechanisms for modelling speech. " -Raj Reddy

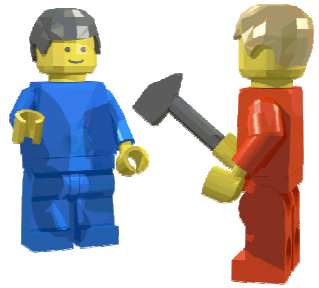
Walter F. Tichy, "Should Computer Scientists Experiment More?," *IEEE Computer*, May, pp. 32-40, 1998.

Raj Reddy, "To Dream The Possible Dream - Turing Award Lecture," *Communications of the ACM*, vol. 39, no. 5, pp. 105-112, 1996.

Benefits of Benchmarking

- Stronger consensus on the community's research goals
- Greater collaboration between laboratories
- More rigorous validation of research results
- Rapid dissemination of promising approaches
- Faster technical progress

- Benefits derive from process, rather than end product



Dangers of Benchmarking

- Subversion and competitiveness
  - “Benchmarketing” wars
- Costs to develop and maintain
- Committing too early
- Overfitting
  - General performance is sacrificed for improved performance on benchmark
- Non-independent probabilistic results
- Closing off other research directions (temporarily)

Why is benchmarking effective?

- Explanation is based in philosophy of science.
- Conventional view: scientific progress is linear.
- Thomas Kuhn introduced the idea that science moves from paradigm to paradigm.
  - During normal science, progress is linear.
  - Canonical paradigm shift is change from Newtonian mechanics to quantum mechanics.
- A scientific paradigm consists of all the information that is needed to function in a discipline. It includes technical facts and implicit rules of conduct.
- Paradigm is created by community consensus.

Thomas S. Kuhn, *The Structure of Scientific Revolutions, Third Edition*. Chicago: The University of Chicago Press, 1996.

Theory of Benchmarking

- Process of benchmarking mirrors process of scientific progress.

  Progress = technical facts + community consensus

- A benchmark operationalizes a paradigm.
  - Takes an abstract concept and turns it into a concrete guide for action.

Sensemaking vs. Know-how

- Beneficial to both main activities of RELEASE
  - Understanding evolution as a noun– what, why
  - Understanding evolution as a verb– how

- Focusing attention on a technical evaluation brings about a new understanding of the underlying phenomenon
  - Assumptions
  - Problem frames and world views

Overview

- What is a benchmark?
- Why benchmark?
- What to benchmark?
- When to benchmark?
- How to benchmark?

What to benchmark?

- Benchmarks are best used to evaluate technology
  - When a result to be use *for* something
- Where engineering issues dominate
  - Example: algorithms vs. implementations

- For RELEASE, this is the *how* of software evolution

Benchmark Components

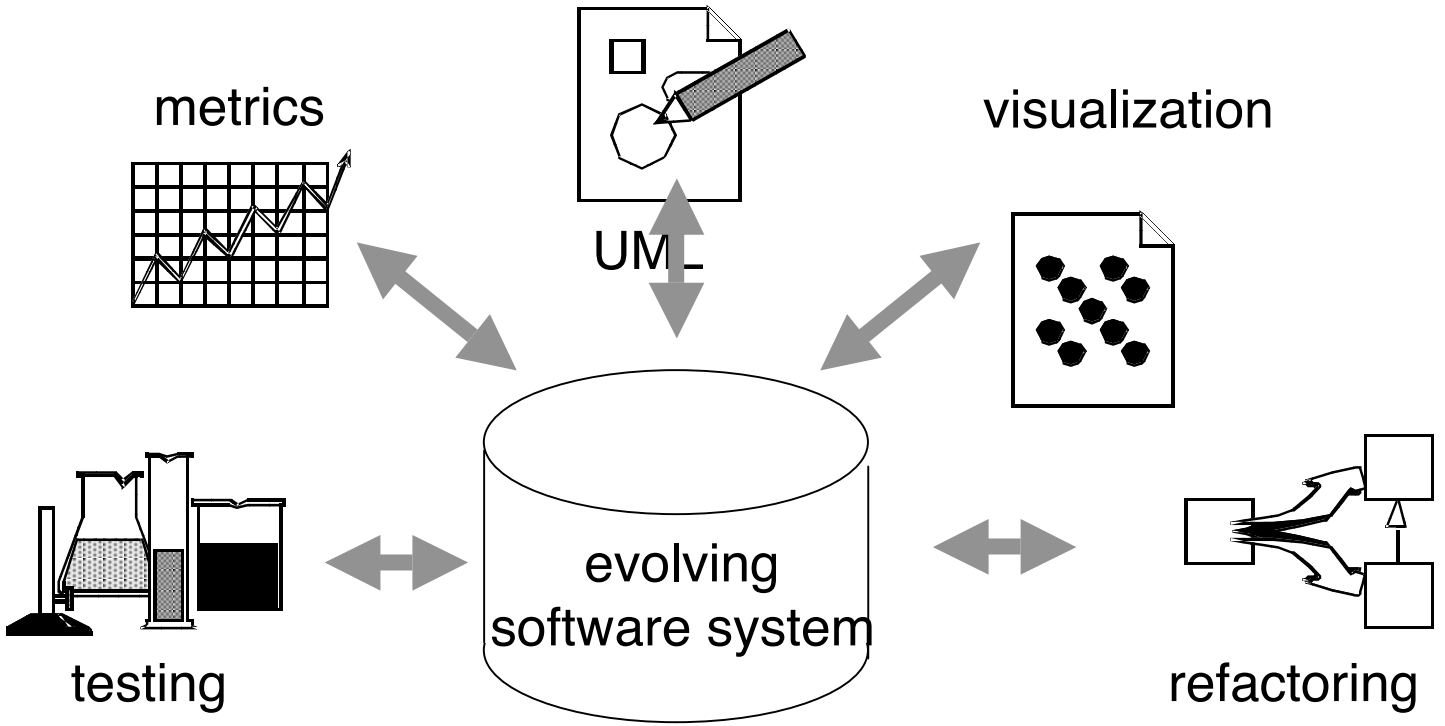
- The design of a benchmark is closely related to the scientific paradigm for an area.
  - Deciding what to include and exclude is a statement of values.
  - Discussions tend to be emotional.
- Benchmarks can fulfill many purposes, often simultaneously.
  - Advance a single research effort
  - Promoting research comparison and understanding
  - Setting a baseline for research
  - Providing evidence for technology transfer

Motivating Comparison

- Examples:
  - To assess information retrieval system for an experienced searcher on ad hoc searches. (TREC)
  - To rate DBMSs on cost effectiveness for a class of update-intensive environments. (TPC-A)
  - To measure the performance of various system configurations on realistic workloads. (SPEC)

- Can a context for specified for the software evolution benchmark?

Software Evolution Techniques



Which techniques do complement each other ?

Take from Tom Mens, RELEASE meeting, 24 October 2002, Antwerp

Task Sample

- Representative of domain problems encountered by end user
  - Focus on the problems, not the tools to be compared
    - Tool view: Retrospective, Curative, Predictive
    - User view: Due diligence, bid for outsourcing
  - Key or typical problems act as surrogates for a class
- Possible to include a suite of programs, but need to keep the benchmark accessible
  - Does not take too much time and effort to use
  - Automation can mitigate these costs.

Performance Measures

- Do accepted measures already exist?
- Are there right answers (ground truth)?
- Does close count? How do you score?
- Initial performance measures can be “rough and ready”
  - Human judgments
  - Approximations
  - Qualitative

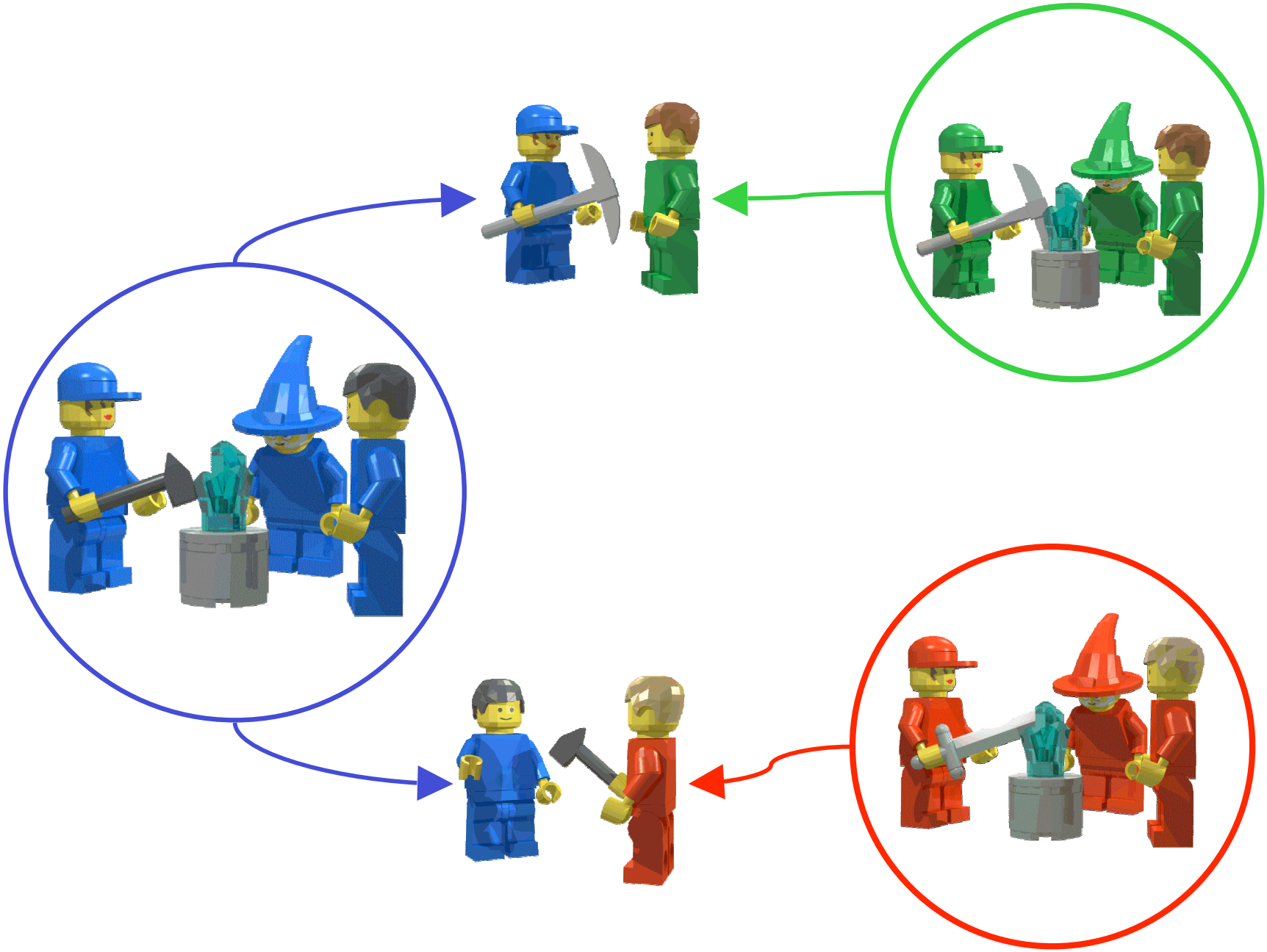
- Process of measuring often defines what is.
  - Should first decide what is and then figure out how to measure.

Overview

- What is a benchmark?
- Why benchmark?
- What to benchmark?
- When to benchmark?
- How to benchmark?

When to benchmark?

- Process model for benchmarking
- Knowledge and consensus move in lock-step
- Pre-requisites
  - Indicators of readiness
- Features



Prerequisites for Benchmarking

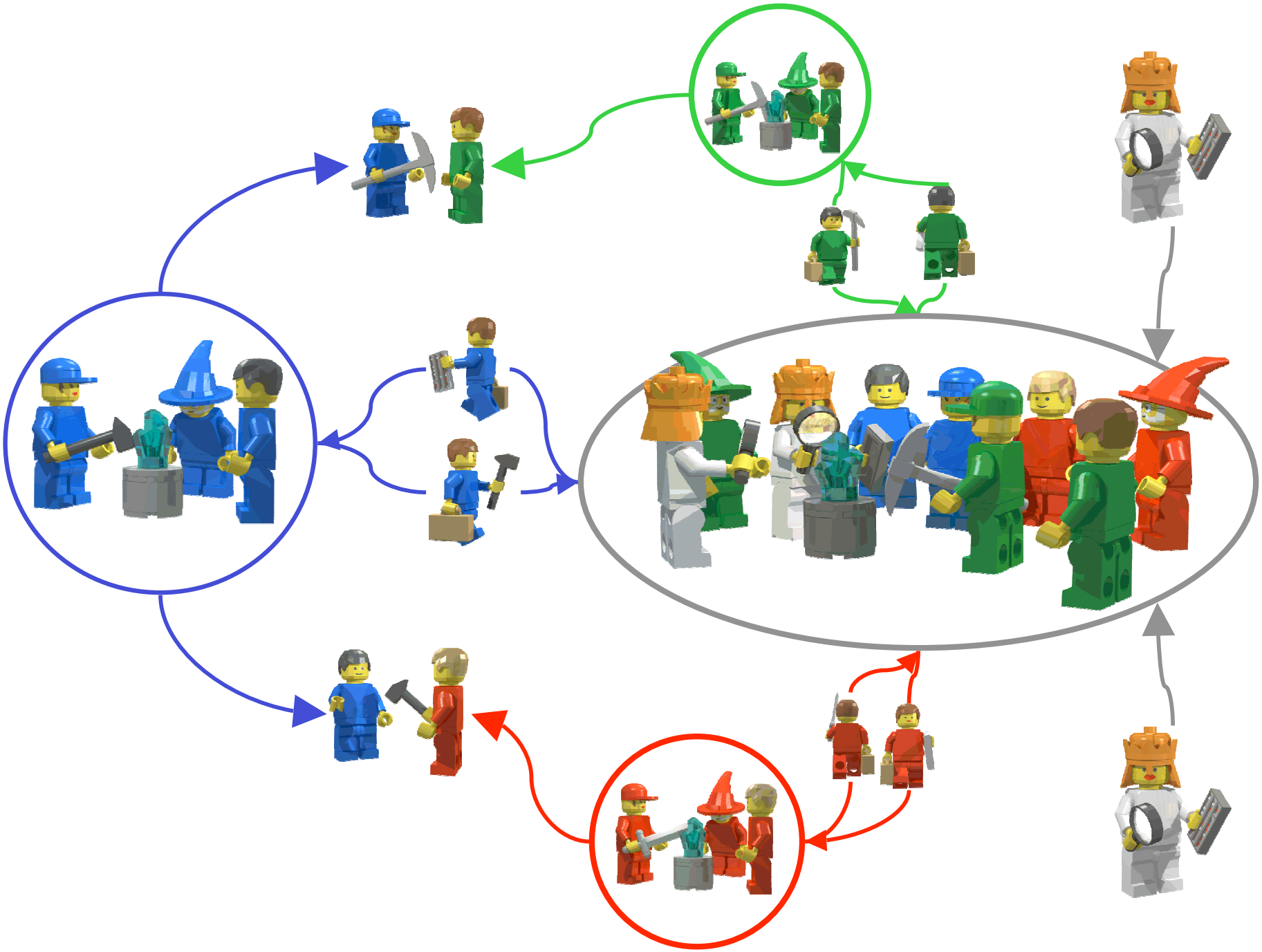
- Minimum Level of Maturity
  - Proliferation of approaches and implementations
  - Recognized separate research area
  - Participants self-identify as community members
- Ethos of Collaboration
  - Research networks
  - Seminars, workshops, meetings
  - Standards for data, files, reports, papers
- Tradition of Comparison
  - Accepted research strategies, especially validation
  - Evidence in the literature
  - Use of common examples

Overview

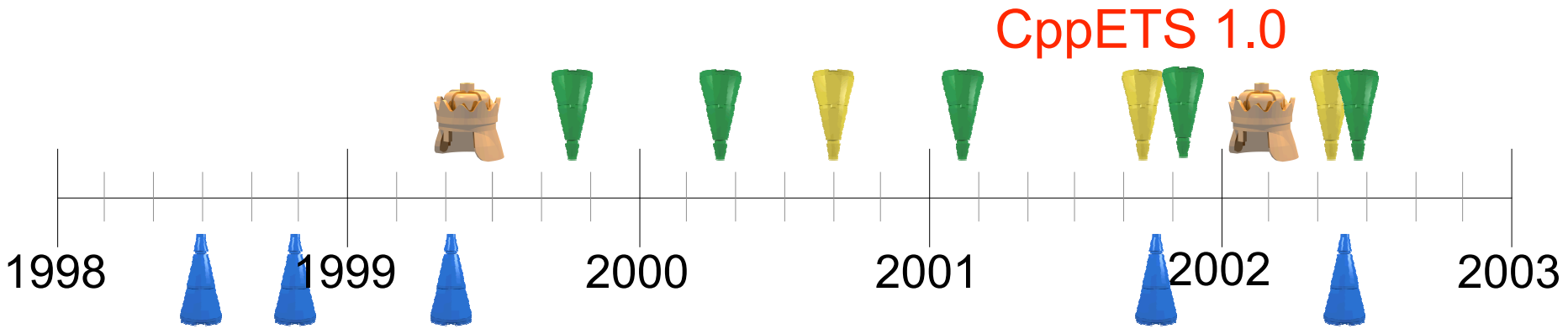
- What is a benchmark?
- Why benchmark?
- What to benchmark?
- When to benchmark?
- How to benchmark?

How to benchmark?

- Knowledge and consensus move in lock-step
- Features of a successful benchmarking process
  - Led by a small number of champions
  - Supported by laboratory work
  - Many opportunities for community participation and feedback



Emergence of CppETS



Paper

Meeting

Laboratory work

Implications for Software Evolution

- Steps taken so far fits with the process model
  - Papers, workshops, champions
- Many years (and iterations) are needed to build a widely-accepted benchmark
  - Time is needed to build consensus
- Many elements already in place
  - Champions
  - A research network that meets regularly
  - Funding for laboratory work

The Way Forward…

- Start with an exemplar.
  - Motivating Comparison + Task Sample
- Use the exemplar within the network to learn about each other's research
  - Comparison, discussions, relative strengths and weaknesses
  - Cross-fertilization, codification of knowledge
  - Hold meetings, workshops, symposia
- Add Performance Measures
- Use the exemplar (or benchmark) in publications
  - Common validation
- Promote use of exemplar (or benchmark) in broader research community

More Information

- Paper from ICSE 2003
  - http://www.cs.utoronto.ca/~simsuz/papers/icse03-challenge.pdf
- *xfig* structured demonstration
  - http://www.csr.uvic.ca/~mstorey/cascon99/
- CppETS 1.0
  - http://www.cs.utoronto.ca/~simsuz/cascon2001
- CppETS 1.1
  - http://cedar.csc.uvic.ca/kienle/view/IWPC2002/WebHome

Virtual LEGO Construction

- All software is free, thanks to the spirit of James Jessiman.
  - http://www.ldraw.org
- LD Design Pad Minifig Plug-In
  - Uses LDraw parts library and DAT file format
  - http://www.pobursky.com/LDrawBody3.htm
- MLCad
  - Creates models and scenes
  - http://www.lm-software.com/mlcad
- L3P
  - Converts DAT to POV format
  - http://home16.inet.tele.dk/hassing/index.html
- POV-Ray
  - Renders the model into a drawing
  - http://www.povray.org/