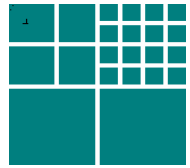


Is there a future for Empirical Software Engineering?

Victor R. Basili
Department of Computer Science
University of Maryland
College Park, Maryland

ISESE 2006

Approach

- Claes asked me to take a 40 year perspective (-20, now, +20)
- This talk provides a ***personal*** perspective on the evolution of empirical software engineering
- I will try to map change across several variables
  - Kinds of studies
  - Set of methods
  - Publication/review issues
  - Community of researchers
  - Replication / Meta analysis
  - Attention to context variables

Outline

- Phase I
  - Early days: Running studies
- Phase II
  - Tying studies together within an environment and domain
- Phase III
  - Expanding out across environments, limiting techniques
- Phase IV
  - Focusing on a domain

Phase I: 1974 – 1985

Early days: Running isolated studies for a particular purpose

Independently using case studies and controlled experiments

Phase I
Iterative Enhancement Case Study

- **Problem**: Can we quantitatively measure the effect of the application of a method on the product?
  - Method produced incremental versions of the product, each with more functionality
- **Empirical Approach**: ***Case study***
- **Issues**: quantitative, observations over time, measuring the product, comparing a product with itself (baseline issue), using feedback

- V. Basili and **A. Turner**, "Iterative Enhancement: A Practical Technique for Software Development," IEEE Transactions on Software Engineering, vol. 1(4), December **1975**

Learned

- **single study, single project, single environment**
  - Like what Belady and Lehman did on OS360
  - we were authors of the method and builders of the system
  - thought we should be able to empirically demonstrate what people said about the product and whatr we thought hsould have happened

- **Learned**:
  - about developing metrics,
  - the measurement process,
  - running an empirical study,
  - evaluating needs a basis for comparison (baselines)

Phase I
Methodology Evaluation Controlled Experiment

- **Problem**: Can we measure and differentiate the effects of different processes on products?
- **Empirical Approach**: A replicated study (controlled experiment) with three treatments
- **Issues**: controlled experiment, unobtrusive observations, multiple treatments, teams as a unit of measure

- V. Basili and **R. Reiter, Jr**., "A Controlled Experiment Quantitatively Comparing Software Development Approaches" IEEE Transactions on Software Engineering, vol. 7(3): 299-320 (IEEE Computer Society Outstanding Paper Award), May **1981**.(early version in **1979**)

Learned

- **Single study, set of methods, single environment**
  - ~ Jerry Weinberg did but with teams and on bigger programs
  - we were neither method authors nor system builders
  - trying to study the proposed methods of the time

- It was difficult to advice for performing a controlled experiment in the software engineering domain
  - Chose Educational Research as the model to follow
  - Campbell and Stanley became our bible for many years.

- **Learned**:
  - About controlled experiments,
  - using nonparametric statistics,
  - developing proxies for measures,
  - automated measures, ..

Phase I

The Early Software Engineering Laboratory

- **Problem**: Can we increase the quality of the ground support software systems NASA/GSFC produces by improved processes?

- **Empirical Approach**:
  - Build baselines of various project variables (defects, effort, project metrics) and identify where methods might make a difference.
  - Choose methods from what is currently available.

- **Issues**: Collecting data from live projects, feedback on data collection and measures, large amounts of data to collect, store, analyze

- V. Basili and **M. Zelkowitz**, “Analyzing Medium Scale Software Development,” Proceedings of the Third International Conference on Software Engineering, May **1978**.

Learned

- Multiple projects, Set of methods, Single environment and domain

- **Learned:**
  - Importance of the understanding the environment (context variables?)
  - Need to build our own models to understand and characterize
  - Need to model the environment, projects, processes, products, etc.
  - Data collection has to be goal driven and well defined

➔ Many studies on effort estimation, defects classification, identification of common projects

➔ **Goal/Question/Metric Approach**

V. Basili and **D. Weiss**, "Evaluation of a Software Requirements Document by the analysis of Change Data," Proceedings of the Fifth International Conference on Software Engineering, pp. 314-323, March **1981**.

New Thinking after Phase I

- Software development follows an **experimental paradigm**,
  - Design of experiments is an important part of improvement
  - Evaluation and feedback are necessary for learning

- Need to **experiment** with technologies to reduce risk, tailor to the environment, make improvements

➔ **Quality Improvement Paradigm**

V. Basili, “Quantitative Evaluation of Software Methodology,” Keynote Address, Proceedings of the First Pan Pacific Computer Conference, vol. 1, pp. 379-398, September **1985**.

Phase I: State of the Variables

- **Running Studies**
  - Mostly characterizing in a single environment, single domain
- **Publication/ Reviews**: many project studies, mixed reviews
- **Community of Researchers**: almost empty: model builders and other individuals
- **Set of methods** for empirical study: mostly quantitative, nonparametric, nominal and ordinal measurement
- **Context variables**: taken as a given, not measured
- **Replication / Meta Analysis**:?

Phase II 1986 - 1999

Tying studies together in one environment, one domain

Controlled experiments, case studies, quasi-experiments, qualitative analysis tied together

Defining an Experimentation Framework

- For the 10th anniversary of TSE (1986), we defined a framework for experimentation in software engineering to
  - Help structure the process and provide a classification scheme for understanding and evaluating experimental studies
  - Classify experiments from the literature according to the framework
  - Identified problem areas and lessons learned

- State of the Field
  - References: 116, (not only studies)
  - Mostly
    - Empirical studies of programmers in the small doing controlled experiments
    - Data collection on projects in the large

The Experimentation Framework

The set of categories

- Definition
  - Motivation, Object, Purpose, Perspective, Domain, Scope
- Planning
  - Design, Criteria, Measurement
- Operation
  - Preparation, Execution, Analysis
- Interpretation
  - Context, Extrapolation, Impact

- V. Basili, **R. Selby**, and D. **Hutchens**, "Experimentation in Software Engineering," *IEEE Transactions on Software Engineering* vol. 12(7): 733-743, July **1986**

The Experimentation Framework

Experiment Scopes

		#Projects	
		One	More than one
# of Teams per Project	One	**Single Project**	**Multi-Project Variation**
	More than one	**Replicated Project**	**Blocked Subject-Project**

Phase II
Mixing the Software Engineering Laboratory with Classroom Studies

- **Problem**: Can we use empirical studies to identify and tailor potential quality improving processes,
  - reducing the risk to projects,
  - to increase the quality of the ground support software systems at NASA/GSFC?

- **Empirical Approach**: Use the university to test out techniques, transfer them to NASA and evolve them based upon observation

- **Issues**: Collecting data from live projects, costs, minimizing aggravation of developers, feedback on data collection and measures

Experimental Learning Mechanisms

Series of Studies

Code reading, functional and structural testing
Unit test size programs seeded with faults
Blocked subject-project: Fractional factorial design
Three replications: 42 UM (2), 32 NASA/CSC

# of Teams per Project			
	One		
	More than one		**Reading vs. Testing**

Lessons Learned

- Intuition is not always consistent with reality
- Motivation about the technique (treatment variable) plays a key role
- Methods and techniques are different
  - reading vs. testing, inspections vs. test plan
- Different techniques, methods may be more effective for different types of defects (problems, environments, product characteristics)

Experimental Learning Mechanisms

Series of Studies

Scaled up to teams and larger projects
Compared 15 teams using Cleanroom and not using it
When reading is motivated it is very effective

# of Teams per Project	One	
	More than one	**2. Cleanroom at Maryland**
		1. Reading vs. Testing

Experimental Learning Mechanisms

Series of Studies

Training and tailoring of Cleanroom for the SEL
Integrated into the existing process
Very effective on live project (40K SLOC) at NASA

		ne	More than one
# of Teams	One	**3. Cleanroom (SEL Project 1)**	
per Project	More than one	**2. Cleanroom at Maryland**	**1. Reading vs. Testing**

Experimental Learning Mechanisms

Effective over a series of projects
Some modification for contracted out projects
Recognized need to reading at higher level,
e.g. Requirements reading

		One	More than one
# of Teams per Project	One	**3. Cleanroom (SEL Project 1)**	**4. Cleanroom (SEL Projects, 2,3,4,...)**
	More than one	**2. Cleanroom at Maryland**	**1. Reading vs. Testing**

Experimental Learning Mechanisms

Series of Studies

Developed perspective based reading techniques
Experimented with requirements reading
Effective on controlled experiments with NASA developers

# of Teams	One	**3. Cleanroom (SEL Project 1)**	**…room …rojects, 2,3,4,...)**
per Project	More than one	**2. Cleanroom at Maryland**	**1. Reading vs. Testing** **5. Scenario reading vs. ...**

Lessons Learned

- Multiple studies, multiple experimental designs, two environments NASA and small projects, multiple project types and sizes

- Learned:
  - Can reduce risk by running smaller experiments off-line
  - Can build confidence in a theory based upon multiple treatments
  - Studies need to be focused on the opportunities
  - There is a measurable relationship between process and product
  - Techniques can be developed based upon goals

New Thinking after Phase II

- **Experience** needs to be evaluated, tailored, and **packaged** for reuse
  - Software processes must be put in place to support the reuse of experience
  - Packaged experiences need to be integrated

➔ Evolved **QIP** (packaging) and **GQM** (templates and models)

V. Basili and H.D. **Rombach**, "The TAME Project: Towards Improvement-Oriented Software Environments," *IEEE Transactions on Software Engineering*, vol. 14(6), June **1988**.

➔ Formalized the organization via the **Experience Factory Organization**

V. Basili, "Software Development: A Paradigm for the Future," Proceedings of COMPSAC '89, pp. 471-485, September **1989**.

State of the Variables: Phase II

- **Studies** to
  - Package knowledge (build models) to improve software quality based upon experience in an environment
- **Publication/ Reviews**: project-based stuff easier, experiments not so easy to publish in conferences
- **Community of Researchers**: ISERN (1993), EMSE (1996)
- **Set of methods** for empirical study: mostly quantitative, nonparametric, some qualitative, nominal and ordinal measurement,
- **Context variables**: taken as a given but recognized a important
- **Replication/Meta analysis**: can build a sequence of studies that vary the context, threats to validity; building knowledge across studies about a particular technology

Phase III 2000 - 2004

Expanding out across domains, environments

Focusing on building knowledge for a limited number of techniques in different environments and domains

i.e., studying the effect of context on techniques

CeBASE
Center for Empirically Based Software Engineering

Problem: Can we build a body of knowledge about specific techniques (defect reduction, COTS based development, agile development) supported by empirical evidence?

CeBASE Project Goal: Enable a **decision framework and experience base** that forms a basis and infrastructure needed to evaluate and choose among software development technologies

CeBASE Research Goal: Create and evolve an **empirical research engine** for building the research methods that can provide the empirical evidence of what works and when

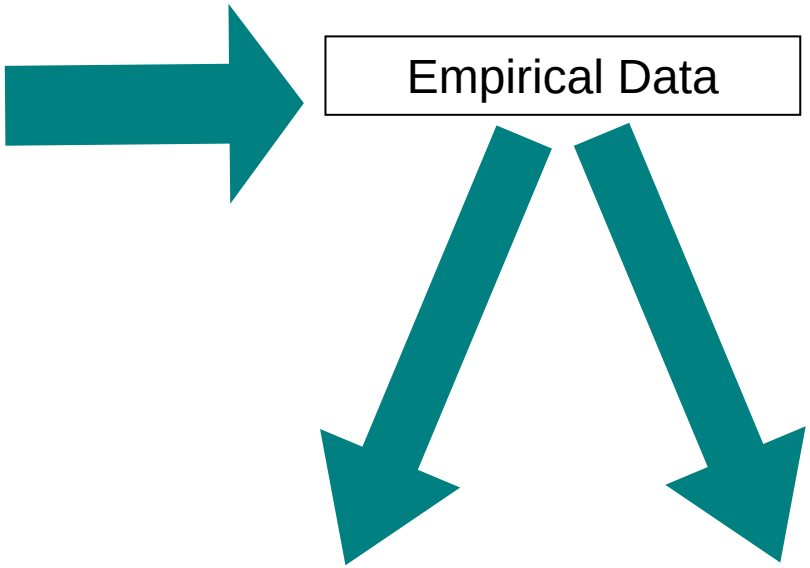
Partners: Victor Basili (UMD), Barry Boehm (USC)

CeBASE NSF Center for Empirically Based Software Engineering

University of Maryland • Fraunhofer Center-MD • University of Southern California • University of Nebraska • Mississippi State University

CeBASE Approach

Observation and Evaluation Studies of Development Technologies and Techniques



Predictive Models

(Quantitative Guidance)

General Heuristics

(Qualitative Guidance)

E.g. COCOTS excerpt:

Cost of COTS tailoring = f(# parameters initialized, complexity of script writing, security/access requirements, ... **)**

E.g. Defect Reduction Heuristic:

For faults of **omission** and **incorrect specification, peer reviews** are more effective than functional testing.

CeBASE Basic Research Activities

Define and improve methods to

- Formulate evolving hypotheses regarding software development **decisions**
- Collect empirical **data** and experiences
- Record **influencing variables (context)**
- Build **models** (Lessons learned, heuristics/patterns, decision support frameworks, quantitative models and tools)
- Integrate models into a **framework**
- Testing **hypotheses** by application
- **Package** what has been learned so far so it can be used and evolved

Lessons Learned

- There is a great deal more research to do before we can solve this problem
  - Building the research engine
  - Building the decision support system with partial knowledge
  - Integrating the process
- Context can change everything and is hard to identify
- This is **big science**, involving many researchers, many environments, many domains
- Won't evolve the knowledge base without collaboration
- Need to shrink the focus

Applied Research

NASA High Dependability Computing Program

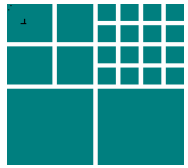
Problem: How do you elicit the software dependability needs of various stakeholders and what technologies should be applied to achieve that level of dependability?

Project Goal: Increase the ability of NASA to engineer highly dependable software systems via the development of new technologies in systems like Mars Science Laboratory

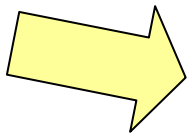
Research Goal: Quantitatively define dependability, develop high dependability technologies and assess their effectiveness under varying conditions and transfer them into practice

Partners: NASA, CMU, MIT, UMD, USC, U. Washington, Fraunhofer-MD

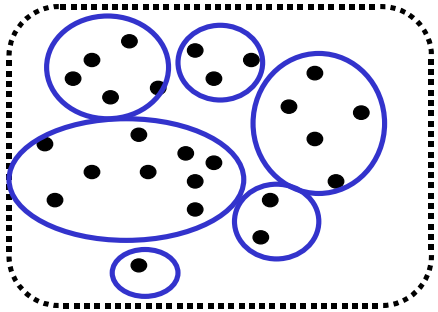
What are the top level research problems?



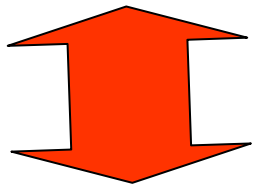
System Users



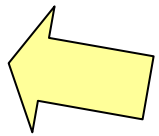
Failures Space



Research Problem 1
Can the quality needs be understood and modeled?

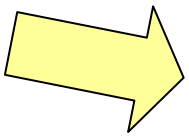


Research Problem 3
What set of technologies should be applied to achieve the desired quality? (Decision Support)

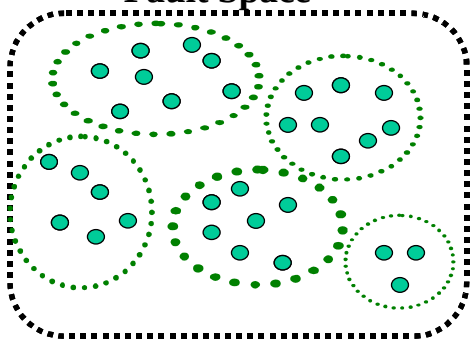


System Developers

Technology Developers



Fault Space



Research Problem 2
What does a technology do?
Can it be empirically demonstrated?

Lessons Learned

- Collecting experience across environments, domains, technologies, is very difficult

- Studying and maturing techniques require testbeds
  - Can be built for classes of techniques
  - Need to be maintained

- Technologists need to be more specific about what their technologies do and do not do

➔ Evolved empirical evidence about various techniques

- **B. Boehm** and V. Basili, "Software Defect Reduction Top 10 List," *IEEE Computer*, vol. 34(1): 135-137, January **2001**.
- J82. V. Basili and **B. Boehm**, "COTS-Based Systems Top 10 List," *IEEE Computer*, vol. 34(5): 91-93, May **2001**.

New Thinking after Phase III

- The problems are
  - eliciting and quantifying (non-functional) requirements
  - Specifying the effects of technologies
  - Empirically identifying the effects, limits and bounds of techniques
  - Integrating the processes to satisfy the developers need to satisfy the users' needs

- We need to concentrate on building a body of knowledge based upon empirical evidence

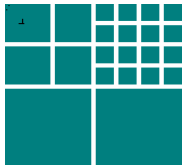
State of the Variables: Phase III

- **Studies** to evaluate techniques in multiple contexts and define the relationship between user needs and what's available
- **Publication/Reviews**: domain journals and conferences, SE journals
- **Community of Researchers**: ISERN, EMSE, ISESE → ISESEM
- **Set of methods** a rich palate of tools: full mix of qualitative and quantitative methods, controlled and quasi-experiments, case studies, surveys, folklore gathering, structured interviews and reviews, …
- **Context variables**: being studied and characterized
- **Replication**: numerous repetitions of a few experiments (PBR, reading vs. testing)
- **Meta Analysis**: building knowledge across studies

Phase IV now and the future

Focusing on a domain to build a body of knowledge

Folklore gathering, interview, case studies, controlled experiments, experience bases, …



Building an Experience Base
DARPA High Productivity Computing Systems

Problem: How do you improve the time and cost of developing high end computing (HEC) codes?

Project Goal: Improve the buyers ability to select the high end computer for the problems to be solved based upon productivity, where productivity means

Time to Solution = Development Time + Execution Time

Research Goal: Develop theories, hypotheses, and guidelines that allow us to characterize, evaluate, predict and improve how an HPC environment (hardware, software, human) affects the development of high end computing codes.

Partners: MIT Lincoln Labs, MIT, UCSD, UCSB, UMD, USC, FC-MD

Development Time Goals

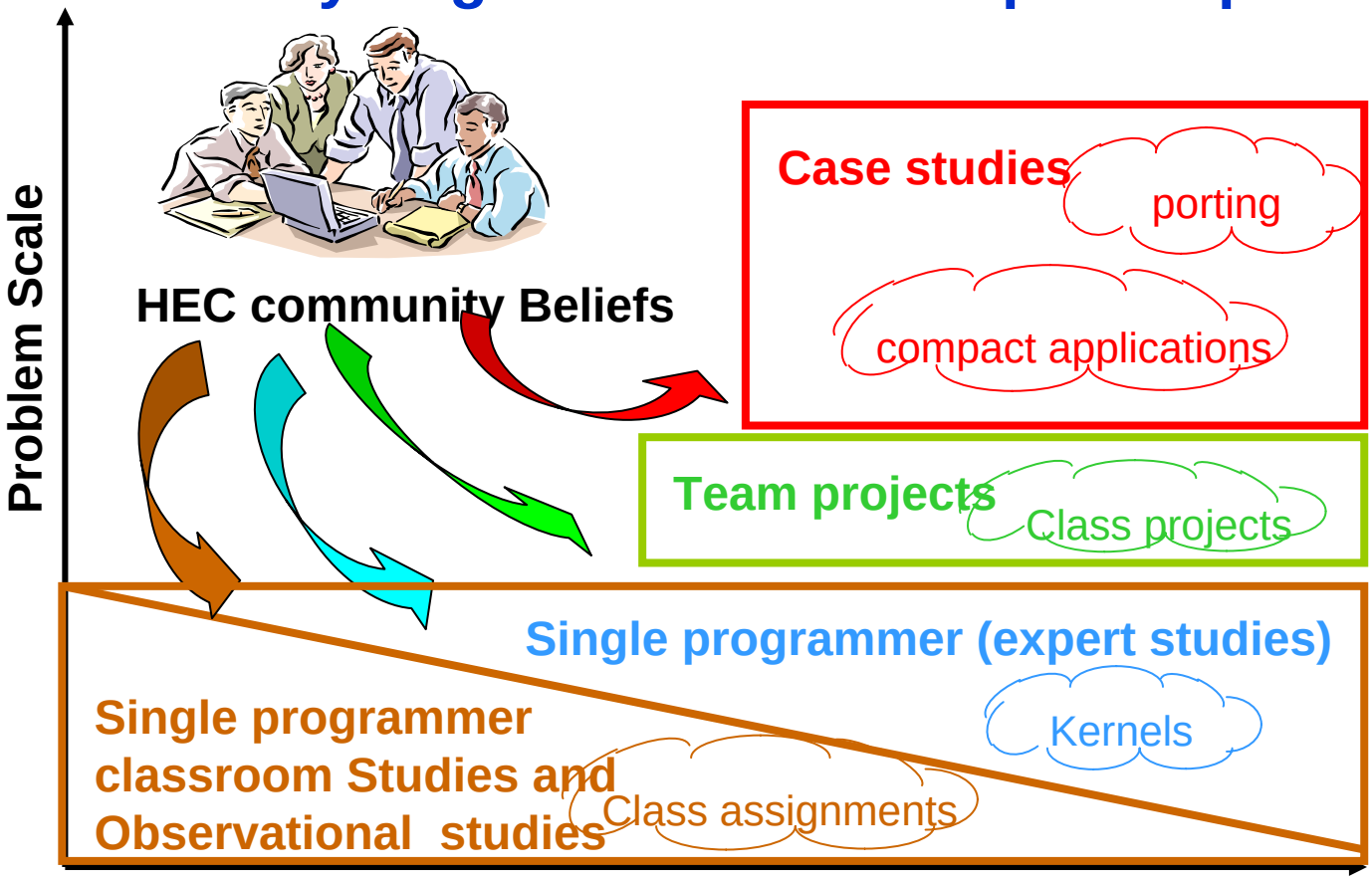
- Obtain a more complete view of the total time to solution
- Better understand software development for HPC
- Provide empirical evidence for assumptions made by the HPC community about development time issues
- Develop criteria for HPC systems productivity evaluation
- Provide better guidance for planning and decision-making in HPC code development

By evolving a series of studies with novices and professionals

- Controlled experiments (grad students)
- Observational studies (professionals, grad students)
- Case studies (class projects, HPC projects in academia)
- Surveys, interviews (HPC experts)

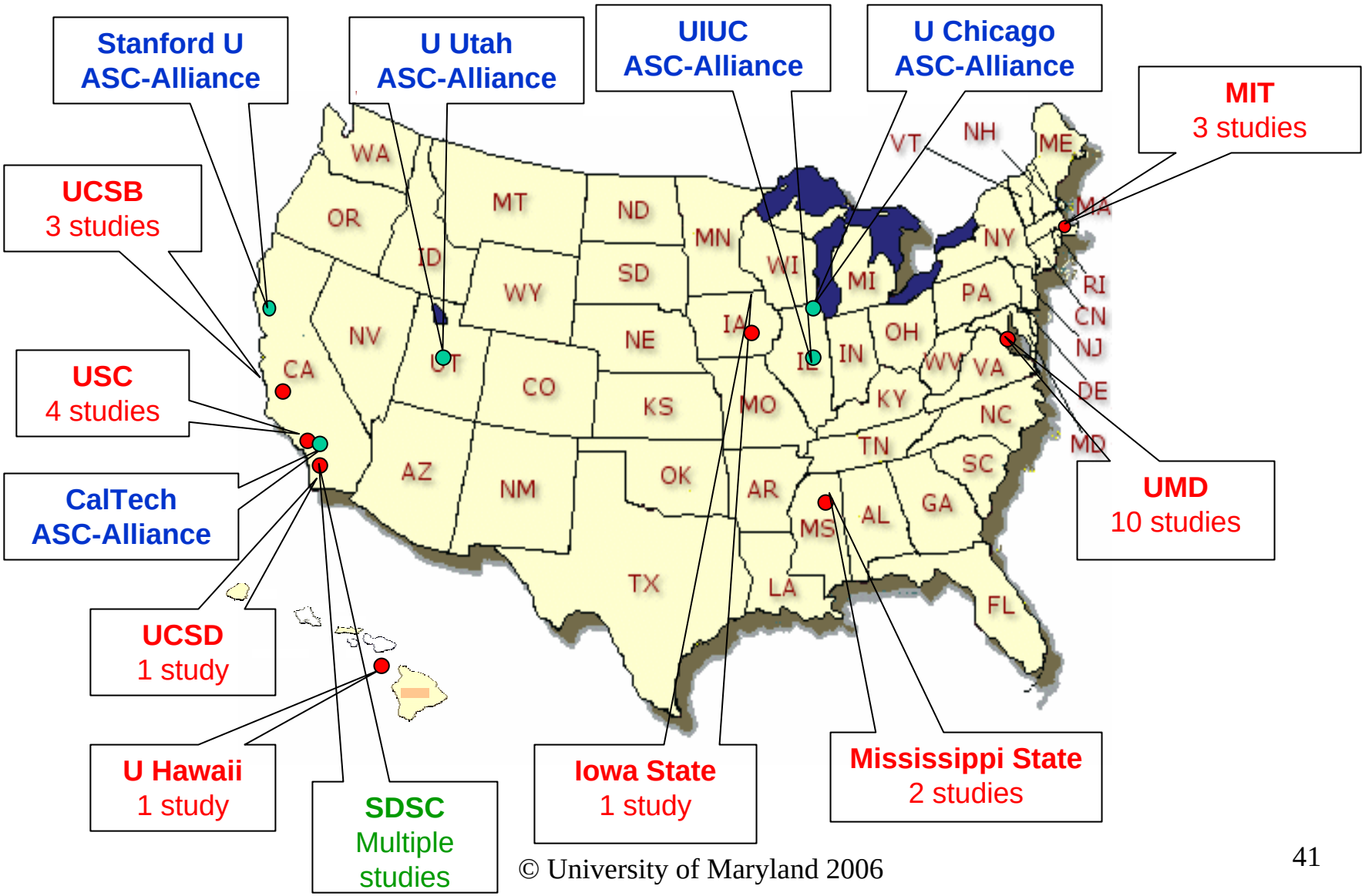
Overall Research Plan

HEC community provides questions to study that lead to successively larger and more complex experiments



Evolving measurements, Models, Hypotheses

Study Locations



Elements of the Knowledge Base

- **Testbeds**:
  - **Classroom assignments** (Array Compaction, the Game of Life, Parallel Sorting, LU Decomposition, …
  - **Compact Applications** (Combinations of Kernels, e.g., Embarrassingly Parallel, Coherence, Broadcast, Nearest Neighbor
  - **Full scientific applications** (nuclear simulation, climate modeling, …
- **Experience Bases**:
  - **Results EB**: Hypotheses, evidences, implications
  - **Defect EB**: Defect patterns, symptoms, causes cures, preventions, …)
- **Experimenters' Package:**
  - A checklist for instructors and experts running studies.
  - Includes templates, forms, and reusable project descriptions
  - Experiment Manager that supports data collection and analysis

Journey

- Phase I
  - Early days: Running isolated studies for a particular purpose
  - Independently using case studies and controlled experiments
- Phase II
  - Tying studies together in one environment, one domain
  - Controlled experiments, case studies, quasi-experiments, qualitative analysis tied together
- Phase III
  - Expanding out across domains, environments, technologies
  - Focusing on building knowledge for a couple of techniques
- Phase IV
  - Focusing on a domain to build a body of knowledge
  - Folklore gathering, interview, case studies, controlled experiments, experience bases, …

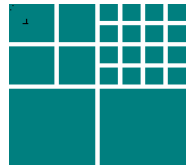
Evolving Empirical Studies Since 1976

- Early work was to characterize the effects of various methods, (*fixed all study variables*)
  - e.g., Iterative Enhancement, Chief Programmer Teams
- Then built baselines of various project variables (defects, effort, product and project metrics) for a single domain and environment, identifying where methods might make a difference (*fixed context, varied techniques*)
  - e.g., ground support software at NASA/GSFC (SEL).
- Then expanded out across several domains, environments, focusing on building knowledge for a couple of techniques (*fixed the techniques to study context*),
  - e.g., defect removal techniques, COTS-based development, and agile methods (CeBASE)

Evolving Empirical Studies Since 1976

- Then did empirical work to elicit and quantitatively define the software dependability needs of various stakeholders, identify the appropriateness and effectiveness of technologies to satisfy those needs under varying conditions before transferring them into practice, (*introduced testbeds (context) to study techniques*)
  - e.g., increasing the ability of NASA to engineer highly dependable software systems via new technologies (HDCP)

- Now working on building knowledge in a particular domain, packaging that knowledge in an experience base so it can be used by others, demonstrating the effectiveness of various approaches and in what context they are effective (*fixed domain, studying techniques and context variables*)
  - e.g., building a software domain experience base to help understand and increase the time and cost of developing high end computing (HEC) codes (HPCS)



Where are we and where are we going?

Types of Studies (1)

- Studies of **Techniques**
  - Feasibility
    - No technique should be published without trying it out
  - Feedback for improvement
    - Technique should be tested to see where they can be improved
  - Evaluation
    - We need to test the bounds and limits of each technique
  - Integration
    - We need to see how techniques can be integrated and what their integration buys you

Types of Studies (2)

- **Building knowledge** of a domain
  - Identify folklore, theories, …
    - Ethnographic studies, interviews, observations, …
  - Build models
    - Grounded theory, case studies, quasi-experiments, controlled experiments, …
  - Evolve models supported by evidence
  - Test models and hypotheses
    - experiments of all kinds
  - Integrate models
  - Find out what works

Community of Empiricists

- We have been evolving a community that talks to each other
- This is the 14th **ISERN** workshop and the number of members has grown dramatically
- The **Empirical Software Engineering Journal** is 11 years old and has a very good ISI Impact rating (.965, among SE journal)
- **ISESE** is in its fifth year
- But we need more of a community that works with each other
  - we haven't solved the terminology problem
  - Collaboration is necessary for defining a research agenda

Publication Experiment Guidelines

- Victor Basili, et. al.
  - Experimentation in Software Engineering, TSE 1986
- Barbara Kitchenham, et. al.
  - Preliminary Guidelines for Empirical Research in Software Engineering, TSE 2002
- Dag Sjoberg, et. al.
  - A Survey of Controlled Experiments in Software Engineering, TSE 2005
- Textbooks: Claes Wolin, et. al., Natalia Juristo, et. al.
- Others: Andreas Jedlitschka , ISERN 2004,

Publication Support Material

- Guidelines very long, especially for conference papers

- Need to be able to break studies into
  - Small useful modules
  - Backed up by TRs that deal with all guideline issues
  - Backed up by web site material

- Where to publish
  - Journals better than conferences due to feedback and dialog
  - We need to identify conference guidelines
  - Need to write Technical Reports

Publication Reviews Issues

- Papers need to build on each others work
  - Part of the history of isolated events
  - There is now a lot more literature
  - Need to grow a culture of reading, referencing, and assimilating existing material

- Example:
  - Criticized for lots of studies about “inspections” that don’t seem to recognize or integrate with the past work
  - **Issue 1**: there are many reading techniques, like many testing techniques that need to be developed, evolved, evaluated, etc.
  - **Issue 2**: as a community we have not always distinguished so why should anyone else – we need to be more scholarly

Context Variables

- This is the biggest problem
- There are the small variances, e.g., professional vs. student
- There are the big categories: environment, domain, class of SE technologies applied (how many variables are hidden in these?)
- If we are to build knowledge – we need to focus on specific domains, classes of technologies, environments, expanding out slowly, unifying across the differences
- Like we did in a single environment like the SEL.

Replications / Meta-Analysis

- Building theories requires replication
  - Varying the threats
  - Varying the artifacts
  - Varying the population
  - …
- Requires coordination and collaboration
  - It takes a team to run an experiment, hard to do alone
  - Multiple groups
  - Multiple disciplines

Convincing a software domain community

- This is our best bet at trying to
  - study the effectiveness of techniques
  - build a body of knowledge

- Look at the work of
  - Barry Boehm (government contractors and agencies)
  - Nancy Leveson (aeronautical engineering)
  - Elaine Weyuker (Telephony software)
  - …

- So do we work with software engineers or with software engineers in a domain?

Convincing the software engineering research community

- Empirical Study is here to stay
  - Software engineering techniques need to do studied empirically if it is to be anything other than a theoretical discipline
  - Many technology developers are already doing feasibility studies (although not called that) to study
  - They are not empiricists and don't want to or need to learn what it takes to do it
  - We need to supply them with some goals and methods, etc. or team with them

- What is the role of our community?
  - To develop such techniques and/or work with them, e.g., the HPCS project

Is there a future for empirical software engineering?

We have matured a lot in terms of the questions we ask, the types of studies we perform, and the development of a community

Software Engineering is **“big science**”;

and empiricism is a necessary ingredient of any big science