

Empirical Software Engineering: The Synergistic Relationship between Research and Practice

Victor R. Basili

Professor, University of Maryland

and

Director, Fraunhofer Center - Maryland

Outline

- Empirical Studies
 - Motivation
 - Specific Methods
 - Example: SEL
- Example Consortia
 - Fraunhofer Center – Maryland
 - CeBASE

Motivation for Empirical Software Engineering

Understanding a discipline involves **building models**,
e.g., application domain, problem solving processes

And checking our understanding is correct,
e.g., testing our models, **experimentation** in the real world

Analyzing the results involves **learning**, the **encapsulation of knowledge** and the ability to change or refine our models over time

The understanding of a discipline evolves over time

This is the paradigm that has been used in many fields,
e.g., physics, medicine, manufacturing

But this requires real world laboratories

Motivation for Empirical Software Engineering

Like other disciplines, **software engineering** requires the cycle of model building, experimentation, and learning

The study of software engineering is a **laboratory science**

Research needs laboratories to observe & manipulate the variables

- they only exist where developers build software systems

Development needs to understand how to build systems better

- research can provide models to help

Research and Development have a **symbiotic relationship**

requires a working relationship between industry and academe

Motivation for Empirical Software Engineering

For example, a software organization needs to ask:

What is the right combination of technical and managerial solutions?

What are the right set of process for that business?

How are they tailored?

How do they learn from their successes and failures?

How do they demonstrate sustained, measurable improvement?

More specifically:

When are peer reviews more effective than functional testing?

When is an agile method appropriate?

When do I buy rather than make my software product elements?

Examples of Using Empirical Results

Example: When are peer reviews more effective than functional testing?

- **Peer reviews** are more effective than functional testing for faults of **omission** and **incorrect specification** (Based upon studies at UMD and 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 applied research**:

- How can peer reviews be improved with better techniques and technology to identify faults of omission and incorrect fact?

Basic Concepts for Empirical Software Engineering

This process of model building, experimentation and learning requires the development, tailoring and evolution of methods that support evolutionary learning, closed loop processes, well established measurement processes and the opportunity to build software core competencies

As well as processes that support the development of software that is relevant to the needs of the organization can be predicted and estimated effectively satisfies all the stakeholders does not contain contradictory requirements

Basic Concepts for Empirical Software Engineering

The following concepts have been applied in a number of organizations

Quality Improvement Paradigm (QIP)

An evolutionary learning paradigm tailored for the software business

Goal/Question/Metric Paradigm (GQM)

An approach for establishing project and corporate goals and
a mechanism for measuring against those goals

Experience Factory (EF)

An organizational approach for building software competencies and
supplying them to projects

Basic Concepts for Empirical Software Engineering

The following approaches have been applied in a number of organizations

COCOMO Cost and Schedule Models

A family of parametric models that support software prediction and tracking

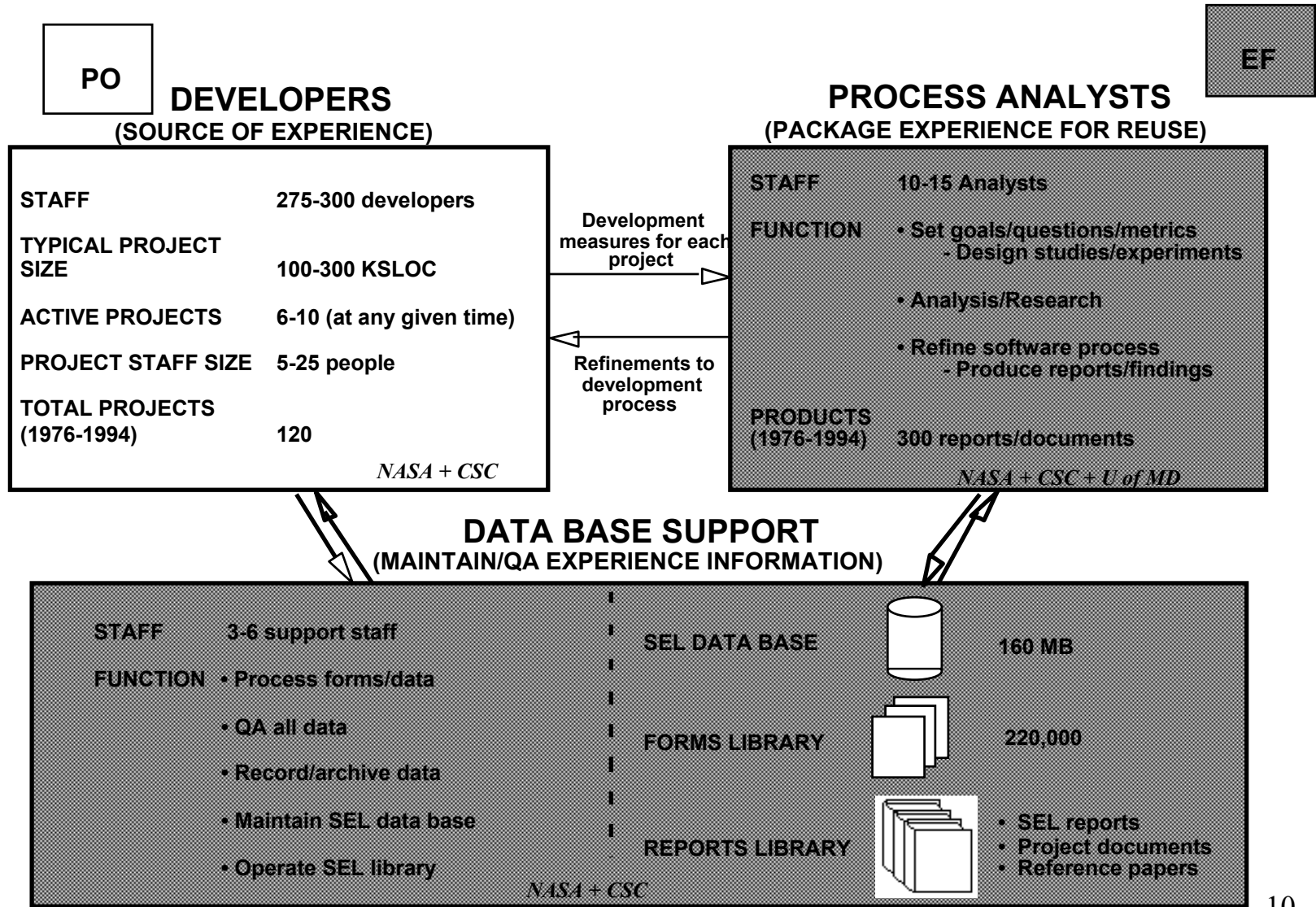
Win/Win Spiral Model Processes

An approach for establishing agreed upon project goals and developing a software system in a risk averse evolutionary way

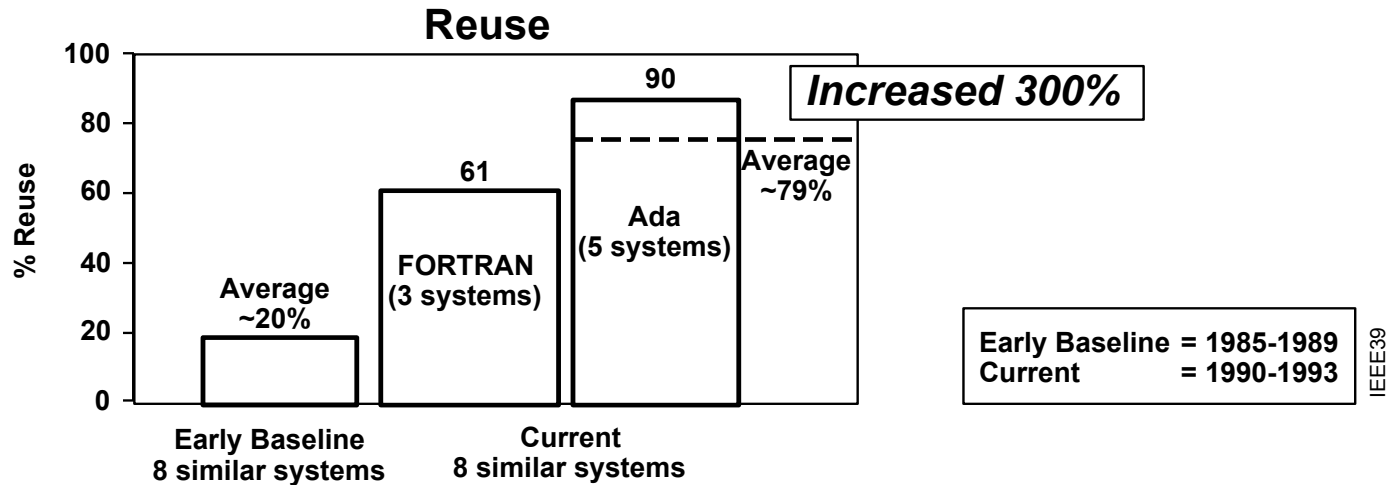
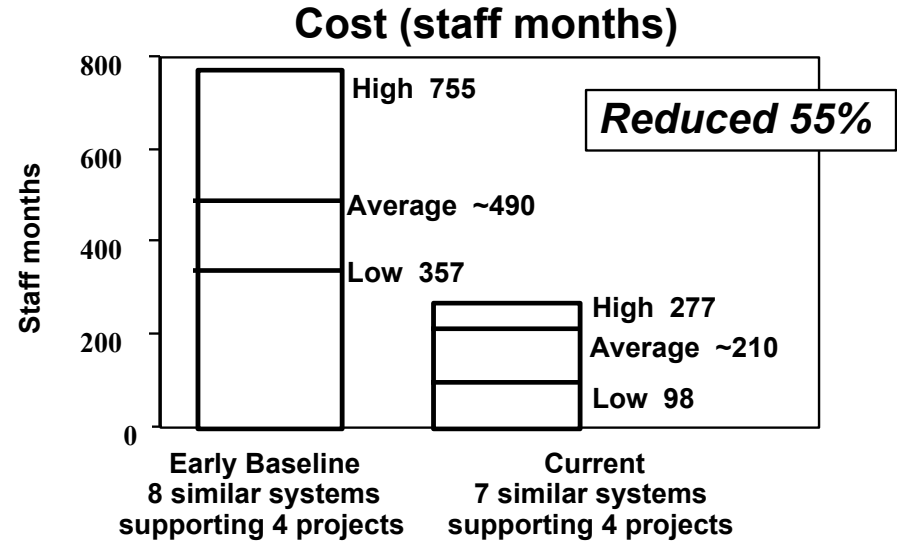
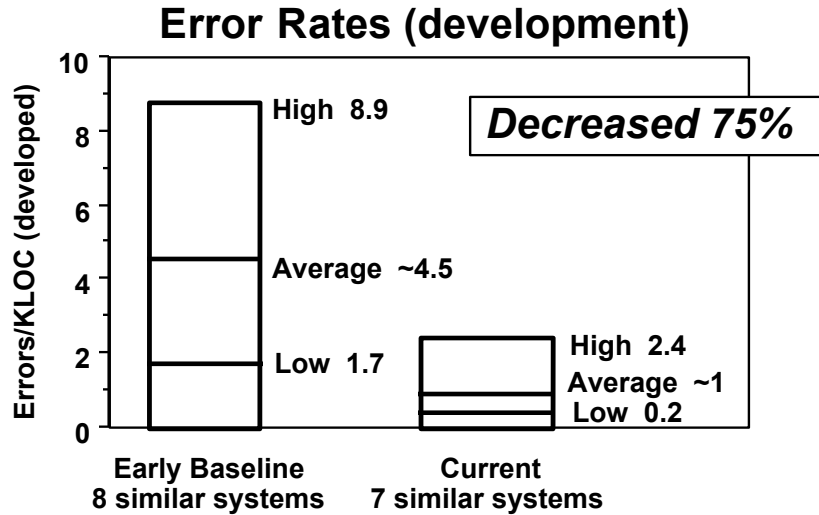
MBase

A guideline for developing software systems that permits anchor points for checking project status and maturity

SEL: An Example Experience Factory Structure



Using Baselines to Show Improvement 1987 vs. 1991



Using Baselines to Show Improvement 1987 vs. 1991 vs. 95

Continuous Improvement in the SEL

Decreased **Development Defect rates** by
75% (87 - 91) **37%** (91 - 95)
Reduced **Cost** by
55% (87 - 91) **42%** (91 - 95)
Improved **Reuse** by
300% (87 - 91) **8%** (91 - 95)
Increased **Functionality** five-fold (76 - 92)

CSC officially assessed as CMM level 5 and ISO certified (1998),
starting with SEL organizational elements and activities

Fraunhofer Center for Experimental Software Engineering - 1998

CeBASE Center for Empirically-based Software Engineering - 2000

Example Model for Learning Organization

Interact with various industrial, government and academic organizations to open up the domain for learning

Partner with other organizations to expand the potential competencies

Observe and gather as much information as possible

Analyze and synthesize what has been learned into sets of best practices recognizing what has been effective and under what circumstances allowing for tailoring based up context variables

Package results for use and feed back what has been learned to improve the practices

Example 1: Fraunhofer Center for Experimental Software Engineering, Maryland

Fraunhofer Center - Maryland

Not-for profit, applied research, technology transfer organization

Affiliated with the **University of Maryland, College Park**

Founded 1998 as part of Fraunhofer USA (Fraunhofer Gesellschaft)

Work with academic, government and industrial organizations

Budget of \$2.5 million/year (\$600K base funding)

FC Vision

Apply a scientific/engineering method to software engineering

Utilize past results to guide development choices

Use organizational learning as the key to improvement

Fraunhofer Center, Maryland

Principal Core Competency Areas

Experience Factory, Measurement, Evaluating Technology Maturity, Process Improvement, Reading Techniques, Agile Development, Architectural Evaluation, Risk Management, Process Modeling, COTS-based Development, Security

Approach: Partner with leading world class organizations and build competencies by identifying specific needs and responding by selecting and evolving the relevant tools and techniques

Use this interaction to enhance knowledge in a competence area

Fraunhofer Center, Maryland

Core Competence: Experience Factory Organization

Build learning organizations

Develop experience bases and tools for using them

Use knowledge management to analyze and synthesize information for managers and developers

Customers: NSF, DoD, Boeing, SAIC, ...

Partners: UMD, USC, ...

Project Focus (Use/support the EF/EB Competence):

EB/EF tools - VQI, FAQ, eWorkshop, LL EB, ...

EB for Defect Reduction, COTS, Agile

EB for Software Process Improvement for small companies

EB for Risk Management and Lessons Learned

EB of Complex System of Systems Lessons Learned

.....

Example 2: CeBASE

Center for Empirically Based Software Engineering

The **CeBASE** project (a Virtual Center) was created to support the symbiotic relationship relationship between research and development, academia and industry

Virtual Research Center

Created by the NSF Information Technology Research Program

Co-Directors: Victor Basili (UMD), Barry Boehm (USC)

Initial technology focus: Defect reduction techniques, COTS based development, Agile Methods

Initial CeBASE Funding: \$2.4 million for the first two years

CeBASE

Center for Empirically Based Software Engineering

CeBASE Framework

Integration of a proven set of methods

Experience Factory, Goal/Question/Metric Approach, Spiral Model extensions, MBASE, WinWin Negotiations

Existing CeBASE-developed tools

Electronic Process Guide, EasyWinWin requirements, eWorkshop collaboration, COCOMO cost family, EMS Experience Base, VQI (Virtual Query Interface)

CeBASE

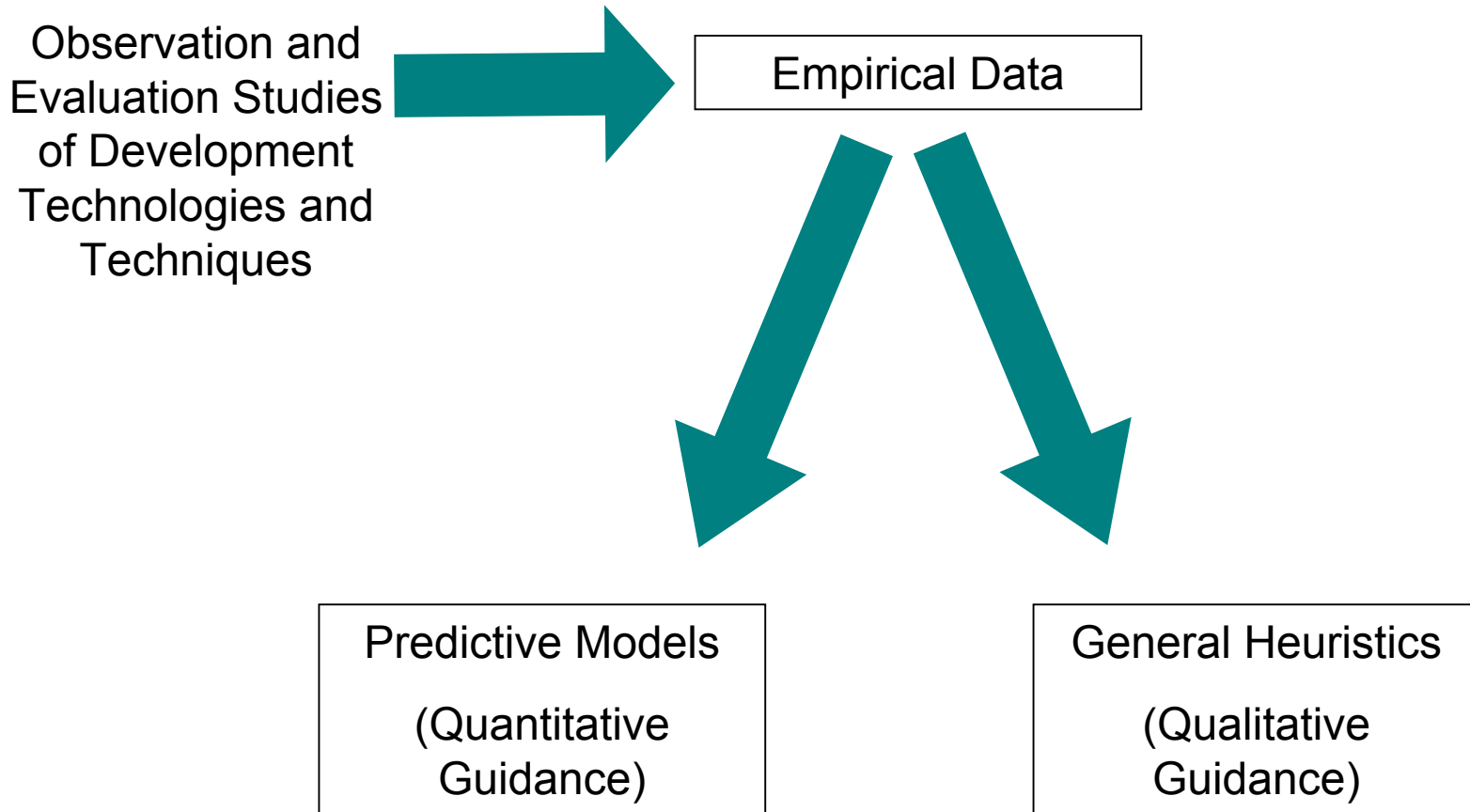
Center for Empirically Based Software Engineering

CeBASE Project Goal: Enable a **decision framework and experience base** that forms a basis and an infrastructure for research and education in empirical methods and software engineering

CeBASE Research Goal: Create and evolve an **empirical research engine** for evaluating and choosing among software development technologies



CeBASE Research Activities



E.g. COCOTS excerpt:

Cost of COTS tailoring = $f(\# \text{ parameters initialized, complexity of script writing, security/access requirements, ...})$

E.g. Defect Reduction Heuristic:

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

CeBASE

Center for Empirically Based Software Engineering

To define, test, and evolve the **empirical research engine**, it is necessary to apply the concepts and get feedback on their effectiveness

Needed to build a useable, effective **decision framework and experience base** that supports research and education in empirical methods and software engineering

To this end, CeBASE has expanded to three levels of activity

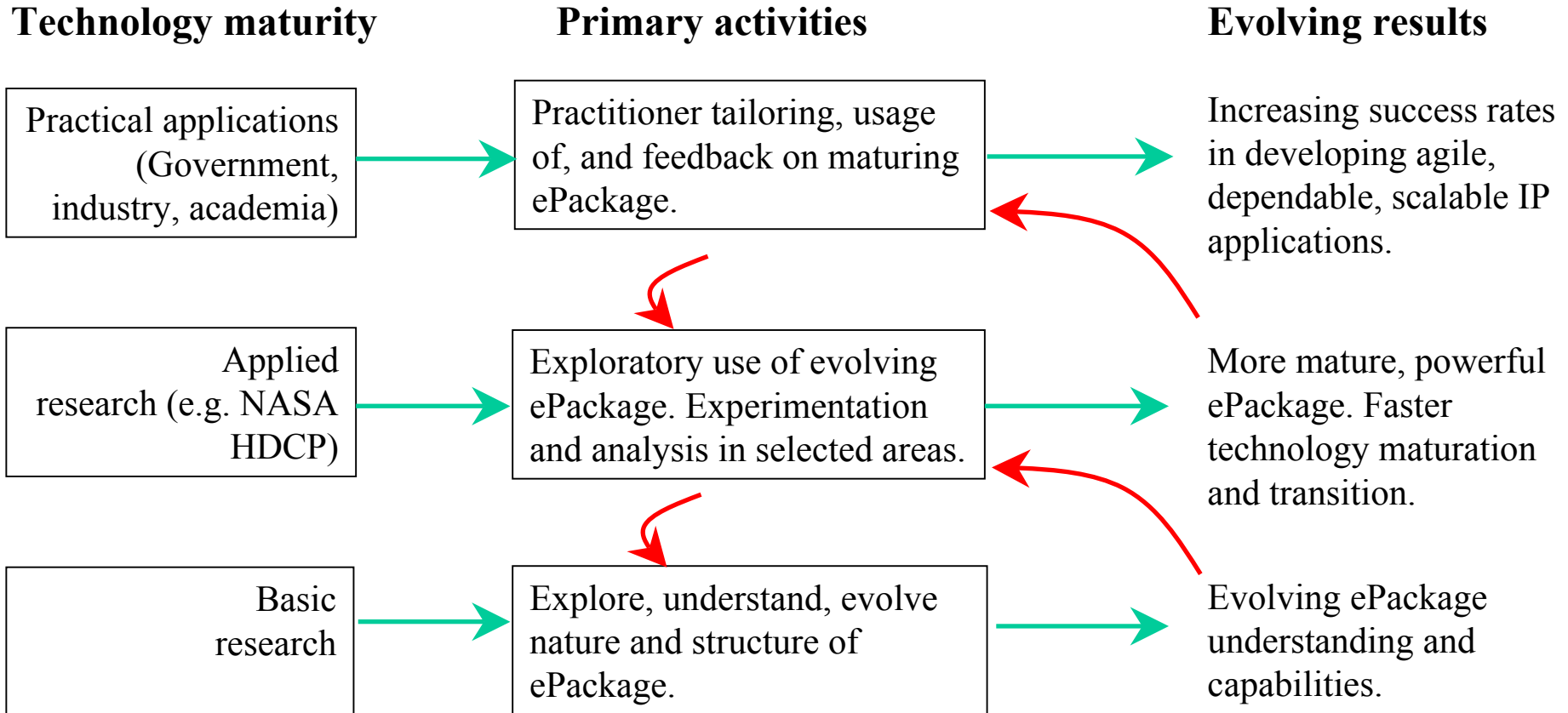
Basic Research: to develop and package the components of the empirical research engine, and empirical decision framework (ePackage)

Applied Research: to explore the use of this evolving ePackage by experimentation and analysis in selected areas

Technology Transfer: to apply and tailor the approach for practice

CeBASE

Three-Tiered Empirical Research Strategy



(ePackage = Empirical Research Engine, eBase, empirical decision framework)

Example CeBASE Project: Applied Research NASA High Dependability Computing Program

Project Goal: Increase the NASA's ability to **engineer highly dependable software systems** via the development of new techniques, processes, and technologies.

Research Goal: **Develop high dependability technologies** and assess their effectiveness under varying conditions and transfer them into practice at NASA

Partners: CMU (PI), MIT, USC, U. Washington, 2002-2006

CeBASE Funding: \$1 million/year for 5 years

Example CeBASE Project: Technology Transfer Future Combat Systems

Project Goal: Support FCS Program Management Office in the development of the Future Combat Systems (FCS), focusing on the complex system of systems (software) development risk, e.g., acquisition, architecture, ... and build lessons learned for future iterations of FCS and future CSoS.

Research Goal: Build a risk experience Base and a Complex System of Systems Lessons Learned Experience Base.

Partners: UMD, USC, FC-MD, SEI, Sandia, LSI: Boeing, SAIC

CeBASE Funding: ~ \$1 million / year

Conclusion

- This talk is about
 - the evolution of knowledge in software engineering
 - The use of empirical software engineering research
 - **The synergistic relationship between research, applied research, and practice**
- Software developers need to know what works and under what circumstances
- Technology developers need feedback on how well their technology works and under what conditions
- We need
 - to continue to collect empirical evidence
 - analyze and synthesize the data into models
 - evolve software engineering into an engineering discipline