

<http://cebase.org>



Fraunhofer USA Center for
Experimental Software Engineering
Maryland

Outline

- **Motivation for the Center**
- **Center Vision and Approach**
- **Center Organization**
- **Track Record**
- **Examples of Existing Empirical Results**
- **Expected Benefits for**
 - **Industry**
 - **Research**
 - **Education**
- **Collaborations**

Motivation for the Center

- Software development teams need to understand the right models and techniques to support their projects. For example:
 - When are peer reviews more effective than functional testing?
 - When should you use a procedural approach to code reviewing?
 - How should you tailor a life cycle model for your environment?
- Too often, such decisions are based on anecdote, hearsay, or hype
- Developers are often surprised to learn that 25 years of empirical software engineering research has provided usable answers to these and numerous related questions.

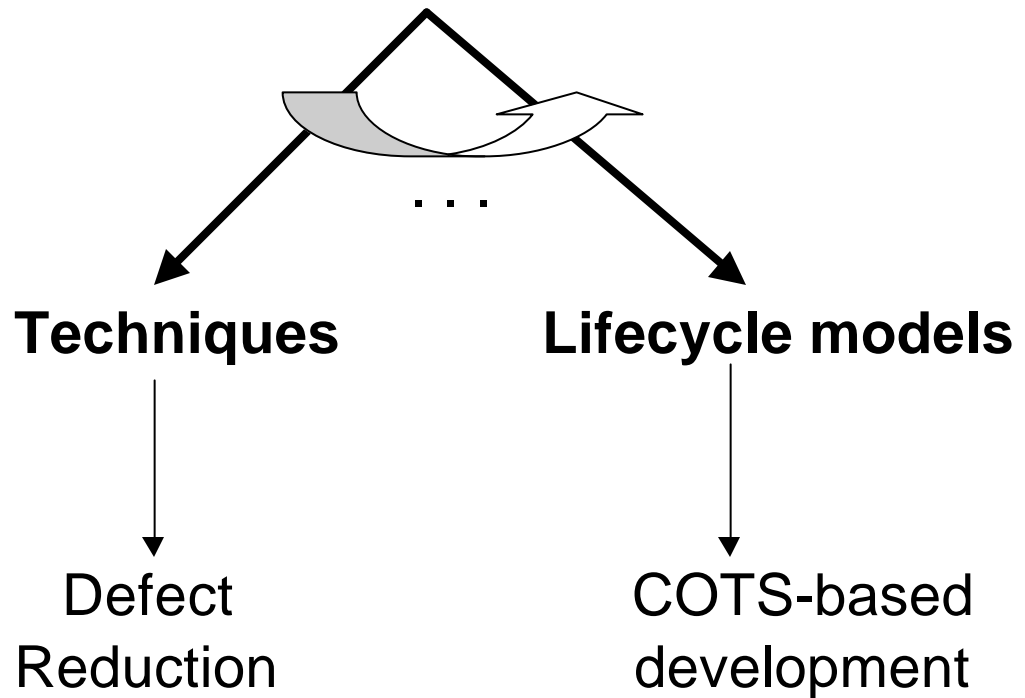
CeBASE Vision and Approach

- **Goal: An empirically based software development process**
 - covering high level lifecycle models to low level techniques
 - necessary step toward a scientific foundation for software engineering in which the effects of development decisions are well understood
 - we can't improve without a rigorous foundation
- **A first step is an empirical experience base**
 - validated guidelines for selecting techniques and models
 - ever 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...

CeBASE Approach

Empirical Software Engineering Research

Investigates a spectrum of activities...



Initially we will focus on...

Center Organization

- **Research center sponsored by**
 - NSF Information Technology Research Program
- **Co-Directors**
 - Victor Basili (UMD), Barry Boehm (USC)
- **Co-PI's**
 - Marvin Zelkowitz (UMD), Rayford Vaughn (MSU), Forrest Shull (FC-MD), Dan Port (USC), Ann Majchrzak (USC), Scott Henninger (UNL)
- **Initial 2-year funding: \$2.4 M**

Track Record of the PI's

- **UMD and FC-MD**
 - 25 years of strong large-project research results
 - framework for empirical research widely adopted (GQM, EF, ...)
 - leading to improved software engineering at NASA/Goddard Space Flight Center, CSC, Daimler Chrysler, Motorola...
- **USC-CSE**
 - Process frameworks and predictive models widely adopted
 - COCOMO, Spiral model, risk management framework, ...
 - Network of collaborators: FAA, US Army Research Labs, Hughes, Rational, ...

25 Years of Learning

Experiences with the Software Engineering Laboratory (SEL)

Consortium of

NASA/GSFC

Computer Sciences Corporation

University of Maryland

Established in 1976

Goals have been to

- better understand software development

- improve the process and product quality

at Goddard, formerly in the Flight Dynamics Division, now at the Information Systems Center

using observation, experimentation, learning, and model building

Observation, Feedback, Learning, Packaging

Learned a great deal

Observation played a key role

Feedback loops have provided an environment for learning

Generated lessons learned that have been packaged into our process, product and organizational structure

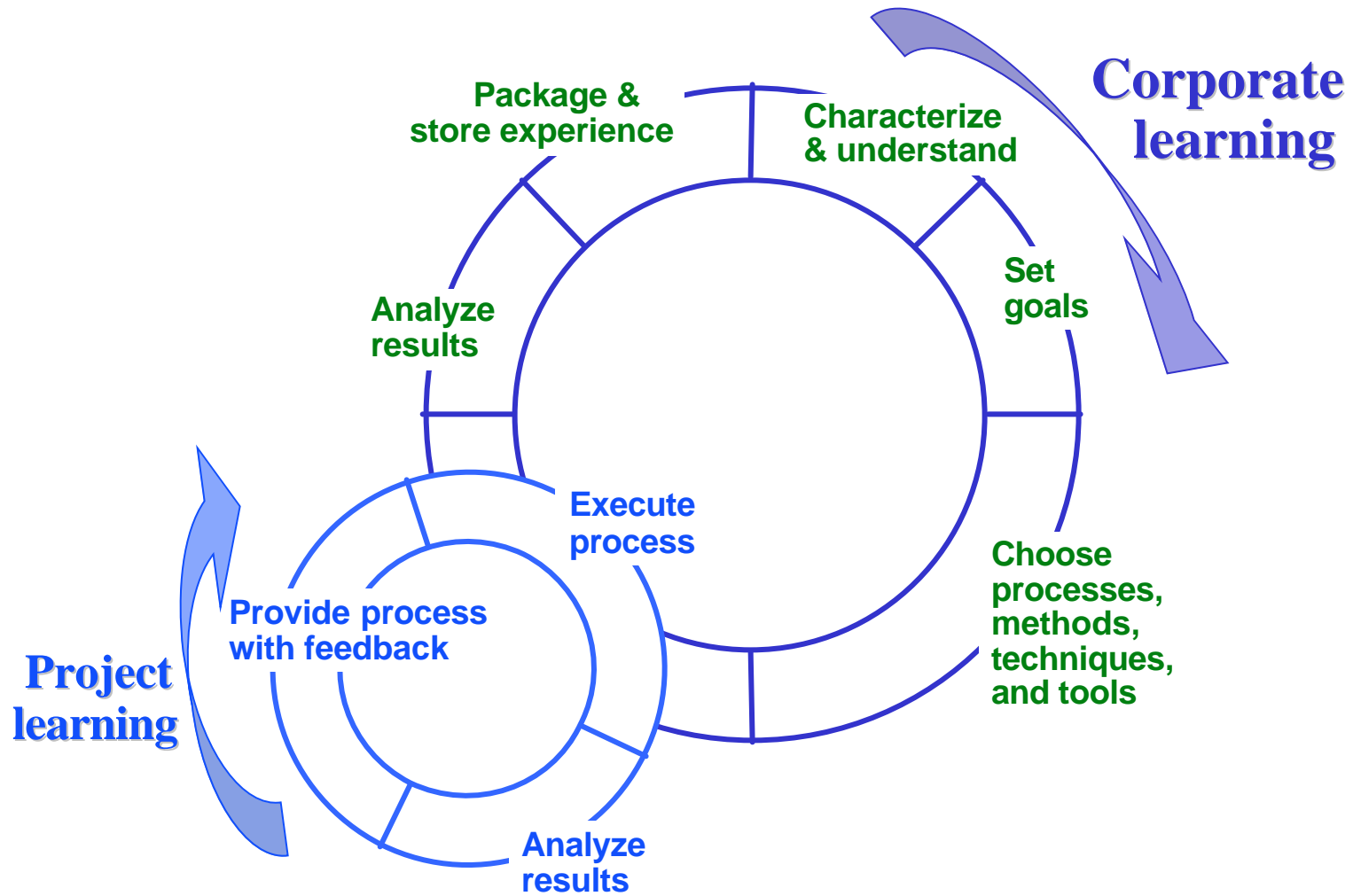
Used the SEL as a laboratory to build models, test hypotheses,

Used the University to test high risk ideas

Developed technologies, methods and theories when necessary

Learned what worked and didn't work, applied ideas when applicable

Kept the business going with an aim at improvement, learning





Characterize

metrics ----> baselines ----> models

Set Goals

data driven ----> goal driven ----> goal/model driven

Select Process

heuristic ----> defined ----> high impact ----> evolving
combinations technologies combinations processes

Execute Process

add-on data collection ----> less data ----> data embedded in process

Analyze

correlations ----> regressions ----> model ----> qualitative analysis

Package

recording ----> lessons learned ----> focused tailored packages
defect ----> resources ----> product ----> process x product
baselines models characteristics relationships

Quality Improvement Paradigm 1976 - 1980



~~Characterize/Understand~~ Apply Models

Looked at other people's models, e.g., Rayleigh curve, MTTF models

~~Set Goals~~ Measurement

Decided on **measurement as an abstraction mechanism**

Collected data from half a dozen projects for a simple data base

Defined the **GQM** to help us organize the data around a particular study

~~Select Process~~ Study Process

Used heuristically defined combinations of existing processes

Ran **controlled experiments** at the University

Execute Process

Data collection was an add-on activity

Analyze Data Only

Mostly **built baselines** and looked for correlations

~~Package~~ Record

Recorded what we found, built defect baselines and resource models

Quality Improvement Paradigm 1976 - 1980



Learned

Need to **better understand** environment, projects, processes, products, etc.

Need to **build our own models** to understand and characterize locally

Data collection has to be **goal driven**

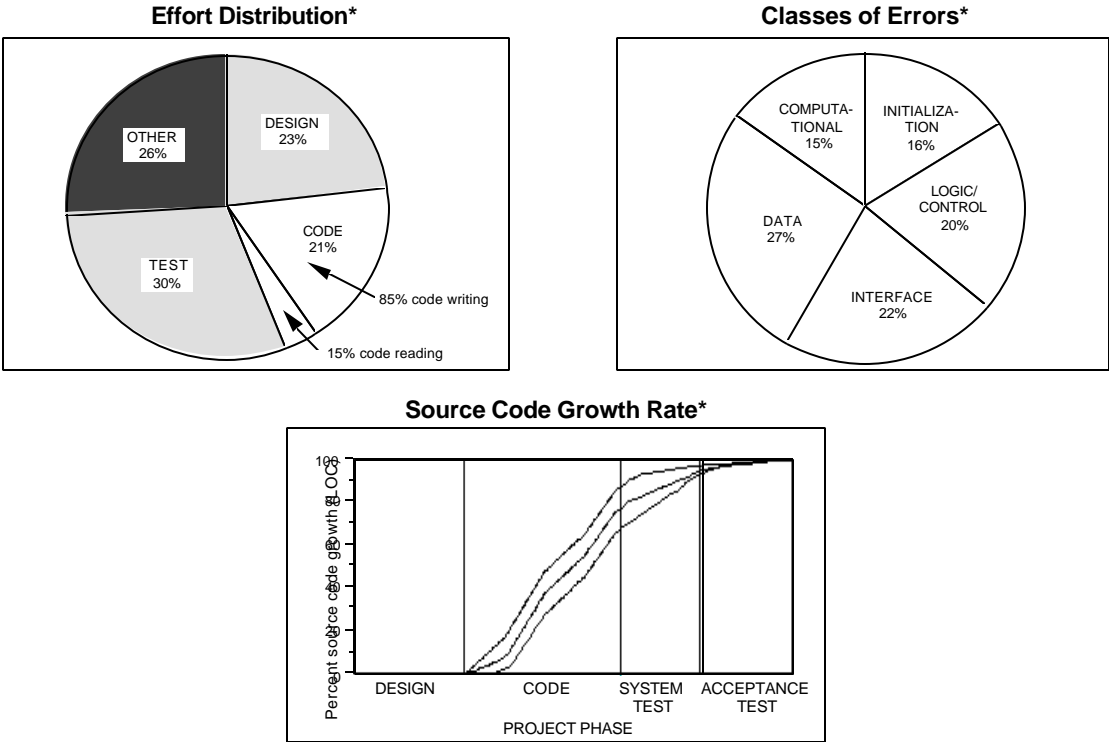
...

The Goal/Question/Metric Paradigm

Creating Baselines



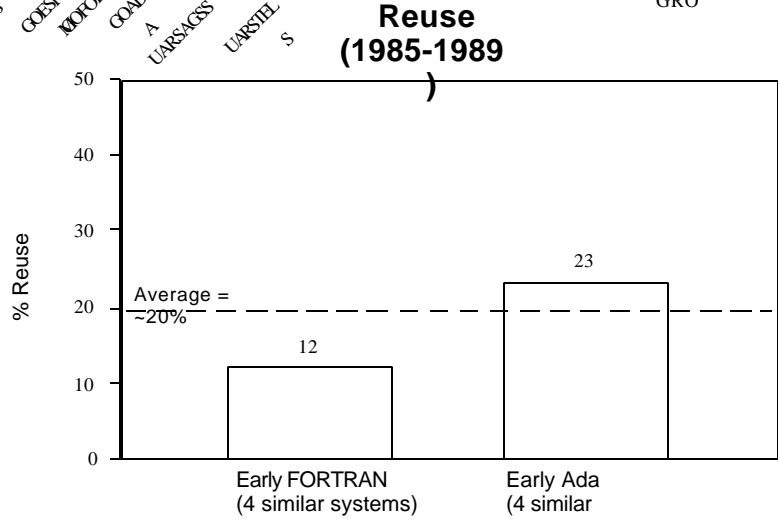
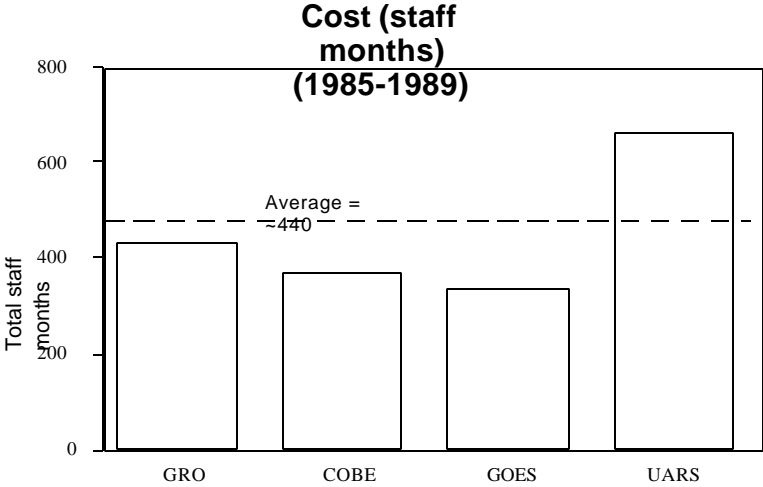
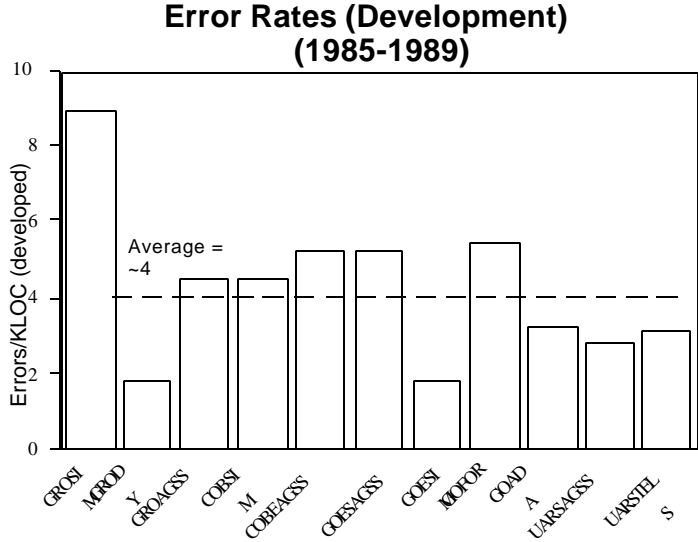
NASA/SEL PROCESS BASELINE EXAMPLE



*Data from 11 Flight Dynamics projects (mid 1980s)



NASA/SEL Product Baseline Example





~~Characterize/Understand~~

Built our own baselines/models of cost, defects, process, etc.

Set Goals

Set GQM goals to study multiple areas

Incorporated subjective measures or indices

Select Process

Experimented with well defined technologies, e.g., Ada & OOD

Execute Process

Combine experiments and case studies

Collected less data

Analyze

Emphasis on process and its relation to product characteristics

~~Package Record~~

Recorded lessons learned

Formalize process, product, knowledge and quality models

Quality Improvement Paradigm 1981 - 1985



Learned

Software development follows an **experimental paradigm**

Need to **experiment** with technologies

Need to **learn about relationships**

Reuse experience of all kinds

Can drown in **too much data**

...



Characterize/Understand

Capturing experience in models

Set Goals

Goals and Models commonplace driver of measurement

Built a **model-based experience base** with dozens of projects

Select Process

Tailored and evolved processes based on experience

Execute Process

Embedded data collection into the processes

Analyze

Demonstrated various **(process, product) relationships**

Package

Developed focused tailored packages, e.g., generic code components

Learned to transfer technology better through organizational structure, experimentation, and evolutionary culture change

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



Learned

Experience needs to be **evaluated, tailored, and packaged** for reuse

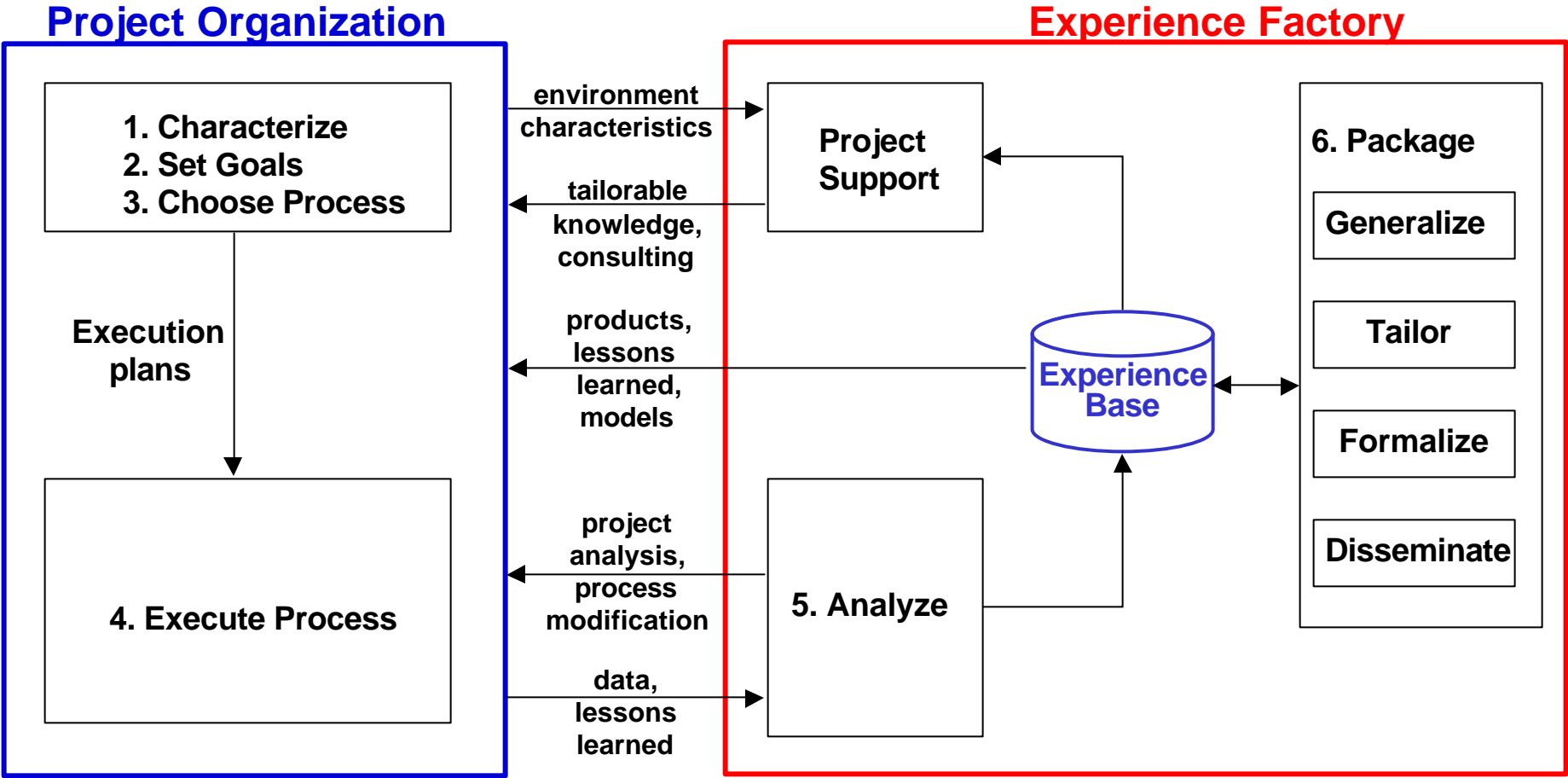
There is a **tradeoff between reuse and improvement**

Processes must be put in place to **support the reuse of experience**

Packaged experiences need to be integrated

...

The Experience Factory Organization





A Different Paradigm

Project Organization Problem Solving

Experience Factory Experience Packaging

Decomposition of a problem
into simpler ones

Unification of different solutions
and re-definition of the problem

Instantiation

Generalization, Formalization

Design/Implementation process

Analysis/Synthesis process

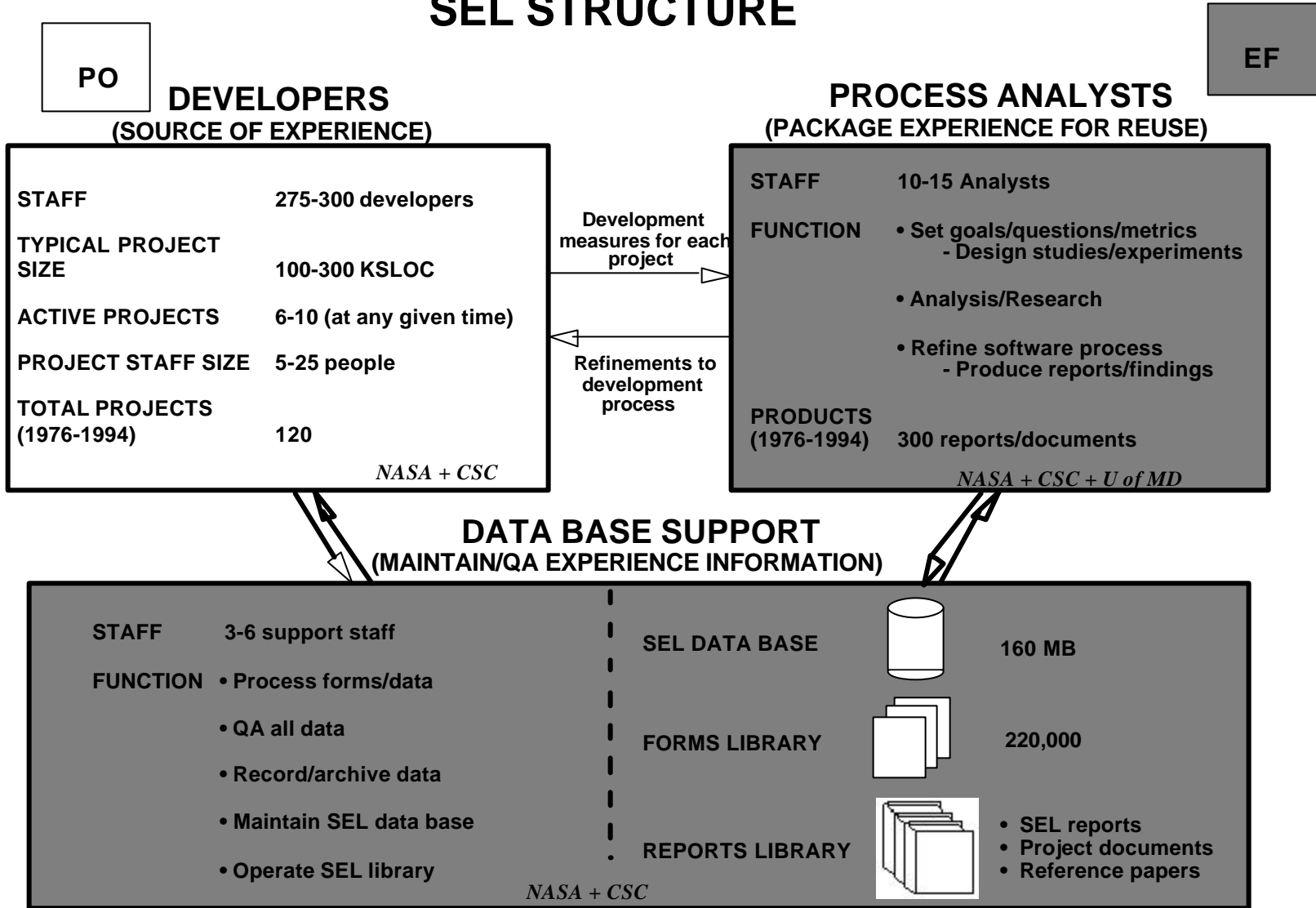
Validation and Verification

Experimentation

**Product Delivery within
Schedule and Cost**

**Experience / Recommendations
Delivery to Project**

SEL STRUCTURE





Characterize

Built baselines and used them to show differences, improvements
Built (process,product) relationship models

Set Goals

Used baselines to establish usable goals, provide evaluation criteria

Select Process

Studied process conformance and domain understanding
Developed **reading techniques** (understanding for use)
Developed **OO framework** for flight dynamics software

Execute Process

Captured the details of experience - more effective feedback

Analyze

More qualitative analysis to extract experiences, . e.g., interviews

Package

Evolved and packaged the Experience Factory Organization



Learned

Can **develop technology based upon need**, e.g., reading techniques

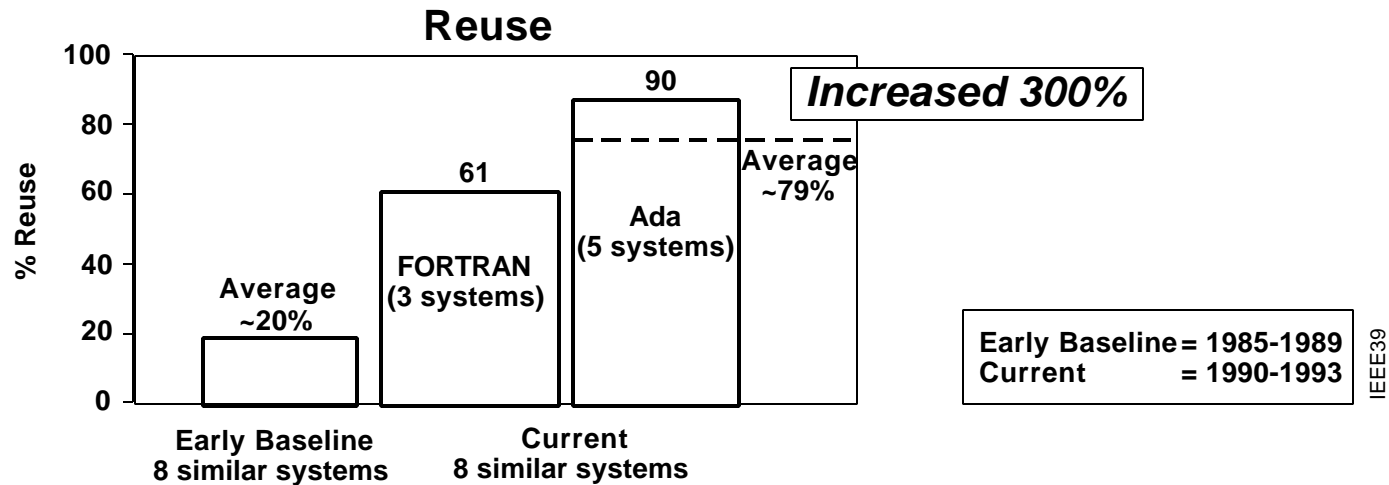
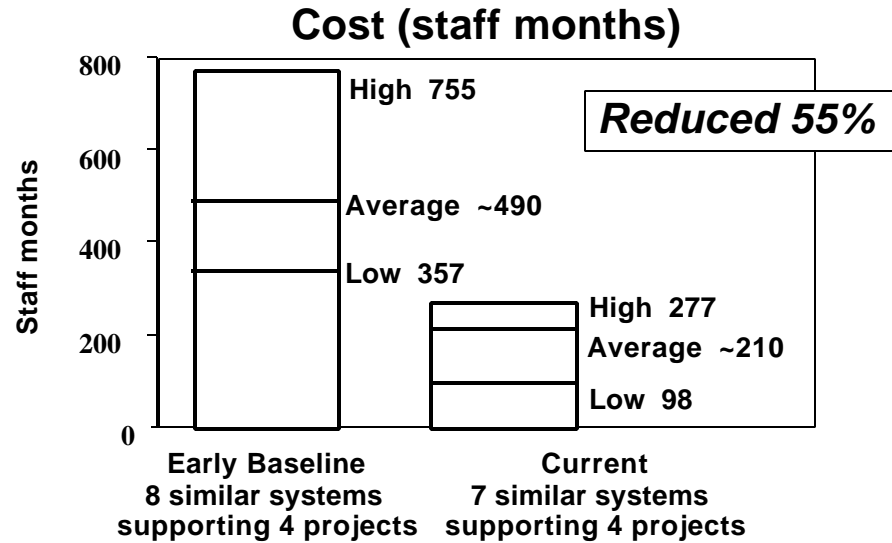
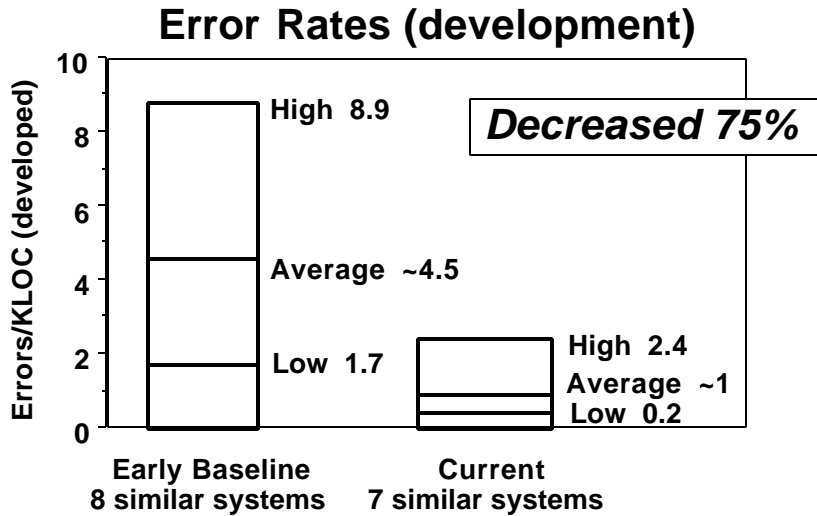
Need to provide projects with short term results

Learning in an organization is time consuming and sequential

Need to find ways to speed up the learning process

Can better understand the criteria for **sharing best practices**

Can **package the meta-models**, e.g., Experience Factory





The Software Engineering Laboratory

was awarded the first

**IEEE Computer Society Award
for
Software Process Achievement**

The award

an international award established in 1994
sponsored by the Software Engineering Institute (SEI)
for demonstrable, sustained, measured, significant process improvement

Effects of the SEL Activities Since 1996



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 - Maryland created 1998

CeBaSE

Center for Empirically-Based Software Engineering created 2000



Since 1996

ISC Baseline and Measurement

- characterize processes products and people
- effort and defect prediction models for the various branches
- core metrics for contracting and development

COTS Studies

- study and evolve the SEL COTS process
- define classification schemes for COTS integration
- build cost estimation models for COTS development

Reuse/Frameworks

- defining a framework-based product line for flight software

Reading Techniques

- perspective-based requirements reading
- object oriented design reading



Center for Experimental Software Engineering, Maryland (FC-MD)

- FC-MD
 - Applied Research and Technology Transfer Focus
 - Part of Fraunhofer USA, a U.S., not-for-profit affiliate of [Fraunhofer Gesellschaft](#)
 - Affiliated with Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern, Germany
- Background
 - Opened 1998
 - Located adjacent to University of Maryland campus
 - Center Directors are Victor Basili and Marvin Zelkowitz (University of Maryland)
- Size (October 2000)
 - 11 permanent Fraunhofer employees
 - 10 part-time faculty/visitors/students
 - Growing in staff to meet business needs



Expanding the Learning Organization The Fraunhofer Center since 1998

- Consortia Projects
 - Software Experience Center Project (w/ IESE & multi-national companies)
 - Maryland Software Industrial Consortium (state government)
- Applied Research Projects and Tech Transfer Projects
 - **Experience Management System**, EMS (FC-MD/UMD)
 - **Reading Techniques** to Improve Inspections (industry, government)
 - Experience Factory Support (industry)
 - Experience Management System Support (industry)
 - **Measurement** Program Support (industry)
 - **Dynamic Process models** for Testing (industry)
 - **ROI** Project for IV&V (government)
 - NASA/Software Engineering Laboratory (government)
 - Small Business Learning Organization/CMM Project (industry)
 - Software Acquisition Support (FC-MD, IESE, industry)



Conclusion

Since 1976 have **learned a great deal** about software improvement

Learning process has been **continuous and evolutionary**

Packaged learning into our process, product and organizational structure

Build bodies of **empirically validated results**

Supported by the **symbiotic relationship between research and practice**

Requires **patience and understanding** on both sides,
but when nurtured, really pays dividends!

25 Years of Learning Conclusion

Improvement of software competence is an essential business need

We need to

- build software core competencies as part of our overall business strategy**
- create organizations for continuous learning to improve software competence**
- generate a tangible corporate asset: an experience base of competencies**

QIP/GQM/EF represents a promising approach

- a Lean Software Development concept**
- compatible with TQM concepts**
- offering a CMM level 5 organizational structure**

Examples of Useful Empirical Results

“Under specified conditions, ...”

Technique Selection Guidance

- **Peer reviews** are more effective than functional testing for faults of **omission** and **incorrect specification** (UMD, USC)
- **Functional testing** is more effective than reviews for faults concerning **numerical approximations** and **control flow** (UMD, USC)

Technique Definition Guidance

- For a reviewer with an average experience level, a **procedural approach** to defect detection is more effective than a less procedural one. (UMD)
- Procedural inspections, based upon **specific goals**, will find defects related to those goals, so inspections can be customized. (UMD)
- Readers of a software artifact are more effective in uncovering defects when each uses a **different and specific focus**. (UMD)

Examples of Useful Empirical Results

Life Cycle Selection Guidance

- The **sequential waterfall model** is suitable if and only if
 - The **requirements** are **knowable** in advance,
 - The **requirements** have no **unresolved**, high-risk implications,
 - The **requirements satisfy** all the key stakeholders' **expectations**,
 - A viable **architecture** for implementing the requirements is known,
 - The **requirements** will be **stable** during development,
 - There is **enough calendar time** to proceed sequentially. (USC)
- The **evolutionary development model** is suitable if and only if
 - The **initial release** is **good** enough to keep the key stakeholders involved,
 - The **architecture** is **scalable** to accommodate needed system growth,
 - The operational user organizations can adapt to the **pace of evolution**,
 - The **evolution dimensions** are compatible with legacy system replacement,
 - **appropriate** management, financial, and incentive **structures** are in place. (USC)

Industry Benefits

- **Software too fragile, unpredictable (Presidential Commission Report)**
- **“No-surprise” software development (NSF Workshop Report)**
- **Industry needs empirically-validated decision aids for choosing and customizing development approaches**
- **We will support industry by developing**
 - ***an understanding of defects (and a means to minimize them) grounded in careful empirical analysis instead of folklore***
 - ***empirical metrics and predictive models for project monitoring***

Research Benefits

- **To advance software engineering, SE researchers must**
 - identify and solve significant software development problems
 - validate the solutions
- **We will support empirical researchers to**
 - *engage in collaborations with industry*
 - *enable integration of results for more robust conclusions*
 - *evaluate, refine, and extend results and methods*
 - *package and disseminate results via educational materials and activities*

Educational Benefits

- **To advance software engineering, educators must**
 - teach high impact methods
 - offer courses with relevant and timely results
 - give students experience with realistic artifacts
 - educate a stronger community of empirical researchers
- **We will support educators by**
 - *providing material for training students on how to select and tailor, not just apply, the right SE methods and tools*
 - *providing realistic artifacts as teaching materials*
 - *designing SE educational techniques supporting experimentation*

Essential Interactions

- **We are looking for**
 - **development projects and support groups**
 - **research organizations**
 - **educational institutions**
- **To help build, evaluate, and share the empirical experience base**
- **Contact us at** <http://www.cebase.org>

Essential Interactions

- **Research Collaborators**
 - **Coordinate definitions of data, models**
 - **Coordinate observations, methods, data collection & analysis**
 - **Share data and results (to the extent possible)**
 - **Improve scoping, compatibility of results**
 - **Collaboration on implications, packaging, dissemination of results**
 - **Participate in research workshops**

Essential Interactions

- **Practice Collaborators**
 - **Coordinate definitions of data, models**
 - **Provide data; participate in experiments**
 - **Collaboration on implications, packaging, dissemination of results**
 - **Use results, provide feedback for improvement**
 - **Participate in practice workshops**
 - **Evaluation of results, technology transition**
 - **Issue identification**
 - **Priorities for future research**