Software Improvement Feedback Loops: The SEL Experience

Victor R. Basili

Institute for Advanced Computer Studies
Department of Computer Science
University of Maryland
and
Fraunhofer Center - Maryland

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

Quality Improvement Paradigm

Characterize the current project and its environment with respect to the appropriate models and metrics

Set quantifiable **goals** for project and corporate success and improvement

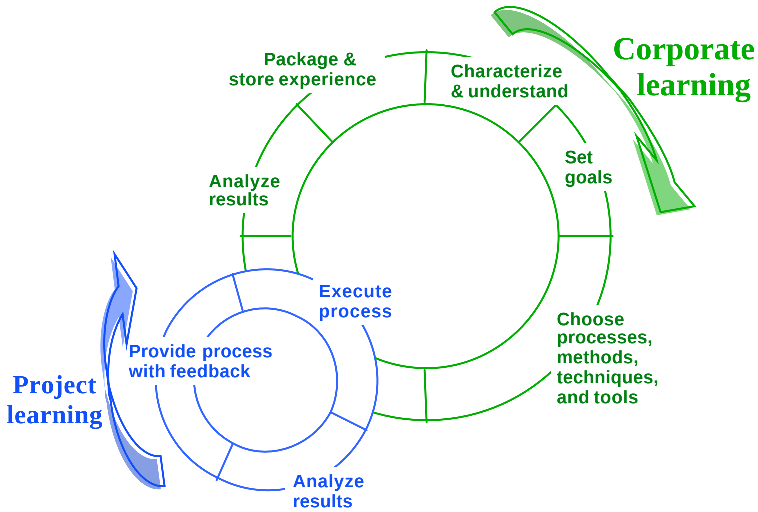
Choose the appropriate project **processes**, supporting methods and tools

Execute the **processes,** construct the products, collect, validate and analyze the data to provide real-time feedback for corrective action

Analyze the **data** to evaluate current practices, determine problems, record findings, recommend improvements for future project

Package the **experience** in the form of updated and refined models and save it in an experience base to be reused on future projects.

Quality Improvement Paradigm



Maturing the Improvement Paradigm
Major Activity Evolution

Characterize

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

Set Goals

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

Select Process

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

Execute Process

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

Analyze

correlations , regressions ----> quantitative/qualitative analysis

Package

recording ----> lessons learned ----> focused tailored packages

defect baselines , resources models , product characteristics ----> process x product 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 and was loosely monitored

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.

which factors create similarities and differences among projects

how to choose the right processes for the desired product characteristics

how to evaluate and feed back information for project control

Need to build our own models to understand and characterize locally

- can't just use other people's models

Data collection has to be goal driven

- can't just collect data and then figure out what to do with it

...

Developed the Goal/Question/Metric Paradigm

Quality Improvement Paradigm
Goal/Question/Metric Paradigm

A mechanism for defining and interpreting operational, measurable goals

It uses four parameters:

a **model** of an object of study,
e.g., a process, product, or any other experience model

a **model** of one or more focuses,
e.g., models that view the object of study for particular characteristics

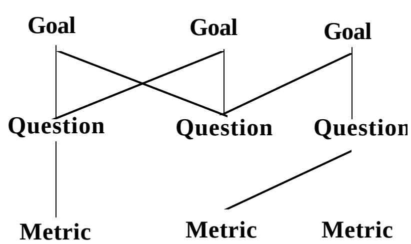
a point of view,
e.g., the perspective of the person needing the information

a purpose,
e.g., how the results will be used

to generate a GQM model

relative to a particular environment

Goal/Question/Metric Paradigm
Goal and Model Based Measurement



A Goal links two models: a model of the object of interest and a model of the focus to develop an integrated GQM model

Goal: Analyze the final product to characterize it with respect to the various defect classes from the point of view of the organization

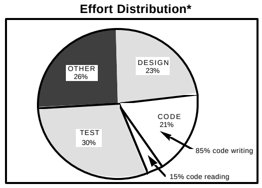
Question: What is the error distribution by phase of entry?

Metric: Number of Requirements Errors, Number of Design Errors, ...

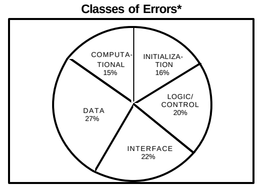
The Goal/Question/Metric Paradigm
Creating Baselines

NASA/SEL PROCESS BASELINE EXAMPLE

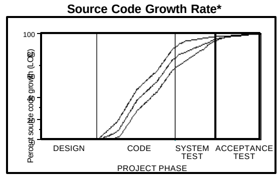
Effort Distribution*



Classes of Errors*



Source Code Growth Rate*



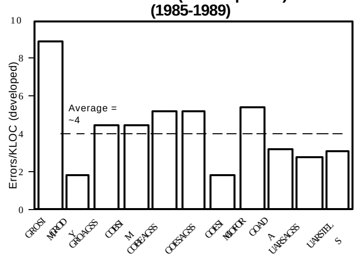
*Data from 11 Flight Dynamics projects (mid 1980s)

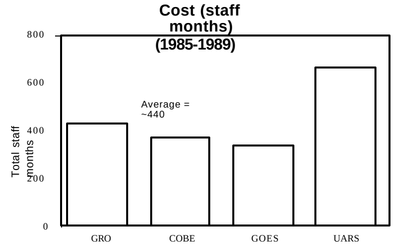
93M11U1.007

The Goal/Question/Metric Paradigm
Creating Baselines

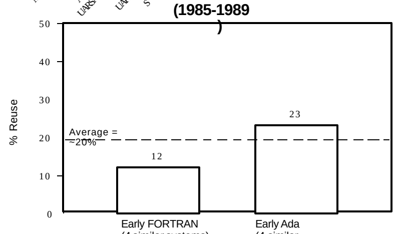
NASA/SEL Product Baseline Example

Error Rates (Development)
(1985-1989)





Reuse
(1985-1989
)



Quality Improvement Paradigm 1981 - 1985

~~Characterize~~/Understand
- Built our own baselines/models of cost, defects, process, etc.

Set Goals
- Set GQM goals to study multiple areas
- Incorporated subjective metrics 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, i.e.,
- Design of experiments is an important part of improvement
- Evaluation and feedback are necessary for learning

Need to experiment with technologies

Need to learn about relationships
- - process, product, and quality models need to be better defined

Reusing experience of all kinds is essential for improvement

Can drown in too much data, especially if you don't have goals and models

...

Developed the QIP as:

Characterize, Set goals, Choose process, Execute, Analyze, and Record

Quality Improvement Paradigm
1986 - 1990

Characterize/Understand

Captured experience in models

Set Goals

Goals and Models commonplace driver of measurement

Built SME, a model-based experience base with dozens of projects

Select Process

Tailored and evolved technologies based on experience

Experimentation and feedback made explicit in the QIP

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

Quality Improvement Paradigm
1986 - 1990
Learned

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

There is a tradeoff between reuse and improvement

Software processes must be put in place to support the reuse of experience

A variety of experiences can be reused,

e.g., process, product, resource, defect and quality models

Experiences can be packaged in a variety of ways,

e.g., equations, histograms, parameterized process definitions

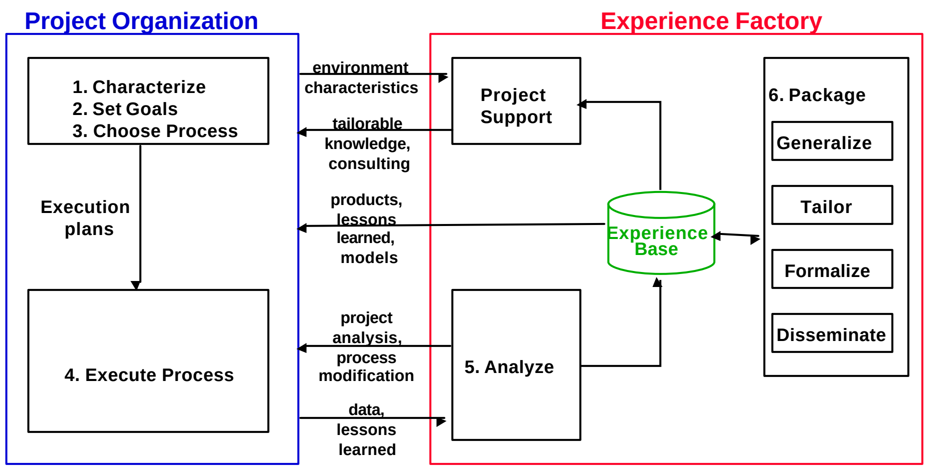
Packaged experiences need to be integrated

...

Evolved GQM, QIP

Formalized organizational structure via the Experience Factory Organization

The Experience Factory Organization

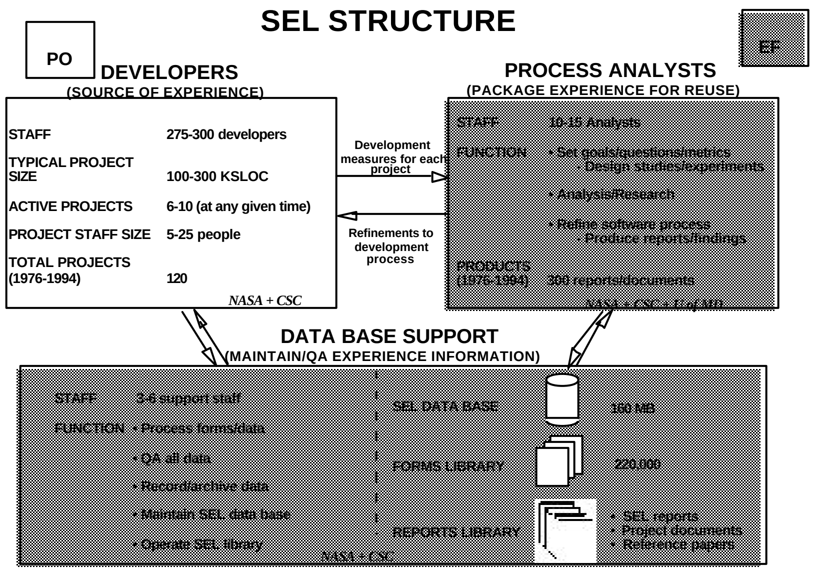


The Experience Factory Organization

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**

An Example Experience Factory



Quality Improvement Paradigm
1991 - 1995

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 for export

Quality Improvement Paradigm
1991 - 1995

Learned

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

Need to provide projects with short term results

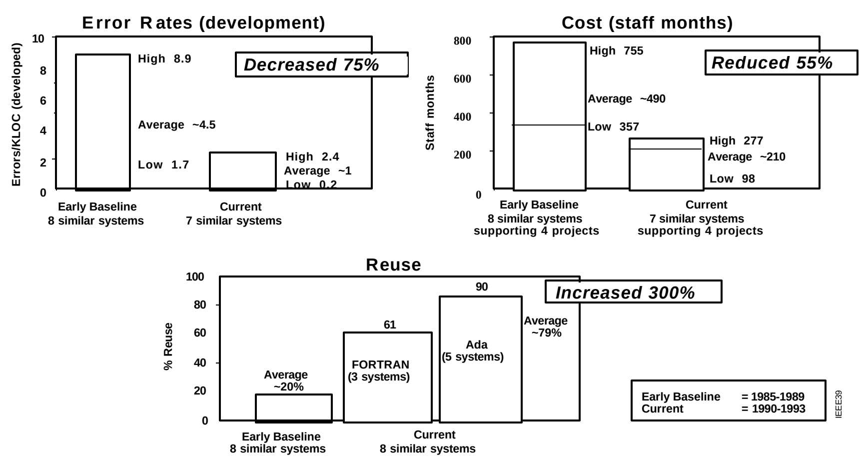
Learning in an organization is time consuming and sequential

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

Quality Improvement Paradigm
1991-1995



The Software Engineering Laboratory was awarded the first
IEEE Computer Society Award for Software Process Achievement in 1994
for demonstrable, sustained, measured, significant process improvement

Effects of the SEL Activities
1996 - 2000

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

Expanding the Learning Organization
Motivation

- 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 lifecycle model for your environment?
- We need to develop an empirically based software development process
  - covering high-level lifecycle models to low-level techniques
  - in which the effects of process decisions are well understood
  - relative to the development context and project goals
- Involves a mix of applied research and technology transfer activities

Example Projects

- **Applied Research Projects**
  - Experience Management System EMS
  - COTS based Development
  - Software Reading Techniques
  - …
- **Technology Transfer Projects**
  - NSF CeBASE Center (UMD,USC, FC-MD, MSU, UNL)
  - High Dependability Computing Consortium Project (NASA Ames, UMD, FC-MD, USC, CMU, MIT, …)
  - SEC (ABB, Boeing, Daimler Chrysler, Motorola, Nokia, FC-MD, FIESE)
  - …

Applied Research
Building an Experience Base

Goals of the EMS project

define, study, and experiment with the concept of automated support for building a learning organization

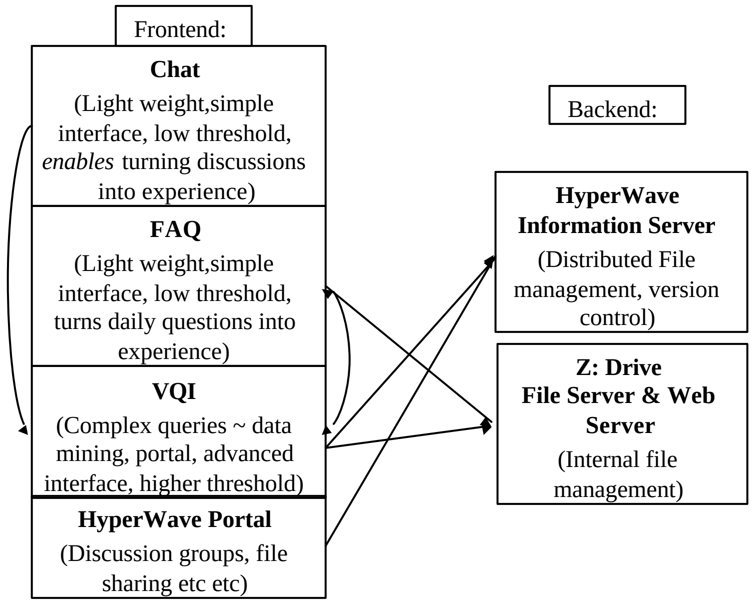
EMS consists of

- different tools, techniques and methodologies for Experience Management
- support for representing various kind of experience

We are building several prototypes for different organizations

- A not-for-profit organization (FC-MD)
- Lessons learned EB (car interior design)
- A software consulting organization
- An EB for implementing CMMI (just started)

Components of the FC-MD EMS



Applied Research
COTS Based System Development

Goal of the CBS project
- support the development of COTS-based software systems

CBS research include
- observation of CBS development at various sights
- development of empirical models
  - COTS evaluation and selection
  - cost estimation (COCOTS)
  - architectural incompatibilities

Current work involves the
- study and evolution CBS development process at NASA
- definition & application of classification schemes for COTS integration
- building of cost estimation/integration models for CBS development

Applied Research
Empirically Based Technique Development

Goals of the project are to develop:
- Families of **techniques** empirically evaluated for context
- **Evaluation approaches** and **criteria** to assess the techniques
- An expanding **Experience Base** of technique evaluations

For example: We have defined an approach to generating a family of reading techniques, called **operational scenarios**, that are
- document and notation specific
- procedurally defined
- goal driven
- tailorable to the project and environment
- focused to provide a particular coverage of the document
- empirically verified to be effective for its use
- usable in existing methods, such as inspections

So far, five techniques have been defined and evaluated in a series of experiments. They analyze **requirements documents, object oriented design, user interface design, and frameworks**

Applied Research
Abstracting across Reading Experiments

We have generated useful **empirical results for 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.
- Procedural inspections, based upon **specific goals**, will find defects related to those goals, so inspections can be customized.
- A set of readers of a software artifact are more effective in uncovering defects when each uses a **different and specific focus**.

Technology Transfer
CeBASE Project

The goal of the Center for empirically-Based Software Engineering (CeBASE) is to accumulate **empirical models** to provide validated guidelines for selecting techniques and models, recommend areas for research, and support education

A first step is to build an empirical **experience base**
- continuously 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
  - **Defect Reduction**, e. g. **reading techniques** (see top ten issues)
  - **COTS Based Development** (see top ten issues)

Examples of Using Empirical Results for development, research, education
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 than use peer reviews.

Implications for **software engineering research**:
- How can peer reviews be improved with better reading techniques for faults of omission and incorrect fact?

Implications for **software engineering education**:
- Teach how to experiment with and choose the appropriate analytic techniques

Examples of Useful Empirical Results
Lifecycle Selection Guidance

Lifecycle 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)

Technology Transfer
High Dependability Computing Project

The goal of the HDC Project is to develop high dependability technologies, study their effectiveness under varying conditions, and transfer them into practice

Approach: View each technology passing through a *series* of testbeds.

Testbeds are used to

- stress the technology and demonstrate its context of effectiveness
- help the researcher identify the strengths, bounds, and limits of the particular technology at different levels
- provide insights into the models of dependability
-

Goals are defined as measurable (GQM)

- help define models of dependability
- establish criteria for each technology
- identify testbed characteristics and vary with the testbed level
- represent the effectiveness of the collection of the technologies

Technology Transfer
Software Experience Center

The goal of the Software Experience Center (SEC) is to leverage the experience of several leading software competency companies by sharing their experiences in software process improvement

The approach is to

- Run Workshops (2x a year)
- Develop Experience Packages in various forms
  - "Real-time" workshop packages of presentation and discussion materials
  - Post-workshop packages of application-level methods, processes for use by SEC user community (tech-transfer to business units)
- Support On-line interaction
  - Cooperative workspace for use by SEC personnel and their companies
  - Open access to selected topic areas

25 Years of Learning
Conclusion

Improvement of software competence is an essential business need

The software engineering discipline needs to

- build software core competencies as part of overall business strategy
- create organizations for continuous learning to improve software competence
- generate a tangible corporate asset: an experience base of competencies
- build an empirically-based, tailorable software development process

QIP/GQM/EF represents a Lean Software Development concept and a CMM level 5 organizational structure

Since 1976 have learned a great deal about software improvement

- Learning process has been continuous and evolutionary
- Supported by the symbiotic relationship between research and practice
- Packaged what's been learned into our process, product and structure