Software Improvement
Feedback Loops:
The SEL Perspective

Victor R. Basili

Institute for Advanced Computer Studies
Department of Computer Science
University of Maryland
and
Fraunhofer Center - Maryland
What is 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
Since 1976 we have learned a great deal, e.g.,
understand before you assess
data should be goal and model driven
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

This talk offers a retrospective and a look at our directions
Observation, Feedback, Learning, Packaging

The Quality Improvement Paradigm

The SEL
  1976 -1980 (Goal Question Metric Paradigm)
  1981 - 1985 (Baselining and Experimenting)
  1986 - 1990 (Experience Factory Organization)
  1991 - 1995 (Effects)

The SEL and Fraunhofer Center for Experimental Software Engineering
  1996-present
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

Set goals

Choose processes, methods, techniques, and tools

Analyze results

Package & store experience

Characterize & understand

Choose process, methods, techniques, and tools

Corporate learning

Project learning

Provide process with feedback

Execute process

Analyse results
Maturing the Improvement Paradigm
Major Activity Evolution

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
loosely monitored ----> closely monitored/feedback

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 and was loosely monitored

Analyze Data Only
Mostly build 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
- 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

...
### Quality Improvement Paradigm 1976 - 1980

#### Trying to Apply the 40/20/40 Rule in SEL

<table>
<thead>
<tr>
<th></th>
<th>TRW</th>
<th>IBM</th>
<th>SEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase</td>
<td>Activity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design</td>
<td>40%</td>
<td>35%</td>
<td>20%</td>
</tr>
<tr>
<td>Code</td>
<td>20</td>
<td>30</td>
<td>45</td>
</tr>
<tr>
<td>Checkout/Test</td>
<td>40</td>
<td>25</td>
<td>28</td>
</tr>
<tr>
<td>Other</td>
<td>10</td>
<td>5</td>
<td>27</td>
</tr>
</tbody>
</table>

*The 40/20/40 rule does not apply to us*

*The rule does not imply what you may think*
Quality Improvement Paradigm
1976 - 1980

Applying a Resource Allocation Model

Need to understand the local context
Local context makes a big difference
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**,  
ed.g., a process, product, or any other experience model

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

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

a **purpose**,  
ed.g., how the results will be used

to generate a **GQM model**

relative to a **particular environment**
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*

- DESIGN 23%
- CODE 21%
- TEST 30%
- OTHER 26%

Classes of Errors*

- COMPUTATIONAL 15%
- DATA 27%
- INTERFACE 22%
- LOGIC/CONTROL 20%
- INITIALIZATION 16%

Source Code Growth Rate*

- Percent source code effort vs. LOC
  - DESIGN
  - CODE
  - SYSTEM TEST
  - ACCEPTANCE TEST

*Data from 11 Flight Dynamics projects (mid 1980s)
The Goal/Question/Metric Paradigm
Creating Baselines

NASA/SEL  Product  Baseline Example

Error Rates (Development)  (1985-1989)

- Average = ~4

Cost (staff months)  (1985-1989)

- Average = ~440

Reuse  (1985-1989)

- Average = ~20%

- 12: Early FORTRAN (4 similar systems)

- 23: Early Ada (4 similar systems)
Characterize/Understand
   Built our own baselines/models of cost, defects, process, etc.

Set Goals
   Set GQM goals to study multiple areas
   Incorporated subjective metrics

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** in the form of processes, products, and other forms of knowledge is essential for improvement

Can drown in **too much data**, especially if you don’t have goals

...  

**Developed the QIP** as:
Characterize, Set goals, Choose process, Execute, Analyze, and **Record**
We need to understand the relationship among variables.
The relationship between fault rate and size is non-linear.
Quality Improvement Paradigm
1986 - 1990

Characterize/Understand
Capturing 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
Learned

**Quality Improvement Paradigm**

*1986 - 1990*

**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**

..."Reformulated QIP as:

  Characterize, Set goals, Choose process, Execute, Analyze, and **Package**

Evolved GQM to include templates and models

Formalized the organization via the Experience Factory Organization
Evaluating and Integrating Reading

Testing vs. Reading experiment
  Reading more effective and efficient than testing

Reading in Practice
  Reading had little effect

Reading as part of Cleanroom at the University
  Reading had a high impact

Reading as part of Cleanroom in the SEL
  Reading had a high impact

*How a technology is packaged and integrated has a strong effect*
*Reading more effective when not followed by testing*
1. Characterize
2. Set Goals
3. Choose Process
4. Execute Process

Execution plans

Experience Factory

Project Support

Experience Base

5. Analyze

products, lessons learned, models
project analysis, process modification
data, lessons learned

environment characteristics
tailorable knowledge, consulting

6. Package

Generalize
Tailor
Formalize
Disseminate
Experience Factory Organization

A Different Paradigm

<table>
<thead>
<tr>
<th>Project Organization</th>
<th>Experience Factory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem Solving</td>
<td>Experience Packaging</td>
</tr>
</tbody>
</table>

Decomposition of a problem into simpler ones

Instantiation

Design/Implementation process

Validation and Verification

Product Delivery within Schedule and Cost

Unification of different solutions and re-definition of the problem

Generalization, Formalization

Analysis/Synthesis process

Experimentation

Experience / Recommendations Delivery to Project
An Example Experience Factory

**SEL STRUCTURE**

**DEVELOPERS**
- **STAFF**: 275-300 developers
- **TYPICAL PROJECT SIZE**: 100-300 KSLOC
- **ACTIVE PROJECTS**: 6-10 (at any given time)
- **PROJECT STAFF SIZE**: 5-25 people
- **TOTAL PROJECTS (1976-1994)**: 120

**DATA BASE SUPPORT**
- **STAFF**: 3-6 support staff
- **FUNCTION**:
  - Process forms/data
  - QA all data
  - Record/archive data
  - Maintain SEL data base
  - Operate SEL library

**PROCESS ANALYSTS**
- **STAFF**: 10-15 Analysts
- **FUNCTION**:
  - Set goals/questions/metrics
  - Design studies/experiments
  - Analysis/Research
  - Refine software process
  - Produce reports/findings

**PRODUCTS**
- **(1976-1994)**: 300 reports/documents

**SEL DATA BASE**
- **160 MB**
- **FORMS LIBRARY**
  - 220,000
- **REPORTS LIBRARY**
  - SEL reports
  - Project documents
  - Reference papers

**STAFF 275-300 developers**

**TYPICAL PROJECT SIZE**: 100-300 KSLOC

**ACTIVE PROJECTS**: 6-10 (at any given time)

**PROJECT STAFF SIZE**: 5-25 people

**TOTAL PROJECTS (1976-1994)**: 120

**STAFF 10-15 Analysts**

**FUNCTION**

**PRODUCTS**
- **(1976-1994)**: 300 reports/documents

**SEL DATA BASE**
- **160 MB**
- **FORMS LIBRARY**
  - 220,000
- **REPORTS LIBRARY**
  - SEL reports
  - Project documents
  - Reference papers

**NASA + CSC**
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 framework for flight dynamics software

Execute Process
Captured the details of experience - more interaction between developers and experimenters - more effective feedback

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

Package
Studied what was exportable
Evolved and packaged the Experience Factory Organization
Quality Improvement Paradigm
1991 - 1995

Learned

Learning in an organization is time consuming and sequential

Need to provide projects with short term results

Need to find ways to speed up the learning process

Need to feed interim results back into the project faster

Need to better understand the criteria for sharing best practices

Need to better package the meta-models, e.g., Experience Factory

Formulated:
Concepts for building bodies of SE knowledge
Experience Factory Methods
Requirements reading techniques
Quality Improvement Paradigm 1991-1995

Error Rates (development)

- Early Baseline: 8 similar systems
  - High: 8.9
  - Average: ~4.5
  - Low: 1.7

- Current: 7 similar systems
  - High: 2.4
  - Average: ~1
  - Low: 0.2

Error Rates Decreased 75%

Cost (staff months)

- Early Baseline: 8 similar systems supporting 4 projects
  - High: 755
  - Average: ~490
  - Low: 357

- Current: 7 similar systems supporting 4 projects
  - High: 277
  - Average: ~210
  - Low: 98

Cost Reduced 55%

Reuse

- Early Baseline: 8 similar systems
  - FORTRAN (3 systems): 61
  - Average: ~20%

- Current: 8 similar systems
  - Ada (5 systems): 90
  - Average: ~79%

Reuse Increased 300%

Early Baseline = 1985-1989
Current = 1990-1993
Quality Improvement Paradigm
An Experience Factory Example

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
was created in Maryland in 1998
SEL Studies
Information Systems Center at NASA

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**
  – Part of Fraunhofer USA, a U.S., not-for-profit affiliate of Fraunhofer Gesellschaft

• **Fraunhofer Society**
  – Fraunhofer Gesellschaft: Joint government - corporate sponsored applied research and technology transfer organization in Germany; Started in 1949
    • 48 institutes, 9000 employees, $800 million annual budget
    • Covers many industrial technologies: computers, electronics, materials, chemicals, production, optics, biomedical, …
    • Each institute focuses on a specific technology
  
  – Fraunhofer USA: Started in 1994
    • Each Fraunhofer Center affiliated with one German Fraunhofer institute
    • Centers in Michigan, Delaware, Illinois, Massachusetts, Maryland
    • College Park center affiliated with Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern, Germany
Status of Fraunhofer Center - Maryland

• **Background**
  – Preliminary operations began November, 1997
  – Official opening held February, 1998
  – Moved April, 2000, to location adjacent to University of Maryland campus
  – Center Directors are Victor Basili and Marvin Zelkowitz (University of Maryland)

• **Size (July 2000)**
  – 10 permanent Fraunhofer employees
  – 10 part-time faculty/visitors/students
  – Growing in staff to meet business needs
  – Going from $1.1 million in 1999 revenues to $1.6 million in 2000 revenues
  – Roughly tripled physical space - have expansion room for additional hires
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**
  – Experience Management System (FC-MD/UMD)
  – Reading Techniques for Improving Software Inspections (FC-MD/UMD)

• **Problem Study and Direct Services Projects**
  – NASA IV&V ROI Project (government)
  – Small Business Learning Organization/CMM Project (FC-MD/IESE)
  – NASA/Software Engineering Laboratory (government)
  – Experience Factory Support (industry)
  – Motorola Testing (industry)
  – Partnering with Companies on Proposals (industry)
  – Software Acquisition Support (FC-MD, IESE, industry)
Maryland Software Industry Consortium (SwIC)

- **Vision**: Increase the business growth and competitive edge of Maryland companies by improving the quality of their software- and IT-related business aspects.

- **Description**: The consortia consists of a group of Maryland-based organizations:
  - small to mid-size companies - for state matching funds
  - large organizations - for mentoring and improved supplier base
  - established by the state and coordinated by a FC-MD Project Director

- **Benefits to member companies**:
  - integrating research and experience into practical improvement (business results)
  - accelerating new software technology adoption
  - leveraging member company experience
  - promoting inter-corporate cooperation of member organizations
  - providing education and training

- **Status**:
  - Began Software Process Improvement (SPI) Program Group 1 with 15 attendees from 6 companies (Southern Maryland)
  - Membership renewals and new member solicitations underway
**Software Experience Center Project**

- **Vision:** A consortium of multi-national companies optimizing the learning process in the software engineering domain by sharing and packaging state of the art knowledge in areas of interest and importance to them.

- **Description:** The Fraunhofer organizations (IESE and FC-MD)
  - organize and run two workshops a year for the member companies
  - package sets of experience based upon member interests
  - build an evolving experience base of evaluated experience

- **Benefits to member companies**
  - Learn, tailor, implement, and evolve best practices
  - Maximize investment in best practices to support their business goals
  - Institutionalize improvement practices as part of their corporate culture

- **Status:**
  - Current Membership: Daimler-Chrysler, Motorola, ABB, Nokia, Boeing
  - Next biannual workshop planned for November 2000, hosted by Nokia
Experience Management System

• **Vision:** Develop and validate a knowledge management system (consisting of organizational practices, user interface, and repository technology) required to support an operational Experience Factory

• **Description:**
  – collaboration of experts in user interface design and database technology, in addition to software engineering
  – deliver organizational, user interface, and repository infrastructure
  – support the Software Experience Center consortium
  – build assets for future Experience Factory work

• **Benefits to companies**
  – provide intelligent access to a company’s electronic assets
  – manage and link diverse, distributed sources of business knowledge

• **Status:**
  – have a first prototype system built for a distributed consulting company
  – building system for FC-MD operational use (Fall 2000 availability)
Reading Techniques for Improved Software Inspections

• **Vision:** Inspections are among the most effective software quality practices. FC-MD offers reading techniques that provide customized, procedural support for helping businesses improve inspections, especially in earlier stages of software development.

• **Description:** FC-MD scientists will integrate customized reading technique procedures with a company’s existing practices to improve their effectiveness. FC-MD will tailor the procedures to the skill set of the organization’s developers and help the organization understand the most important types of problems to target. FC-MD will monitor the effectiveness of inspections in the organization to provide further tailoring.

• **Benefits to companies**
  – Improve existing inspection and review methods
  – Improve software quality

• **Status:**
  – Requirements and OO reading work with NASA/GSFC/JP, Telcordia, ...
  – Tutorial presented at ICSE; scheduled for NASA Workshop
  – Empirical studies continuing with our university partners
Maturing the Improvement Paradigm

Conclusion

Since 1976 we have learned a great deal about software improvement.

Our learning process has been continuous and evolutionary like the evolution of the software development process itself.

We have packaged what we have learned into our process, product and organizational structure.

The evolution is supported by the symbiotic relationship between research and practice.

It is a relationship that requires patience and understanding on both sides, but when nurtured, really pays dividends!
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 **level 5 organizational structure**