Process Improvements for Software Quality and Reliability

Victor R. Basili

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

OUTLINE

The Software Business

Measurement:

The Goal Question Metric Paradigm

Process Improvement:

The Quality Improvement Paradigm

Evolutionary Learning:

The Experience Factory

An Example Experience Factory

THE SOFTWARE BUSINESS
Business Requirements

Any successful business requires:

- combination of **technical and managerial** solutions
- well-defined set of **product needs**
- well-defined set of **processes**
- **closed loop processes** that support project control, learning and improvement

Key technologies for supporting these needs include:

modeling, measurement, reuse

of processes, products, and other knowledge relevant to the business

THE SOFTWARE BUSINESS
Implications for the Software Business

Understand the process and product

Define process and product qualities

Evaluate successes and failures

Feedback information for project control

Learn from our experiences

Package successful experiences and core competencies

Use those experiences and core competencies

THE SOFTWARE BUSINESS
The Nature of Software

Learning in the software discipline is **evolutionary** and **experimental**

Software is **development** (design) not production

Software technologies are **human based**

There is a **lack of models** for reasoning about the process and product

All software is not the same; processes, goals are variable

Packaged, reusable, experiences require a **additional resources** in the form of organization, processes, people, etc.

Software is **difficult**

THE SOFTWARE BUSINESS
Software Quality Needs

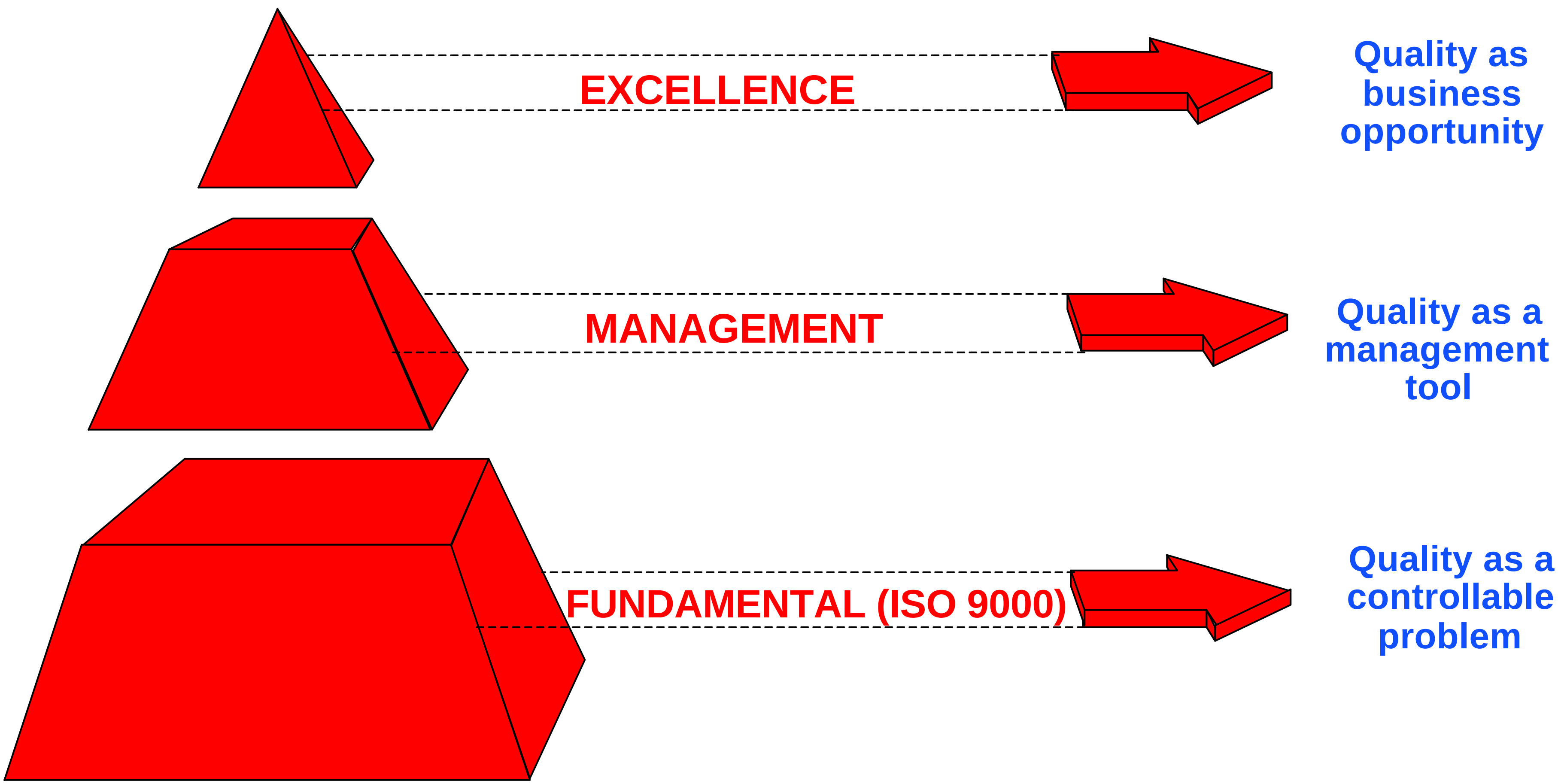
Quality Definition: Define qualities and quality goals operationally relative to the project and the organization

Process Selection: Find criteria for selecting the appropriate methods and tools and tailoring them to the needs of the project and the organization

Quality Evaluation: Evaluate the quality of the process and product relative to the specific project and organizational goals

Quality Organization: Organize quality assurance from planning through execution through evaluation, feedback and improvement

THE SOFTWARE BUSINESS
The Pyramid of Quality



Towards Software Quality Improvement

The following concepts have been developed and evolved based upon experience in a number of organizations

A paradigm for establishing project and corporate goals and a mechanism for measuring against those goals

Goal/Question/Metric Paradigm

An evolutionary improvement paradigm tailored for the software business

Quality Improvement Paradigm

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

Experience Factory

SOFTWARE MEASUREMENT

What can we do with measurement?

Create a corporate memory - baselines/models of current practices
e.g., how much will a new project cost?

Determine strengths and weaknesses of the current process and product
e.g., are certain types of errors commonplace?

Develop a **rationale** for adopting/refining techniques
e.g., what techniques will minimize the problems, change the baselines?

Assess the **impact** of techniques
e.g., does functional testing minimize certain error classes?

Evaluate the **quality** of the process/product
e.g., are we applying inspections appropriately?
what is the reliability of the product after delivery?

SOFTWARE MEASUREMENT

Measurement is not just the collection of data/metrics

calendar time

number of open problems

number of defects found in inspections

cyclomatic complexity

machine time

lines of code/module

total lines of code

severity of failures

total effort

total number of defects

lines of code/staff month

number of failures during system test

SOFTWARE MEASUREMENT

We need a measurement framework to

Characterize

Describe and differentiate software processes and products

Build descriptive models and baselines

Understand

Explain associations/dependencies between processes and products

Discover causal relationships

Analyze models

Evaluate

Assess the achievement of quality goals

Assess the impact of technology on products

Compare models

Predict

Estimate expected product quality and process resource consumption

Build predictive models

Motivate

Describe what we need to do to control and manage software

Build prescriptive models

MEASUREMENT FRAMEWORKS

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

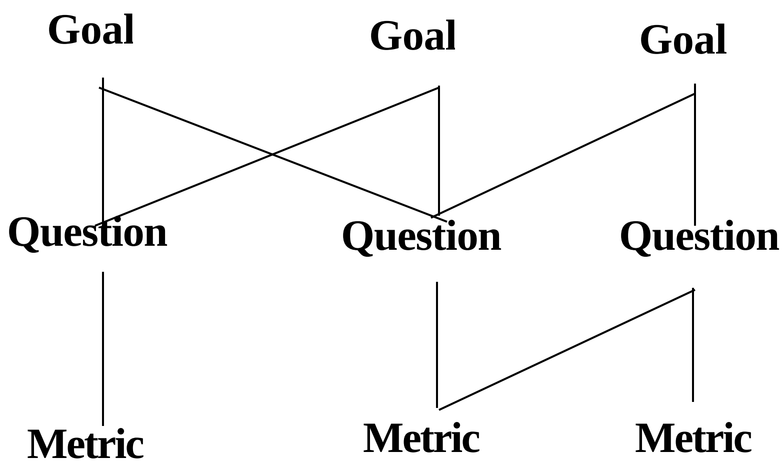
a **point of view,**
e.g., the perspective of the person needing the information

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 and develops 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, ...

GOAL/QUESTION/METRIC PARADIGM
Overview of the GQM Approach

Develop a set of corporate, division and project **goals** for productivity and quality, e.g., customer satisfaction, on time delivery, improved quality, developing reusable objects, reusing experiences

Generate questions (based upon models) that define those goals as completely as possible in a quantifiable way.

Specify the **measures** needed to be collected to answer those questions and track process and product conformance to the goals.

Develop mechanisms for data collection.

Collect, validate and **analyze** the **data** in real time to provide feedback to projects for corrective action.

Analyze the **data** in a postmortem fashion to assess conformance to the goals and make recommendations for future improvements.

GOAL/QUESTION/METRIC PARADIGM
Characterizing Goals

Analyze the **software products**

in order to **characterize** them with respect to

development error rates

cost in staff months

% of code reused

from the point of view of the **organization** relative to the **SEL environment**

Analyze the **software processes**

in order to **characterize** them with respect to

effort distributions

classes of errors

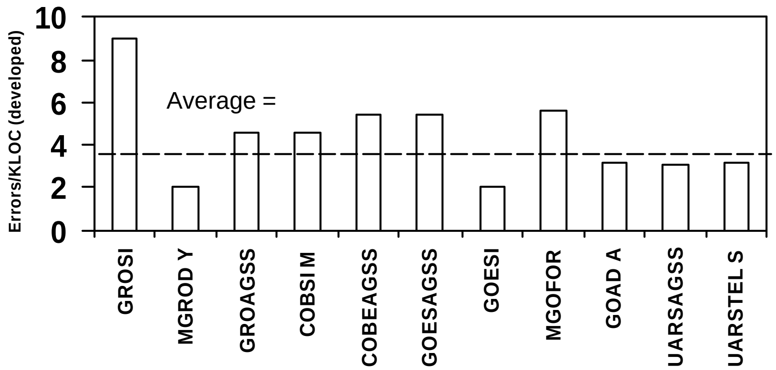
source code growth

from the point of view of the **organization** relative to the **SEL environment**

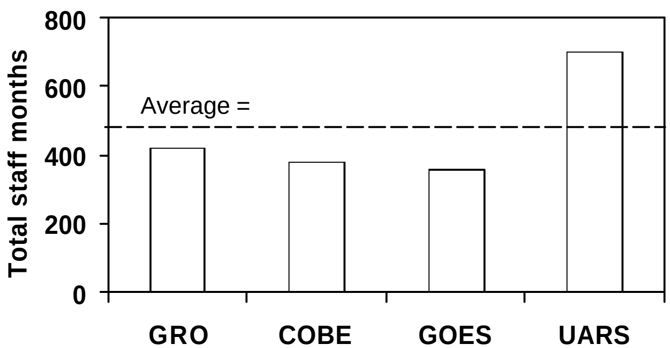
THE EXPERIENCE FACTORY ORGANIZATION

NASA/SEL PRODUCT BASELINE EXAMPLE

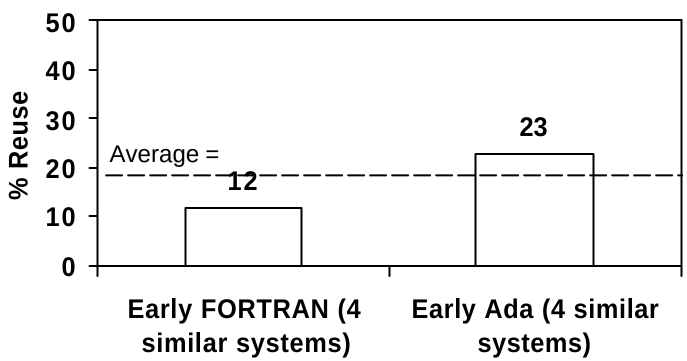
Error Rates (Development)
(1985-1989)



Cost (staff months)



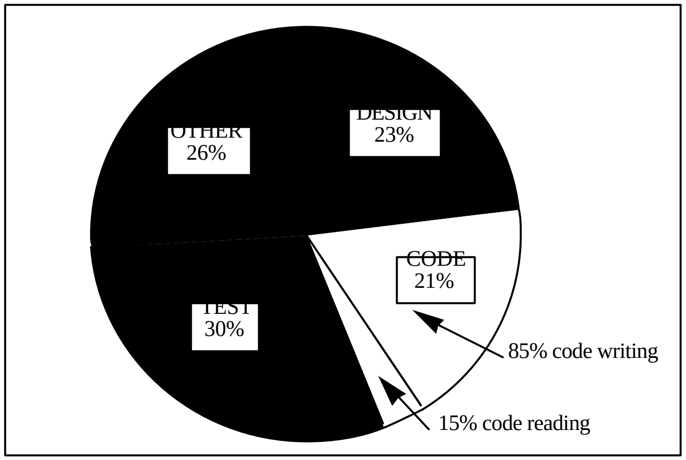
Reuse (1985-1989)



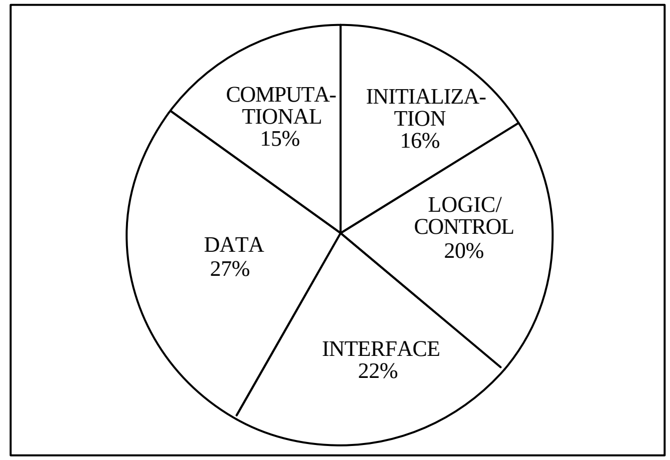
GOAL/QUESTION/METRIC PARADIGM

NASA/SEL PROCESS BASELINE EXAMPLE

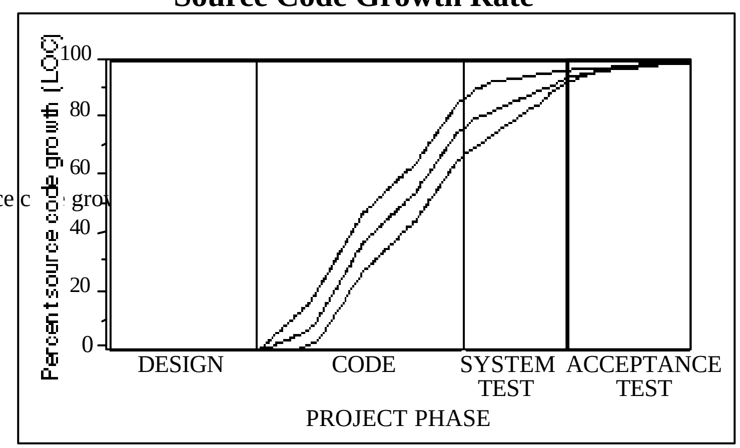
Effort Distribution*



Classes of Errors*



Source Code Growth Rate*



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

GOAL/QUESTION/METRIC PARADIGM
Process Goal: Question Guidelines

Process Conformance:

Characterize the process quantitatively and assess how well the process is performed.

Domain Understanding:

Characterize the object of the process and evaluate the knowledge of the object and its domain by the process performers.

Focus:

Analyze the output of the process according to some quality model and some viewpoint.

Feedback:

What has been learned about the process, its application, the product domain, or any other process or product?

GOAL/QUESTION/METRIC PARADIGM
Process Goal: Example

Analyze the system test process for the purpose of evaluation with respect to defect slippage from the point of view of the corporation.

System Test Process Model:

Goal: Generate a set of tests consistent with the complexity and importance of each requirement.

Procedure: (1) Enumerate the requirements, (2) Rate importance by marketing, (3) Rate complexity by system tester, (4) ...

Defect Slippage Model:

Let **Fc** = the ratio of faults found in system test to the faults found after system test on this project.

Let **Fs** = the ratio of faults found in system test to the faults found after system test in the set of projects used as a basis for comparison.

Let **QF = Fc/Fs =** the relationship of system test on this project to faults as compared to the average the appropriate basis set.

GOAL/QUESTION/METRIC PARADIGM

Simple Interpretation of Defect Slippage Model

if QF > 1 then

method better than history

check process conformance

if process conformance poor

improve process or process conformance

check domain understanding

if domain conformance poor

improve object or domain training

if QF = 1 then

method equivalent to history

if cost lower than normal then method cost effective

check process conformance

if QF < 1 then

check process conformance

if process conformance good

check domain conformance

if domain conformance good

method poor for this class of project

GOAL/QUESTION/METRIC PARADIGM
Product Goal: Question Guidelines

Product Model/Definition:

Characterize the product qualitatively independent of the perspective of interest. Aspects of interest include:

Logical/Physical Attributes:

Characterize the logical and physical attributes of the product e.g.,

logical attributes: application domain, function

physical attributes: size, complexity, interfaces

dynamic attributes: coverage, reliability

Cost:

Characterize the resources expended, e.g., effort, computer time

Changes:

Characterize the modifications associated with the product, e.g., enhancements, errors, faults, failure

Context:

Characterize the customer community and their operational profiles

GOAL/QUESTION/METRIC PARADIGM
Product Goal: Question Guidelines

Perspective/Focus:

Analyze the product models from each perspective of interest, e.g., reliability, user friendliness, specifying the following:

Major model(s) used

Specify some perspective model/definition and viewpoint

Validity of the model for the project

Evaluate the appropriateness of the model for the project environment

Validity of the data collected

Evaluate the quality of the data

{**Substantiation of the model**

Given any alternate perspectives that provide support for the quality of the results}

Feedback:

What has been learned about the product, the processes that produced it, or any other product that will improve this project and future projects?

GOAL/QUESTION/METRIC PARADIGM
Product Goal Example

Analyze the design document for the purpose of evaluation with respect to the design inspection defects uncovered from the point of view of the project manager.

Design Inspection Process Model:

Goal: Analyze the design document for the purpose of characterization with respect to its correct and complete implementation of the requirements from the point of views of the user, developer and tester.

Procedure:

(1) Disseminate the appropriate part of the requirements and design documents,

(2) Read the document by the appropriate set of readers from the appropriate points of view,

(3) Report defects by various classification schemes, including omission and commission defects,

(4) ...

GOAL/QUESTION/METRIC PARADIGM
Product Goal Example

Design Document Product Model/Definition:

Logical/Physical Attributes:

logical attributes: application domain, function

physical attributes: size: **lines of design language**, complexity, interfaces

Cost:

total effort, effort by activity (effort in design inspection)

Changes:

# of enhancements

 # faults found during design inspection

Context:

Customer community: designers, coders, users, ...

GOAL/QUESTION/METRIC PARADIGM

Simple Document/Defect Evaluation Model

KLOD = number of thousand lines of design language

Fc = number of faults/**KLOD** found in design inspections on this project

Fs = number of faults/**KLOD** found in design inspections in the set of projects used as a basis for comparison (same size, application, …)

QF = Fc/Fs = the relationship of faults found on this project as compared to the average the appropriate basis set

if **QF** > 1 then QF = H (worse than history)

if **QF** <= 1 then QF = L (better than history)

PC = the process conformance rating on this project
= C if inspections are performed to the definition, N otherwise

DU = the domain understanding rating on this project
= S if domain understanding is satisfactory, U otherwise

GOAL/QUESTION/METRIC PARADIGM

Simple Document/Defect Evaluation Model

QF = H if more faults found when compared with history
QF = L if less faults found when compared with history

PC = C if inspections are performed to the definition
N otherwise

DU = S if domain understanding is satisfactory
U otherwise

PC	DU	QF	Design-in	Design-out	Design Process	Inspection Process
C	S	L	good	good	effective	effective
C	S	H	poor	fixed-up	not-effective	effective
N	X	X	?	?	?	?
X	U	X	?	?	?	?

EXAMPLE G/Q/M GOALS

Defining the System Test Process Goal:

Analyze the software product requirements for the purpose of characterizing them with respect to a set of tests consistent with the complexity and importance of each requirement from the point of view of the tester and marketer respectively.

Evaluating the System Test Process:

Analyze the system test process for the purpose of evaluation with respect to defect slippage from the point of view of the corporation.

Defining the Design Inspection Process Goal:

Analyze the design document for the purpose of characterization with respect to its correct and complete implementation of the requirements from the point of views of the user, developer, and tester.

Evaluating the Design Document:

Analyze the design document for the purpose of evaluation with respect to the design inspection defects uncovered from the point of view of the project manager.

Organizational Frameworks

Quality Improvement Paradigm

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

Set quantifiable **goals** for successful project performance and improvement.

Choose the appropriate **process** model and supporting methods and tools for this project.

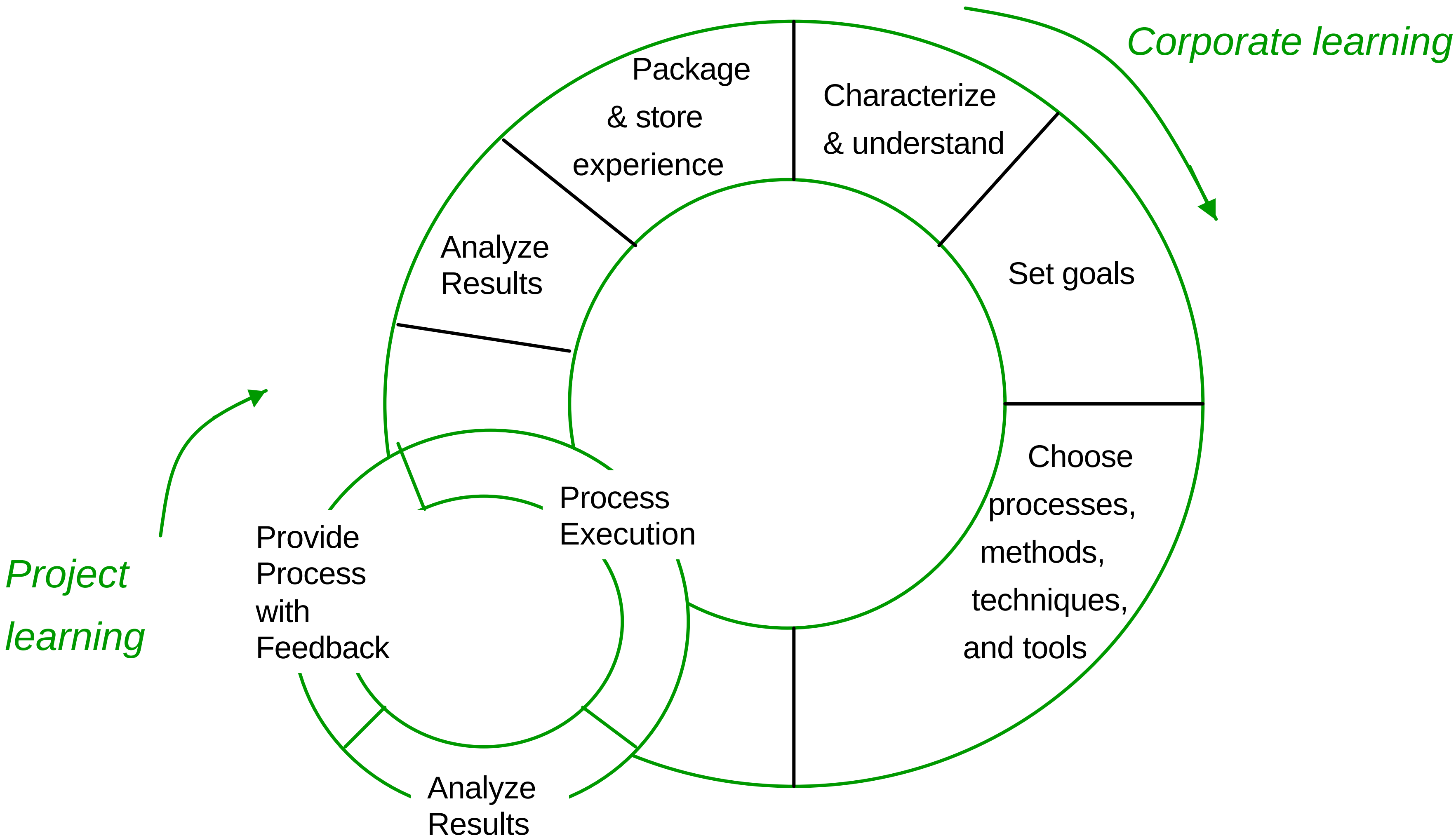
Execute the **processes**, construct the products, collect,validate, and analyze the data to provide real-time feedback for corrective action.

Analyze the **data** to evaluate the current practices, determine problems, record findings, and make recommendations for future project improvements.

Package the **experience** in the form of updated and refined models and other forms of structured knowledge gained from this and prior projects and save it in an experience base to be reused on future projects.

Approaches To Quality

Quality Improvement Paradigm



Quality Improvement Paradigm
Step 1: Characterizing the Project and Environment

Build models to

help us understand what we are doing
provide a basis for defining goals
provide a basis for measurement

Build models of

people, processes, products
and study their interactions

Use models to

classify the current project
distinguish the relevant project environment
find the class of projects with similar characteristics and goals

Models provides a context for

Goal Definition
Reusable Experience/Objects
Process Selection
Evaluation/Comparison
Prediction

Characterization

Project Characteristics and Environmental Factors

People Factors: number of people, level of expertise, group organization, problem experience, process experience,...

Problem Factors: application domain, newness to state of the art, susceptibility to change, problem constraints, ...

Process Factors: life cycle model, methods, techniques, tools, programming language, other notations, ...

Product Factors: deliverables, system size, required qualities, e.g., reliability, portability, ...

Resource Factors: target and development machines, calendar time, budget, existing software, ...

Quality Improvement Paradigm
Step 2: Goal Setting and Measurement

Need to **establish goals** for the processes and products

Goals should be **measurable,** driven by the **models**

Goals should be defined from a **variety of perspectives:**

Customer:	predictable schedule, correct functionality
Project:	quality controllable process, adherence to schedule
Corporation:	reusable experiences, improved quality/productivity over time

There are a variety of mechanisms for defining measurable goals:

- Goal/Question/Metric Paradigm **(GQM)**
- Software Quality Metrics Approach **(SQM)**
- Quality Function Deployment Approach **(QFD)**

Quality Improvement Paradigm
Step 3: Choosing the Processes

We need to **choose** and **tailor** an appropriate generic process model, integrated set of methods, and integrated set of techniques

We need to **define their goals** and give its definitions (models)

Choosing and tailoring are always done **in the context of the environment,** project characteristics, and goals established for the products and other processes

Examples:

If problem and solution well understood

choose **waterfall process model**

If high number of faults of omission expected

emphasize **traceability reading** approach embedded in **design inspections**

When embedding **traceability reading in design inspections,** make sure **traceability matrix** exists

Choose The Process

Choosing the Technique: Reading

Input object: Requirements, specification, design, code, test plan,...

Output object: set of anomalies

Approach: Sequential, path analysis, stepwise abstraction, ...

Formality: Reading, correctness demonstrations, ...

Emphasis: Fault detection, traceability, performance, ...

Method: Walk-throughs, inspections, reviews, ...

Consumers: User, designer, tester, maintainer, ...

Product qualities: Correctness, reliability, efficiency, portability,..

Process qualities: Adherence to method, integration into process,...

Quality view: Assurance, control, ...

Choose The Process

Choosing the Technique: Testing

Input object: System, subsystem, feature, module,..

Output object: Test results

Approach: structural, functional, error-based, statistical testing,..

Formality: Full adherence, partial adherence, ...

Emphasis: Fault detection, new features, reliability, performance,..

Method: As specified in the test plan

Consumers: Various classes of customer/hardware configurations,

Product qualities: Reliability, efficiency, ...

Process qualities: Adherence to method, integration into process,...

Quality view: Assurance, control

Quality Improvement Paradigm
Step 4: Executing the Processes

The development process must **support** the access and reuse of packaged experience

Data items must be **defined** by the models and driven the by the goals

Data collection must be **integrated** into the processes, not an add on, e.g., defect classification forms part of configuration control mechanism

Data validation important and necessary. e.g., defect data is error prone

Education and training in data collection are necessary, everyone must understand the models

Some analysis must be done in close to **real time** for **feedback** for corrective action

The **suppliers** of the data **need to gain** from the data too

Automated support is necessary to:

- support mechanical tasks
- deal with large amounts of data and information needed for analysis

however, the collection of the most interesting data cannot be automated

Executing The Processes
Kinds of Data Collected

Resource Data:

Effort by activity, phase, type of personnel
Computer time
Calendar time

Change/Defect Data:

Changes and defects by various classification schemes

Process Data:

Process definition
Process conformance
Domain understanding

Product Data:

Product characteristics
- logical, e.g., application domain, function
- physical, e.g. size, structure
- dynamic, e.g., reliability, coverage

Use and context information, e.g., design method used

Quality Improvement Paradigm
Step 5: Analyzing the Data

Based upon the goals, we interpret the data that has been collected.

We can use this data to:

characterize and understand, e.g.,

- what project characteristics effect the choice of processes, methods and techniques?
- which phase is typically the greatest source of errors?

evaluate and analyze, e.g.

- what is the statement coverage of the acceptance test plan?
- does the Cleanroom Process reduce the rework effort?

predict and control, e.g.,

- given a set of project characteristics, what is the expected cost and reliability, based upon our history?

motivate and improve, e.g.,

- for what classes of errors is a particular technique most effective

Quality Improvement Paradigm
Step 6: Packaging the Experience

Resource Models and Baselines,

e.g., local cost models, resource allocation models

Change and Defect Baselines and Models,

e.g., defect prediction models, types of defects expected for application

Product Models and Baselines,

e.g., actual vs. expected product size and library access over time

Process Definitions and Models,

e.g., process models for Cleanroom, Ada

Method and Technique Evaluations,

e.g., best method for finding interface faults

Products, e.g., Ada generics for simulation of satellite orbits

Quality Models,

e.g., reliability models, defect slippage models, ease of change models

Lessons Learned, e.g., risks associated with an Ada development

Packaging Experience
Forms of Packaged Experience

Equations defining the relationship between variables,
e.g. Effort = $1.48 * KSLOC^{.98}$, Number of Runs = 108 + 150*KSLOC

Histograms or **pie charts** of raw or analyzed data,
e.g., Classes of Faults: 30% data, 24% interface, 16% control, 15% initialization, 15% computation
Effort Distribution: 23% design, 21% code, 30%test, 26% other

Graphs defining ranges of “normal”
e.g., Fault Slippage Rate: halve faults after each test phase (4,2,1,.5)

Specific lessons learned, e.g.,
an Ada design should use library units rather than a deeply nested structure
minimize the use of tasking as its payoff is minimal in this environment
size varies inversely with defect rate up to about 1KLOC per module

Processes descriptions (adapted to SEL), e.g.,
Recommended Approach, Manager’s Handbook,
Cleanroom Process Handbook,
Ada Developer’s Guide, Ada Efficiency Guide

Reuse Inhibitors

Need to reuse **more than** just **code**, need to reuse all kinds of experience

Experience requires the **appropriate context** definition for to be reusable

Experience needs to be **identified and analyzed** for its reuse potential

Experience cannot always be reused as is, it needs to be **tailored**

Experience needs to be **packaged** to make it easy to reuse

Reuse of experience has been too informal, not **supported** by the organization

Reuse has to be **fully incorporated** into the development or maintenance process models

Project focus is delivery, not reuse,
i.e., **reuse cannot be a byproduct** of software development

*Need a separate organization to support the reuse of **local** experience*

Quality Improvement Paradigm

Activity Support for Improvement

Improving the software process and product requires

Learning

- the continual accumulation of evaluated experiences

Experience models

- in a form that can be effectively understood and modified

Experience base

- stored in a repository of integrated experience models

Reuse

- accessible and modifiable to meet the needs of the projects being developed by the organization

Quality Improvement Paradigm

Activity Support For Improvement

Systematic **learning** requires support for

- recording, ***off-line*** generalizing, tailoring, synthesizing and formalizing experience

Packaging and **modeling** useful experience requires

- a variety of models and formal notations that are tailorable, extendible, understandable, flexible and accessible

An effective **experience base** must contain

- accessible and integrated set of models that capture the ***local*** experiences

Systematic **reuse** requires support for

- using existing experience
- on-line generalizing or tailoring of candidate experience

Quality Improvement Paradigm

Organizational Support for Improvement

This combination of ingredients requires an **organizational structure** that supports:

A software evolution model that supports reuse

Processes for learning, packaging, and storing experience

The integration of these two functions

It requires **separate logical or physical organizations**:

with different focuses/priorities,

process models,

expertise requirements

Quality Improvement Paradigm

Organizational Support for Experience Reuse

Project Organization

focus/priority is delivery

supported by packaged experiences

Experience Factory

focus is project development support

analyzes and synthesizes all kinds of experience

acts as a repository for such experience

supplies that experience to various projects on demand

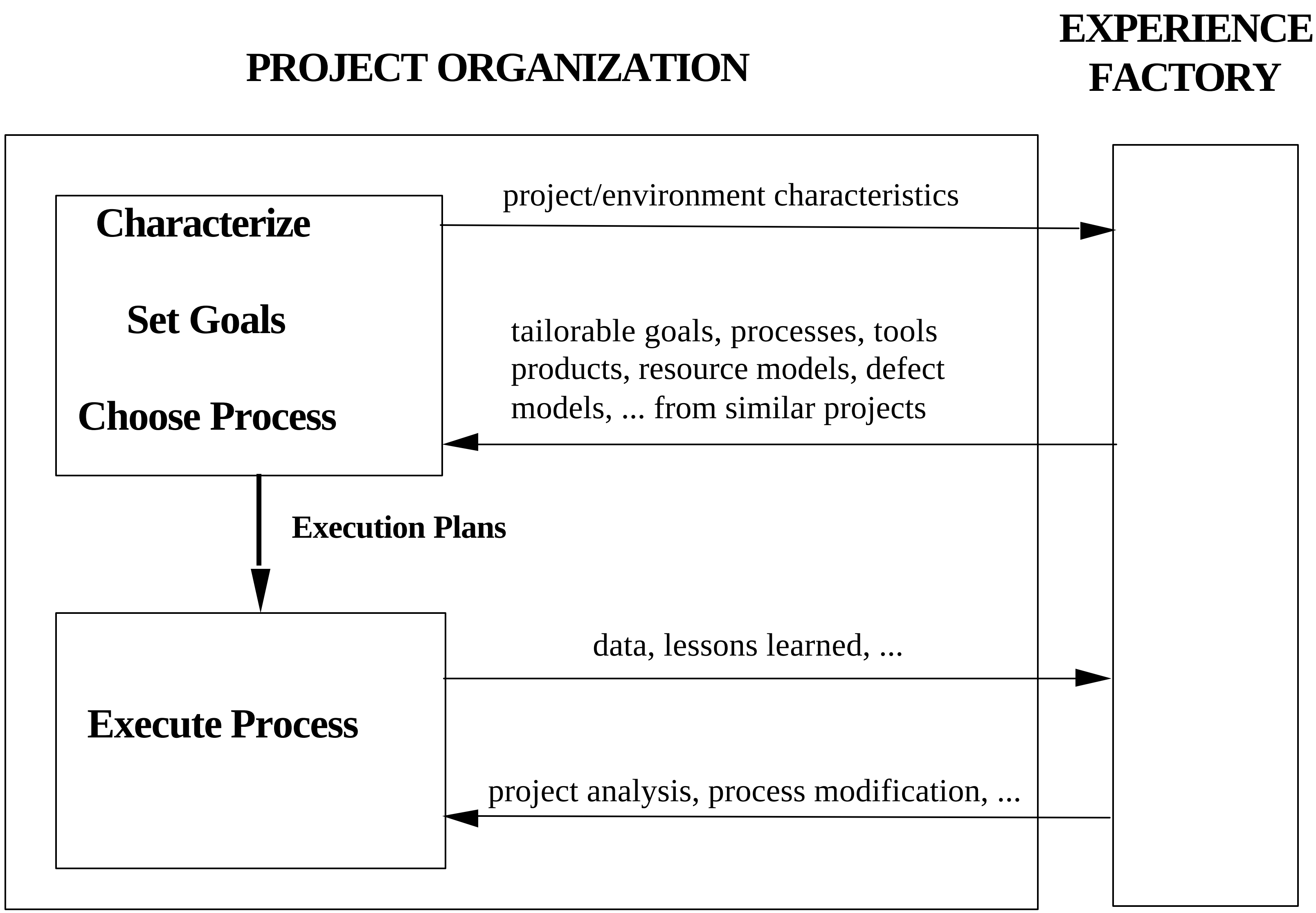
The **Experience Factory** packages experience by building

informal, formal or schematized, and productized models and measures

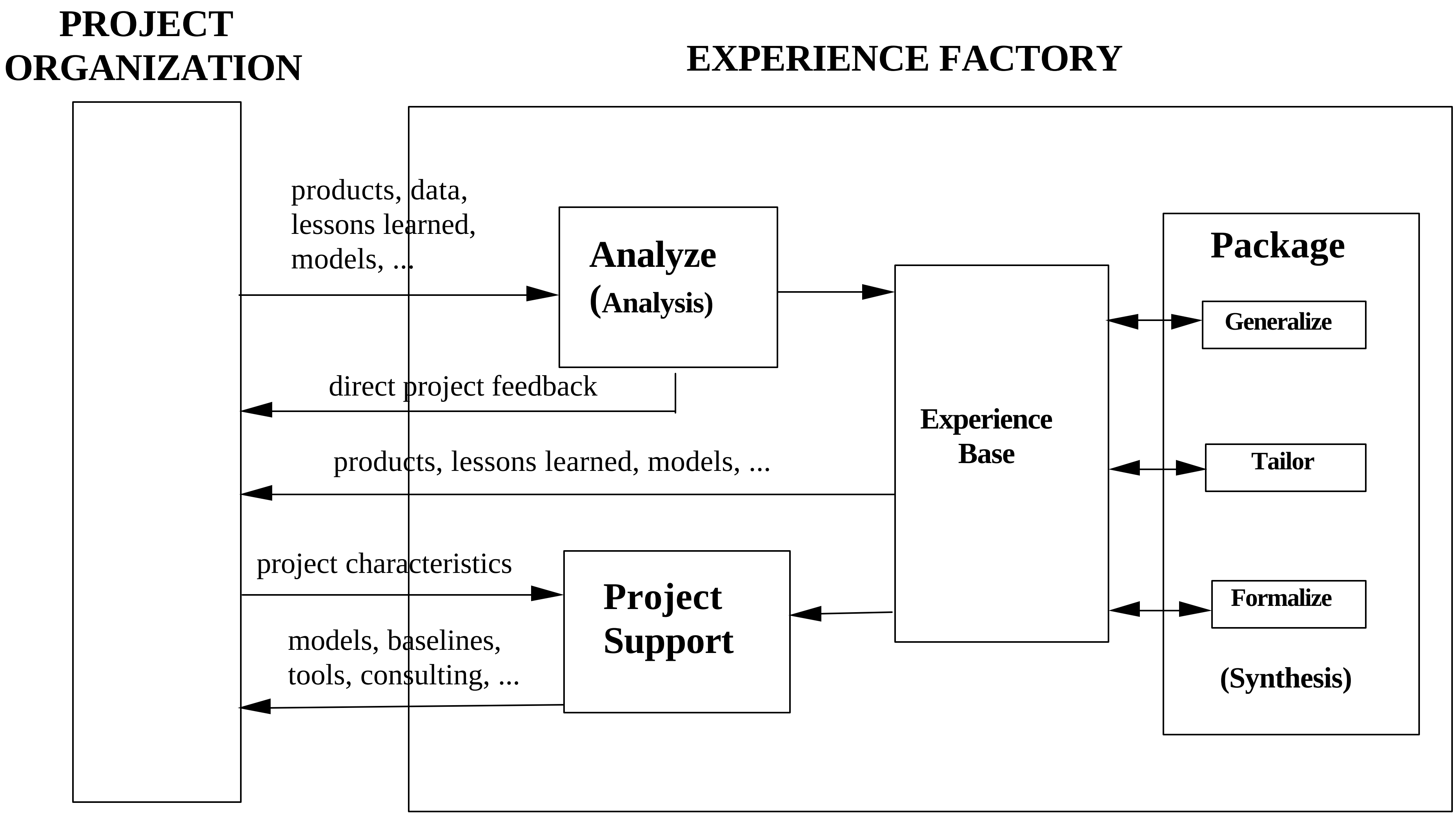
of various software processes, products, and other forms of knowledge

via people, documents, and automated support

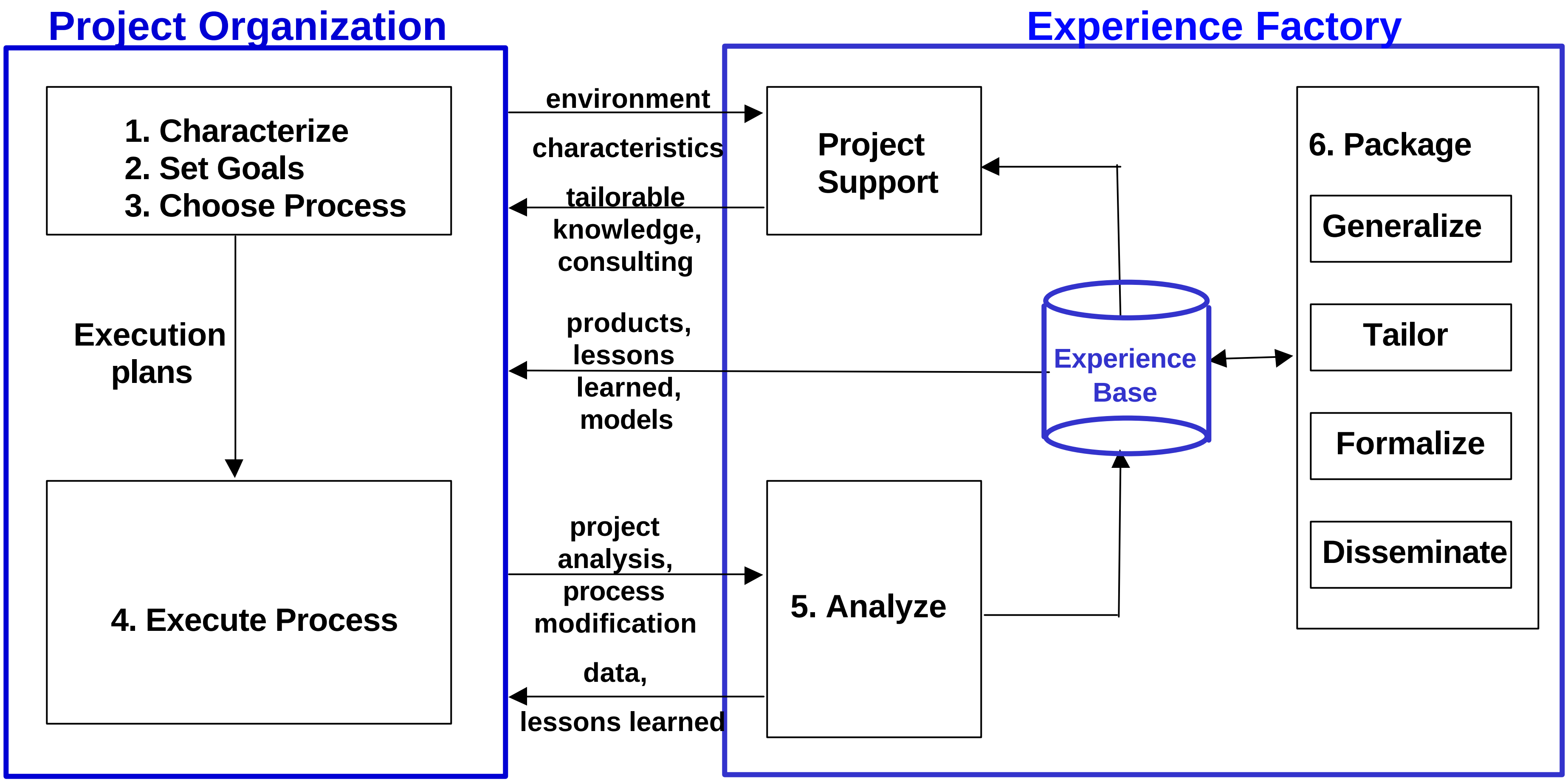
Experience Factory Organization
Role of the Project Organization



Experience Factory Organization
Role of the Experience Factory



Experience Factory Organization



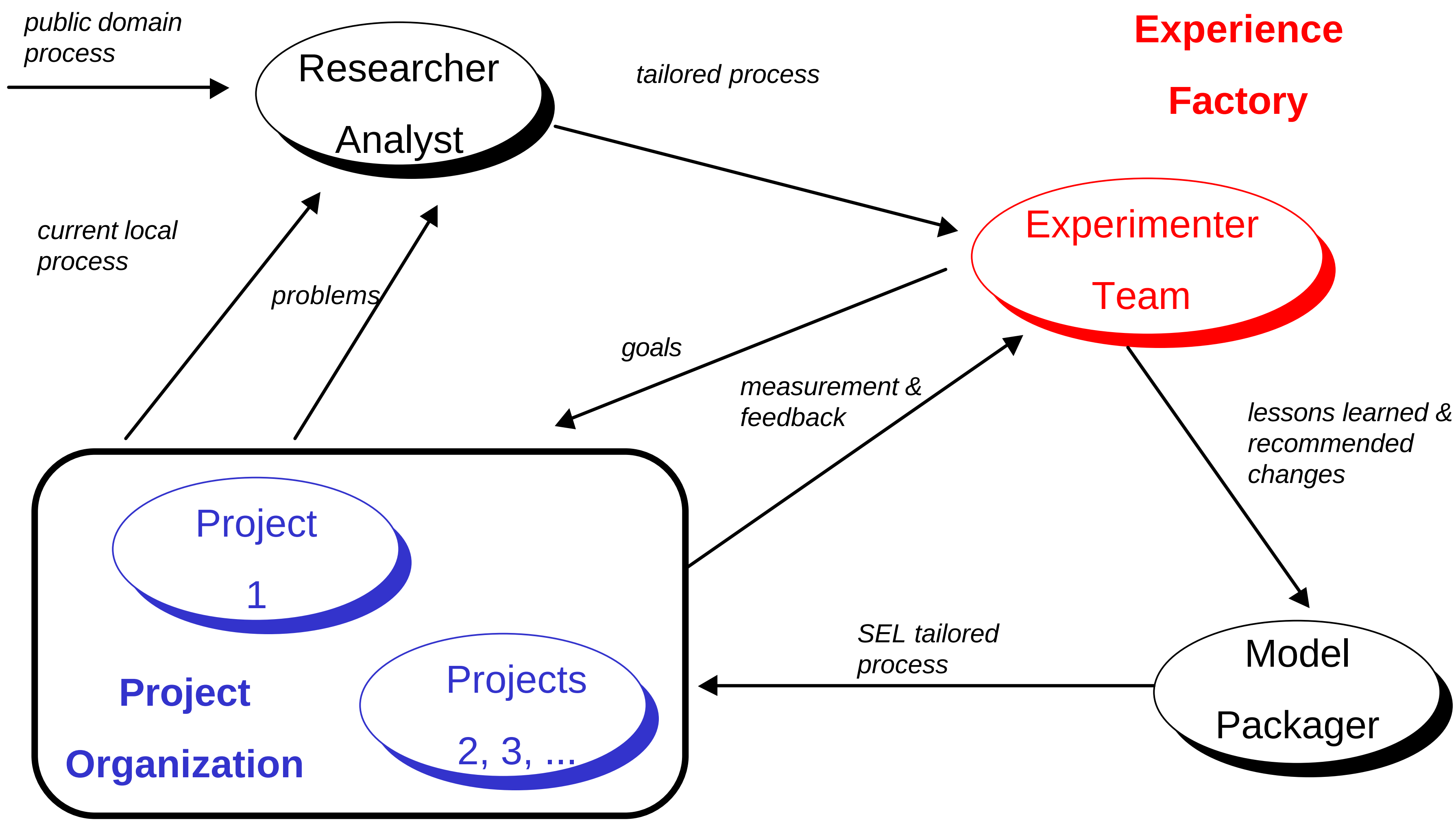
Experience Factory Organization

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**

Multi-Project Analysis Study
Improving via the Experience Factory

Process Evolution/Evaluation



An Example Experience Factory
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

An Example Experience Factory

SEL STRUCTURE

PO

DEVELOPERS
(SOURCE OF EXPERIENCE)

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

NASA + CSC

Development measures for each project

Refinements to development process

EF

PROCESS ANALYSTS
(PACKAGE EXPERIENCE FOR REUSE)

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

NASA + CSC + U of MD

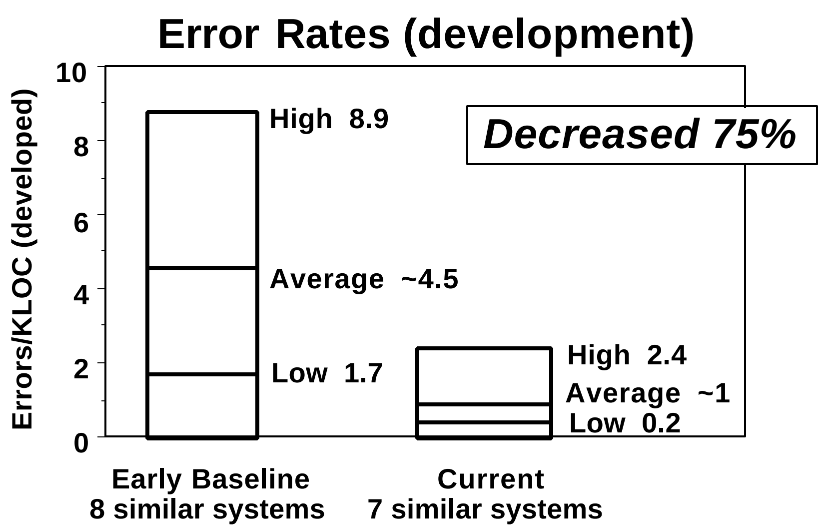
DATA BASE SUPPORT
(MAINTAIN/QA EXPERIENCE INFORMATION)

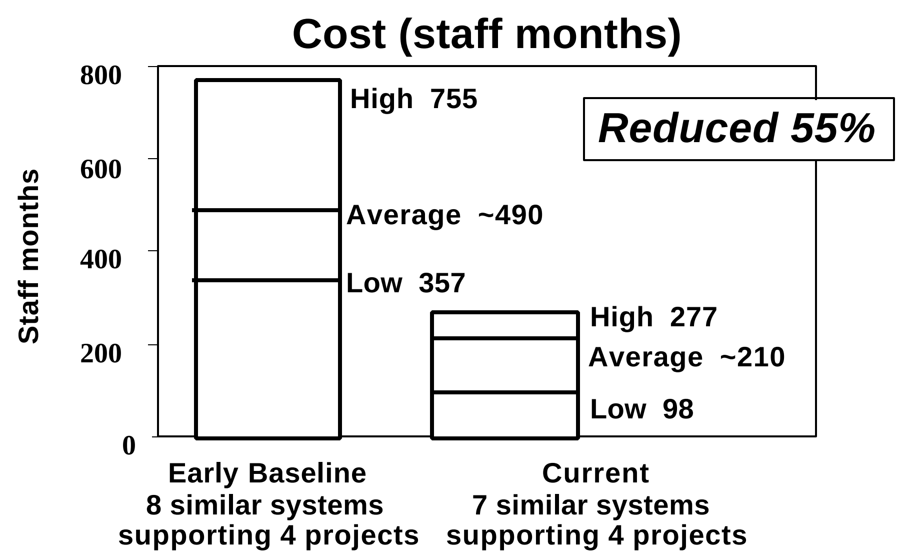
STAFF	3-6 support staff
FUNCTION	• Process forms/data • QA all data • Record/archive data • Maintain SEL data base • Operate SEL library

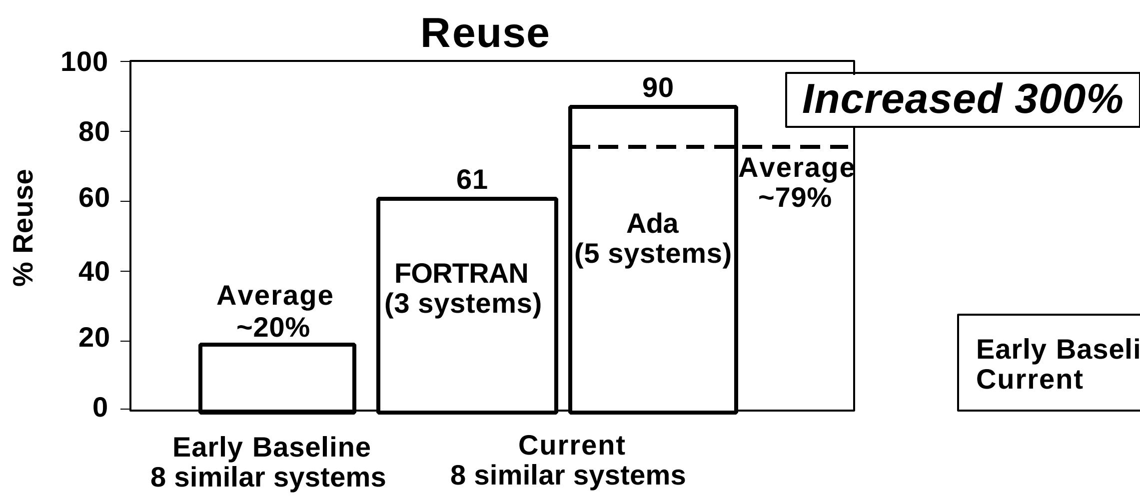
SEL DATA BASE	160 MB
FORMS LIBRARY	220,000
REPORTS LIBRARY	• SEL reports • Project documents • Reference papers

NASA + CSC

The Software Engineering Laboratory
Baselines 1987 and 1991







Early Baseline = 1985-1989
Current = 1990-1993

IEEE39

The Software Engineering Laboratory
An Experience Factory Example

The Software Engineering Laboratory
is the winner of the first

IEEE Computer Society Award
for
Software Process Achievement

The award is

- an international award established in 1994
- sponsored by the U.S. Software Engineering Institute
- for demonstrable, sustained, measured, significant software improvement

The Software Engineering Laboratory
Baselines 1987, 1991, 1995

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

THE EXPERIENCE FACTORY ORGANIZATION

Benefits

Separation of concerns and focus for development and packaging

Support for learning and reuse

Formalization and integration of management and development technologies

Generation of a tangible corporate asset:
an experience base of competencies

Offers a Lean Software Development Concept
compatible with TQM
A level 5 organizational structure

Practical link between focused research and development

The ability to start small and expand, e.g., you can focus on
a homogeneous set of projects,
a particular set of packages

THE EXPERIENCE FACTORY
Specific Steps

We need to:

Make the commitment
- Decide to make the change
- Involve top level management
- Think differently about software

Define a set of improvement goals
- Based on intuition/available data
- Look at high payoff areas, problem areas
- Need to debug the process

Choose a project
- Something mainstream
- Medium size
- Committed people

Organize to support the change
- Recognize the new processes
- Assign roles and resources

THE EXPERIENCE FACTORY
Specific Steps

Experiment with technology
- Don’t introduce too many changes
- Refine the technology to be measurable

Measure against the goals
- Collect data
- Validate
- Feedback

Learn
- Create database
- Do post-mortem analysis
- Write lessons learned documents

Modify the process
- Based upon learning
- Package what you have learned

Choose more projects and areas for improvement
- Number depends upon success of first

THE EXPERIENCE FACTORY ORGANIZATION

Conclusions

Integration of the **Improvement Paradigm**
Goal/Question/Metric Paradigm
Experience Factory Organization

Provides a **consolidation of activities**, e.g., packaging experience, consulting, quality assurance, education and training, process and tool support, measurement

Based upon our experience, it helps us

- understand how software is built and where the problems are
- define and formalize effective models of process and product
- evaluate the process and the product in the right context
- predict and control process and product qualities
- package and reuse successful experiences
- feed back experience to current and future projects

Can be applied today and evolve with technology

THE EXPERIENCE FACTORY ORGANIZATION

The approach provides:

a framework for defining quality operationally relative to the project and the organization

justification for selecting and tailoring the appropriate methods and tools for the project and the organization

a mechanism for evaluating the quality of the process and the product relative to the specific project goals

a mechanism for improving the organization's ability to develop quality systems productively

The approach is being adopted by several organizations, **but**

it is not a simple solution

it requires long-term commitment by top level management