ORGANIZATIONAL FRAMEWORKS

Organizational Frameworks

What is needed?

Measurement provides information but how does it get accumulated from project to project?

How does an organization develop core competencies?

How do I accumulate and build upon knowledge that is relevant to the domain?

We need to

- build baselines of resource use, defects, etc. over time
- build a measurement program that lasts across individual projects
- capture what we have learned about individual programs
- learn from project project to project
- store results of our measurement program for future analysis

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.

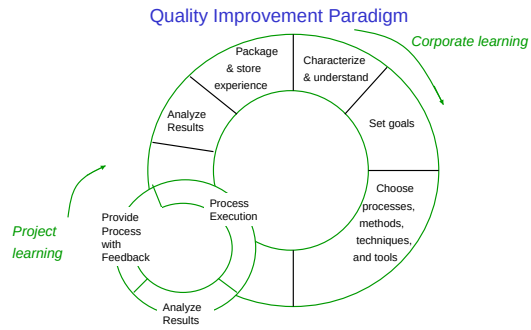
Choose the appropriate **process** model and supporting methods and tools for this project.

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
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

Quality Improvement Paradigm
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

Project Organization

Experience Factory

Quality Improvement Paradigm

Organizational Support for Experience Reuse

Project Organization

- focus/priority is delivery
- supported by packaged experiences

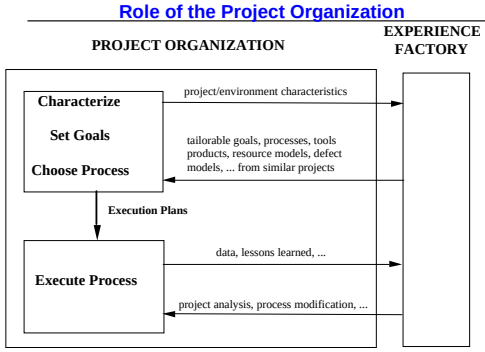
Experience Factory

- 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

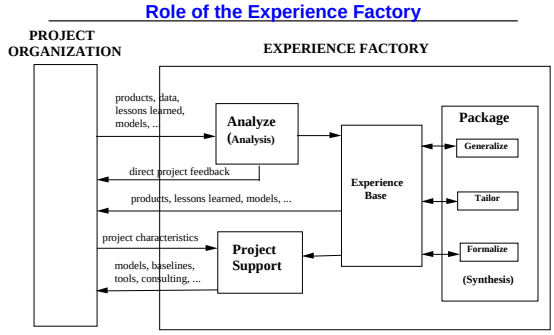
The **Experience Factory** packages experience by building

- 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

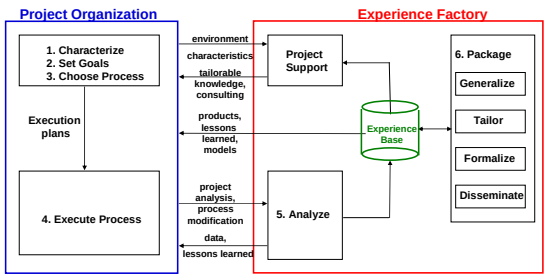
Experience Factory Organization
Role of the Project Organization



Experience Factory Organization
Role of the Experience Factory



THE EXPERIENCE FACTORY ORGANIZATION



Experience Factory Organization

A Different Paradigm

Project Organization	Experience Factory
Problem Solving	Experience Packaging
Decomposition of a problem into simpler ones	**U**nification of different solutions and re-definition of the problem
Instantiation	**G**eneralization, **F**ormalization
Design/**I**mplementation process	**A**nalysis/**S**ynthesis process
Validation and **V**erification	**E**xperimentation

The Experience Factory Organization

Some Important Characteristics

The QIP process is **iterative**
- don't be overly concerned with perfecting any step on the first pass
- the better your initial guess at the baselines, the sooner it will converge

No method is "packaged" that hasn't been tried:
- **applied, analyzed, tailored**

Experience Factory provides a way to evaluate
- **process conformance** and **domain understanding**

The Experience Factory Organization

Some Important Characteristics

Everyone is **part of** the technology infusion **process**
- can be a developer on one project and an experimenter on another

Project personnel play the **major role** in the feedback mechanism
- if they are not using the technology right it can be because:
  - they don't understand it / it wasn't taught right
  - it doesn't fit/interface with other project activities
  - it needs to be tailored
  - it doesn't work
- and you need the user to tell you how to change it

Technology infusion is **motivated by** the **local problems**,
- so people are more willing to try something new

The Experience Factory Organization

Iterating the QIP

Get the **commitment**

Put the **organization in place**, collect data to **establish baselines**
- e.g., defects and resources that are process and product independent

Measure your **strengths** and **weaknesses**
- Provides a focus and goals for improvement

Select and **experiment** with methods and techniques
- to improve process based upon product quality needs

Evaluate improvement based upon existing resource and defect baselines

Understand **process characteristics** and **product qualities relationship**
- Manipulate process to achieve those product characteristics
- Define and tailor better and measurable processes based upon
  - experience and knowledge of the environment
  - process conformance and domain understanding

Establish **new baselines**

Repeat the process and find the next opportunity for improvement

The Experience Factory Organization

Comparison with Other Approaches to Quality

Plan-Do-Check-Act
- a quality improvement process based upon a feedback cycle for optimizing a single process model/production line

Total Quality Management
- a management approach to long term success through customer satisfaction based on the participation of all members of an organization

SEI Capability Maturity Model
- staged process improvement based upon assessment with regard to a set of key process areas until you reach a level 5 which represents a continuous process improvement

Lean Enterprise Management
- principle supporting the concentration of production on "value added" activities and the elimination or reduction of "not value added" activities

Approaches To Quality

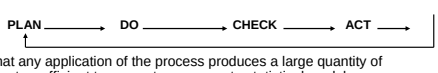
Plan-Do-Check-Act Cycle (PDCA)

Based upon work by W. A. Shewart and made popular by W. E. Deming

Goal: optimize and improve a single process model/production line

Approach: uses such techniques as
- feedback loops
- statistical quality control
- design of experiments
- data models based upon multiple replications

Result: predictive models of the relationship between process and product



Note: that any application of the process produces a large quantity of products, sufficient to generate an accurate statistical model

Approaches To Quality

PDCA vs. EF

Similarities
- scientific method
- feedback loops from product to process
- learn from experiments

Differences
- PDCA based upon production
  - it attempts to optimize a single process model/production line
  - based upon continual repetition of the same process
  - can collect sufficient data to develop quantitative models
  - can evaluate/predict accurately effects of the process
  - can use accurate models for statistical quality control
- EF based upon development,
  - rarely replicate the same thing twice
  - must learn from one process about another
  - models are less rigorous and more abstract
  - processes more human based
  - effects building, use, and accuracy of models built

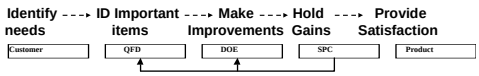
Approaches To Quality

Total Quality Management

Term coined by Navy in 1985. Based upon work by Feigenbaum, Taguchi, ...

Goal: generate institutional commitment to success through customer satisfaction

Approach: varied, a philosophy supported by a variety of techniques, e.g., *
- Quality Function Deployment (QFD)
- design of experiments (DOE)
- statistical process control (SPC)



Result: An customer driven organization and a satisfied set of customers

***Source: Michael Deutsch at Hughes**

Approaches To Quality

TQM vs. EF

Similarities
- cover goals that are customer satisfaction driven
- based upon the philosophy that quality is everyone's job
- everyone is part of the technology infusion process
- can be on project team on one project, experimenting team on another
- all the project personnel play the major role in the feedback mechanism

Differences

EF provides
- specific steps, model types
- more specific and aimed at software

APPROACHES TO QUALITY

Lean Enterprise Management (LEM)

Philosophy used to improve factory output. Book by Womack, et. al. (1989), on the application of lean enterprises in the automotive industry

Goal: to build products using the minimal set of activities needed, eliminating non essential steps, i.e., tailoring the process to the product needs

Approach: uses such concepts as
- technology management
- human centered management
- decentral organization
- quality management
- supplier and customer integration
  - internationalization/regionalization

Result: A set of processes individualized for each particular product line

Approaches To Quality

LEM vs. EF

Similarities
- scientific method /PDCA philosophy
- feedback loops, learn from experiments, process/product relationship
- goal is to generate an optimum set of processes
- based upon tailoring a set of processes for particular product

Differences

LEM based upon production
- model building based upon continual repetition of the same process
- can use accurate models for statistical quality control

EF based upon development,
- must learn from one process about another
- models are less rigorous and more abstract
- processes more human based
- effects building, use, and accuracy of models built

Approaches To Quality

SEI Capability Maturity Model (CMM)

Organizational/quality management maturity models by R. Likert/P. Crosby, Software model by R. Radice, made popular by Watts Humphrey at SEI

Goal: a level 5 maturity rating, implying continuous process improvement via defect prevention, technology innovation, and process change management

Approach: A 5 level process maturity model defined. Maturity level defined based on repeated assessment of an organization's capability in key process areas. Improvement achieved by action plans for poorly assessed processes

Level	Focus
5 Optimizing	Continuous Process Improvement
4 Managed	Product & Process Quality
3 Defined	Engineering Process
2 Repeatable	Project Management
1 Initial	Heros

Result: A set of well-defined key processes

Approaches To Quality

CMM vs. EF

Similarities
- characterize processes

Differences

CMM
- goal is to improve process
- characterize processes
- baseline is process assessment
- common yardstick drives change (key process areas)
- change based upon assessment of processes
- measurement plays key role at level 4
- process emphasis is on management activities

EF
- goal is to improve product
- characterizes all kinds of experiences: products, defects, resources
- baseline is process and product understanding
- many goals drive change, e.g., customer satisfaction
- change based upon achieving goals
- measurement fundamental at all stages
- process emphasis is on technological and management activities

Approaches to Quality
SEI Process Improvement Cycle

Initialize

Establish Sponsorship
Create vision and strategy
Establish improvement structure

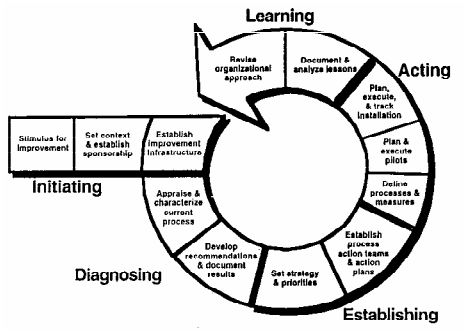
For Each Maturity Level

- Characterize current practice in terms of key process areas
- Assessment recommendations
- Revise strategy (generate action plans and prioritize key process areas)

For Each key Process Area

- Establish process action teams
- Implement tactical plan, define processes, plan and execute pilot(s), plan and execute institutionalization
- Document and analyze lessons
- Revise organizational approach

SEI's Ideal Improvement Model



Evolution of Process Capability

Level	Process Characteristics	Predicted Performance
Optimizing	Process improvement is institutionalized	
Managed	Product and process are quantitatively controlled	
Defined	Software engineering and management processes defined and integrated	
Repeatable	Project management system in place; performance is repeatable	
Initial	Process is informal and ad hoc; performance is unpredictable	

Source: Carnegie Mellon University, Software Engineering Institute

Experience Factory Organization

Can it make you a 5?

Using the Experience Factory Organization:

You pull yourself up from the top rather than pushing up from the bottom

At step 1 you start with a level 5 organization but not level 5 capabilities

You are driven by an **understanding** of **your** business, **your** product and process problems, **your** business goals, **your** experience with methods, etc.

You learn from your business, not on an external model of process

You make process improvements based upon an understanding of the relationship between process and product in your organization

Experience Factory Organization

Can it make you a 5?

What does a level 5 organization mean?

It is an organization that can manipulate process to achieve various product characteristics.

This requires that we have a process and an organizational structure to help us:

- Understand our processes and products
- Measure and model the project and the organization
- Define and tailor process and product qualities explicitly
- Understand the relationship between process and product qualities
- Feedback information for project control
- Experiment with methods and techniques
- Evaluate our successes and failures
- Learn from our experiences
- Package successful experiences
- Reuse successful experiences

Experience Factory Organization

Can it make you a 5?

Using EF may not get you a level 5 rating
(depending on how it gets defined when you get there)
because your technologies are not from the "key set of processes"
but you are operating at a level 5 definition
and have chosen and tailored processes to create a
lean, optimizing, continuously improving organization

How does this fit in with the CMM?

EF is not incompatible with the SEI CMM model
can use key process assessments to evaluate where you stand
(along with your internal goals, needs, etc.).

Using the EF will move you up the maturity scale faster
offers experience early on with an improvement-based organization
can demonstrate product improvement benefits early

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
Business Requirements

Business goal: To win in the market place

Approach:

Develop strategy for your business
Map strategy onto different parts of the business, e.g., software
Choose where you want to be strong
This provides a road map for the business
Set goals for each part of the business
Understand your existing capabilities and competencies
what are your competencies relative to your needs
Reset your goals in the context of this understanding
Put the competencies in place

Need to:

Recognize the role software plays in our business
Invest heavily in our software core competencies

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
Software Quality: State of the Practice

Quality Definition:
Quality means less errors reported from the customer

Process Selection:
Companies have a software development methodology standards document
(which few projects follow)

Quality Evaluation:
Evaluation means counting the number of customer trouble reports generated

Quality Organization:
A quality assurance organization means providing management with the trouble report data

THE SOFTWARE BUSINESS
Software Quality: State of the Practice

What's wrong with these definitions?

Quality Definition: *Customer reported defects*
Too narrow - Quality means more than error reports

Process Selection: *Methodology standards document*
Too general - Not all projects are the same

Quality Evaluation: *Number of customer trouble reports*
Passive rather than active -
- No way to learn how to do better
- No evaluation of the methods or tools

Quality Organization: *Trouble Report graphs*
Not quality oriented - Quality is the result not the driver

THE SOFTWARE BUSINESS
Quality Control vs. Quality Assurance

Quality control:

The act of directing, influencing, verifying, and correcting for ensuring the conformance of a specific product to a design or specification.

Quality assurance:

The act of leading, teaching, auditing the process by which the product is produced to provide confidence in the conformance of a specific product to a design or specification.

THE SOFTWARE BUSINESS
Quality Control Organization

(Inside the project)

Activities:
- **Evaluates products**
- **Provides feedback to project**

Infrastructure
- **Part of the project**
- **Interactive with the project**

People requirements:
- **Knowledge of the processes**
- **Knowledge of the product requirements**
- **Understanding of the solutions**

THE SOFTWARE BUSINESS
Quality Assurance Organization

Activities:
- **Defines and evaluates processes**
- **Collects data from quality control**
- **Provides feedback to projects**
- **Provides feedback to the organization**
- **Provides feedback to QA**

Infrastructure
- **Independent chain of command**
- **Interactive with the projects**

People requirements:
- **High level with respect to technology and management**
- **Deep understanding of the process and the product**

THE SOFTWARE BUSINESS
Quality Control

What types of constructive **and** analytic **activities do we perform?**

CONSTRUCTIVE ACTIVITIES	ANALYTIC ACTIVITIES
Specifying	Reviewing
Designing	Inspecting
Coding	Testing

What % of the time is spent in constructive vs. analytic activities?

How do we define and differentiate these activities?

What are the goals of each activity?

From whose perspective are they studied?
e.g., customer, manager, corporation

THE SOFTWARE BUSINESS
Quality Control and Assurance Needs

What power do QC and QA have?

Can a project be stopped from moving on to the next phase?
Can part of a design be rejected because it doesn't pass standards?

What is needed?

- **We need better models of the processes and products**
- **We need better feedback of information for project control**
- **We need to have prioritize goals (e.g., schedule vs. quality)**
- **We need a basis for comparison in the form of baselines (measurement data)**
- **We need to evolve to something akin to statistical quality control**
- **We need to improve with experiences**

THE SOFTWARE BUSINESS
Software Quality Needs

Characterize/Understand

Differentiate project environments and understand the current software process and product

Provide baselines for future assessment

Build descriptive models and baselines

Evaluate/Assess

Assess the achievement of quality goals

Assess the impact of technology on products

Understand where technology needs to be improved, tailored

Compare models

Predict

Be able to control and understand the relationships between and among processes and products

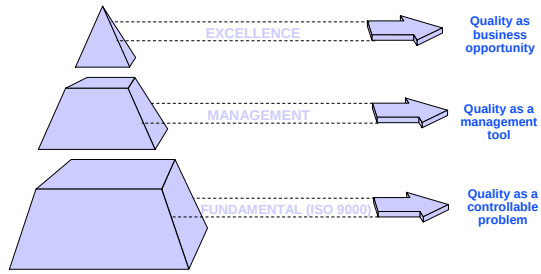
Find patterns in models

Motivate/Manage/Control

Provide quantitative motivation and guidelines that support what it is we are trying to accomplish

Build and Package prescriptive models for reuse

THE SOFTWARE BUSINESS
The Pyramid of Quality



CORE COMPETENCIES
Four Principles of Corporate Success

Business processes are the building blocks of the corporate strategy

Competitive success depends on understanding and transforming the key business processes into **strategic capabilities**

Strategic capabilities are created by sustained investments in a support infrastructure that links together and transcends the business units

A capability-based strategy must be sponsored by the top management of the corporation

From: **Stalk, Evans and Shulman in HBR**

CORE COMPETENCIES
Definitions

- **Strategic Capabilities** are corporate goals defined by the business position of the organization and implemented by key business processes

  E.g., **Cycle-time reduction**

- **Core competencies** are aggregate tailored technologies that play a key role in supporting the strategic capabilities of an organization

  E.g., **Integrated software engineering environment**

- **Technologies/methodologies** are basic processes and tools,associated artifacts used in any stage of the life cycle (from project generation to post-deployment operation and support)

  E.g., **Domain specific architectures**

CORE COMPETENCIES
Examples of technologies and methodologies

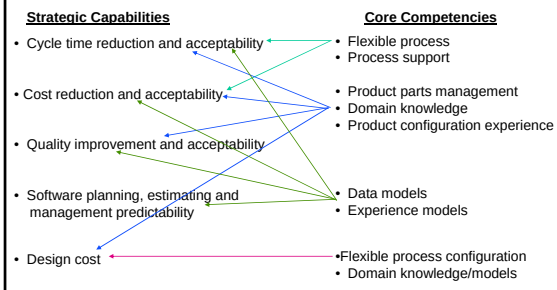
- **Domain analysis and architectures**
- Tool integration
- **Structured analysis and design**
- Object-Oriented Techniques (OOA, OOD, OOPL)
- **Reuse libraries, mechanisms and methods**
- DB compression techniques
- **Data and process modeling**
- Data sharing and communication in heterogeneous environments
- **Business process re-engineering**
- Inspections
- **Defect counting, categorization and analysis**
- Defect prevention methods
- **Testing technologies**
- Domain specific algorithms
- **Measurement and data collection and analysis**
- Reliability modeling and prediction

CORE COMPETENCIES
Example Technologies and Core Competencies

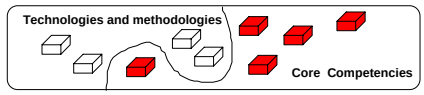
Core competencies	**Technologies**
• Availability and use of a software management environment based on "local" data for estimate, control and prediction of projects	‹ Measurement and data collection and analysis ‹ Data and process modeling ‹ Defect counting, categorization and analysis
• Use of an integrated software engineering environment tailored to one or more specific application domains	‹ Tool integration ‹ Domain analysis and architectures ‹ Data sharing and communication in heterogeneous environments
• Availability of reusable components (modules, algorithms, architectures) and tools portable across different platforms	‹ Reuse libraries, mechanisms and methods ‹ Domain analysis and architectures ‹ Object-Oriented Techniques (OOA, OOD, OOPL)

CORE COMPETENCIES AND STRATEGIC CAPABILITIES

Examples

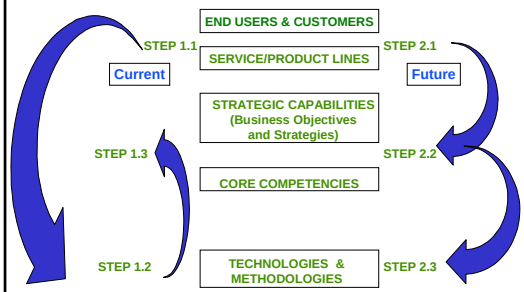


CORE COMPETENCIES Characteristics



- **Characteristics of core competencies among technologies:**
  - non-transitional
  - have a consistent evolution
  - require commitment, investment and leadership to remain
  - typically fueled by and work with multiple technologies
  - generally support multiple product/service lines
- **Existing products and services can become core competencies if the appropriate strategy is decided**

STRATEGIC CAPABILITIES Planning Process



STRATEGIC CAPABILITIES Example Industrial Groups

G1 **Computer manufacturers**: Software represents the operating environments.
Examples: IBM/Commercial, Siemens, Sun, DEC, ...

G2 **System integrators/Contractors**: Software delivers services to users.
Examples: EDS, CSC, SAIC, Hughes/IS, ...

G3 **Software Houses**: Software commodities are their product.
Examples: Microsoft, Lotus, Borland, ...

G4 **Manufacturers of Systems**: Systems are critically dependent on software.
Examples: ABB, Alcatel, Daimler-Benz, Ericsson, Motorola, Siemens...

STRATEGIC CAPABILITIES Relevance by Industry Group

Strategic Capability	Relative Importance			
	G1 (Hw)	G2 (Contr)	G3 (Sw House)	G4 (Syst.)
Software planning, estimating and management predictability	Low	High	Medium	Medium
Cycle time reduction/acceptability	High	Low	High	Medium
Cost reduction/acceptability	Medium	High	Medium	Medium
Quality Improvement/acceptability	Low	Low	Medium	High

References

V. Basili, Data Collection, Analysis and Validation, chapter in Software Metrics, Massachusetts Institute of Technology Press, pp. 143-160, 1981.

V. Basili, Quantitative Evaluation of Software Methodology, Keynote Address, First Pan Pacific Computer Conference, Proceedings, Volume 1, pp. 379-398, Melbourne, Australia, September 10-13, 1985.

V. Basili, Software Development: A Paradigm for the Future, COMPSAC '89, Orlando, Florida, pp. 471-485, September 1989.V. Basili, The Experience Factory and Its Relationship to Other Quality Approaches, Academic Press, Inc., Advances in Computers, Volume 41, 1995.

V. Basili and H.D. Rombach, The TAME Project: Towards Improvement-Oriented Software Environments, IEEE Transactions on Software Eng., vol. 14, #6, June 1988.

S.H. Kan, V. Basili and L.N. Shapiro, Software Quality: An Overview from the Total Quality Management Perspective, IBM Systems Journal, Vol. 33, No. 1, March 1994.

V. Basili and S. Green, Software Process Evolution at the SEL, IEEE Software, pp. 58-66, July 1994.

V. Basili, M. Zelkowitz, F. McGarry, J. Page, S. Waligora, and R. Pajerski, Special Report: SEL's Software Process-Improvement Program, IEEE Software, Volume 12, Number 6, pp. 83-87, November 1995.

References

M. Daskalantonakis, V. Basili, and R. Yacobellis, A Method for Assessing Software Measurement Technology, Quality Engineering 3(1), pp. 27-40, 1990-91.

V. Basili and H.D. Rombach, Support for Comprehensive Reuse, Software Engineering Journal, IEE British Computer Society, Vol. 6, No. 5, pp. 303-316, September 1991.

V. Basili and J. Musa, The Future Engineering of Software: A Management Perspective, IEEE Computer Magazine, Vol. 24, No. 9, pp. 90-96, September, 1991. And reprinted in Software Project Management Readings and Cases by Chris F. Kemerer, pp. 30-40, The McGraw-Hill Companies, Inc., 1997.

V. Basili, The Experience Factory and its Relationship to Other Improvement Paradigms, 4th European Software Engineering Conference (ESEC) in Garmish-Partenkirchen, Germany. The Proceedings appeared as the Springer-Verlag Lecture Notes in Computer Sciences Series 717, September 1993.