The Experience Factory and its Relationship to Other Improvement Paradigms[1]

Victor R. Basili

Institute for Advanced Computer Studies
Department of Computer Science
University of Maryland

Abstract. This paper describes the Quality Improvement Paradigm and the Experience Factory Organization as mechanisms for improving software development. It compares the approach with other improvement paradigms.

1. Introduction

The concepts of quality improvement have permeated many businesses. It is clear that the nineties will be the quality era for software and there is a growing need to develop or adapt quality improvement approaches to the software business. Thus we must understand software as an artifact and software as a business.

Any successful business requires a combination of technical and managerial solutions. It requires that we understand the processes and products of the business, i.e., that we know the business. It requires that we define our business needs and the means to achieve them, i.e., we must define our process and product qualities. We need to define closed loop processes so that we can feedback information for project control. We need to evaluate every aspect of the business, so we must analyze our successes and failures. We must learn from our experiences, i.e., each project should provide information that allows us to do business better the next time. We must build competencies in our areas of business by packaging our successful experiences for reuse and then we must reuse our successful experiences or our competencies as the way we do business.

Since the business we are dealing with is software, we must understand the nature of software and software development. The software discipline is evolutionary and experimental; it is a laboratory science. Software is development not production. The technologies of the discipline are human based. There is a lack of models that allow us to reason about the process and the product. All software is not the same; process is a variable, goals are variable, etc. Packaged, reusable, experiences require additional resources in the form of organization, processes, people, etc.

2. Experience Factory /Quality Improvement Paradigm

The **Experience Factory /Quality Improvement Paradigm** [Ba85a], [Ba89], [BaRo87], [BaRo88] aims at addressing the issues of quality improvement in the software business by providing a mechanism for continuous improvement through the experimentation, packaging and reuse of experiences based upon a business's needs. The approach has been evolving since 1976 based upon lessons learned in the SEL [BaCaMc92].

[1] This paper was presented at the 4th European Software Engineering Conference (ESEC) in Garmisch-Partenkirchen, Germany, September, 1993. The preedings appears as the Springer-Verlag Lecture Notes in Computer Science Series 717, edited by Ian Sommerville and Manfred Paul.

The basis for the approach is the **Quality Improvement Paradigm** which consists of six fundamental steps:

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

Set the 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 and validate the prescribed data, and analyze it 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.

Although it is difficult to describe the Quality Improvement Paradigm in great detail here, we will provide a little more insight into the six steps.

Characterizing the Project and Environment. Based upon a set of models of what we know about our business we need to classify the current project with respect to a variety of characteristics, distinguish the relevant project environment for the current project, and find the class of projects with similar characteristics and goals. This provides a context for goal definition, reusable experiences and objects, process selection, evaluation and comparison, and prediction. There are a large variety of project characteristics and environmental factors that need to be modeled and base lined. They include various
- people factors, such as the number of people, level of expertise, group organization, problem experience,
- process experience,etc.; problem factors, such as the application domain, newness to state of the art, susceptibility to change, problem constraints, etc.,
- process factors, such as the life cycle model, methods, techniques, tools, programming language, other notations, etc.,
- product factors, such as deliverables, system size, required qualities, e.g., reliability, portability, etc., and
- resource factors, such as target and development machines, calendar time, budget, existing software, etc.

Setting Measurable Goals. We need to establish goals for the processes and products. These goals should be measurable, driven by models of the business. During this step, the measurement data is also defined. There are a variety of mechanisms for defining measurable goals: Quality Function Deployment Approach (QFD) [KoAk83], the Goal/Question/Metric Paradigm (GQM) [WeBa85], and the Software Quality Metrics Approach (SQM) [McRiWa77].

Goals may be defined for any object, for a variety of reasons, with respect to various models of quality, from various points of view, relative to a particular environment. For example, goals should be defined from a variety of points of view: user, customer, project manager, corporation, etc.

Choosing the Process Models. We need to be able to choose a generic process model appropriate to

the specific context, environment, project characteristics, and goals established for the project at hand, as well as any goals established for the organization, e.g., experimentation with various processes or other experience objects. This implies we need to understand under what conditions various processes are effective. All processes must be defined to be measurable and defined in terms of the goals they must satisfy. The concept of defining goals for processes will be made clearer in later chapters.

Once we have chosen a particular process model, we must tailor it to the project and choose the specific integrated set of sub-processess, such as methods and techniques, appropriate for the project. In practice, the selection of processes is iterative with the redefinition of goals and even some environmental and project characteristics. It is important that the process model resulting from these first three steps be integrated in terms of its context, goals and sub-processes. The real goal is to have a set of processes that will help the developer satisfy the goals set for the project in the given environment. This may sometimes require that we manipulate all three sets of variables to assure this consistency.

Executing the Processes. The development process must support the access and reuse of packaged experience of all kinds. One the other hand it needs to be supported by various types of analysis, some done in close to real time for feedback for corrective action. To support this analysis, data needs to be collected from the project. But this data collection must be integrated into the processes, not be an add on, e.g. defect classification forms can be part of the configuration control mechanism. Processes must be defined to be measurable to begin with, e.g., design inspections can be defined so that we keep track of the various activities, the effort expended in those activities, such as peer reading, and the effects of those activities, such as the number and types of defects found. This allows us to measure such things as domain conformance and assures that the processes are well defined and can evolve.

Support activities, such as data validation, education and training in the models and metrics and data forms are also important. Automation is necessary to support mechanical tasks and deal with the large amounts of data and information needed for analysis. It should be noted however, that most of the data cannot be automatically collected. This is because the more interesting and insightful data tends to require human intervention.

The kinds of data collected include:
- resource data such as effort by activity, phase, type of personnel, computer time, and calendar time;
- change and defect data, such as changes and defects by various classification schemes,
- process data such as process definition, process conformance, and domain understanding;
- product data such as
    - product characteristics, both
        - logical, e.g., application domain, function, and
        - physical, e.g. size, structure, and
    - use and context information, e.g., who will be using the product and how will they be using it so we can build operational profiles.

Analyzing the Data. Based upon the goals, we interpret the data that have been collected. We can use these data to characterize and understand, so we can answer questions like "What project characteristics affect the choice of processes, methods and techniques?" and "Which phase is typically the greatest source of errors?" We can use the data to evaluate and analyze to answer questions like "What is the statement coverage of the acceptance test plan?" and "Does the Cleanroom Process reduce the rework effort?" We can use the data to predict and control to answer questions like "Given a set of project characteristics, what is the expected cost and

reliability, based upon our history?" and "Given the specific characteristics of all the modules in the system, which modules are most likely to have defects so I can concentrate the reading or testing effort on them?" We can use the data to motivate and improve so we can answer questions such as "For what classes of errors is a particular technique most effective?" and "What are the best combination of approaches to use for a project with a continually evolving set of requirements based upon our organization's experience?"

Packaging the models. We need to define and refine models of all forms of experiences, e.g., resource models, such as baselines, change and defect models, product models, process definitions and models, method and technique evaluations, products and product parts, quality models, and lessons learned. These can appear in a variety of forms, e.g., we can have mathematical models, informal relationships, histograms, algorithms and procedures, based upon our experience with their application in similar projects, so they may be reused in future projects.

The six steps of the Quality Improvement Paradigm can be combined in various ways to provide different views into the activities. First note that there are two feedback loops, a project feedback loop that takes place in the execution phase and an organizational feedback loop that takes place after a project is completed and changes the organization's understanding of the world between the packaging of what was learned form the last project and the characterization and base lining of the environment for the new project. It should be noted that there are numerous other loops visible at lower levels of instantiation, but these high level loops are the most important from an organizational structure point of view.

One high level organizational view of the paradigm is that we must **understand** (Characterize), **assess** (Set goals, Choose processes, Execute processes, Analyze data) and **package** (Package experience). Another view is to **plan** for a project (Characterize, Set goals, Choose processes), **develop** it (Execute processes), and then **learn** from the experience (Analyze data and Package experience).

2.1 The Experience Factory Organization

To support the Improvement Paradigm, an organizational structure called the Experience Factory Organization was developed [Ba89]. It recognizes the fact that improving the software process and product requires the continual accumulation of evaluated experiences (**learning**), in a form that can be effectively understood and modified (**experience models**), stored in a repository of integrated experience models (**experience base**), that can be accessed/modified to meet the needs of the current project (**reuse**).

Systematic **learning** requires support for recording, ***off-line*** generalizing,tailoring, formalizing and synthesizing of experience. The off-line requirement is based upon the fact that reuse requires separate resources to create reusable objects. 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 and on-line generalizing or tailoring of candidate experience.

This combination of ingredients requires an organizational structure that supports: (1) a software evolution model that supports reuse, (2) processes for learning, packaging, and storing experience, and (3) the integration of these two sets of activities. It requires separate logical or physical organizations with different focuses/priorities, process models, expertise requirements.

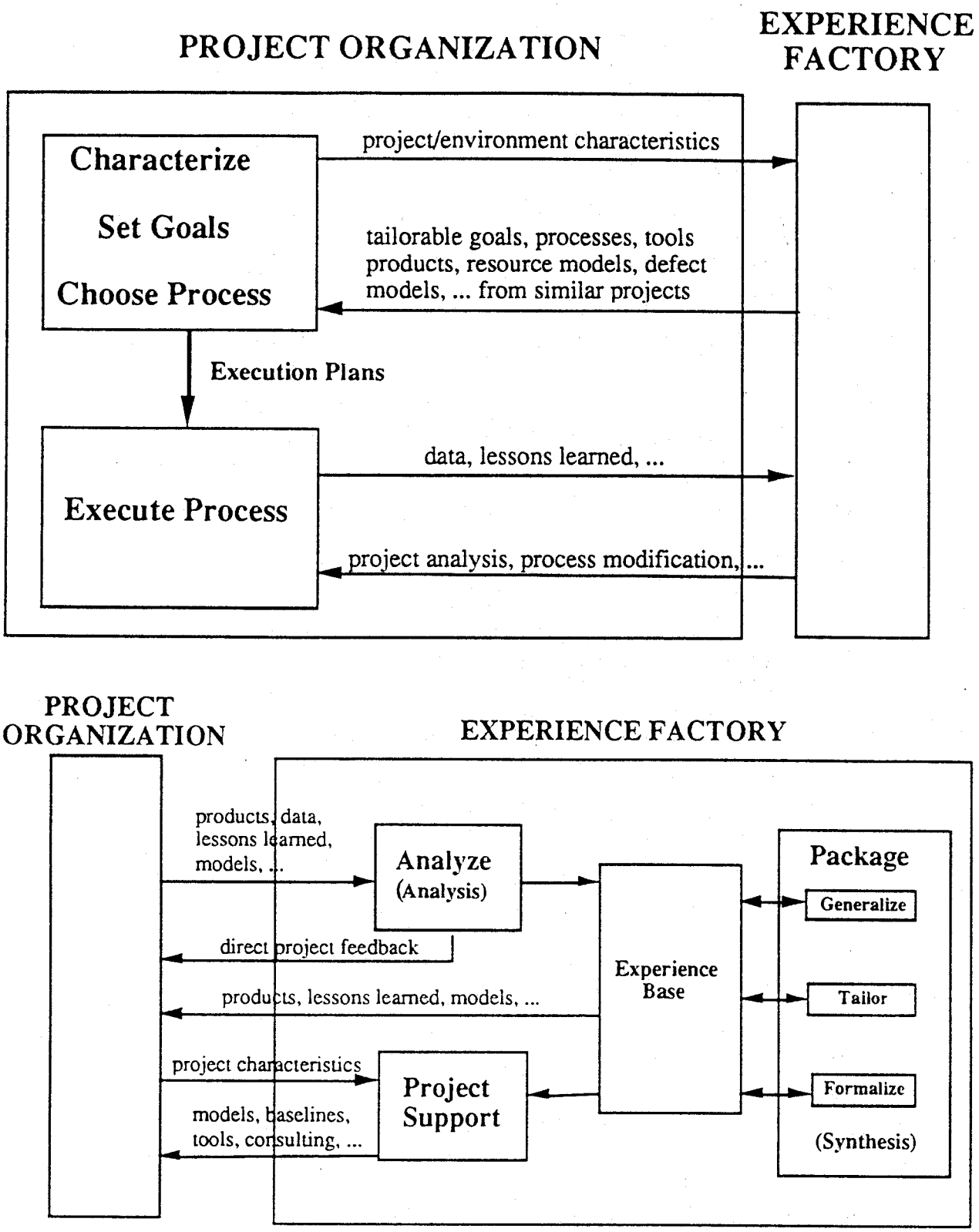


Figure 1. EXPERIENCE FACTORY ORGANIZATION

We divide the activities into two separate logical or physical organizations (Figure 1): the **Project Organization** whose focus/priority is delivery, supported by packaged reusable experiences, and an **Experience Factory** whose focus is to support project developments by analyzing and synthesizing all kinds of experience, acting as a repository for such experience, and supplying 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.

As shown in Figure 1, the steps of the Improvement Paradigm are assigned various roles in the two organizational structures. In the Project Organization, the project management characterizes the project, sets the measurable goals and gets support from the Experience Factory to develop an executable process model, tailored to the project at hand. It then executes the processes, providing information to the experience factory which offers feedback on status, improvements, and risks. On the other hand the Experience Factory views the project as a source of information and data for model building. It analyzes what it gets and provides feedback to the project for real time project control. It also analyzes the data to build reusable experience models, stores them in an experience base, and provides support for projects based upon these packaged models.

The Experience Factory deals with reuse of all kinds of knowledge and experience. But what makes us think we can be successful with reuse this time, when we have not been so successful in the past? Part of the reason is that we are not talking about reuse of just code in isolation but the reuse of all kinds of experience and context for that experience. The Experience Factory recognizes and provides support for the fact that experience requires the appropriate context definition for it to be reusable and it needs to be identified and analyzed for its reuse potential. It recognizes that experience cannot always be reused as is, it needs to be tailored and it needs to be packaged to make it easy to reuse. In the past, reuse of experience has been too informal, not supported by the organization. It has to be fully incorporated into the development or maintenance process models. Another major issue is that a project's focus is delivery, not reuse, i.e., reuse cannot be a byproduct of software development. It requires a separate organization to support the packaging and reuse of local experience.

The Experience Factory really represents a paradigm shift from current software development thinking. It separates the types of activities that need to be performed by assigning them to different organizations, recognizing that they truly represent different processes and focuses. Project personnel are primarily responsible for the planning and development activities (**Project Organization**) and a separate organization (**Experience Factory**) is primarily responsible for the learning and technology transfer activities. In the **Project Organization,** we focus on problem solving. The processes we perform to solve a problem consist of the decomposition of a problem in to simpler ones, instantiation of higher level solutions into lower level detail, the design and implementation of various solution processes, and activities such as validation and verification. In the **Experience Factory**, we focus on understanding solutions and packaging experience for reuse. The processes we perform are the unification of difference solutions and re-definition of the problem, generalization and formalization of solutions in order to abstract them and make them easy to access and modify, an analysis synthesis process where we understand and abstract, and various experimentation activities so we can learn. These sets of activities are totally different.

2.2 Examples of Packaged Experience in the SEL

The Software Engineering Laboratory (SEL) has been in existence since 1976 and is a consortium

of three organizations: NASA/Goddard Space Flight Center, the University of Maryland, and Computer Sciences Corporation [Mc85],[BaCaMc92]. When it was established, its goals were to (1) understand the software process in a production environment, (2) determine the impact of available technologies, and (3) infuse identified/refined methods back into the development process. The approach has been to identify technologies with potential, apply and extract detailed data in a production environment (experiments), and measure the impact (cost, reliability, quality,..).

Over the years we have learned a great deal and packaged all kinds of experience. We have built
- resource models and baselines, e.g., local cost models, resource allocation models,
- change and defect models and baselines, e.g., defect prediction models, types of defects expected for the application,
- product models and baselines, e.g., actual vs. expected product size, library access, over time,
- process definitions and models, e.g., process models for Cleanroom, Ada waterfall model,
- method and technique models and evaluations, e.g., best method for finding interface faults,
- products and product models, e.g., Ada generics for simulation of satellite orbits,
- a variety of quality models, e.g., reliability models, defect slippage models, ease of change models, and
- a library of lessons learned, e.g., risks associated with an Ada development.

We have used a variety of forms for packaged experience. There are
- 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,
- graphs defining ranges of "normal" ,e.g., graphs of size growth over time with confidence levels,
- specific lessons learned

  associated with project types, phases, activities,

  e.g., reading by stepwise abstraction is most effective for finding interface faults, or

  in the form of risks or recommendations,

  e.g., definition of a unit for unit test in Ada needs to be carefully defined, and
- models or algorithms specifying the processes, methods, or techniques,

  e.g., an SADT diagram defining Design Inspections with the reading technique a variable dependent upon the focus and reader perspective.

These models are used on new projects to help management control development [Va87] and provide the organization with a basis for improvement based upon experimentation with new methods. It is an example of the Experience Factory/Quality Improvement Paradigm (EF/Quality Improvement Paradigm) in practice.

How does the EF/Quality Improvement Paradigm approach work in practice? You begin by getting a commitment. You then define the organizational structure and the associated processes. This means collecting data to establish baselines, e.g., defects and resources, that are process and product independent and measuring your strengths and weaknesses to provide a business focus and goals for improvement, and establish product quality baselines. Using this information about your business, you select and experiment with methods and techniques to improve your processes based upon your product quality needs and evaluate your improvement based upon existing resource and defect baselines. You can define and tailor better and measurable processes, based upon the experience and knowledge gained within your own environment. You must measure for process conformance and domain understanding to make sure that your results are valid. In this

way, you begin to understand the relationship between some process characteristics and product qualities and are able to manipulate some processes to achieve those product characteristics. As you change your processes you will establish new baselines and learn where the next place for improvement might be.

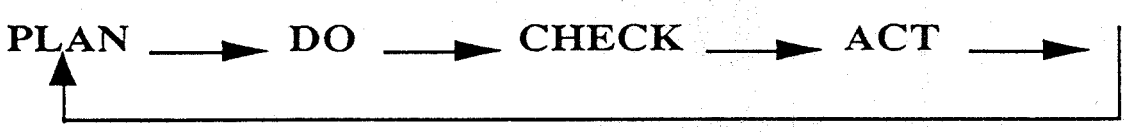
3. Other Improvement Paradigms

Aside from the **Experience Factory /Quality Improvement Paradigm,** there has been a variety of organizational frameworks proposed to improve quality for various businesses. The ones discussed here include:

Plan-Do-Check-Act is a quality improvement process based upon a feedback cycle for optimizing a single process model/production line. **Total Quality Management** represents a management approach to long term success through customer satisfaction based on the participation of all members of an organization. The **SEI Capability Maturity Model** is a 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 (Software) Development** represents a principle supporting the concentration of the production on "value added" activities and the elimination or reduction of "not value added" activities. In what follows, we will try to define these concepts in a little more detail to distinguish and compare them.

3.1 Plan-Do-Check-Act Cycle (PDCA)

The approach is based upon work by W. A. Shewart [Sh31] and was made popular by W. E. Deming [De86]. The goal of this approach is to optimize and improve a single process model / production line. It uses such techniques as feedback loops and statistical quality control to experiment with methods for improvement and build predictive models of the product.



If a family of Processes (P) produces a family of Products (X) then the approach yields a series of versions of product X (each meant to be an improvement of X), produced by a series of modifications (improvements) to the processes P,

$$P_0, P_1, P_2, ..., P_n \text{ ----------> } X_0, X_1, X_2, ..., X_n$$

where P_i, represents an improvement over P_{i-1} and X_i has better quality than X_{i-1}.

The basic procedure involves four basic steps:

Plan: Develop a plan for effective improvement, e.g., quality measurement criteria are set up as targets and methods for achieving the quality criteria are established.

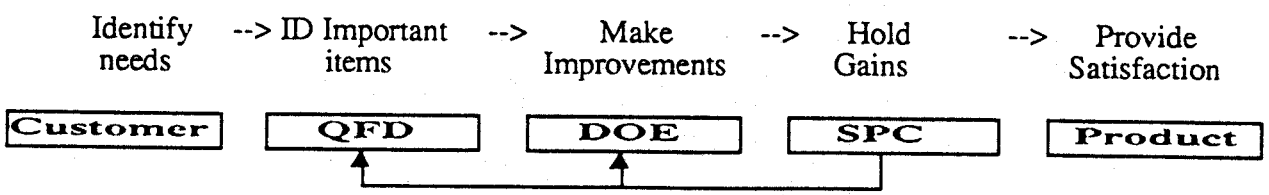
Do: The plan is carried out, preferably on a small scale, i.e., the product is produced by complying with development standards and quality guidelines.

Check: The effects of the plan are observed; at each stage of development, the product is checked against the individual quality criteria set up in the Plan phase.

Act: The results are studied to determine what was learned and what can be predicted, e.g., corrective action is taken based upon problem reports.

3.2 Total Quality Management (TQM)

The term TQM was coined by the Naval Air Systems Command in 1985 to describe its Japanese style management approach to quality improvement [Fe90]. The goal of TQM is to generate institutional commitment to success through customer satisfaction. The approaches to achieving TQM vary greatly in practice so to provide some basis for comparison, we offer the approach being applied at Hughes. Hughes uses such techniques as Quality Function Deployment (QFD), design of experiments (DOE), and statistical process control (SPC), to improve the product through the process.



The approach has similar characteristics to the PDCA approach. If Process (P) --> Product (X) then the approach yields

$$P_0, P_1, P_2, ..., P_n \text{ ----------> } X_0, X_1, X_2, ..., X_n$$

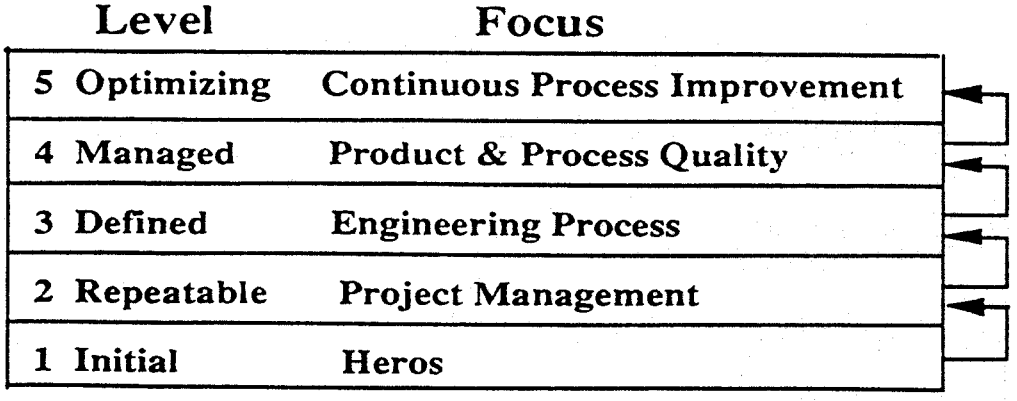
where P_i, represents an improvement over P_{i-1} and X_i provides better customer satisfaction than X_{i-1}.

3.3 SEI Capability Maturity Model (CMM)

The approach is based upon organizational and quality management maturity models developed by R. Likert [Li67] and P. Crosby [Cr80], respectively. A software maturity model was developed by Ron Radice, et. al. [RaHaMuPh85] while he was at IBM. It was made popular by Watts Humphrey [Hu89] at the SEI. The goal of the approach is to achieve a level 5 maturity rating, i.e.,continuous process improvement via defect prevention, technology innovation, and process change management.

As part of the approach, a 5 level process maturity model is defined. A maturity level is defined based on repeated assessment of an organization's capability in key process areas. Thus for example at the lowest level, the focus is on heroes or good people who can rescue a project, at the second level, the focus is on good project management, at the third level, the focus is on good engineering process, etc. Improvement is achieved by action plans for poorly assessed processes.

Thus, if a Process (P) is level i then modify the process based upon the the key processes of the model until the process model is at level i+1.



The SEI has developed a Process Improvement Cycle to support the movement through process levels. Basically is consists of the following activities:

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 prioritorize 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
    Institutionalize
    Document and analyze lessons
    Revise organizational approach

3.4 Lean Software Development

The approach is based upon Lean Enterprise Management, a philosophy that has been used to improve factory output. Womack, et. al. [WoJoRo90], have written a book on the application of lean enterprises in the automotive industry. The goal is to build software using the minimal set of activities needed, eliminating non essential steps, i.e., tailoring the process to the product needs. The approach uses such concepts as technology management, human centered management, decentral organization, quality management, supplier and customer integration, and internationalization/regionalization.

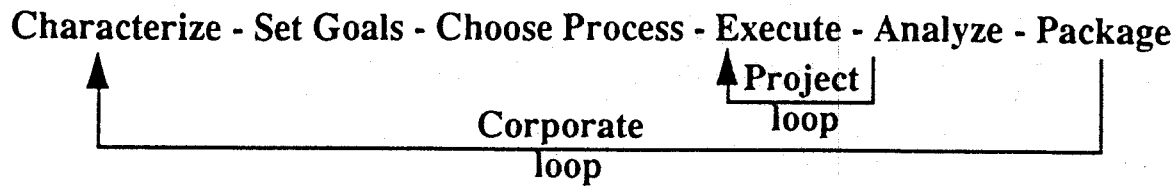
Given the characteristics for product V, select the appropriate mix of sub-processes $p_i, q_j, r_k, \ldots$ to satisfy the goals for V, yielding a minimal tailored process P_V which is composed of $p_i, q_j, r_k, \ldots$

Process (P_V) ----------> Product (V)

The notation P_V is used to imply that the process is dependent on the product characteristics.

3.5 Quality Improvement Paradigm

As stated above, this approach has evolved over 17 years based upon lessons learned in the SEL [Ba85a], [Ba89], [BaRo87], [BaRo88], [BaCaMc92]. Its goal is to build a continually improving organization based upon its evolving goals and an assessment of its status relative to those goals. The approach uses internal assessment against the organizations own goals and status (rather than process areas) and such techniques as GQM, model building, qualitative/quantitative analysis to improve the product through the process.



If Processes (P_X, Q_Y, R_Z, ...) ---> Products (X, Y, Z, ...) and we want to build V, then based upon an understanding of the relationship between P_X, Q_Y, R_Z, ... and X, Y, Z, ... and goals for V we select the appropriate mix of processes p_i, q_j, r_k, ... (i.e., we use experience from different projects) to satisfy the goals for V, yielding a tailored

Process (P_V) ---> Product (V)

4. A Comparison of the Improvement Paradigms

The Quality Improvement Paradigm / Experience Factory Organization is similar to the Plan-Do-Check-Act paradigm in that it is evolutionary, based upon feedback loops, and learns from experiments. It is different in the sense that the Plan-Do-Check-Act paradigm is based upon production, i.e., it attempts to optimize a single process model/production line. In development, we rarely replicate the same thing twice. In production, we can collect a sufficient set of data based upon continual repetition of the same process to develop quantitative models of the process that will allow us to evaluate and predict quite accurately the effects of the single process model. We can use statistical quality control approaches. This is not possible for development, i.e. we must learn form one process about another, so our models are less rigorous and more abstract. Development processes are also more human based. This again affects the building, use, and accuracy of the types of models we can build.

The Quality Improvement Paradigm approach is compatible with TQM in that it can cover goals that are customer satisfaction driven and it is based upon the philosophy that quality is everyone's job. That is, everyone is part of the technology infusion process. Someone can be on the project team on one project and experimenting team on another. All the 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 , e.g., it wasn't taught right, it doesn't fit/interface with other project activities, it needs to be tailored, or it simply doesn't work. You need the user to tell you how to change it. This is consistent with the Quality Improvement Paradigm philosophy that no method is "packaged" that hasn't been tried (applied, analyzed, tailored).

The Quality Improvement Paradigm/EF organization is different from the SEI CMM approach, in that you pull yourself up from the top rather than pushing up from the bottom. At step 1 you start with a level 5 style organization even though you do not yet have level 5 process capabilities. That is, 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 from an external model of process. You make process improvements based upon an understanding of the relationship between process and product in your organization. Technology infusion is motivated by the local problems, so people are more willing to try something new.

But what does a level 5 organization really 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, feed back information for project control, experiment with methods and techniques, evaluate our successes and failures, learn from our experiences, package successful experiences, and reuse successful experiences. This is compatible with the Quality Improvement Paradigm/EF organization.

Quality Improvement Paradigm is not incompatible with the SEI CMM model in that you can still use key process assessments to evaluate where you stand (along with your internal goals, needs, etc.). However, using the Quality Improvement Paradigm, the chances are you will move up the maturity scale faster. You will have more experience early on operating within an improvement organization structure, and you can demonstrate product improvement benefits early.

The Quality Improvement Paradigm approach is most similar to the concepts of Lean Software Development in that it is based upon the ideas of tailoring a set of processes to meet particular problem/product under development. The goal is to generate an optimum set of processes, based upon models of the business and our experience about the relationship between process characteristics and product characteristics.

5. Conclusion

Some important characteristics of the EF/Quality Improvement Paradigm process is that it is iterative, you should converge over time so don't be overly concerned with perfecting any step on the first pass. However, the better your initial guess at the baselines the quicker you will converge. No method is "packaged" that hasn't been tried (applied, analyzed, tailored). Everyone is part of the technology infusion process. Someone can be on the project team on one project and experimenting team on another. Project personnel play the major role in the feedback mechanism. We need to learn from them about the effective use of technology. If they are not using the technology right it can be because they don't understand it or it wasn't taught right, it doesn't fit/interface with other project activities, it needs to be tailored, or 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. And, it is important to evaluate process conformance and domain understanding or you have very little basis for understanding and assessment.

The integration of the Improvement Paradigm, the Goal/Question/Metric Paradigm, and the Experience Factory Organization provides a framework for software engineering development, maintenance, and research. It takes advantage of the experimental nature of software engineering. Based upon our experience in the SEL and other organizations, 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, and feed back experience to current and future projects. It can be applied today and evolve with technology.

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, and a mechanism for improving the organization's ability to develop quality systems productively. The approach is being adopted by several organizations to varying degrees, e.g., Motorola, but it is not a simple solution and it requires long term commitment by top level management.

In summary, the Quality Improvement Paradigm approach provides for a separation of concerns/focus in differentiating between problem solving and experience modeling/packaging. It offers a support for learning and reuse and a means of formalizing and integrating management and development technologies. It allows for the generation of a tangible corporate asset: an experience base of software competencies. It offers a Lean Software Development approach compatible with TQM while providing a level 5 CMM organizational structure. It links focused research with development. Best of all you can start small, evolve and expand, e.g., focus on a homogeneous set of projects or a particular set of packages and build from there.

References

[Ba85a]
V. R. Basili, "Quantitative Evaluation of Software Engineering Methodology," Proc. of the First Pan Pacific Computer Conference, Melbourne, Australia, September 1985 [also available as Technical Report, TR-1519, Dept. of Computer Science, University of Maryland, College Park, July 1985].

[Ba89]
V. R. Basili, "Software Development: A Paradigm for the Future", Proceedings, 13th Annual International Computer Software & Applications Conference (COMPSAC), Keynote Address, Orlando, FL, September 1989

[BaRo87]
V. R. Basili, H. D. Rombach, "Tailoring the Software Process to Project Goals and Environments," Proc. of the Ninth International Conference on Software Engineering, Monterey, CA, March 30 - April 2, 1987, pp. 345-357.

[BaRo88]
V. R. Basili, H. D. Rombach "The TAME Project: Towards Improvement-Oriented Software Environments," IEEE Transactions on Software Engineering, vol. SE-14, no. 6, June 1988, pp. 758-773.

[BaCaMc92]
V. R. Basili, G. Caldiera, F. McGarry, R. Pajerski, G. Page, S. Waligora, "The Software Engineering Laboratory - an Operational Software Experience Factory", International Conference on Software Engineering, May, 1992, pp. 370-381.

[Cr80]
Philip B. Crosby, "Quality is free : the Art of Making Quality Certain," New American Library, New York, 1980.

[De86]
W. Edwards Deming,"Out of the Crisis," MIT Center for Advanced Engineering Study, MIT Press, Cambridge MA, 1986.

[Fe90]
Armand V. Feigenbaum, "Total Quality Control", fortieth anniversary edition, McGraw Hill, NY, 1991.

[Hu89]
Humphrey, Watts S., "Managing the software process," SEI series in software engineering, Addison-Wesley, Reading, Mass., 1989.

[KoAk83]
M. Kogure, Y. Akao, "Quality Function Deployment and CWQC in Japan," Quality Progress, October 1983, pp.25-29.

[Li67]
Rensis Likert, "The human organization: its Management and Value, " McGraw Hill, New York, 1967.

[McRiWa77]
J. A. McCall, P. K. Richards, G. F. Walters, "Factors in Software Quality," RADC TR-77-369, 1977.

[Mc85]
F. E. McGarry, "Recent SEL Studies," Proceedings of the Tenth Annual Software Engineering Workshop, NASA Goddard Space Flight Center, December 1985.

[Sh31]
Walter A. Shewhart, "Economic control of quality of manufactured product," D. Van Nostrand Company, Inc., New York, 1931.

[RaHaMuPh85]
Ron Radice, A. J. Harding, P. E. Munnis, and R. W. Phillips, "A programming process study," IBM Systems Journal, vol.24, no. 2, 1985.

[Va87]
J. D. Valett, "The Dynamic Management Information Tool (DYNAMITE):Analysis of the Prototype, Requirements and Operational Scenarios," M.Sc. Thesis, University of Maryland, 1987.

[WeBa85]
D. M. Weiss, V. R. Basili, "Evaluating Software Development by Analysis of Changes: Some Data from the Software Engineering Laboratory," IEEE Transactions on Software Engineering, vol. SE-11, no. 2, February 1985, pp. 157-168.

[WoJoRo90]
James P. Womack, Daniel T. Jones, Daniel Roos, "The machine that changed the world : based on the Massachusetts Institute of Technology 5-million dollar 5-year study on the future of the automobile," Rawson Associates, New York, 1990.