

N86 - 30358

PANEL #1

**RESEARCH IN THE SOFTWARE ENGINEERING
LABORATORY (SEL)**

**V. Basili, University of Maryland
F. McGarry, NASA/GSFC**

Measuring the Software Process and Product:
Lessons Learned in the SEL

Victor R. Basili
Department of Computer Science
University of Maryland

There are numerous reasons to measure the software development process and product. It is important to create a corporate memory in the software area to support planning, e.g. to answer questions about predicting the cost of a new project. We need to determine the strengths and weaknesses of the current process and product, e.g. to determine what types of errors are commonplace. We need to develop a rationale for adopting and refining software development and maintenance techniques, e.g. to help us decide what techniques actually minimize current problems. We need to assess the impact of the techniques we are using, e.g. to determine whether our current approach to functional testing actually does minimize certain classes of errors, as we might believe it does. Finally, we should evaluate the quality of the software process and product, e.g. to assess the reliability of the product after delivery.

We have tried to address all of these problems to varying degrees within the Software Engineering Laboratory at NASA Goddard Space Flight Center, grouping studies into four general categories: the problem, the process, the product, and the environment. Within these categories, we have concentrated on three aspects of measurement in the SEL: visibility, quality, and technology. With regard to visibility we have tried to better understand how software is being developed by making the current practices and products as visible as possible using measurement. Areas of measurement have been based upon models of the resources, errors, environment, problem and the product. We have tried to assess the quality of the process and product by examining such characteristics as productivity, reliability, maintainability, portability and reusability. Technology has been measured in an attempt to ascertain how much, if at all, certain techniques help in the development and to isolate those practices and tools which improve productivity.

To achieve the goals related to visibility, quality and technology, we have collected a variety of data. Table 1 provides some idea of the type of data collected. The scope of activity in the SEL from 1977 through 1984 is shown in Table 2.

Visibility	Quality	Technology
Resource Data	Productivity	How much do certain techniques help?
Error Data	Reliability	
Environment	Maintainability	Which tools improve productivity?
Characteristics	Portability	
Problem Complexity	Reusability	
Product Data		

Table 1

SEL
1977 - 1984

Number of Projects	41
Number of Source Lines of Code	1.3 million
Development Cost	\$11 million
Number of Data Forms	30 thousand

Table 2

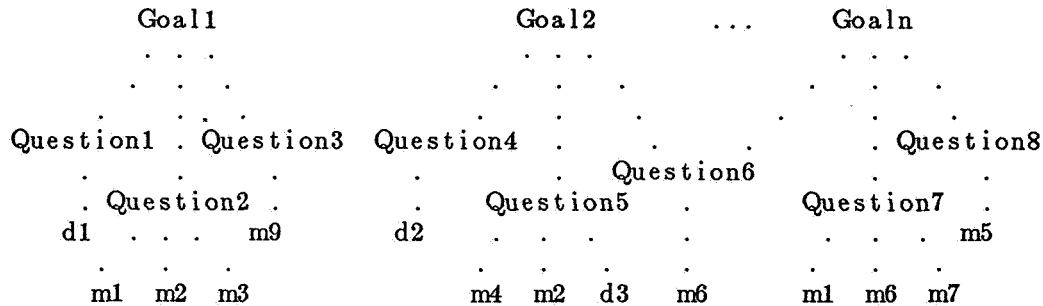
GOAL/QUESTION/METRIC PARADIGM

There have been many lessons learned in the the SEL about measurement but the most important one has been the need for a goal-driven paradigm for data collection. That is data collection must be driven top down. What you measure is based upon a carefully articulated set of goals stating what it is you want to know and whether you can gather the appropriate and valid data needed to answer your questions. Whenever we have violated these rules we either ended up collecting data that was not used or have not been successful in performing our task. For example we have discarded data, such as run analysis data, even though it may be interesting information, it was not associated with a specific goal of the laboratory. Also we have not had success in areas where there was not a carefully focused goal allowing us to control for extraneous effects, e.g. measuring the effectiveness of detailed techniques.

The approach to measurement used in the SEL has been the goal / question / metric paradigm [Basili & Weiss 1984] developed specifically to help us define the areas of study and help in the interpretation of the results of the data collection process. The paradigm does not provide a specific set of goals but rather a framework for stating goals and refining them into specific questions about the software development process and product that provide a specification for the data needed to help answer the goals.

The paradigm provides a mechanism for tracing the goals of the collection process, i.e. the reasons the data are being collected, to the actual data. It is important to make clear at least in general terms the organization's needs and concerns, the focus of the current project and what is expected from it. The formulation of these expectations can go a long way towards focusing the work on the project and evaluating whether the project has achieved those expectations. The need for information must be quantified whenever possible and the quantification analyzed as to whether or not it satisfies the needs. This quantification of the goals should then be mapped into a set of data that can be collected on the product and the process. The data should then be validated with respect to how accurate it is and then analyzed and the results interpreted with respect to the goals.

The actual data collection paradigm can be visualized by a diagram:



Here there are n goals shown and each goal generates a set of questions that attempt to define and quantify the specific goal which is at the root of its goal tree. The goal is only as well defined as the questions that it generates. Each question generates a set of metrics (mi) or distributions

of data (di). Again, the question can only be answered relative to and as completely as the available metrics and distributions allow. As is shown in the above diagram, the same questions can be used to define different goals (e.g. Question6) and metrics and distributions can be used to answer more than one question. Thus questions and metrics are used in several contexts.

Given the above paradigm, the data collection process consists of six steps:

1. Generate a set of goals based upon the needs of the organization.

The first step of the process is to determine what it is you want to know. This focuses the work to be done and allows a framework for determining whether or not you have accomplished what you set out to do. Sample goals might consist of such issues as on time delivery, high quality product, high quality process, customer satisfaction, or that the product contains the needed functionality.

2. Derive a set of questions of interest or hypotheses which quantify those goals.

The goals must now be formalized by making them quantifiable. This is the most difficult step in the process because it often requires the interpretation of fuzzy terms like quality or productivity within the context of the development environment. These questions define the goals of step 1. The aim is to satisfy the intuitive notion of the goal as completely and consistently as possible.

3. Develop a set of data metrics and distributions which provide the information needed to answer the questions of interest.

In this step, the actual data needed to answer the questions are identified and associated with each of the questions. However, the identification of the data categories is not always so easy. Sometimes new metrics or data distributions must be defined. Other times data items can be defined to answer only part of a question. In this case, the answer to the question must be qualified and interpreted in the context of the missing information. As the data items are identified, thought should be given to how valid the data item will be with respect to accuracy and how well it captures the specific question.

4. Define a mechanism for collecting the data as accurately as possible

The data can be collected via forms, interviews, or automatically by the computer. If the data is to be collected via forms, they must be carefully defined for ease of understanding by the person filling out the form and clear interpretation by the analyst. An instruction sheet and glossary of terms should accompany the forms. Care should be given to characterizing the accuracy of the data and defining the allowable error bounds.

5. Perform a validation of the data

The data should always be checked for accuracy. Forms should be reviewed as they are handed in. They should be read by a data analyst and checked with the person filling out the form when questions arise. Sample sets should be set to determine accuracy the data as a whole. As data is entered into the data base, validity checks should be made by the entering program. Redundant data should be collected so checks can be made.

The validity of the data is a critical issue. Interpretations will be made that will effect the entire organization. One should not assume accuracy without justification.

6. Analyze the data collected to answer the questions posed

The data should be analyzed in the context of the questions and goals with which they are associated. Missing data and missing questions should be accounted for in the interpretation.

The process is top down, i.e before we know what data to collect we must first define the reason for the data collection process and make sure the right data is being collected, and it can be interpreted in the right context. To start with a set of metrics is working bottom up and does not provide the collector with the right context for analysis or interpretation.

WRITING GOALS AND QUESTIONS:

In writing down goals and questions, we must begin by stating the purpose of the study. This purpose will be in the form of a set of overall goals but they should follow a particular format.

The format should cover the purpose of the study, the perspective, and any important information about the environment. The format might look like:

Purpose of Study: To (characterize, evaluate, predict, motivate) the (process, product, model, metric) in order to (understand, assess, manage, engineer, learn, improve) it. E.g. To evaluate the system testing methodology in order to assess it.

Perspective: Examine the (cost, effectiveness, correctness, errors, changes, product metrics, reliability, etc.) from the point of view of the (developer, manager, customer, corporate perspective, etc) E.g. Examine the effectiveness from the developer's point of view.

Environment: The environment consists of the following: process factors, people factors, problem factors, methods, tools, constraints, etc. E.g. The product is an operating system that must fit on a PC, etc.

Process Questions:

For each process under study, there are several subgoals that need to be addressed. These include the quality of use (characterize the process quantitatively and assess how well the process is performed), the domain of use (characterize the object of the process and evaluate the knowledge of object by the performers of the process), effort of use (characterize the effort to perform each of the subactivities of the activity being performed), effect of use (characterize the output of the process and the evaluate the quality of that output), and feedback from use (characterize the major problems with the application of the process so that it can be improved).

Other subgoals involve the interaction of this process with the other processes and the schedule (from the viewpoint of validation of the process model).

Product Questions:

For each product under study there are several subgoals that need to be addressed. These include the definition of the product (characterize the product quantitatively) and the evaluation of the product with respect to a particular quality (e.g. reliability, user satisfaction)

The definition of the product consists of:

1. Physical Attributes. e.g. size (source lines, #units, executable lines), complexity (control and data), programming language features, time space.
2. Cost. e.g. effort (time, phase, activity, program)
3. Changes. e.g. errors, faults, failures and modifications by various classes.
4. Context. e.g. customer community, operational profile.

The evaluation is relative to a particular quality e.g. reliability. Thus the physical characteristics need to be analyzed relative to these. Template questions for evaluation include:

How do you measure the quality?

Is the model used valid?

Are the measures used valid?

Are there checks?

Do they agree with the reliability data?

Thus a sample would be:

To evaluate the product (system) in order to assess its quality. Examine the reliability relative to the customer's point of view.

INVESTIGATION LAYOUT

The original goal/question/metric paradigm has been refined with experience [Basili & Selby 1984] to include a step which provides for help in planning the type of investigative analysis possible based upon the scope of the evaluation and the type of data available. Between steps 3 and 4 above is a step to plan the investigation layout and analysis methods. This step is important because it allows the questions to reflect the types of result statements that can be used in the quantitative analysis.

With all the different methods and tools available, we need to better quantitatively understand and evaluate the benefits and drawbacks of each of them. There are several different approaches to quantitatively evaluating methods and tools: blocked subject-project, replicated project, multi-project variation, and single project case study. The approaches can be characterized by the number of teams replicating each project and number of different projects analyzed as shown in Table 3.

		# of projects			
		one	more than one		
# of teams	one	single project	multi-project variation		
per project	more than one	replicated project	blocked subject-project		

Table 3

The blocked subject-project type of analysis allows the examination of several factors within the framework of one study. Each of the technologies to be studied can be applied to a set of projects by several subjects and each subject applies each of the technologies under study. It permits the experimenter to control for differences in the subject population as well as study the effect of the particular projects.

The replicated project analysis involves several replications of the same project by different subjects. Each of the technologies to be studied is applied to the project by several subjects but each subject applies only one of the technologies. It permits the experimenter to establish control groups.

Multi-project variation analysis involves the measurement of several projects where controlled factors such as methodology can be varied across similar projects. This is not a controlled experiment as the previous two approaches were, but allows the experimenter to study the effect of various methods and tools to the extent that the organization allows them to vary on different projects.

The case study is where most methodology evaluation begins. There is a project and the management has decided to make use of some new method or set of methods and wants to know whether or not the method generates any improvement in the productivity or quality. A great deal depends upon the individual factors involved in the project and the methods applied.

The approaches vary in cost and the level of confidence one can have in the result of the study. Clearly, an analysis of several replicated projects costs more money but will generate stronger confidence in the conclusion. Unfortunately, since a blocked subject-project experiment is so expensive, the projects studied tend to be small. The size of the projects increase as the costs go down so it is possible to study very large single project experiments and even multi-project variation experiments if the right environment can be found.

The SEL has had some experience in almost all of these categories. A blocked subject-project study was the comparison of functional testing, structural testing and code reading [Basili & Selby 1985]. Here programs of 145 to 365 lines of code were analyzed by programmers using each of the techniques on different types of applications, e.g. a text formatter, a plotter, an abstract data type, and a database. The goal was to compare the techniques with respect to fault detection effectiveness, fault detection cost, and classes of faults detected. We were also able to compare performance with respect to the software type and the level of expertise of the programmer.

Due to cost, we have only used the replicated project analysis to a limited degree. Here comparisons have been of only two projects, e.g. comparing the development of a dynamic simulator in the standard FORTRAN and Ada [Agresti 1985]. The limitation of only two replicated developments makes the analysis more like a pair of cases studies than a true replicated project analysis. However replicated-project analysis has been used at the University of Maryland to study similar issues to the SEL on a smaller scale, e.g. the effect of a set of software development methods on the process and product [Basili & Reiter 1981], [Basili & Hutchens 1983].

A large number of projects have fit into the multi-project variation category. Various subsets of the 41 projects have been analyzed for a variety of purposes. Studies have been performed to develop and evaluate cost models [Basili & Zelkowitz 1978], [Basili & Beane 1981], [Basili & Freburger 1981], [Bailey & Basili 1981], evaluate the relationships of product and process variables [Basili, Selby & Phillips 1983], [Basili & Selby 1985a], [Basili & Panlilio-Yap 1985], measure productivity [Basili & Bailey 1980], characterize changes and errors [Weiss & Basili 1984], predict problems based upon previous projects [Doerflinger & Basili 1985], and evaluate methodology [Bailey & Basili 1981], [Card, Church & Agresti 1986].

Many projects have been studied in isolation as cases studies, to analyze the effects of changes and errors [Basili & Perricone 1984], to measure the testing approach [Ramsey & Basili 1985], to study the modular structure of programs [Hutchens & Basili 1985].

METHODOLOGY IMPROVEMENT PARADIGM

All this leads us to the following basic paradigm for evaluating and improving the methodology used in the software development and maintenance process [Basili 1985].

1. Characterize the approach/environment.

This step requires an understanding of the various factors that will influence the project development. This includes the problem factors, e.g. the type of problem, the newness to the state of the art, the susceptibility to change, the people factors, e.g. the number of people working on the project, their level of expertise, experience, the product factors, e.g. the size, the deliverables, the reliability requirements, portability requirements, reusability requirements, the resource factors, e.g. target and development machine systems, availability, budget, deadlines, the process and tool factors, e.g. what techniques and tools are available, training in them, programming languages, code analyzers.

2. Set up the goals, questions, data for successful project development and improvement over previous project developments.

It is at this point the organization and the project manager must determine what the goals are for the project development. Some of these may be specified from step 1. Others may be chosen based upon the needs of the organization, e.g. reusability of the code on another project, improvement of the quality, lower cost.

3. Choose the appropriate methods and tools for the project.

Once it is clear what is required and available, methods and tools should be chosen and refined that will maximize the chances of satisfying the goals laid out for the project. Tools may be chosen because they facilitate the collection of the data necessary for evaluation, e.g. configuration management tools not only help project control but also help with the collection and validation of error and change data.

4. Perform the software development and maintenance, collecting the prescribed data and validating it.

This step involves the collection of data by forms, interviews, and automated collection mechanisms. The advantages of using forms to collect data is that a full set of data can be gathered which gives detailed insights and provides for good record keeping. The drawback to forms is that they can be expensive and unreliable because people fill them out. Interview can be used to validate information from forms and gather information that is not easily obtainable in a form format. Automated data collection is reliable and unobtrusive and can be gathered from

program development libraries, program analyzers, etc. However, the type of data that can be collected in this way is typically not very insightful and one level removed from the issue being studied.

5. Analyze the data to evaluate the current practices, determine problems, record the findings and make recommendations for improvement.

This is the key to the mechanism. It requires a post mortem on the project. Project data should be analyzed to determine how well the project satisfied its goals, where the methods were effective, where they were not effective, whether they should be modified and refined for better application, whether more training or different training is needed, whether tools or standards are needed to help in the application of the methods, or whether the methods or tools should be discarded and new methods or tools applied on the next project.

6. Proceed to step 1 to start the next project, armed with the knowledge gained from this and the previous projects.

This procedure for developing software has a corporate learning curve built in. The knowledge is not hidden in the intuition of first level managers but is stored in a corporate data base available to new and old managers to help with project management, method and tool evaluation, and technology transfer.

SEL EXPERIENCE

There are several areas where we believe we have been successful in the measurement area. We have been able to collect reasonably accurate effort data especially with regard to weekly effort hours. The attribution of that effort data to various phases and activities has also been reasonably successful.

We have been successful in extracting realistic histories of the errors and changes on a project but have not been so successful in capturing detailed data on the effectiveness of the various error detection techniques. The latter problem is due to the ad hoc way programmers tend to apply techniques, not always recording all their efforts and to the common use of combinations of techniques. We have been successful in capturing product characteristics but problem characteristics are more difficult to capture. This is largely because they are difficult to quantify and differentiate. We have been able to measure the relative level of the total set of methods used in a project but less effective in isolating the effects of specific methods. This is because most of the studies have been of the multi-project or case study type analysis and it has been difficult to delineate the effects of a specific technique. One successful isolation of techniques was the blocked subject-project study of testing techniques vs. reading.

With regard to the cost of the measurement program in the SEL, the data collection overhead to tasks has been about 3% of total project cost and the processing of the data has been about 5%. It is actually the analysis, interpretation and reporting of the results that have been the most expensive in the SEL. This has been in the order of 15% to 20% but includes all the research support, paper publication, report generation and technology transfer activities.

We have studied the question of what measurement can be automated, i.e. what tools can be used to relieve the impact of measurement on the development or management team. We have automated such things as computer utilization, code and changes growth, product complexity, product characteristics (e.g. size) and source code change count. We have tried to automate but failed with regard to error reporting, weekly resources, and effort by activity. Part of the lack of success has been due to the variation in the development environments, i.e. the use of different mainframes for development, the lack of consistent interactive development across projects. We have not even tried to automate information about the techniques used, resources by component, the environment, changes to the design and specifications, and problem complexity.

We have standardized on various measures of quality in the SEL. Productivity is defined as developed source lines of code (SLOC) per day. Reliability is the number of errors after unit test per 1000 SLOC. Maintainability is the average reported effort to modify or correct the software. Reusability is the percent of components reused on new projects.

RECOMMENDATIONS AND CONCLUSIONS

From our experience within the SEL we would argue that software technology can and should be measured. The measurement overhead to projects should be about 3%. You should not spend excessive effort in trying to automate the data collection process. You should not collect and store data that is not goal driven, i.e. you should collect the minimal set of data needed for the purpose. You should measure top level information for all projects and detailed data for specific experiments. It is difficult to measure the effects of specific techniques in a production environment.

It is best to use the data to characterize the environment, making the problems visible. You should set up both corporate and project goals and use the goal/question/metric paradigm to articulate the process and product needs.

REFERENCES

- [Agresti 1985]
William Agresti and the SEL Staff, Measuring Ada as a Software Development Technology in the SEL, Eighth Minnowbrook Workshop on Software Performance Evaluation, Blue Mountain Lake, New York, July 30, 1985.
- [Bailey & Basili 1981]
John W. Bailey and Victor R. Basili, A Meta-Model for Software Development Resource Expenditures, Proceedings of the Fifth International Conference on Software Engineering, San Diego, California, pp 107-116, 1981.
- [Basili 1985]
Victor R. Basili, Quantitative Evaluation of Software Methodology, Proceedings of the First Pan Pacific Computer Conference, 1985.
- [Basili & Bailey 1980]
Victor R. Basili and John W. Bailey, The Software Engineering Laboratory: Measuring the Effects of Software Methodologies within the Software Engineering Laboratory, Proceedings of the Fifth Annual Software Engineering Workshop, November 1980.
- [Basili & Beane 1981]
Victor R. Basili and John Beane, Can the Parr Curve Help with Manpower Distribution and Resource Estimation Problems?, Journal of Systems and Software, pp 59-69, Volume 2, 1981.
- [Basili & Freburger 1981]
Victor R. Basili and Karl Freburger, Programming Measurement and Estimation in the Software Engineering Laboratory, Journal of Systems and Software, pp 47-57, Volume 2, 1978.
- [Basili & Hutchens 1983]
Victor R. Basili & David H. Hutchens, An Empirical Study of a Syntactic Complexity Family, IEEE Transactions on Software Engineering, pp 664-672, November 1983.
- [Basili & Panlilio-Yap 1985]
Victor R. Basili and N. Monina Panlilio-Yap, Finding Relationships between Effort and other Variables in the SEL, 9th COMPSAC Computer and Software Applications Conference, pp 221-228, October, 1985.
- [Basili & Perricone 1984]
Victor R. Basili and Barry T. Perricone, Software Errors and Complexity: An Empirical Investigation, Communications of the ACM, pp 42-52, January, 1984.
- [Basili & Reiter 1981]
Victor R. Basili and Robert W. Reiter, Jr., A Controlled Experiment Quantitatively Comparing Software Development Approaches, IEEE Transactions on Software Engineering, Vol. SE-7, No. 3, pp 299-320, May 1981.

- [Basili & Selby 1984]
Victor R. Basili and Richard W. Selby, Jr., Data Collection and Analysis in Software Research and Management, Proceedings of the American Statistical Association, pp 21-30, 1984.
- [Basili & Selby 1985]
Victor R. Basili and Richard W. Selby, Jr., Comparing the Effectiveness of Software Testing Strategies, University of Maryland Technical Report TR-1501, May 1985.
- [Basili & Selby 1985a]
Victor R. Basili and Richard W. Selby Jr., Calculation and Use of an Environment's Characteristic Software Metric Set, IEEE Proceedings 8th International Conference on Software Engineering, pp 386-391, August 1985.
- [Basili, Selby & Phillips 1983]
Victor. R. Basili, Richard W. Selby, Tsai-Yun Phillips, Metric Analysis and Data Validation Across FORTRAN Projects, IEEE Transactions on Software Engineering, pp 652-663, November, 1983.
- [Basili & Weiss 1984]
Victor R. Basili and David M. Weiss, A Methodology for Collecting Valid Software Engineering Data, IEEE Transactions on Software Engineering, Vol. SE-10, No. 3, pp 728-738, November 1984.
- [Basili & Zelkowitz 1978]
Victor R. Basili and Marvin V. Zelkowitz, Analyzing Medium Scale Software Development, IEEE 3rd International Conference on Software Engineering, pp 116-123, May 1978.
- [Card, Church & Agresti 1986]
D.N. Card, V. E. Church, and W. W. Agresti, An Empirical Study of Software Design Practices, IEEE Transactions on Software Engineering, pp 264-271, February 1986.
- [Doerflinger & Basili]
Carl W. Doerflinger and Victor R. Basili, Monitoring Software Development Through Dynamic Variables, IEEE Transaction on Software Engineering, pp 978-985, September 1985.
- [Hutchens & Basili 1985]
David H. Hutchens and Victor R. Basili, System Structure Analysis: Clustering with Data Bindings, IEEE Transactions on Software Engineering, pp 749-757, August, 1985.
- [Ramsey & Basili 1985]
James Ramsey and Victor R. Basili, Analyzing the Test Process Using Structural Coverage, 8th International Conference on Software Engineering, pp 306-311, August, 1985.
- [Weiss & Basili 1985]
Evaluating Software Development by Analysis of Changes: Some Data from the Software Engineering Laboratory, IEEE Transactions on Software Engineering, pp 157-168, February 1985.


```
per      *      *
project  * more than * replicated      blocked      *
         *   one     * project        subject-project *
         *      *
*****
```

Table 3

THE VIEWGRAPH MATERIALS
FOR THE
VIC BASILI PRESENTATION FOLLOW

MEASURING THE SOFTWARE PROCESS AND PRODUCT:
LESSONS LEARNED IN THE SEL

VICTOR R. BASILI
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

WHY MEASURE SOFTWARE?

CREATE A CORPORATE MEMORY (SUPPORT PLANNING)

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 CURRENT PROBLEMS?

ASSESS THE IMPACT OF TECHNIQUES

E.G., DOES FUNCTIONAL TESTING MINIMIZE CERTAIN
ERROR CLASSES?

EVALUATE THE QUALITY OF THE PROCESS/PRODUCT

E.G., WHAT IS THE RELIABILITY OF THE PRODUCT AFTER
DELIVERY?

3 ASPECTS OF MEASURES IN THE SEL

VISIBILITY	QUALITY	TECHNOLOGY
RESOURCE DATA	PRODUCTIVITY	HOW MUCH DO CERTAIN TECHNIQUES
ERROR DATA	RELIABILITY	HELP?
ENVIRONMENT CHARACTERISTICS	MAINTAINABILITY	
PROBLEM COMPLEXITY	PORTABILITY	WHICH TOOLS IMPROVE PRODUCTIVITY?
PRODUCT DATA	REUSABILITY	

CLASSES OF PROJECT DATA

RESOURCE DATA

- WEEKLY HRS. BY PERSON
- HOURS BY ACTIVITY
- HOURS BY COMPONENT
- COMPUTER UTILIZATION
-
-
-

ERROR/CHANGE DATA

- SOFTWARE FAILURES
BY TIME
- TYPE
- MAGNITUDE
- SOFTWARE CHANGES
- DESIGN CHANGES
- SPECIFICATION CHANGES

PROJECT CHARACTERISTICS

- PHASE DATES
- SIZE (LINES OF CODE, COMPONENTS)
- SIZE BY TIME
- ENVIRONMENT
- PROBLEM COMPLEXITY
- PRODUCT COMPLEXITY
-
-
-

'TECHNIQUES'

- TOOLS USED
- TECHNIQUES APPLIED
- ORGANIZATION STRUCTURE(IV&V, CPT, ...)
- TRAINING PROVIDED
-
-

PROJECTS STUDIED IN SEL

1977-1984

NUMBER OF PROJECTS	41
TOTAL PROJECTS SIZE	1.3 M SLOC
DEVELOPMENT COST	\$11M (84 DOLLARS)
DATA 'FORMS' COLLECTED	30,000

MAJOR LESSON LEARNED

DEVELOP A GOAL-DRIVEN PARADIGM FOR DATA COLLECTION

EVIDENCE

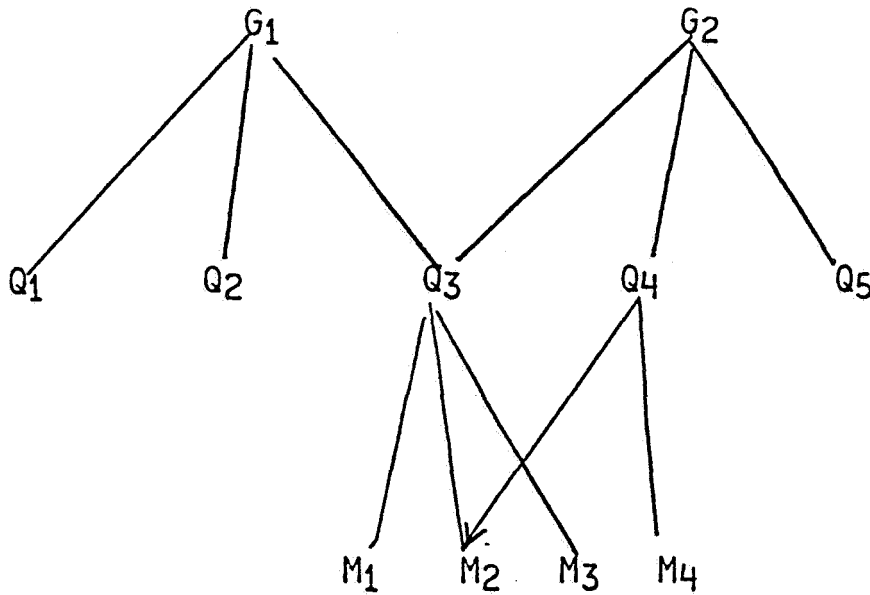
HAVE THROWN AWAY DATA GATHERED BOTTOM UP, E.G.

RUN ANALYSIS

HAVE NOT HAD SUCCESS IN AREAS WHERE THERE WAS NOT
APPROPRIATE FOCUS, E.G., EFFECTIVENESS OF DETAILED

TECHNIQUES

GOAL/QUESTION/METRIC PARADIGM



MANAGEMENT-ORIENTED GOAL
(CHARACTERIZE ERRORS)

SPECIFIC QUESTION
OR HYPOTHESIS
(WHAT PHASE WAS GREATEST
SOURCE OF ERRORS?)

QUANTITATIVE METRIC
OR DISTRIBUTION
(ERROR DISTRIBUTION BY PHASE)

	Q ₁	Q ₂	Q ₃	...	Q _M
G ₁	M ₁ .M ₂		M ₁ .M ₂		
			M ₃		
G ₂				M ₂ .M ₁	
G ₃					
⋮					⋮
⋮					⋮
⋮					⋮
G _N				...	

SEL
DATA COLLECTION METHODOLOGY

1. ESTABLISH THE GOALS OF DATA COLLECTION; E.G.,
CHARACTERIZE CHANGES DURING SOFTWARE DEVELOPMENT.
2. DEVELOP A LIST OF QUESTIONS OF INTEREST; E.G.,
WHAT PERCENTAGE OF THE CHANGES WERE MODIFICATIONS
AND ERRORS?
3. DETERMINE THE METRICS AND DISTRIBUTIONS NEEDED TO
ANSWER THE QUESTIONS.
4. DESIGN AND TEST DATA COLLECTION FORM.
5. COLLECT AND VALIDATE DATA.
6. ANALYZE AND INTERPRET THE DATA

SAMPLE GOALS

ON TIME DELIVERY

HIGH QUALITY PRODUCT

HIGH QUALITY PROCESS

CONTAINS NEEDED FUNCTIONALITY

SALABLE PRODUCT

CUSTOMER SATISFACTION

CHARACTERIZE ERRORS AND CHANGES TO LEARN

FROM THIS PROJECT

LOW COST

TIMELINESS

. . .

CHARACTERIZING GOALS

1. CHARACTERIZE RESOURCE USAGE ACROSS THE PROJECT
2. CHARACTERIZE CHANGES AND ERRORS ACROSS LIFE CYCLE
3. CHARACTERIZE THE DIMENSIONS OF THE PROJECT
4. CHARACTERIZE THE EXECUTION TIME ASPECTS
5. CHARACTERIZE THE ENVIRONMENT

- QUALITY GOALS
 - PRODUCTIVITY GOALS
 - MAINTENANCE GOALS
 - TOOL AND METHOD EVALUATION GOALS
 - COST-ESTIMATION GOALS
- ETC.

Quantitative Analysis Methodology

- Methodology for data collection & quantitative analysis
 1. **Formulate** goals
 2. **Develop** and refine subgoals & questions
 3. **Establish** appropriate metrics
 4. **Plan** investigation layout & analysis methods
 5. **Design** & test data collection scheme
 6. **Perform** investigation concurrently w/ data validation
 7. **Analyze** data
- Goal/question/metric paradigm defines analysis purpose, required data, and context for interpretation
- Questions are coupled with measurable attributes and reflect the types of result statements from quantitative analysis
- Identifies aspects of a well-run analysis
- Intended to be applied to different types of studies from a variety of problem domains

Analysis Classification: Scopes of Evaluation

#Teams per Project	#Projects	
	One	More Than One
One	Single Project	Multi-Project Variation
More Than One	Replicated Project	Blocked Subject-Project

GOAL SETTING TEMPLATE

PURPOSE OF STUDY:

TO (CHARACTERIZE, EVALUATE, PREDICT, MOTIVATE) THE
(PROCESS, PRODUCT, METRIC) IN ORDER TO (UNDERSTAND,
ASSESS, MANAGE, ENGINEER, LEARN, IMPROVE, COMPARE) IT
E.G., TO EVALUATE THE SYSTEM TEST METHODOLOGY IN ORDER
TO ASSESS IT.

PERSPECTIVE:

EXAMINE THE (COST, EFFECTIVENESS, RELIABILITY, CORRECTNESS,
MAINTAINABILITY, EFFICIENCY, ETC.) FROM THE POINT OF
VIEW OF THE (DEVELOPER, MANAGER, CUSTOMER, CORPORATION,
ETC.)
E.G., EXAMINE THE EFFECTIVENESS FROM THE DEVELOPER'S POINT
OF VIEW.

ENVIRONMENT:

LIST THE VARIOUS PROCESS FACTORS, PROBLEM FACTORS, PEOPLE
FACTORS, ETC.

HIERARCHY OF PERSPECTIVES

<u>DOMAIN</u>	<u>CONCERNS</u>
1) INDUSTRY-WIDE	- TECHNOLOGICAL CAPABILITY, INTERNATIONAL COMPETITION
2) CORPORATE	- PROFIT, MARKET POSITION
3) UNIT MANAGEMENT	- RESOURCE ALLOCATION
4) PROJECT MANAGEMENT	- PROGRESS AGAINST MILESTONES
5) PROJECT TEAM	- INTEGRATION OF INDIVIDUAL PRODUCTS
6) INDIVIDUAL	- PRODUCT QUALITY, WORK RATE

GOAL AREA: PROCESS QUALITY

PURPOSE:

PERSPECTIVE:

ENVIRONMENT:

DEFINITION OF THE PROCESS:

QUALITY OF USE

DOMAIN OF USE

KNOWLEDGE OF DOMAIN

VOLATILITY OF DOMAIN

COST OF USE

EFFECTIVENESS OF USE

RESULTS

QUALITY OF RESULTS

FEEDBACK FROM USE

LESSONS LEARNED

MODEL VALIDATION

INTEGRABILITY WITH OTHER TECHNIQUES

EXAMPLE

PURPOSE OF STUDY: TO EVALUATE THE SYSTEM TEST
METHODOLOGY IN ORDER TO ASSESS IT'S EFFECT

PERSPECTIVE: EXAMINE THE EFFECTIVENESS FROM THE
DEVELOPER'S POINT OF VIEW

DEFINITION OF PROCESS:

1. QUALITY OF USE

1.1 HOW MANY REQUIREMENTS ARE THERE?

1.2 WHAT IS THE DISTRIBUTION OF TESTS OVER
REQUIREMENTS?

NUMBER OF TESTS/REQUIREMENT

1.3 WHAT IS THE IMPORTANCE OF TESTING EACH
REQUIREMENT?

RATE 0-5

1.4 WHAT IS THE COMPLEXITY OF TESTING EACH
REQUIREMENT?

RATE 0-5

SUBJECTIVE

FANOUT TO COMPONENTS AND/OR NAMES

1.5 IS Q1.2 CONSISTENT WITH Q1.3 AND Q1.4?

EXAMPLE (CONT'D)

2. DOMAIN OF USE

KNOWLEDGE:

2.1 HOW PRECISELY WERE THE TEST CASES KNOWN
IN ADVANCE?

RATE 0-5

2.2 HOW CONFIDENT ARE YOU THAT THE RESULT IS
CORRECT?

VOLATILITY:

2.3 ARE TESTS WRITTEN/CHANGED CONSISTENT WITH
Q1.3 AND Q1.4?

2.4 WHAT PERCENT OF THE TESTS WERE RERUN?

3. COST OF USE

3.1 COST TO MAKE A TEST

3.2 COST TO RUN A TEST

3.3 COST TO CHECK A RESULT

3.4 COST TO ISOLATE THE FAULT

3.5 COST TO DESIGN AND IMPLEMENT A FIX

3.6 COST TO RETEST

EXAMPLE (CONT'D)

4. EFFECTIVENESS OF USE

QUALITY OF RESULTS

- 4.1 HOW MANY FAILURES WERE OBSERVED?
- 4.2 WHAT PERCENT OF TOTAL ERRORS WERE FOUND?
- 4.3 WHAT PERCENT OF THE DEVELOPED CODE WAS EXERCISED?
- 4.4 WHAT IS THE STRUCTURAL COVERAGE OF THE ACCEPTANCE TESTS?

RESULTS:

- 4.5 HOW MANY ERRORS WERE DISCOVERED DURING EACH PHASE OF DEVELOPMENT ANALYZED BY CLASS OF ERROR AND IN TOTAL?
- 4.6 WHAT IS THE NUMBER OF FAULTS PER LINE OF CODE AT THE END OF EACH PHASE? ONE MONTH, SIX MONTHS, ONE YEAR?
- 4.7 WHAT IS THE COST TO FIX AN ERROR ON THE AVERAGE AND FOR EACH CLASS OF ERROR AT EACH PHASE?
- 4.8 WHAT IS THE COST TO ISOLATE AN ERROR ON THE AVERAGE AND FOR EACH CLASS OF ERROR AT EACH PHASE?

GOAL AREA: HIGH QUALITY PRODUCT

PRODUCT:

PURPOSE OF STUDY:

ENVIRONMENT:

DEFINITION OF PRODUCT:

PHYSICAL ATTRIBUTES

COST

CHANGES AND ERRORS

CONTEXT

CUSTOMER COMMUNITY

OPERATIONAL PROFILES

PERSPECTIVE:

MAJOR MODEL(S) USED:

VALIDITY OF THE MODEL FOR THE PROJECT

VALIDITY OF THE DATA COLLECTED

MODEL EFFECTIVENESS

SUBSTANTIATION OF THE MODEL

IMPROVING METHODOLOGY, PRODUCTIVITY AND QUALITY
THROUGH PRACTICAL MEASUREMENT

1. CHARACTERIZE THE ENVIRONMENT
2. SET UP THE GOALS FOR IMPROVEMENT
E.G., HIGHER QUALITY, LOWER COST, ON-TIME DELIVERY
3. REFINE AND ADJUST APPROACH/ENVIRONMENT TO
SATISFY THE GOALS
4. BUILD THE SYSTEM, COLLECT AND VALIDATE THE DATA
5. INTERPRET AND ANALYZE THE DATA TO CHECK IF THE
GOALS ARE SATISFIED
EVALUATE METHODOLOGY, PRODUCTIVITY AND QUALITY, ETC.
6. GO TO STEP 1, ARMED WITH NEW KNOWLEDGE

SEL SUCCESSES/FAILURES

EFFORT DATA

- WEEKLY EFFORT HOURS CAN BE ACCURATELY CAPTURED
- EFFORT BY PHASE AND ACTIVITY CAN BE IMPROVED

ERROR/CHANGE DATA

- CAN EXTRACT REALISTIC HISTORY OF ERRORS AND CHANGES
- CANNOT CAPTURE DETAILED TECHNIQUE INFORMATION
(FOR ERROR DETECTION)

PROJECT CHARACTERISTICS

- PRODUCT CHARACTERISTICS CAN BE ACCURATELY CAPTURED
- PROBLEM CHARACTERISTICS DIFFICULT TO CAPTURE

TECHNIQUES

- CAN MEASURE RELATIVE LEVEL OF TOTAL METHODOLOGY
- DIFFICULT TO ISOLATE EFFECTS OF SPECIFIC METHODS

COST OF DATA COLLECTION

- ° OVERHEAD TO TASKS DOES NOT HAVE TO EXCEED 3%

- ° PROCESSING OF DATA CAN BE CUT TO 5%

- ° ANALYSIS, INTERPRETATION AND REPORTING
 - MOST EXPENSIVE
 - ° 15 - 20% IN SEL
 - ° INCLUDES RESEARCH SUPPORT
 - PAPER PUBLICATION
 - TECHNOLOGY TRANSFER

CAN WE AUTOMATE* MEASUREMENT?

	RESOURCE MEASURES	ERROR/CHANGE DATA	CHARACTERISTICS	'METHODOLOGY'
MANUAL	EFFORT BY PHASE WEEKLY HOURS EFFORT BY COMPONENT	ERROR DATA CHANGE INFO ◦ WEEKLY CHANGES TO SOURCE CODE ◦ GROWTH OF SOURCE	PHASE DATES ◦ ◦ ◦ CODE COMPLEXITY	TOOLS USED TECHNIQUES ◦ ◦ ◦
AUTOMATED	COMPUTER RESOURCE			

AUTOMATED

- COMPUTER UTILIZATION
- CODE CHANGES/GROWTH
- PRODUCT COMPLEXITY
- PRODUCT CHARACTERISTICS (SIZE...)
- SOURCE CODE CHANGE COUNT

NOT AUTOMATED

TRIED/BUT FAILED

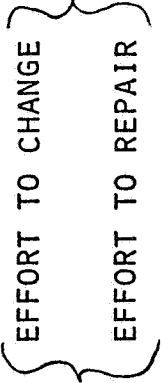
- ERROR REPORTING
- WEEKLY RESOURCES
- EFFORT BY ACTIVITY

ALWAYS MANUAL

- TECHNIQUES USED
- RESOURCES BY COMPONENT
- ENVIRONMENT
- CHANGES TO DESIGN/SPECS
- PROBLEM COMPLEXITY

* IMPLIES TOOL OR PROCEDURE TO RELIEVE ANY IMPACT TO DEVELOPMENT TEAM OR MANAGEMENT

MEASURES OF SOFTWARE 'QUALITY' IN SEL

- PRODUCTIVITY
DEVELOPED SLOC/PERSON DAY
- RELIABILITY
NUMBER OF ERRORS (AFTER UNIT TEST)1000 SLOC
- MAINTAINABILITY


EFFORT TO CHANGE
EFFORT TO REPAIR

AVERAGE REPORTED EFFORT TO MODIFY (CORRECT)SOFTWARE
- REUSABILITY
% OF COMPONENTS REUSED ON NEW PROJECTS

SPECIFIC SEL EXPERIENCE

- SOFTWARE (TECHNOLOGY) CAN AND SHOULD BE MEASURED
- MEASUREMENT OVERHEAD TO DEVELOPMENT PROJECTS ABOUT 3+%
- DON'T SPEND EXCESSIVE EFFORT IN TRYING TO AUTOMATE COLLECTION
- DO NOT COLLECT/STORE 'DATA' THAT IS NOT GOAL-DRIVEN
(COLLECT MINIMUM SET OF DATA)
- DIFFICULT TO MEASURE SPECIFIC SOFTWARE TECHNIQUES
- MEASURE TOP LEVEL INFORMATION FOR ALL PROJECTS - DETAILED DATA FOR
SPECIFIC EXPERIMENTS (E.G., TESTING TECHNIQUES)

OVERALL RECOMMENDATION

USE DATA TO CHARACTERIZE THE ENVIRONMENT, MAKING
PROBLEMS VISIBLE

SET UP CORPORATE AND PROJECT GOALS AND USE
GOAL/QUESTION/DATA PARADIGM TO ARTICULATE
PROCESS AND PRODUCT NEEDS