CMSC 735:
A Quantitative Approach to Software Management and Engineering

Victor R. Basili

Experimental Software Engineering Group

Institute for Advanced Computer Studies

Department of Computer Science

University of Maryland

and

Fraunhofer Center for Experimental Software Engineering - Maryland

CMSC 735

A Quantitative Approach to Software Management and Engineering

Outline

Introduction

- Experimentation, Modeling, Evolution of Knowledge

Software Models and Measures

- Resource, Change, Defect, Process, Product Models and Metrics

Measurement and Organizational Frameworks

- Goal Driven Measurement, Evolutionary Learning

Experimentation

- Experimental Methods, Threats to Validity, Building Knowledge

Example Studies

Evolving Knowledge
Model Building, Experimenting, and Learning

Understanding a discipline involves **building models,**
e.g., application domain, problem solving processes

Checking our understanding is correct involves

- testing our models
- **experimentation**

Analyzing the results of the experiment involves **learning,** the **encapsulation of knowledge** and the ability to change and refine our models over time

The understanding of a discipline evolves over time

Knowledge encapsulation allows us to deal with higher levels of abstraction

This is the paradigm that has been used in many fields,
e.g., physics, medicine, manufacturing.

Evolving Knowledge
Model Building, Experimenting, and Learning

What do these fields have in common?

They evolved as disciplines when they began applying the cycle of model building, experimenting, and learning

Began with observation and the recording of what was observed

Evolved to manipulating the variables and studying the effects of change in the variables

What are the differences of these fields?

Differences are in the objects they study, the properties of those object, the properties of the system that contain them, the relationship of the object to the system, and the culture of the discipline

This effects

how the models are built

how the experimentation gets done

Evolving Knowledge
Model Building, Experimenting, and Learning

Physics

- understand and predict the behavior of the physical universe
- researchers: theorists and experimentalists
- has progressed because of the interplay between the groups

Theorists build models to explain the universe

- predict the results of events that can be measured
- models based on
  - theory about the essential variables and their interaction
  - data from prior experiments

Experimentalists observe, measure, experiment to

- test or disprove a hypothesis or theory
- explore a new domain

But at whatever point the cycle is entered there is a modeling, experimenting, learning and remodeling pattern

Early experimentalists only observed, did not manipulate the objects
Modern physicists have learned to manipulate the physical universe, e.g. particle physicists.

Evolving Knowledge
Model Building, Experimenting, and Learning

Medicine
- researcher and practitioner
- clear relationship between the two
- knowledge built by feedback from practitioner to researcher

Researcher aims at understanding the workings of the human body to predict effects of various procedures and drugs

Practitioner applies knowledge by manipulating processes on the body for the purpose of curing it

Medicine began as an art form
- evolved as a field when it began observation and model building

Experimentation
- from controlled experiments to case studies
- human variance causes problems in interpreting results
- data may be hard to acquire

However, our knowledge of the human body has evolved over time

Evolving Knowledge
Model Building, Experimenting, and Learning

Manufacturing
- domain researcher and manufacturing researcher
- understand the process and the product characteristics
- produce a product to meet a set of specifications

Manufacturing evolved as a discipline when it began process improvement

Relationship between process and product characteristics
- well understood

Process improvement based upon models of
- problem domain and solution space
- evolutionary paradigm of model building, experimenting, and learning
- relationship between the three

Models are built with good predictive capabilities
- same product generated, over and over, based upon a set of processes
- understanding of relationship between process and product

Engineering Discipline Requirements

The application of a **successful engineering discipline** requires:

- A combination of technical and managerial solutions
- A well defined set of product needs
  - to satisfy the customer
  - to assist the developer in accomplishing those needs
  - to create competencies for future business
- A well defined set of processes
  - to accomplish what needs to be accomplished
  - to control development
  - to improve the business
- A closed loop process that supports learning and feedback
- Key technologies for supporting these needs include:
  - modeling, measurement, reuse of processes, products, and other forms of knowledge relevant to the discipline

Engineering Discipline Requirements

Understand process and product

we must model the elements of the discipline

Define process and product qualities

we must define and model the characteristics of the elements

Evaluate successes and failures

we must evaluate whether the elements satisfy the models in practice

Feedback information for project control

we must have a closed loop process to learn

Learn from our experiences

each application of the discipline should provide information that allows us to evolve the discipline

Package successful experiences

we must build models and other abstractions that represent our current knowledge

Engineering Discipline Elements

Understanding (Model Assumptions)

There are factors that create similarities and differences among projects

one model does not work in all situations

There is a direct relationship between process and product

we must choose the right processes to create the desired product characteristics

Measurement is necessary and must be based on the appropriate goals and models

appropriate measurement provides visibility and definition

Evaluation and feedback are necessary for project control

we need a closed loop process for learning

Engineering Discipline Elements

Packaging Experience (Building Models)

Experience needs to be packaged

we must build models in software

Experiences must be evaluated for reuse potential

an analysis processes is required

Software development and maintenance processes must support reuse of experience

we must say how and when to reuse

A variety of experiences can be packaged

we can build process, product, resource, defect, and quality models

Experiences can be packaged in a variety of ways

we can use equations, histograms, algorithms

Packaged experiences need to be integrated

we need an experience base of integrated information

Engineering Discipline Elements

Continuous Improvement (Evolving models)

Software development follows an experimental paradigm

learning and feedback are natural activities for software development and maintenance

Process, product, knowledge, and quality models need to be better defined and tailored

we need evolving definitions of the components of the software business

Evaluation and feedback are necessary for learning

we need a closed loop for long range improvement

New technologies must be continually introduced

we need to experiment with technologies

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

reuse of knowledge is the basis of improvement

What is software and software engineering?

Software

part of a system that can be encoded to execute on a computer as a set of instructions; it includes all the associated documentation necessary to understand, transform and use that solution

the collection of computer programs, procedures, rules, and associated documentation and data (IEEE)

Software engineering

the disciplined development and evolution of software systems based upon a set of principles, technologies, and processes

the systematic approach to the development, operation, maintenance, and retirement of software (IEEE)

the application of science and mathematics by which the capabilities of computer equipment are made useful to man via computer programs, procedures, and associated documentation (Boehm)

the application and tailoring of techniques, methods, and life cycle models to the software problem, project, and organization

Software Engineering
The Nature of the Discipline

Like other disciplines, software engineering requires the cycle of model building, experimentation, and learning

Software engineering is a **laboratory science**

The **researcher's role** is to understand the nature of the processes, products and the relationship between the two in the context of the system

The **practitioner's role** is to build "improved" systems, using the knowledge available

More than the other disciplines these **roles are symbiotic**

The researcher needs laboratories to observe and manipulate the variables
- they only exist where practitioners build software systems

The practitioner needs to better understand how to build better systems
- the researcher can provide models to help

Software Engineering
The Nature of the Discipline

Software engineering is **development** not production

The technologies of the discipline are **human based**

Software is **inherently complex** to build and understand
- part of the system solution we least understand
- often something new
- requirement for change/evolution of function or structure

All software is not the same
- there are a **large number of variables** that cause differences
- their effects need to be understood

Currently,
- **insufficient set of models** that allow us to **reason** about the discipline
- **lack of recognition of the limits** of technologies for certain contexts
- there is **insufficient analysis and experimentation**

What are the problems of interest in software engineering?

Practitioners want

- the ability to control and manipulate project solutions
  - based upon the environment and goals set for the project
- knowledge based upon empirical and experimental evidence
  - of what works and does not work and under what conditions

Researchers want to understand

- the basic elements of the discipline, e.g., products, processes, and their characteristics (build realistic models)
- the variables associated with the models of these elements
- the relationships among these models

Researchers need laboratories for experimentation

This will require a research plan that will take place over many years

- coordinating experiments
- evolving with new knowledge

What is Empirical Software Engineering?

Empirical software engineering requires the scientific use of quantitative and qualitative data to understand and improve the software product, software development process and software management

Empirical means "based on observation"

Mary Shaw differentiated it from other techniques that can be used to validate research results, such as:

- Persuasion
- Implementation (existence proof)
- Analysis

Mary Shaw, "The Coming-of-Age of Software Architecture Research" (keynote address). Proceedings of the 23rd International Conference on Software Engineering, Toronto, Canada, IEEE Computer Society, 2001, pp. 656-664a

Misconceptions About Empirical Study

- Commonly (but wrongly) understood to mean “controlled experiment using lots of quantitative data”
  - The type of information collected and the level of rigor needs to be tailored for researcher goals and the level of maturity of the technology

- Empirical studies are not “one-shot deals.” Studies on live development projects are not the only ones that matter.
  - Basic Premise: Software engineering is a laboratory science.
  - Understanding our discipline involves observation, reflection, model building, experimentation followed by iteration
  - Almost all of our studies are exploratory, NOT confirmatory – they create an opportunity for dialogue, evolving hypotheses, etc.
  - It is the best way to evolve our knowledge, understand the parameters and context variables, learn

Misconceptions About Empirical Study

- Overall purpose
  - Is NOT to be a yes/no certification of the technology
  - IS to yield insights and answers that can
    - » Assist evolution of technology
    - » Find the appropriate environments for its use

- "We ran a study of technology X and now we have some evidence that …"
  - Technology X doesn't work. (NO!)
  - Technology X performed worse than technology Y in our environment. (Better)
    - » "Environment" includes people & their expertise, project goals, etc.
    - » Measuring performance implies we decided on some metric that we felt was an important indicator
- No solution is really expected to be better for all users under all conditions.

Outputs Of Empirical Study

- Empirical study can help to provide information of interest to teams that might eventually adopt a technology:
  - Does it work better for certain types of people?
    - » Novices: It's a good solution for training
    - » Experts: Users need certain background knowledge…
  - Does it work better for certain types of systems?
    - » Static/dynamic aspects, complexity
    - » Familiar / unfamiliar domains
  - Does it work better in certain development environments?
    - » Users [did/didn't] have the right documents, knowledge, amount of time… to use it.

Software Engineering
Early Observation

Belady & Lehman ('72,'76)

- **observed** the behavior of OS 360 with respect to releases
- posed theories based on observation concerning entropy

The idea that you might redesign a system rather than continue to change it was a revelation

But, Basili & Turner ('75)

- **observed** a compiler system
- developed using an incremental development approach
- gained structure over time, rather than lost it

How can these **seemingly** opposing statements be true?

What were the variables that caused the effects to be different?

Size, methods, nature of the changes, context?

Software Engineering
Early Observation

Walston and Felix ('79) identified **29** variables that had an effect on software productivity in the IBM environment

Boehm ('81) observed that **15** variables seemed sufficient to explain/predict the cost of a project across several environments

Bailey and Basili ('81) identified **2** composite variables that when combined with size were a good predictor of effort in the SEL environment

There are numerous cost models with different variables

Why were the variables different?

What does the data tell us about the relationship of variables?

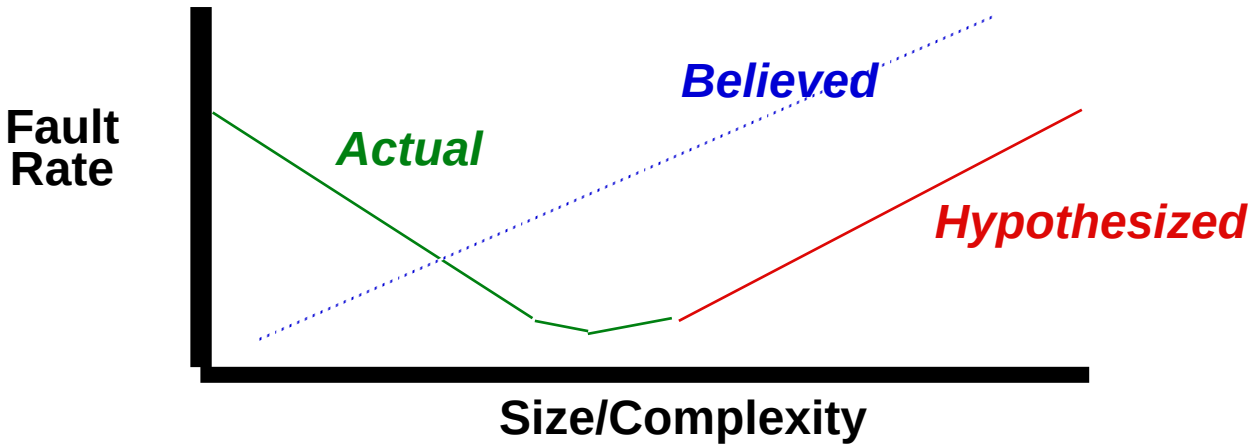
Which variable are relevant for a particular context?

What determines their relevance?

What are the ranges of the values variables and their effects?

Software Engineering
Early Observation

Basili & Perricone (‘84) **observed** that the defect rate of modules shrunk as module size and complexity grew in the SEL environment



Seemed counter to folklore that smaller modules were better, **but**

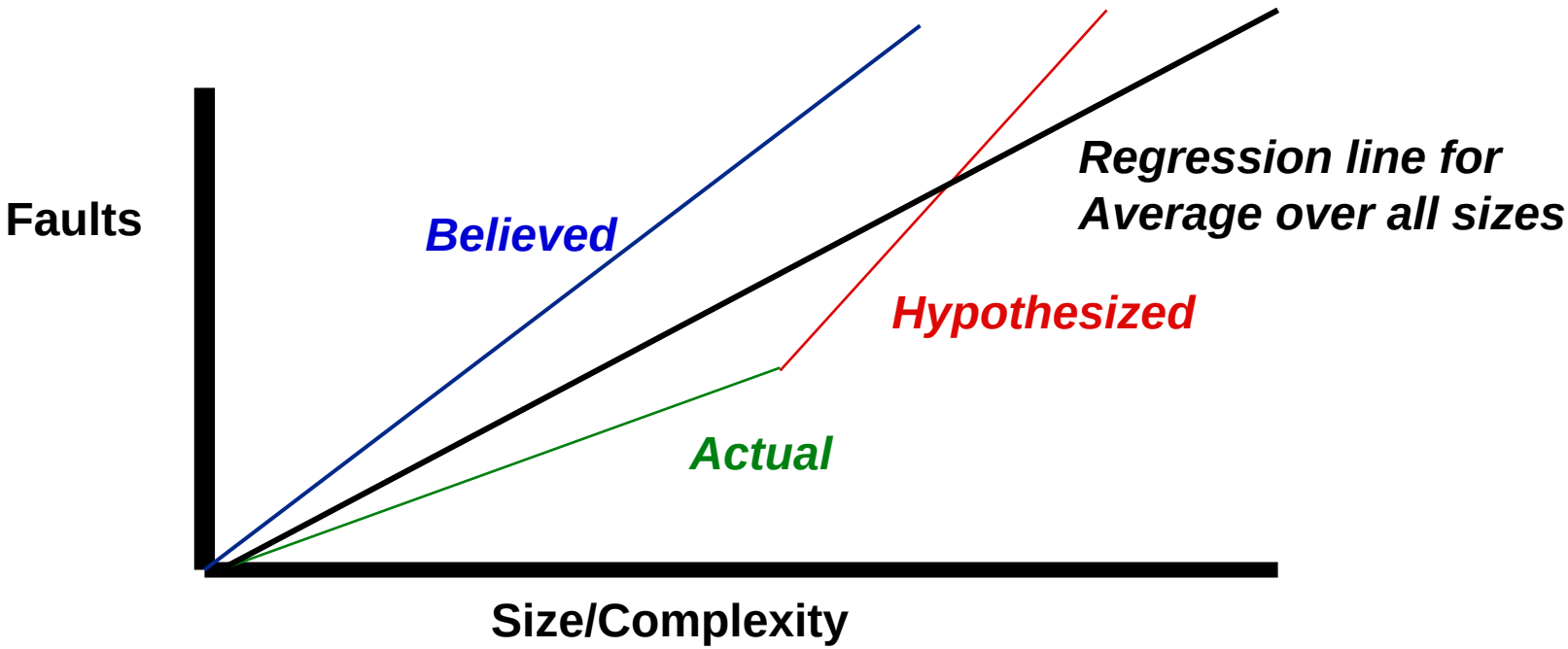
- interface faults dominate
- developer tend to shrink size when they lose control

This result has been observed by numerous other organizations

But **defect rate is only one dependent variable**

What is the effect on other variables? What size minimizes the defect rate?

Software Engineering
Early Observation



Available Research Paradigms?

The **analytic paradigm**:

- propose a formal theory or set of axioms
- develop a theory
- derive results and
- if possible, verify the results with empirical observations.

Experimental paradigm:

- observing the world (or existing solutions)
- proposing a model or a theory of behavior (or better solutions)
- measuring and analyzing
- validating hypotheses of the model or theory (or invalidate
- repeating the procedure evolving our knowledge base

The experimental paradigms involve

- experimental design
- observation
- quantitative or qualitative analysis
- data collection and validation on the process or product being studied

Available Research Paradigms?

Quantitative Analysis
- obtrusive controlled measurement
- objective
- verification oriented

Qualitative Analysis
- naturalistic and uncontrolled observation
- subjective
- discovery oriented

Study
- an act to discover something unknown or of testing a hypothesis
- can include all forms of quantitative and qualitative analysis

Studies can be
- **experimental**
    - driven by hypotheses; quantitative analysis
    - controlled experiments
    - quasi-experiments or pre-experimental designs (e.g.,X,O)
- **observational**
    - driven by understanding; qualitative analysis dominates
    - qualitative/quantitative study
    - pure qualitative study

The Status of Model Building

Modeling research

- software product
  - mathematical models of the program function
  - product characteristics, such as reliability models
- variety of process notations
- cost models, defect models

Little experimentation

- implementation yes, experimentation no

Why? Model builders

- theorists, expect the experimentalists to test the theories
- view their "models" as self evident, not needing to be tested

For any technology, questions of interest include:

- Can it be applied by a practitioner?
- Under what conditions its application is cost effective?
- What kind of training is needed for its successful use?

What is the effect of the technique on product reliability, given an environment of expert programmers in a new domain, with tight schedule constraints, etc.?

The Status of the Experimental Discipline

Where are we in the spectrum of model building, experimentation, and learning in the software engineering discipline?

These have been formulated as three questions

What are the components and goals of the software engineering studies?
- what we are studying and why

What kinds of experiment have been performed?
- the types and characteristics of the experiments run

How is software engineering experimentation maturing?
- judgements against some criteria and examples

The Status of the Experimental Discipline

What are the components of the studies?

We use four parameters (based on the GQM template):

object of study: a process, product, any form of model

purpose: characterize (what happens?)

- evaluate (is it good?)
- predict (can I estimate something in the future?)
- control (can I manipulate events?)
- improve (can I improve events?)

focus: the aspect of the object of study that is of interest

- reliability of the product
- defect detection/prevention capability of the process
- accuracy of the cost model

point of view: the person who benefits from the information

- the researcher in understanding something better

Identified two patterns:

human factor studies

project-based studies

The Maturing of the Experimental Discipline

What are the components of the studies?

Human-factor studies

- object of study: a small cognitive task
- focus: some performance measure
- purpose: evaluation
- point of view: researcher

Done by/with cognitive psychologists comfortable with experimentation

Have remained studies in the small

Project-based studies

- object of study: software process, product, ...
- focus: a variety from product reliability and cost to process effect
- purpose: evaluation, some prediction; characterization/understanding
- point of view: the researcher (often a practitioner view)

Done mostly by software engineers, less adept at experimentation

Have evolved from small, specific items,

- like particular programming language features
- to include entire development processes, like Cleanroom

The Status of the Experimental Discipline

What kinds of studies have been performed?

1. **Are the study results descriptive, correlational, cause-effect?**

Descriptive: there may be patterns in the data but the relationship among the variables has not been examined

Correlational: the variation in the dependent variable(s) is related to the variation of the independent variable (s)

Cause-effect: the treatment variable(s) is the only possible cause of variation in the dependent variable(s)

Human factor: mostly cause-effect

- Sign of maturity of experimentalists; size nature of problem

Project-based: evolved (?) from correlational to descriptive studies

- Reflects early beliefs that problem was simple and some simple combination of metrics could explain cost, quality, etc.
- Don't have an observational knowledge base

The Status of the Experimental Discipline

What kinds of studies have been performed?

2. Is the study performed on novices or experts or both?

novice: students or individuals not experienced in domain

experts: practitioners or people with experience in domain

Human-Factor: investigate difference between novices and experts

Project-based: more studies with experts, especially descriptive studies of organizations and projects

3. Is the study performed in vivo or in vitro?

In vivo: in the field under normal conditions

In vitro: in the laboratory under controlled conditions

Human-Factor: more in vitro

Project-based: more in vivo

4. Is it an experiment or an observational study?

Experiment: at least one treatment or controlled variable

Observational study: no treatment or controlled variables

The Status of the Experimental Discipline

What kinds of studies have been performed?

Experiments can be
- controlled experiments
- quasi-experiments or pre-experimental designs

Controlled experiments, typically:
- small object of study
- in vitro
- a mix of both novices (mostly) and expert treatments

Sometimes, novice subjects used to “debug” the experimental design

Quasi-experiments or Pre-experimental design, typically:
- large projects
- in vivo
- with experts

These experiments tend to involve a qualitative analysis component, including at least some form of interviewing

The Maturing of the Experimental Discipline

What kinds of studies have been performed?

Experiment Classes

		#Projects	
		One	More than one
# of Teams per Project	One	Single Project	Multi-Project Variation
	More than one	Replicated Project	Blocked Subject-Project

The Maturing of the Experimental Discipline

What kinds of studies have been performed?

Observational studies

- qualitative/quantitative study
- pure qualitative study

Qualitative/quantitative analysis: observer has identified, a priori, a set of variables for observation

There are a large number of case studies and some field studies

- in vivo
- descriptive
- experts

Pure qualitative analysis: no variables isolated a priori, open observation

- deductions made using non-mathematical formal logic

  e.g., verbal propositions

Found only one pure qualitative study, a Field Qualitative Study, in vivo, descriptive, experts

The Status of the Experimental Discipline

What kinds of studies have been performed?

Observational Studies

		Variable Scopes	
		A priori defined variables	No a priori defined variables
# of Sites	One	Case Study	Case Qualitative Study
	More than One	Field Study	Field Qualitative Study

The Maturing of the Experimental Discipline

How is experimentation maturing?

Sign of maturity in a field:

level of sophistication of the goals of an experiment

understanding interesting things about the discipline

For software engineering that might mean:

Can we build models that allow use to measure and differentiate processes and products?

Can we measure the effect of a change in a particular process variable on the product variable?

Can we predict the characteristics of a product (values of product variable) based upon the model of the process (values of the process variables), within a particular context?

Can we control for product effects, based upon goals, given a particular set of context variables?

The Maturing of the Experimental Discipline

How is experimentation maturing?

Sign of maturity in a field:
a **pattern of knowledge** built from a **series of experiments**

Does the discipline build on prior (knowledge, models, experiments).

Was the study an isolated event?

Did it lead to other studies that made use of the information obtained from it?
Have studies been replicated under similar or differing conditions?

Does the building of knowledge exist in one research group or environment, or has it spread to others - researchers building on each other's experimental work?

For example, inspections, in general, are well studied experimentally

However, there has been very little combining of results, replication, analysis of the differentiating variables

The Maturing of the Experimental Discipline

How is experimentation maturing?

There is some evidence that researchers appear to be

- asking more **sophisticated questions**
- **studying relationships** between processes/product characteristics
    - doing more studies **in the field** than in the laboratory
    - **combining** various **experimental classes** to build knowledge

Experimentation can provide us with

- better **scientific and engineering basis** for the software engineering
- **better models** of
    - software processes and products
    - various environmental factors, e.g. the people, the organization
- better **understanding of the interactions** of these models

Software Models and Measures

Basic Concepts

Modeling and **Measurement** are fundamental concepts
- needed to encapsulate the objects of the discipline
- represent our observations
- describe phenomena
- allow prediction

Experimentation provides
- the needed discovery, and evaluation mechanisms
- to evolve the models

Software Models and Measures

Scope: What can we model/measure?

Perspective: From whose viewpoint are we modeling/measuring?
Why are we modeling/measuring?

Framework: How are the appropriate models/metrics determined?
How are models/metrics integrated, interpreted and used?
How is the measurement process organized?

Refinement: Are models and metrics generally applicable?
What kinds of tailoring are performed?

Automation: What measurements have been automated?

Application: What is the state of measurement in practice?

Software Models and Measures

Scope

We need to define models
- To help us understand what we are doing
- Provide a basis for defining goals
- Provide a basis for measurement

We need models of
- The people, e.g., customer, manager, developer
- The processes, e.g., a life cycle, method, technique
- The products, the system, a component, a test plan

We need to study the interactions of these models
- What is the effect of a process change on a product?

We need to associate metrics with these models
- How do we measure process?

Software Models and Measures

Scope

What can we measure?

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
- operational, e.g., reliability

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

Software Models and Measures

Scope

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

Changes and Defects
- e.g., defect prediction models, types of defects expected for the application

Product Progress
- e.g., actual vs. expected product size, library access, over time

Processes
- e.g., process models for Cleanroom, Ada waterfall model

Method and Technique Evaluations
- e.g., best method for finding interface faults

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

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

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

Software Models and Measures

Perspectives

From whose viewpoint are we measuring?

There are a variety of viewpoints and they determine what we measure, e.g.,

Management, Customer, User, Organization, Developer

Why are we measuring?

There are a large number of reasons for measuring and they help determine what we measure, e.g., Characterization and Understanding

Assessment and Evaluation

Prediction and Control

Motivation and Prescription

Improvement

Software Models and Measures

Perspectives

Characterize

Describe and differentiate software processes and products

Build descriptive models and baselines

Understand

Explain associations/dependencies between processes and products

Discover causal relationships

Analyze models

Evaluate

Assess the achievement of quality goals

Assess the impact of technology on products

Compare models

Predict

Estimate expected product quality and process resource consumption

Build predictive models

Motivate

Describe what we need to do to control and manage software

Build prescriptive models

Software Models and Measures

Perspectives

What can we do with measurement?

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

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

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

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

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

Software Models and Measures

Perspectives

In order to:

Plan the software development and maintenance process so that
- adequate resources can be available when needed
- cost/benefit analysis and risk assessment can be performed

Monitor the process to prevent or alleviate difficulties when still possible

Control the process by taking corrective or preventive actions based on quantitative analysis

Evaluate the efficiency of the phases and activities of the development or maintenance process based on objective information

Refine the development and maintenance processes

Software Models and Measures

Frameworks

How are the appropriate metrics determined?

Measurement is not just the collection of data/metrics

Measurement must

be done for a purpose,
have carefully defined and specified objectives
be driven by higher level concepts

so that the right metrics/data are collected and the right interpretations given

There are measurement frameworks to support metric definition and interpretation

Example Frameworks:

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

Software Models and Measures

Frameworks

How is the measurement process embedded in the organization?

We have learned that we have to have an organizational framework that allows the integration of measurement over many projects.

Examples:

Quality Improvement Paradigm (QIP)

Plan-Do-Check-Act (PDCA)

Lean Enterprise Management (LEM)

Total Quality Management (TQM)

Capability Maturity Model (CMM)

Software Models and Measures

Refinements

Are models and metrics generally applicable?

We have learned that different environments display different characteristics and that models need to be tailored to the specific environment.

What kinds of tailoring are performed?

Example: Resource Models

Several models offer different baselines for different functions and allow a variety of parameters to help differentiate the environments (COCOMO, SLIM,..).

The meta-model approach recommends building separate models for different environments, even within the same organization.

Software Models and Measures

Refinements

Is the model correct in principle?

Does the model actually describe what we are doing?

How can we improve the model based on theory, practice and analysis?

How do we feed back what we have learned to improve the model or our adherence to it?

Can we associate measurement with the model?

We want to build descriptive models to explain what is happening.

We want to define prescriptive models to motivate improvement.

Software Models and Measures

Automation

What aspects of measurement have been automated?

There are a variety of commercially available tools that generate various code metrics for a variety of languages.

Examples: Code Metric Tools such as Analysis of Complexity Tool (ACT) and BattleMap (McCabe&Associates) and Logiscope (Verilog)

There are several measurement environments that go beyond code metrics and support various management functions. These use historical data from the existing or other environments.

Examples: SPQR AND Checkpoint (Software Productivity Research, Inc.) and

PADS (Quantitative Software Management, Inc.)

Software Models and Measures

Application

What is the state of measurement in practice?

Many companies have full-scale measurement programs, not just at the project level but at the division or corporate level.

Typically, the most advanced are using some form of framework.

Examples:

HP employing an early version of the GQM

Motorola employing GQM and Quality Improvement Paradigm

NEC employing SQMAT, an improvement on SQM, and Plan-Do-Check-Act

AT&T employing QFD, adapted to software

Software Models and Measures

Assessing the Levels of Measurement within companies

(using the SEI Process Assessment Notation)

	Scope of Measurement
Optimized	Measured Continuous Feedback for Process Improvement
Managed	Process Measurement
Defined	Product Measurement
Repeatable	Project Measurement
Initial	Little or No Measurement

Software Models and Measures

Views of Software Metrics

There are various ways of discussing metrics:

- accuracy level, e.g., objective, subjective
- measurement scales, e.g., nominal, ordinal, interval, ratio
- object of measurement, e.g., process or product

Objective Metric:

- An absolute measure taken on the product or process
- Usually done on an interval or ratio scale
- Examples: time for development, number of lines of code, number of errors or changes

Subjective Metric:

- An estimate of extent or degree in the application of some technique
- A classification or qualification of problem or experience
- Usually done on a nominal or ordinal scale
- Examples: quality of use of a method or technique, experience of the programmers in the application or process

Software Models and Measures

Views of Software Metrics

Measurement Scales

Scale	Basic Operations	Typical Examples
Nominal	Determination of Equality	Application areas Types of defects
Ordinal	Determination of Greater or Less	Level of training or understanding
Interval	Determination of Equality of intervals or differences	Calendar dates
Ratio	Determination of the equality of ratios	Lines of Code Number of defects Code Complexity

Software Models and Measures

Definitions

We can develop models and measures of various software phenomenon

- Processes being used
- Products of all forms
- Project Characteristics
- People
- Resources being expended
- Changes and defects associated with projects

We can associate data with these models

And we can combine these models to create new models and measures

In what follows, we will offer example models and metrics

- To demonstrate the various aspects the discipline
- To provide a basis for you to develop other models and metrics

References

- Basili, Models and Metrics for Software Management and Engineering, IEEE Computer Society Press, 1980
- Conte, Dunsmore, & Shen, Software Engineering Metrics and Models, Benjamin/Cummings, 1986.
- Victor R. Basili, The Role of Experimentation in Software Engineering: Past, Current, and Future, Keynote Address, 18th International Conference on Software Engineering (ICSE 18), Berlin, Germany, March 25-29, 1996.
- Fenton & Pfleeger, Software Metrics: A Rigorous and Practical Approach, PWS Publishing Company, 1997
- Wohlin, Claes, et.al., Experimentation in Software Engineering: An Introduction, Kluwer Academic Publishers, 2000.
- Juristo, Natalia and Moreno, Ana, Lecture Notes on Empirical Software Engineering, World Scientific, Singapore, 2003.