Chapter 1 - Introduction

(*) Some of these slides are based on Prof. Basili's & Rombach's slides from CMSC 735 @ UMD, USA
1. Introduction - Outline

• 1.1. Motivation
• 1.2. Evolving Knowledge, Model Building, Experimentation & Learning
• 1.3. Engineering Disciplines
• 1.4. Software Engineering
• 1.5. Empirical Software Engineering
• 1.6. Available Research Paradigms
• 1.7. Status of Model Building
• 1.8. Status of Experimentation
1.1. Motivation

Project Organisation 1

- Project Planning
- Project Plan
- SW-System/Product
- Reuse (Models)
- Storage (products, measures)
- Experience Factory
- Experience database
- Project database
- Process models
- Product models
- Quality models
- T/M/W
- Products
- Project plans
- ...
1.1. Motivation

• V-Model

Problem Stmt ($\leftarrow$ SoP) with Improvement Hyp.

Solution Stmt ($\leftarrow$ SoR) with Improvement Hyp.

Research

Technical Solution

Emp. Testing of Problem hypothesis

Emp. Testing of solution hypothesis
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.
1.2. Evolving Knowledge, Model Building, Experimenting & 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 objects, the properties of the systems that contain them, the relationship of the object to the system, and the culture of the discipline.

This affects

how the models are built
how the experimentation gets done
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.
1.2. Evolving Knowledge, Model Building, Exp & 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
1.2. Evolving Knowledge, Model Building, Exp & 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
1.3 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
1.3 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
1.3 Engineering Discipline Requirements

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*
Packaging Experience (Building Models)

Experience needs to be packaged
  *we must build models in software engineering*
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*
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*
1.4 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
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 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**
1.4 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
1.5 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

1.5 Misconceptions About Empirical Software Engineering

- 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
1.5 Misconceptions About Empirical Software Engineering

• **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 know…**”
  – Technology X doesn’t work. (NO!)
  – Technology X performed worse than technology Y in our environment. (YES)
    → “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.**
• **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.
1.5 Empirical Software Engineering - Early Observations

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?
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?
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 about type of software?)

What is the effect on other variables? What size minimizes the defect rate?
1.6 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
1.6 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
- observational
  - driven by understanding; qualitative analysis dominates
  - qualitative/quantitative study
  - pure qualitative study
1.7 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.?
1.8 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
1.8 The Status of the Experimental Discipline

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
1.8 The Maturing of the Experimental Discipline

**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
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
1.8 The Status of the Experimental Discipline

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
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
### Experiment Classes

<table>
<thead>
<tr>
<th># of Teams per Project</th>
<th># Projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>One</td>
<td>Single Project</td>
</tr>
<tr>
<td>More than One</td>
<td>Replicated Project</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>One</th>
<th>Multi-Project Variation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Blocked Subject-Project</td>
</tr>
</tbody>
</table>
1.8 The Maturing of the Experimental Discipline

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
### Observational Studies

#### Variable Scopes

<table>
<thead>
<tr>
<th># of Sites</th>
<th>A priori defined variables</th>
<th>No a priori defined variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>One</td>
<td>Case Study</td>
<td>Case Qualitative Study</td>
</tr>
<tr>
<td>More than One</td>
<td>Field Study</td>
<td>Field Qualitative Study</td>
</tr>
</tbody>
</table>

What kinds of studies have been performed?

1. **Case Study**
2. **Field Study**
3. **Case Qualitative Study**
4. **Field Qualitative Study**
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 us 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?
1.8 The Maturing of the Experimental Discipline

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.
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