A Role for Empirical Methods in Answering the Question: What is Software?

Prof.-Dr. Leon J. Osterweil
University of Massachusetts
Amherst, MA  USA

In Honor of the 60\textsuperscript{th} Birthday of
Prof.-Dr. Dieter Rombach
Two of Dieter’s Four Questions

• How should software engineering be positioned as an academic discipline (part of computer science, or separate like other engineering disciplines)?
• What are current and future “big challenges” in our field?
Two of Dieter’s Four Questions

• How should software engineering be positioned as an academic discipline (part of computer science, or separate like other engineering disciplines)?
• What are current and future “big challenges” in our field?

Maybe best answered in the context of a deeper question
What IS Software?

• We have been engineering it for >50 years
  – Aren’t we curious about its inherent nature?
• Struggling with hard vexatious questions can be
  – Exhilarating
  – Illuminating
  – Satisfying
• For example: “What is love?” has stimulated
  – Poetry
  – Music
  – Art
  – Philosophy
What IS Software?

• We have been engineering it for >50 years
  – Aren’t we curious about its inherent nature?
• Struggling with hard vexatious questions can be
  – Exhilarating
  – Illuminating
  – Satisfying
• For example: “What is love?” has stimulated
  – Poetry
  – Music
  – Art
  – Philosophy

There are more pragmatic reasons too
We need to set future directions for our community: Could this help?

• Our community has focused on the engineering (and science) of Computer Software

• What comprises this thing that we engineer?
  – Only code?
  – Code + Design?
  – Requirements, Design, Code, Test cases/results?

• The Software Engineering community has taken a broad view
  – As evidenced by ICSE, TSE, TOSEM, FSE papers
How Broad?: Should software engineering encompass all issues relating to all of these kinds of artifacts?

- Psychology to make requirements elicitation more effective?
- Economics to improve profitability of software development?
- Management to improve human development in creating software?
- Physics to understand modern storage devices?
- Mathematics to reason about complex type systems?
And what about Other Kinds of “Software”

- “Software Processes are Software Too”
- Actually all kinds of processes are software too
- Entertainment programs are software (?)
- Laws are software (just ask Katayama)
- And other things too
And what about Other Kinds of “Software”

- “Software Processes are Software Too”
- Actually all kinds of processes are software too
- Entertainment programs are software (?)
- Laws are software (just ask Katayama)
- And other things too

All areas of all disciplines that contribute to all of these other kinds of software?
If we knew the nature of “software” it should help us to distinguish what we should study from what we should feel safe in excluding.
The state of software engineering research today

- Range of papers appearing in top venues seems to be narrowing
- Papers on older areas seem fewer in number
- Other areas of CS (Databases, PL, BPM) continue to chew at our margins
- Papers that carry learning from other areas back to software engineering have a hard time
Some **qualitative** aspects of “software” (borrowed from an earlier paper)

- It is non-physical, non-tangible
  - But manages physical, tangible things
- Hierarchical structure is common
- Typically built in phases
  - Consists of components with different purposes
- Expected to require modification, evolution
- Interconnections are key
- Analysis and verification are essential
But can we be **Qualitative**?

- What about measuring software?
- A role for Empirical Methods?
- Continuing Dieter Rombach’s work and goals
“...when you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory kind; it may be the beginning of knowledge, but you have scarcely in your thoughts advanced to the state of Science, whatever the matter may be. “

Lord Kelvin
How Do We Measure Software Now?

- Counting lines of code
  - And how do we do that?
- Measuring the cost to create software
  - Only the code, or all of the artifacts?
- Measuring the time needed to create software
- Volumes (or weight) of documentation
- Number of installations, users, ??
How Do We Measure Software Now?

• Counting lines of code
  – And how do we do that?
• Measuring the cost to create software
  – Only the code, or all of the artifacts?
• Measuring the time needed to create software
• Volumes (or weight) of documentation
• Number of installations, users, ??

Surely we can do better, can’t we?
A Modest Proposal
(To start conversation)

• Measure software by its capacity to do work
  – Sort of the way voltage was used for electricity

• What is the “size” of a line of code?
  – The extent to which it is able to change the state of a computation

• Does an iterator multiply the size of iterated code?

• Size of a basic block, procedure, module, program is some function of its lower level constructs
This is probably not the right idea

• But could empirical methods be used to evaluate this, and some variants?
• Struggling with it could help distinguish the important from the trivial
• And help us understand the nature of software
• And help guide the continued development of a more coherent, vibrant, thriving discipline
Happy Birthday, Dieter!

We look forward to many, many more years of your community leadership