A Future for Software Engineering?

Leon J. Osterweil
University of Massachusetts
Amherst, MA, USA

23 May 2007
How to Project the Future?

• Extrapolate (?)
  – Current Velocity
    • Where are we coming from?
  – Current Position
    • Where are we now?
  – Future Position
    • Where will we be?
  – Unless forces change…..

• But, will the forces change?
  – Forces that be imposed upon us
  – Forces that we might impose upon ourselves
The NATO Conferences

• The dawn of history for Software Engineering
• Enunciation of a set of problems and issues
  – 1968 in Garmisch
  – 1969 in Rome
• Set the initial trajectory for our discipline
Some problems posed there

• Can system software be written in a high-level language?
• What is nature of design? How to fit it into development?
• How should software be tested?
• What to do about problems of scale?
• What is Software Quality? And how to achieve it?
• What processes can deliver high quality products
• How can/should we do component-based development?
Some problems posed there

- Can system software be written in a high-level language?
- What is nature of design? How to fit it into development?
- How should software be tested?
- What to do about problems of scale?
- What is Software Quality? And how to achieve it?
- What processes can deliver high quality products?
- How can/should we do component-based development?

Most are still before us
Some problems posed there

- Can system software be written in a high-level language?
- What is nature of design? How to fit it into development?
- How should software be tested?
- What to do about problems of scale?
- What is Software Quality? And how to achieve it?
- What processes can deliver high quality products
- How can/should we do component-based development?

Most are still before us. Is that bad?
How have we been functioning?

Real World

Problems

Solutions

Research
Maybe more like--

- Real World
- Problems
- Research
- Ideas, Theories, Prototypes
- Feedback
- Evaluation
- Solutions
- Data, Experience
Actually maybe even more like…..

- From Impact Project report on SCM
- J.Estublier, A. van der Hoek, et. al.
- ACM TOSEM Oct. 2005

- Products
- Research
- from A. van der Hoek
Present Position of Software Engineering

• Strong and Positive
  – Strong impact on the world
  – By a broad and powerful community

• Should be a source of pride
  – More than we seem to feel
    • Especially in the research community
  – More on this problem soon....
Some Impressive Numbers

- Millions of jobs worldwide
  - Many are very high paying
- Annual revenue in hundreds of billions USD
  - Trillions?
- Conferences
  - ICSE, FSE/ESEC, and dozens more
- Journals
  - TSE, TOSEM, and dozens more
- Magazines
  - IEEE Software, and dozens more
Non Quantitative Measures

• A core competency in diverse disciplines
  – Telecommunications
  – Aerospace
  – Hardware
  – Automotive
  – Medical
  – ……

• Subject of major national initiatives
  – India, China, Germany, Ireland, ….
  – Counted on for wealth generation
Impact from Synergy between Research and Practice

• The Impact Project
  – Assessing the interplay between Software Engineering Research and Practice

• Numerous, well-documented examples
  – In TOSEM papers
  – And SEN preprints

• Come to the three sessions of the ICSE 2007 Impact Project Track
Extrapolating Forward

• Continue to be driven by problems arising in the real world
  – This has been a great success
• New forces from new problems that are growing in size, scope, and complexity
  – New disciplines, new problems, etc.
• A promising and tractable Future
  – But, A future that is driven by others
  – How do we feel about a future that is so reactive?
Do we have an agenda of our own?

And should we?
Other scientific communities derive strength from agendas driven by deep questions

- **Physics**
  - What is the nature of matter, energy, time?
  - What is the world made of?

- **Biology**
  - What is the nature of life?
  - Where did we come from?

- **Astronomy**
  - How did the universe begin/how will it end?
Other scientific communities derive strength from agendas driven by deep questions

• Physics
  – What is the nature of matter, energy, time?
  – What is the world made of?
• Biology
  – What is the nature of life?
  – Where did we come from?
• Astronomy
  – How did the universe begin/how will it end?

Note: None of these has much practical value
Deep questions followed long observation of practice

- Phenomena are observed
- Primitive practice is established
  - And nagging questions arise
- Leads to curiosity about deep questions
- Leads to deeper understandings
- Leads to more mature and effective practice
Deep questions followed long observation of practice

- Phenomena are observed
- Primitive practice is established
  - And nagging questions arise
- Leads to curiosity about deep questions
- Leads to deeper understandings
- Leads to more mature and effective practice

In disciplines such as: Thermodynamics, Medicine, Electrical Engineering
Do we have analogous questions?

Can our curiosity about them help us in similar ways?
Shaping our agenda

• Our conferences focus on answers
• Should they also address deep questions?
• Should they encourage more curiosity about our problem domains?
• Do we need a different format for this?
• Do we need this at all?
Shaping our agenda

- Our conferences focus on answers
- Should they also address deep questions?
- Should they encourage more curiosity about our problem domains?
- Do we need a different format for this?
- Do we need this at all?

Maybe it is time for us to talk about all of this?
Grappling with Hard Problems Should Pique Curiosity

• It need not be embarrassing to be working on the same question for years, decades, even centuries
  – Physicists and biologists are proud to do this
• Having such problems can be an encouraging sign of the profundity of a discipline
• Software Engineering has such questions
  – Could we be proud and relieved?
  – Should we emphasize a focus on enunciating them and addressing them?
Curiosity-Driven Research

Real World → Problems → Research → Evaluation → Feedback → Data, Experience → Solutions → Ideas, Theories, Prototypes → Curiosity-Driven Research
Well-articulated deep and enduring questions can help us

- Define what we are all about
- Set overall directions
- Maintain and build a sense of progress
- Explain and justify ourselves to others
  - Faculty colleagues
  - Industrial colleagues
  - Government funding sources
Deep, Curiosity-Driven Questions

• We have them
  – They have been there from the start
• Should we look for more of them?
  – Maybe we just have to acknowledge them?
• Should we talk about them?
  – In our best venues?
• Can they constitute a force that affects the future trajectory of our community?
• Should the community support this activity by
  – Seeking them?
  – Evaluating them?
Alan J. Perlis, in 1968

“...we recognize that a practical problem of considerable difficulty and importance has arisen: The successful design, production and maintenance of useful software systems....The source of difficulty is...easy to identify, and yet its cure is hard to pinpoint.... Our problem has arisen from a change of scale....we must assume that additional magnification of goal will take place without necessarily being preceded by the emergence of a satisfactory theory or....tools. Not only must we know how to build special purpose systems, but how to combine them into larger ones.”
Other examples of Deep Questions?
What is Design?

• The noun
• The verb
• The US NSF Science of Design program is currently probing this.
What is the Relationship Between a Model and Reality?

- A representation of reality?
- What does that mean?
- Pondering that goes back (at least) to Plato
  - Allegory of the Cave
How to determine software quality?

• Or do we mean qualities?
• Which are they?
• Can we quantify them? How?
• Would help to know the nature of software
What is Execution?
What is software?

• Many negative characterizations
  – Not tangible, physical, sensible, etc.
• But few positive characterizations
• What are its properties, characteristics?
• Maybe understanding design, quality, modeling, execution will help?
• How can we engineer it without knowing its nature?
What is our Future?

• Should we be proactive in determining it?
• Should it be a balance between
  – Problems from real-world clients
  – Deep questions of our own formulation
• How can we nurture and foster discussion about this?
Should we be proactive in shaping A Future for Software Engineering?