The Impact Project:
Tracing the Source of Software Engineering Technology to its Origins

Leon J. Osterweil

Professor, Dept. of Computer Science

Dean, College of Natural Sciences and Maths

University of Massachusetts

Amherst, MA 01002

USA

Institution of Electrical Engineers, Savoy Place, London

26 October 2001

Where Does Software Technology Come From?

- Who should get the credit?
  - What credit should they get?
- How to award credit?
  - What measures?
  - How to determine them?
- Does this really matter?
  - To whom?
  - For what reasons?

Credit is due to:

- Commercializers
- Researchers
- Tech Transfer agents
- Early adopters
- Scientific and Technical Communities
- Students with new degrees
- New Hires
- ETC.

What are the natures of their contributions?
How to value them?

Contributions Differ

- Initial conceptualization of idea
- Evangelism
- Prototype demonstration
- Public promulgation
- Nurturing by community activities
- Indoctrination and training (students)
- Product commercialization

How to Evaluate These?

- Qualitatively
- Quantitatively
- Different perceptions by different parties
- Difficulties in assessing contributions

Facile answers are misleading

- It comes from:
  - Sun, Microsoft, IBM, Rational, the web,
- Yes, but!
  - Where did they get it from? And how?
- It comes from Dr. X's research
  - Published a seminal paper
- Yes, but!
  - Someone else cleaned it up, crafted code
- It was "in the air"
  - How did it get there? Who nurtured it?

Why should we care? (as users, beneficiaries)

- Some technology isn't great
  - Why are we stuck with it?
  - Why isn't it better?
- Some technology seems useful
  - How can we get more of same?
  - How can we speed its appearance?
  - Are there institutions that need to be
    - Strengthened
    - Demolished

Why Should We Care? (As Researchers)

- Altruistic reasons
  - More effective tech transfer
  - Better technologies in use
- More self-serving reasons
  - Self-image
  - Academic status
  - Positive Attitude
  - Funding prospects

Software Engineering Myths

- Software engineering research has had minimal impact on practice
- Software engineering research results have effected minimal increases in productivity
- Software engineering research has been a poor investment, giving little return
- Industry leads, research follows
  - (Industry cleans up; research sweeps up)

(Some myths are false, some are not)

Software Engineering Facts

(Some “facts” are false, some are not)

- Theory:
  - Software engineering research problems are hard, fundamental, and enduring
  - The research community has an increasingly strong grip on these problems
- Practice:
  - Software practice has achieved orders of magnitude productivity increases
  - Research results have driven much of this

More “Facts”

- We don’t know what the truth really is
- We are not really sure how to decide what “the truth” really is
- We really need to find out…….

The Impact Project: Tracing the Source(s) of Technology to its Origins

- Focus is on Software Technology
- Start with technologies in widespread use
- Trace back to how they came into widespread use
- Document and analyze
  - What facilitates/inhibits technology flow?
  - How to make more good things happen more easily and more often

Impact Project Structure

- Software community volunteer effort
- Modest sponsorship by:
  – ACM Sigsoft, US NSF, IEE, Japan
- Collection of reports on key areas
- Appearing 2002 onwards

Selected Report Topics

- Configuration Management
- Reviews and Walkthroughs
- Design Methods and Tools
- Programming Environments
- Modern Programming Languages
- Cost and Economic Modeling
- Testing and Evaluation
- ….. And more…..

Project Products

- Set of reports
  - Organized around subject areas
  - Range of sizes
    - Full (25-30 pages?): journal quality
    - Condensed (3-5 pages?): magazine style
    - Popular press (?): Scientific American?
    - Abstracts (one pager, one paragraph)
- Briefing materials
  - For all occasions

Project Organization

- Steering Group:
  - L. Osterweil, J. Kramer, C. Ghezzi, A. Wolf
- Subject Area-Based Author Groups
  - 12-20 Subject areas
  - 8-10 Authors per subject area
  - 1 or 2 CoLead Authors per subject area
  - Inclusive, open to broad community participation
- Panel of Distinguished Reviewers

Status Report on Software Configuration Management Study

Alexander L. Wolf

Department of Computer Science

University of Colorado at Boulder

Team

- Lead Authors
  - Jacky Estublier, U. Grenoble/IMAG (Adele)
  - David Leblang, retired (Clear Case, DSEE)
- Contributing Members
  - G. Clemm, Rational (Clear Case, Odin)
  - R. Conradi, U. Trondheim (EPOS)
  - A. van der Hoek, UCI (NUCM)
  - W. Tichy, U. Karlsruhe (RCS)
  - D. Wiborg-Weber, Continuus (CCM)

What is SCM?

- Managing a repository of components
  - Version Control; Product Models; Composition and Selection
- Helping engineers in their usual activities
  - Building (derived object control); Work Space Control
- Controlling and supporting the process
  - Change Control; Cooperative Work; Process Support

SCM Is in Wide Use

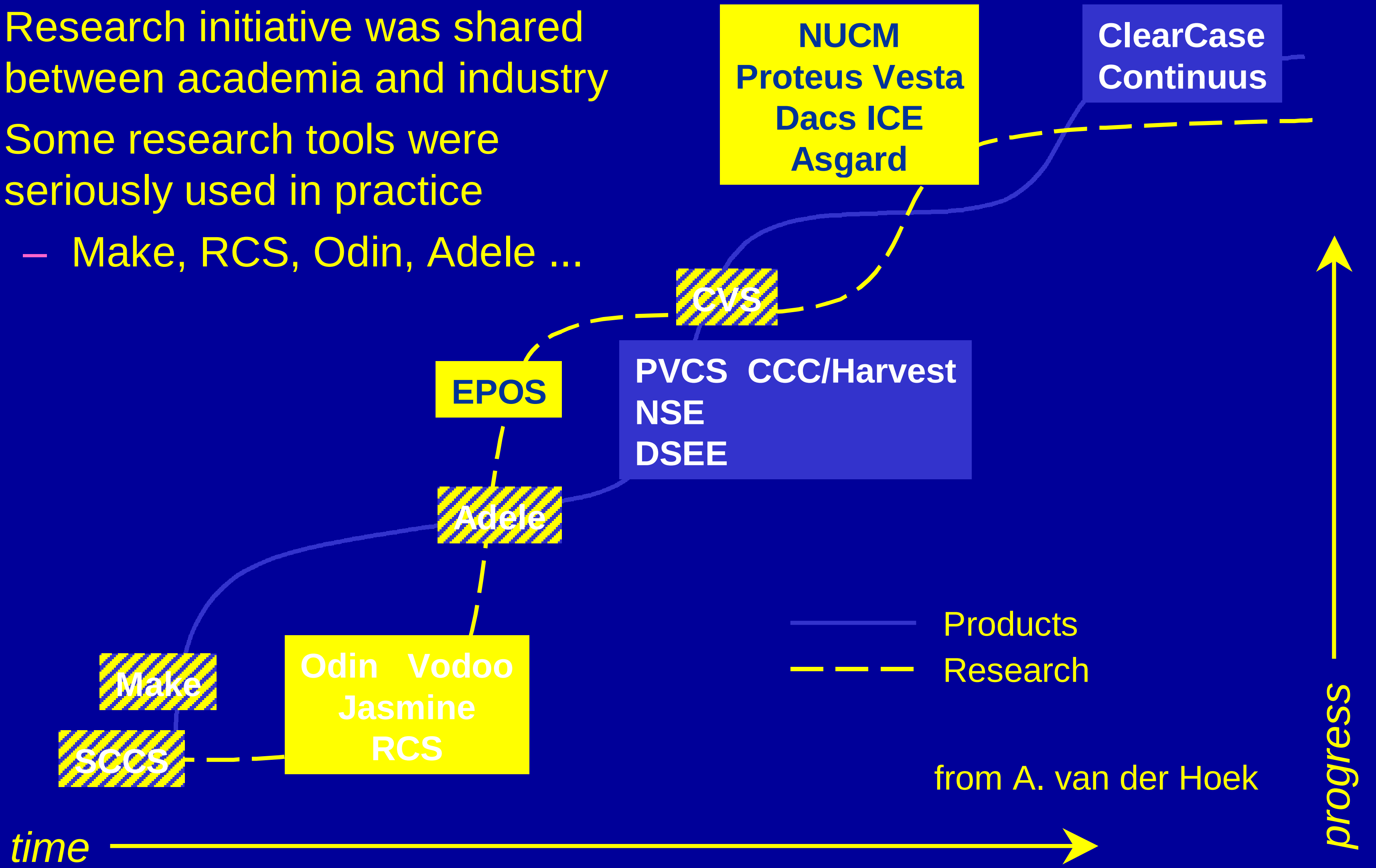
- Ovum estimates:
  - $1B (1998), $2B (2000), $3.3B (2002)
  - 25% mainframe;15%-20% workstations; 5%-10% PC
- Gartner estimates:
  - $6B (2003)

SCM Impact Study Plan

- Examine characteristics/features of leading products in SCM market
- Assume that products used in practice
- Trace characteristics/features back to research ideas and prototypes
- Try to establish arguments for/against influence of research on practice (via products)

Complex Interplay between Research and Commercialization

- Research initiative was shared between academia and industry
- Some research tools were seriously used in practice
  - Make, RCS, Odin, Adele ...



Other Factors As Well

- Role of Community Nurturing
  - Product architects present at nearly all SCM workshops (1988-2001)
  - Cagan, Clemm, Dart, Leblang, Wiborg-Weber
- Movement of key people precipitated progress

Impact of Software Engineering Research on Modern Programming Languages

A work in progress

Barbara G. Ryder and Mary Lou Soffa

Approach

- Focus on languages currently in widespread use: e.g., Java, C++, Ada, Perl
- Find origins of, and influences on, essential features of these languages

Challenges

- PL and SE have a complex, close, synergistic relationship
- Hyp 1: SE research impacted PL design
- Hyp 2: PL impacted SE research and practice
- SE(PL) research also influenced SE(PL)
  - Parnas on modularity -> OO design
  - Simula 67 -> C++,Java

Modularity and Information Hiding

- Roots of OO languages – (modules, objects, inheritance, dynamic method binding) – Simula 67
- Parnas (1972) was one of the first to recognize value of modularity (code + data)
- Parnas first to come up with notion of information hiding (1972)
- Refined notion of encapsulation appeared in CLU in 1977

Exceptions

- Introduction of exceptions - ON conditions in PL/I in early 1970's
- J. Gannon, J.J. Horning (CACM 1975) explore issues in PL design for reliability
- J.Goodenough (CACM 1975) defines exception conditions and proposes PL features for handling
- Mesa, Mitchell et al (1979) and Ada, J. Ichbiah et al. (Ada Rationale 1979)
- CLU, B. Liskov, A. Snyder (IEEE-TSE 1979) offers clean definition of handling

Some Broader Lessons

- Vendors tend to see value (impact) in
  - algorithms (e.g., differencing)
  - pieces of reusable code (e.g., RCS)
- But not in
  - concepts (e.g., hierarchical workspaces)
  - architectures (peer-to-peer repositories)
  - Which are often seen as "engineering common sense"
  - "Research had very little influence …"
  - "We do not sell ideas, but tools. We (re)invented everything we needed…"

More Lessons

- Researchers tend to see impact in
  – Precedence
  – Concepts
  – Prototypes
- But tend to devalue importance of
  – Efficiency
  – Usability
  – Reliability
  – seeing them as "engineering common sense"
  – "We invented almost everything …"
  – "Tools are only an engineering issue …"

Still More Lessons

- **Both are right, both are wrong**
- **A good idea is had more than once**
- **Vendors have disincentives for distributing credit for ideas**
- **Researchers have incentives for claiming credit for ideas**
- **Research and productization both require *engineered creativity***

Final Observations

- Cultural chasms between research and commercialization
  - Probably deeper than either realizes
- Each needs the other more than it realizes
  - Causes more damage than realized
- Archeology is hard
  - But very timely now
  - Can get contentious
- History is hard too
  - Especially for non-historians

"Those who refuse to study history are doomed to relive it"

"Those who refuse to study history are doomed to relive it"

"History teaches us that History teaches us nothing"