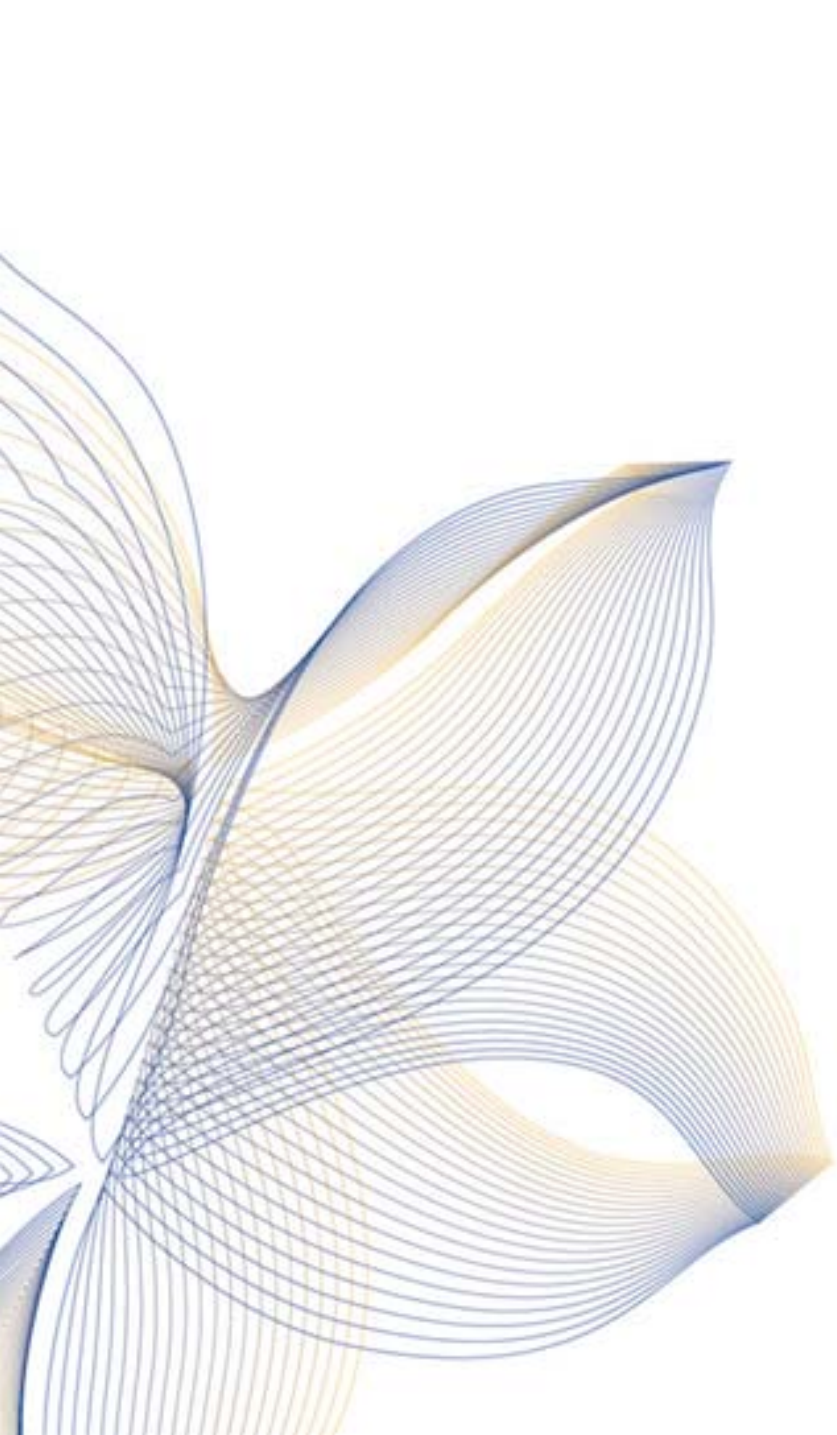

Industrializing Software Development

Alexander Stepanov

12/09/2004

Adobe

Abstract

The objective of the talk is to discuss economic, organizational, and technological aspects of software industrialization. While it is impossible to predict exactly when the industrial revolution in software will occur, it is clear that when it happens it will cause a dramatic redistribution of wealth and a decline of the software monopolies.

There is the economic reason why software components as an industry (predicted in the late sixties by Doug McIlroy) never materialized: it is the emergence of the software industry, whose very existence is based on unspecified, irregular and extremely complex interfaces.

Organizationally, there is no division of labor, a very low level of professionalism, and a reward system that is based on number of features, rather than on the level of reliability, correctness, and security.

Finally, technologically we still have to learn to produce comprehensive, well-organized catalogs of highly generic, reliable components with precise time and space performance characteristics.

Adobe

Doug McIlroy's paper

- **Software Engineering, Report on a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7th to 11th October 1968**

- **http://cm.bell-labs.com/cm/cs/who/doug/components.txt**

McIlroy's Problem - 1968

"We undoubtedly produce software by backward techniques. We undoubtedly get the short end of the stick in confrontations with hardware people because they are the industrialists and we are the crofters. Software production today appears in the scale of industrialization somewhere below the more backward construction industries. I think its proper place is considerably higher, and would like to investigate the prospects for mass-production techniques in software."

Adobe

McIlroy's solution: Component Industry

"The most important characteristic of a software components industry is that it will offer families of routines for any given job. No user of a particular member of a family should pay a penalty, in unwanted generality, for the fact that he is employing a standard model routine. In other words, the purchaser of a component from a family will choose one tailored to his exact needs. He will consult a catalogue offering routines in varying degrees of precision, robustness, time-space performance, and generality. He will be confident that each routine in the family is of high quality - reliable and efficient. … He will expect families of routines to be constructed on rational principles so that families fit together as building blocks. In short, he should be able safely to regard components as black boxes. "

Adobe

Choices of components

- **Precision**
- **Robustness**
  - Checking for “bad” inputs
- **Generality**
  - Compile vs. Run time
- **Time-space behavior**
- **Algorithm**
- **Interfaces**
- **Accessing method**
- **Data structures**

Choice of Data structures

- Algorithms are as insensitive to changes of data structure as possible

- When radically different structures are useful for similar problems (e.g., incidence matrix and list representations for graphs), several algorithms may be required

The world in 2004

- **The industrialization did not happen**
- **There are experts who claim that it cannot happen:**
  - http://research.microsoft.com/Lampson/Slides/ReusableComponentsAbstract.htm

Adobe

Cui bono?

Who benefited?

Adobe

Market capitalization

- Microsoft - $300B
- Google - $50B
- Yahoo - $50B

- AT&T - $12B
- Ford - $23B
- GM - $21B
- Sony - $32B

Software drains capital from tangible goods

Adobe

It is not a conspiracy

- **Market forces can lead toward creation of monopolies**
  - "The invisible hand" leads to a local optimum
  - Monopolies cause stagnation

- **Societal action is needed to avoid stagnation**

Adobe

Paradoxes of Software Economics

- The industry with the smallest productivity growth has the greatest capital accumulation
- While the quality of tangible goods has increased dramatically over the last 30 years the quality of software artifacts has been steadily declining

Adobe

End-User License “Agreements”

DISCLAIMER OF WARRANTIES: YOU AGREE THAT THE COMPANY HAS MADE NO EXPRESS WARRANTIES TO YOU REGARDING THE SOFTWARE AND THAT THE SOFTWARE IS BEING PROVIDED TO YOU "AS IS" WITHOUT WARRANTY OF ANY KIND. THE COMPANY DISCLAIMS ALL WARRANTIES WITH REGARD TO THE SOFTWARE, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, MERCHANTABLE QUALITY, OR NONINFRINGEMENT OF THIRD-PARTY RIGHTS.

Adobe

Evolutionary pressures

- **Low productivity requires huge numbers of programmers for relatively simple tasks**
  - Barriers to entry
- **Manual development generates defects**
  - Users need new, “better,” releases
- **Poorly specified interfaces assure non-portability**
  - Platform lock-in

Complexity creates “opportunity”

- **Call centers**
- **Integrators & Services**
- **CIOs and IT organizations**

Adobe

Glue vs. substance in Photoshop

- **Glue - 90% of the present day codebase**
  - Memory management
  - Scripting
  - UI management
  - File I/O
  - Color Management
- **Substance - 10% of the present day codebase**
  - Specialized Image Processing
  - UI Design

Scott Byer - Photoshop Architect

Percentage of substance in other products

- **Word processing - 3%**
- **Presentation App - 1%**
- **Databases - 10%**
- **Enterprise Application Software - 1%**
- **Technical CAD - 30%**
- **Operating System - 1%**

A consensus estimate

Adobe

Science and engineering

- **Every engineering discipline is based on science**

- **Science feeds off engineering**

Scientific prerequisites

- Science is based on reproducible results
- Open interactions
- Lack of hierarchical control
- Social methods of validation

Solution: Open Source + Science

- Organized catalogues
- Funding
- Education
- Industrial accounting
- International legal framework

Adobe

Solution: Web catalogues

- **Massive catalogs of components**
- **Systematic organization**
- **Careful selection, testing and measurement**
- **Statistic of usage**
  - The only obligation for the user is to report
  - Funding should be proportionate to use
- **Multi-language, multi-platform**

Solution: Funding

- World-wide Software tax - 1% of the price of software
- Open source infrastructure
- Components
- Careful accounting of the return on the investment

Adobe

Solution: Computer Science Education

- **Train different categories**
  - Component designers (few)
  - System designers (many)
- **Strong emphasis on traditional mathematics**
  - Euler - not Bourbaki
- **Faculty has to learn to program**
  - Nemo dat quod non habet
    - *No one gives what he does not have*
  - Producing *usable* and *used* software artifacts should be the main requirement for tenure

Solution: Software Accounting

- **Code is liability**
  - Depreciation
  - Maintenance
- **Organizational tax on code**
  - Lines
  - Changes across releases
  - Bugs

Conclusion

- Business as usual is not tenable
- The problem is economic, not technological
- Industrialization requires science
- Science requires openness
- Open source libraries and systems are the path to the future
- Economic and organizational changes are necessary

Adobe