Our civilization runs on software.

Bjarne Stroustrup
You have riches and freedom here but I feel no sense of faith or direction. You have so many computers, why don’t you use them in the search for love?

Lech Walesa

The Promise

- Software-intensive systems can amplify human intelligence
  - They cannot replace human judgment

- Software-intensive systems can fuse, coordinate, classify, and analyze information
  - They cannot create knowledge
The Limits

- Not everything we want to build can be built
  - There exist pragmatic theoretical and technical limits

- Not everything we want to build should be built
  - There exist moral, economic, social, and political limits
The Beauty

- Software-intensive systems are perhaps the most intellectually complex artifacts created by humans
  - Software is invisible to most of the world
  - The best software is simple, elegant, and full of drama as manifest in the cunning patterns that form its structure and command its behavior
The Contradiction

- The Internet changes the way that individuals communicate and businesses collaborate
  - The Internet provides an environment where terrorists and other criminal enterprises can operate with little fear of detection

Internet Mapping Project @ http://www.cheswick.com/ches/map/index.html
The Contradiction

- The Web provides unprecedented mechanisms for social networking
  - The Web creates a greater opportunity for the exploitation of children as well as fraud and theft directed against individuals and organizations

Britney Spears @ http://www.myspace.com/britneyspears
The Contradiction

- Software-intensive systems permit real time and distributed access to information
  - Software-intensive systems can erode personal privacy and other basic human rights

WeatherLink @ http://www.davisnet.com
The Contradiction

- Email and other software-intensive mechanisms increase the velocity of communication
  - Email and the aging of digital archives threatens the preservation of history

Project Gutenberg @ http://www.gutenberg.org
The Contradiction

- Software-intensive systems create new forms of artistic expression
  - Piracy disrupts the economic underpinnings of traditional media companies and can dilute the intellectual property of artists

Halion @ http://www.steinberg.net
The Contradiction

- **Software-intensive systems enable and accelerate scientific research**
  - Software-intensive systems are at the center of a new generation of offensive and defensive weapons

Entrez Genome Project @ http://www.ncbi.nlm.nih.gov/entrez
The Contradiction

- **Software is a part of the very fabric of civilization, living in its interstitial spaces**
  - The complexity of software-intensive systems continues to grow; this complexity impacts its users as well as the stakeholders to develop, deploy, operate, and evolve them.
Alan Turing: The Promise

I believe that in about 50 years time it will be possible to program computers...to make them play the imitation game so well that an average interrogator will not have more than 70% chance of making the right identification after five minutes of questioning.

Alan Turing

Turing, A., Computing Machinery and Intelligence,  *Mind*, vol. 49, 1950
This special property of digital computers, that they can mimic any discrete-state machine, is described by saying that they are universal machines.

_Alan Turing_
Alan Turing: The Beauty

Mathematics, rightly viewed, possesses not only truth but supreme beauty, a beauty cold and austere like that of sculpture.

Alan Turing
Alan Turing: The Contradiction

- With his work in cryptanalysis, Turing helped defeat Hitler’s oppressive forces
  - Later in his life, Turing himself faced personal oppression by the society he helped preserve
Alan Turing: The Discoveries

- 1935: Quantum mechanics, probability, logic
- 1936-38: Turing machine, computability
- 1938-39: Logic, algebra, number theory
- 1939-42: Bombe (machine for Enigma decryption)
- 1947-48: Computer and software design
- 1948: Programming, neural nets, AI
- 1950: First serious mathematical use of computer
- 1952: Non-linear theory of biological growth

The Limits of Software

- Laws of physics
- Laws of software
- Challenge of algorithms
- Difficulty of distribution & concurrency
- Problems of design
- Importance of organization
- Impact of economics
- Influence of politics
- Limits of human imagination
Laws of Physics

- **The speed of light**
  - Has pragmatic implications for distributed systems

- **Relativistic effects**
  - There is no such thing as absolute time

- **Quantum effects**
  - There are theoretical as well as practical limits to information density

- **Thermodynamic effects**
  - Software is weightless/containers for software are not
  - Computation dissipates heat
Laws of Software

- **Sometimes we can’t do it**
  - Halting problem
  - Gödel's theorem/highly non-computable problems

- **Sometimes we can’t afford to do it**
  - Sorting problem/Towers of Hanoi/chess
  - Exponential time

- **Sometimes we just don’t know**
  - Traveling salesman/scheduling/bin packing
  - NP complete

Challenge of Algorithms

- Compression
- Photorealistic rendering
- Speech recognition
- Simulation
- Knowledge representation
- Intimate/massive parallelism
Difficulty of Distribution/Concurrency

- A distributed system is one in which the failure of a computer you didn’t even know existed can render your computer unusable (Leslie Lamport)

- The average developer does not have, as a core competency, the ability to develop secure, concurrent, and distributed systems
Problems of Design

- The entire history of software engineering is that of the rise in levels of abstraction
- The limitations of human understandability
- Building for resilience
- Sometimes worse is better
- The discovery of patterns
Importance of Organization

- All meaningful development is formed by the resonance of activities that beat at different rhythms
  - The activities of the individual developer
  - Social dynamics among small sets of developers
  - Dynamics among teams of teams

- Work products and work flows
- Parallel development
- The cacophony of stakeholders
- The tension of high and low ceremony processes
Impact of Economics

Performance = (Complexity)^{Process} \times (Team) \times (Tools)

- **Performance**: Effort or time
- **Complexity**: Volume of human-generated code
- **Process**: Methods, notations, maturity
- **Team**: Skill set, experience, motivation
- **Tools**: Software process automation
Influence of Politics

- Success as defined by the software development team is sometimes misaligned with success as defined by the management team
- Software as a strategic weapon
- Software as a pawn
Limits of Human Imagination

- The visionaries
  - Alan Kay, Danny Hillis, Marvin Minsky, Rodney Brooks, Bill Joy, Bill Gates, Adele Goldburg, Tim Berners-Lee, Nicholas Negroponte...

- The dreamers
  - Arthur C. Clarke, Neal Stephenson
Software development has been, is, and will remain fundamentally hard
What We Know

**Fundamentals**
- Craft crisp and resilient abstractions
- Maintain a good separation of concerns
- Create a balance distribution of responsibilities

**Process**
- Grow a system’s architecture through the incremental and iterative release of testable executables
What We Know

- Reuse
- Patterns
- Languages
- Architectural codification of certain domains
Representing Software Architecture

Logical View
End-user
Functionality

Implementation View
Programmers
Configuration management

Process View
System integrators
Performance
Scalability
Throughput

Deployment View
System engineering
System topology
Communication
Provisioning

Use Case View

Conceptual

Physical

Kruchen, P. The 4+1 Model View, IEEE Computer, vol. 12 (6), November 1995
Gallery of Software Architecture: Air Traffic Control

Simulation and Training

Display & User Interface

External Interfaces - Gateways

Flight management

Air Traffic Management

Aeronautical Information

Mechanisms Services

Basic elements
Gallery of Software Architecture: C3I

Gallery of Software Architecture @ http://www.booch.com/architecture/architecture.jsp?part=Gallery
Gallery of Software Architecture: Games

A 2D Game Circa 1994

- sound
- main/misc.
- simulation
- fast 2D graphics
- streaming file I/O

Gallery of Software Architecture @ http://www.booch.com/architecture/architecture.jsp?part=Gallery
Gallery of Architecture: Games

Gallery of Software Architecture @ http://www.booch.com/architecture/architecture.jsp?part=Gallery
Gallery of Software Architecture: Google

Gallery of Software Architecture @ http://www.booch.com/architecture/architecture.jsp?part=Gallery
Gallery of Software Architecture: Pathfinder

Gallery of Software Architecture @ http://www.booch.com/architecture/architecture.jsp?part=Gallery
Gallery of Software Architecture: Speech Recognition
 Movements in Web-centric Architectures

- Simple documents
- Colorful clients
- Simple scripting
- Rise of middleware
- Rise of simple frameworks
- Emergence of dynamic frameworks
- Semantic web
Handbook of Software Architecture

- No architectural reference exists for software-intensive systems
- Goals of the handbook
  - Codify the architecture of a large collection of interesting software-intensive systems
  - Study these architectural patterns in the context of the engineering forces that shaped them
  - Satisfy my curiosity
Preservation Of Classic Software

- No comprehensive and intentional activity has yet been undertaken to preserve the industry’s seminal software artifacts
- There are a number of reasons to act now
  - Many of the authors of such systems are still alive
  - Many others may have the source code or design documents for these systems collecting dust in their offices or garages
  - Time is our enemy

Computer History Museum @ http://www.computerhistory.org
How We Got Here

1910s beginning of automation
1920s beginning of expansion
1930s beginning of dependence
1940s beginning of von Neuman machines
1950s rise of the machines
1960s rise of the languages and methods
1970s death of the mainframe
1980s age of the personal computer
1990s age of the Internet and new methods
2000s retrenchment
The Current State

- **The typical software-intensive system is**
  - Continuously evolving
  - Connected, distributed & concurrent
  - Multilingual and multiplatform
  - Secure
  - Autonomic

- **Most systems are actually systems of systems**
  - Services and other messaging mechanisms dominate
Where We Are Going

2010s age of transparency
2020s total dependence
2030s rise of the machines
The Future State

- Every advance leading to the future state of the world requires the presence of software yet-unwritten as of today
Languages & Algorithms

- Most programmers still write algorithmic snippets in the context of a sea of objects
- Legacy XML, Java, C++, and UML persist
- Some algorithmic breakthroughs have emerged
  - Searching massive quantities of information is still a bit of a struggle
- Domain-specific frameworks are mainstream
Platforms

- Moore’s law has died
- The typical personal computer contains multiple processors, a petabyte of main memory, an exabyte of external memory, and untethered terabit connectivity
- Virtual high resolution displays dominate; 3D windows, mice, gestures, and voice are the usual mechanisms for interaction
- Form factors will change such that most personal computers will be wearable or embedded; most software is embedded in devices
Operating Systems & Middleware

- Operating systems have largely been commoditized
- Middleware that does transaction isolation, load balancing, resource management, and data access still dominates
  - but it too has largely been commoditized, forcing the platform vendors to keep growing the value pile
More than ever, the network is the computer

- Monolithic -> client/server -> Web -> grid

Network access is a global utility

Not everything is an enterprise system, but most applications are connected to several
Security

- New kinds of cybercrime have arisen
  - Unlimited piles of money still do not yield secure systems
  - Air gaps are still not enough

- Rolling failures still plague some systems
Autonomics

- No computer has yet passed the Turing Test (but we have come close)
- Most interesting systems exhibit signs of agency and self-repair

Star Trek@ http://www.startrek.com
Developer Experience

- Most developers have grown up believing that the Internet has always existed.
- Most programming occurs on the edge of a system and in the interstitial spaces among systems.
- Most programming is now done by domain-specific developers who only incidentally know how to program.
- There have been only incremental improvements in programmer productivity and the programming model:
  - CLI -> IDE -> XDE -> CDE
It is a tremendous privilege to be a software professional

it is also a tremendous responsibility
The Promise
The Limits
The Beauty of Software

Grady Booch
IBM Fellow