# Evolutionary Design

Idaho State University | Computer Science

Isaac Griffith

CS 4422 and CS 5599
Department of Computer Science
Idaho State University



ROAR

Idaho State University
Computer Science

# Outcomes

At the end of Today's Lecture you will be able to:

- Describe a brief history of design.
- Describe how the internet revived evolutionary design in software engineering.
- Describe the costs of software engineering in today's world.



ROAR

# Inspiration

"Every large system that works started as a small system that worked." – Anonymous

# A Brief History of Design

# Some Historical Perspective

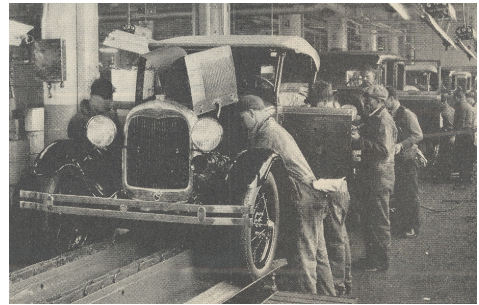
- Building new technology incurs several **costs**
- In this talk I will separate costs into **four areas**
  1. **Design**
  2. **Production**
  3. **Distribution**
  4. **Support**
- Over time, the relative amount of these costs have **continuously changed**
- We started with the ability to **evolve our designs** slowly

# Pre-1850: Hand-crafting

- **Design evolved** over time, each new object better than the last
  - Low **design** costs
- **Very high production** cost–weeks of labor
- Low **distribution** cost–customers walked into the shop
- Little or no **support** cost

# 1850s: Assembly Line

- **Manufacturing** started to change this equation
- The same design quickly put into thousands of products
  - Much higher **design** costs
- **Very low production** cost
- **Distribution** costs started to increase
- **Support** costs increased–but were outsourced



Idaho State University
Computer Science

# 1900s: Automated Manufacturing

- **Robots** increased the speed and efficiency of production
- Instead of training people, **design** costs now included creating expensive robots
- **Production** cost continued to decline
- **Distribution** costs continued to increase
- **Support** costs continued to increase



ROAR

# Post-WWII Worldwide Distribution

- **Design** costs continued to increase
- **Production** costs continued to decline
- **Distribution** capabilities increased exponentially, decreasing cost
- **Support** started to become "**replace**"


# 2000s: Free Trade

- This process has **continued** …
  - Free trade agreements
  - Cheap oil
  - Decreases in shipping costs
  - Decreases in production costs
- The **ultimate effect**?

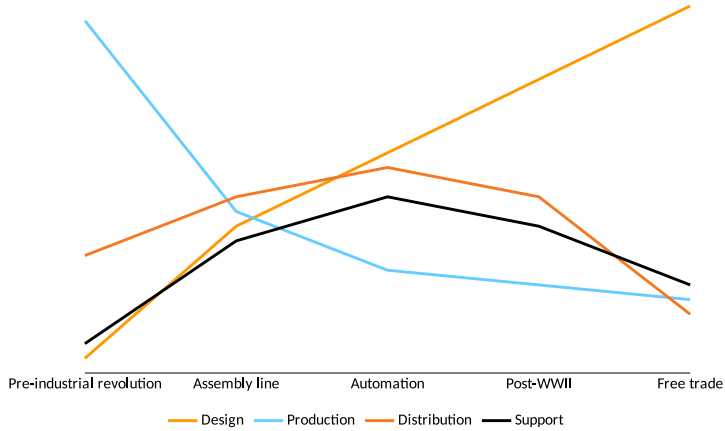
**Design is very expensive**
**production, distribution, & support**
**are cheap**

**Manufacturing defeated evolutionary design!**

**We now emphasize quantity over quality**

Idaho State University
Computer Science

# Relative Cost Trends






ROAR

# Losing Evolutionary Design

- Thousands of products are incredibly **cheap**
- Many products are very **low quality**
- Designed to **last a few months** or years, instead of decades
- Instead of **evolution**, we have
  - **Maintenance**
  - Or **replacement**

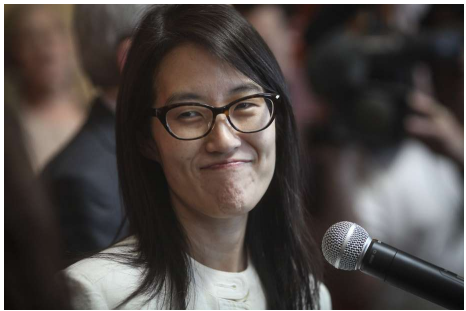
**But we lost something wonderful**
**Craftsmanship**

# A Return to Evolutionary Design in Software Engineering

Idaho State University
Computer Science

# Where is Software Engineering?

## Ummm ... Excuse me, Professor ...

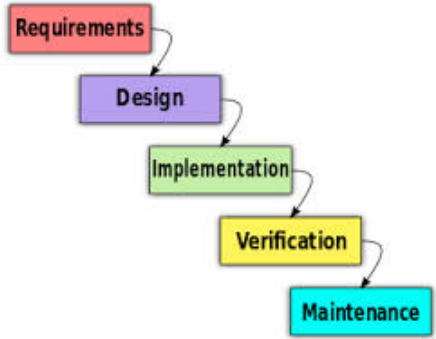


## What does this have to do with SOFTWARE ENGINEERING???

ROAR

# Traditional Software Development

- **Production** cost for software is very low
- **Distribution** cost is substantial
  - Includes marketing, sales, shipping
- **Support** costs escalated
- Software splits design into **design** and **implementation**
  - **Both very expensive!**
- Instead of one design for each artifact, software has one design for **many** artifacts



# 1900s Software Costs

- Millions of **customers** skewed costs to the back end
  - High support costs
  - High distribution costs
- New versions **shipped** every 4 to 6 years
  - MS Office, CAD, compilers, operating systems
- Software needed to be "**perfect out of the box**"
  - Very **expensive design**
  - Very **expensive implementation**—including testing more than 50% of the cost

**Software evolution was very slow!**

Idaho State University
Computer Science

# Research Agenda

- The need to be "**perfect out of the box**" heavily influenced decades of SE research
  - **Formal methods**
  - **Modeling** the entire system at once
  - **Process**
  - **Testing** finished products
  - **Maintenance** in terms of years
- Much of our **research focus** and results assume:
  - High **design** costs
  - High **implementation** costs
  - High **distribution** costs
  - High **support** costs

ROAR

# Distribution Costs

- In the 1980s, technology started **driving down** distribution costs for software …


# Usability and Support

As **usability** started to increase ...

The need for **support** decreased

**Then the Internet changed everything**

# 2000s: The Web

1. The web rearranged the importance of quality criteria, including making **usability** and **reliability** crucial
2. The web created a new way to **deploy** and **distribute** software

Idaho State University
Computer Science

# Web Deployment

Traditional software deployment methods

1. Bundle
2. Shrink-wrap
3. Embed
4. Contract
5. **Web Deployment**

ROAR

# Distributing Web Software

- **Desktop software** can be distributed across the web
  - **Zero-cost** distribution
  - **Instantaneous** distribution
  - This allows more **frequent updates**
- **Web applications** are not distributed at all in any meaningful sense
  - Software resides on the **servers**
  - **Updates** can be made weekly … daily … hourly … continuously!
- **Mobile applications** allow the artisan to come into your “home” to improve that rocking chair

# Evolutionary Design

- Near-zero **production** costs …
- Immediate **distribution** …
- Near-zero **support** costs …

**This resuscitates evolutionary design!**

# Evolutionary Software Design

- **Pre-Web** software design & production
  - Strive for a **perfect design**, **expensive development**
  - **Deployed** a new version every 4 to 6 years
  - **Evolution** was very slow
- **Post-Web** software production
  - Initial "**pretty good**" design and development
  - **Slowly** make it bigger and better
  - Faster **evolution**
  - **Immediate changes to web applications**
    - **Automatic updates** of desktop applications
    - **Software upgrades pushed out** to mobile devices hourly
    - **Replacing chips** in cars during oil changes

**This changes all of software engineering!!**

# Industry Impacts

- How often is **GMail** updated?
  - Daily … sometime hourly
- **Sarah Allen** invented youtube
  - She advises people with 5-year ideas to think about how they can achieve 1 idea in 6 months, and grow to the 5-year goal.

# Current Software Engineering

- Software will not be designed and built – **Software Grows**
- Software needs to take responsibility for its own **behavior**
- **Waterfall** is now, finally, thankfully, completely dead
- **Testing** must focus on evolution, not new **software**
- **Yes, the web really does change EVERYTHING**

# Software Process

- We have already seen **process changes** that are a direct result of web deployment & distribution
- **Agile** processes goals:
  - Have a **working, preliminary, version** as fast as possible
  - Continue **growing** the software to have more functionality and better behavior
  - Easy and fast to **modify**
  - Adapt to sudden and **frequent changes** in planned behavior
- Agile processes are **widely used**
- Results are mixed, but **use is growing** quickly

# Architecture

- Software architects often assume their high level design will **not change** throughout development
  - And the lifetime of the system
- It is not clear how this supports **software growth**, **rapid deployment**, and **instantaneous distribution**
- Is this attitude **compatible** with agile processes?
- How does architecture design interact with **refactoring**?

Idaho State University
Computer Science

# Software Self-Responsibility

- Evolutionary design means we **cannot know** everything software will ever do
- **Self-management** means the software adapts behavior to runtime changes–crucial for evolutionary design
- **Fault localization** tries debug automatically, which can dramatically cut the human effort required to fix software after testing
- **Automated defect repair** goes one step further, and attempts to automatically fix faults

**Are you ready for the adaptive software revolution?**

ROAR

# Software Testing

- **Test-driven design** uses tests to drive requirements
  - Every step is evolutionary
- We must stop thinking of **regression testing** as something special done "late in the process"
  - Virtually all testing is now regression testing
- **Model-based testing** allows test design to quickly and easily adapt to changes
- **Test automation** is the key to running tests as quickly as software is now changed
  - Model to implementation?
  - Test oracle strategy?

**TDD is an important part of this class**

Idaho State University
Computer Science

# Software Costs Then & Now

**Old**

- Design: High
- Implementation: High
- Production: Low
- Distribution: High
- Support: High

**New**

- Design: **Medium**
- Implementation: **Medium**
- Production: **Zero**
- Distribution: **Zero**
- Support: **Low**

ROAR

# Long Term Impact of Evol Design

- The end-result of large scale manufacturing was a heavy emphasis on **quantity over quality**
- The **web enables evolutionary design**, which can allow us to focus on quality over quantity.

**What engineer would NOT love that?**

# Are there any questions?