How the Web Brought Evolution Back into Design

Brittany Johnson
SWE 437

Adapted from slides by Paul Ammann & Jeff Offutt
Some engineering historical perspective

Building new technology incurs several costs
In today’s lesson, 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 costs** – 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
Quickly put same design into **thousands** of products
**Higher design** costs; **very low production** costs
**Distribution** costs started to increase
**Support** costs increased – but were outsourced
1900s: Automated manufacturing

Robots increased speed and efficiency of production

**Design costs** = create expensive robots

**Production cost** continued to decrease

**Distribution costs** continued to **increase**

**Support costs** also continued to **increase**
Post WWII worldwide distribution

**Design** costs continued to *increase*

**Production** costs continued to *decrease*

**Distribution** capabilities increased exponentially, decreasing cost

**Support** started to become “replace”
2000s: Free trade

This process had 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!

Now we emphasize quantity over quality
Despite all these “gains”...

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
Sooo...

Why do I care...

What does this have to do with software engineering???
Traditional software development

**Production** costs for software is *very low*

**Distribution** cost is *substantial*
  - includes marketing, sales, shipping

**Support** costs escalated

Software splits design into **design** and **implementation**
  - both are 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 4-6 years
- MS Office, CAD, compilers, operating systems

Software needed to be “perfect out the box”
- Very expensive design
- Very expensive implementation – including testing more than 50% of the cost

Software evolution was very slow!
Effects on research

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
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 World Wide Web 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.
Web deployment

Traditional software deployment methods

1. Bundle
2. Shrink-wrap
3. Embed
4. Contract

5. Web deployment
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
The rebirth of evolutionary design

Near-zero production costs...

Immediate distribution...

Near-zero support costs...

This resuscitates evolutionary design!
Evolutionary software design

Pre-web software design & production
- Strived for a **perfect design, expensive development**
- **Deployed** a new version every 4-6 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!**
Impact on industry

How often is Google mail updated?
- Daily ... sometimes hourly

Piazza class support system
- Jeff report a bug the first day he used it
- It was fixed before he met for class that afternoon

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

The web really does change EVERYTHING!
Software process

We have already seen **process changes** that are a direct result of web deployment & distribution

**Agile** process 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
Software architecture

Software architects often assume their high level design will not change throughout development - and system lifetime. 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?

Your generation needs to deal with this!
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 to 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
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

**TDD** is an important part of this class
## Software costs (then & now)

<table>
<thead>
<tr>
<th></th>
<th>Old</th>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Implementation</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Production</td>
<td>Low</td>
<td>Zero</td>
</tr>
<tr>
<td>Distribution</td>
<td>High</td>
<td>Zero</td>
</tr>
<tr>
<td>Support</td>
<td>High</td>
<td>Low</td>
</tr>
</tbody>
</table>
Long term impact of evolutionary 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 wouldn’t LOVE that?!