CSC 436, Fall 2017

Software Development Processes

Ravi Sethi

Software Development Processes

"Software development … requires choosing the most appropriate tools, methods, and approaches for a given development environment."

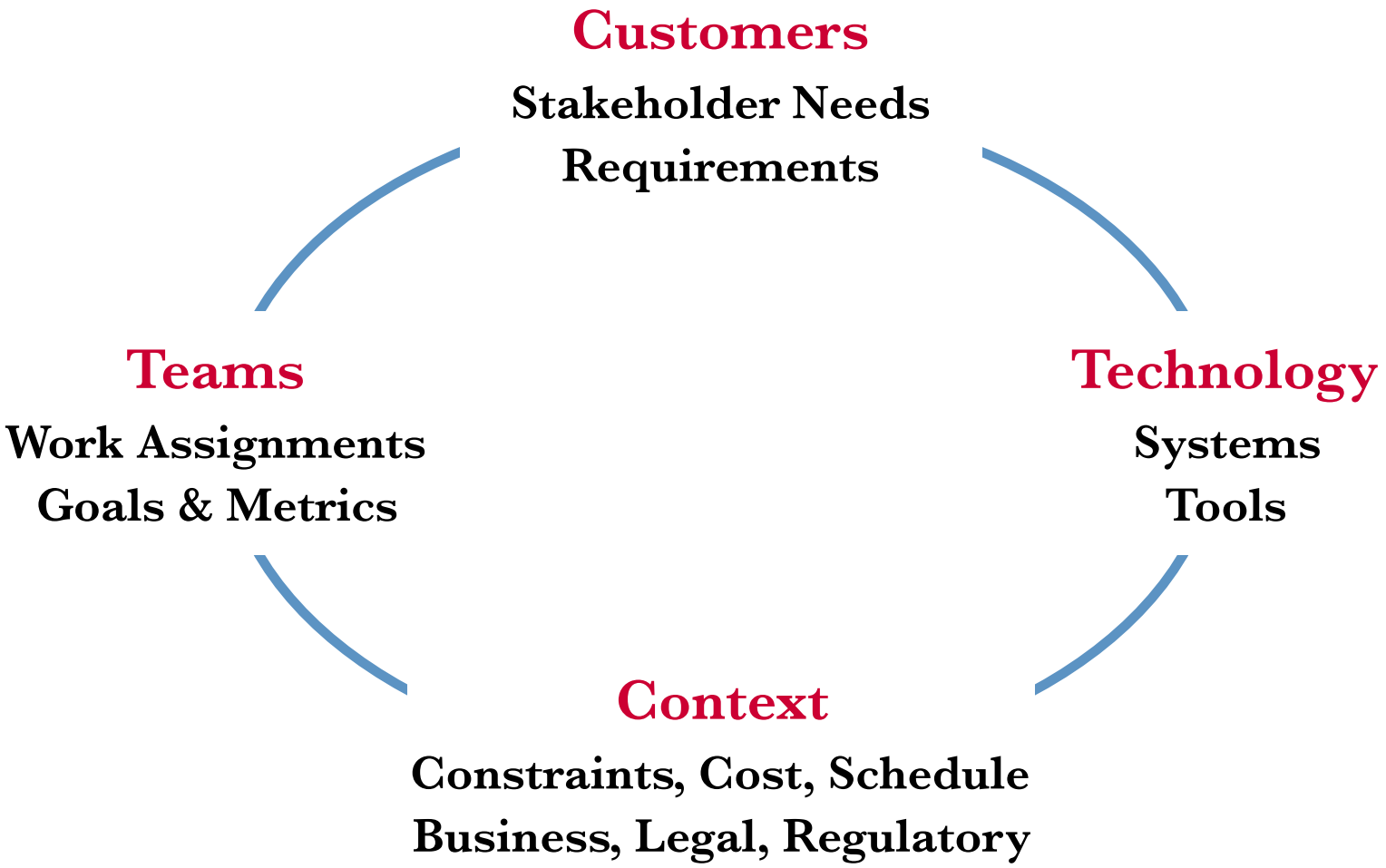
Processes help assure that …

Examples

- The system does what the customer/market wants or what the organization wants to build
- Work assignments are properly apportioned and teams at different sites understand how their work interacts with the work of others
- The code works together; e.g., modules work together to produce the desired result or that requirements are met
- Work assignments can be completed in time for the project to meet its release date(s)

Introduction to Processes

Processes guide all activities: who will do what by when?



Introduction to Processes

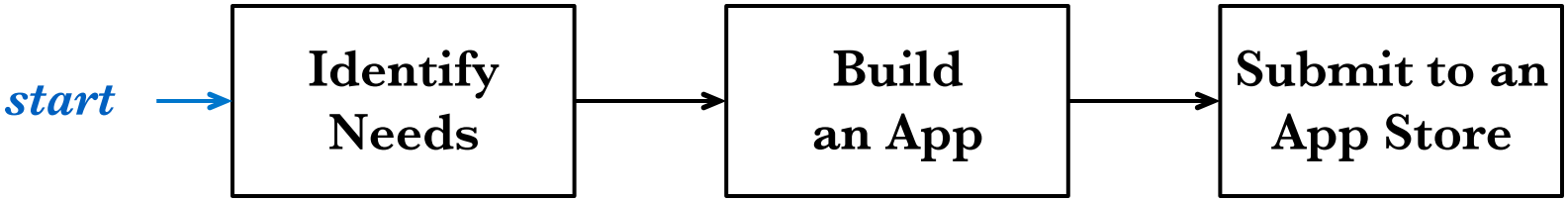
"Software Life Cycle" is a synonym for Process

- **Definition of *software development process***
  - Systematic method for meeting the goals of a project by
    - defining and organizing development activities
    - determining the roles and skills needed to do the project
    - setting criteria and deadlines for progressing from one activity to the next

- **Dictionary definition of process:**
  - A series of actions to produce a result

A Simple Process

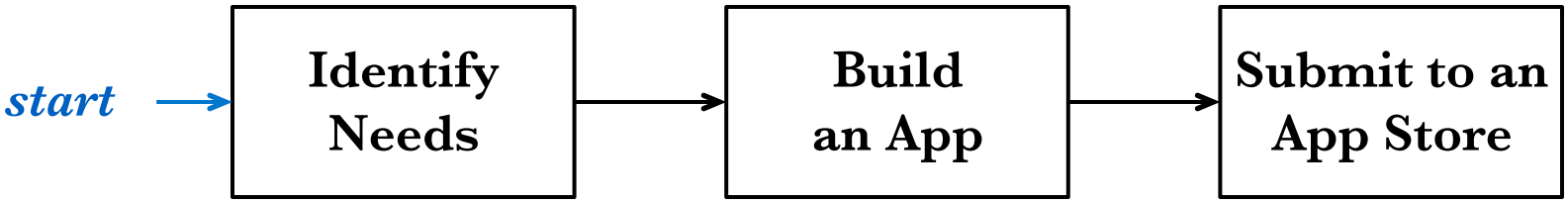
A process diagram shows activities and transitions



- **What are the activities?**
  - Activities are represented by the boxes
- **What are the roles?**
  - What roles and skills would you want to do the job?
- **Criteria for progressing from an activity?**
  - Arrows represent progression between activities.

A Simple Process: Possible Roles

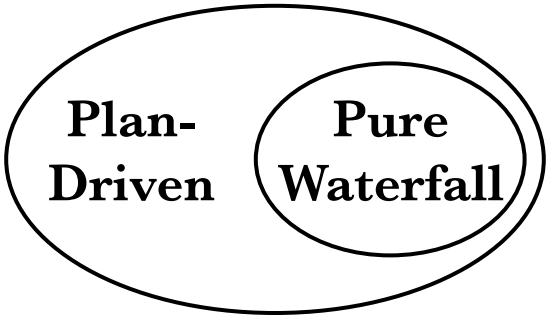
Roles: customer, product owner, developer, app-store mgr.

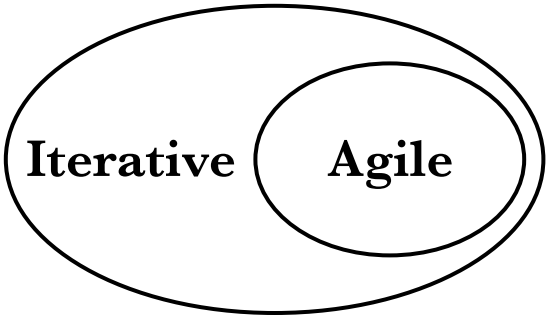


- **Identify Needs**
  - Product owner with selected customers, helped by developers
- **Build an App**
  - Developers build the app to the product owners requirements
- **Submit to an App Store**
  - Completes when app-store manager has verified

Process Classes: Informal Description

Process classes are also known as process models





Plan Driven Processes

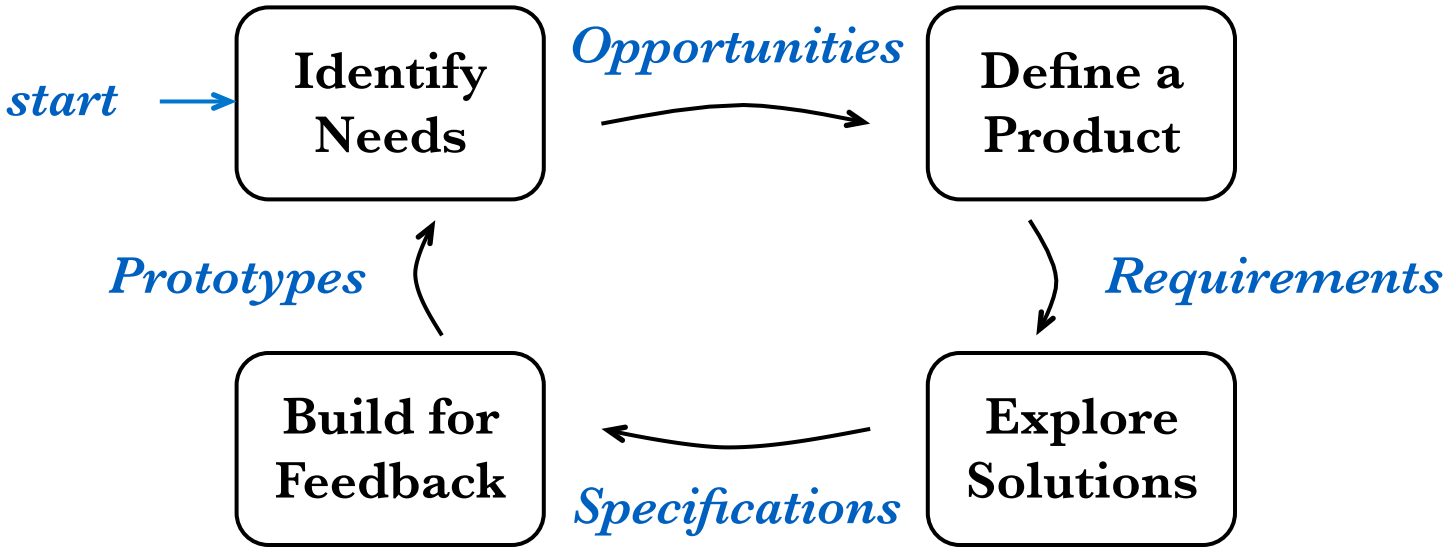
- Development driven by up front planning, design, documentation
- Assumes stable requirements
- Minimal customer feedback during development
- Delivery times measured in months; e.g., 18-24 months

Iterative Processes

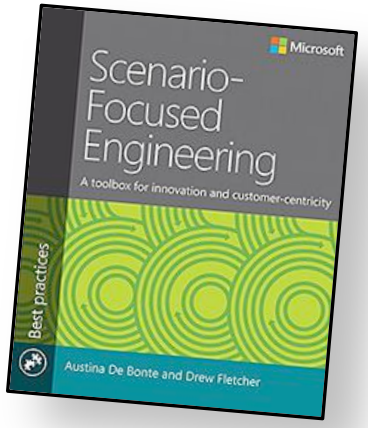
- Iterations deliver increasingly functional versions of the system
- Handles changing requirements
- Frequent customer feedback guides iteration planning
- Quick iterations, measured in weeks; e.g., weekly agile iterations

An Iterative Process

Based on a Microsoft Process: Scenario Focused Engineering



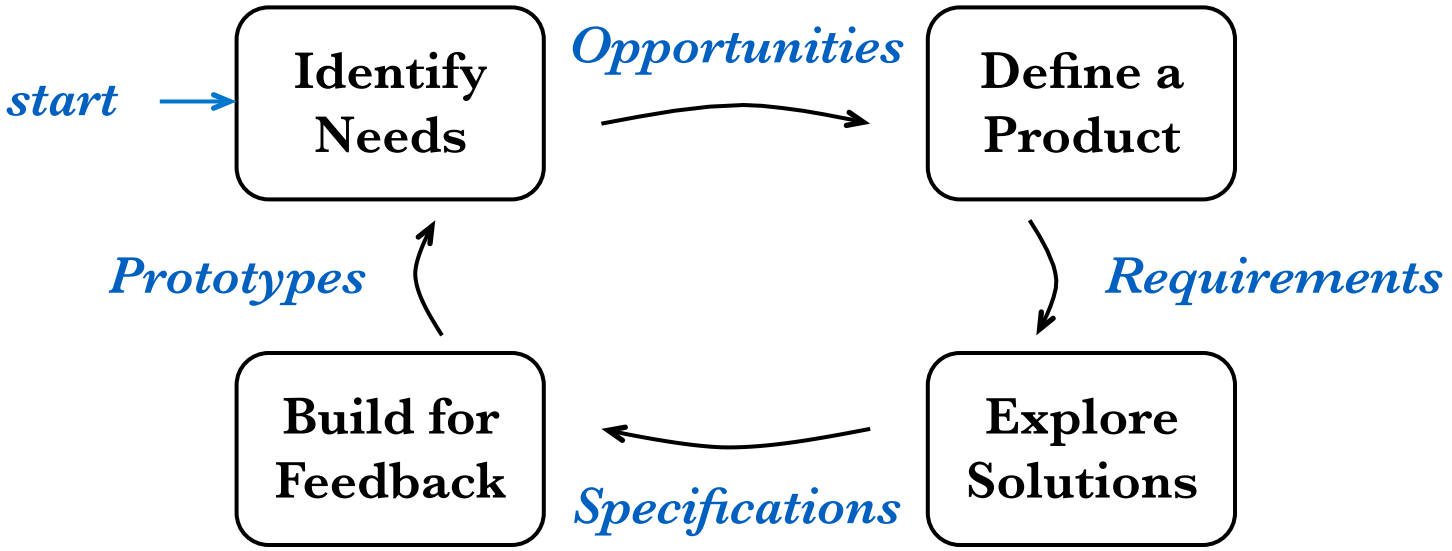
- **Bringing design thinking to engineers**
  - Training taken by 22,000 engineers since 2008



De Bonte and Fletcher [2014]

An Iterative Process

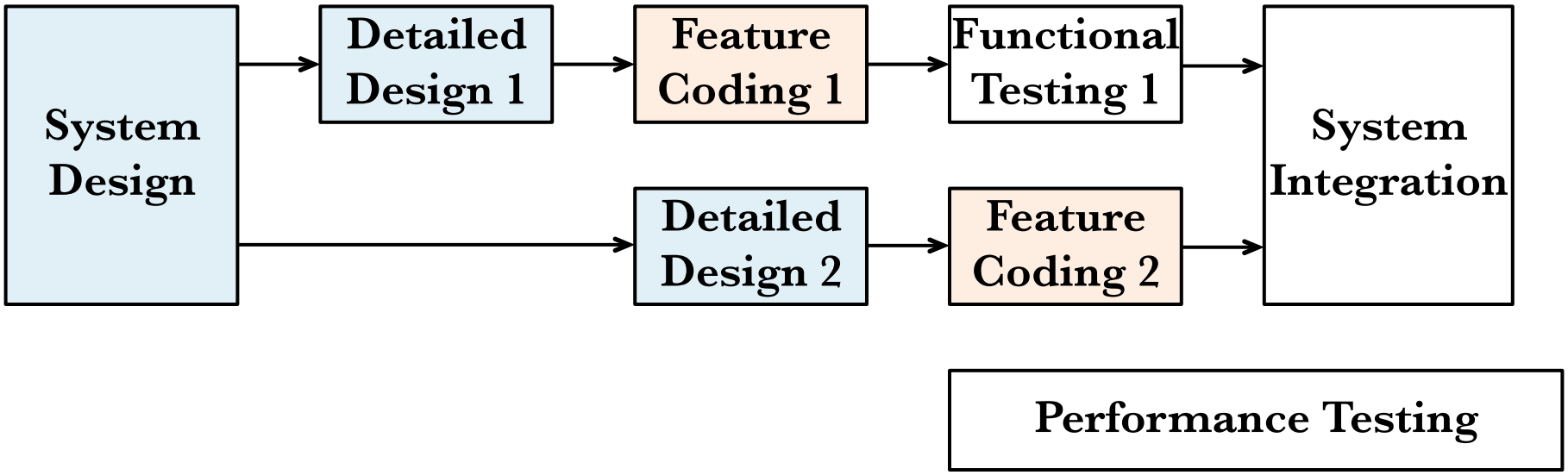
Activities produce *artifacts* – shown as labels on arrows



- **Overview of an iteration**
  - Identify Customer Needs: output is a list of opportunities
  - Define a Product: requirements for addressing some unmet needs
  - Explore Solutions: brainstorm to come up with design specs
  - Build for Feedback: prototypes that get better with each iteration

A Plan-Driven Process

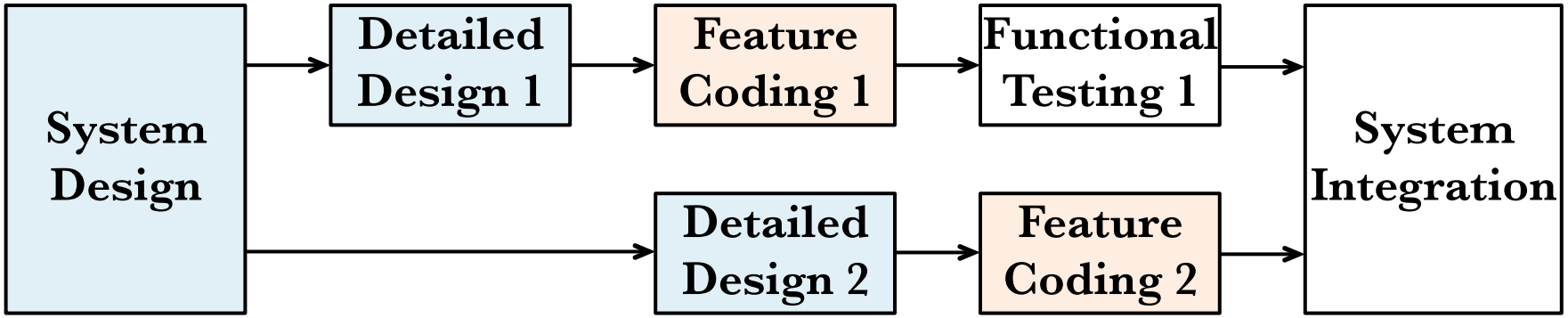
Simplification of the process for a highly reliable tech. product



- **Process focuses on building phases**
  - Customer interaction and product definition are not shown

A Plan-Driven Process

Takes planning to coordinate designers, developers, testers



- **In the original process, features were developed in parallel**
  - Note, designers can move from one feature to another in sequence
  - Developers can do coding for one feature then move to another
- **Original process tested early and often**
  - Functional testing, system integration, performance testing
  - Unit testing is not shown

Trend: To iterative and faster deliver times

Examples of successful projects

PLAN DRIVEN

SAGE Air Defense
Prototyped all functions
prior to development

AT&T 5ESS
Telephone Switch
2 years per release

ITERATIVE

Space Shuttle Software
17 iterations, 31 months

Netscape 3.0 Browser
Monthly betas

CONTINUOUS DELIVERY

Amazon, Netflix, …
Parallel code lines
Multiple releases a day

1950s 1970s 1990s 2010s

Continuous Delivery Examples

Release engineering is beyond the scope of this course

Back in 2009

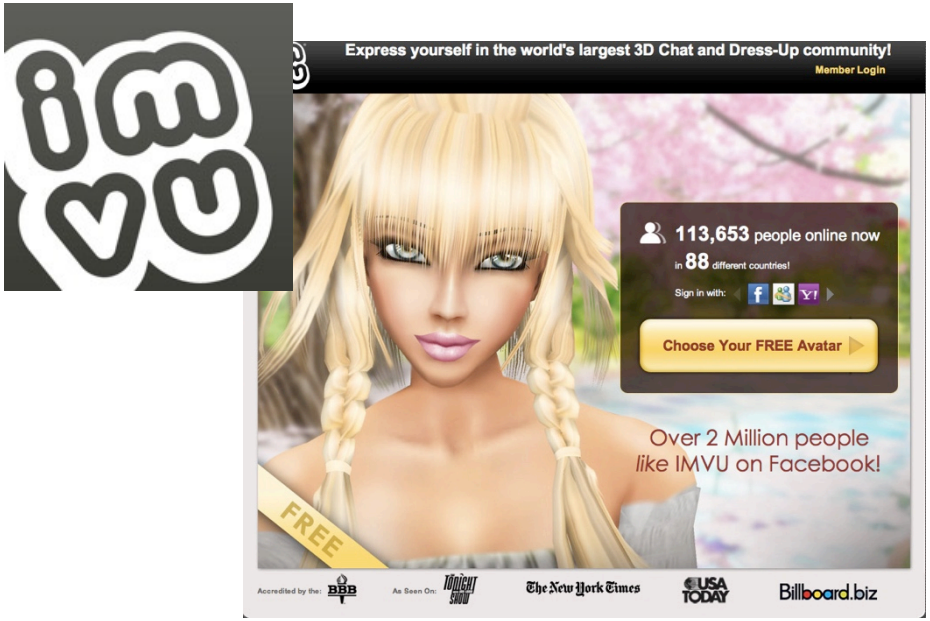
IMVU: "On average we deploy new code fifty times a day"

Integration: Continuous

15K Tests: 9 min.

Staging: 6 min.

Version Control System



2015: Feature-Based Releases

Release when it's done versus time-based releases

6 months	6 weeks	2 weeks (mobile)	Twice a day (web)	Dozens of times a day
openstack™				NETFLIX

Mozilla: Before and After

- **How long to "chemspill" release?**
  - ~~4-6 weeks~~ **11 hours**
- **How long to "new feature" release?**
  - ~~12-18 months~~ **6 weeks**
- **How many active code lines?**
  - ~~1 ½~~ **42**
- **How many checkins per day?**
  - ~~~15 per day~~ **325 per day**

Cycle Time ≠ Development Time

Mozilla Repositories

Nightly: with experimental features

Aurora: for web platform developers & early adopters

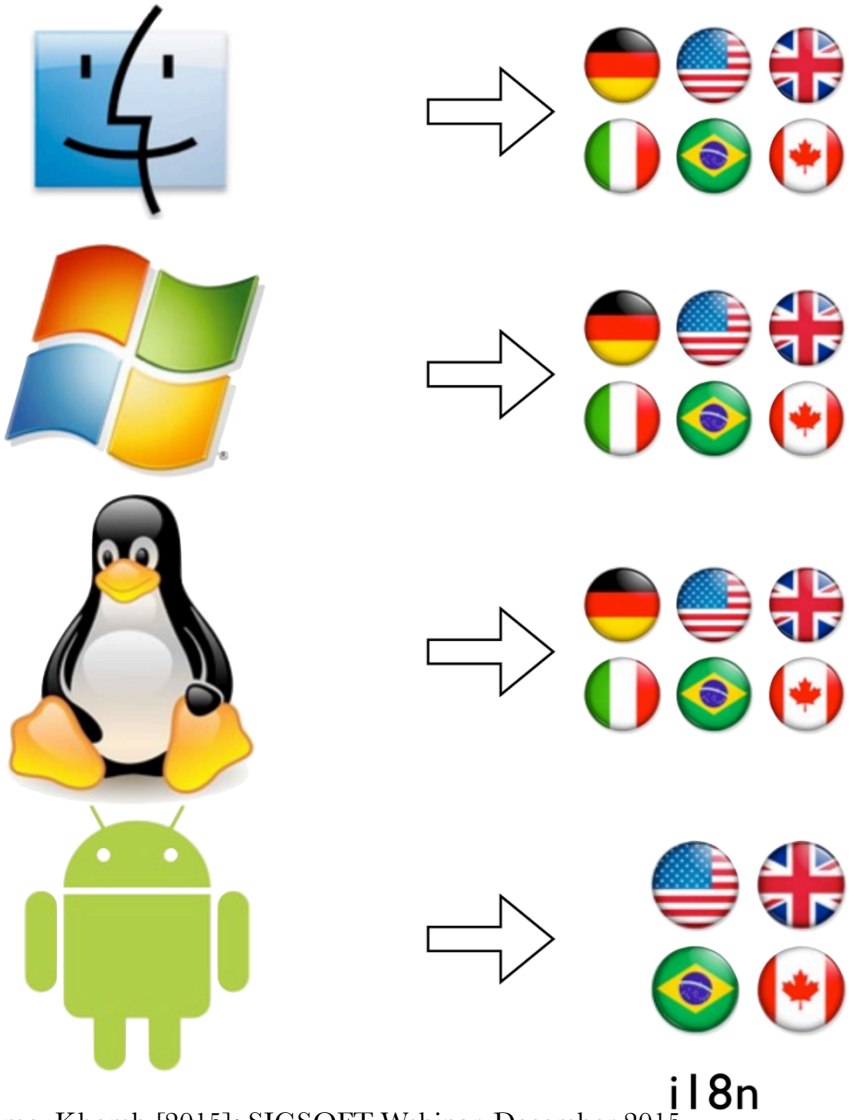
Beta: testing the next version before official release

Release: official release

https://wiki.mozilla.org/Release_Management/Release_Process

Proliferation of Versions

Generate incremental updates for each supported version



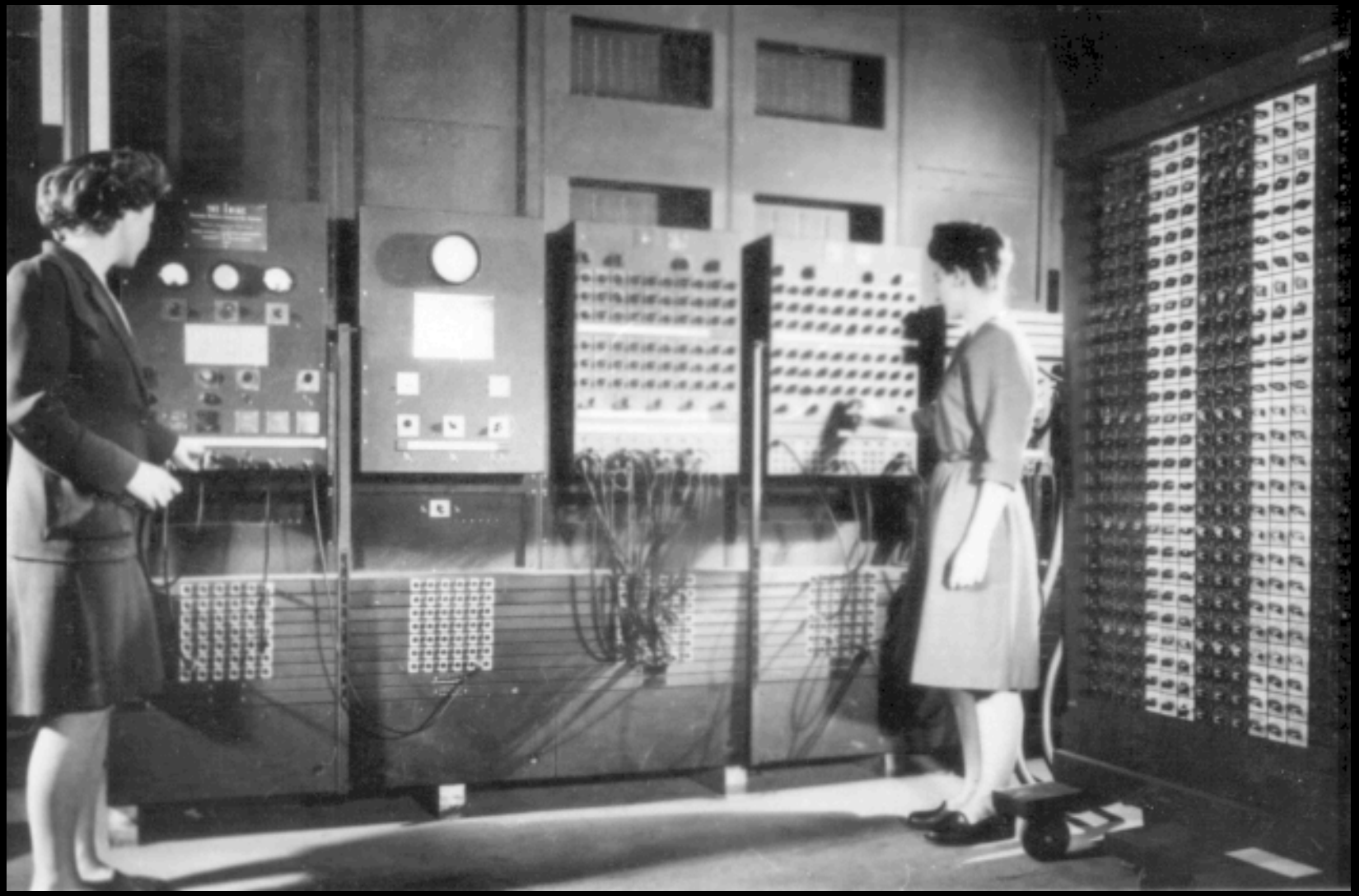
Early Developments: Pure Waterfall

Sequential: one phase completes before the next begins

Royce's [1970] waterfall diagram was a straw proposal

Software Development for the ENIAC (1948)

Early Programmers: Betty Jean Jennings and Fran Bilas



https://en.wikipedia.org/wiki/ENIAC -- US Army Photo

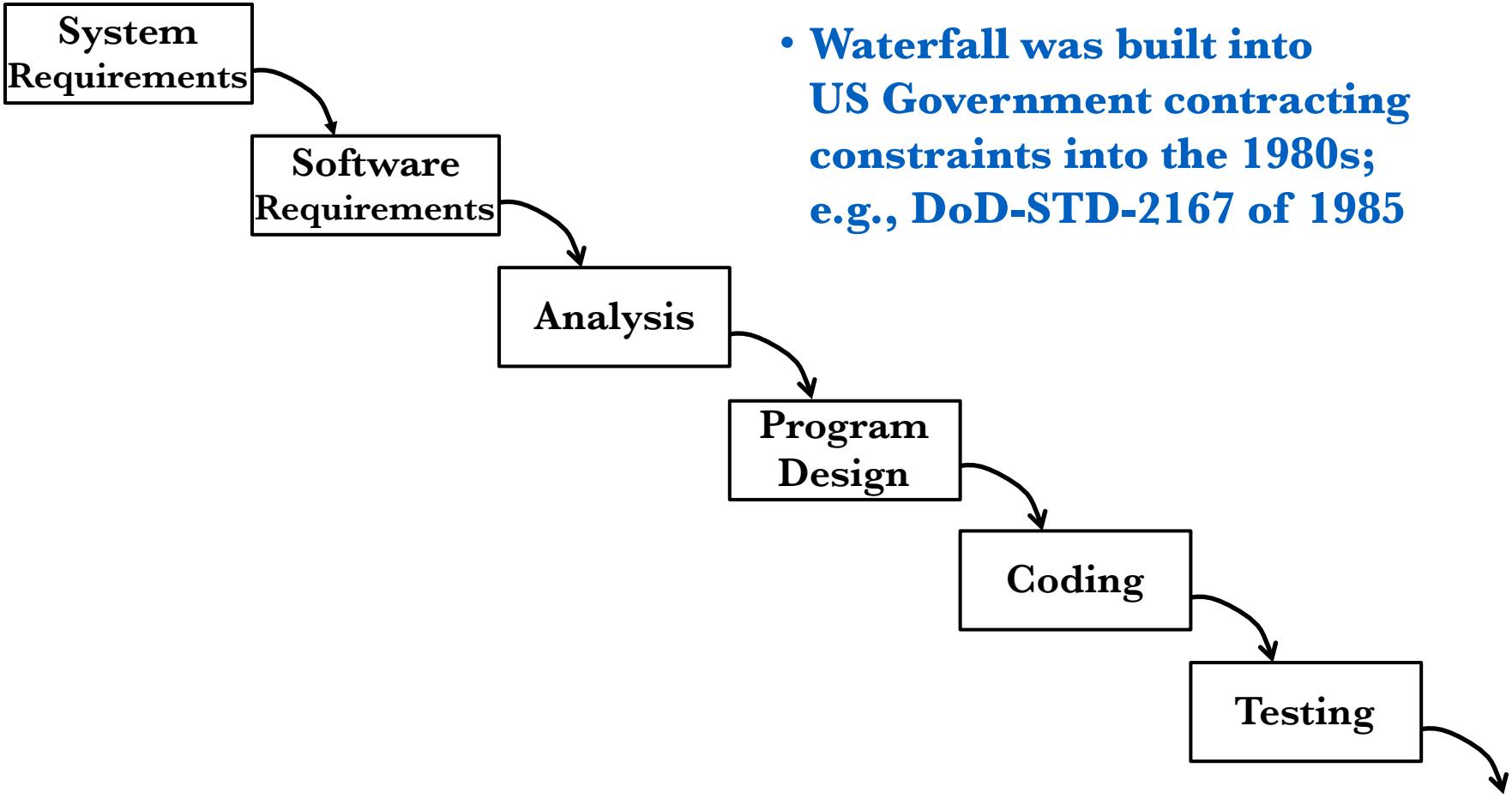
1950s: Code-and-Fix Programming

Efficient use of computing resources was all important

- **Computers were expensive and bulky**
  - One hour of computer time
  - was 300 times the cost of an hour of programmer's time
- **Computer industry faced a software crisis**
  - Need for systematic approaches, instead of code-and-fix

Waterfall Process: Sequential Activities

Offered order amid chaos



- **Waterfall was built into US Government contracting constraints into the 1980s; e.g., DoD-STD-2167 of 1985**

Source: Royce [1970]

DoD Reviews Promoted Waterfall

MIL-STD-1521B (June 4, '85) gates for software and hardware

- **System Requirements**: Review the Statement of Work
- **System Design**: Review total system requirements
- **Software Specification**: Review detailed system spec
- **Preliminary Design**: Top level design & the test plan (20 pages)
- **Critical Design**: Review(s) prior to coding
- **Test Readiness**: Procedures for testing and fixing
- **Functional Configuration**: Compliance with system specs
- **Physical Configuration**: Audit the documentation
- **Formal Qualification**: Review the performance
- **Production Readiness**

Problems with Pure Waterfall

Shortcomings were known all along

- **Requirements change**
  - Pure waterfall does not have customer involvement after requirements
  - Requirements changed before the project completed, years later
- **Issues surface late in the project's life cycle**
  - Problems can lurk undetected until Testing at the end
  - Result: redesign and rework; cost and schedule overruns

HealthCare.gov: Troubled Launch, Oct 2013

From Energy and Commerce Committee hearings

- **Biggest overhaul of US healthcare in decades**
  - Two year project with components built by several contractors
- **"Big Bang" testing, near the end of the project**
  - Contractors admitted that integration testing for the website began two weeks before launch
  - Another contractor admitted that full end-to-end system testing did not occur until "the couple of days leading up to the launch."
- **At the time, $118 million had already been spent on the website alone.**

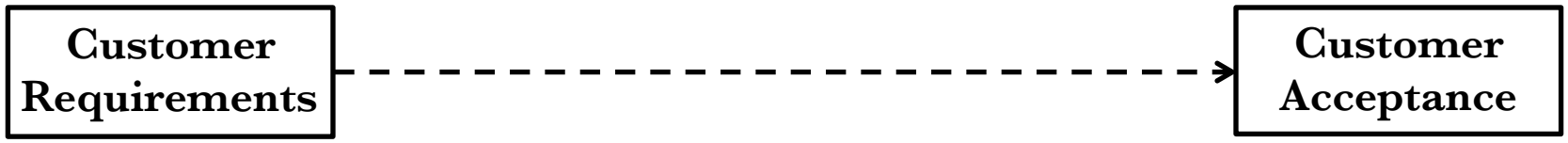
Source: House of Representatives: Hearings on PPACA Failures

V-Process

SAGE was a successful distributed air defense system in the 1950s that grew to 24 radar data collection centers and 3 combat centers across the US.

Pair Specifications and Testing

Can separate test planning and test execution



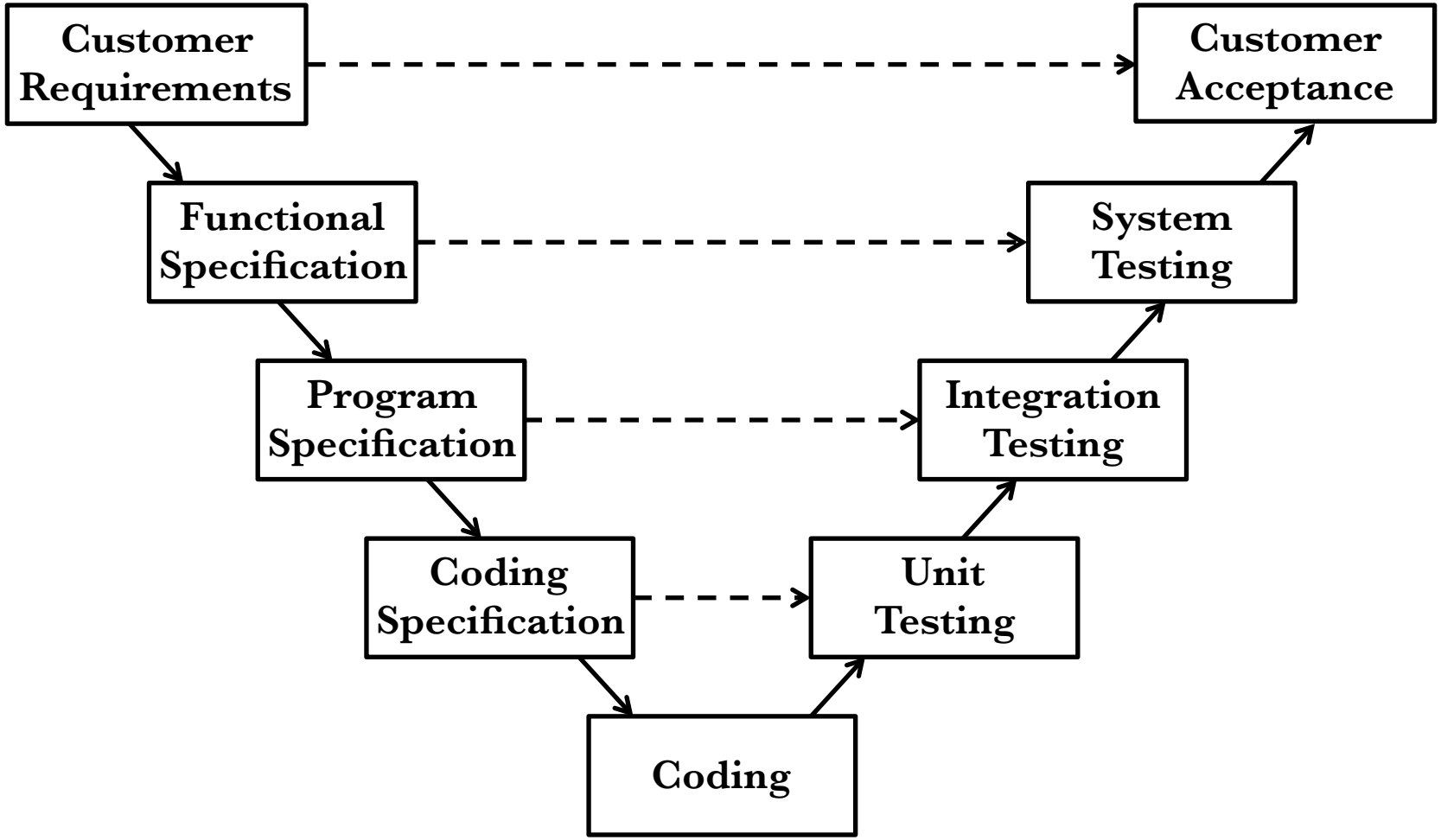
Acceptance Tests can be Planned early

Acceptance Tests are Run at the end

- **Note**
  - Acceptance Tests can be defined based solely on the Customer Requirements, independent of implementation
  - At the next level of specification, System Tests can be defined solely on the Functional Specification
  - And so on – see next slide

V-Process: Pair Specification and Testing

Waterfall variant



SAGE Air Defense System: Circa 1956

Source: https://commons.wikimedia.org/wiki/File:IBM's_$10_Billion_Machine.jpg

SAGE Air Defense System: 1950s

Successful system used Disposable Prototype + V-Process

- **SAGE = Semi-Automated Ground Environment**
- **Ambitious distributed system grew to**
  - 24 radar data collection centers and 3 combat centers across the US
  - 100,000 lines of assembly code
- **Disposable Prototype**
  - To explore tradeoffs between performance, cost, and risk
  - *"Twenty people understood in detail the performance of those 35,000 instructions; they knew what each module would do, they understood the interfaces, and they understood the performance requirements."*

Source: Bennington [1956], Bennington [1983]

Successful Plan-Driven Processes

Defer detailed up-front planning until the risks in the later phases of development have been thoroughly assessed

Overcoming the Limitations of Waterfall

Defer planning until …

- **Ensure that requirements are relatively stable**
  - Either freeze requirements
  - Or allow them to vary within known parameters that are planned for
- **Surface issues early**
  - Anticipate development challenges
  - Test early and often
- **Early testing**
  - Software systems have components
  - Can start testing one component before the other is fully designed

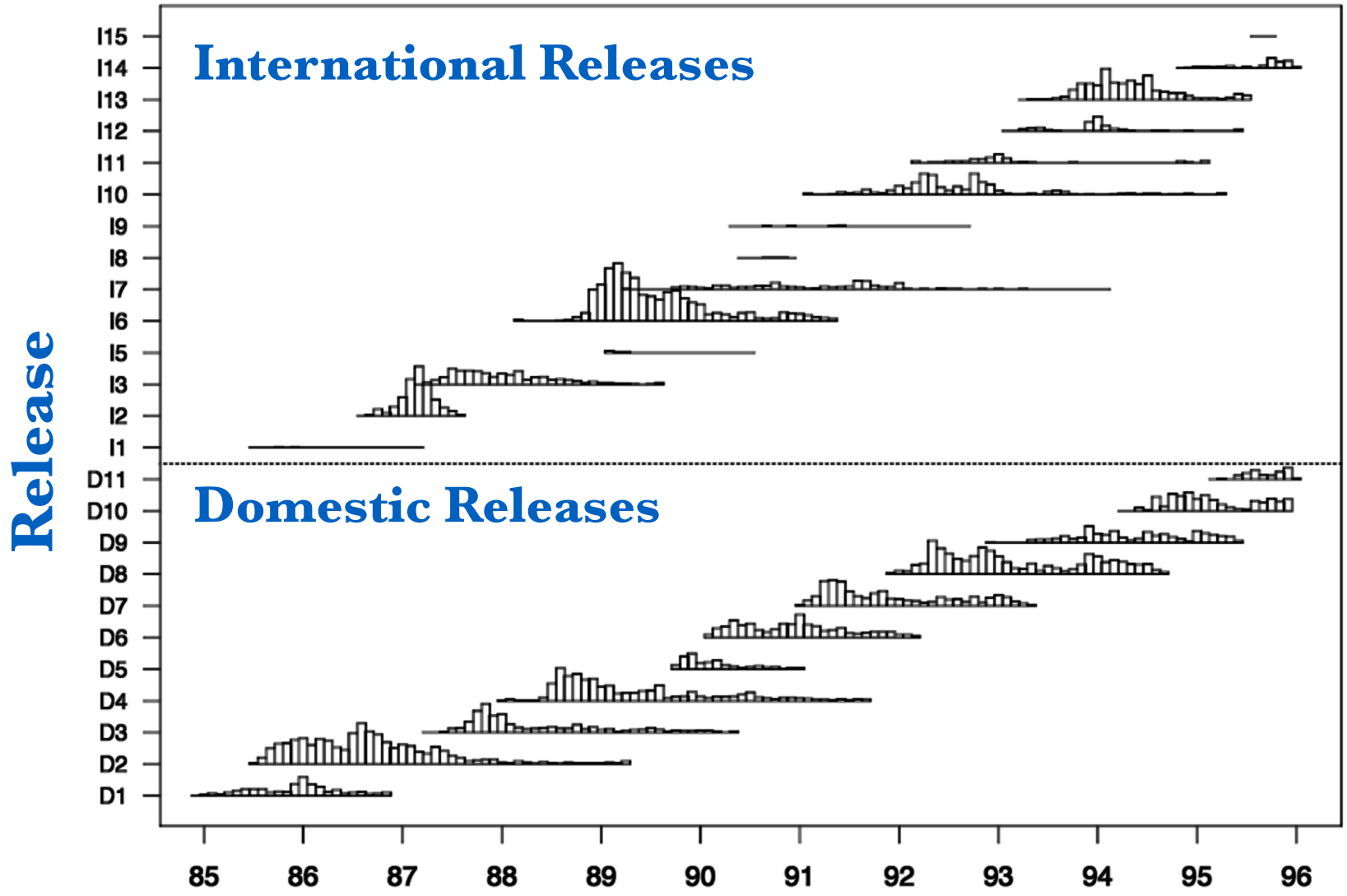
Overcoming the limitations of Waterfall

Risk reduction techniques

- **Build a Disposable Prototype**
  - Used successfully by SAGE
  - Prototype all functions to understand requirements and design
- **Leverage Past Experience from Similar Projects**
  - Works for products with multiple releases
  - Works for product families
  - Used successfully for AT&T 5ESS switching system

AT&T 5ESS Switching System Releases

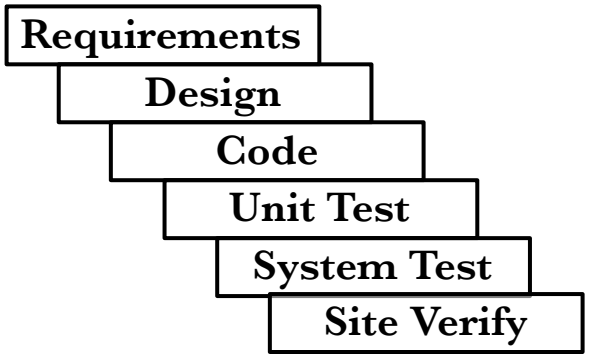
Each histogram represents work being done for one release



Source: Siy and Perry [1998]

AT&T 5ESS Switching System Releases

Notes based on historical data, 1985-1996



Phase lengths are not to scale; e.g. for one release, Coding took 10 months, while Unit Testing took 13.

- **Software Process**
  - Unprecedented parallel development
  - Overlapping phases within a release (see stylized figure to the right)
- **Highly reliable flagship product**
  - 10M lines of code divided into 50 subsystems, 3,000 developers
  - Roughly 2 years per release, for Requirements through Site Verify
- **Multiple releases; multiple versions per release**
  - Change Management Layer to track changes in a release
  - Configuration Management Layer to manage versions of files

Source: Siy and Perry [1998]

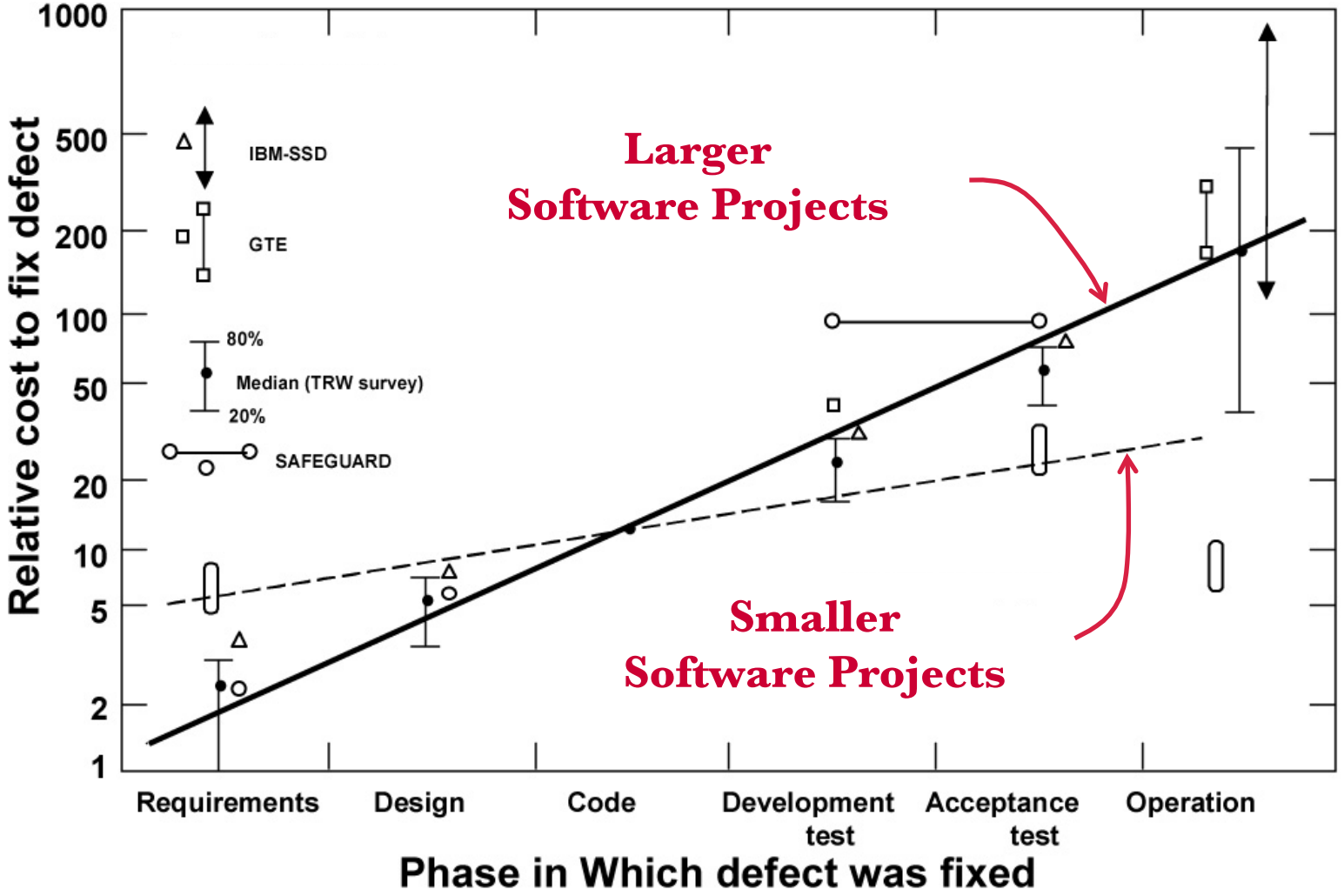
Cost of Change Curve

The earlier the fix, the lower the cost

Cost to Fix a Severe Software Defect

The later the fix, the greater the cost.

This 1976 chart due to Barry Boehm has been widely cited.

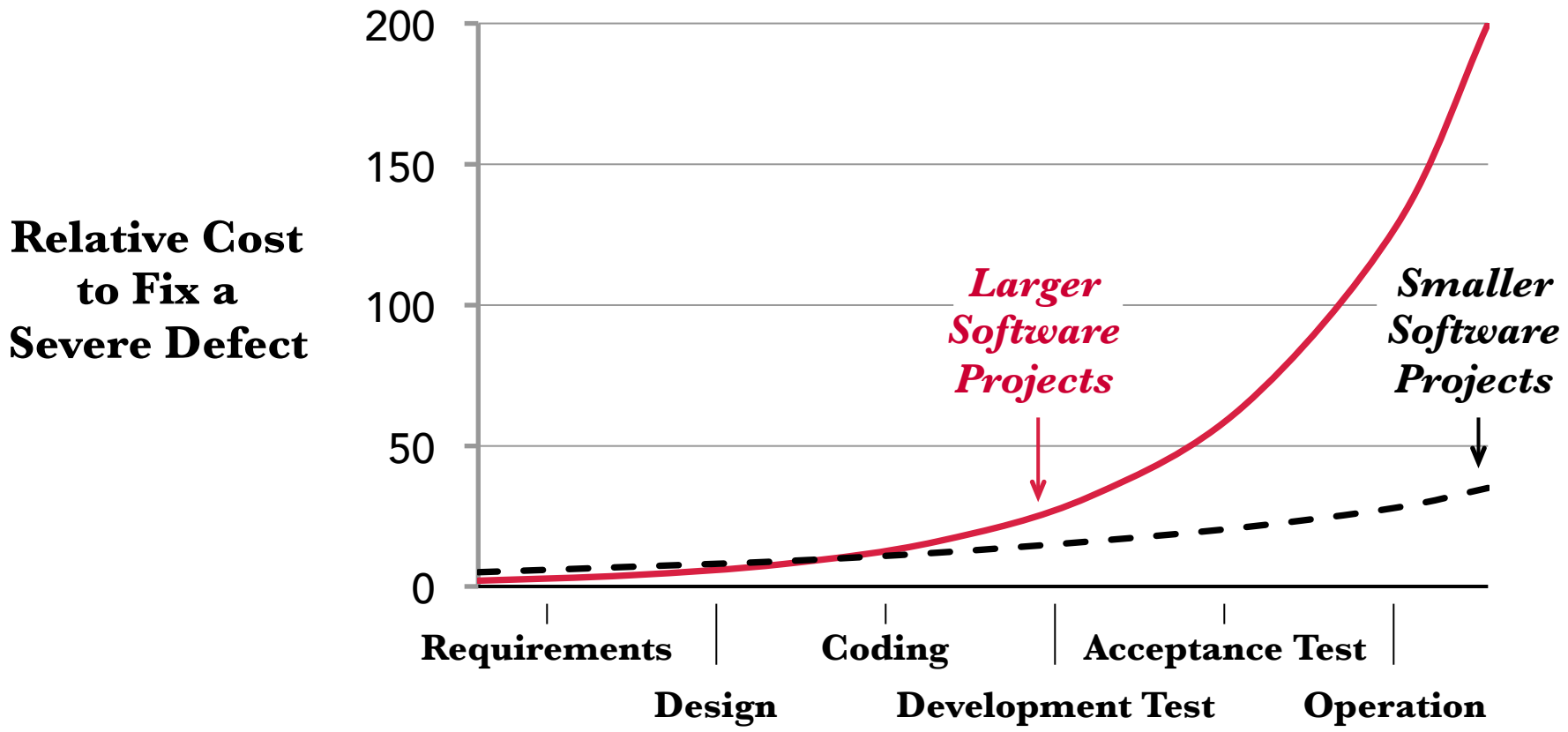


1976 chart, as presented in Boehm [2006]

Cost to Fix a Severe Software Defect

The later the fix, the greater the cost.

Same data: chart redrawn with a linear rather than a log scale



Data from 1976 chart, as presented in Boehm [2006]

~100:1 cost ratio for fixing severe defects

Between Initial Requirements and Operations in the Field

- **Data from a 2002 workshop**
  - IBM Rochester: 117:1 increase in effort
    - The increase was 13:1 between coding and testing, and then a further 9:1 between testing and operation.
  - Toshiba: 137:1 ratio for time to fix before and after shipment
    - Software factory of 2,600 workers
  - Other organizations reported similar experiences.

~2:1 cost ratio for fixing non-severe defects

Source: Shul [2002]

Cost of Change Curve

Cost of fixing defects is a proxy of the cost of changes

- What is the cost of changing requirements?
- What is the cost of a change during a project?
- Implications for how much to invest in design and architecture at the start of a project

Implications of the Cost of Change Curve

- **If the curve is steep**
  - Then fix earlier
  - Motivates Plan-Driven Processes
- **If the curve is relatively flat**
  - Then defer changes to allow requirements to settle
  - Fundamental assumption behind Agile Processes: they "flatten" the cost of change curve

CSC 436, Fall 2017

Iterative Processes

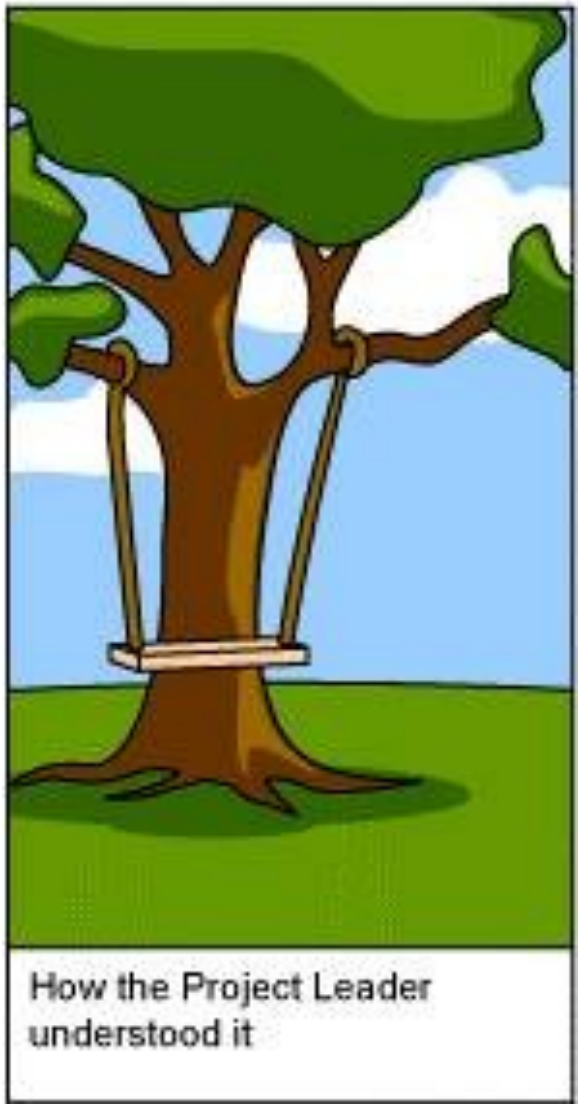
Ravi Sethi

Why are Requirements a Problem?

How the Customer Explained it

Source unknown. See http://www.businessballs.com/treeswing.htm .

How the Project Leader Understood It



Source unknown. See http://www.businessballs.com/treeswing.htm .

How the Analyst Designed It

Source unknown. See http://www.businessballs.com/treeswing.htm .

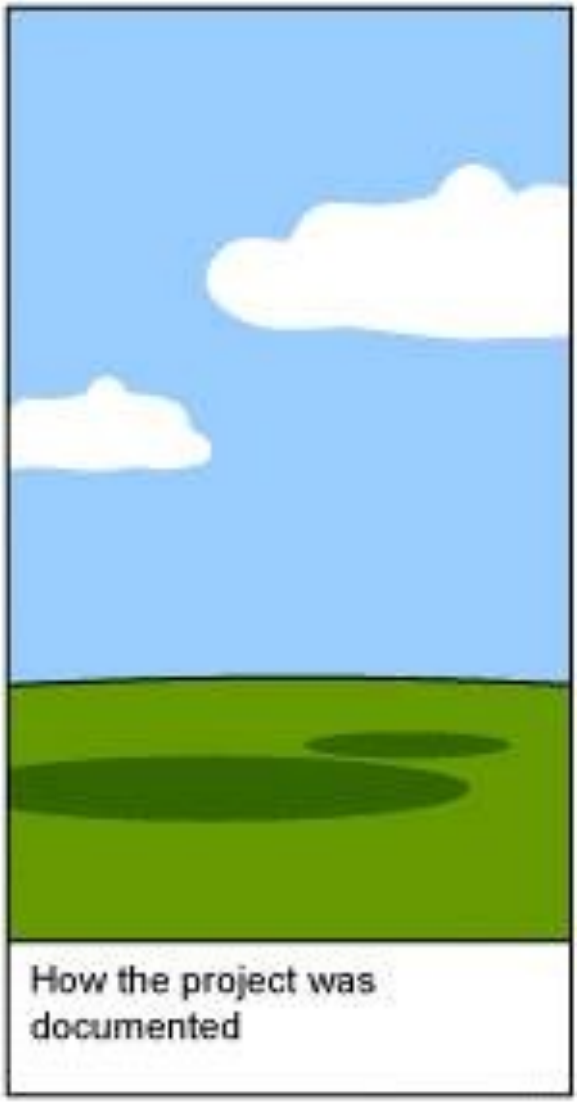
How the Programmer Wrote It

Source unknown. See http://www.businessballs.com/treeswing.htm .

How the Business Consultant Described It

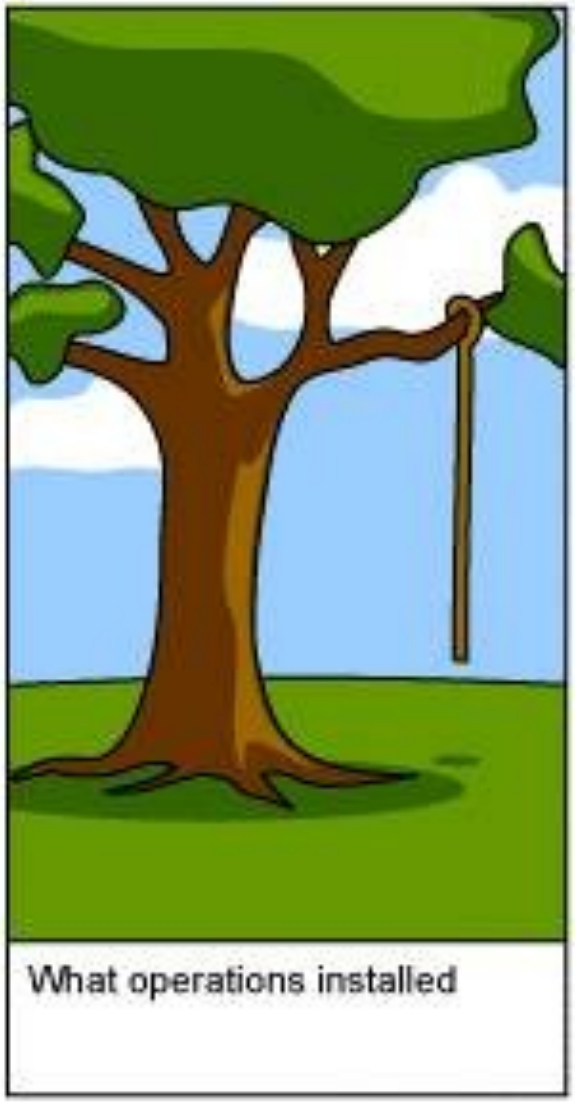
Source unknown. See http://www.businessballs.com/treeswing.htm .

How the Project was Documented



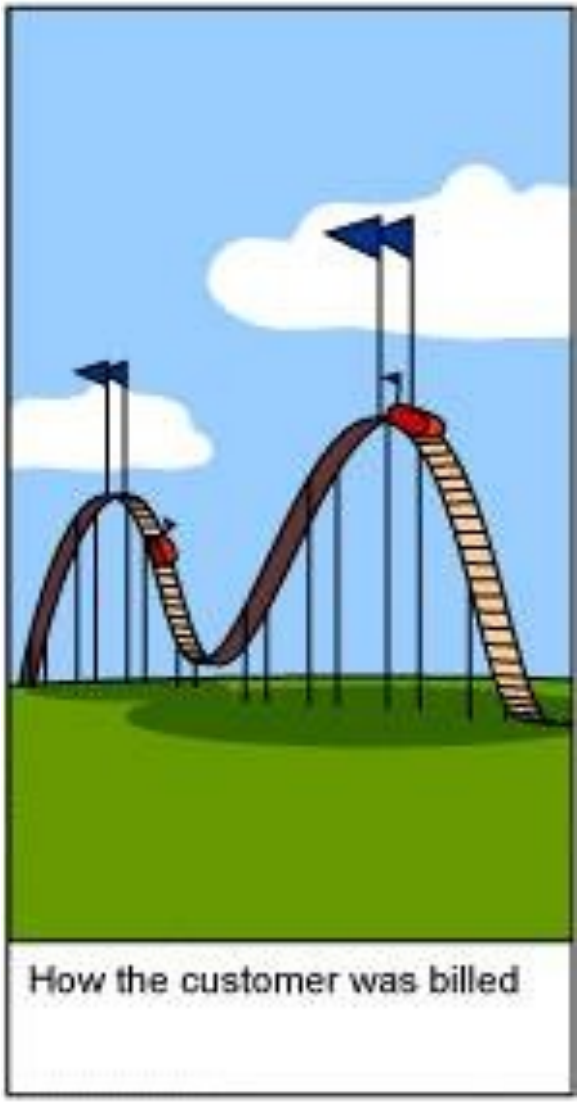
Source unknown. See http://www.businessballs.com/treeswing.htm .

What Operations Installed



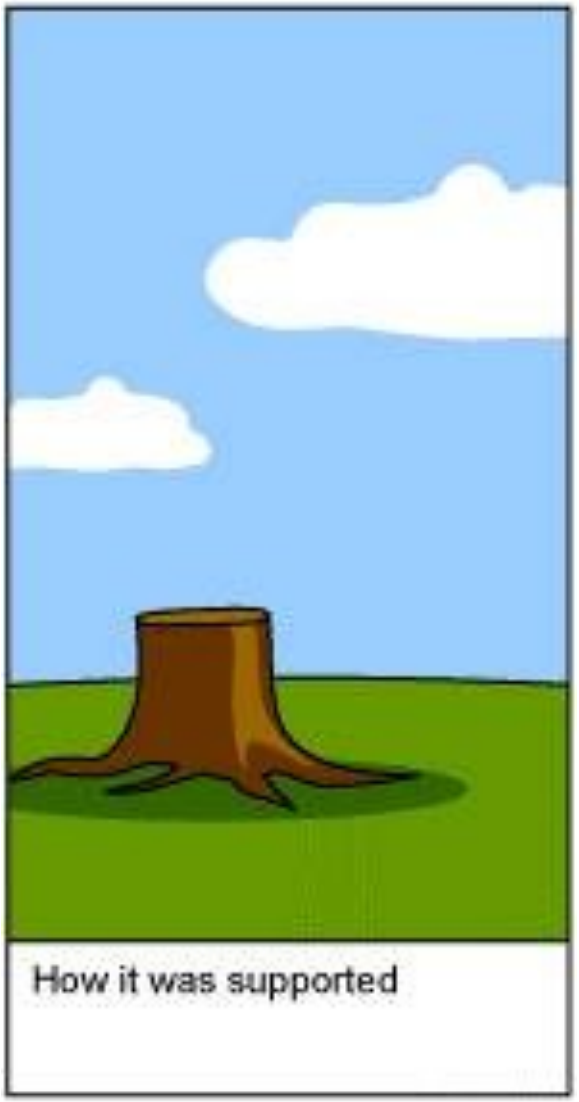
Source unknown. See http://www.businessballs.com/treeswing.htm .

How the Customer was Billed



Source unknown. See http://www.businessballs.com/treeswing.htm .

How It was Supported



Source unknown. See http://www.businessballs.com/treeswing.htm .

What the Customer Really Needed

Source unknown. See http://www.businessballs.com/treeswing.htm .

Iterative Processes

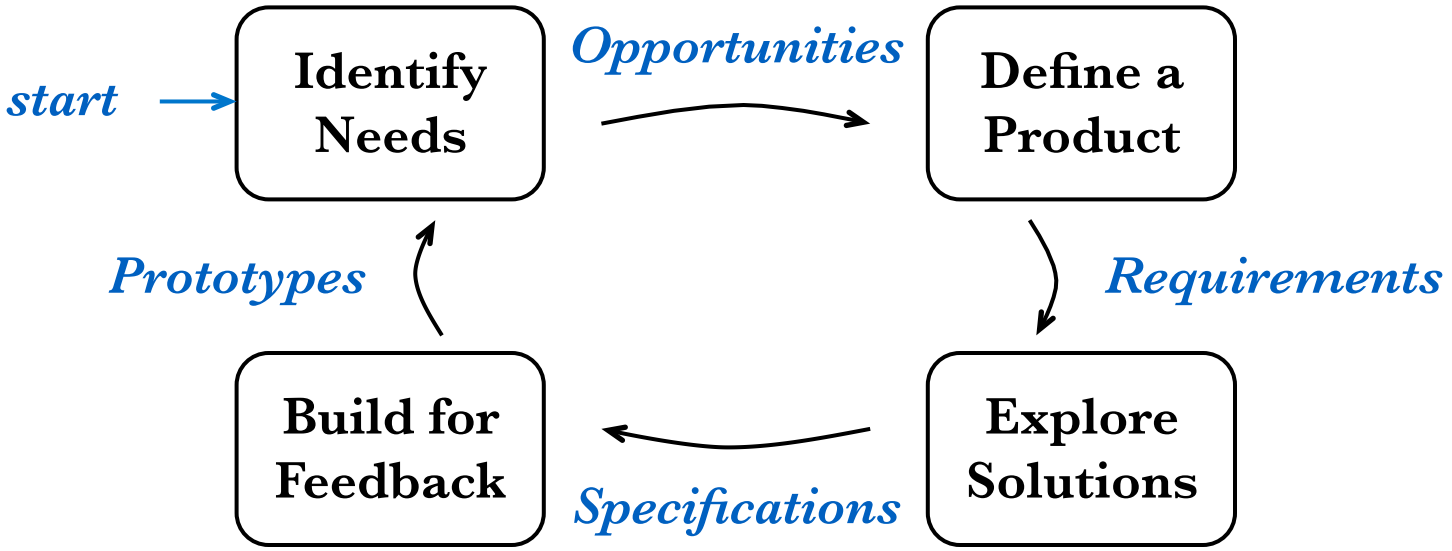
"Design and build software, even operating systems, to be tried early, ideally within weeks. Don't hesitate to throw away the clumsy parts and rebuild them."

– Doug McIlroy, Elliot Pinson, Berkley Tague from their 1978 intro to Unix

Source: McIlroy, Pinson, Tague [1978]

An Iterative Process

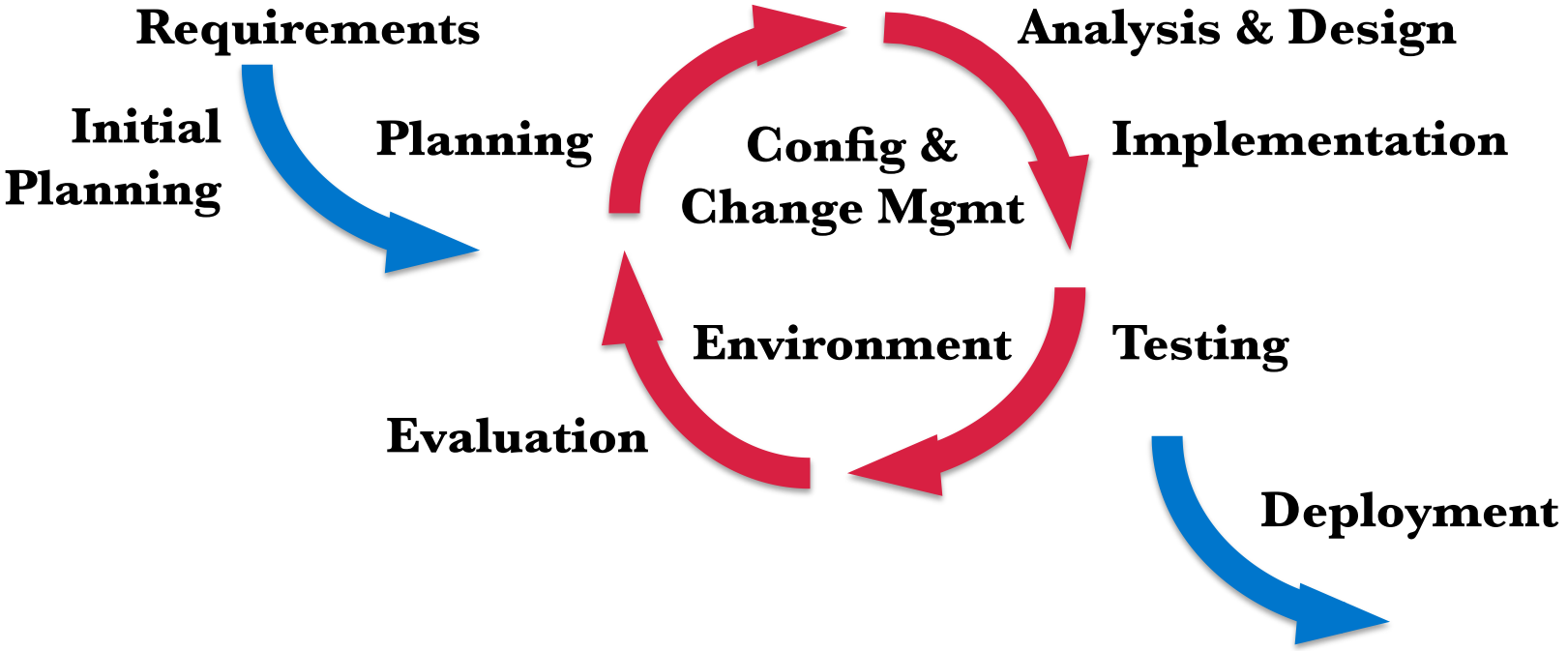
This process emphasizes "What to Build"



- **Overview of an iteration**
  - Identify Customer Needs: output is a list of opportunities
  - Define a Product: requirements for addressing some unmet needs
  - Explore Solutions: brainstorm to come up with design specs
  - Build for Feedback: prototypes that get better with each iteration

Iterative and Incremental Software Processes

Another view of iterative processes



An iteration is a distinct sequence of activities based on an evaluation plan and evaluation criteria, resulting in an executable release (internal or external)

Iterative Processes

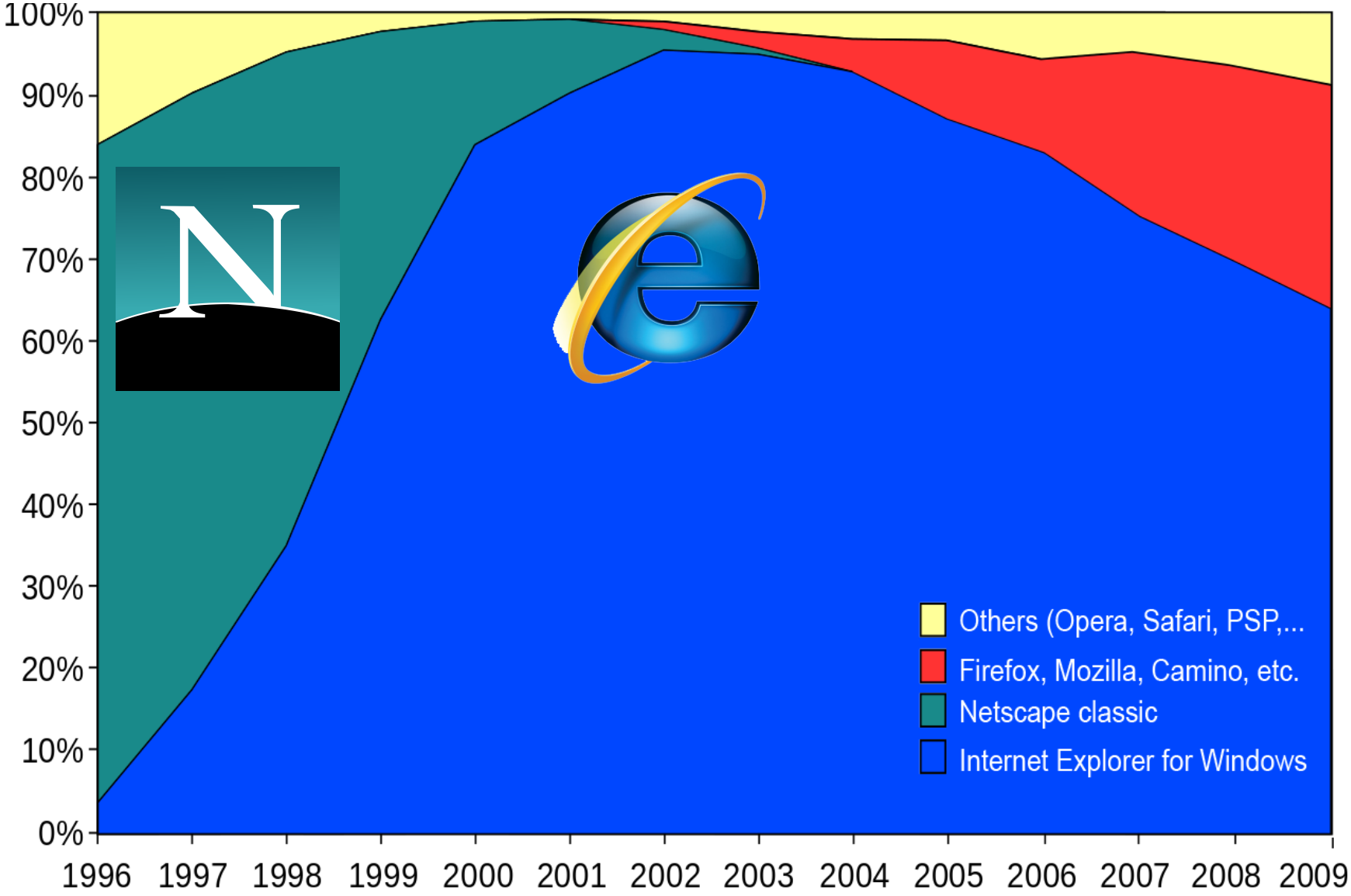
- **Definition**
  - Sequence of steps or iterations that produce increasingly functional versions of a system
  - Customer feedback and course corrections with each iteration
  - With each iteration, the system does more of what customers want
  - Iteration lengths have been getting shorter
- **Benefits of Iterative Processes**
  - Handle uncertain or dynamically changing requirements
  - Improve design and quality as early users uncover issues
  - Enable parallel development of software, hardware, training, …

Case Study: Iterative Processes

Process for Netscape Navigator 3.0 Browser

Netscape Navigator

Dominant browser in the early days of the Internet



Wereon https://en.wikipedia.org/wiki/Browser_wars#/media/File:Browser_Wars_%28en%29.svg

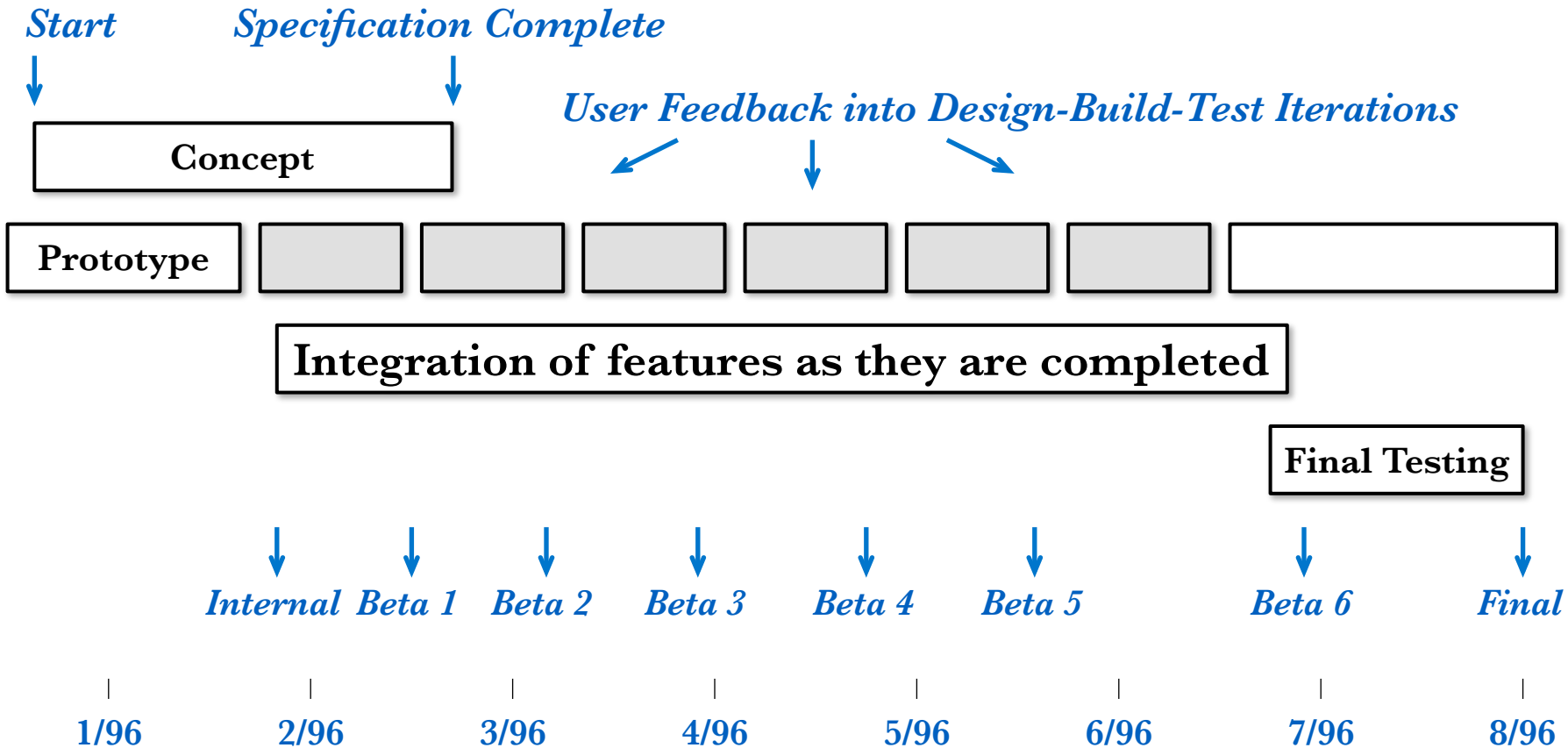
Netscape Navigator 3.0: Quick Iterations

Software Development Process for Navigator 3.0

- **Uncertain requirements**
  - Known unknowns – will customers like the features
- **Dynamically changing requirements**
  - Unknown unknowns – what will Microsoft do?
- **Background: Microsoft appeared to have missed the Internet "Tidal Wave"**
  - In December 1995, Microsoft launched an all out effort
  - Compared itself to a sleeping giant that had been awakened

Netscape Navigator 3.0: Iterative Process

External Beta Releases roughly monthly



Space Shuttle Software

An early application of Iterative development

Space Shuttle: First Flight April 12, 1981

Why an Iterative Process?

- **Sequential would be too slow**
  - Develop orbiter hardware
  - Then develop software for navigation and flight control of orbiter
  - Then develop simulator and training
- **Parallel development**
  - IBM delivered the software iteratively
  - "The first drop for each release represented a basic set of operational capabilities and provided a structure for adding other capabilities on later drops."

First Shuttle Launch

Madden and Rone [1984]

Space Shuttle Software

Overall, an early iterative development success story

- **Software Process**
  - 17 iterations over 31 months
  - Each iteration: about 8 weeks
- **Changing Requirements**
  - NASA and its contractors made over 2,000 requirements changes between 1975 and the first flight in 1981
- **Cost of changing requirements**
  - About $200 M spent on software
  - Initial estimate: $20 M

First Shuttle Launch

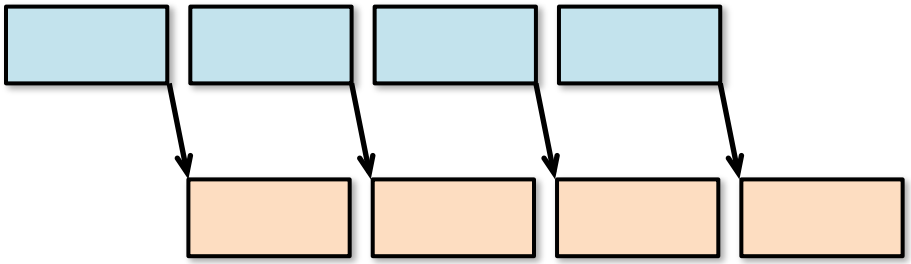
Source: http://history.nasa.gov/computers/Ch4-5.html

One of the Benefits of Iterative Processes

Parallel development of subsystems

- **Sequential Process: Software, then Simulators and Training**

- **Iterative Process enables Parallel Development**



Enabling Practices for Iterative Processes

Do the lessons carry over to Agile Processes?

- **Study of 29 software projects from 17 companies**
  - Conducted between 1996 and 1998
- **Measured product quality:**
  - Combination of performance, functionality, reliability
- **4 practices correlated with successful projects**
  - Early customer feedback to evolve functionality and design
  - Daily builds and automated tests to integrate new code
  - Major investments in a flexible loosely-couple architecture
  - Experienced team, with experience on diverse projects

MacCormack [2001]