SOFTWARE ENGINEERING

Introduction to Software Engineering

Course ID 06016410, 06016321

Nont Kanungsukkasem, B.Eng., M.Sc., Ph.D.
nont@it.kmitl.ac.th

Channels for Communication

Google Classroom:

https://classroom.google.com/c/NzE1OTM5MTYyNzc1?cjc=vy5znce

Class code

vy5znce

Facebook Group:

https://www.facebook.com/groups/547387538050695

Course Description

- ความรู้เบื้องต้นเกี่ยวกับวิศวกรรมซอฟต์แวร์กรอบงานของกระบวนการวิศวกรรมซอฟต์แวร์หลักปฏิบัติทางวิศวกรรมซอฟต์แวร์ กระบวนการพัฒนาซอฟต์แวร์แบบอไจล์ การประชุมของอไจล์ การวิเคราะห์และรวบรวมความต้องการแบบอไจล์ การสร้างแบบจำลองและจัดลำดับความสำคัญความต้องการ กระบวนการจัดการโครงการพัฒนาซอฟต์แวร์แบบสกรัมอย่างสมบูรณ์ พื้นฐานกระบวนการจัดการโครงการพัฒนาซอฟต์แวร์แบบลีนและคานบาน การออกแบบเชิงวัตถุขั้นสูง (ดีไซน์แพตเทิร์น) ความรู้เบื้องต้นเกี่ยวกับการทดสอบซอฟต์แวร์ พื้นฐานการพัฒนาโดยใช้การทดสอบเป็นตัวขับเคลื่อน แบบจำลองกระบวนการสำหรับการพัฒนาซอฟต์แวร์
- Introduction to Software Engineering, Software Engineering Process Framework, Software Engineering Practice, Agile Software Development, Agile Ceremonies, Agile Requirement Analysis and Gathering, Requirement Modeling and Prioritization, Complete Scrum, Basic Lean and Kanban, Advanced OO Design (Design Pattern), Fundamental Software Testing, Basic Test-Driven Development, Process Model for Software Development

SE Course Syllabus (Mid-Term)

Week	Date	หัวข้อ/หัวข้อย่อย	อาจารย์ผู้สอน
1	27, 28 Nov	Introduction to Software Engineering	ผศ.ดร. นนท์ คนึงสุขเกษม
2	4, 12 Dec	Software Process (Predictive and Adaptive)	
3	11, 19 Dec	Agile, SCRUM	
4	18, 26 Dec	SCRUM, XP, Lean, Kanban, DevOps	
5	25 Dec, 2 Jan	Requirement Engineering (Clip)	
6	1, 2 Jan	Requirement Modeling (Clip)	
7	8, 9 Jan	Software Design, Architectural Design	
8	15, 16 Jan	Component-level Design, Design patterns (Creational)	
Mid-Term Exam, TBD			

SE Course Syllabus (Final)

Week	Date	หัวข้อ/หัวข้อย่อย	อาจารย์ผู้สอน
9	29, 30 Jan	Design patterns (Structural)	ผศ.ดร. นนท์ คนึงสุขเกษม
10	5, 6 Feb	Design patterns (Behavioral), Implementation	
11		TBD	ผศ.ดร. สมเกียรติ วังศิริพิทักษ์
12		TBD	
13		TBD	
14		TBD	
15		TBD	
Final Exam, 19 March 2025, 9.30 am -12.30 pm			

SE Assessment Criteria

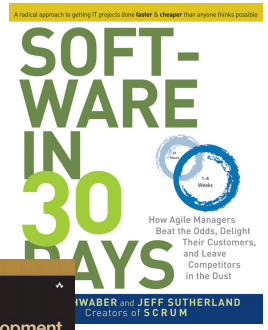
- Assessment
  - (1) การเข้าเรียน และกิจกรรมในห้องเรียน 15%
  - (2) สอบกลางภาค / Mid-term (%) 40%
  - (3) สอบปลายภาค / Final (%) 45%

Software Engineering Textbooks (1)

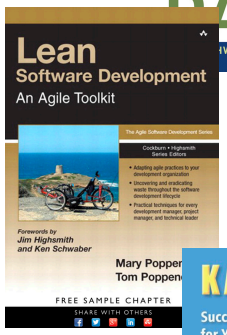
- Software Engineering: A Practitioner's Approach 9th Edition, Roger Pressman, Bruce Maxim, 2020, McGraw-Hill Education
- Software Engineering: Modern Approaches 2nd Edition, Eric J. Braude and Michael E. Bernstein, 2011, Waveland Press, Inc.
- Software Engineering (10th Edition), Ian Sommerville 2016, Pearson
- Design Pattern: Elements of Reusable Object-Oriented Software, Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, 1994, Addison-Wesley Professional Computing Series
- Head First Design Pattern, Eric Freeman, Elisabeth Freeman, Kathy Sierra, Bert Bates, 2004, O'Reilly

Software Engineering Textbooks (2)

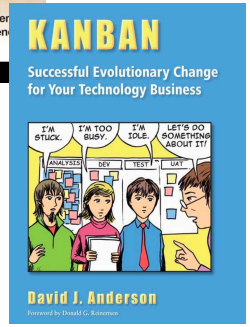
- "The scrum guide." The definitive guide to scrum: The rules of the game, Sutherland, Jeff, and Ken Schwaber, 2020, Scrum.org, https://www.scrum.org/resources/scrum-guide



- Software in 30 Days: How Agile Managers Beat the Odds, Delight, Ken Schwaber, 2012, Jeff Sutherland, John Wiley & Sons



- Lean software development: an agile toolkit, Poppendieck, Mary, and Tom Poppendieck, 2003, Addison-Wesley



- Kanban: successful evolutionary change for your technology business, Anderson, David J., 2010, Blue Hole Press https://resources.kanban.university/wp-content/uploads/2021/06/The-Official-Kanban-Guide_A4.pdf
- Some research materials in leading software engineering conferences and journals

INTRODUCTION TO SOFTWARE ENGINEERING

What is Software?

Software is:

(1) instructions (computer programs) that when executed provide desired features, function, and performance;

(2) data structures that enable the programs to adequately manipulate information and

(3) descriptive information (documentation) in both hard copy and virtual forms that describes the operation and use of the programs

Software: Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market.

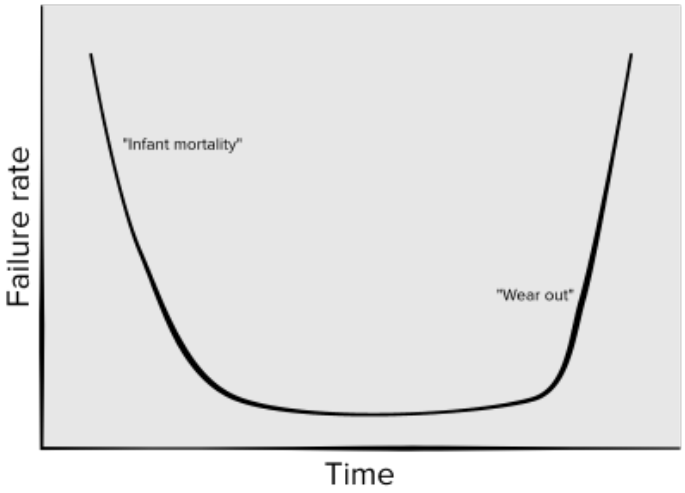
What is Software?

- *Software is a **logical** system element*
- *Software doesn't "**wear out**."*

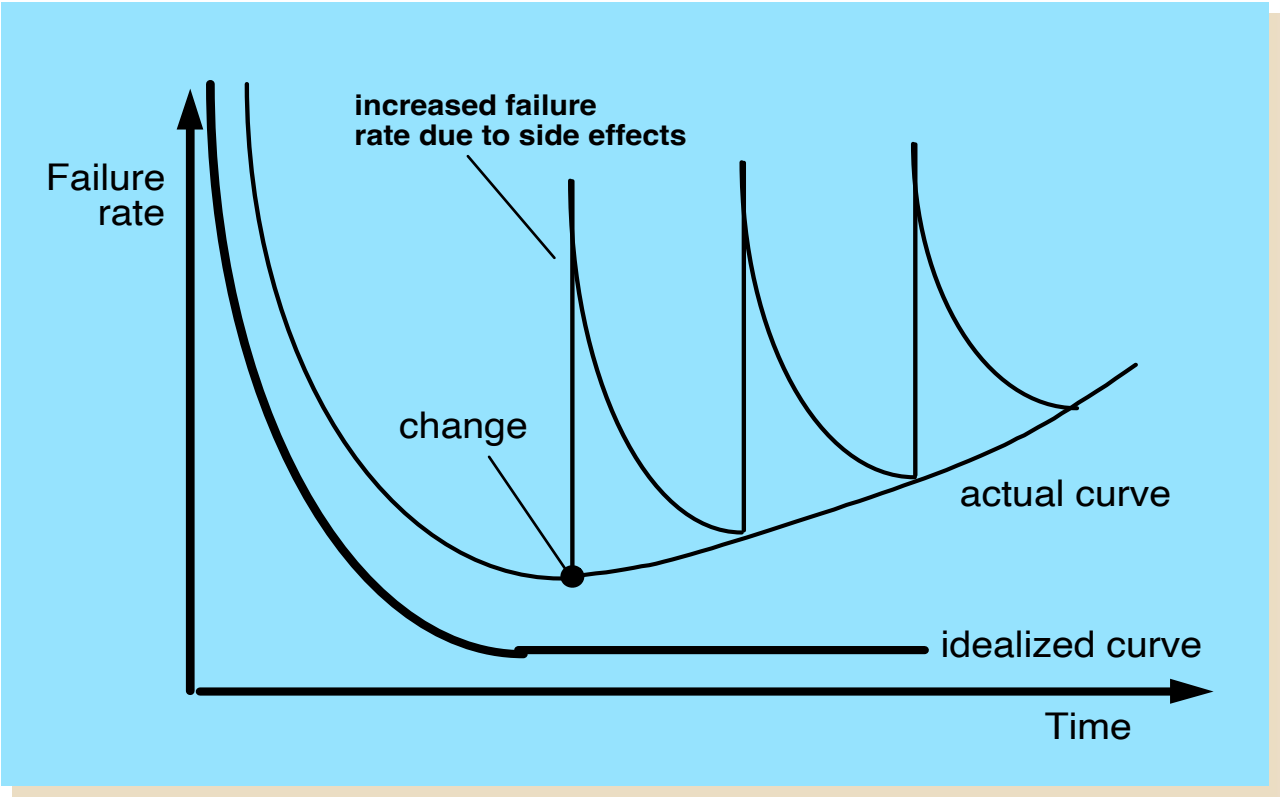
- *Software is an information transformer—*

  producing, managing, acquiring, modifying, displaying, or transmitting information that can be as simple as a single bit or as complex as an augmented-reality representation derived from data acquired from dozens of independent sources and then overlaid on the real world.

Wear Out vs. Deterioration



Failure curves for hardware



Idealized and actual failure curves for software

Essential Attributes

- *Acceptability*
- *Dependability and security*
- *Efficiency*
- *Maintainability*

Software Application Domains

- System software
- Application software
- Engineering/scientific software
- Embedded software
- Product-line software
- Web/Mobile applications
- AI software (robotics, neural networks, game playing)

Software Application Types

- Stand-alone applications
- Interactive transaction-based applications
- Embedded control systems
- Batch processing systems
- Entertainment systems
- Systems for modeling and simulation
- Data collection and analysis systems
- Systems of systems

Legacy Software

*Legacy systems often **evolve** for one or more of the following reasons:*

- software must be adapted to meet the needs of new computing environments or technology.
- software must be enhanced to implement new business requirements.
- software must be extended to make it interoperable with other more modern systems or databases.
- software must be re-architected to make it viable within an evolving computing environment.

WebApps

- Modern WebApps are much more than hypertext files with a few pictures
- WebApps are augmented with tools like XML and Java to allow Web engineers including interactive computing capability
- WebApps may standalone capability to end users or may be integrated with corporate databases and business applications
- Semantic web technologies (Web 3.0) have evolved into sophisticated corporate and consumer applications that encompass semantic databases that require web linking, flexible data representation, and application programmer interfaces (API's) for access
- The aesthetic nature of the content remains an important determinant of the quality of a WebApp.

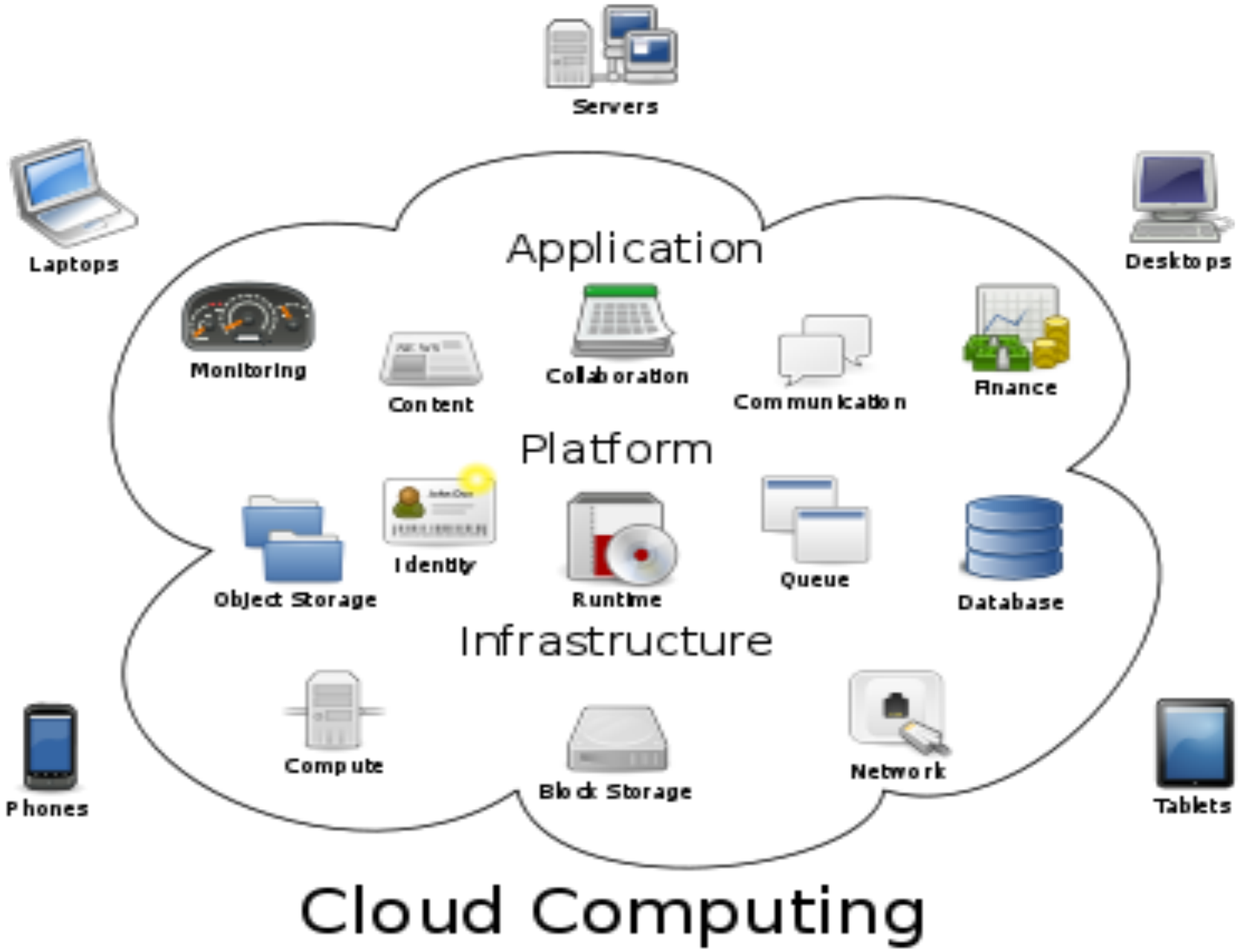
Characteristics of WebApps

- **Data driven.** The primary function of many WebApps is to use hypermedia to present text, graphics, audio, and video content to the end-user.
- **Content sensitive.** The quality and aesthetic nature of content remains an important determinant of the quality of a WebApp.
- **Continuous evolution.** Unlike conventional application software that evolves over a series of planned, chronologically-spaced releases, Web applications evolve continuously.
- **Immediacy.** Although *immediacy*—the compelling need to get software to market quickly—is a characteristic of many application domains, WebApps often exhibit a time to market that can be a matter of a few days or weeks.
- **Security.** Because WebApps are available via network access, it is difficult, if not impossible, to limit the population of end-users who may access the application.
- **Aesthetics.** An undeniable part of the appeal of a WebApp is its look and feel.

Mobile Apps

- Reside on mobile platforms such as cell phones or tablets
- Contain user interfaces that take both device characteristics and location attributes
- Often provide access to a combination of web-based resources and local device processing and storage capabilities
- Provide persistent storage capabilities within the platform
- A *mobile web application* allows a mobile device to access to web-based content using a browser designed to accommodate the strengths and weaknesses of the mobile platform
- A *mobile app* can gain direct access to the hardware found on the device to provide local processing and storage capabilities
- As time passes these differences will become blurred

Cloud Computing



Cloud Computing

- *Cloud computing* provides distributed data storage and processing resources to networked computing devices
- Computing resources reside outside the cloud and have access to a variety of resources inside the cloud
- Cloud computing requires developing an architecture containing both frontend and backend services
- Frontend services include the client devices and application software to allow access
- Backend services include servers, data storage, and server-resident applications
- Cloud architectures can be segmented to restrict access to private data

THE CONTEXT OF SOFTWARE ENGINEERING

Definition of "Engineering"

The *profession* in which

a knowledge of the *mathematical* and *natural sciences* gained by study, experience, and practice is *applied with judgment* to develop ways to *utilize*, economically, the *materials and forces of nature for the benefit of mankind*

-- Accreditation Board for Engineering and Technology, 1996

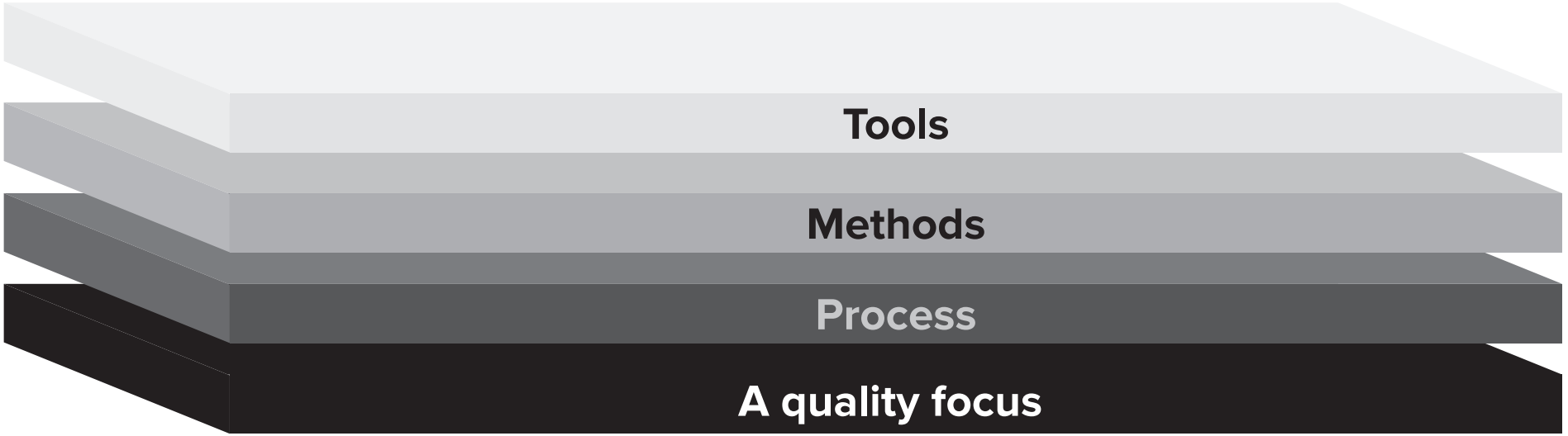
What is Software Engineering?

- The IEEE has developed the following definition for software engineering:
  - Software Engineering: The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.

What is Software Engineering?

- Software engineering is
  - an engineering **discipline** that is concerned with all aspects of *software production*
  - from the early stages of system specification through to maintaining the system after it has gone into use.

Software Engineering Layers



Why?

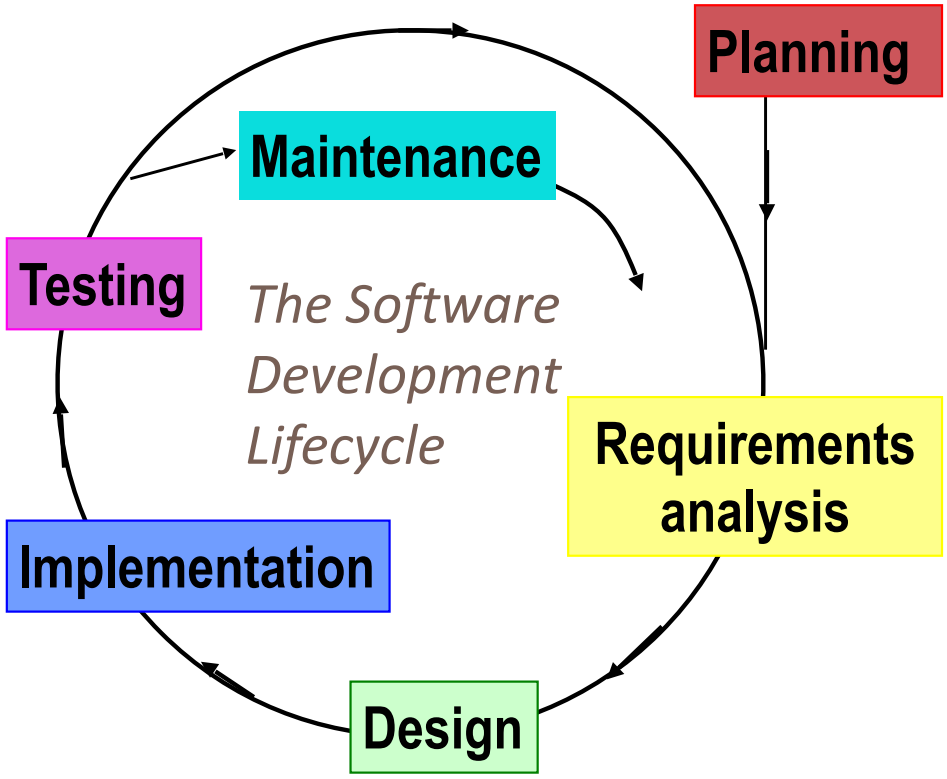
- Why does it take so long to get software finished?
- Why are development costs so high?
- Why can't we find all errors before we give the software to our customers?
- Why do we spend so much time and effort maintaining existing programs?
- Why do we continue to have difficulty in measuring progress as software is being developed and maintained?

THE ACTIVITIES OF SOFTWARE ENGINEERING

Basic Activities of Software Engineering

- ***defining*** **the software development *process* to be used**
- ***managing*** **the development *project***
- ***describing*** **the intended software *product***
- ***designing*** **the *product***
- ***implementing*** **the *product***
- ***testing*** **the *parts* of the product**
- ***integrating*** **the parts and testing them as a whole**
- ***maintaining*** **the *product***

Software Development Life Cycle (SDLC)

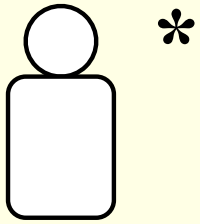


Goals of SWE

- To create SW that meet the *requirements* of users or customers
- To make the created SW *reliable*, *efficient* and *maintainable*
- Additionally, to make the production of SW *on time* and within *planned budget*.

The Four "P's" of Software Engineering

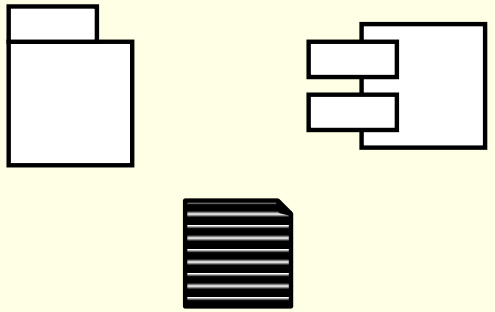
People



(by whom it is done)

Project stakeholders

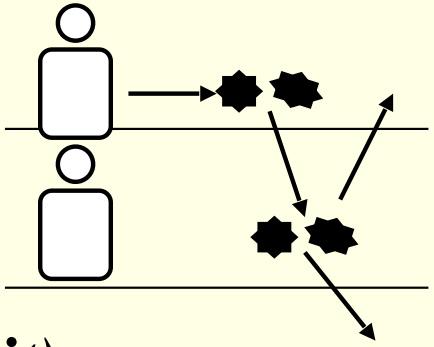
Product



(the application artifacts)

SW product + associated documents

Project



(the doing of it)

Activities carried out to build the product

Process

(the manner in which it is done)

Framework within which the team carries out the activities to build the product

People

- Most important resources in software project
- Competent people must be recruited, trained and motivated for project succeed.
- Not limited to a group of software engineers and developers.
- These people include all *stakeholders* associated with the project.

People - Stakeholders

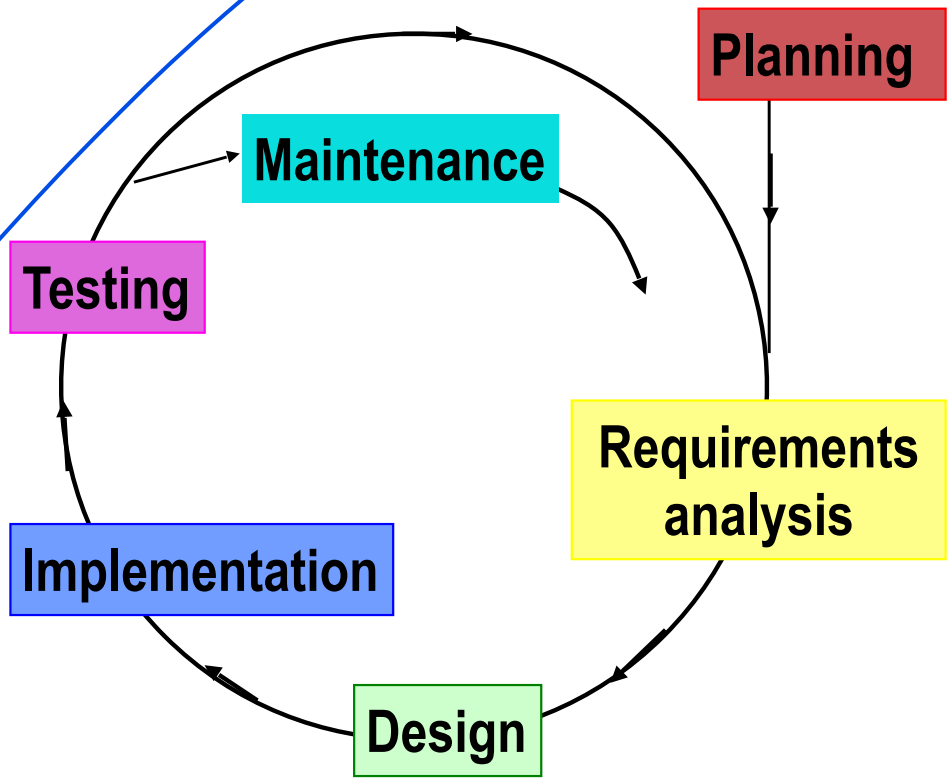
1. Senior managers (product owners) who define the business issues that often have a significant influence on the project.
2. Project (technical) managers (Scrum masters or team leads) who must plan, motivate, organize, and coordinate the practitioners who do software work.
3. Practitioners who deliver the technical skills that are necessary to engineer a product or application.
4. Customers who specify the requirements for the software to be engineered and other stakeholders who have a peripheral interest in the outcome.
5. End users who interact with the software once it is released for production use.

Product

- **Project documentation**
  Documents produced during software definition and development.
- **Code**
  Source and object.
- **Test documents**
  Plans, cases, and results.
- **Customer documents**
  Documents explaining how to use and operate product.
- **Productivity measurements**
  Analyze project productivity.

Project

- A software project defines the activities and associated results needed to produce a software product.

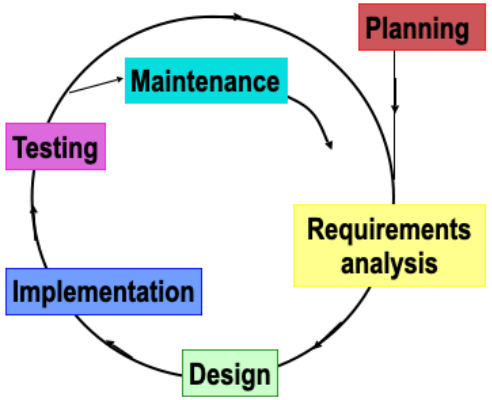


Project

- Other activities
  - Development paradigms
    - object-oriented paradigm
  - Techniques
  - Tools

Process

Framework for carrying out a set of activities in a project in an organized and disciplined manner.



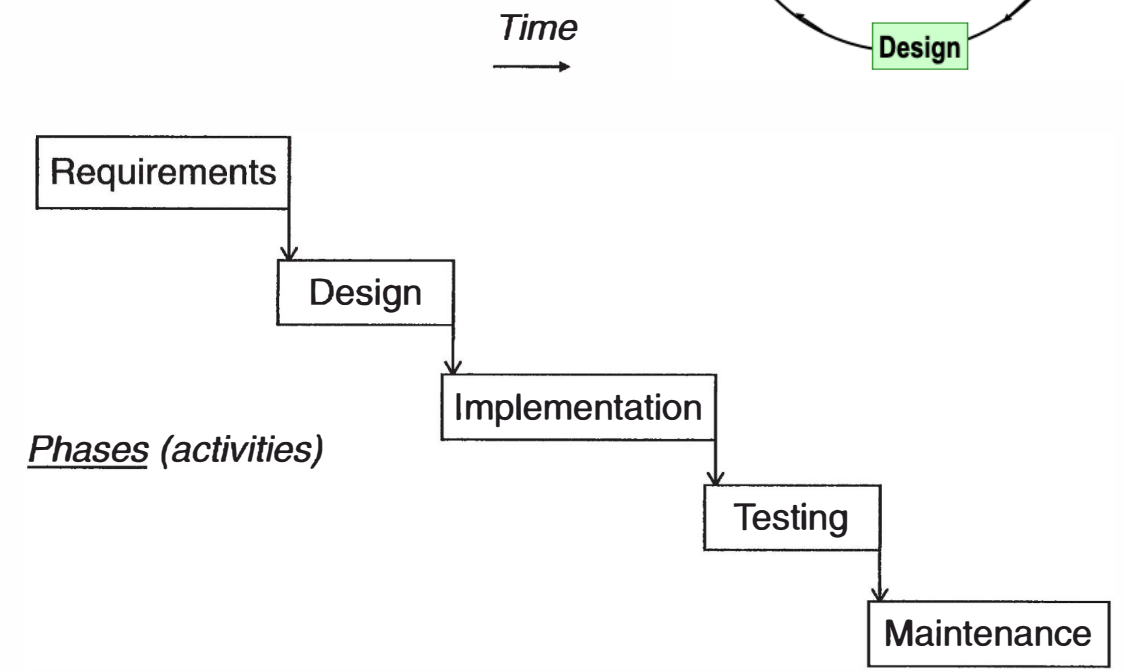
***Development sequence*:**

Predictive

- **Waterfall**
- **Iterative**
- **Spiral**
- **Unified process**

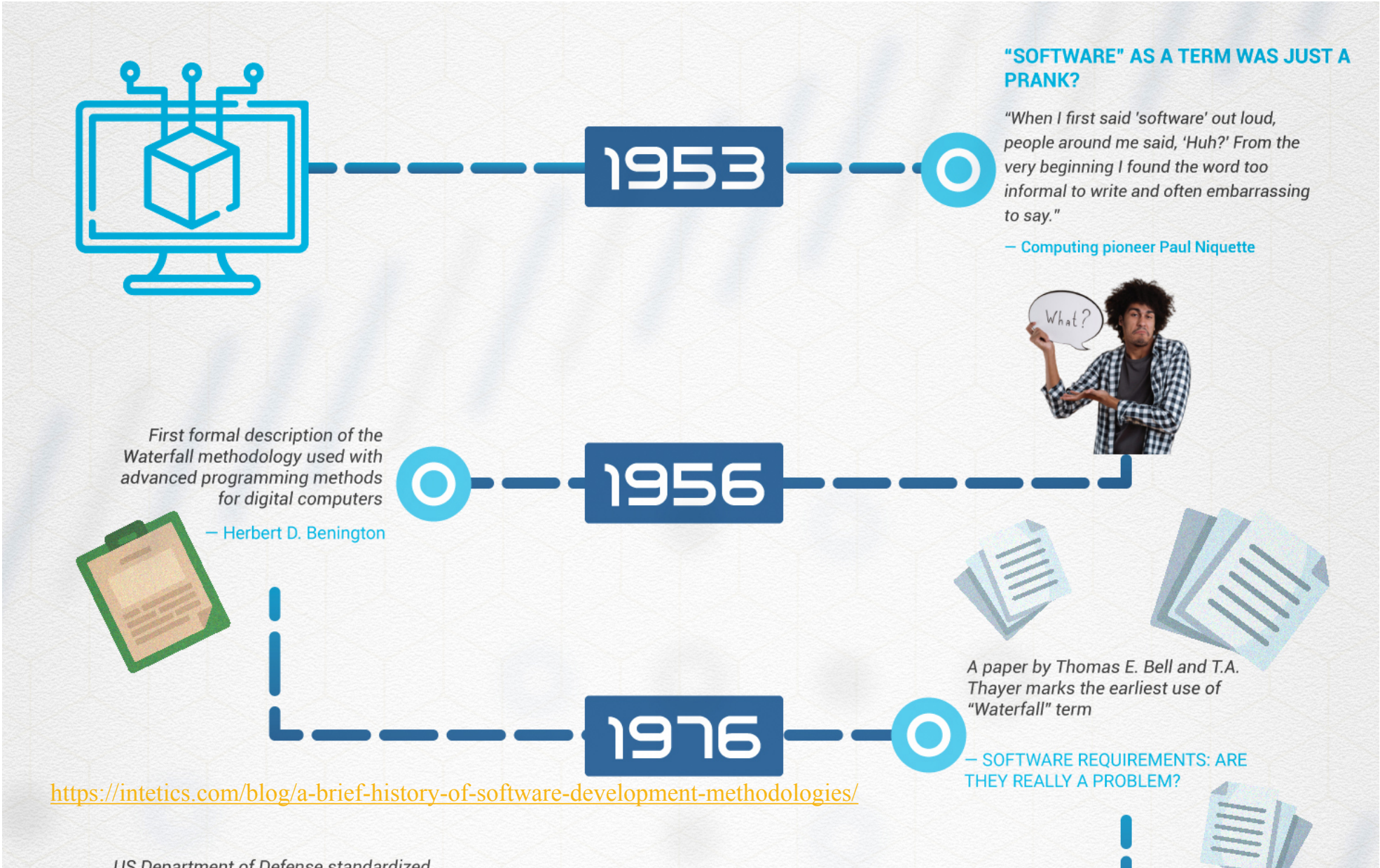
Adaptive

- **Agile !!!** **- This is what we will focus on.**



Process (History)

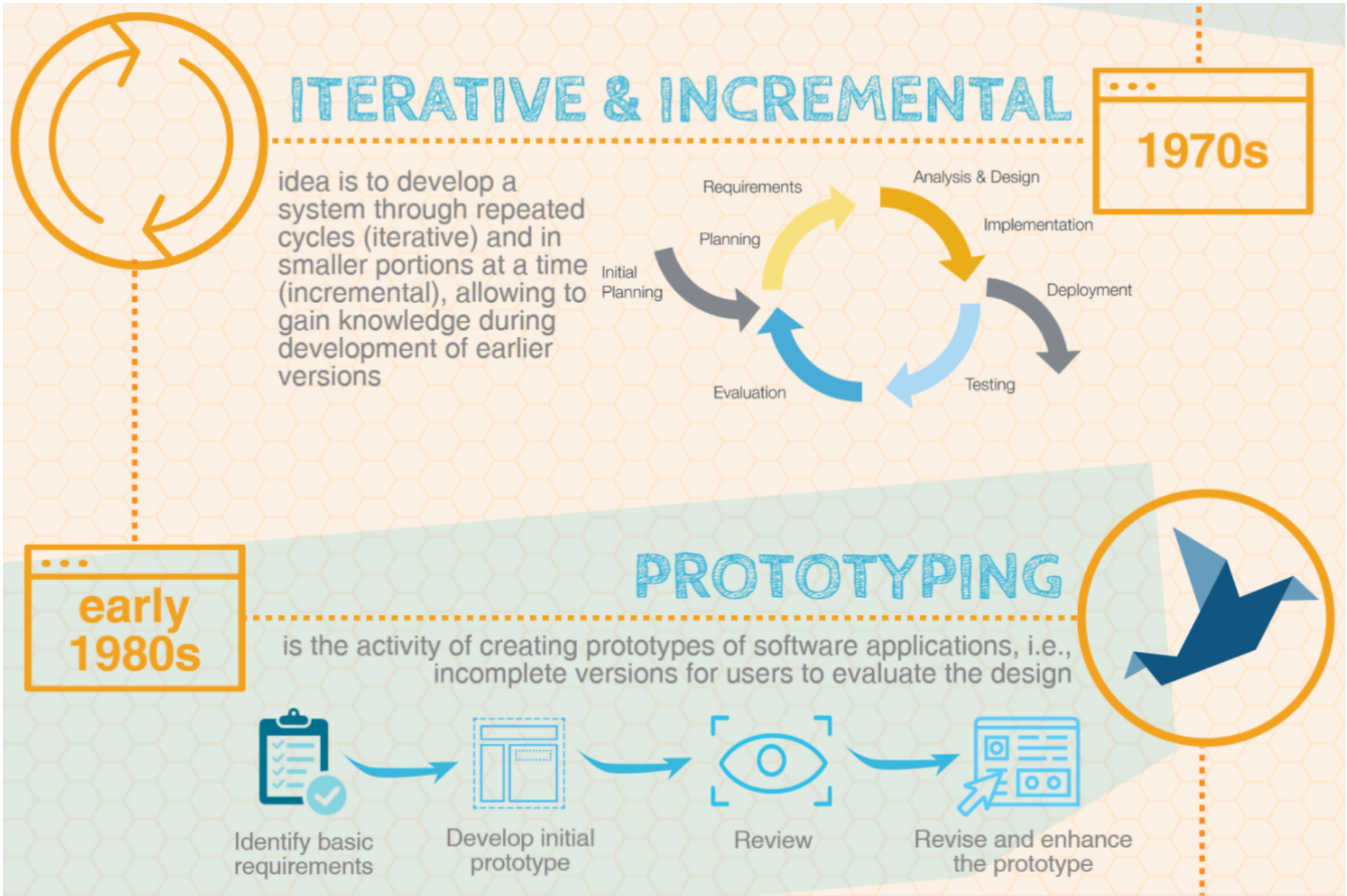
https://www.techopedia.com/infographic-celebrating-6-decades-of-software-development-methodologies/2/34269



https://intetics.com/blog/a-brief-history-of-software-development-methodologies/

Process (History)

https://intetics.com/blog/a-brief-history-of-software-development-methodologies/

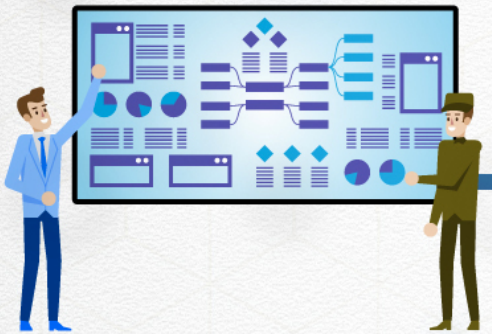


Process (History)

US Department of Defense standardized the Waterfall methodology for its software development providers.

6 standardized stages: Preliminary Design, Detailed Design, Development and Unit Testing, Integration, and Testing.

1985



1986

Spiral model was first mentioned by Barry Boehm

A combination of a waterfall model and iterative model. Each phase in Spiral starts with a design goal and ends with the client progress review.

Fred Brooks published his paper No Silver Bullet—Essence and Accident in Software Engineering

"There is no single development, in either technology or management technique, which by itself promises even one order-of-magnitude improvement within a decade in productivity, in reliability, in simplicity."

ADDITIONALLY, BROOKS' 1975 BOOK: The Mythican Man-Month, becomes the foundational text for software engineering. And in 1999, Brooks won the Turing award for his contributions.

Takeuchi and Nonaka introduced the term Scrum for the first time in: The New New Product Development Game.

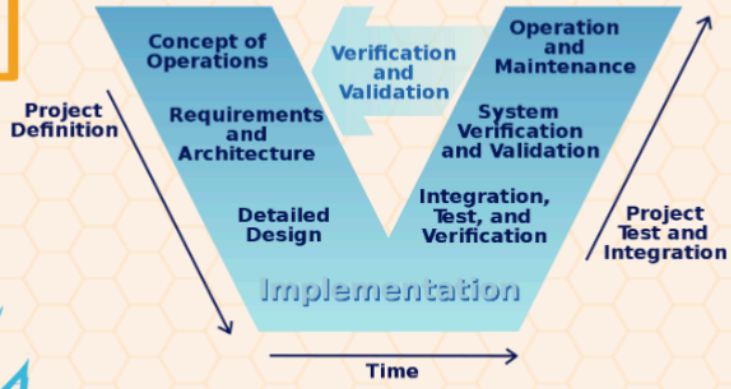
"In today's fast-paced, fiercely competitive world of commercial new product development,

KEN SCHWABER and

Process (History)

late 1980s

V-MODEL



is an extension of the waterfall model.

The process steps are bent upwards after the coding phase, to form the typical V shape

Waterfall era ENDs

PREDICTIVE

ADAPTIVE

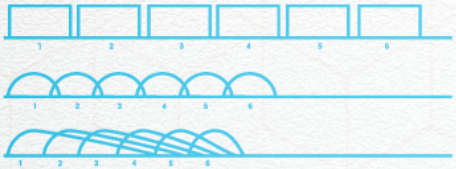
Start of AGILE era

Process (History)

Takeuchi and Nonaka introduced the term Scrum for the first time in: The New New Product Development Game.

"In today's fast-paced, fiercely competitive world of commercial new product development, speed and flexibility are essential."

Takeuchi and Nonaka claim that "Scrum" is an approach to organizational knowledge creation.



The Mythican Man-Month, becomes the foundational text for software engineering. And in 1999, Brooks won the Turing award for his contributions.

1990

KEN SCHWABER and **JEFF SUTHERLAND** use *"Scrum"* at their respective software development companies.

1995

Schawaber and Shuterland publish Scrum Methodology.

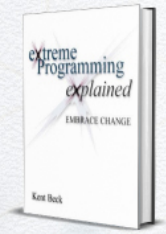
RUP®
Rational Unified Process®

1996-1998

RATIONAL SOFTWARE COMPANY DEVELOPS THE RATIONAL UNIFIED (SOFTWARE) PROCESS.

THE RATIONAL UNIFIED PROCESS BEST PRACTICES: Develop Software Iteratively, Manage Requirements, Use Component-Based Architectures, Visually Model Software, Verify Software Quality, and Control Changes to Software.

Process (History)



KENT BECK PUBLISHES EXTREME PROGRAMMING EXPLAINED, introducing the Agile revolution.

1999

2001

THE AGILE MANIFESTO IS PUBLISHED:

The 4 Agile Values: **Individuals and interactions** over processes and tools, **Working software** over comprehensive documentation, **Customer collaboration** over contract negotiation, **Responding to change** over following a plan.

TOM AND MARY POPPENDIECK PUBLISH THEIR BOOK LEAN SOFTWARE DEVELOPMENT.

Mary and Tom Poppendieck identify seven fundamental *"lean"* principles that serve as the foundation for agile development approaches that work.

THE 7 FUNDAMENTALS:
Eliminate Waste, Build Quality In, Create Knowledge, Defer Commitment, Deliver Fast, Respect People, Optimize the Whole.

2003

Robert C. Martin (Uncle Bob)

During the Agile conference, Robert "Uncle Bob" Martin engaged new discussions around "professionalism in programming" by advising that the Agile Manifesto be updated to a fifth value: "Craftsmanship over Execution".

Process (History)

that serve as the foundation for agile development approaches that work.

THE 7 FUNDAMENTALS:
Eliminate Waste, Build Quality In, Create Knowledge, Defer Commitment, Deliver Fast, Respect People, Optimize the Whole.

Robert C. Martin (Uncle Bob)
Author of Clean Code and many influential books, Co-author of the Agile Manifesto, **USA**

During the Agile conference, Robert "Uncle Bob" Martin engaged new discussions around "professionalism in programming" by advising that the Agile Manifesto be updated to a fifth value: "Craftsmanship over Execution".

Soon later in 2009, the principles of the movement known as Software Craftmanship are established.

The metaphor of the system professional is replaced, going from Engineer to *"Medieval Artisan"*.

2008

THE TERM DEVOPS BECOMES POPULAR IN A SERIES OF "DEVOPSDAYS".

Patrick Debois watched a live-stream presentation hosted in Belgium and became inspired to start his own conference in Belgium — DevOpsDays.

A forward-thinking group of minds who attended the conference came together in their efforts to improve software deployment.

DEVOPS DAYS

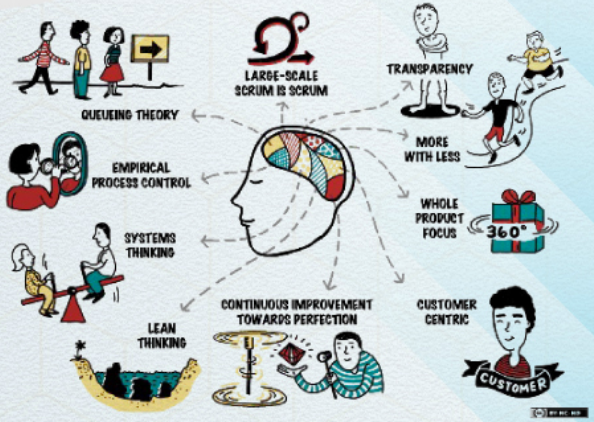
2009

David Anderson published his book Kanban: the most comprehensive definition of the Kanban Method for knowledge work.

2010

Process (History)

TODAY



With tools such as Puppet, Chef and Docker, DevOps is becoming widely accepted and practiced, and the walls continue to crumble between development and operations.

Agile frameworks are facing scalability issues, opening new doors of opportunity to new implementations, such as Less (Large-Scale-Scrum) and Safe (Scaled Agile Framework).

Process

Process framework measurements:

Capability Maturity Model℠

Personal Software Process℠

Team Software Process℠

Standards:

Institute of Electrical and Electronic Engineers

International Standards Organization

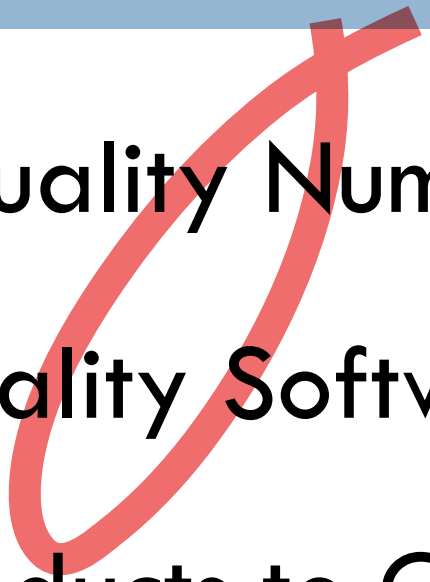
Software Engineering Institute

Assess the SW development capabilities

SOFTWARE ENGINEERING PRINCIPLES

Principles of Software Engineering

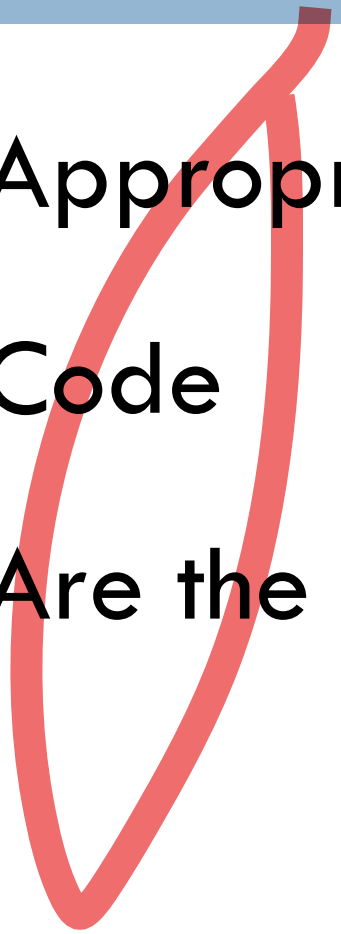
- Make Quality Number 1
- High-Quality Software Is Possible
- Give Products to Customers Early



Davis, Alan, "201 Principles of Software Engineering," 1995, McCraw Hill.

Principles of Software Engineering

- Use an Appropriate Software Process
- Inspect Code
- People Are the Key to Success



Davis, Alan, "201 Principles of Software Engineering," 1995, McCraw Hill.

Seven Principles of Software Development

1. The Reason It All Exists
2. KISS (Keep It Simple, Stupid!)

   All design should be as simple as possible, but no simpler.
3. Maintain the Vision
4. What You Produce, Others Will Consume

http://c2.com/cgi/wiki?SevenPrinciplesOfSoftwareDevelopment

Seven Principles of Software Development

5. Be Open to the Future *Never design yourself into a corner*
6. Plan Ahead for Reuse
7. Think!

http://c2.com/cgi/wiki?SevenPrinciplesOfSoftwareDevelopment

ETHICS

Code of Ethics and Professional Practice

https://ethics.acm.org/code-of-ethics/software-engineering-code/

- Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a beneficial and respected profession.
- In accordance with their commitment to the health, safety and welfare of the public, software engineers shall adhere to the following Eight Principles:

1. Public
2. Client and Employer
3. Product

Code of Ethics and Professional Practice

https://ethics.acm.org/code-of-ethics/software-engineering-code/

4. Judgment

5. Management

6. Profession

7. Colleagues

8. Self