
Software Engineering
Dr. Rasha Montaser

- Email: rashamontaser@gmail.com
- Course Material:
  - http://www.acadox.com/join/MAZVIG
    - Code: MAZVIG
  - **Book:** Software Engineering A practitioner's Approach – Roger S. Pressman

Grading System

- Midterm: 15
- Project: 25
- Oral: 10
- Final: 50

Project

Phase	Deliverables	Deadline	Mark
Phase 1-a	Initial SRS Document (Introduction and Requirements sections of CIE203-2016-Phase1-SRS-Template-v1.0)	8 March	1
Phase 1-b	Final SRS Document	15 March	5
Phase 2-a	Initial SDD Document	22 March	2
Phase 2-b	Final SDD Document	29 March	5
Phase 3-a	Initial Partial Implementation	5 April	2
Phase 3-b	Final Partial Implementation	12 April	5
Phase 3-c	Project Presentation	19 April	5
Total Project Grade			25

Levels of SW Developers

- Coders - Can pretty much figure out it. It'll work, but it won't be pretty.
- Hackers - usually low level folks, skillful, with detailed understanding of some area deeply, often scarily deeply.
- Programmer - Write code and understand algorithms. Often work alone and well.
- Software Engineer - Are the best generalists, can use lots of different systems and languages and get them to talk to each other. Are true and broad professionals, work with people, and communicate well.

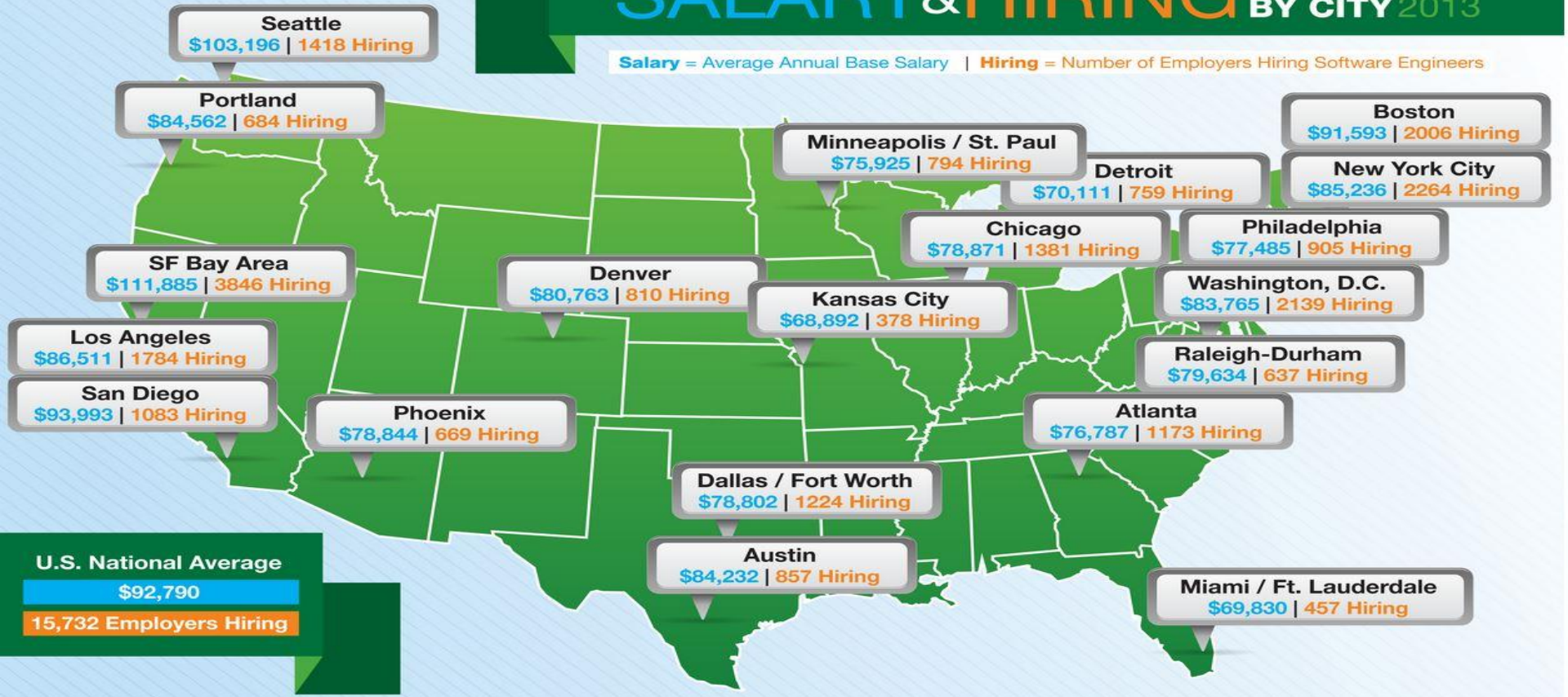
High Demand for Qualified Software Engineers

- Competition for software engineers continues to accelerate, with salaries being an area employers compete aggressively on to win talent.

glassdoor™

SOFTWARE ENGINEER SALARY & HIRING COMPARISON BY CITY 2013

Salary = Average Annual Base Salary | **Hiring** = Number of Employers Hiring Software Engineers



U.S. National Average

$92,790

15,732 Employers Hiring

Average annual base salaries based on at least 150 software engineer salary reports per U.S. metro, while U.S. average annual base salary based on more than 33,000 software engineer salary reports nationally, respectively, between 10/1/12 – 9/30/13. Metro & national hiring data based on number of employers with at least one software engineer job listing posted on Glassdoor as of 10/7/13.

glassdoor

25 HIGHEST PAYING COMPANIES FOR SOFTWARE ENGINEERS 2013

RANK	COMPANY	AVERAGE BASE SALARY
1	Juniper Networks	$159,990
2	LinkedIn	$136,427
3	YAHOO!	$130,312
4	Google	$127,143
5		$124,863

6		$124,630
7	ORACLE	$122,905
8	Walmart	$122,110
9		$121,507
10	integral	$117,927
11	ARISTA	$116,067
12	NVIDIA	$115,649
13	ebay	$114,720

14	amazon	$110,907
15	hp	$110,506
16	BROCADE	$110,069
17	cisco	$109,491
18	Microsoft	$108,611
19	intel	$108,210
20	intuit	$107,440
21	Expedia®	$105,126

Topics

1. Introduction to software engineering
2. Process Models
3. An agile view of process
4. Software engineering practice
5. Requirement engineering
6. Building the analysis model
7. Design engineering
8. Creating an architecture design
9. Performing user interface design
10. Testing strategies
11. Testing tactics
12. Product metrics
13. Project Management
14. Project scheduling

Lec 1
Introduction to Software Engineering

Ref: Chapter 1

Topic

- What is it?
- Who Does it?
- Why it is important?
- What are the steps?
- What is the work product?
- How do I ensure that I have done it right?

What Is Software?

- More than *computer programs*.
- The collection of programs, documentation and configuration data that ensures correct execution.

Characteristic of software

1. *Software is developed or engineered, it is not manufactured in the classical sense.*
2. *Software doesn't "wear out."*
3. *Although the industry is moving toward component-based construction, most software continues to be custom-built.*

Software Applications

- system software
- application software
- engineering/scientific software
- embedded software
- product-line software
- WebApps (Web applications)
- AI software

What Is Engineering?

- **Engineering** is the discipline of applying technical and scientific knowledge and physical resources to design and produce materials, structures, machines, devices, systems, and processes that meet a desired objective under specified criteria.

What Is Software Engineering?

- The process of solving customers' problems by the systematic development and evolution of large, high-quality software systems within cost, time, and other constraints

When Did Software Engineering Start?

- A term used occasionally in 1950s, 1960s
- Popularized in 1968 at NATO Software Engineering Conference
- http://homepages.cs.ncl.ac.uk/brian.randell/NATO/

What Is Software Engineering?

- IEEE Standard 610.12:
  - The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, that is, the application of engineering to software.
- "Designing, building and maintaining large software systems". - I. Sommerville
- "*Multi-person* construction of *multi-version* software". - D. L. Parnas

What Is Software Engineering?

- IEEE Standard 610.12:
  - The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, that is, the application of engineering to software.
- "Designing, building and maintaining large software systems". - I. Sommerville
- "*Multi-person* construction of *multi-version* software". - D. L. Parnas

The Goal of Software Engineering Is Producing Quality Software

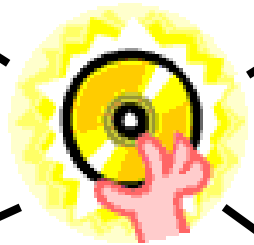
- Can be quite different based on your viewpoint:

Customer:

- Solves problems at acceptable cost (time and resource).

User:

- Easy to learn
- Efficient to use
- Get work done



Developer:

- Easy to design and maintain
- Successfully used and deployed

Developer Manager:

- Sells more and pleases customers
- Costing less to develop and maintain

The Essence of Practice

1. *Understand the problem* (communication and analysis).
2. *Plan a solution* (modeling and software design).
3. *Carry out the plan* (code generation).
4. *Examine the result for accuracy* (testing and quality assurance).

Understand the Problem

- *Who has a stake in the solution to the problem?* That is, who are the stakeholders?
- *What are the unknowns?* What data, functions, and features are required to properly solve the problem?
- *Can the problem be compartmentalized?* Is it possible to represent smaller problems that may be easier to understand?
- *Can the problem be represented graphically?* Can an analysis model be created?

Plan the Solution

- *Have you seen similar problems before?* Are there patterns that are recognizable in a potential solution? Is there existing software that implements the data, functions, and features that are required?
- *Has a similar problem been solved?* If so, are elements of the solution reusable?
- *Can subproblems be defined?* If so, are solutions readily apparent for the subproblems?
- *Can you represent a solution in a manner that leads to effective implementation?* Can a design model be created?

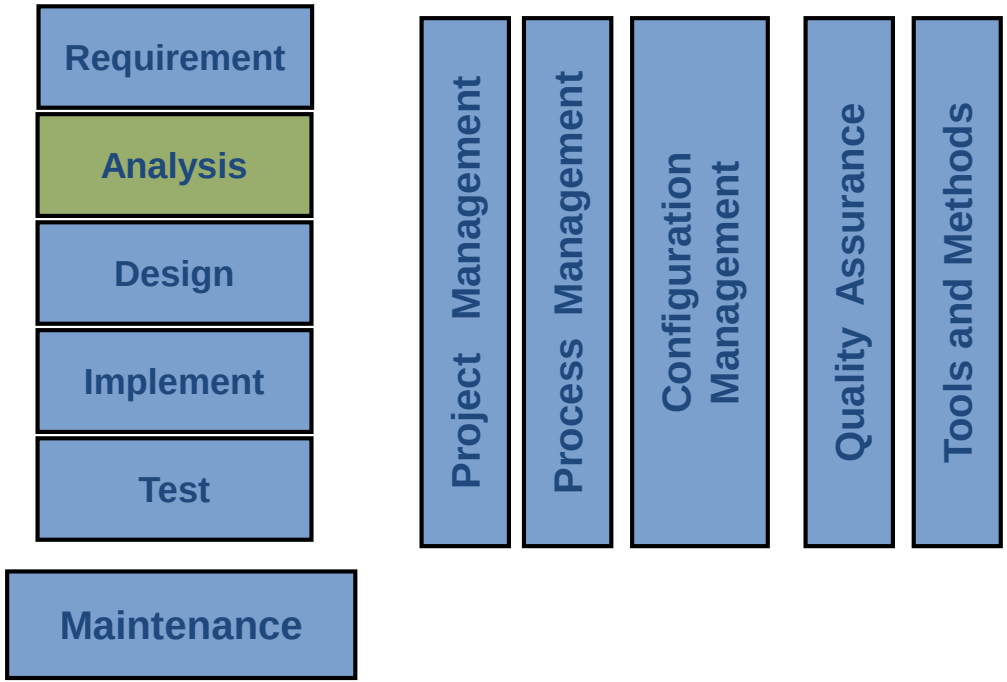
Carry Out the Plan

- *Does the solution conform to the plan?* Is source code traceable to the design model?
- *Is each component part of the solution provably correct?* Has the design and code been reviewed, or better, have correctness proofs been applied to algorithm?

Examine the Result

- *Is it possible to test each component part of the solution?* Has a reasonable testing strategy been implemented?
- *Does the solution produce results that conform to the data, functions, and features that are required?* Has the software been validated against all stakeholder requirements?

Key Stages of Producing SW



SWEBOK's Key 15 Knowledge Areas

REQ	Software Requirements
DES	Software Design
CST	Software Construction
TST	Software Testing
MNT	Software Maintenance
CNF	Software Configuration Management
MGT	Software Engineering Management
PRC	Software Engineering Process
TLS	Software Engineering Tools and Methods
QLY	Software Quality

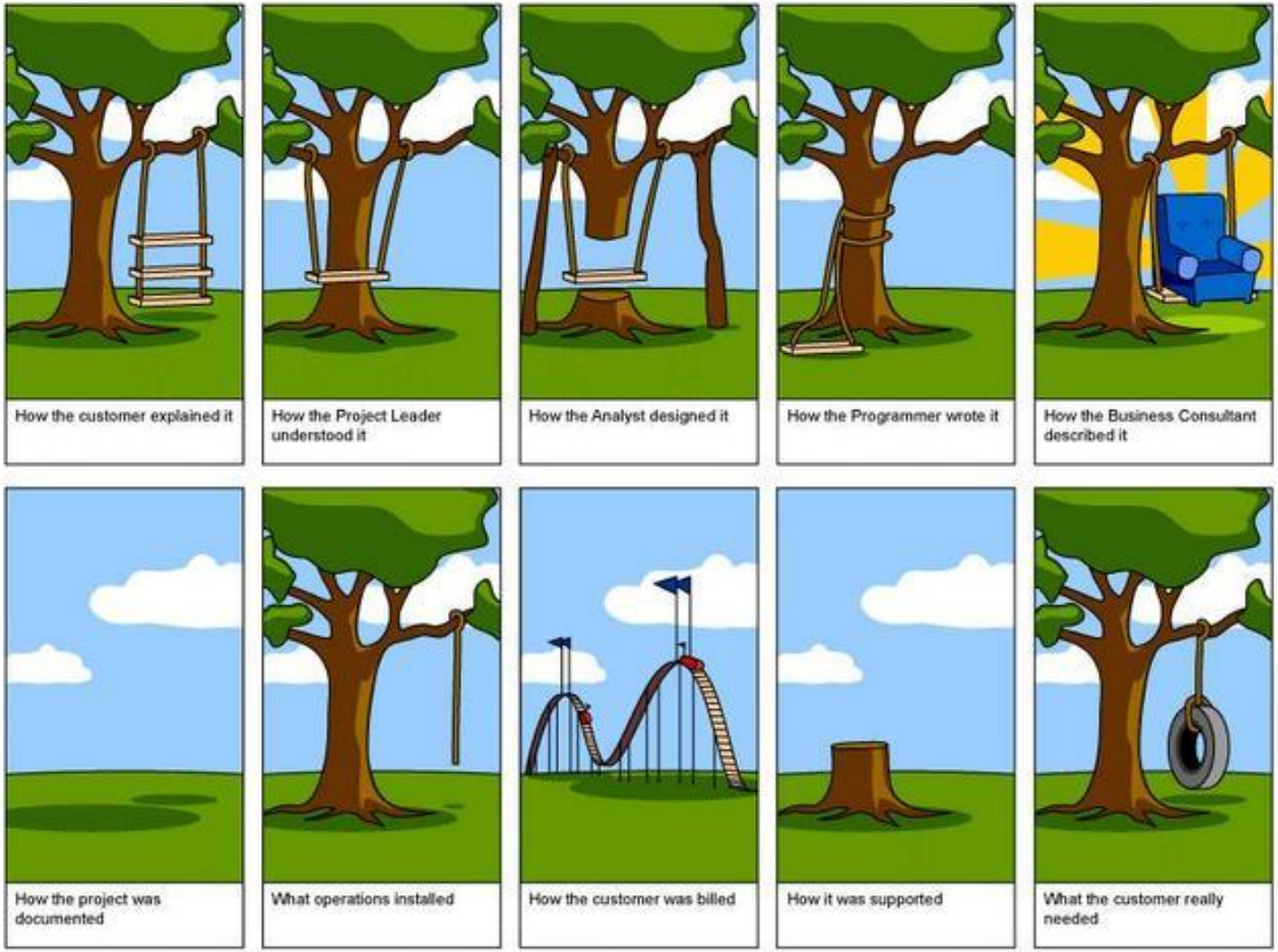
SWEBOK's Key 15 Knowledge Areas

	Software engineering professional practice
	Software engineering economics
	Computing foundations
	Mathematical foundations
	Engineering foundations

http://www.sebokwiki.org/wiki/An_Overview_of_the_SWEBOK_Guide

KA1. Software Requirements

- The Software Requirements Knowledge Area (KA) is concerned with the elicitation, analysis, specification, and validation of software requirements.
- It is widely acknowledged within the software industry that software engineering projects are critically vulnerable when these activities are performed poorly.
- Software requirements express the **needs** and **constraints** placed on a software product that contribute to the solution of some real-world problem.


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

KA2. Software Design

- Design is defined as both “the **process** of defining the architecture, components, interfaces, and other characteristics of a system or component” and “the **result** of [that] process.” Viewed as a process, software
- Software design is the **activity** in which software requirements are analyzed in order to produce a **description** of the software’s internal structure that will serve as the basis for its construction.

KA3. Software Construction

- The term *software construction refers to the detailed* creation of *working*, *meaningful* software through a combination of coding, verification, unit testing, integration testing, and debugging.

K4. Software Testing

- Testing is performed to evaluate and improve product quality by identifying defects and problems.
- Software testing consists of the *dynamic verification of* a program's behavior on a *finite set of test cases*, suitably *selected from the usually infinite executions*, against the *expected behavior*.

KA5. Software Maintenance

- Software development efforts result in the delivery of a software product that satisfies user requirements. Accordingly, the software product must change or evolve.
- Once in operation, defects are uncovered, operating environments change, and new user requirements surface.
  - The maintenance phase of the life cycle begins following a warranty period or post-implementation support delivery, but maintenance activities occur much earlier.

KA6. Software Configuration Management

- Software configuration management (SCM) is the task of tracking and controlling changes in the software, part of the larger cross-discipline field of configuration management.”
- SCM practices include *revision control, naming conventions,* and the establishment of *baselines*.

KA6. Software Configuration Management

- SCM practices include *revision control, naming conventions,* and the establishment of *baselines*.
  - RCS: automates the storing, retrieval, logging, identification, and merging of revisions.
- If something goes wrong, SCM can determine what was changed and who changed it.

KA7. Software Engineering Management

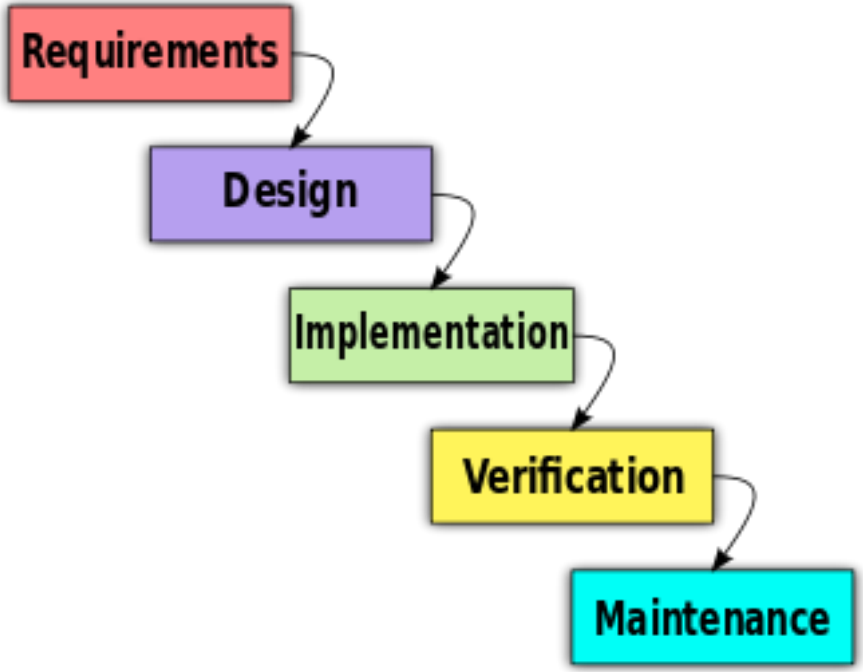
- Software Engineering Management can be defined as the application of **management activities** — planning, coordinating, measuring, monitoring, controlling, and reporting—to ensure that the development and maintenance of software is systematic, disciplined, and quantified.

KA8. Software Engineering Process

- In this knowledge area “software engineering processes” are concerned with work activities accomplished by software engineers to develop, maintain, and operate software, such as software requirements, software design, software construction, software testing, software configuration management, and other software engineering processes.

KA8. Software Engineering Process

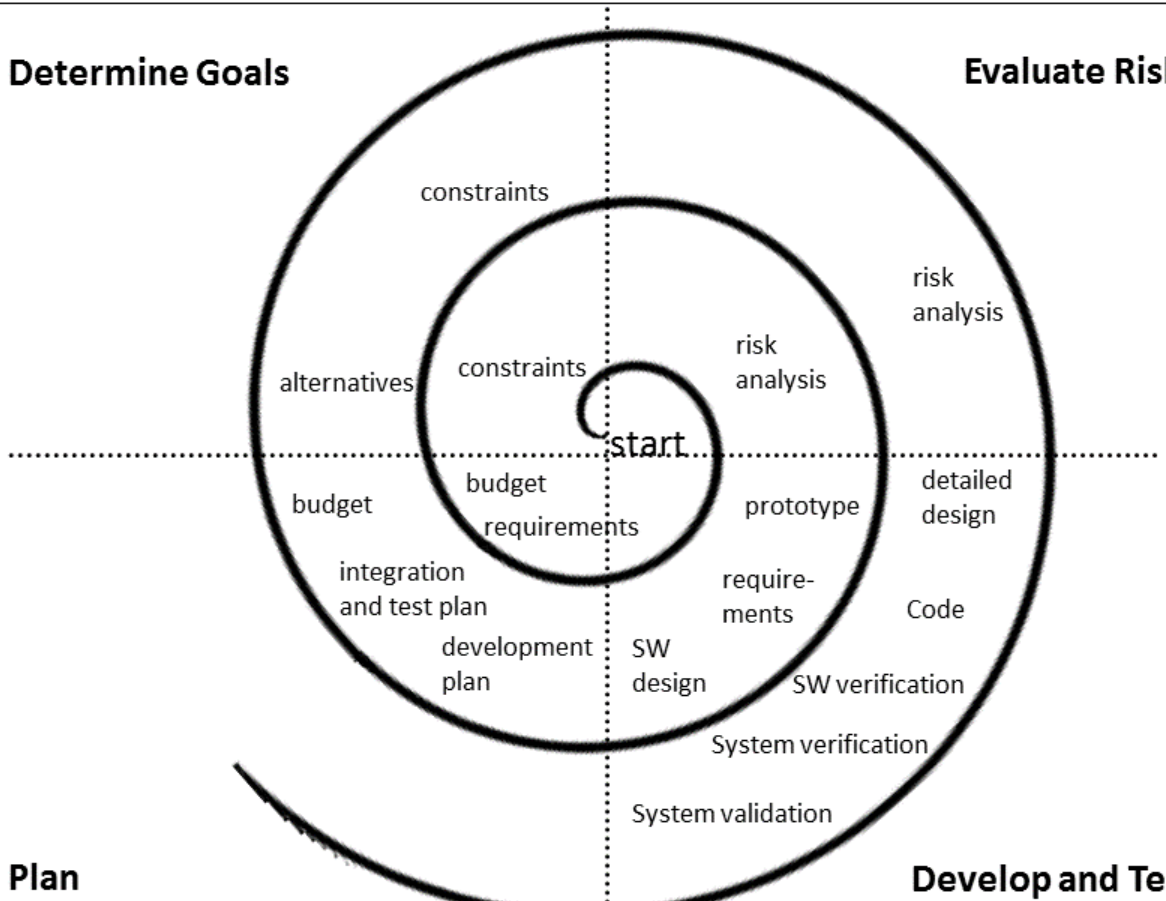
- A SW Process defines who does what and when to produce which artifact for what purpose.



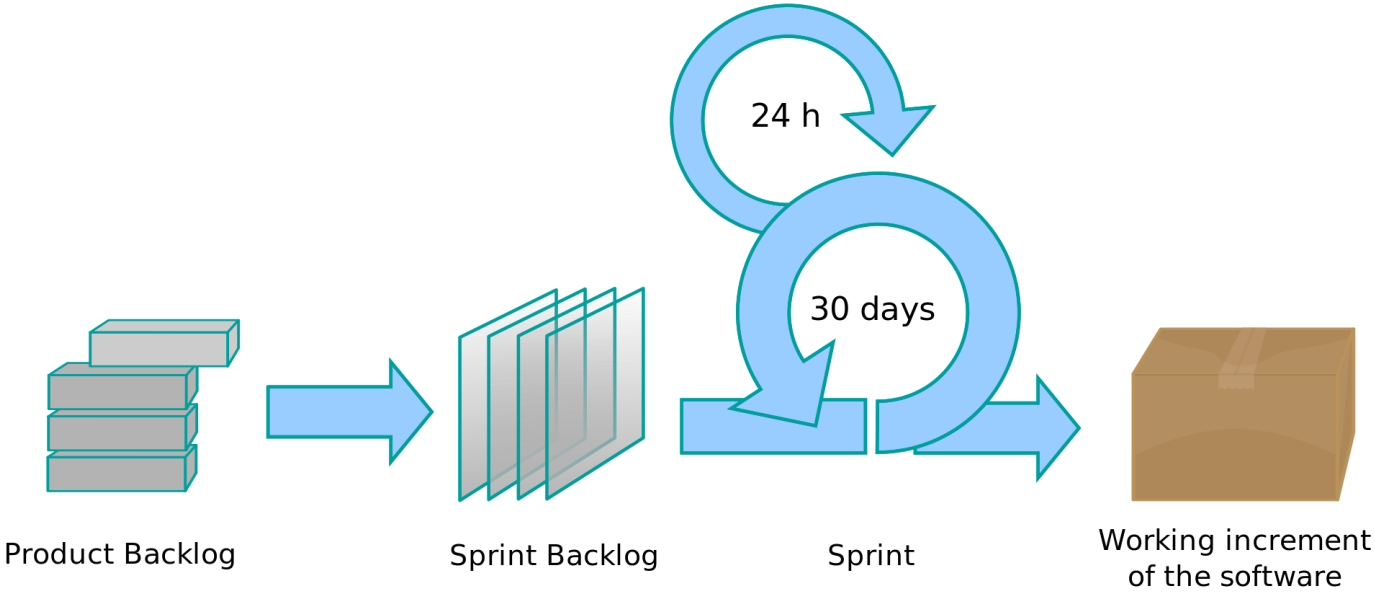
Waterfall Model

KA8. Software Engineering Process

The Spiral Model



KA8. Software Engineering Process



Scrum (Agile Model)

KA10. Software Quality

- Software quality assurance (SQA) consists of a means of monitoring the software engineering processes and methods used to ensure quality. The methods by which this is accomplished are many and varied, and may include ensuring conformance to one or more standards, such as ISO 9000 or a model such as CMMI.

Hooker's General Principles

- 1: *The Reason It All Exists*
- 2: *KISS (Keep It Simple, Stupid!)*
- 3: *Maintain the Vision*
- 4: *What You Produce, Others Will Consume*
- 5: *Be Open to the Future*
- 6: *Plan Ahead for Reuse*
- 7*: Think!*

Course Tools

- This course will introduce a combination of new tools for software Engineering:
  - A Java IDE environment like **Eclipse**, **NetBeans** or **Jcreator**
  - A UML modeling tool like **ArgoUML**, **Visual-Paradigm** or **Rational Rose**
    - https://www.visual-paradigm.com/solution/freeumltool/
  - Project Management tools open project, Microsoft project
  - A configuration management tool / platform like **GitHub, GitLab** or **BitBucket**