Laboratorio di Architetture Software e Sicurezza Informatica [AAF1569]

1 - Software Development Process

Part of therein contents are based on slides of Proff, John Yang and Ian Sommerville

SAPIENZA
UNIVERSITÀ DI ROMA

DIAG
Dipartimento di Ingegneria
informatica, automatica e gestionale
Antonio Ruberti

Intro to Software Engineering

How to **prevent** a software project/product from “being a **failure**”?

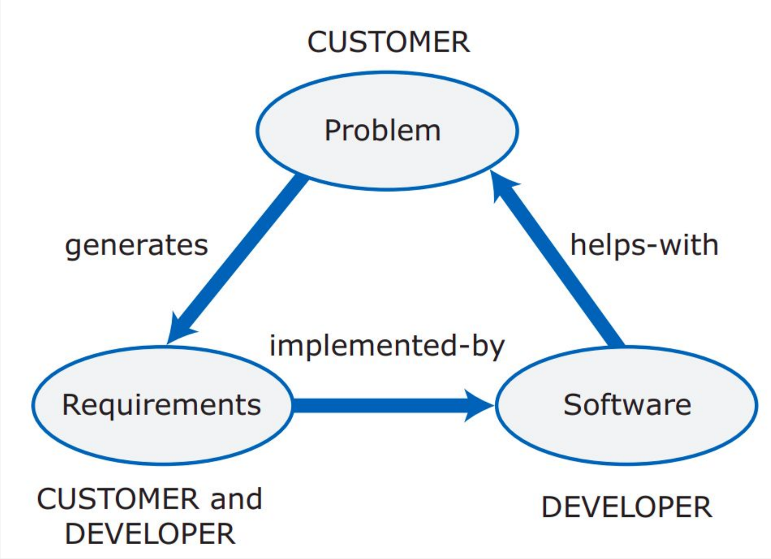
Some causes of a software project/product failure:

- “low quality solution” (“buggy”, unreliable, etc.)
- out of budget
- out of time
- misunderstanding of customer expectations
- several others...

Intro to Software Engineering, Project-base software

Initially, companies and governments wanted to **automate their businesses, custom software**.

Projects involve an external client or **customer who decides** on the **functionality** of the system and enters into a **legal contract** with the software development company.



Project-based software engineering

Intro to Software Engineering, Product-based software

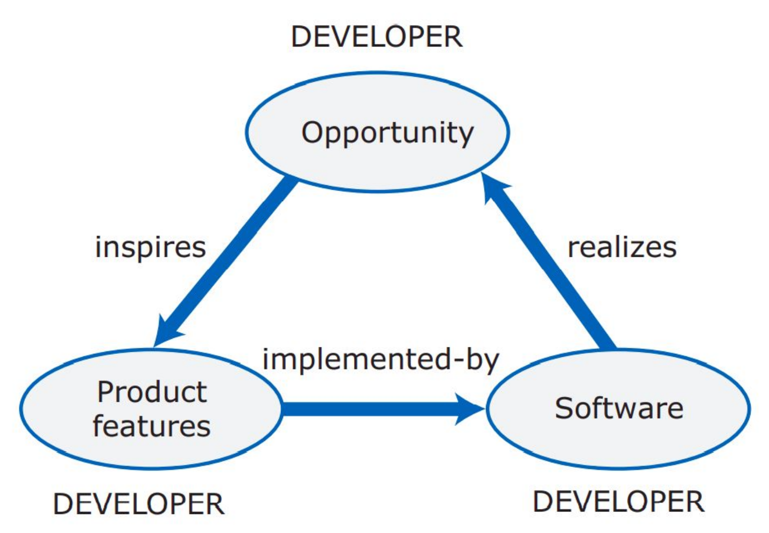
After a while, most businesses didn't really need custom software: **common business problems**

Project-based software ⇒ **Product-based software**

The starting point is an **opportunity**

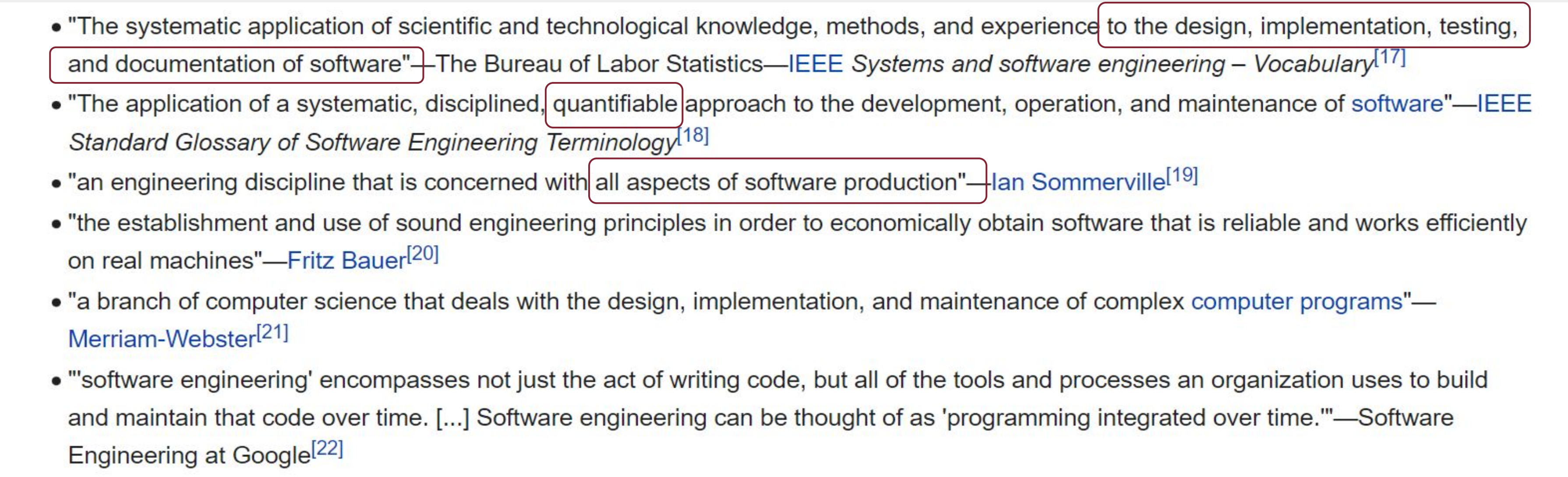
There is **no external customer** who creates requirements.

Most of the times, getting the **product to customers quickly is critical**.



Product-based software engineering

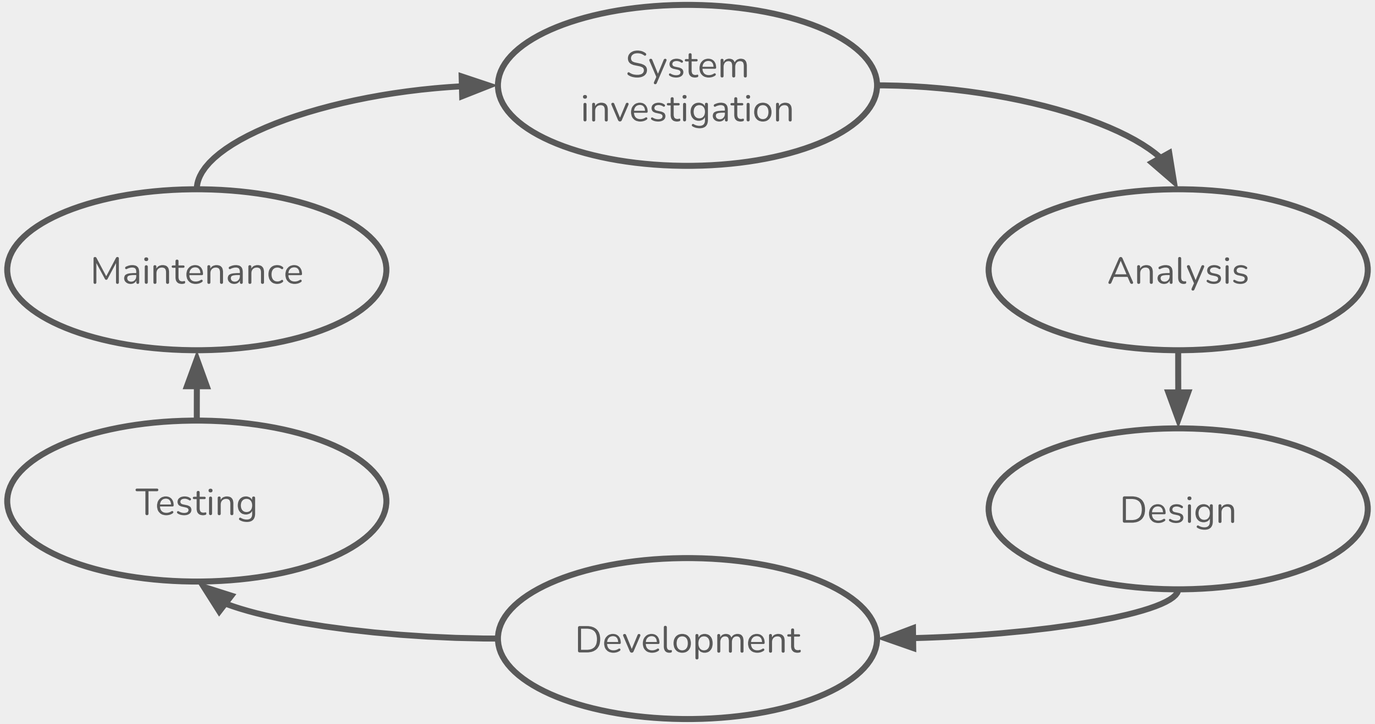
Intro to Software Engineering



- "The systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software"—The Bureau of Labor Statistics—IEEE *Systems and software engineering – Vocabulary*[17]
- "The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software"—IEEE *Standard Glossary of Software Engineering Terminology*[18]
- "an engineering discipline that is concerned with all aspects of software production"—Ian Sommerville[19]
- "the establishment and use of sound engineering principles in order to economically obtain software that is reliable and works efficiently on real machines"—Fritz Bauer[20]
- "a branch of computer science that deals with the design, implementation, and maintenance of complex computer programs"—Merriam-Webster[21]
- "'software engineering' encompasses not just the act of writing code, but all of the tools and processes an organization uses to build and maintain that code over time. [...] Software engineering can be thought of as 'programming integrated over time.'"—Software Engineering at Google[22]

from Wikipedia

Software Lifecycle



Software Development Processes

Several Software Development Process **Models**

They **differ** from each other for the foreseen **activities** and for the **documents** that are **produced**

- **Plan-and-document**
- AGILE

Plan-and-Document Processes

- Before coding, **make the plan**
- Write **detailed documentation** all phases of plan
- **Progress measured** against the plan
- **Changes** to project must be **reflected in documentation** and possibly to plan

⇒ **significant overhead** in planning, designing, and documenting the system.

“In the 1980s and early 1990s, there was a widespread view that the best way to create good software was to use controlled and rigorous software development processes.” source: Engineering Software Products: An Introduction to Modern Software Engineering by Ian Sommerville

Waterfall

1. Requirements analysis and specification
2. Architectural Design
3. Implementation + Integration
4. Verification
5. Operation + Maintenance

Good: earlier you find an error the cheaper it is to fix

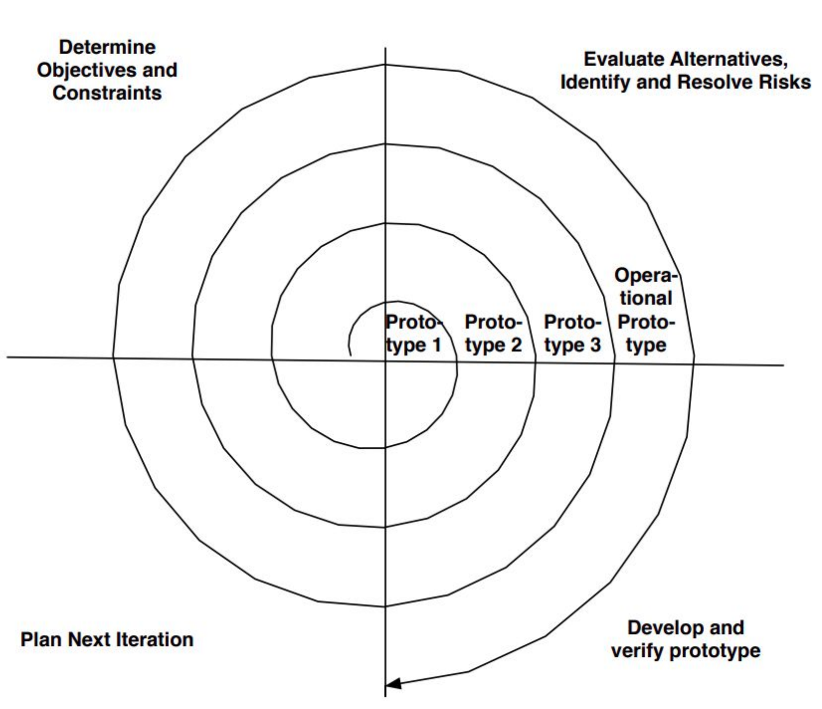
Problem: Specification **errors**, omissions, and misunderstandings are often discovered **only after** a significant chunk of the system has been implemented.

Problem: Easier for customers to understand what they want once they see a prototype and/or **customers change their minds**, propagating changes takes time

Spiral

Combine Waterfall + Prototypes ⇒ Spiral Model

1. **Determine objectives** and constraints of this iteration
2. **Evaluate alternatives** and identify and resolve risks
3. **Develop and verify** the prototype for this iteration
4. **Plan** the next iteration


Determine Objectives and Constraints
Evaluate Alternatives, Identify and Resolve Risks
Proto-type 1
Proto-type 2
Proto-type 3
Opera-tional Proto-type
Plan Next Iteration
Develop and verify prototype

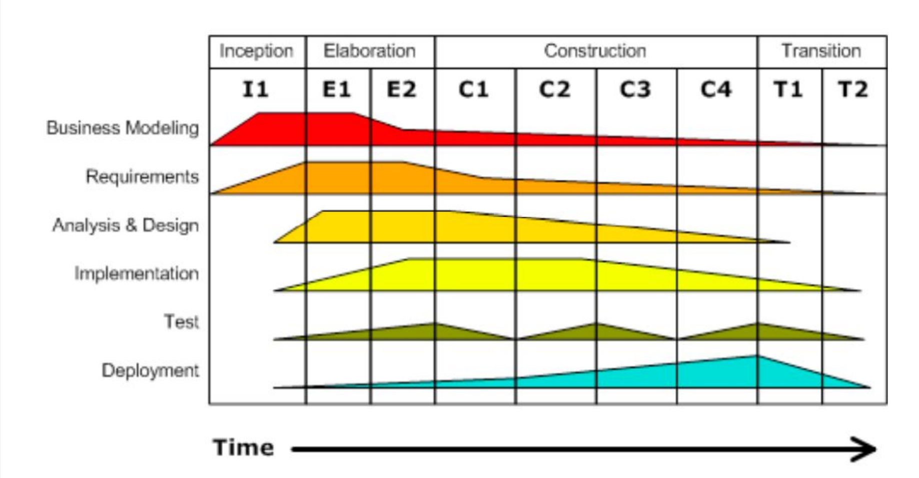
Iterations **involve the customer** before the product is completed

Rational Unified Process (RUP)

“Allied **closer to business than technical issues**”. Software life cycle is made of development cycles, divided in turn in phases:

1. **Inception**: Make business case + scope project, justify costs + assess risks
2. **Elaboration**: Identify use cases w/ customers, set dev plan + build prototype
3. **Construction**: Code+Test product, push first external release
4. **Transition**: Move product from release to production

Each phase may have several iterations



Agile Methods

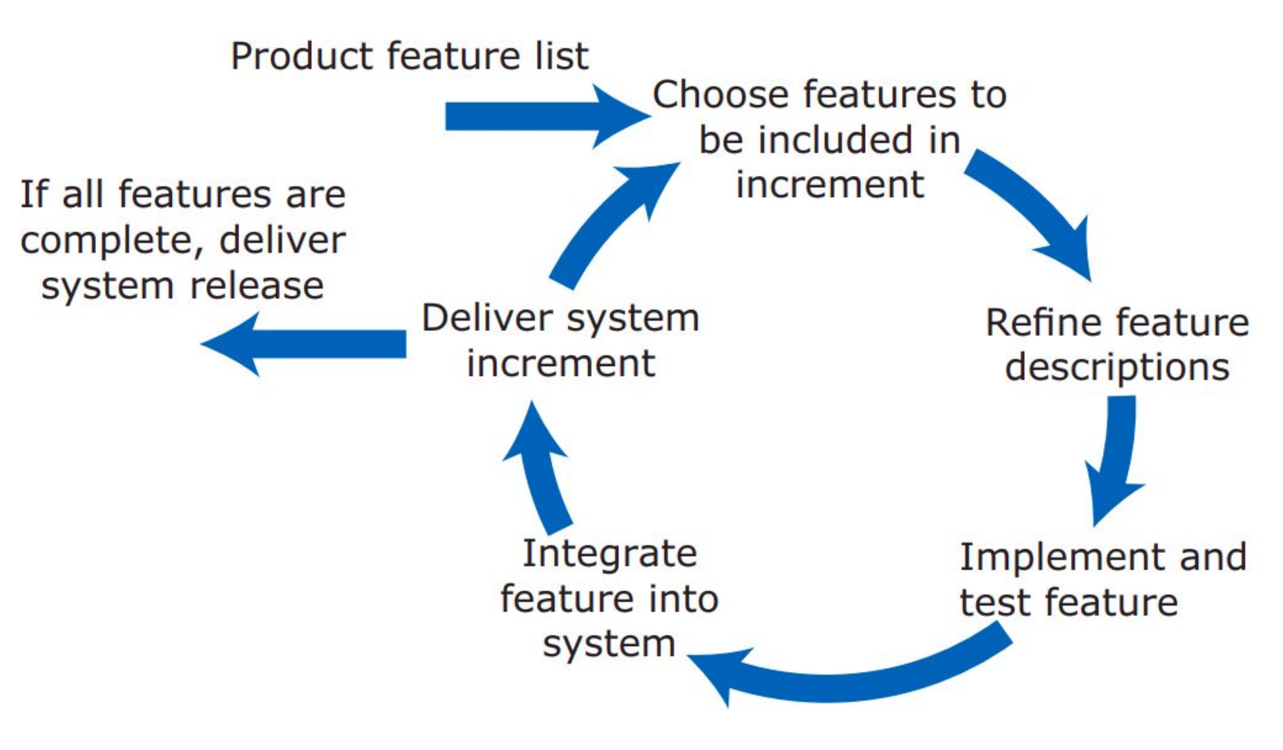
1990s. **Focus on the software development itself**, rather than on its design and documentation.

The Agile Manifesto’s values http://agilemanifesto.org/

- “***Individuals and interactions*** *over processes and tools*”
- “***Working software*** *over comprehensive documentation*”
- “***Customer collaboration*** *over contract negotiation*”
- “***Responding to change*** *over following a plan*”

⇒ **Embrace change**
⇒ **Continuously refine** a working but incomplete prototype
⇒ Relies on quick, constant **customer feedback**
⇒ emphasizes **test-driven development** (TDD)
⇒ **user stories** to reach agreement
⇒ velocity to **measure project progress**

Incremental Development



Product feature list

Choose features to be included in increment

Refine feature descriptions

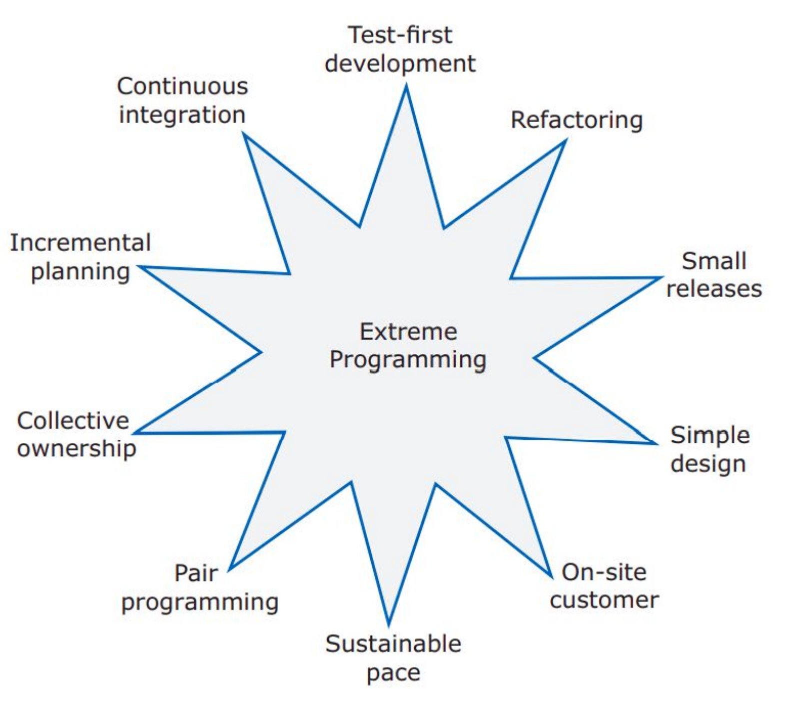
Implement and test feature

Integrate feature into system

Deliver system increment

If all features are complete, deliver system release

Extreme Programming (XP)



Extreme Programming **practices**

“YAGNI” (You Ain’t Gonna Need It)

⇒ **include only functionality that is requested**

⇒ customers rarely understand **system-wide issues** such as security and reliability

⇒ cope with situations that customers are unlikely to foresee

Scrum

Managers **need to know what is going on** and whether or not a software development project is likely to **deliver** the software **on time** and **within its budget**

Scrum is **not based on a set of technical practices**. Rather, it is designed to **provide a framework for agile project organization.**

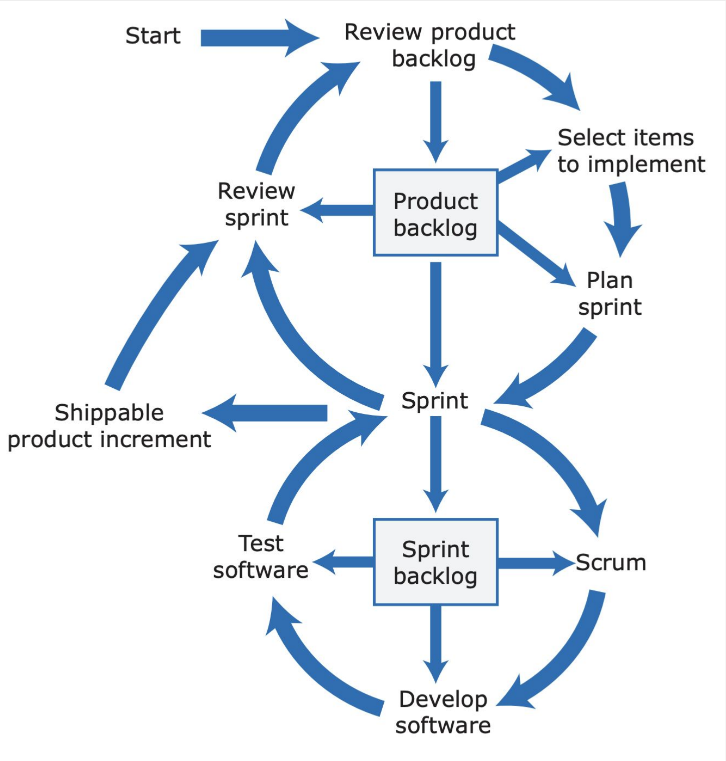
Scrum

A **sprint** is a fixed-length activity (two to four weeks) defined in a **sprint planning meeting**

The team has **daily meetings** (**Scrums**) to review the work done so far and to agree on that day's activities.

The **sprint backlog** is used to keep track of work that is to be done.

On completion of a sprint, a sprint **review meeting** is held involving all team members: learn from each other to avoid problems and to improve productivity in later sprints.


Start
Review product backlog
Select items to implement
Product backlog
Review sprint
Plan sprint
Sprint
Shippable product increment
Test software
Sprint backlog
Scrum
Develop software