Week 1 Software Development: Life Cycle and Methodologies

- Developing Methods

Code-a-bit-test-a-bit (CABTAB)

Waterfall

Rapid prototyping

Iterative and incremental

Agile methods

CABTAB

Code-a-bit-test-a-bit (CABTAB) is hardly recognized as a methodology, although it is widely used in programming.

It is unsuitable for anything other than very small systems of limited scope.

Software development life cycle (SDLC)

The SDLC is the sequence of activities covering

requirements

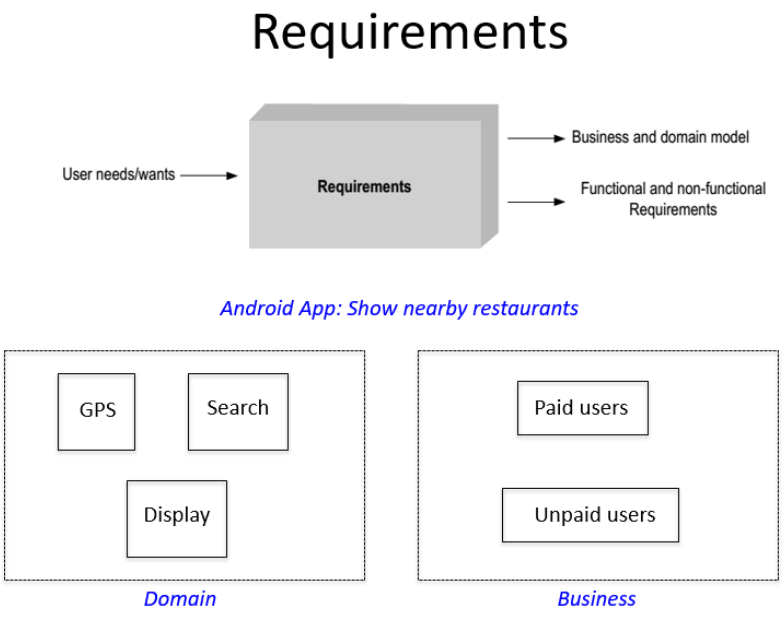
analysis

design

implementation

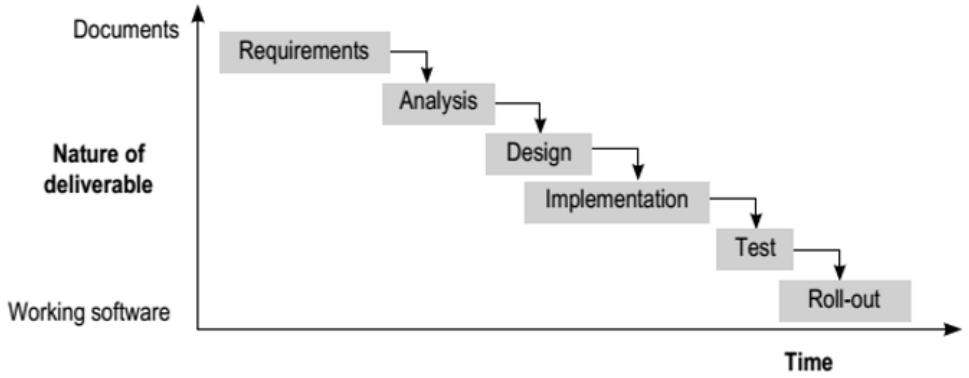
testing

over which a software system is developed.



Waterfall Model

Waterfall model



Software Engineering: Concepts and Applications by Subhajit Datta (Oxford University Press, 2010)

Limitation: The waterfall model assumes that the different stages of software development are sequential.

Rapid Prototyping

Rapid prototyping recommends the building of prototypes to clarify requirements and system scope.

The prototypes, however, should never become the final system.

Once the requirements have been sort out, the system is built formally

Timing is crucial for the prototype

Iterative and Incremental Development

In iterative and incremental development, the software system is built through a series of time-boxed development cycles—iterations—leading to tangible and testable additions to the overall system functionality—increments.

This is an expedient model for building systems with initial ambiguity of scope and changing requirements.

The Agile Manifesto

•Customer satisfaction by rapid delivery of useful software

•Welcome changing requirements, even late in development

•Working software is delivered frequently (weeks rather than months)

•Close, daily cooperation between business people and developers

•Projects are built around motivated individuals, who should be trusted

•Face-to-face conversation is the best form of communication (co-location)

•Working software is the principal measure of progress

- Sustainable development, able to maintain a constant pace
- Continuous attention to technical excellence and good design
- Simplicity—the art of maximizing the amount of work not done—is essential
- Self-organizing teams
- Regular adaptation to changing circumstance

The Software Equation

$$(B^{1/3} * Size) / Productivity = Effort^{1/3} * Time^{4/3}$$

B = Special skill factor
Productivity = a productivity parameter (depends on team)
Effort = Human effort in years or months
Time = Time to complete project in years or months

The models basis was formed through analysis of productivity data collected from over 4000 modern day software development projects.[1]

The software equation was derived from the Putnam-Norden-Rayleigh Curve which can be used to show the non-linear correlation between time to complete the project and applied human effort.[2]

OBJ04-J

Provide mutable classes with copy functionality to safely allow passing instances to untrusted code

Mutable classes allow code external to the class to alter their instance or class fields. Provide means for creating copies of mutable classes so that disposable instances of such classes can be passed to untrusted code.

Noncompliant code example

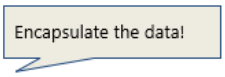
```
public final class MutableClass {
    private Date date;
    public MutableClass(Date d) {
        this.date = d;
    }
    public void setDate(Date d) {
        this.date = d;
    }
    public Date getDate() {
        return date;
    }
}
```

What if a malicious users uses this class?

OBJ04-J

Provide mutable classes with copy functionality to safely allow passing instances to untrusted code

Mutable classes allow code external to the class to alter their instance or class fields. Provide means for creating copies of mutable classes so that disposable instances of such classes can be passed to untrusted code.



Compliant code example

```
public final class MutableClass {
    private final Date date;
    public MutableClass(MutableClass mc) {
        this.date = new Date(mc.date.getTime());
    }
    public MutableClass(Date d) {
        this.date = new Date(d.getTime());
    }
    public Date getDate() {
        return (Date) date.clone();
    }
}
```