Software Platform

Lecture 1

Emanuele Di Buccio

26/02/2024

Master Degree in Computer Engineering, A.A. 2023/2024

Course Teacher and Website

Teacher

- Emanuele Di Buccio (emanuele.dibuccio@unipd.it)
- Use [softplat] as a prefix to the email object

Website

- https://stem.elearning.unipd.it/course/view.php?id=8355
- https://en.didattica.unipd.it/off/2023/LM/IN/IN2547/004PD/INQ0091600/N0

Office Hours

- Contact the teacher via email by your institutional account: @studenti.unipd.it

General Information on the Course 1/-

- Credits: 6
- Number of lectures: 24
- Teaching hours: 48
- Hours of Individual study: 102
- Lecture schedule and rooms:
  - Monday 10:30-12:30 - Room Fe
  - Friday 10:30-12:30 - Room Fe

In this lecture

- Course Topics
- Exam
- Material

Course Topics

Course Topics Overview

1. Software Platforms, Software Engineering, and DevOps
2. Micro-services
3. Java for Complex Software Platforms
4. Cloud Services

Exam

Exam

- questionnaire (50% of the final score)
- group project (50% of the final score)

Questionnaire

- Questions on the course topics
- How many questions? How many minutes?
  - 30 questions
  - 90 minutes
- (Possible) types of questions
  - True/False, but you must provide a motivation
  - Multiple choice
  - Fill with missing words/terms
  - Comment fragments of code

Project (1)

- Group Project
- Design and development of platform using multiple services
- Details on the project, e.g., requirements, will be presented in a dedicated lecture
- The project must use Java
- Please ask the lecturer to use additional programming languages and/or frameworks

Project (2)

Project evaluation criteria

- Documentation
- Code
- Reproducibility
- Presentation

Material

Material

- Material used during the lectures (e.g. slides, source code) will be made available on the course page
- If the slides are not easily readable because of the font, the font size, or the colors, please let me know
- Books (suggested)
    - Software Engineering [Sommerville, 2016]
    - Distributed Systems [Tanenbaum and Steen, 2023]
    - Microservice Patterns [Richardson, 2019]

More on Course Topics

What is a software platform?

- Software
- Platform

What is software?

- “Computer programs and associated documentation.” [Sommerville, 2016]
- “Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system” [IEEE, 1990]
- therefore, not only *instructions* provided to a computer
- (word) first published by John Wilder Tukey, mathematician and statistician (FFT)

What is a platform?

https://flic.kr/p/qTATnx

What is a software platform? (1)

What is a 'Platform' anyway?

- (one of the) most ambiguous terms we could use . . .
- "Definitions for software and hardware platforms abound, generally describing **an operating environment upon which applications can execute and which provides reusable capabilities** such as file systems and security."

What is a software platform? (2)

From [Sommerville, 2016]

> *Most software interfaces with other software systems. These other systems include the operating system, database, middleware, and other application systems. These make up the "software platform,' the environment in which the software will execute*

What is a software platform? (3)

From [Salatino, 2023]

> *Platforms are a collection of services that help companies get their software running in front of their customers (internal or external).*

What is a software platform? (4)

Java? See here

Definition of Software Platform from Wikipedia

What is a software platform? (5)

From the Oxford Dictionary

> *“an online system, application, storefront, or website, through which a digital service is provided, or digital products are sold, particularly a website or application that hosts user-generated content that can be shared and engaged with by other users”*

What is a software platform? (6)

Online Platforms

- “[. . .] cover a wide range of activities including online marketplaces, social media, creative content outlets, app stores, price comparison websites, platforms for the collaborative economy as well as search engines.”
- “share key characteristics, such as the use of information and communication technologies to facilitate interactions between users, collection and use of data about such interactions, and network effects”

Look at The Digital Service Act

What is a software platform? (7)

(Again) What is a 'Platform' anyway?

- "A **digital platform** is a foundation of self-service APIs, tools, services, knowledge and support which are arranged as a compelling internal product. Autonomous delivery teams can make use of the platform to deliver product features at a higher pace, with reduced co-ordination."

What is Software Engineering?

Software Engineering

- term suggested at conferences organized by NATO in 1968 and 1969 to discuss the ‘software crisis’ [see here]
- “[Software engineering is] the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.” [Fritz Bauer]
- “Software Engineering: (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1).” [IEEE, 1990]

Software Engineering Recap Topics

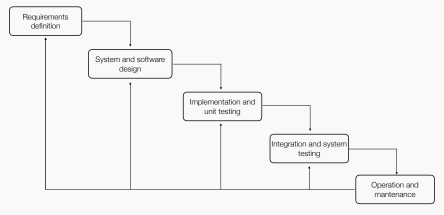
We are going to recap some Software Engineering topics/content:

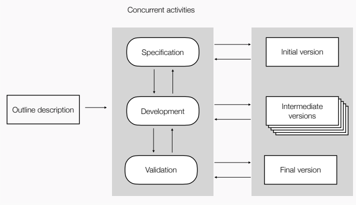
- Software Processes and Models, Agile
- Patterns
- Software Architectures and Architectural Patterns
- Testing

Software processes and paradigms (1)

- “A software process is a set of related activities that leads to the production of a software system” [Sommerville, 2016]
- “A *process* is a collections of activities, actions, and tasks that are performed when some work product is to be created” [Pressman, 2009]
- fundamental activities
    - software specification
    - software development
    - software validation
    - software evolution

Software processes and paradigms (2)


Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and mantenance



Figures adapted from [Sommerville, 2016]

Software processes and paradigms (3)

Process framework [Pressman, 2009]: a small number of activities applicable to all software processes:

- communication
- planning
- modeling
- construction
- deployment

Software processes and paradigms (4)

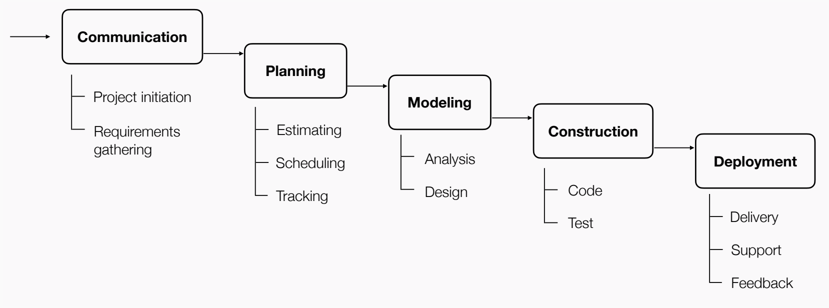
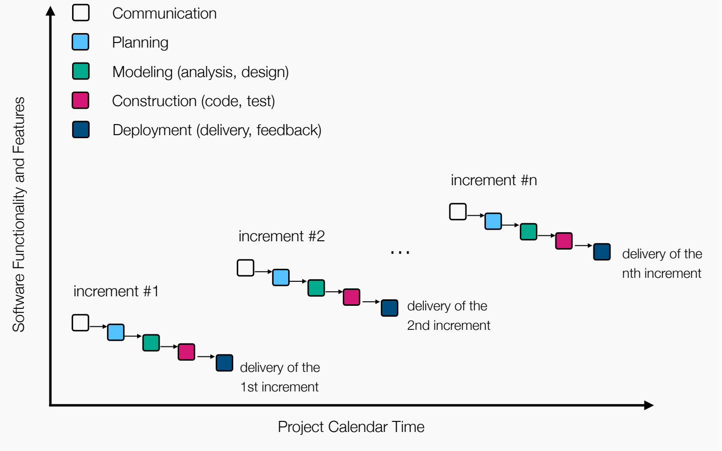


Figure adapted from [Pressman, 2009]

Software processes and paradigms (5)



Communication
Planning
Modeling (analysis, design)
Construction (code, test)
Deployment (delivery, feedback)
Software Functionality and Features
increment #1
delivery of the 1st increment
increment #2
delivery of the 2nd increment
. . .
increment #n
delivery of the nth increment
Project Calendar Time

Figure adapted from [Pressman, 2009]

Deployment (1)

From [Humble and Farley, 2010]

- “If somebody thinks of a good idea, how do we deliver it to users as quickly as possible?”
- “What happens once requirements are identified, solutions designed, developed, and tested?”
- “Effective pattern for getting software from development to release”

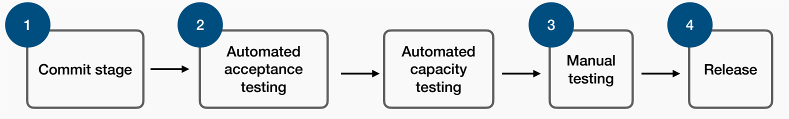
⇒ **Deployment pipeline**

Deployment (2)

What is a deployment pipeline?

- Automated implementation of application build, deploy, test, and release process
- “At an abstract level, a deployment pipeline is an automated manifestation of your process for getting software from version control into the hands of your users.” [Humble and Farley, 2010]
- Basic idea: every change in the configuration, source code, environment, or data creates a new instance of the pipeline
- Two (major) parts:
  - create binaries and installers
  - runs a series of tests to prove that the new version can be **released**

Deployment (3)



Stages

1. asserts that the system compiles, passes a suite of automated tests, and runs code analysis
2. asserts that the system works at the functional and nonfunctional level
3. asserts that the system is usable and fulfills its requirements, detects any defects not caught by automated tests, and verifies that it provides value to its users
4. delivers the system to users, either as packaged software or by deploying it into a production or staging environment

Patterns (1)

What is a *pattern*?

- “the regular way in which something happens or is done” (from the *Oxford Dictionary*)
- “Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution of that problem, in such away that you can use this solution a million times over, without ever doing ti the same way twice.” [Alexander et al., 1977]

Patterns (2)

Examples of patterns (in other fields) [Gamma et al., 1995]

- the “Tragically Flawed Hero” (Plots)
- the “Romantic Novel” (Plots)
- open stairs (Architecture, Pattern 158 from [Alexander et al., 1977])
- short passages (Architecture, Pattern 132 from [Alexander et al., 1977])

Patterns (3)

What is a *pattern language*?

- $\Rightarrow$ a set of organized patterns

In [Alexander et al., 1977] a language for building and planning

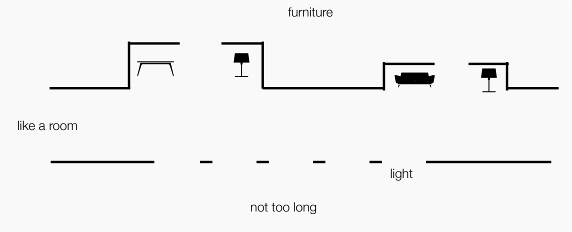
Patterns (4)

Short passages (Pattern 132 from [Alexander et al., 1977])

- **Name**: Short passages
- **Context**: "THE FLOW THROUGH ROOMS (131) describes the generosity of light and movement [. . .] But when there has to be a passage in an office or a house and when it is too small [. . .]".
- **Problem**: "long, sterile corridors set the scene for everything bad about modern architecture"

Patterns (5)

- **Solution**: "Keep passages short. Make them as much like rooms as possible, with carpets or wood on the floor, furniture, bookshelves, beautiful windows. Make them generous in shape, and always give them plenty of light; the best corridors and passages of all are those which have windows along an entire wall."



Patterns (6)

We can benefit from patterns also in Computer Engineering

- Architectural Patterns
- Design Patterns (GoF)
- GRASP Patterns
- Microservice Patterns

Patterns (7)

We can benefit from patterns also in Computer Engineering

- Architectural Patterns
- Design Patterns (GoF)
- GRASP Patterns
- Microservice Patterns

What is a Software Architecture?

- “The software architecture of a computing system is the set of structures needed to reason about the system, which comprises software elements, relations among them, and properties of both.” [Bachmann et al., 2011]
- “A software architecture is a description of how a software system is organized. Properties of a system such as performance, security, and availability are influenced by the architecture used.” [Sommerville, 2016]

Why Microservices? A scenario (1)

Scenario proposed in [Richardson, 2019]

- You are the Chief Technology Officer (CTO) of the Food To GO, Inc. (FTGO)
- You spent the last weeks at an exciting conference on the latest software development techniques
- You come back to your company and . . . to a deadline probably the team is missing
- Why? A case of **monolithic hell**

Why Microservices? A scenario (2)

What does the FGTO company do?

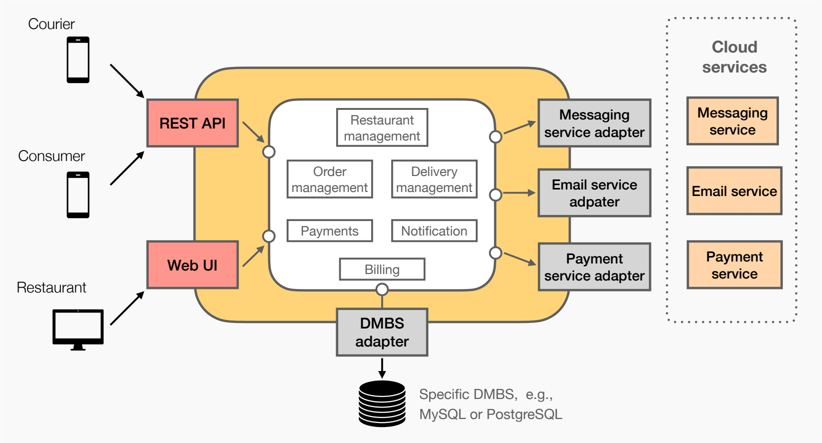
- fictitious company
- launched in 2005, now a leading online food delivery company
- provides an application (web and mobile) that allows customers to place orders at local restaurants
- coordinates a network of couriers who deliver the orders

Why Microservices? A scenario (3)

The FGTO application

- allows customers to place orders at local restaurants
- is responsible for paying couriers and restaurants
- allows the restaurants to edit the menu
- **uses various Web services** for payment, messaging, and email
- consisting of a single Java Web Application Archive (WAR) file

Why Microservices? A scenario (4)



Based on Figure 1.1 from [Richardson, 2019], accessible here

Why Microservices? A scenario (5)

- The FTGO application is an example of **monolithic** architecture
- Should I stay always away from **monolithic** architectures? **No!**

  Some **advantages**:

  - “Simple” to develop
  - Easy to make radical changes to the application
  - Straightforward to test
  - Straightforward to deploy
  - Easy to scale

Why Microservices? A scenario (6)

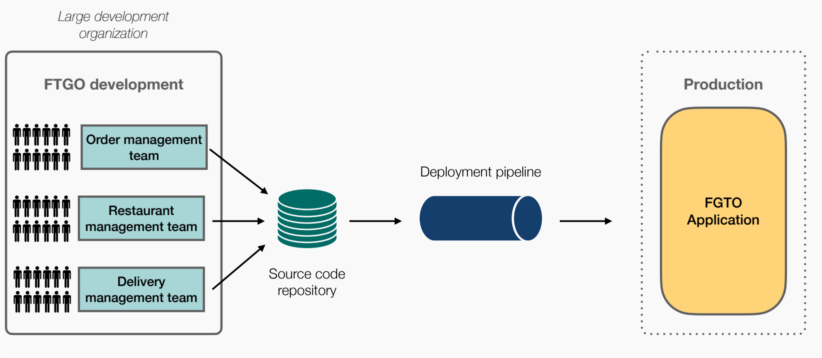
Let us assume that FGTO is successful.

Possible issues:

- FGTO is too complex/large for any developer to fully understand, and that affects bug fixing and new feature implementation
- development is slow because building the application and its start-up take a long time
- path from commit to deployment can be long
- difficulty in scaling, e.g., because of conflicting resource requirements
- locked into the increasingly obsolete technology stack

Why Microservices? A scenario (7)

Let us focus on: **path from commit to deployment can be long**



What are microservices?

Possible definition (by Adrian Cockcroft here, slide 24)

- "Loosely coupled service-oriented architecture with bounded context"

We will "unpack" the definition in the next lecture, focusing on the following expressions:

- Loosely coupled
- service-oriented architecture
- bounded context

References

References (1)

Alexander, C., Ishikawa, S., and Silverstein, M. (1977).
A Pattern Language: Towns, Buildings, Construction.
Oxford University Press.

Bachmann, F., Bass, L., Clements, P., Garlan, D., Ivers, J., Little, R., Nord, R., and Stafford, J. (2011).
Documenting software architectures.
Pearson Education, Inc.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995).
Design Patterns: Elements of Reusable Object-Oriented Software.
Addison-Wesley Longman Publishing Co., Inc., USA.

References (2)

Humble, J. and Farley, D. G. (2010).
Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation.
Addison-Wesley, Upper Saddle River, NJ.

IEEE (1990).
IEEE standard glossary of software engineering terminology.
IEEE Std 610.12-1990, pages 1–84.

Pressman, R. (2009).
Software Engineering: A Practitioner's Approach.
McGraw-Hill, Inc., USA, 7 edition.

References (3)

Richardson, C. (2019).
Microservices Patterns: With examples in Java.
Manning.

Salatino, M. (2023).
Platform Engineering on Kubernetes.
Manning.

Sommerville, I. (2016).
Software Engineering.
Pearson, 10th - global edition edition.

Tanenbaum, A. S. and Steen, M. v. (2023).
”Distributed Systems (4th edition).
Maarten van Steen.

Questions?