Advanced Software Engineering Internet Applications

Foreword

J.P.'s Vision of soft. eng. evolution over the last decades and how this course fits into it

Early Eighties

- It was just programming with a procedural approach
  - The functional requirements were fixed at the beginning.
  - The code was optimized in terms of CPU cycles and memory =>
    - none or very little modularity;
    - none or very little encapsulation, i.e. no separation between the interface (what a piece of software achieves) and its implementation (how it does perform its task).
  - No notion of components and very limited reusability.
  - No distributed system.

Around 1985

- 1985 Software-ICs paper form Ledbetter and Cox in BYTE.
  - The key is encapsulation.
  - In a message/object system, messages should specify the what and leaves the how hidden from users.
  - Object Oriented Programming (i.e. classes / objects + inheritance) start to impose itself as the new standard.
  - Later the idea of Aspect Oriented Programming emerges and finds its final outcome with injection / annotations.

The Nineties

- Notion of Framework
  - Just implement the necessary methods and let the framework do the job.
- In 1994 the gang of four publish their design patterns book.
  - The most applied is probably the MVC one (for almost every modern web applications)

Around 2000

- Notion of Components (Szyperski in 2002) with four concepts :
  1. Interfaces : Must be well-defined.
  2. Explicit context dependencies.
  3. Reusability.
  4. Lifecycle and Container.
- Appearance of performing middlewares for distributed systems.

Now

- The ultimate components are web services and the ultimate "container" is the web itself with its servers. They are mainly :
  - the big WS-* web services, which use SOAP on top of http as a transport protocol,
  - the lighter RESTful web services, which use directly http as an application protocol.
  - a new tendency is programming with micro-services.
- To create new applications, they can be composed, orchestrated or mashed-up (e.g. using a workflow engine).
- Some advocates serverless development using the cloud (often combined with Edge and/or Fog Computing) to provide everything from hardware to specific softwares.
- Development teams use agile (e.g. scrum) methodology.

JP's Vision vs. ASE Course

- Theoretical part
  - Web services, cloud, semantic web, … right on the target of the latest trend in S.E.
- Practical (project) part
  - Python and JavaScript are modern rapid prototyping languages with larges communities.
  - Aiohttp and Koa are frameworks.
  - The thingy API will have to be a modern RESTful light web service.
  - The thingy and the Node-RED clients will have to consume various web services and combine them.
  - The development in small teams will follow a Scrum approach.