SOC

LECTURE NOTES

Service Oriented Computing

By, **Dr. Neila Ben Lakhal**
@ENICARTHAGE

Chapter 0. Course Introduction

Neila BEN LAKHAL

Neila.Benlakhal@gmail.com
Neila.Benlakhal@enicarthage.rnu.tn

M.E. 2004 - Ph.D 2007

東京工業大学
Tokyo Institute of Technology

Assistant Professor in CS since 2009

ENICARTHAGE
المدرسة الوطنية للمهندسين بقرطاج
Ecole Nationale d'Ingénieurs de Carthage

Motivational question

What is Service Oriented Computing (SoC)?

Service-Oriented Computing (SOC) is a computing paradigm for building **distributed** information systems(IS) in which the concepts of distribution, openness, asynchronous messaging and loose coupling take a leading role.

SoC utilizes **services** as fundamental elements for developing applications.

Service-based architectures allow to realize the vision of SOC.

Course main objective

- Lay a strong foundation on which you can build a successful career as a **software architect**.
- How ?
  - Develop a basic understanding of **software architectural styles**.
  - Develop an in-depth understanding of **service architectures.**
- Participants to this course will be able to look at current and future developments in **service architectures** with enough background to be able to judge :
  - Which **service architecture** to choose for your system.
  - How much of a contribution **service architectures** offer.

What this course covers

- Many software architectural styles were used since 1960.
- In this lecture, we are mainly concerned with **service-based** architectural styles. In particular, we will focus on :
  - **Service Oriented Architecture (SOA)**
  - **Microservice Architecture (MSA)**

Course content 1/3

Chapter 1, Demystifying Software Architecture, begins the course by providing a definition of software architecture. This chapter lists and details a number of software architecture styles classifications. We will also present different object-oriented distributed computing technologies, such as Common Object Request Broker Architecture (CORBA), the Distributed Component Object Model (DCOM), and Remote Method Invocation (RMI) and list their limitations that led to service architectural styles emergence. Several examples of each type of architectural style will be also provided.

Chapter 2, Service-Oriented Architecture (SOA), demystifies the popular service-oriented architecture (SOA) patterns that produce service-oriented applications. This chapter provides details on the principles and key concepts of SOA. From this chapter, students will understand the purpose of SOA, the technologies and protocols (web services) used for SOA, and the typical use cases of SOA. Ultimately, students will get to know the deployment difficulties associated with SOA applications.

Course content 2/3

Chapter 3, Web Services-Core functionality and standards will go through the WS-* standards, technologies and APIs that are commonly used to implement a service-oriented application. Bottom-up and top-down service implementations approaches are explored. This chapter introduces also two service integration/composition techniques, namely service orchestration and choreography for composing individual services so as to achieve useful business processes. By the end of this chapter, students will come to know about service implementation, testing, discovery, publication and invocation platforms. Basic concepts of orchestration and choreography, with orchestration and choreography readily available orchestration platforms and workflow engines like BPEL are also explained.

Chapter 4, REST Architecture introduces the concepts related to the programmable Web, shows how HTTP and Web services are related to each other, introduces the principles behind REST Architectural Style, explains how HTTP verbs are used in REST applications and explains the need for RESTFul web services. By the end of this chapter, students will learn the core architectural elements that form a RESTful system and will get introduced to some frameworks and tools that can be used to work with REST.

Course content 3/3

Chapter 5, Microservice Architecture (MSA) : key architectural elements The objective of this chapter is to introduce the essentials of MSA as an architecture that breaks down an application into microservices, each of which can be independently, as well as frequently, deployed and thus facilitate continuous delivery of the application to customers. By the end of the chapter, students will understand the need for MSA, how MSA meets the modern business requirements, and typical use cases, benefits, drawbacks, how MSA differs from SOA. Furthermore, students will learn key architectural elements involved in developing an MSA-based application: the communication models of MSA, microservice discovery and the role of the API gate way in discover. He will understand how services interact with one another synchronously and asynchronously using standard architectures, protocols, and message brokers and how clients discover and access different services using service discovery mechanisms and the API gateway.

Chapter 6, Docker: a Defacto Platform for Microservices Architecture. To leverage the benefits of the microservices architectural style, one needs to use technologies aligned with the characteristics of microservices. Containerization has become a popular deployment format for microservices, and Docker is the leading container platform that packages everything needed to deploy and run microservices (i.e., code, libraries, settings, etc.) .With a lot of advantages, Docker become the best fit to implementing microservices architecture. In this Chapter we will dive deep in Docker to see how Docker can effectively help in leveraging microservices architecture.

Tentative schedule

- Week 0 : Course Introduction
- Week 1: Chap1 (Software architectural styles)
- Week 2: Chap2 (SOA principles-Key actors- Service properties, Lifecyle)
- Week 3: Chap3 (SOAP. Bottom-up WS implementation, Demos, Testing)
- Week 4: Chap3 (WSDL. Top-down WS implementation, Messaging)
- Week 5: Chap4 (REST principles, WADL, HTTP verbs, Demos)

Week 6: Mid-term exam

- Week 7: Chap5 (MSA principles, elements)
- Week 8: Chap6 (Docker)
- Week 9: Chap6 (Docker)
- Week 10: Chap6 (Docker)
- Week 11:Chap6 (Docker)
- Week 12: Evaluation

References and Textbooks

- Chellammal Surianarayanan, Gopinath Ganapathy, Raj Pethuru, **Essentials of Microservices Architecture Paradigms, Applications, and Techniques,** Taylor & Francis; 1st Edition 2019.
- Poulton, Nigel. **Docker Deep Dive**. JJNP Consulting Limited, 2018.
- Nickoloff, Jeff. And Kuenzli, Stephen. **Docker in action.** Manning Publications Co., 2019.
- Nickoloff, Jeff. **Docker in action**. Manning Publications Co., 2016.
- Miell, Ian, and Aidan Hobson Sayers. **Docker in practice**. Manning Publications, 2019.
- Miell, Ian, and Aidan Hobson Sayers. **Docker in practice**. Manning Publications, 2016.
- Kane, Sean P., and Karl Matthias. **Docker: Up & Running: Shipping Reliable Containers in Production**. O'Reilly Media, 2018.
- Chelladhurai, Jeeva S., Vinod Singh, and Pethuru Raj. **Learning docker**. Packt Publishing Ltd, 2017.
- Schenker, Gabriel Nicolas. L**earn Docker-Fundamentals of Docker 19. x**. Packt Publishing, 2020.
- Richards, Mark. Ford, Neal **Fundamentals of Software Architecture** (O'Reilly). 2020.
- Ingeno, Joseph. **Software Architect's Handbook**. Packt Publishing Ltd, 2018.
- Michael P. Papazoglou, **Web services: Principles and Technology**, Pearson Education Limited, second edition, 2012
- Michael P. Papazoglou, **Web Services and SOA: Principles and Technology**, 2nd Edition. Prentice Hall, Pearson Education, 2011.
- Douglas K. Barry, **Web Services, Service-Oriented Architectures, and Cloud Computing: The Savvy Manager's Guide,** Morgan Kaufmann; 2nd Edition, 2013.
- James Bean **SOA and Web Services Interface Design: Principles, Techniques, and Standards**, Morgan Kaufmann Publishers 2010 (chap1,2,3,10)
- Steve Graham and Al. **Building Web Services with Java: Making sense of XML, SOAP, WSDL, AND UDDI**. Second Edition Sams Publishing; 2nd Edition, 2004.
- Bogunuva Mohanram Balachandar. **RESTful Java Web Services : A pragmatic guide to designing and building RESTful APIs using Java**. Packt Publishing, third edition 2017.

Assignments

Every lecture will be subject to : Homework **ToDo**, book Chapters/Papers **ToRead**, and Labs **ToCode**.

- **ToDo** are essential for understanding lecture content and doing in-class labs. e.g., prerequisite tools installation, platform learning... They must be done before attending the lecture.
- **ToRead** are book chapters and papers to read before attending course. They are subject to quizzes and sum-up to be done during the coming lecture.
- **ToCode** are Labs to be submitted after the lecture. Submission process and deadlines are to be defined (TBD) with every Lab.

Grading

This course will consist of Lab assignments, quizzes, lectures, chapter readings and sum-up, presentations and discussions, – each will contribute to your grades.

- Chapters reading, sum-up, discussion and quizzes : 25% (**ToRead**)
- In-class Labs : 25% (**ToCode**)
- At-home Labs : 50% (**ToCode**)

Assignments Submission rules

- Assignments will be due at **11:59PM** on the day of the deadline.
- Chapter/paper reading summaries (if requested) are due at **11:59PM** the night before the lecture.
- Late summaries/submissions will **not** be accepted.
- Appropriate information on submission tool will be provided later.
- Every chapter/paper reading might be subject to a quiz.
- Students are advised to read lecture slides content and download them *before* the lecture.
- Each **un**submitted Lab assignment (**Tocode**) results in a loss of **5%**.

Course Attendance Policy & Guidelines for Absences

- Each student shall adhere with the following lecture attendance rules :

  Before lecture time :

  - [x] Download lecture material files.
  - [x] Download Lab assignments files and folders.
  - [x] Download and install @home necessary software and tools.
  - [x] Download and read assigned book chapters and papers before attending lecture.

  During Lecture :

  - [x] Don't be late. No more than **10** minutes delay will be accepted to attend lecture.
  - [x] As major part of the lecture will require an internet access, - specially when we will reach the service invocation and Docker platform chapters, students are advised to have their own laptop and internet access as some parts of the course require heavy internet access.

Class website

https://neilabenlakhal.github.io/