Institute of Service Science, National Tsing Hua University

Service Oriented Architecture

Fall 2021

Course Duration:	Sep 2020 – Jan 2021	*Instructor:*	Soumya Ray (soumya.ray@iss.nthu.edu.tw)
Class Time:	Mondays, 9am-12pm	*Assistant:*	TBA
Facebook Group:	facebook.com/groups/ISS.SOAD		

How do companies like Netflix and Airbnb constantly innovate on their large online systems? Innovation requires information systems that we can confidently extend, redesign, retest, and redeploy. In this class, you will learn the best practices to develop information systems that are built for scale and innovation.

This class is ideal for students with strong programming backgrounds who want to one day be:

- *Tech leads:* lead highly productive software development teams
- *Software entrepreneurs:* create highly scalable and extensible applications for a startup
- *IT product managers:* use the latest programming practices to lead and mentor a sophisticated IT team

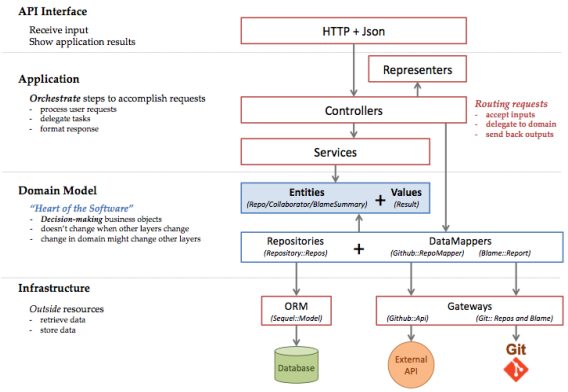
What techniques will I learn?

Learn architectural and refactoring techniques to create systems that are easy to maintain and extend:

Domain Driven Design: layered architecture that separates domain logic from infrastructure and presentation

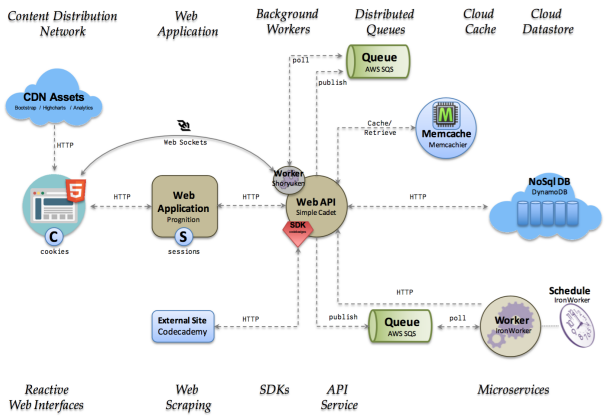
Functional flow: monads, transactions, railway-oriented programming to explicitly manage errors

Automated Testing: unit tests, integration tests, and user acceptance tests to ensure correct system behavior



What tools will we use?

You will use tools favored by cutting-edge startups:



Cloud Services: Amazon AWS, Codeship, and Heroku

Distributed Architecture: APIs, microservices, Docker

Data Stores: Relational & key-value databases; local, hosted, and cloud-based data stores

Reactive UX: background jobs, message queues, web sockets, reactive front-end elements

Responsive UI: mobile-first, responsive web-interfaces

DevOps: Continuous integration, Containerized deploys

Version Control: git, Github Flow, rebase, pull requests

How much coding should I already know?

You should be able to create a basic application using:

Object-oriented programming: any OOP language such as Javascript, Python, PHP, Java, Ruby, C#, C++, etc.

Basic web page design: HTML + CSS

Database design: Relational design + ERD + SQL

This class will use Ruby and the AWS platform,
but the skills you learn will apply to any tech stack

The tools and techniques you will learn are
compatible with any major modern language

The platforms and tools you will use are
free up to the requirements of small businesses

INTRODUCTION

Service Orientation
Service Providers
Service Architecture
Meet Your Tools

Architecture Matters
IT systems become slow and as they grow in complexity. How are leading IT companies managing their code? Let's learn how to improve innovation and bring joy.

Coding Style
Programming Tools
Idiomatic Coding
"Good" Code

Writing Confident Code
We are all afraid to show our code. How can we become confident at coding and be proud of what we make? Let's start by following the best practices of clean coding.

(*evening workshop*)
Programming Workshop
Git Version Control Workshop

BACK TO BASICS

Refactoring & OOP
Refactoring Code
SOA Startup Teams!
Revisiting Object-Oriented Programming

Quantifying Code Quality
How should we structure to solve small problems? Let's quantify what it means to have good code using metrics. Then, let's rethink object-oriented programming.

Web APIs
Serialization/Deserialization
API Exploration
API Library

Getting down with data
We will develop our first library to extract, transform, and load data from interesting APIs.

Testing and Tooling
Testing Tools
Library Tools
UPSTART: Elevator Pitch

Fast and Reliable Tests
Testing applications with external resources is unreliable and inconvenient. Let's use stubs to capture and replay data flows. And let's upgrade our code management tools.

BACK-END ARCHITECTURE

Designing a Web API
Design Patterns: Data Retrieval
Web Architecture: Overview
Web API Application: Roda

Layered architecture
Even a simple modular architecture will make your code clean, will make you more confident to change your code, and will let you sleep better at night.

Object Relational Architecture
API Web Service
Repository Pattern
Layered Architecture

Data versus Domain
Relational data is different from our coding objects. Let's learn to properly manage object-relational mapping, and see how it fits into our growing architecture.

Service Refactoring
Business Logic
Fat Controllers
Representers and Services

Business logic is the heart of your service
Code gets complicated and ugly when it tries to do more than one thing. We need to model our core logic and protect it from other concerns.

Domain in Context
The Heart of our Software
Entities and Values
Bounded Contexts

What's your value proposition?
The value of our software comes from its domain logic. Let's make our code match our business language.

FRONT-END ARCHITECTURE

Presentation Layer
Deploying our API
Web Application
Bootstrap and Slim

Real artists ship!
Let's see how to design and build an interface responsibly. And then let's deploy our system early and often using continuous integration, delivery, and deployment.

Testing and Refactoring Views
Acceptance Tests
Refactoring Views
Prototype Sharing

Testing user experience
Routine user-experience testing shouldn't delay shipping. Let's automate the practice of testing what users see, by writing and simulating user stories.

DISTRIBUTED ARCHITECTURE

Concurrency
Page Objects
Controllers and Domain
Service Concurrency

Multitasking 1
All this network input/output is making our service slow. We've got to learn to multitask: let's start new things while waiting for machines to respond.

Parallel Processing
Parallelism
Message Queues
Background Workers

Multitasking 2
Some tasks need a long time to finish. Can we start serving new users while still processing old tasks? Let's use job queues and background workers to handle all our tasks.

Reactive Architecture
Reactive Architecture
Web Sockets
Faye for Ruby/Javascript

Show me some progress
Users hate waiting. Let's give them a real-time peek at what our system is doing. But HTTP may not be enough: time to bring out websockets.

Scheduled Tasks
Scheduled Workers
Docker Images and Containers
Docker Deployment

Make it snappy!
Can we do what the user wants before they ask for it? Let's schedule regular data analytic tasks that we expect users will want.

PRODUCT LAUNCH

Code Review

Show and Tell
UPSTART: Presentations
Why Architecture Matters
What's Next

It's show time!
Let's see what amazing things we have made. See how confident and proud we can be of our code! And let's talk about what we will accomplish in future.

How will I be graded?

Individual Development Effort:	80%	(from your project contributions on Github)
Class and Online Discussion:	20%	(you must share and discuss your thoughts in-class or on Slack)