COMS W4156 Advanced Software Engineering (ASE)

October 5, 2021

shared google doc for discussion during class

Individual Mini-Project part 2 and part 3 Deadlines Extended

Three parts:

1. Implementing a simple game graded - contact grader for regrade (or contact Head IA - Claire - if you don't know who graded), rubric posted here
2. Testing the game past due, but you can (re)submit until tonight with no penalty
3. Saving game state nominally due tomorrow, but you can submit up to October 9 with no penalty

Waterfall vs. Agile

A student asked when to choose waterfall instead of agile?

1. Any software that involves government procurement processes 😉
2. When you need to get it right the first time
   a. Software difficult or impossible to change after deployment (e.g., spacecraft)
   b. Software that could cause serious damage, injury or death (aka safety-critical software)

How do you go from Use Cases to Code?

Just start coding

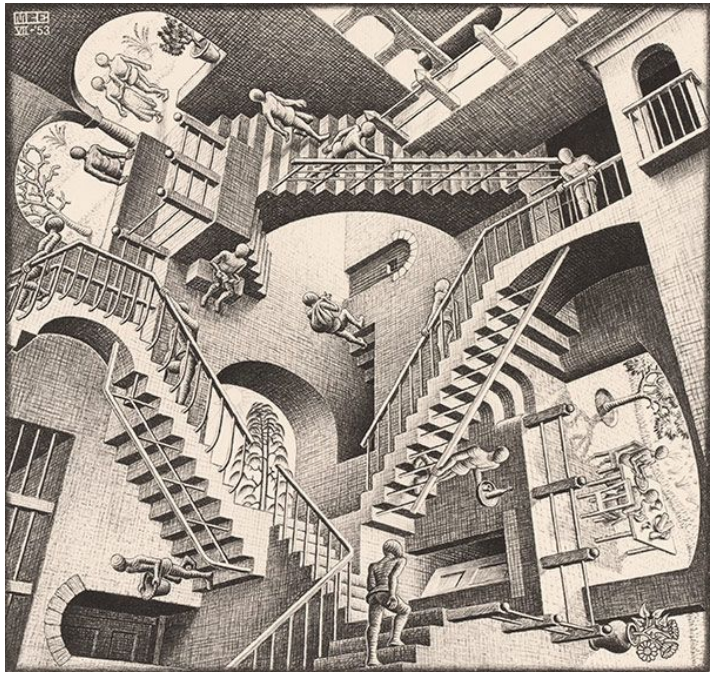
Outsource (Buy vs Build)

Design (and Architecture)

Recall that use cases (and/or user stories) should initially only capture features intended for current iteration (or possibly upcoming release)

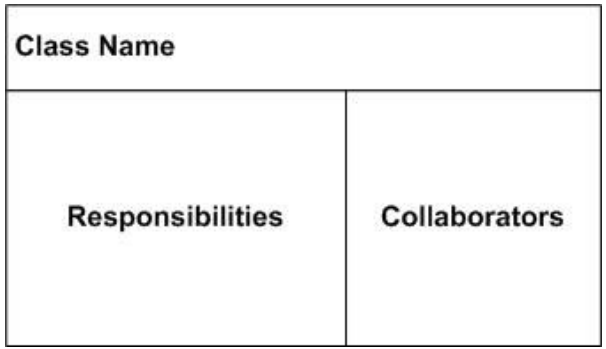
If there is too much to do, prioritize and push off lower priority to later

Example design: good art, bad code



Agile Design: CRC = Class Responsibilities Collaborators

Like user stories, CRC can be written on index cards (probably 4x6 instead of 3x5 since there's likely to be more text), so often known as "CRC cards"



Each *class* is a thing, entity or object

Each *responsibility* is some task the entity needs to do or data it needs to track

Collaborators are other classes the entity interacts with

CRC focuses on the *purpose* of each entity rather than its processes, data flows and data stores

Example

Find Classes

Start with the set of use cases

Find the *nouns* other than the actor (or user role) - classes should be the entities that “do” actions, not the actors who initiate actions

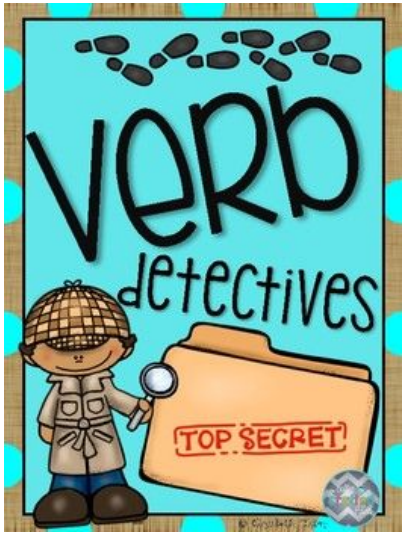
Convert plurals to singular, merge synonyms

Beware adjectives - may mean a whole new class,
e.g., “car” vs. “toy car”

Beware hidden nouns, e.g., passive voice
“the thing is activated” = “SOMETHING activates the thing”

Some nouns may be attributes of entities, not themselves entities, e.g., “the radius of a circle” - here the circle is an entity and the radius is an attribute of the circle, it has no meaning as a standalone entity

Find Responsibilities



Find the *verbs* (some may be CRUD)

Say WHAT gets done, not HOW it gets done

Turn passive verbs into active verbs
“the thing is activated” = “SOMETHING activates the thing”

The classes and responsibilities create a vocabulary for discussing the design

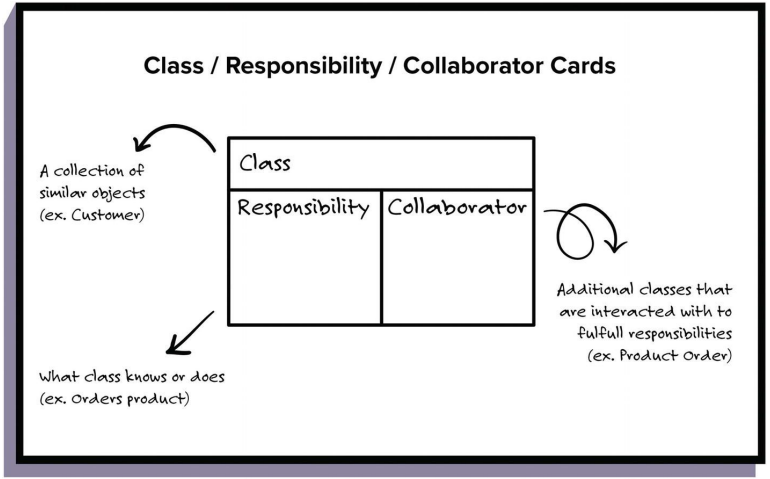
Find Collaborators

Find any relationships or associations among entities - not necessarily symmetric, can be one-way

Any entity that a class needs to know about, communicate with, contain, control, or that otherwise help it perform one or more responsibilities that it cannot perform on its own

If that entity does not already have its own CRC card, make a new card

If that entity does not already have a responsibility to do something or provide something on behalf of the class at hand, add that responsibility (and any additional collaborators it needs)



CRC Design Process

Develop a set of CRC cards for the current set of user stories / use cases (current iteration or upcoming release)

CRC does not need an object-oriented programming language - CRC can stand for *Component* Responsibilities Collaborators

Walk through the scenarios (steps and flows) of
each use case and *animate* the CRC cards:
move index cards on a table or sticky notes on a board

Make sure everything is a responsibility of some class, and figure out what each class needs from other classes to fulfill its responsibilities

There is no Global Master

When “the system” does something, that means some class in the system does it

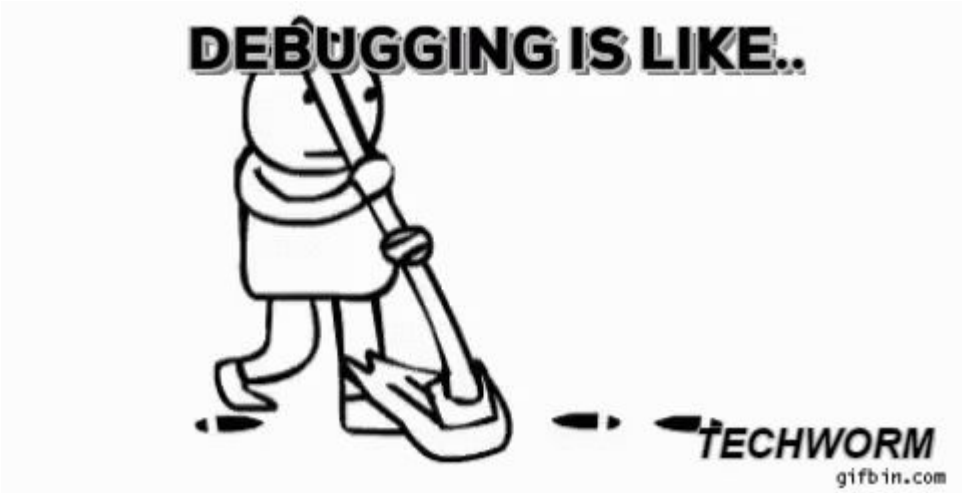
There should not be any global master that does everything and/or is related to everything else

Give up global knowledge of control and rely on local knowledge of objects to accomplish their tasks

Debug the CRC cards

- Remember classes are nouns that “do” operations, not the actors who initiate operations
- Split classes with too many responsibilities
- Combine classes with too few responsibilities
- Remove redundancy

[CRC Card Maker]

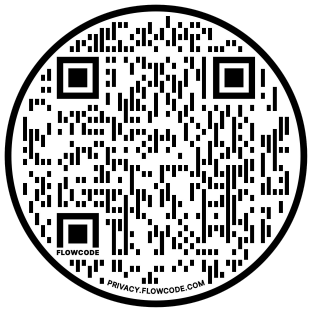


In-Class Exercise

Again your goal is to develop CONTAIN, **CON**tact **T**racing for instruction **A**ss**I**sta**N**ts, which does “contact tracing” based on IA (instruction assistant) connections. An IA for xxx course is a contact of all the other IAs for xxx as well as for every student who takes xxx. Since an IA is also a student, they are a contact for all other students in every course they take as well as for all the IAs of those courses.

To limit false positives, CONTAIN should find connections only among students who were actually in the same classroom at more-or-less the same time, e.g., including those who were leaving the classroom after a class while others were entering for the next class. Your code can access APIs from SSOL, CANVAS, a drone-based attendance-recording service, and any other API you consider useful. All the students have QR code stickers on their foreheads. One possible error condition is multiple students have duplicates of the same QR code.

- ➢ Write some CRC cards for CONTAIN



Blank paper available

timer..

To get from Requirements to Coding, need Design and Architecture

See September 14 lecture and September 14 lecture notes about software (and building) architecture - TL;DR: choose among well-known architectural styles, don't reinvent the wheel

Let's consider some possibilities for how services might interact with their clients:

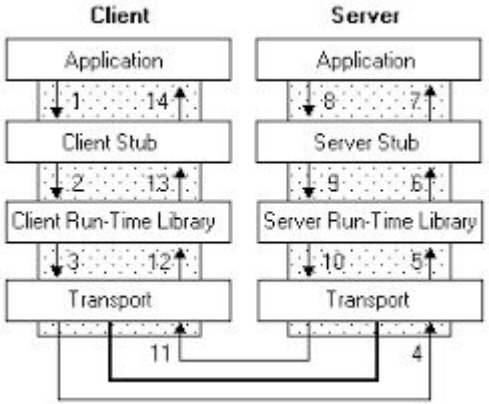
- Client-server or N-tier (request-response)
- Broker or gateway (mediated request-response)
- Publish-subscribe or event-driven (push or pull)
- Service-oriented architecture (SOA) or microservices (services usable by other services)
- …

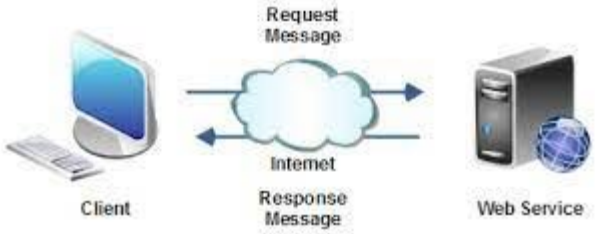
There are other options, and lots more technical details than I'm going to cover. You can use a framework, tool, library, etc. without knowing how it works. To learn how it works, and to learn how to implement the distributed computing mechanisms yourself, take 4113 - Spring 2022 planned

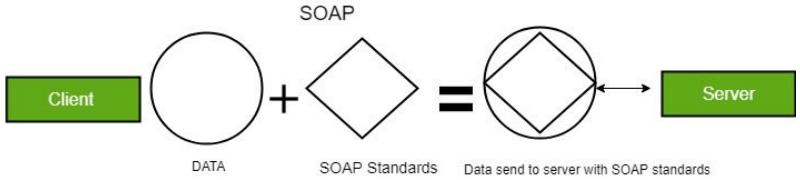
Client-Server Request-Response

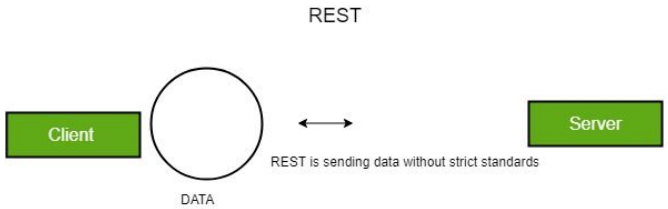
Web service messages: send a message over HTTP/HTTPS, wait for response message

Remote procedure call (RPC): make what looks like a normal function call, wait for it to return, sent over HTTP/HTTPS or another protocol



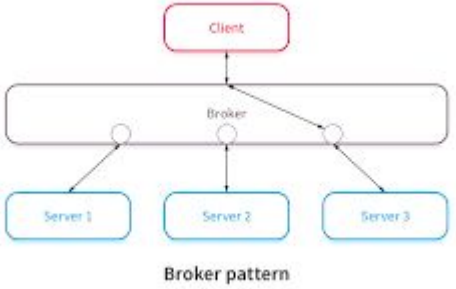


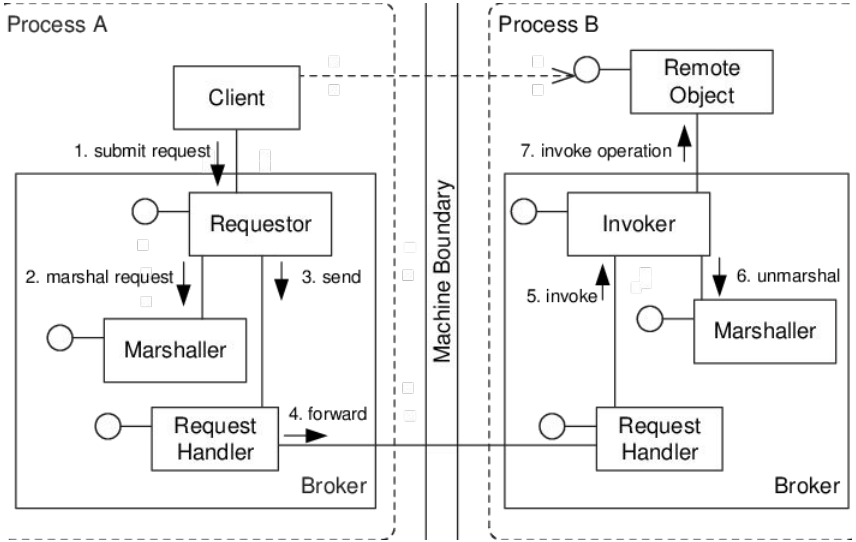


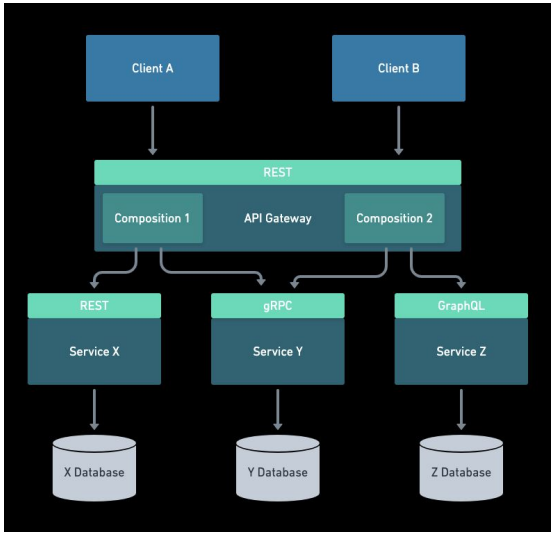


Broker Mediated Request-Response

From client perspective, essentially same as client-server except the client does not know where the server is and sends instead to a broker or gateway





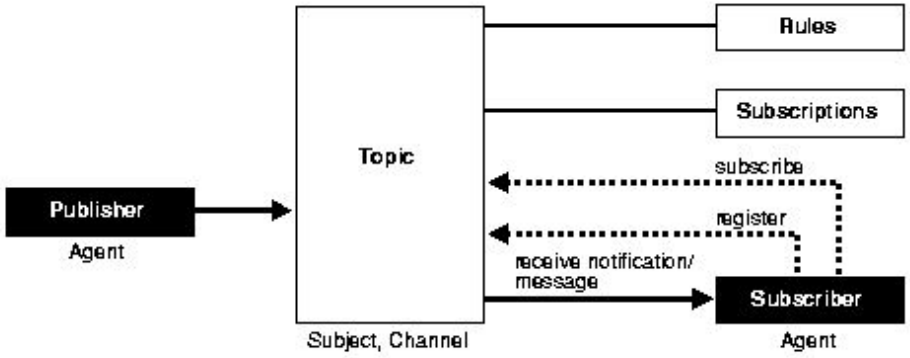


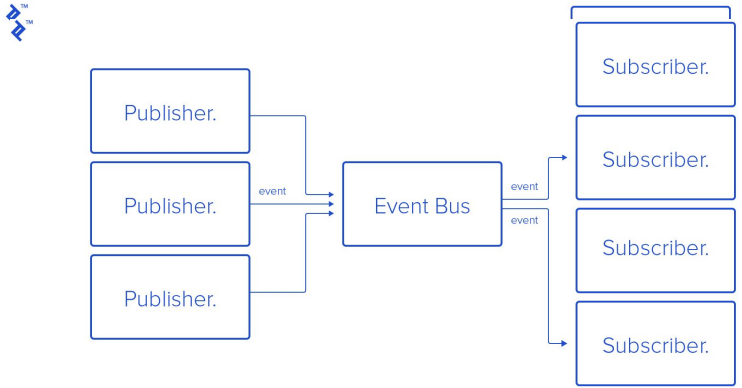
Publish-Subscribe

Twitter for software, not humans

Push - publisher or intermediary sends message or invokes callback for “interested” subscriber

Pull (or poll) - Subscribers keep checking



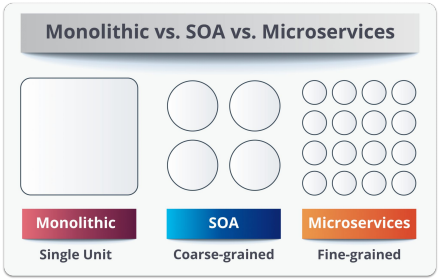


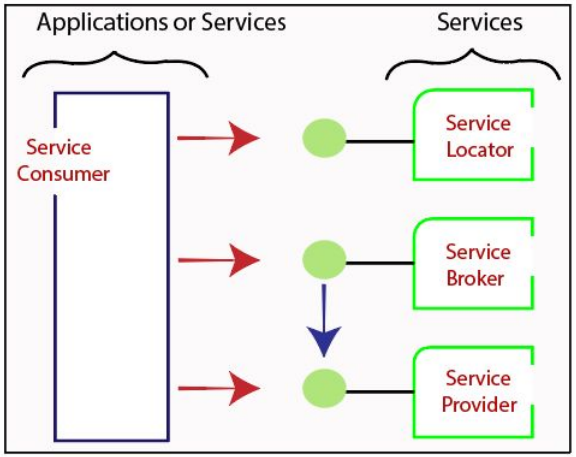
Service-Oriented

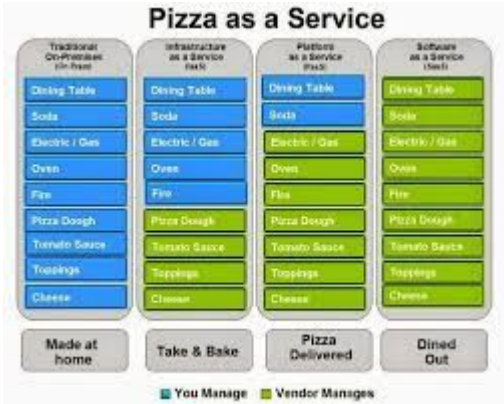
Services interact with other services and with applications over the network

Any of the previously mentioned architectures could include SOA and/or microservices elements

I don't think there's much difference between SOA and microservices, other than "SOA" sounds stodgy and "microservices" sounds cool, but you can find plenty of comparisons online







Team Project

Preliminary Project Proposal due October 13

Your service has to implement some API, but it does not need to be a REST API, and could involve interacting with other services, RPC, mediation, callbacks, events, etc. - but no GUI

No GUI does not necessarily mean no UI, there might be an administrative console using CLI (command line interface), just make sure your service cannot be mistaken for an “app”

➢ Some students still have not submitted T0 even though other students think they’re team members...

Individual Mini-Project part 2 and part 3 Deadlines Extended

Three parts:

1. Implementing a simple game graded - contact grader for regrade (or contact Head IA - Claire - if you don't know who graded), rubric posted here
2. Testing the game past due, but you can (re)submit until tonight with no penalty
3. Saving game state nominally due tomorrow, but you can submit up to October 9 with no penalty