SENG3011 Implementation Workshop

More on REST services

Outline

- Programmable Web
- Resource Oriented Architecture
  - REST (video https://www.youtube.com/watch?v=7YcW25PHnAA)
  - ROA Properties
  - Service interactions
  - Service design issues

The Concept of Programmable Web

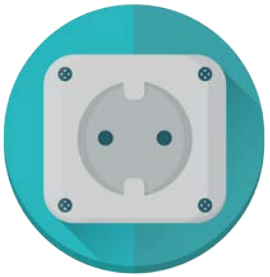
- The Programmable Web use the same technologies and communication protocols of the WWW
- Difference:
  - The data is not delivered necessarily for human consumption
  - A client can be implemented using any programming language
- Technologies
  - Services and APIs
  - Transport protocol: Hyper Text Transfer Protocol (HTTP)
  - Clients: Browser, Java, Web API, …
  - Data serialization languages

Web Services

- 'logical units with clearly defined interfaces(API):'
  - What functionality they perform
  - Which data formats they accept and produce
- They are application independent
- Services can be used by other services and applications
- Web services are not prepared to human consumption (in contrast to websites).
  - Web services require an architectural style to provide clear and unambiguous interaction (clearly defined interfaces).

Web API

- Application Programming Interfaces
  - A good analogy is the electricity wall socket
- Endpoints addressable over the Web are called Web APIs.
- How the service is exposed:
  - Protocol semantics
  - Application semantics
- We frequently use Web API instead of Web services but they are not the same
- We will be focusing on the RESTfull Web API



- Service: Electricity
- Conforms to specs: 220V, 60Hz …
- Fitting patterns are defined
- Through the standard interface all connecting equipment (consumers) work
- A layer of abstraction

Market Impact

- Making functionality available over the web changed the way software functionality delivered.
- If you needed a CRM functionality in 1990s you had to invest in hardware, software, the CRM experts, training …
- Today's CRM providers like Salesforce use cloud to deliver the functionality.
  - Multi-tennacy – sharing common infrastructure among customers.
  - Using web browsers was the norm to access this functionality
  - Today customers are granted API level access
    - Non salesforce applications can easily use the services.
- Thousands of companies are changing their strategies toward delivering functionality through Web APIs:
  - https://www.programmableweb.com/apis/directory is a good source

Representational State Transfer (REST)

- A way of providing interoperability between computer systems on the Internet.
  - REST-compliant Web services allow requesting systems to access and manipulate textual representations of Web resources using a uniform and predefined set of stateless operations.
- An architectural style of building networked systems
  - a "design guideline" for building a system (or a service in our context) on the Web
  - defines a set of architectural constraints in a protocol
- REST is built on standards:
  - HTTP, URL, XML/HTML/JPEG/ … (resource representations)
  - text/xml, text/html, image/gif, image/jpeg, … (MIME Types)
- REST itself is not an official standard specification

What is a Resource

- A thing that users want to create a link to, retrieve, annotate, or perform other operations on.
- A resource:
  - is **unique** (i.e., can be identied uniquely)
  - has at least one **representation**,
  - has one or more **attributes** beyond ID
  - has a potential **schema**, or definition
  - can provide **context**
  - is reachable within the **addressable** universe
- collections, relationships (structural, semantic)

Representational State Transfer?

- Web is comprised of resources
- UNSW can define SENG3011 as a resource
  - Students can access this resource through a URL:
  - http://www.unsw.edu.au/course/SENG3011
- Representation is returned SENG3011.html –
  - The representation place client application in a **state**
  - Client can access another resource in COMP3392.html
  - The new representation places client in another **state**
- The client application **transfer states** with each resource **representation.**

Resource Oriented Architectures

- ROA:
  - Architecture for creating Web APIs that conforms to the REST design principles
  - Base technologies: URLs, HTTP and Hypermedia
- Web Services with a ROA architecture are called RESTful Web Services (Restfull Web APIs)
- HTTP requests are used to manipulate the state of a resource

URI: Identifies the resource to manipulate

http://www.unsw.edu.au/course/SENG3011

HTTP method: The action to be performed to manipulate the resource

ROA Properties

- Addressability
- Uniform interface
- Statelessness
- Connectedness

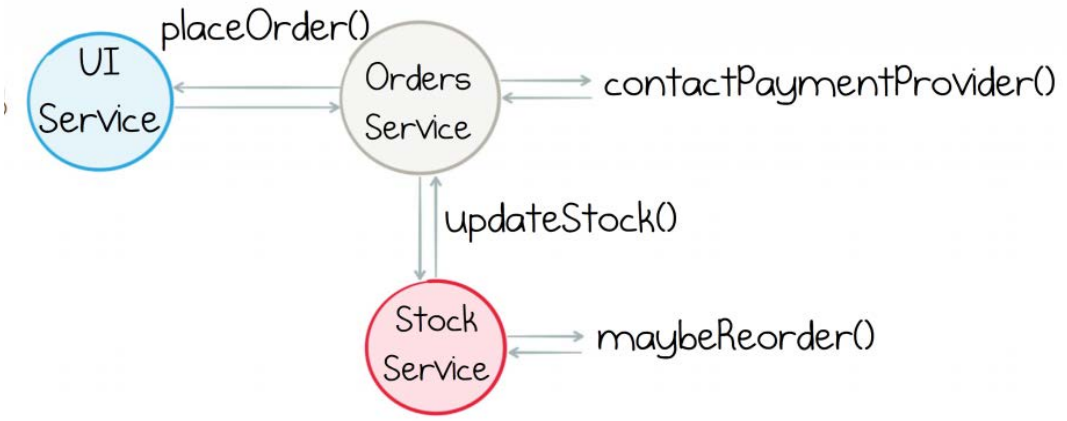
Synchronous

- More suitable:
  - where real-time interaction with minimal delays is needed,
  - where subsequent actions are dependent on the response received for the previous message transferred,
  - further actions need to be performed in sequential manner.
- Example:
  - ATM machine need to interact with the back-end system to check the available balance.

Asynchronous

- More suitable:
  - where systems have long running jobs and there is no need of real-time responses.
  - when you need low latency – blocking a call may slow the system
- Example:
  - An ERP system needs to publish some information so that any interested parties can subscribe to that and get the updates.

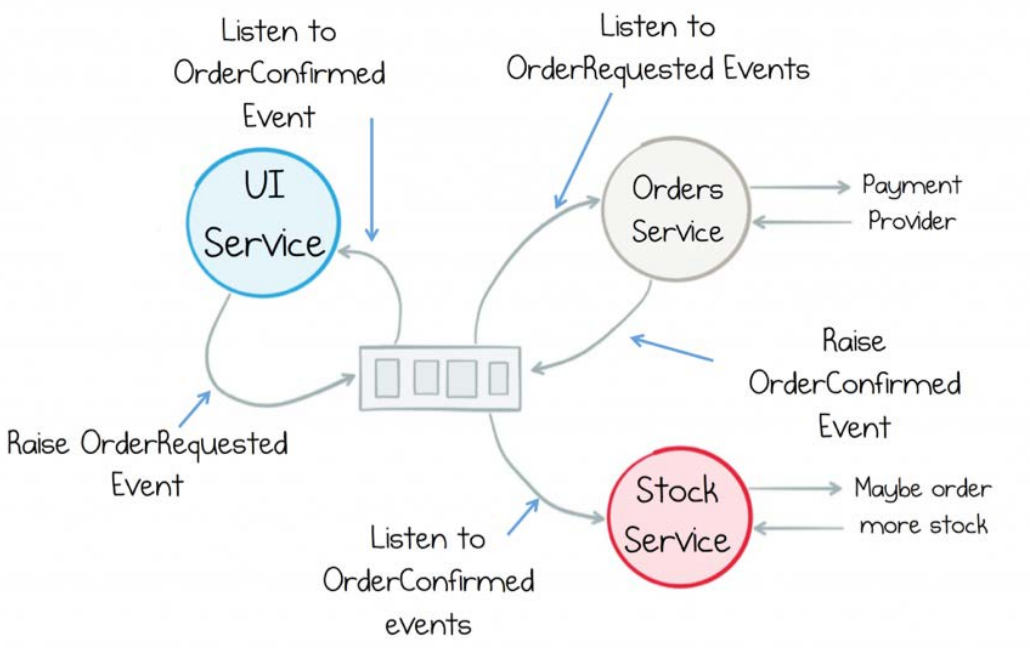
Request/Response Collaboration



1-Customer orders an item
2-Payment is processed
3-The system check the availability and the need for reorder

- Well aligned with synchronous communication
- For asynchronous applications adaptation is required:
  - Start the operation
  - Register a call back
    - ask server to notify when the operation complete

Event based collaboration

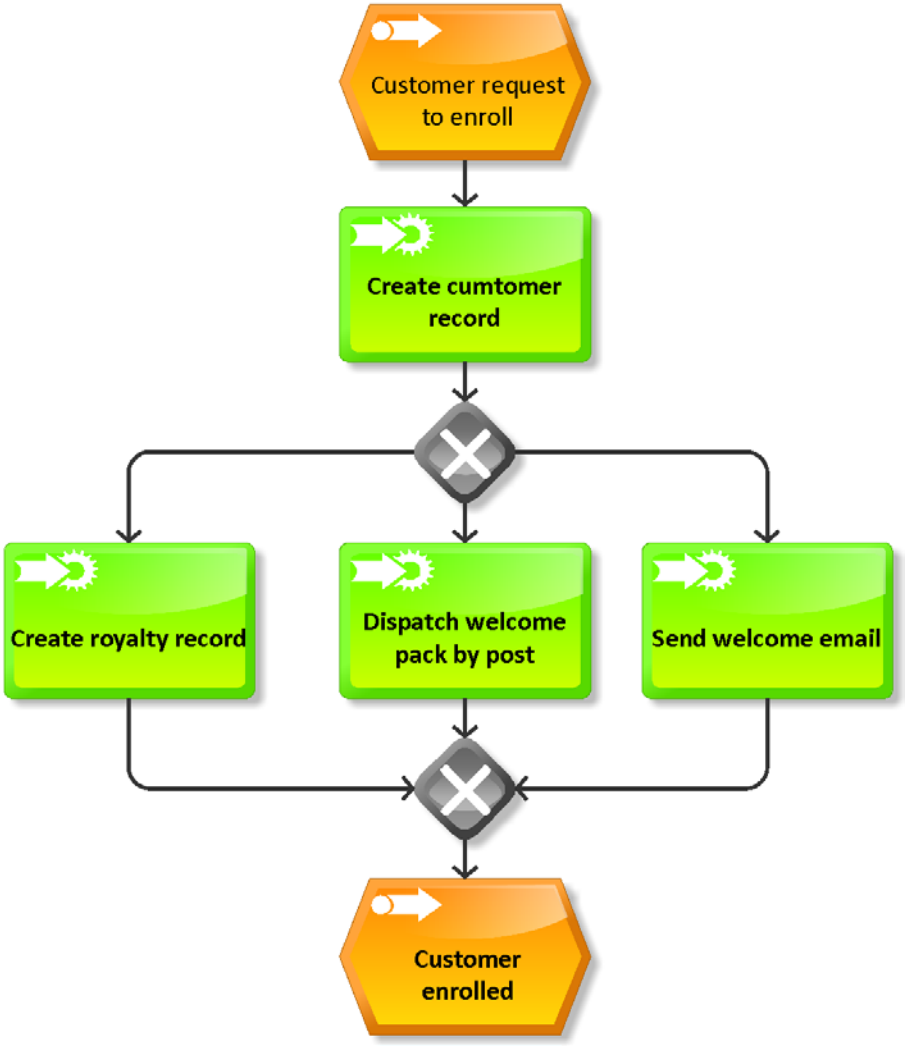


- The UI Service raises Order-Requested event
- Orders Service and the Stock Service react to the raised event.
- Order service raise Order-Confirmed event
- UI Service reacts to Order-Confirmed

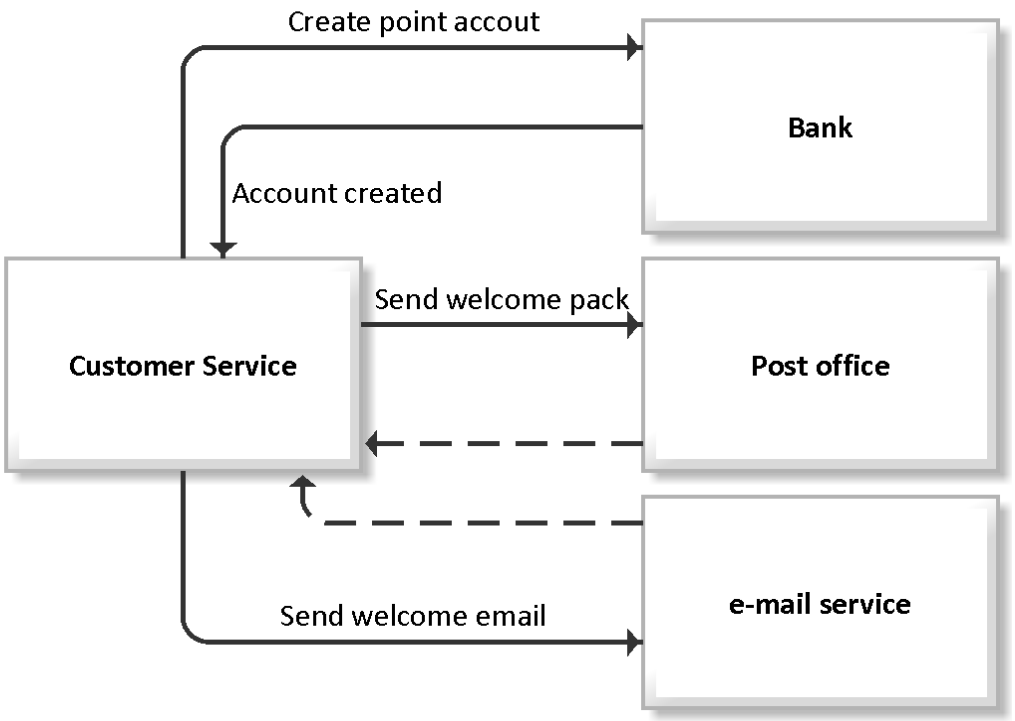
- Process announce what happened
- Other services decides what to do
- Business logic is distributed
- Highly decoupled – can add new services easily.

Processes That Spans Across Services

Customer enrollment



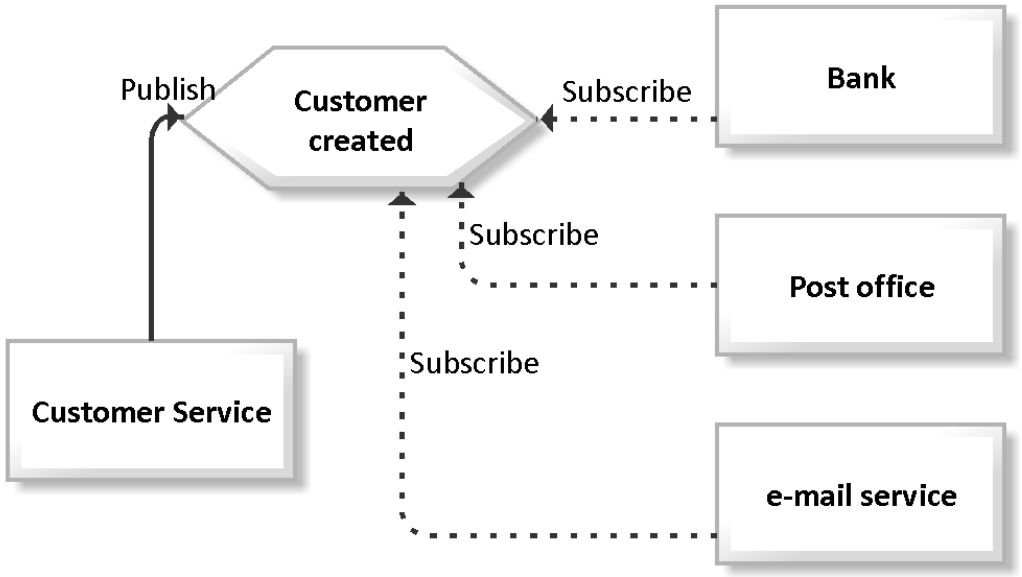
Orchestration



- Create a central control mechanism within:

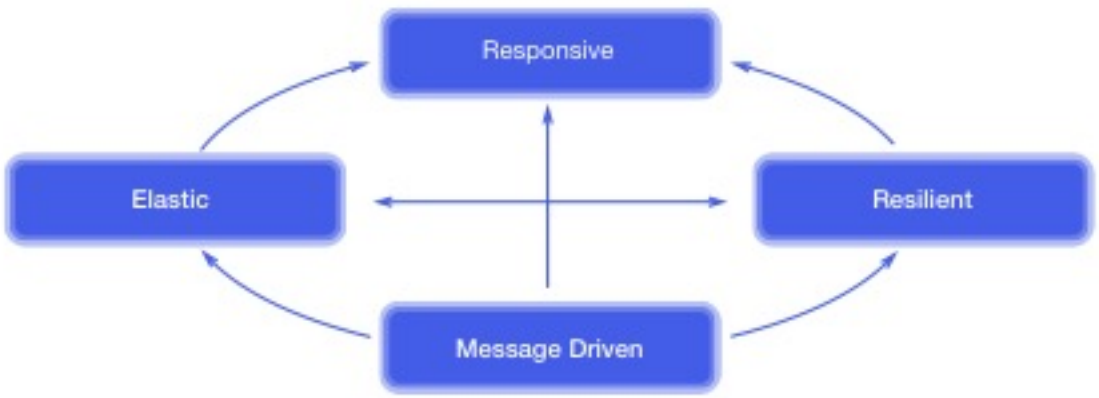
  CustomerService
- Once the process initiated CustomerService send request to other services.
- We can model into code or use BPM software.
- - Tightly coupled
- - High cost to change
- + Can monitor the status of the process.

Choreography



- Customer Service created the event.
- All services subscribe to this event react to it.
- + Loosely coupled
- + Easy to change
- - Additional work is needed to monitor the status of the process.

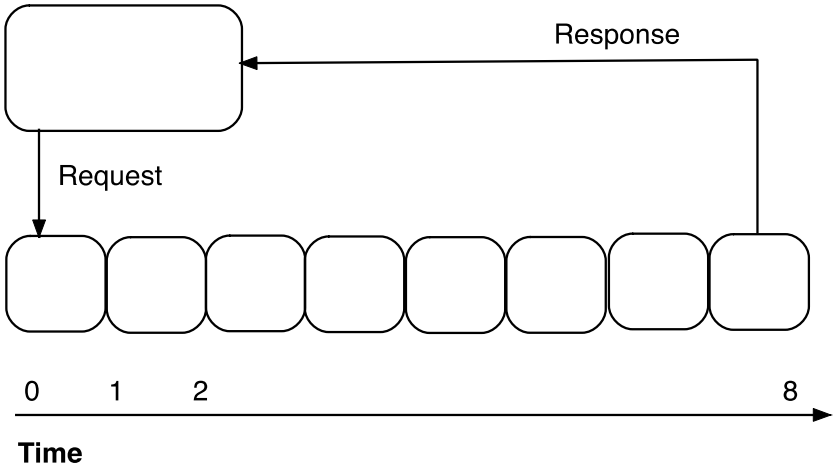
Reactive Systems

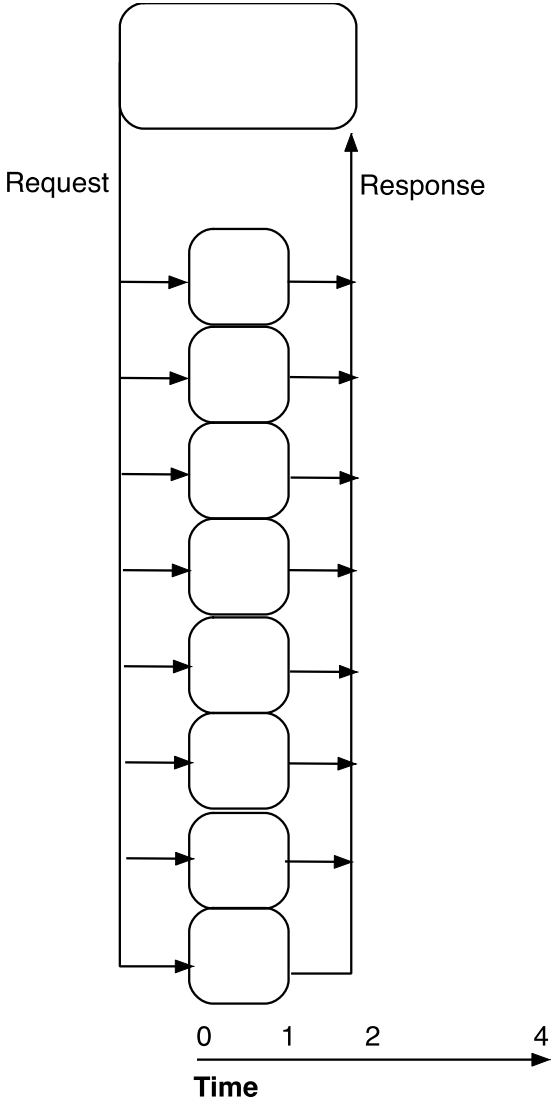


- Systems that are* :
  - Responsive,
  - Resilient,
  - Elastic and
  - **Message Driven**
    - Asynchronous, nonblocking message-passing that establish a boundary between components…
    - that ensures loose coupling, isolation and location transparency.

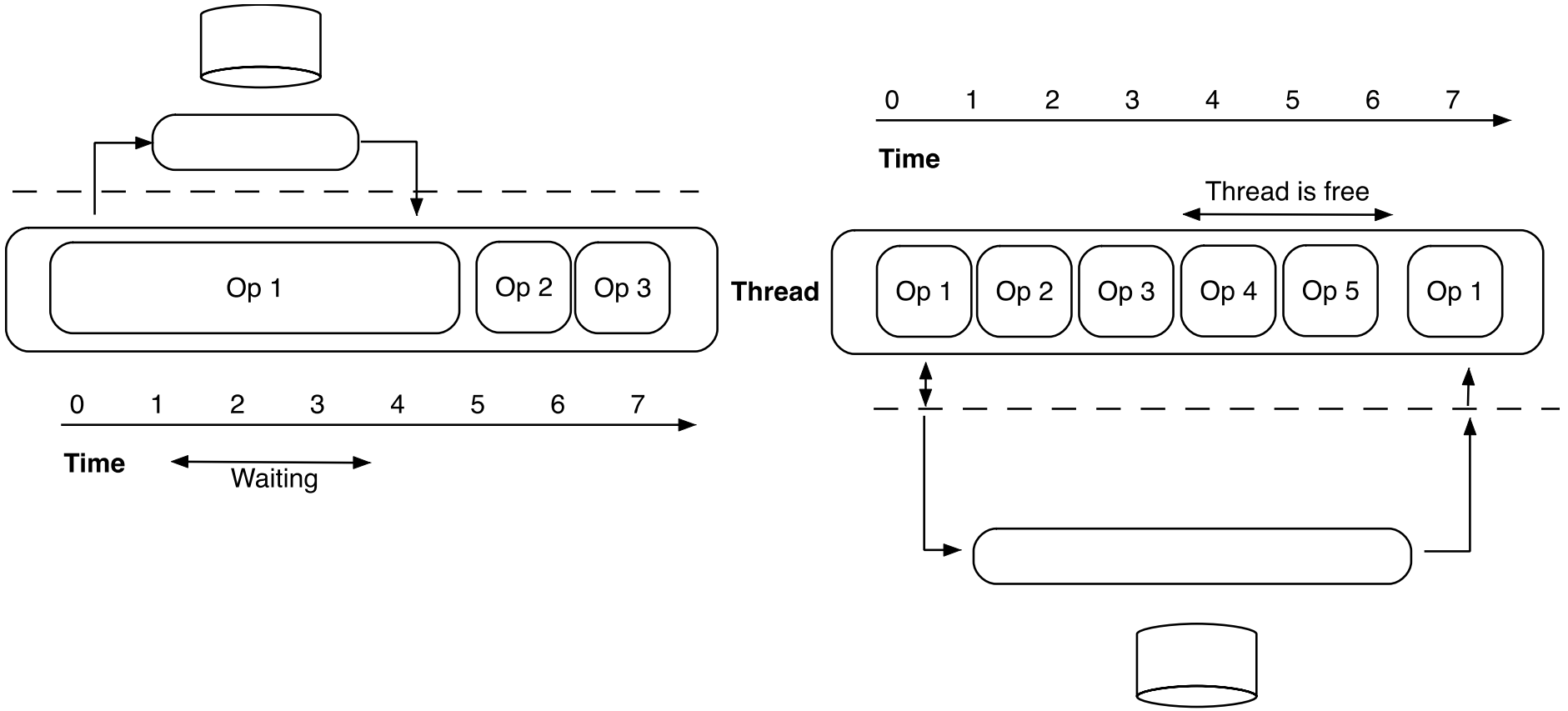
* The Reactive Manifesto

Sequential vs Asynchronous Execution





Synch Blocking vs Asynch Nonblocking



Separation of Stateful from Stateless

- Stateless:
  - Deals with behavior, pure business logic
  - Sending an email
  - Displaying the fuel consumption for the moment
  - HTTP protocol
- Stateful:
  - Deals with keeping records of things
  - Expecting an acknowledgement for the email sent
  - Displaying the average fuel consumption for a period.
  - FTP protocol

Scaling up

- Decoupling behaviour from state enable us to scale up the stateless processes.
- Scaling up stateless processes is easy.
  - You can run KM to Miles Conversion on multiple nodes easily
  - Various platforms exists: AWS Lamda is a popular exaple
- Scaling up the stateful part is difficult
  - The aggregate is the only strongly consistent truth
  - Single active instance can run at a time
  - Usually scaled up by using active/passive availability clusters
    - (Establishing fully redundant instances of nodes, brought online when its associated primary node fails)

Rethinking Persistence

- Before Databases, Accountants used to keep all the transactions that accured: in journals and ledgers.
- Sample: Transactions and a corresponding Journal

Date	Transaction
Jan 2 as	An amount of $36,000 was paid advance rent for three months.
Jan 3 purchase $80,000. The recognized payable.	Paid $60,000 cash on the of equipment costing remaining amount was as a one year note
Jan 4 costing	Purchased office supplies $17,600 on account.
Jan 13 customers in cash.	Provided services to its and received $28,500

Date	Account	Debit	Credit
Jan 1	Cash	100,000	
	Common Stock		100,000
Jan 2	Prepaid Rent	36,000	
	Cash		36,000
Jan 3	Equipment		80,000
	Cash	60,000	
	Notes Payable		20,000
Jan 4	Office Supplies	17,600	
	Accounts Payable		17,600
Jan 13	Cash	28,500	
	Service Revenue		28,500

Journals vs Databases

- Journal
  - Show the complete history of the transactions
  - One never alter the journal
    - If an error is made it is compensated by a new entry into the journal
  - There is no concept of update-in-place (overwriting existing record with new data)
- Database
  - Show the current state of the data.
    - Diskspace was very expensive to depict all the history
  - SQL databeses use CRUD to eliminates redundancy by only depicting the current state of the data

What if we don't have the diskspace constraint?

Event Logging

- Events are stored in the order that they are created
  - It is a database of everything that has happened in the system.
- Time is a natural index
  - You can reverse back for any purpose
  - Debugging, Auditing …
- Event Sourcing – A pattern for event logging.
  - State change is captured as a new event to be stored in event log.
    - OrderCreated, PaymentAuthorized, EmailSent …
  - The aggregate can cache the dataset in memory (the latest state)
  - Event sourced aggregates use ‘Event Streams’ to publish events to other services.

Other Useful Design Patterns

- Back Pressure
  - A pattern for flow control
  - When we have fast producer and slow consumer
  - Consumer manages the flow by signaling producers
- Circuit Braker
  - A Finite State Machine – Closed/Open/Half-Open
  - The default state is Closed
  - When a failure detected it moves to Open State
  - When Open, it does not let any request to go through
  - After time-out, it moves to Half-Open state
  - In Half-Open, if the next request fails it goes to Open otherwise goes to Closed.

Organizations and Microservices

- Microservices are services modeled after a business domain
- Conwey's Principle:
  - Any organization that designs a system (defined more broadly here than just information systems) will inevitably produce a design whose structure is a copy of the organization's communication structure
- Information Systems Department of an Army:
  - How will the communication structure shape?
    - Command and control
  - Who will be the project manager?
    - The highest ranking officer
- A startup ? Will you give the same answers?

Comparing Amazon and Raytheon

- Amazon
  - The culture:
    - Small teams – two pizza teams
    - Teams owns the whole lifecyce of the systems
    - Like a tennis team
  - The Process: Agile
  - The product:
    - Amazon Web Services Platform – Have an array of services created and managed individually
- Raytheon
  - The culture
    - Large teams – Project based organization
    - Process owns the lifecycle
    - Like a cricket team
  - The Process: Well defined, Waterfall
  - The product:
    - Coyote UAS (Small, expendable, unmanned aircraft system) created and managed as a single system

References

- Slides prepared by Prof Onur Demirors
- Dr. Helen Paik's COMP 9322 Course handouts
- Richardson and Amundsen, RESTful Web APIs, O'Reilly, 2013
- www.programmableweb.com
- Richardson and Ruby, RESTful Web Services by, O'Reilly, 2007 (http://oreilly.com/catalog/9780596529260)