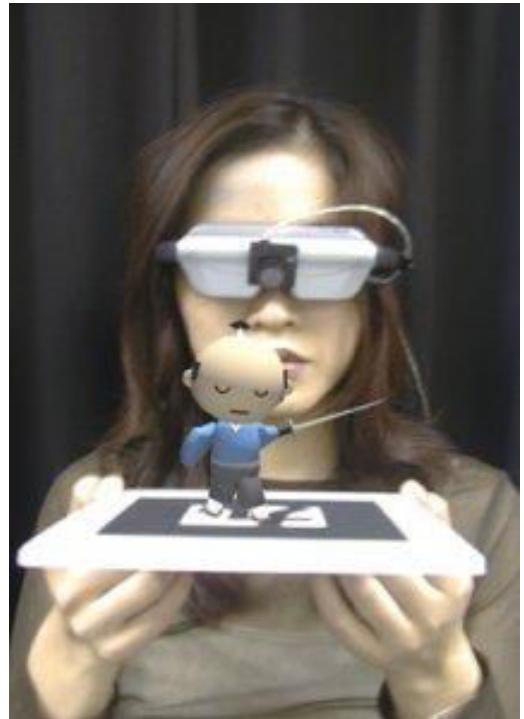


# Augmented Reality

**Modern Topics in Information Technology**  
**4<sup>th</sup> Year – Semester 1**  
**Lecture 02 - Part I**  
**By Udara Samaratunge**

# What is Augmented Reality?



# What is Augmented Reality?

- **Augmented reality (AR)** is a field of computer research which deals with the combination of **real-world** and **computer-generated** data.
- Ronald Azuma defines an augmented reality system as one that:
  - The **overlay of computer-presented information on top of the real world**
  - Combines real and virtual realities
  - Interactive in real time
  - Registered in 3D
  - Not the same as “virtual reality”

# Historical Background

- From 1957 to 1962, Heilig created cinema of visuals, sounds, vibrations, and smells called Sensorama.
- 1966 – Ivan Sutherland, head-mounted display
- 1975 – Myron Krueger, Video place
- 1989 – [Jaron Lanier](#) coined the term **Virtual Reality**
- 1992 – [Tom Caudell](#) coined the term **Augmented Reality**

# Mixing of Realities

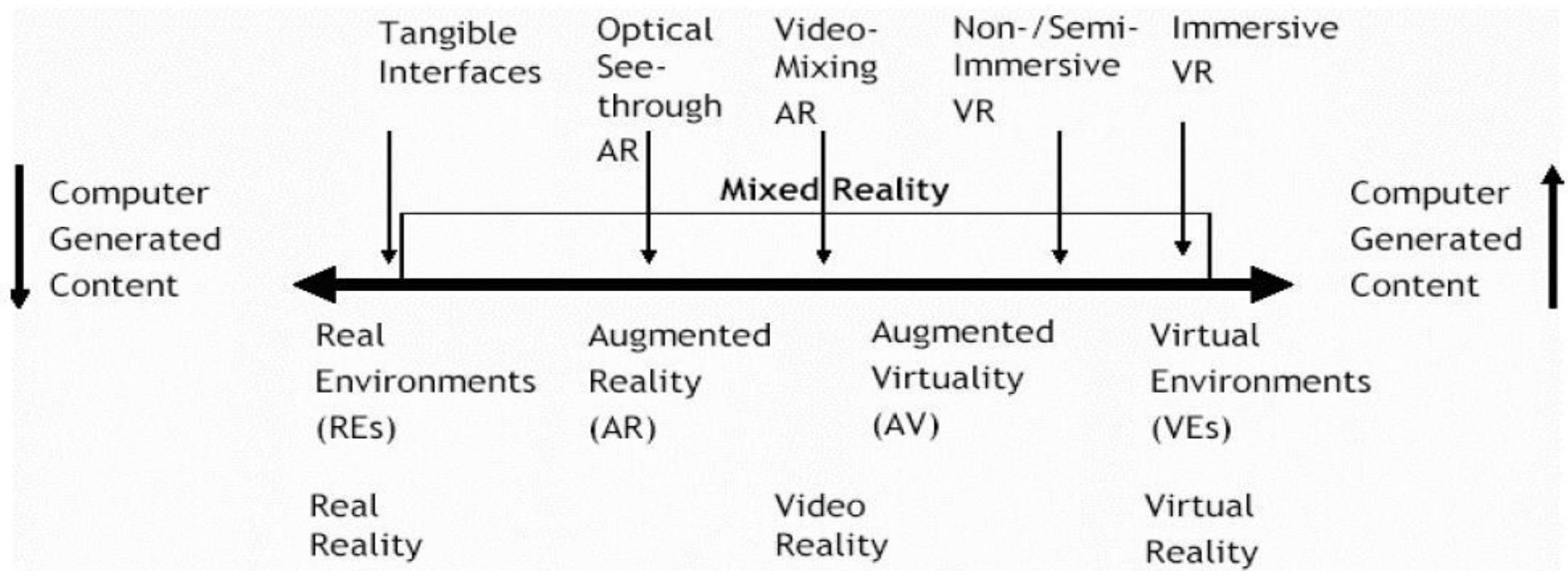
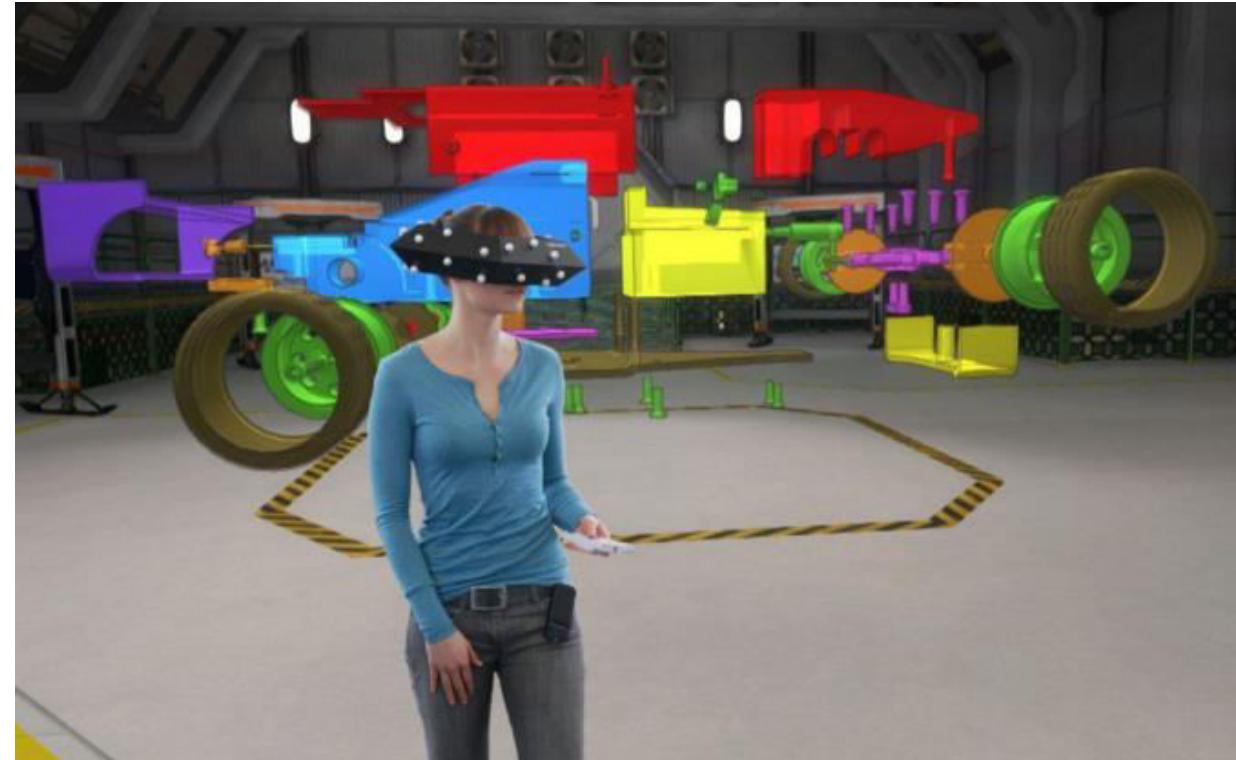


Figure 1-3: Reality-virtuality Continuum (Milgram and Kishino 1994)

# What is Virtual Reality



# Augmented Reality vs. Virtual Reality

- Augmented reality (AR) and virtual reality (VR) are fields in which the lines of **distinction are kind of blurred(බෙඳ වී ඇත)**.
- To put it another way, you can think of VR as the precursor(ප්‍රථමගාමියා) to AR, with some parts overlapping in both. The main difference between the two technologies is that VR

## • **Virtual reality**

- Immerses the viewer into **computer-generated environments**
- Requires equipment(ලුපකරණ) which completely obstructs(බාධා) visual view of physical objects in the real world
- **Does not use a camera feed.**
- All the things displayed in VR are either **animations or prerecorded bits of films**

## • **Augmented reality**

- Augments or adds graphics, audio, and other sensory enhancements to the **natural world as it exists**.
- **Use Camera feed not Animations**
- System augments the real world scene
- Needs a mechanism to combine virtual and real worlds
- User maintains a sense of presence in real world

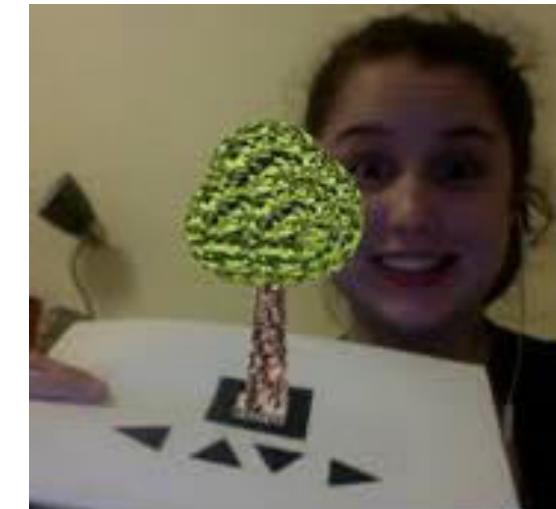
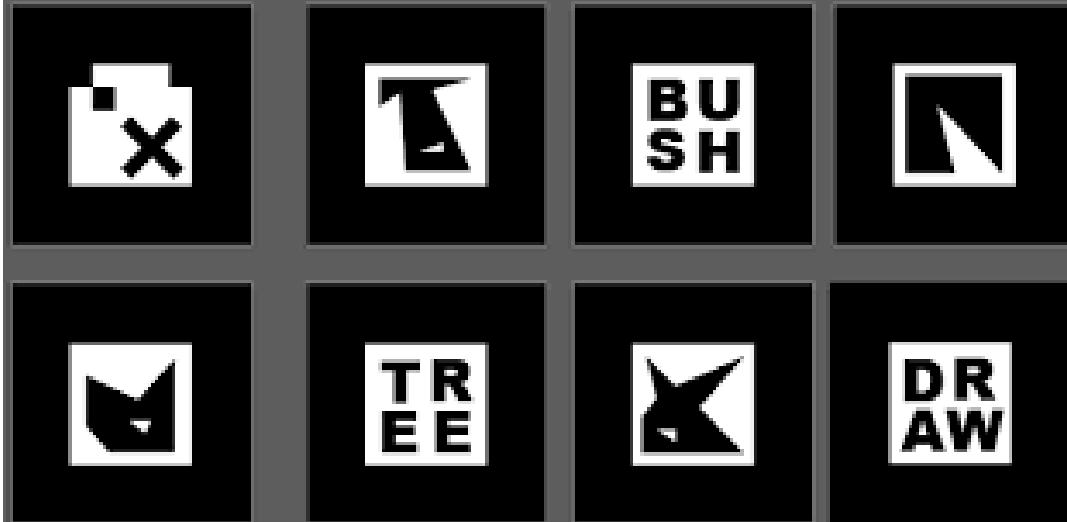
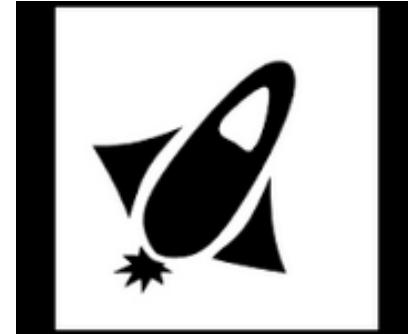
# What is needed?

- There are three components needed in order to make an augmented-reality system work:
  - Head-mounted display
  - Tracking system
  - Mobile computing power



# **Marker-based vs. Marker-less Augmented Reality**

# Markers in Augmented Reality



# Marker-based AR



- In a **marker-based AR** application the images (or the corresponding image descriptors) are provided beforehand.
- In this case you know exactly **what the application should recognize** while (අන්තර් කර ගැනීම) camera data.
- **Marker-based AR** application where image recognition is involved, the marker can be an **image**, or the corresponding descriptors (**features + key points**)
- **Marker-based** augmented reality is when the tracked object is black-white square marker. A great example that is really easy to follow shown.
- These are **easily recognized and tracked** not a lot of processing **power on the end-used device** is needed to perform the recognition (and tracking)

# Marker-less Augmented Reality



- Marker-less AR application recognizes images that **were not provided to the application beforehand**.
- This scenario is **much more difficult** to implement because the **recognition algorithm running in your AR application** should identify patterns, colors or some other "features" that may exist in camera frames.

# Marker-less Augmented Reality

- **Marker-less** AR typically uses the **GPS feature** of a **Smart Phone** to locate and interact with AR resources.
- **Marker less** augmented reality is when the tracked object can be anything else: **picture, human body, head, eyes, hand or fingers etc.** and on top of that you add virtual objects
- **Marker less** augmented reality systems are a better option for final applications, because they use normal images or objects as targets and they are no invasive like marker-based systems.

# **Applications of Augmented Reality**

# Applications of Augmented Reality

- Augmented Reality for Advertising Applications
- Augmented Reality for Marketing Applications
- Augmented Reality for Industrial Applications
- Augmented Reality for Scientific Applications
- Augmented Reality for Arts Applications
- Augmented Reality for Educational Applications

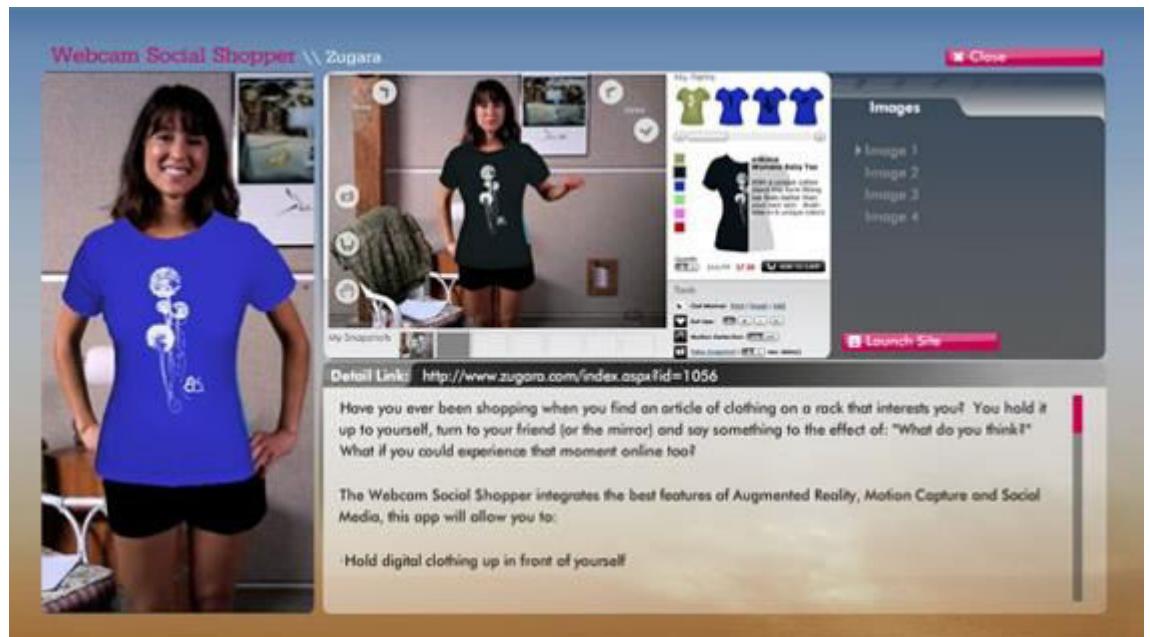
# Augmented Reality for Advertising Applications



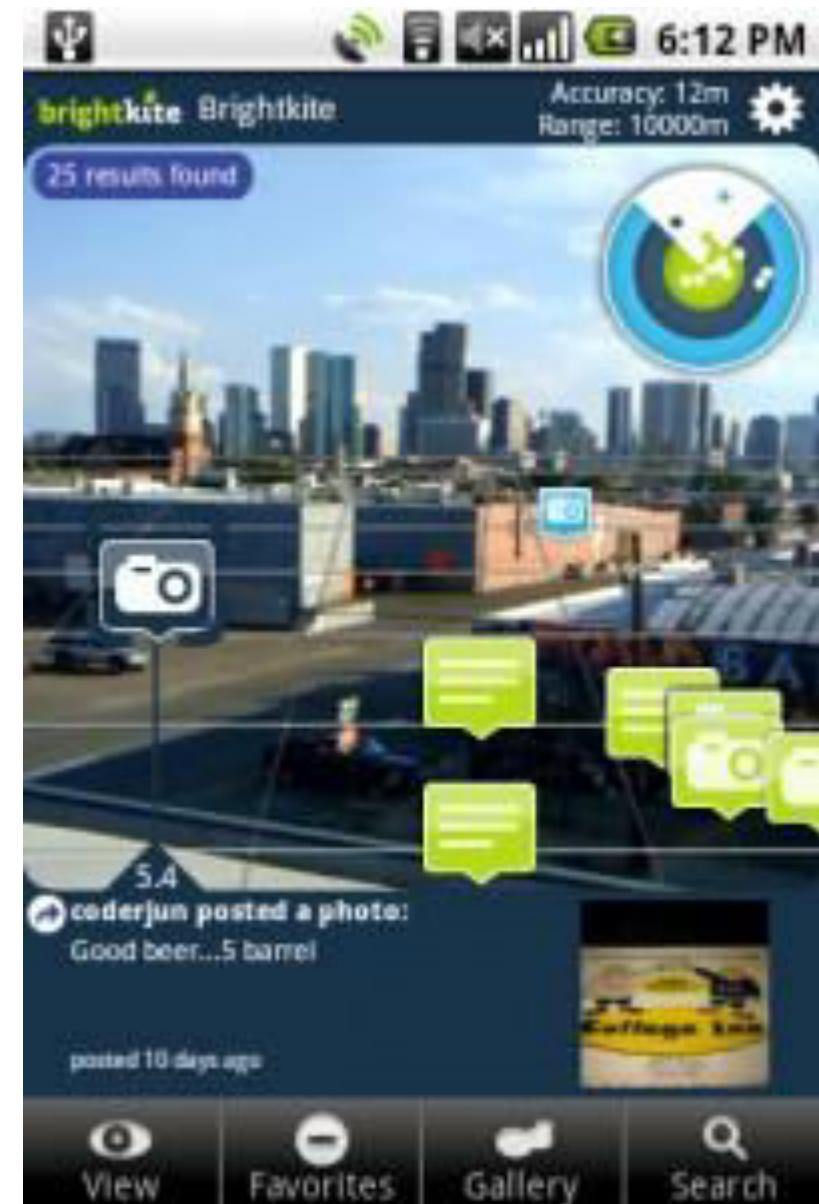
City Sites Tour

- Movie character speaks to you when you pass her outdoor movie poster
- The other example is a **city sites** tour available in most major metropolitan areas.

# Augmented Reality for Marketing Applications

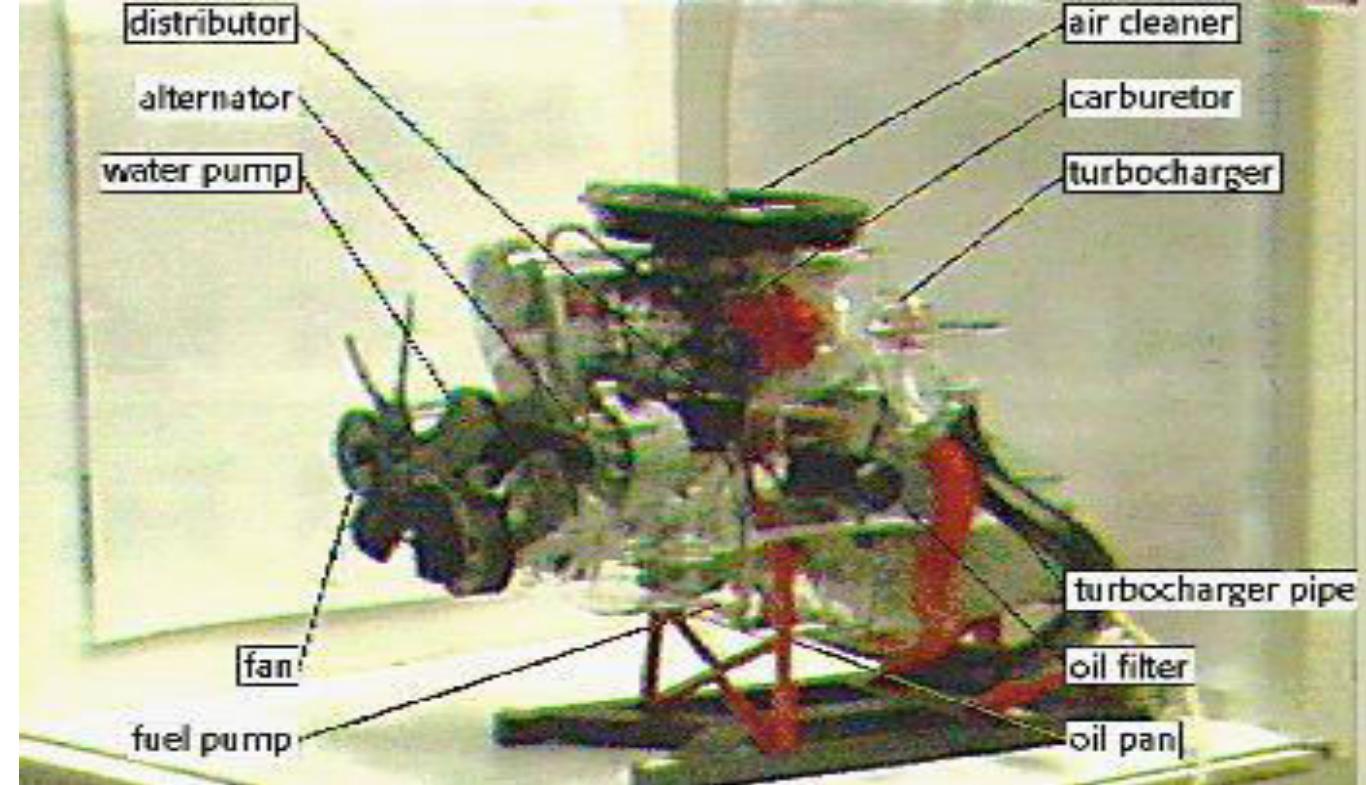


Social shopper



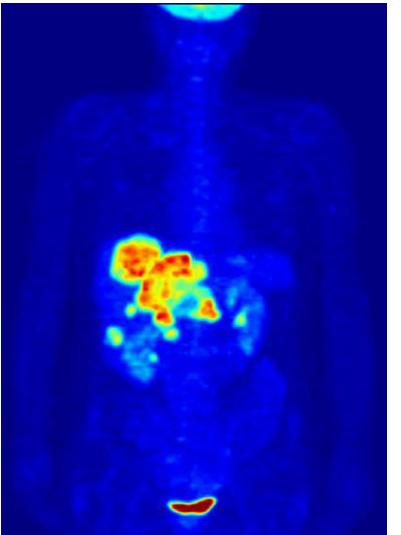
Restaurant search

# Augmented Reality for Industry Applications



- Compare the data of digital mock-ups with physical mock-ups
- Provide instructions, specs, and training for mechanics and machine operators

# AR for Scientific Applications



Whole body PET scan



Terrain rendering

# AR for Arts Applications



Laser Scanned Patient



# AR for Educational Applications



# Wikititude – Augmented Reality Travel Guide

- Mobile travel guide for the Android platform (open source OS for cell phones).
- Plan a trip or find about current surroundings in real-time.



# Braille Reader Application

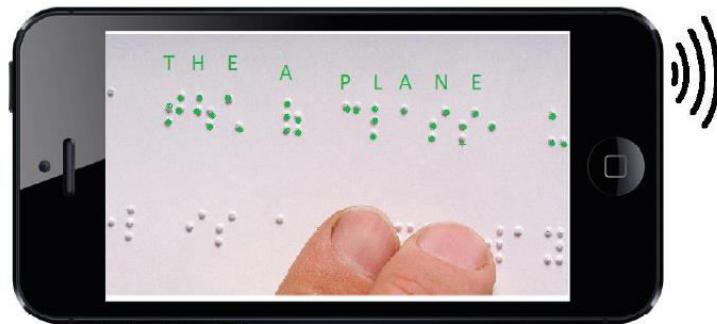


Figure 6: Braille Reader App Design

- The user would **scan the braille** using edge detection software to recognize the braille dots then the **software** would read out loud what the braille says to the user.

# Augmented Audio



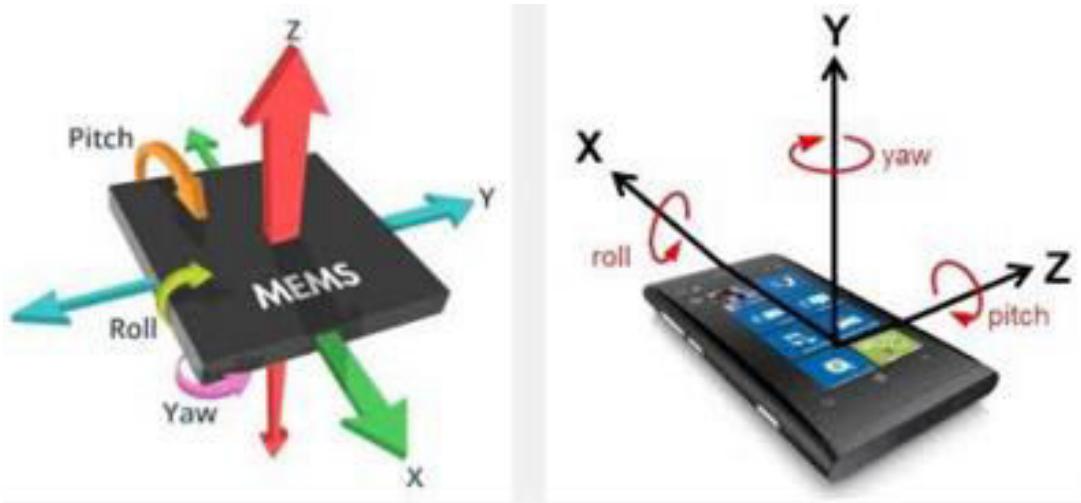
Figure 7: Blind Maps Attachment  
(Designboom 2013)

- Braille-like interface used to **navigate the visually impaired around urban areas**, would further extended the concept.
- The user would **scan written/typed text** translating it into **braille**, raising the dots on the attachment for the user to read.

- The majority of AR apps create **visible augmentation** such as **3D objects**.
- This app would focus on the **audio possibilities** of AR.
- The idea was created whilst reading Harma (2004) paper on augmented audio.

# **How does Augmented Reality Work?**

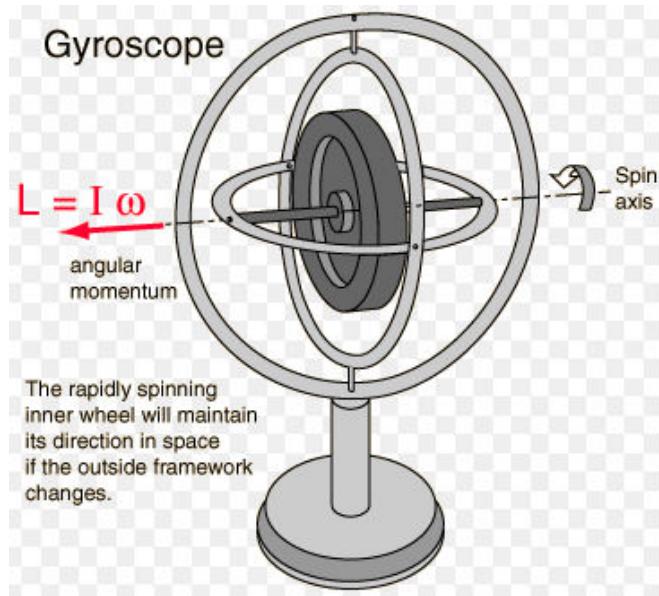
# Sensors support for AR



Accelerometer



Magnetometer



Gyroscope

- **Accelerometer** - Accelerometers in mobile phones are used to **detect the orientation of the phone**.
- **Magnetometer** – To **identify the direction of which the camera is pointed** and **changes the Augmented 3D object direction to match**.
- **Gyroscope** – To **identify the orientation of the floor and the 3D objects camera**.

# Classifications of AR Systems

- Hardware of the AR systems – (Type of tracking systems)
- Visualization approach – (ST, Video mixture) – most well known classification
- Distance – (Indoor, Outdoor)
- Communication – (Wireless, Hardwired)

# Components in AR systems

- There are 4 general components in AR systems
  - 1) Displays - Critical components where the level of immersion is concerned
  - 2) Tracking Systems - Responsible for accurate positioning and orientation in the real world
  - 3) Devices or Interaction – HMD
  - 4) Graphics Systems - Responsible for generating the virtual objects
- Setup of AR Systems
  - 1) MB\_AR - (Monitor-Based AR Systems)
  - 2) ST\_AR – (See-Through AR Systems)
  - 3) SAR – (Spatial AR Systems)

# Monitor-Based AR Systems (MB\_AR)

- Allows the user to observe the real world and the superimposed virtual objects on a regular display, thus **without need to wear special glasses**.
- Widely used in laboratories for testing systems and creating low-cost demonstrations
- Real world are enhanced with the virtual scene (generated by a conventional graphic system) and visualized on the screen

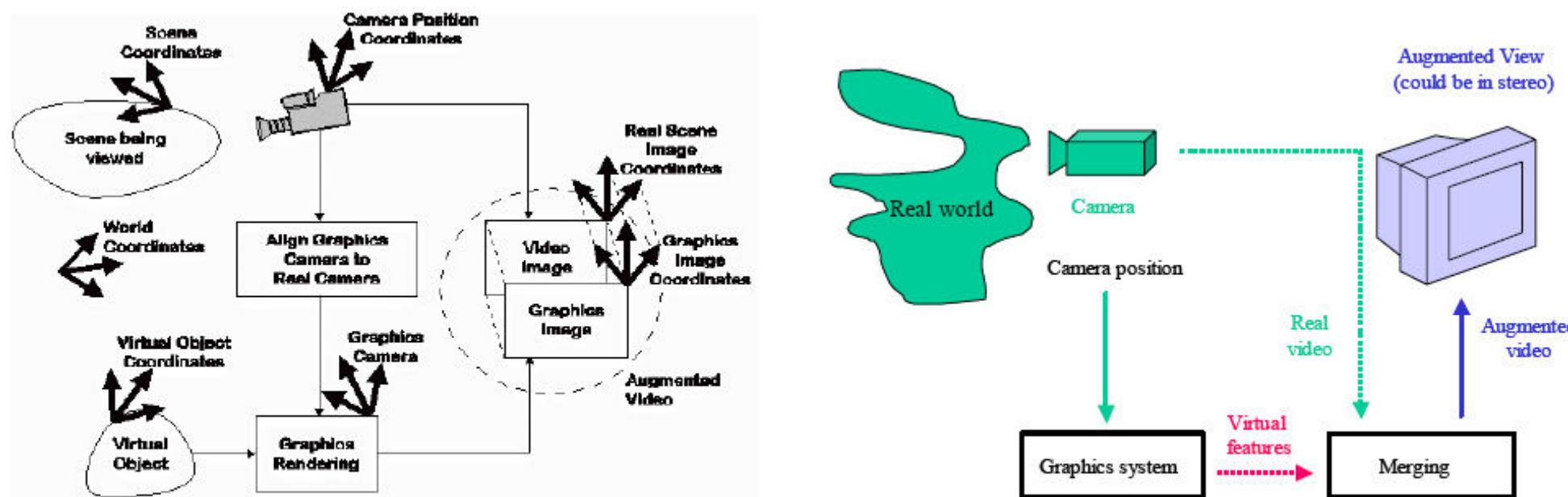


Figure 2-1: Basic components to achieve the augmented view (J. Valino, left) and Monitor-based AR (right)

# See-Through AR Systems (ST\_AR)

- Much more complex - Allow the user to observe the surrounding environment.
- Most of the research and development efforts are toward development of ST-AR systems

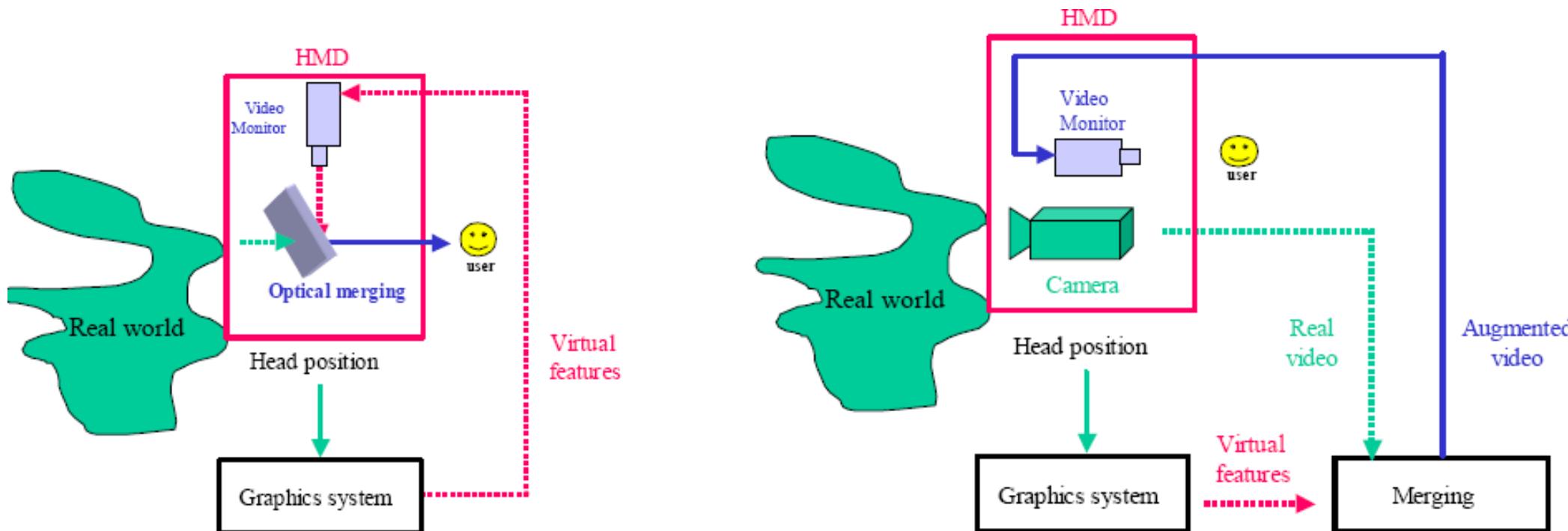
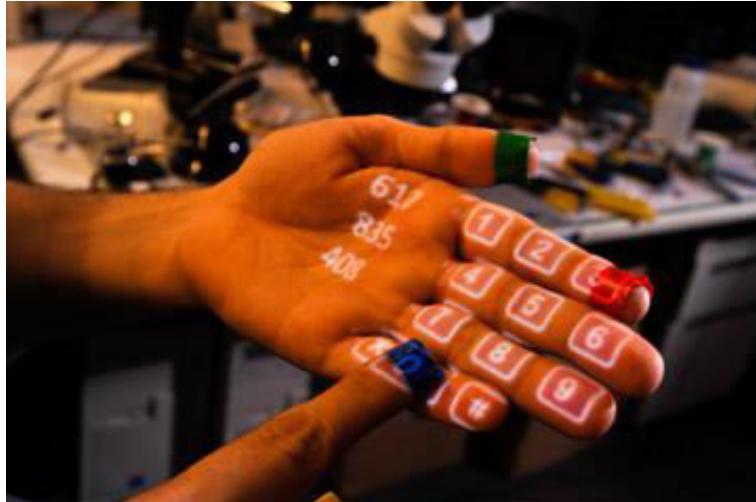


Figure 2-2: ST-AR systems: optical see-trough (left), video see-trough (right)

# Spatial AR Systems

- Spatially Augmented Reality (SAR) systems is also gaining popularity
- Projecting the view in the real world and thus avoiding the use of HMD.
- Virtual Object projected onto physical environment.

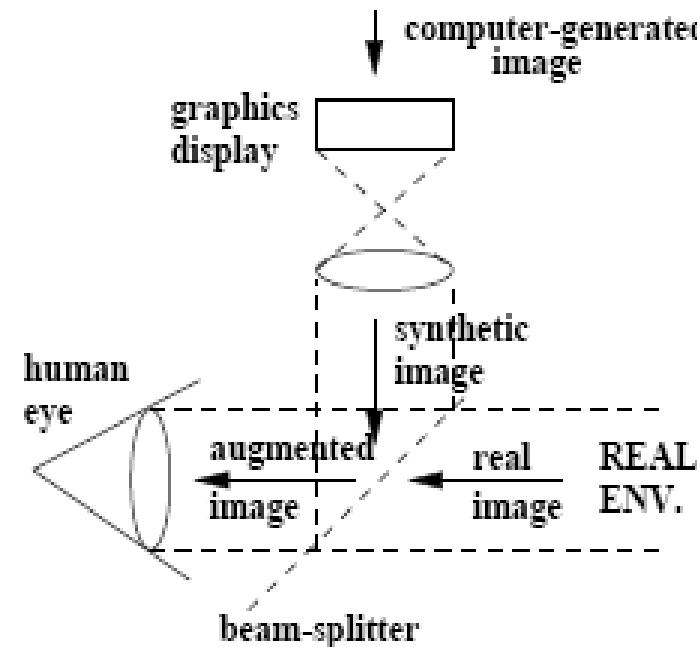
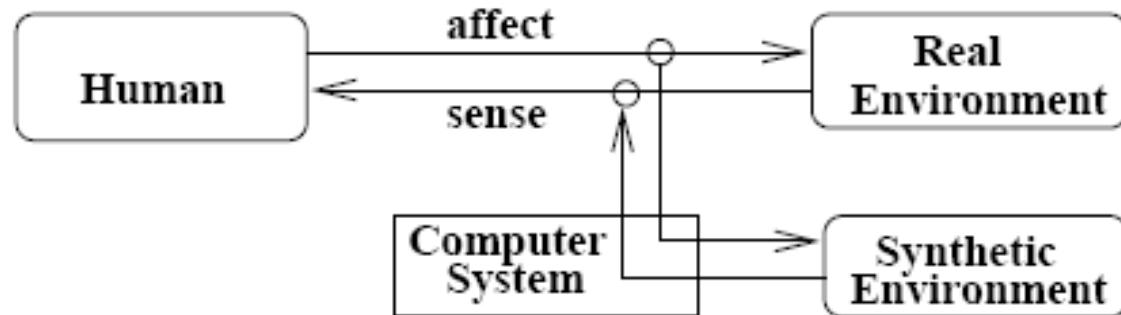


AR Phone Keypad

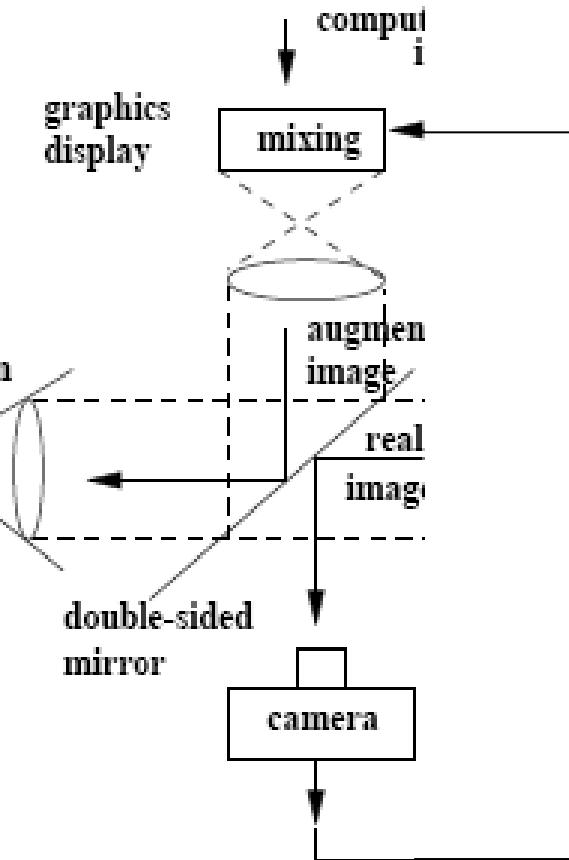


AR Keyboard

# How does Augmented Reality Work?



SEE-THROUGH HMD



OPAQUE HMD

# Head-mounted display

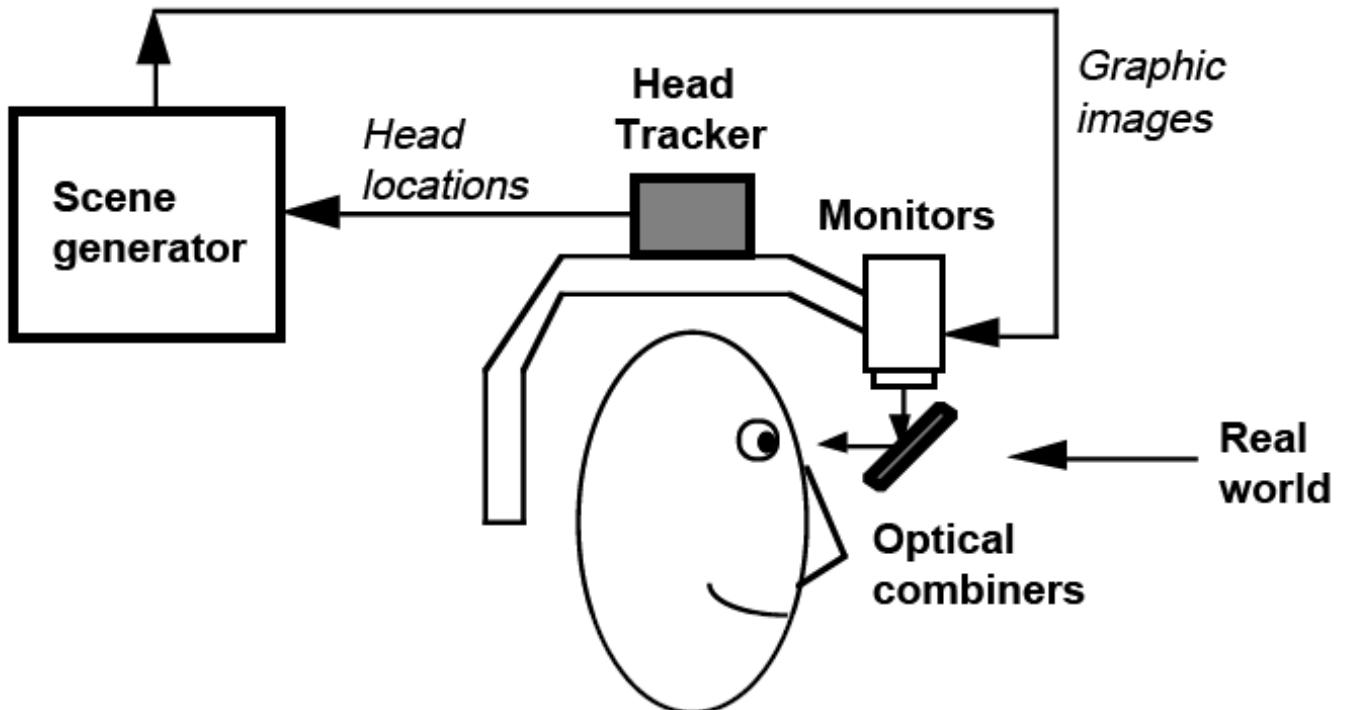


Figure 11: Optical see-through HMD conceptual diagram



Two optical see-through HMDs, made by Hughes Electronics

# Monitor-based AR Concept

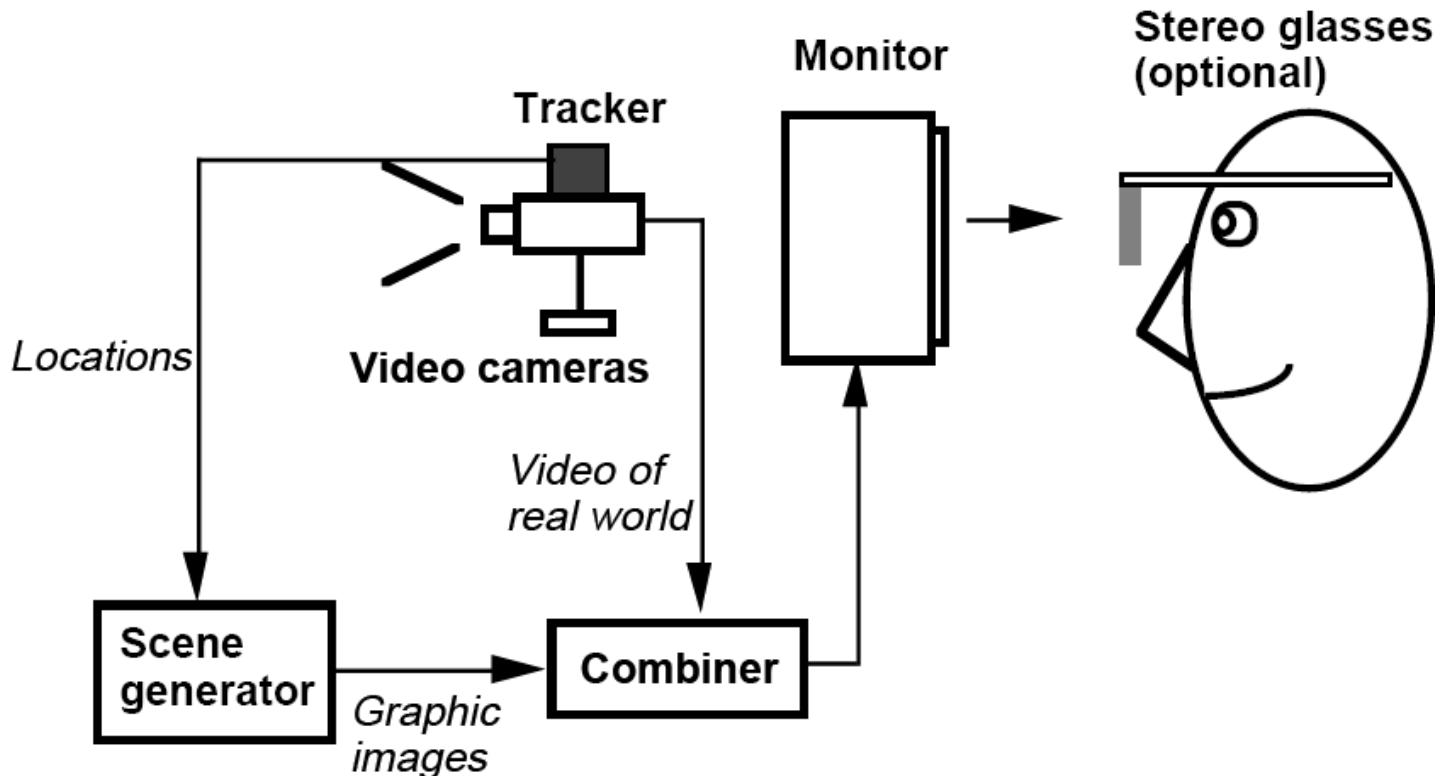
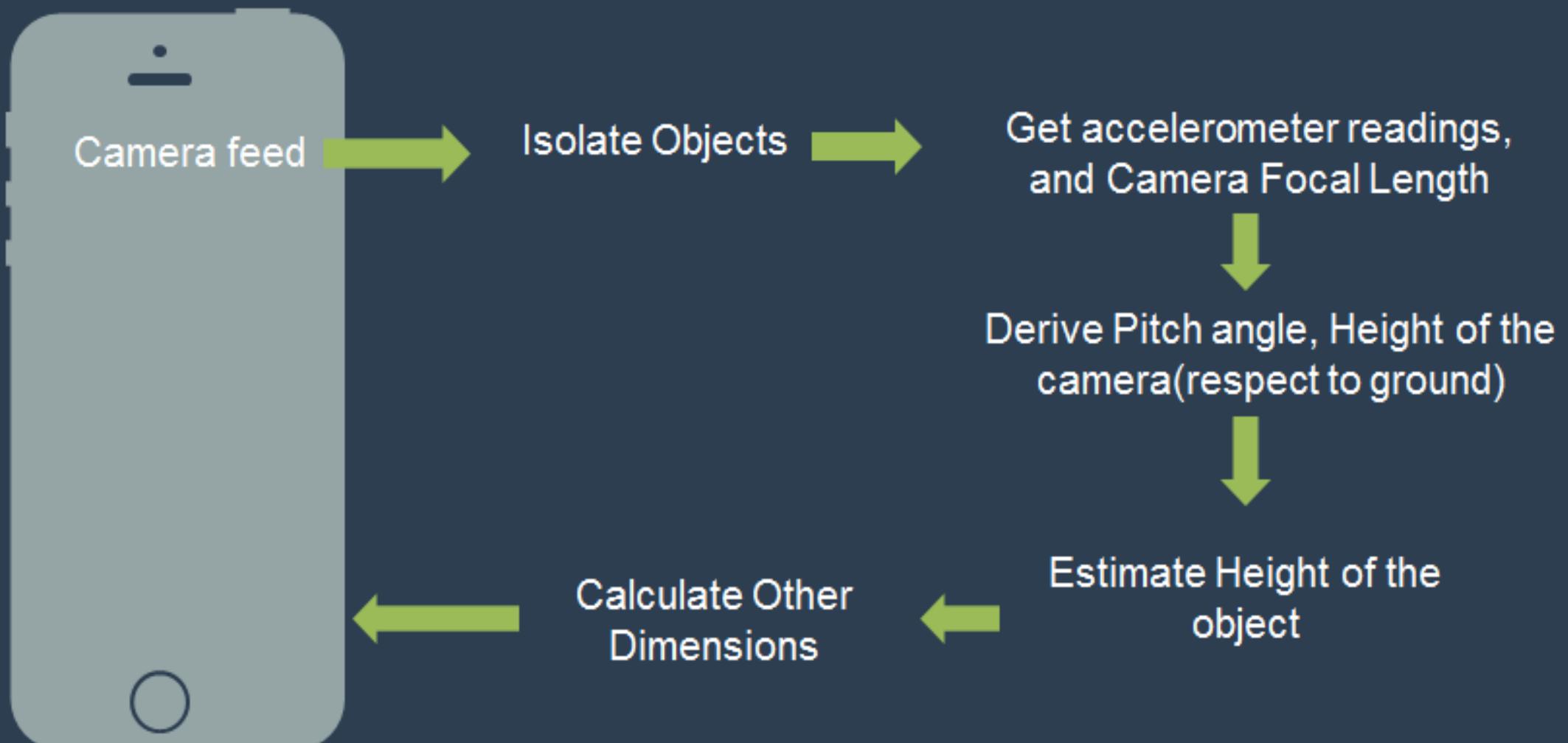


Figure 15: Monitor-based AR conceptual diagram

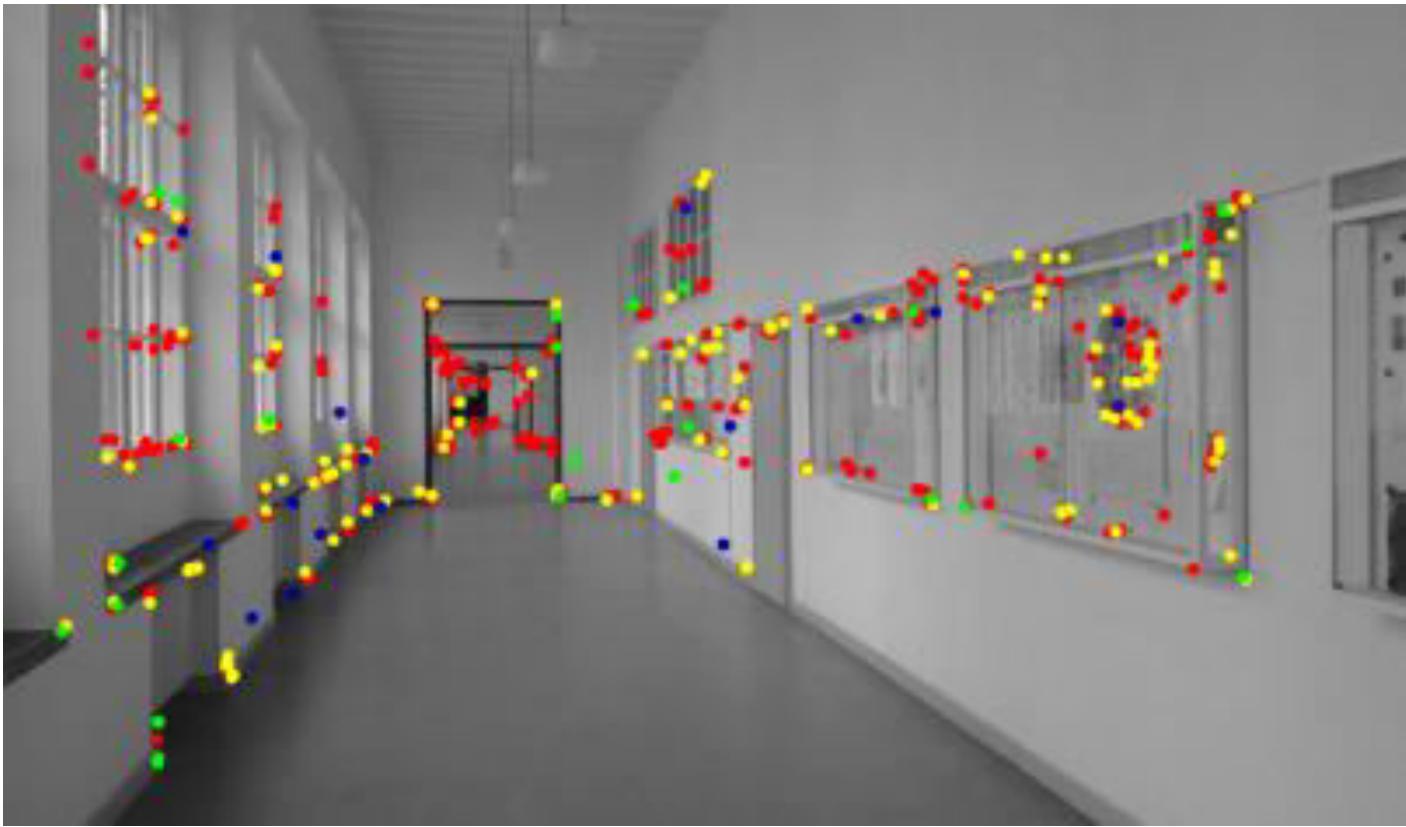
# Scale identification of Real-world objects



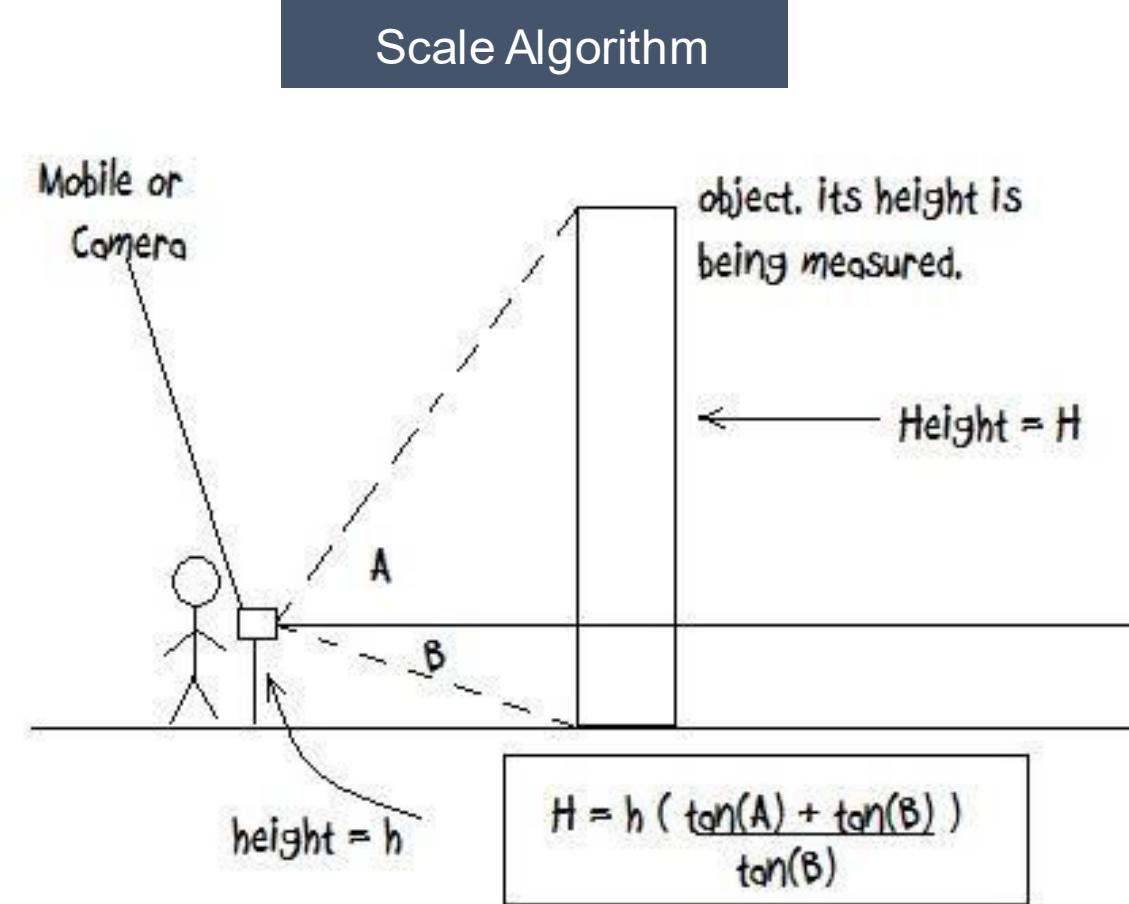
Mapping and 3D object placement component Flow



# Scale Algorithm



Initiate Scanning & obtaining tracking features



Calculate Dimensions through inputs from  
accelerometer , focal length of camera.

# References

- <http://wp.nmc.org/horizon2010/chapters/simple-augmented-reality/> Retrieved September 25, 2010.
- <http://www.bing.com/images/search?q=Augmented+reality+pictures&FORM=IGRE&qpvt=Augmented+realit+y+pictures>. Retrieved October 9, 2010
- <http://www.bing.com/images/search?q=Augmented+reality+pictures&FORM=IGRE&qpvt=Augmented+realit+y+pictures> Retrieved October 9, 2010.
- <http://www.howstuffworks.com/augmented-reality.htm>  
Retrieved October 7, 2010.
- <http://www.britannica.com/EBchecked/topic/1196641/augmented-reality>  
Retrieved October 7, 2010.
- <http://www.bing.com/images/search?q=Augmented+reality+pictures&FORM=IGRE&qpvt=Augmented+realit+y+pictures.#> Retrieved October 13, 2010.
- <http://www.newhorizons.org/strategies/technology/billinghurst.htm>  
Retrieved October 7, 2010.



# Augmented Reality

**Modern Topics in Information Technology  
4<sup>th</sup> Year – Semester 1  
Lecture 02 - Part II  
By Udara Samaratunge**

# Tracking system classification

1. *Magnetic Trackers*
2. *Mechanical Trackers*
3. *Acoustic Trackers*
4. *Trackers using Vision*
5. *Indoor-Tracking system*

# Tracking Systems

- Track user's **Body, Head, Hands and etc.** Examples of tracking systems are ***mechanical, magnetic, acoustic and vision,***
- When track the user within the monitored area, the **position, direction (orientation) of movement and the speed** has to be determined.
- Different techniques were developed to determine object position.
- **Relative Localization –**
  - Evaluating the **position and orientation** by integrating information provided by diverse (usually encoder or inertial) sensors.
  - The integration is started from the initial position and is **continuously updated**
- **Absolute Localization -**
  - Technique that permits the vehicle to **determine its position** in the **domain of motion**.
  - These methods usually rely on **navigation beacons**, active or **passive landmarks, maps** matching or satellite-based signals like **Global Positioning System (GPS)**.

# 1. Magnetic Trackers.

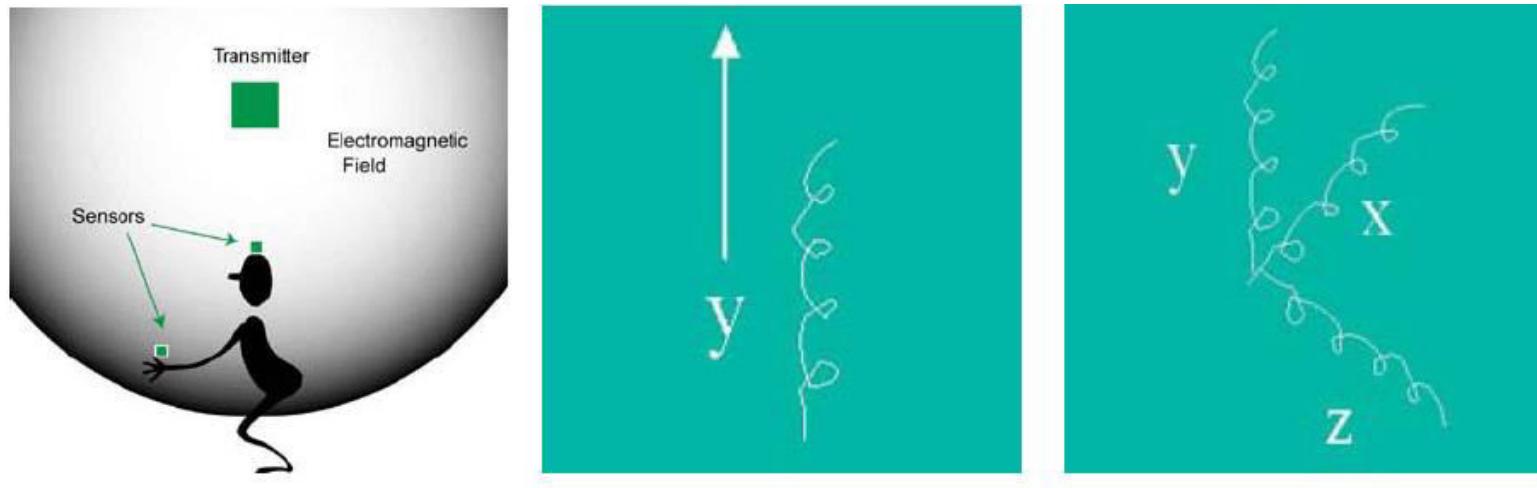


Figure 2-6: Magnetic tracker: general schema (left), 1-DOF (middle) and 3-DOF (right) (Capps, 2000)

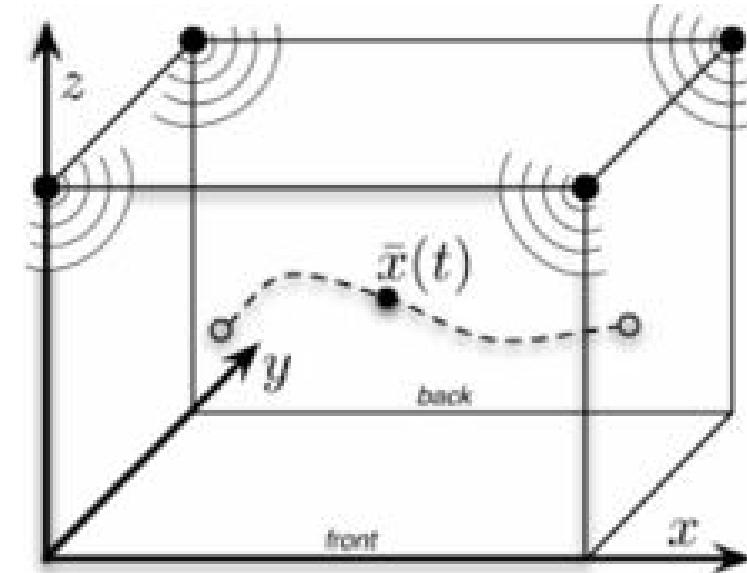
- **Magnetic trackers** are used to capture translation coordinates ( $x,y,z$ ) and yaw, pitch, roll ( $y,p,r$ ) rotation coordinates
- **Magnetic tracking** is most commonly used as an interface to a virtual world, for instance, by tracking head, hand, or input device motion.
- Some magnetic trackers can follow a **number of devices simultaneously**.
- Magnetic tracking technology can be an option for **full-motion body capture**.
- **Real-time**, perhaps to drive the motion of a virtual character, or can be recorded to give virtual actors **realistic motion characteristics**

## 2. Mechanical Trackers

- Mechanical position trackers, also known as **goniometers or exoskeletons**.
- The **exoskeleton** must be physically attached to the user.
- It can be **body-based**, user can freely move around.
- These trackers **uses** couple of miniature gyroscopes to measure orientation changes.
- This resistance can be measured, and **converted** into yaw, pitch, and roll values.



### 3. Acoustic Trackers



- **Acoustic tracking** devices use **ultrasonic (high-frequency)**
- Sound waves for measuring the **position** and **orientation** of the target object

# 4. Trackers using vision



Figure 2-7: Automatically recognised and tracked templates (Davison 1999]. Red squares indicate template content and its real-world position

- Vision is commonly used for AR tracking
- Vision methods can estimate camera pose directly.
- The position estimate is often relative to the object(s) of interest and not to a sensor or emitter attached to the environment.
  - a) Tracking may occur relative to moving objects;
  - b) Tracking measurements made from the viewing position often minimize the visual alignment error;
  - c) Tracking accuracy varies in proportion to the visual size (or range) of the object(s) in the image



Figure 2-8 Point tracking, Harris 1992 (left), Line tracking, Schmid and Zisserman, 1997 (middle, right)

- Features can be tracked using **template, point, lines, corners, color and combinations of them.**

- ***Tracking Corners***

1. E.g. **crossings of lines**, or **gradient and curvature measurements** can also be used to correct drift in their inertial tracking system using **curvature corner detectors**.
2. Matching those corners with a projection of stored geometry (You et al 1999).

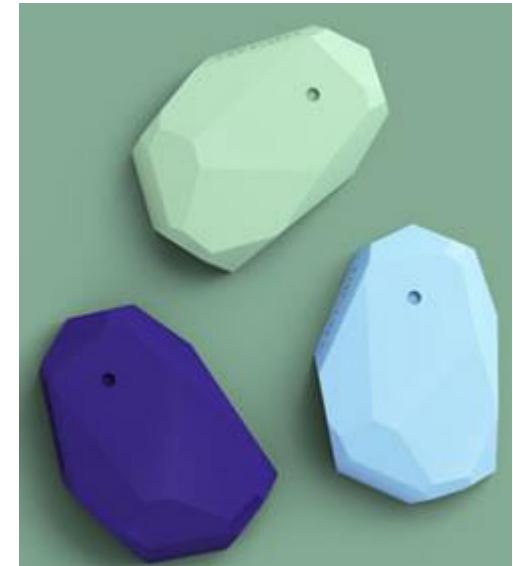
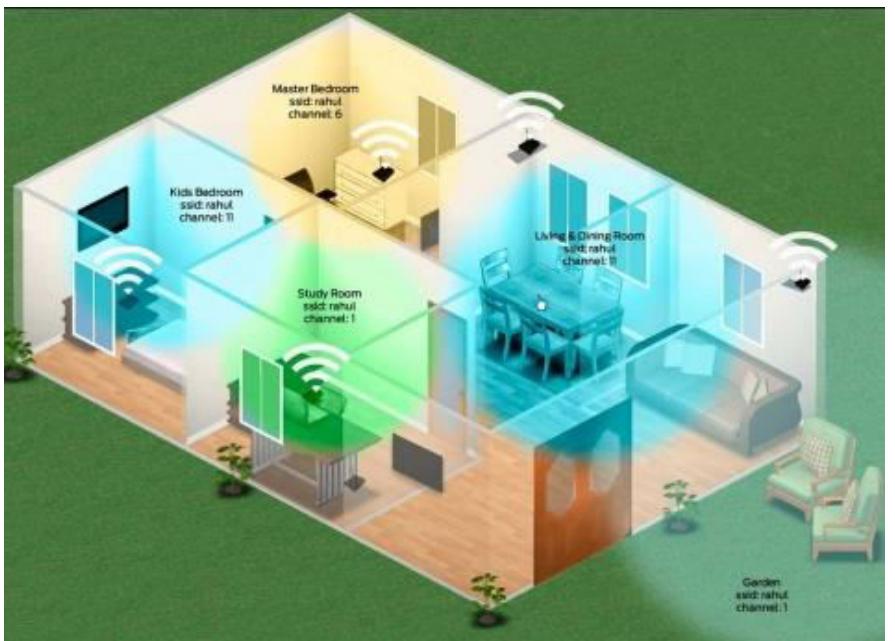
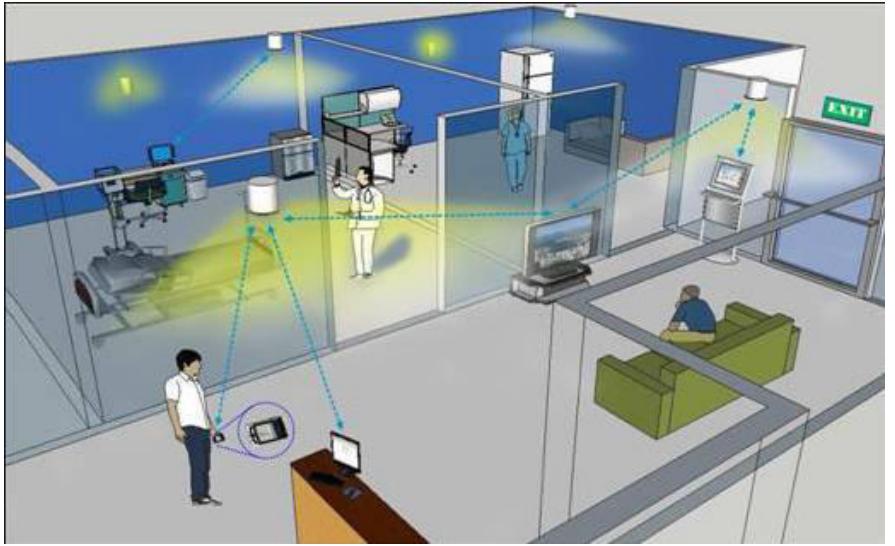
- ***Line matching***

1. Kosaka and Nakazawa 1995
2. Use a **3D model of the environment** and **match lines from this database** using a rough transform of the image.
3. They reach sub-centimeter accuracy and 1 degree orientation accuracy when the distance to a feature is about 20 cm.

- ***Color***

1. Powerful means to enhance tracking robustness.
2. Color tracking is still **not very well explored**, perhaps due to the fact that **colors are highly invariant**.
3. When the user moves around or **lighting conditions change**.

# 5. Indoor Tracking System



# Usage of Beacons



	<b>Location Beacon</b>	<b>Proximity Beacon</b>	<b>Sticker Beacon</b>	<b>Video Beacon</b>
Battery life	5 years	2 years	1 year	Endless (USB-powered)
Range	200 meters	70 meters	7 meters	10 meters
Thickness	24 mm	17 mm	6 mm	14 mm
iBeacon™ or Eddystone™ packets	8 simultaneously	1 at a time	1 at a time	2 simultaneously
Additional packets	connectivity, telemetry, user-defined	connectivity, telemetry	connectivity, nearable with telemetry	connectivity, telemetry, user-defined
Built-in sensors	motion, temperature, ambient light, magnetometer, pressure	motion, temperature	motion, temperature	n/a
Additional tech	GPIO, RTC, LED, 1Mb EEPROM	Programmable NFC	n/a	WiFi, HDMI, USB, 1GB eMMC Storage
Devices in the Kit	3 beacons	3 beacons	10 stickers	3 mirrors

# A Real-time Object Identification & Registration for AR

- AR systems normally measure the position and orientation of a device with 3D sensors (either magnetic or ultrasonic);
- By analyzing the **distortion of the rectangular shape** of the matrix code frame, the system estimates the position and orientation of the video camera.

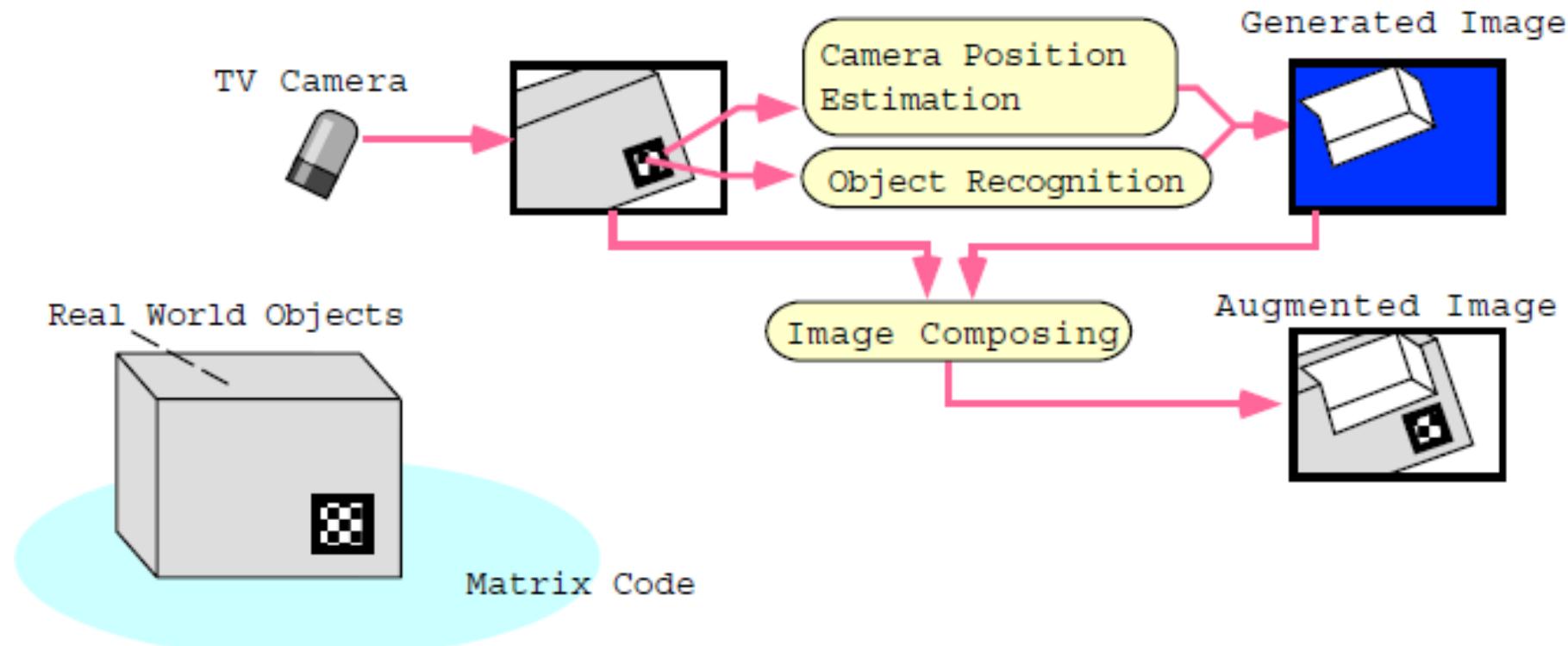
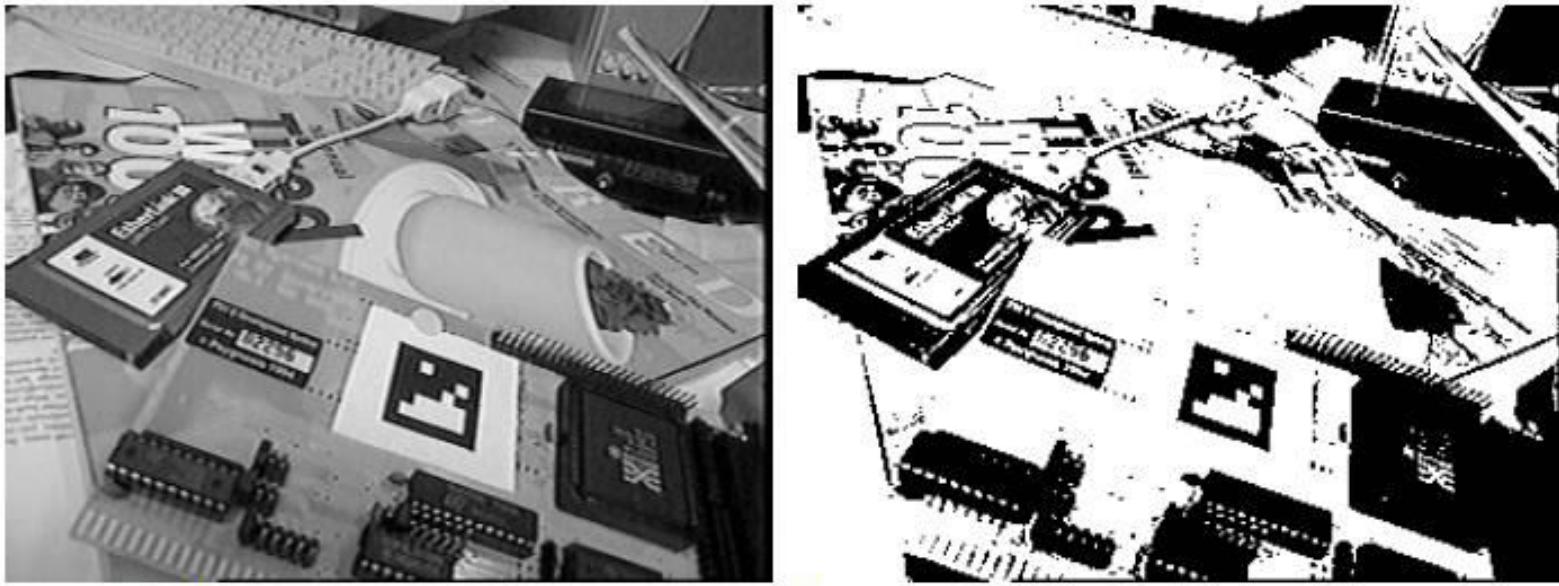


Figure 2. Overview of the proposed vision-based information registration method

# Steps in Recognition Algorithm

1. Binarization
2. Connected Component Analysis
3. Code Frame Fitting
4. Decoding and Error Check
5. Camera Position and Pose Estimation

# Binarization



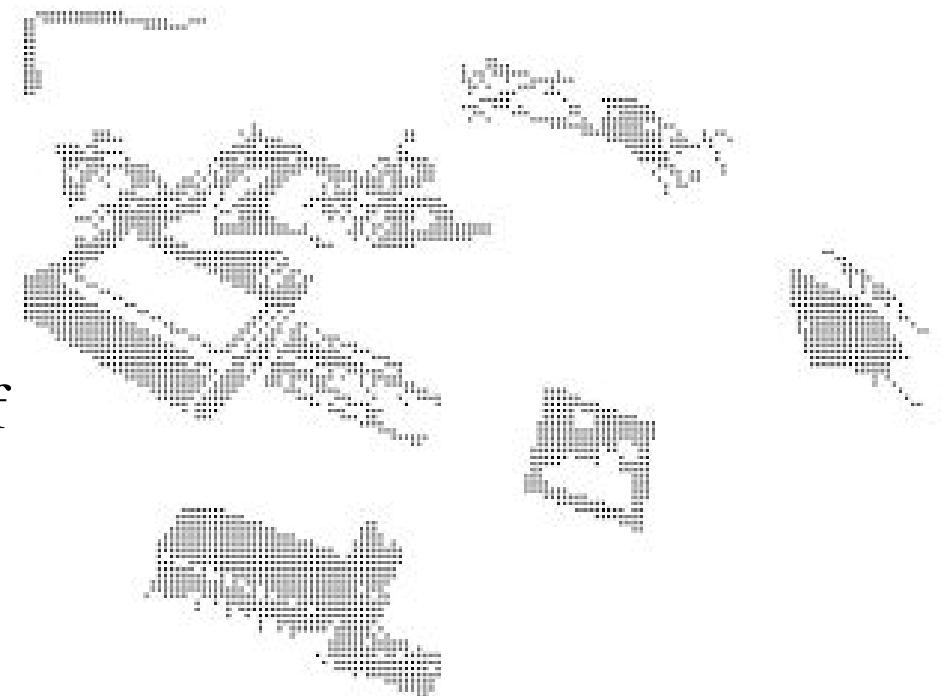
(a) Original image

(b) Binarization

- **Image binarization** converts an image of up to 256 gray levels to a **black and white image**.
- Thresholding is the simplest method of **image segmentation**. From a **grayscale image**, thresholding can be used to **create binary images**.
- The simplest thresholding methods **replace each pixel in an image** with a black pixel if the image **intensity**  $I\{i,j\}$  is less than some fixed constant  $T \Rightarrow (I\{i,j\} < T)$

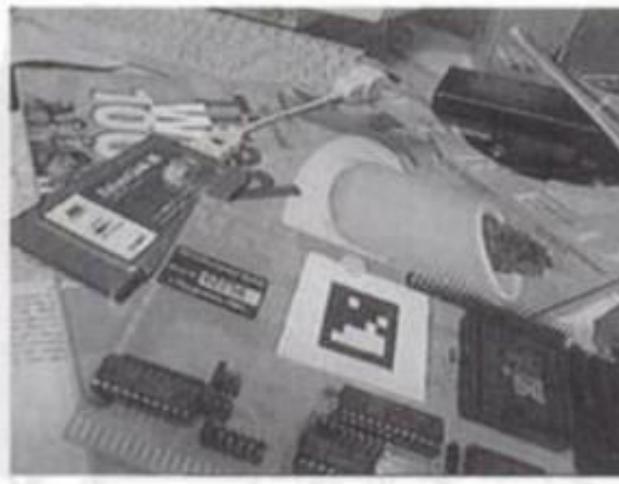
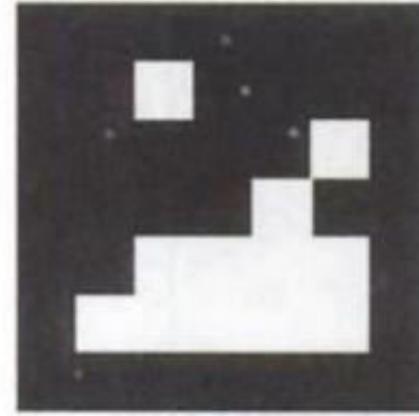
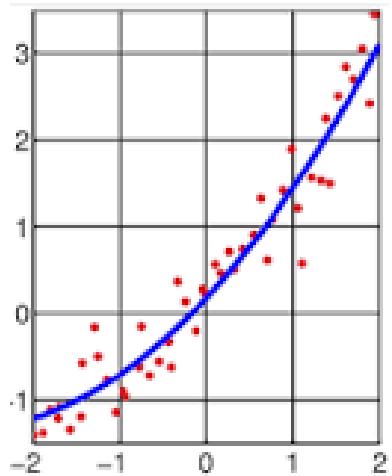
# Connected Component Analysis

- Connected regions => binary-1 (**black**) pixels.
- Identify connected region
- heuristic check (based on the **size** and the **aspect-ratio** of the region bounding rectangle).



(c) Connected components analysis

# Code Frame Fitting



- For each selected region a rectangle is fitted on the **outline** of region using **least-square method**.
- Then, transformation parameters are calculated based on the positions of the **4 corners of this quad-tangle**.
- Then **Distortion-compensation matrix** is calculated based on the **4 corners of quad-tangle**.

# Decoding & Error check

- Using the obtained parameters, the system projects the corresponding image region to the code rectangle space.
- The number of black and white pixels => determine **cell bit (1,0)**
- Finally, the **CRC-error check** is applied on the decoded bit Pattern
- And the certificated **cell value** is regarded as a recognized code ID.

# Camera Position and Code Estimation

- The recognized **code frame** is used for camera pose estimation.
- From **4 known points** on the (real-world) **image plane**
- It is possible to calculate a matrix representing the **translation** and **rotation** of the **camera** at a real-world coordinate.
- We use **4 corners of the 2D-code frame** as these reference points.

# **Tools, Techniques and Approaches**

# Tools, Techniques and Approaches

- Identifying and selecting the appropriate tools that are going to be implemented during the Augmented Reality project.

## Platform Comparison

Platform	Price of development	Equipment owned	Ability to create AR apps
iOS	Positive – X-Code is free  Negative - £90 development license	Positive - iPod Touch - Mac provided by university	Yes
Android	Positive - Android Studio is free - No developer license required	Positive - Laptop capable of running Android Studio  Negative - HTC One Android Device incompatible with Android Studio	Yes

# Tools, Techniques and Approaches cont.

Platform	Price of development	Equipment owned	Ability to create AR apps
Windows	<p>Positive -Visual Studio Express 2013 free basic version Negative - Pay for full version of visual studio</p>	<p>Positive - Laptop capable of running Visual Studio Negative - No mobile device owned</p>	Yes
BlackBerry	<p>Positive - Momentics is free - Developer license free</p>	<p>Positive - Laptop capable of running Momentics Negative - No mobile device owned</p>	Yes

# Integrated Development Environments (IDE)

IDE	Capability with Equipment	Developer Experience	Price	Capability with an independent AR SDK	Online Tutorials and Videos
X-Code	Yes	Positive - Beginner (3 months experience)	Free	Positive - Yes numerous	Positive - Lots, University paid subscription to Lynda.com
Unity	Yes	Positive - Beginner (previous project)	Free (Basic version)	Positive - Yes numerous	Positive - Lots

IDE	Capability with Equipment	Developer Experience	Price	Capability with an independent AR SDK	Online Tutorials and Videos
Visual Studio with Marmalade SDK	Yes	Positive - Moderate (2 Years)	Free	Positive - Yes numerous	Positive - Lots, University paid subscription to Lynda.com
DAQRI 4D	Yes	Negative - None	Negative Paid - £440 month is the cheapest	N/A Dedicated AR Studio	Negative - Minimal
Aurasuma Studio	Yes	Negative - None	Free	N/A Dedicated AR Studio	Negative - Minimal
Metaio Creator	Yes	Negative - None	Free (Basic version)	N/A Dedicated AR Studio	Negative - Minimal
Layar	Yes	Negative - None	Free	N/A Dedicated AR Studio	Negative - Minimal

# Augmented Reality SDKs

AR SDK	Price	Compatibility with iOS/X-Code	Developer Experience	Online Support
Metaio	Positive - Free (Watermarked)	Yes	None	Positive - Lots of Documentation - Metaio HelpDesk - Numerous Tutorials
Wikitude	Positive - Free	Yes	None	Positive - Lots of Documentation
Vuforia	Positive - Free (Watermarked)	Yes	None	Positive - Lots of Documentation - Numerous Tutorials  Negative - Majority tutorials are for Unity

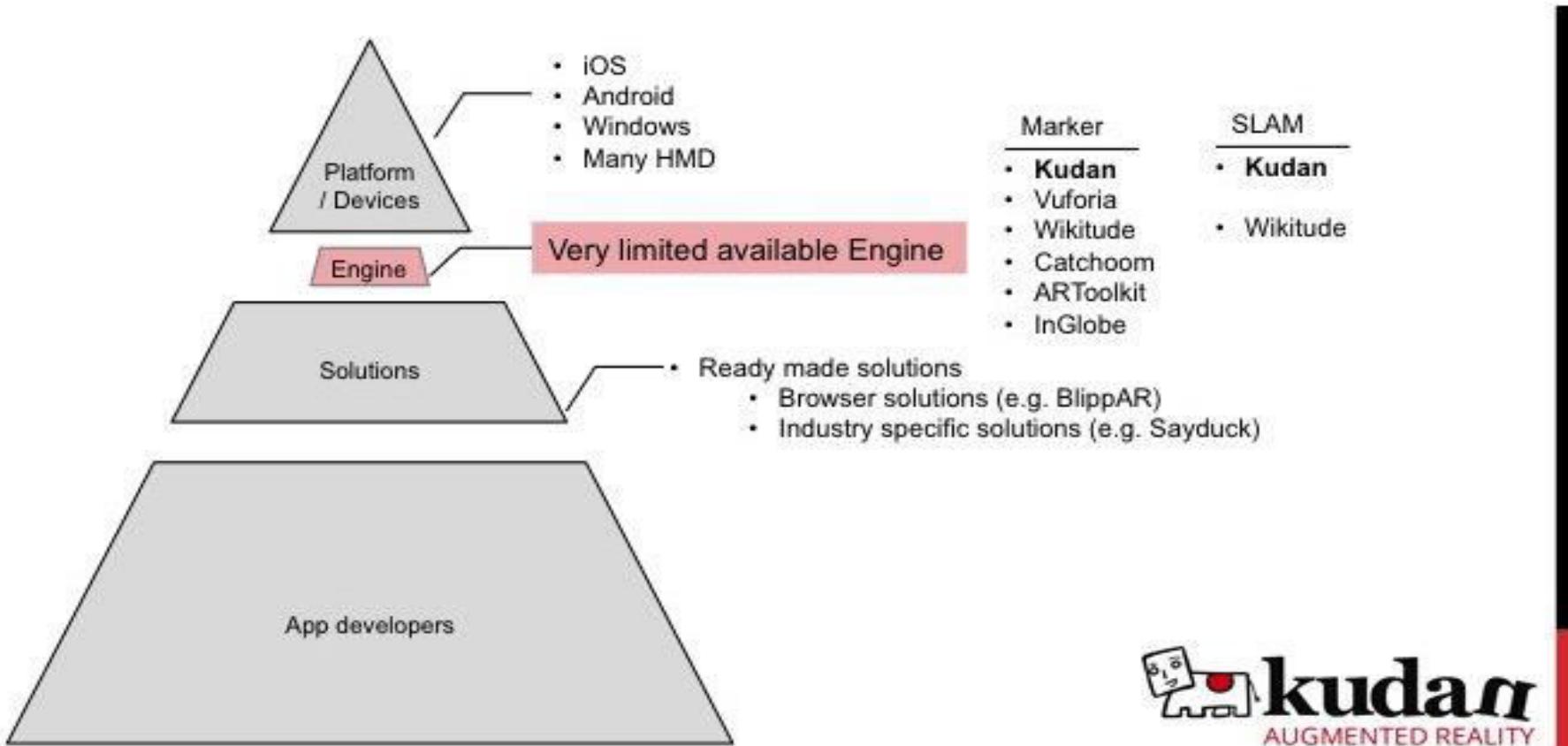
# Feature Comparison

	Vuforia	Metaio	Wikitude
Marker	Frame makers, image target, text target	ID, picture and LLA marker, QR and Barcode	Image, Barcode tracking
GPS	No	Yes	Yes
IMU	No	Yes	Yes
Face	No	Yes	No
Natural Features	Yes	Yes	Yes
3D Objects	Yes	Yes	No
Others	Extended tracking, localize occlusion detection	Instant 3D maps tracking, 3D SLAM, extended image tracking	Hybrid tracking (Image and GPS), extended tracking
Marker Generation	Online Target Manger	No online tool Provides ready made 512 different ID Markers	Online target manager tool

# Feature Comparison Cont.

	Vuforia	Metaio	Wikitude
Benefits	<p>Enable to maintain tracking even when the target is out of view and view them from greater distance.</p> <p>Cloud Database allows storing thousands of image targets.</p>	<p>Powerful 3D rendering engine with capability load 3D model of .obj format.</p> <p>No limit on number of trackable object depends on device memory.</p>	<p>AR content can be programmed using basic HTML5, JavaScript and CSS.</p> <p>Easy portability of AR apps from one platform to another.</p>
Limitations	Majority of Tutorials for the Unity platform	Difficult to render complex 3D objects also limitation is associated with model size.	Doesn't track 3D model which limits is use to only 2D tracking.  Target image to track need to be of solid colors to be recognized

# Existing Technologies for Augmented Reality



# Comparison

	Advantages	Disadvantages	Potential Concepts Uses
Marker Based (QR, Image)	<ul style="list-style-type: none"><li>- Simple to implement (AR Standard for visual applications</li><li>- Cheap in terms of processing power</li></ul>	<ul style="list-style-type: none"><li>- Limited range (the distance from tracer to device)</li><li>- Tracking challenges; unfocused camera, dark/unevenly lit environments, image noise (poor lens, block compression), occlusion (tracker and environment blending)</li><li>- Usually works only when fully in view</li></ul>	<ul style="list-style-type: none"><li>- Laser Quest App</li><li>- Chemistry App</li></ul>
Assisted GPS	<ul style="list-style-type: none"><li>- Augmentation can dynamically change depending where the user is</li></ul>	<ul style="list-style-type: none"><li>- Signal blocking (trees, buildings, mountains, indoors)</li><li>- Inaccuracy/ Bad data from satellites (misreporting their own position)</li></ul>	<ul style="list-style-type: none"><li>- Which Plane?</li><li>- Laser Quest App</li><li>- History App</li><li>- Runner Heart Rate App</li></ul>
Face Recognition		<ul style="list-style-type: none"><li>- New</li><li>- Difficultly remembering/recognizing different faces</li></ul>	<ul style="list-style-type: none"><li>- Facebook App</li></ul>
Natural Feature	<ul style="list-style-type: none"><li>- Works if partially viewed</li><li>- Targets are potentially everywhere</li><li>- targets may catch attention less (blend into the environment)</li></ul>	<ul style="list-style-type: none"><li>- More difficult, markers are designed for their purpose, the natural environment is not</li></ul>	<ul style="list-style-type: none"><li>- Braille App</li></ul>

# **What is not Augmented Reality & Rejected Ideas**

# What is and what is not Augmented Reality?

- All the ways of doing augmented reality have a common point: **The mix of reality and virtual.**
- What is not augmented reality?
  - For example, some cities provide QR Codes, which redirect to a website if you point at it with a smartphone. In these projects, **QR Codes are placed on famous buildings and places, and the website you are redirected** to explains you the building history, and some other extra information.
  - This kind of project is not augmented reality, because it does not mix this extra information with the reality, it just links it.
- Again, in augmented reality applications, reality and virtual have to be mixed.

# Rejected ideas in Augmented Reality

- History App design
- It was rejected because the concept wasn't stressing the boundaries of AR, and the concept resembled a 2D overlay more than AR application a user could interact with.
- Azuma (1997) stated 2D overlays could not be AR because the user interact with it, therefore this concept was rejected.

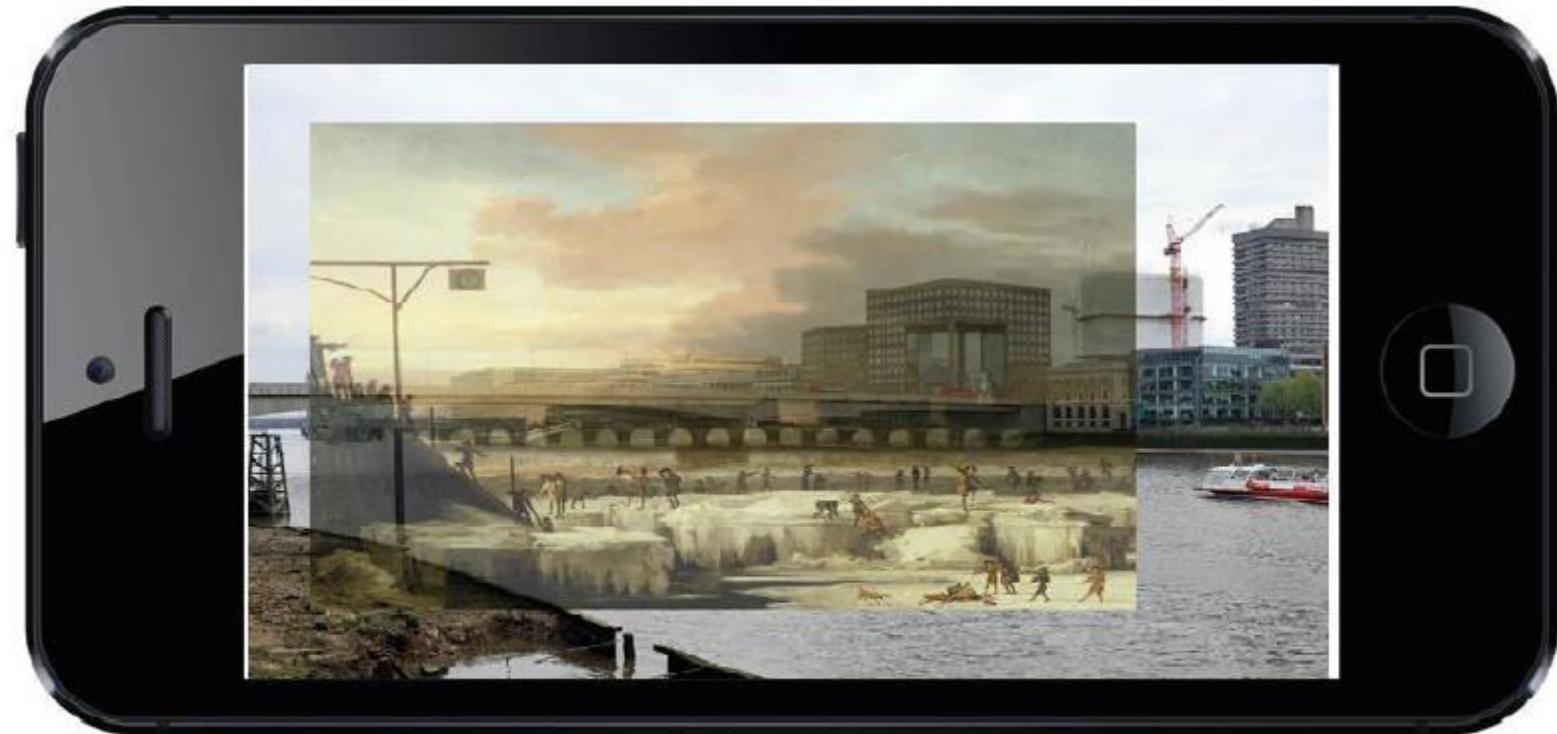


Figure 10: History App Design

# Air Plane details APP

- Using flight data from sources such as [flightaware.com](http://flightaware.com) the user would be able to **track airplanes in the sky by holding their device towards the plane in the sky.**
- The app would **highlight the plane in the sky and display detail such as, which flight it was, departure/arrival locations, flight path, flight boundaries, and flight delay.**



Figure 13: Which Airplane? App design

# **Drawbacks and Limitations**

# Drawbacks & Limitations in Augmented Reality

- **Accuracy**

- When it comes to accuracy, currently the margin of error is large.
- For example, it is 10 meters on today's GPS systems and when you're trying to superimpose an image on another at close range, that is not accurate enough

- **Standards**

- No open standards among Augmented Reality browsers

- **Availability of Augmented Reality -capable devices**

- Smart Phones only due to availability of hardware components (GPS, camera, compass, 3D rendering capability, and more)

- It is not that natural to walk with your phone camera pointing everywhere.

- Some tools exist to avoid this problem such as augmented reality glasses (e.g. 3D Visors)

# References

- <http://wp.nmc.org/horizon2010/chapters/simple-augmented-reality/> Retrieved September 25, 2010.
- <http://www.bing.com/images/search?q=Augmented+reality+pictures&FORM=IGRE&qpvt=Augmented+realit+y+pictures>. Retrieved October 9, 2010
- <http://www.bing.com/images/search?q=Augmented+reality+pictures&FORM=IGRE&qpvt=Augmented+realit+y+pictures> Retrieved October 9, 2010.
- <http://www.howstuffworks.com/augmented-reality.htm>  
Retrieved October 7, 2010.
- <http://www.britannica.com/EBchecked/topic/1196641/augmented-reality>  
Retrieved October 7, 2010.
- <http://www.bing.com/images/search?q=Augmented+reality+pictures&FORM=IGRE&qpvt=Augmented+realit+y+pictures.#> Retrieved October 13, 2010.
- <http://www.newhorizons.org/strategies/technology/billinghurst.htm>  
Retrieved October 7, 2010.



Looks like they've switched to  
cloud computing

Modern Topics in IT(MTIT)  
4<sup>th</sup> Year – Semester 1  
By Isuru Jayakantha

# d Computing

# Cloud Computing

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services)

- It can be rapidly provisioned and released with minimal management effort.
- It provides high level abstraction of computation and storage model.
- It has some essential characteristics, service models, and deployment models.

# Basic Concepts

There are certain services and models working behind the scene making the cloud computing feasible and accessible to end users. Following are the working models for cloud computing:

- Service Models
- Deployment Models

# Service Models

## Cloud Software as a Service (SaaS):

- The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure.
- The applications are accessible from various client devices such as a web browser (e.g., web-based email)
- The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage.

**Examples:** Caspio, Google Apps, Salesforce, Nivio, Learn.com.

## Cloud Platform as a Service (PaaS):

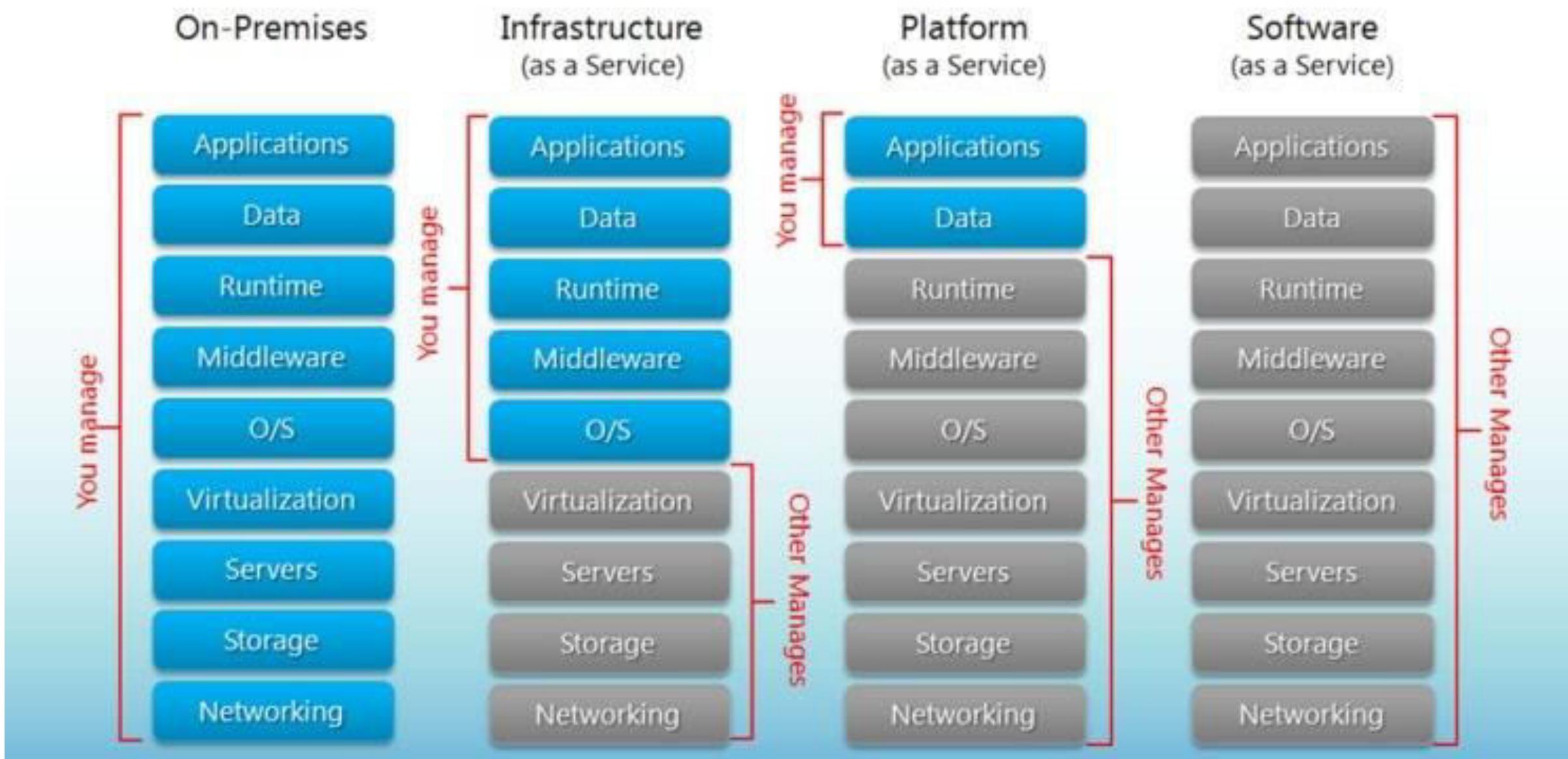
- The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages and tools supported by the provider.
- The consumer does not manage or control the underlying cloud infrastructure.
- Consumer has control over the deployed applications and possibly application hosting environment configurations.

Examples: Windows Azure, Google App Engine.

## Cloud Infrastructure as a Service (IaaS):

- The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources.
- The consumer is able to deploy and run arbitrary software, which can include operating systems and applications.
- The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of select networking components (e.g., host firewalls).

Examples: Amazon EC2, Go-Grid, Rackspace Cloud Servers, Relia Cloud.



# Deployment Models

Deployment models define the type of access to the cloud, i.e., how the cloud is located? Cloud can have any of the four types of access: Public, Private, Hybrid and Community.

## Private Cloud:

The cloud is operated solely for an organization. It may be managed by the organization or a third party and may exist on premise or off premise.

## Community Cloud:

The cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns.

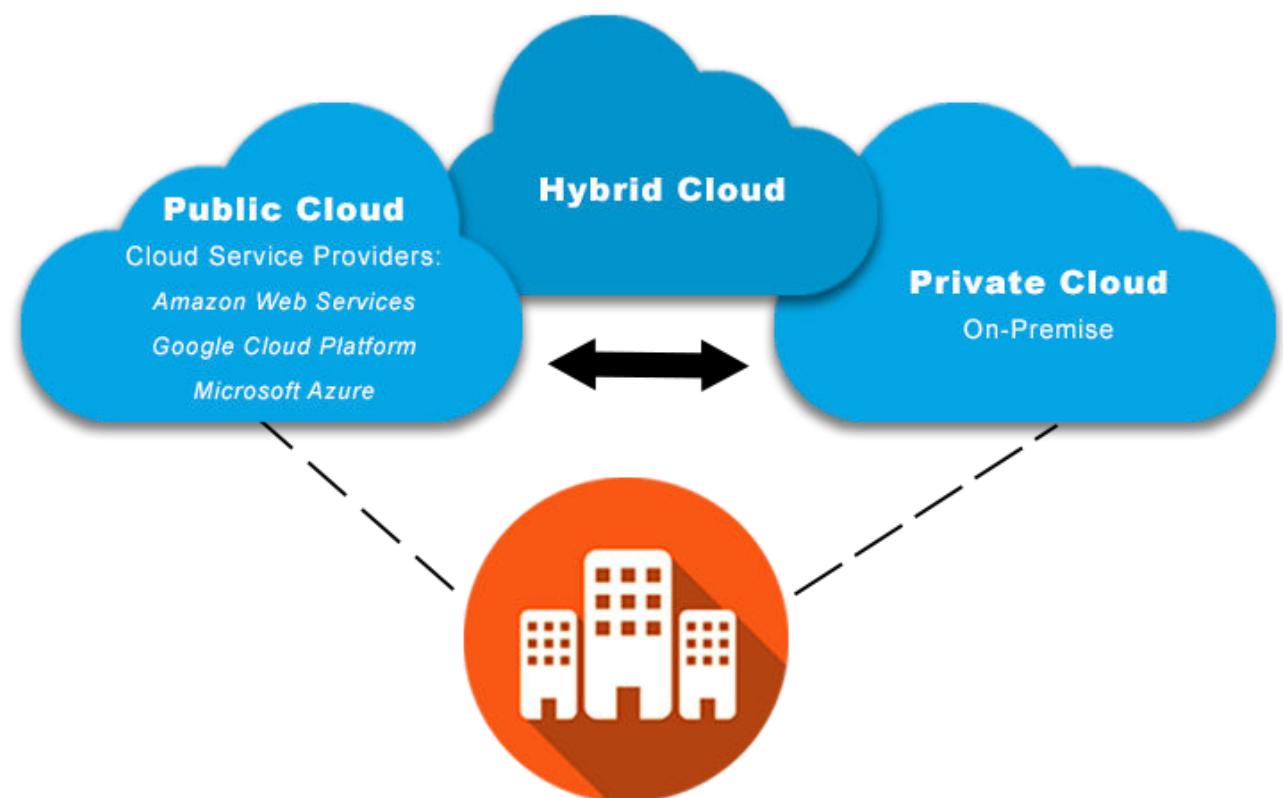
It may be managed by the organizations or a third party and may exist on premise or off premise.

## Public Cloud:

The cloud infrastructure is made available to the general public or a large industry group and it is owned by an organization selling cloud services.

# Hybrid cloud:

The cloud infrastructure is a composition of two or more clouds (private, community, or public).



# Advantages of Cloud Computing

- Cloud computing do not need high quality equipment for user, and it is very easy to use.
- Provides dependable and secure data storage center.
- Reduce run time and response time.
- Cloud is a large resource pool that you can buy on demand service.
- Scale of cloud can extend dynamically providing nearly infinite possibility for users to use internet.

# Cloud computing with AWS

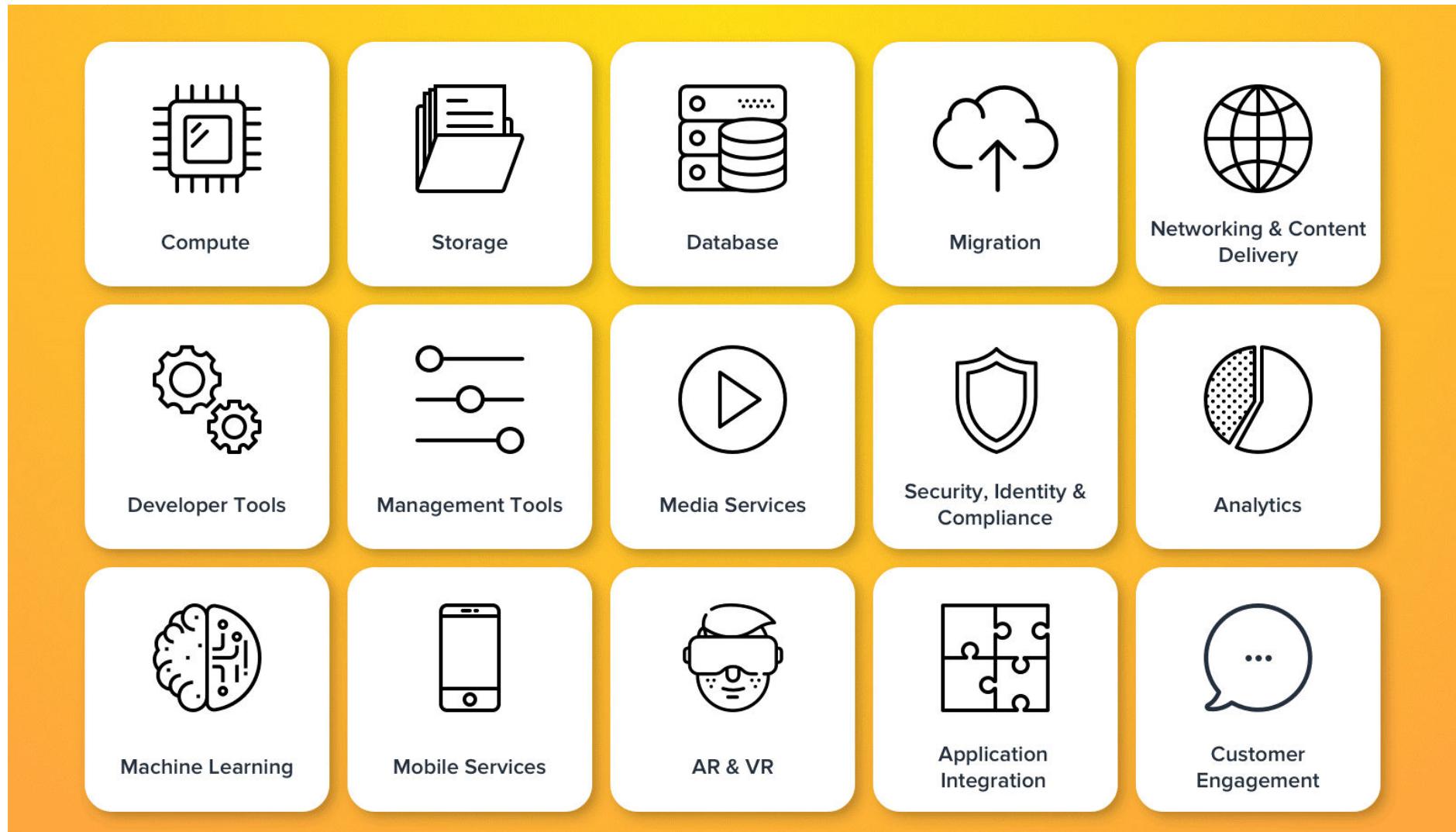
- Amazon Web Services (AWS) is the world's most comprehensive and broadly adopted cloud platform, offering over 175 fully featured services from data centers globally.
- Millions of customers - including the fastest-growing startups, largest enterprises, and leading government agencies - are using AWS to lower costs, become more agile, and innovate faster.

# History of AWS

- 2002- AWS services launched
- 2006- Launched its cloud products
- 2012- Holds first customer event
- 2015- Reveals revenues achieved of \$4.6 billion
- 2016- Surpassed \$10 billion revenue target
- 2016- Release snowball and snowmobile
- 2019- Offers nearly 100 cloud services
- 2020 - AWS comprised more than 212 services



# Important AWS Services



# AWS Compute Services

- **EC2(Elastic Compute Cloud)** - EC2 is a virtual machine in the cloud on which you have OS level control. You can run this cloud server whenever you want.
- **LightSail**-This cloud computing tool automatically deploys and manages the computer, storage, and networking capabilities required to run your applications.
- **Amazon Elastic Kubernetes Service (EKS : Elastic Container Service for Kubernetes)**-The tool allows you to Kubernetes on Amazon cloud environment without installation.
- **AWS Lambda**-This AWS service allows you to run functions in the cloud. The tool is a big cost saver for you as you pay only when your functions execute.
- **Elastic Beanstalk**- Easy-to-use service for deploying and scaling web applications and services developed with Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker.

# Migration

Migration services used to transfer data physically between your datacenter and AWS.

- **DMS (Database Migration Service)**-DMS service can be used to migrate on-site databases to AWS. It helps you to migrate from one type of database to another — for example, Oracle to MySQL.
- **SMS (Server Migration Service)**-SMS migration services allows you to migrate on-site servers to AWS easily and quickly.
- **Snowball**-Snowball is a small application which allows you to transfer terabytes of data inside and outside of AWS environment.

# Storage

- **Amazon Glacier**- It is an extremely low-cost storage service. It offers secure and fast storage for data archiving and backup.
- **Amazon Elastic Block Store (EBS)**- It provides block-level storage to use with Amazon EC2 instances. Amazon Elastic Block Store volumes are network-attached and remain independent from the life of an instance.
- **AWS Storage Gateway**- This AWS service is connecting on-premises software applications with cloud-based storage. It offers secure integration between the company's on-premises and AWS's storage infrastructure.

# Security Services

- **IAM (Identity and Access Management)** - IAM is a secure cloud security service which helps you to manage users, assign policies, form groups to manage multiple users.
- **Inspector** - It is an agent that you can install on your virtual machines, which reports any security vulnerabilities.
- **Certificate Manager** - The service offers free SSL certificates for your domains that are managed by Route53.
- **WAF (Web Application Firewall)** - WAF security service offers application-level protection and allows you to block SQL injection and helps you to block cross-site scripting attacks.

- **Cloud Directory**- This service allows you to create flexible, cloud-native directories for managing hierarchies of data along multiple dimensions.
- **KMS (Key Management Service)** - It is a managed service. This security service helps you to create and control the encryption keys which allows you to encrypt your data.
- **Organizations** - You can create groups of AWS accounts using this service to manage security and automation settings.
- **Shield** - Shield is managed DDoS (Distributed Denial of Service protection service). It offers safeguards against web applications running on AWS.
- **Macie** - It offers a data visibility security service which helps classify and protect your sensitive critical content.
- **GuardDuty** - It offers threat detection to protect your AWS accounts and workloads.

# Database Services

- **Amazon RDS** - This Database AWS service is easy to set up, operate, and scale a relational database in the cloud.
- **Amazon DynamoDB** - It is a fast, fully managed NoSQL database service. It is a simple service which allow cost-effective storage and retrieval of data. It also allows you to serve any level of request traffic.
- **Amazon ElastiCache** - It is a web service which makes it easy to deploy, operate, and scale an in-memory cache in the cloud.
- **Neptune** - It is a fast, reliable and scalable graph database service.
- **Amazon RedShift** - It is Amazon's data warehousing solution which you can use to perform complex OLAP queries.

# Analytics

- **Athena** - This analytics service allows perm SQL queries on your S3 bucket to find files.
- **CloudSearch** - You should use this AWS service to create a fully managed search engine for your website.
- **ElasticSearch** - It is similar to CloudSearch. However, it offers more features like application monitoring.
- **Kinesis** - This AWS analytics service helps you to stream and analyzing real-time data at massive scale.
- **QuickSight** - It is a business analytics tool. It helps you to create visualizations in a dashboard for data in Amazon Web Services. For example, S3, DynamoDB, etc.
- **EMR (Elastic Map Reduce)** - This AWS analytics service mainly used for big data processing like Spark, Splunk, Hadoop, etc.
- **Data Pipeline** - Allows you to move data from one place to another. For example from DynamoDB to S3

# Management Services

- **CloudWatch** - Cloud watch helps you to monitor AWS environments like EC2, RDS instances, and CPU utilization. It also triggers alarms depends on various metrics.
- **CloudFormation** - It is a way of turning infrastructure into the cloud. You can use templates for providing a whole production environment in minutes.
- **CloudTrail** - It offers an easy method of auditing AWS resources. It helps you to log all changes.
- **OpsWorks** - The service allows you to automated Chef/Puppet deployments on AWS environment.
- **Config** - This AWS service monitors your environment. The tool sends alerts about changes when you break certain defined configurations.

# Management Services

- **Service Catalog** - This service helps large enterprises to authorize which services user will be used and which won't.
- **AWS Auto Scaling** - The service allows you to automatically scale your resources up and down based on given CloudWatch metrics.
- **Systems Manager** - This AWS service allows you to group your resources. It allows you to identify issues and act on them.
- **Managed Services** - It offers management of your AWS infrastructure which allows you to focus on your applications.

# Internet of Things

- **IoT Core** - It is a managed cloud AWS service. The service allows connected devices like cars, light bulbs, sensor grids, to securely interact with cloud applications and other devices.
- **IoT Device Management** - It allows you to manage your IoT devices at any scale.
- **IoT Analytics** - This AWS IOT service is helpful to perform analysis on data collected by your IoT devices.
- **Amazon FreeRTOS** - This real-time operating system for microcontrollers helps you to connect IoT devices in the local server or into the cloud.

# Application Services

- **Step Functions** - It is a way of visualizing what's going inside your application and what different microservices it is using.
- **SWF (Simple Workflow Service)** - The service helps you to coordinate both automated tasks and human-led tasks.
- **SNS (Simple Notification Service)** - You can use this service to send you notifications in the form of email and SMS based on given AWS services.
- **SQS (Simple Queue Service)** - Use this AWS service to decouple your applications. It is a pull-based service.
- **Elastic Transcoder** - This AWS service tool helps you to changes a video's format and resolution to support various devices like tablets, smartphones of different resolutions.

# Deployment and Management

- **AWS CloudTrail** - The service records AWS API calls and send backlog files to you.
- **Amazon CloudWatch** - The tools monitor AWS resources like Amazon EC2 and Amazon RDS DB Instances. It also allows you to monitor custom metrics created by user's applications and services.
- **AWS CloudHSM** - This AWS service helps you meet corporate, regulatory, and contractual, compliance requirements for maintaining data security by using the Hardware Security Module(HSM) appliances inside the AWS environment.

# Developer Tools

- **CodeStar**- Codestar is a cloud-based service for creating, managing, and working with various software development projects on AWS.
- **CodeCommit**- It is AWS's version control service which allows you to store your code and other assets privately in the cloud.
- **CodeBuild**- This Amazon developer service help you to automates the process of building and compiling your code.
- **CodeDeploy**-It is a way of deploying your code in EC2 instances automatically.
- **CodePipeline**-It helps you create a deployment pipeline like testing, building, testing, authentication, deployment on development and production environments.
- **Cloud9** - It is an Integrated Development Environment for writing, running, and debugging code in the cloud.

# Mobile Services

- **Mobile Hub** - Allows you to add, configure and design features for mobile apps.
- **Cognito**- Allows users to signup using his or her social identity.
- **Device Farm**- Device farm helps you to improve the quality of apps by quickly testing hundreds of mobile devices.
- **AWS AppSync** - It is a fully managed GraphQL service that offers real-time data synchronization and offline programming features.

# Business Productivity

- **Alexa for Business** - It empowers your organization with voice, using Alexa. It will help you to Allows you to build custom voice skills for your organization.
- **Chime** - Can be used for online meeting and video conferencing.
- **WorkDocs** - Helps to store documents in the cloud
- **WorkMail** - Allows you to send and receive business emails.

# Desktop & App Streaming

- **Work Spaces** - Workspace is a VDI (Virtual Desktop Infrastructure). It allows you to use remote desktops in the cloud.
- **AppStream** - A way of streaming desktop applications to your users in the web browser. For example, using MS Word in Google Chrome.

# Artificial Intelligence

- **Lex** - Lex tool helps you to build chatbots quickly.
- **Polly** - It is AWS's text-to-speech service allows you to create audio versions of your notes.
- **Rekognition** - It is AWS's face recognition service. This AWS service helps you to recognize faces and object in images and videos.
- **SageMaker** - Sagemaker allows you to build, train, and deploy machine learning models at any scale.
- **Transcribe** - It is AWS's speech-to-text service that offers high-quality and affordable transcriptions.
- **Translate** - It is a very similar tool to Google Translate which allows you to translate text in one language to another.

# AR & VR (Augmented Reality & Virtual Reality)

- **Sumerian** - Sumerian is a set of tool for offering high-quality virtual reality (VR) experiences on the web. The service allows you to create interactive 3D scenes and publish it as a website for users to access.

## Customer Engagement

- **Amazon Connect** - Amazon Connect allows you to create your customer care center in the cloud.
- **Pinpoint**-Pinpoint helps you to understand your users and engage with them.
- **SES (Simple Email Service)**-Helps you to send bulk emails to your customers at a relatively cost-effective price.

# Game Development

- **Game Lift** - It is a service which is managed by AWS. You can use this service to host dedicated game servers. It allows you to scale seamlessly without taking your game offline.
- **Amazon Lumberyard** – is a free cross-platform game engine developed by Amazon and based on CryEngine.



# Applications of AWS services

- Web site hosting
- Application hosting/SaaS hosting
- Media Sharing (Image/ Video)
- Mobile and Social Applications
- Content delivery and Media Distribution
- Storage, backup, and disaster recovery
- Development and test environments
- Academic Computing
- Search Engines
- Social Networking

# Companies using AWS

- Instagram
- Zoopla
- Smugmug
- Pinterest
- Netflix
- Dropbox
- Etsy
- Talkbox
- Supercell



# Advantages of AWS

- AWS allows organizations to use the already familiar programming models, operating systems, databases, and architectures.
- It is a cost-effective service that allows you to pay only for what you use, without any up-front or long-term commitments.
- You will not require to spend money on running and maintaining data centers.
- Offers fast deployments.
- You can easily add or remove capacity.

# Advantages of AWS

- You are allowed cloud access quickly with limitless capacity.
- Total Cost of Ownership is very low compared to any private/dedicated servers.
- Offers Centralized Billing and management
- Offers Hybrid Capabilities
- Allows you to deploy your application in multiple regions around the world with just a few clicks

# Disadvantages of AWS

- If you need more immediate or intensive assistance, you'll have to opt for paid support packages.
- Amazon Web Services may have some common cloud computing issues when you move to a cloud. For example, downtime, limited control, and backup protection.
- AWS sets default limits on resources which differ from region to region. These resources consist of images, volumes, and snapshots.
- Hardware-level changes happen to your application which may not offer the best performance and usage of your applications.

# Best practices of AWS

- You need to design for failure, but nothing will fail.
- It's important to decouple all your components before using AWS services.
- You need to keep dynamic data closer to compute and static data closer to the user.
- It's important to know security and performance tradeoffs.
- Pay for computing capacity by the hourly payment method.

# AWS Certifications

**aws  CERTIFIED**

Professional



Associate



Foundational



## Role-Based Certifications



## Specialty Certifications



# References

- <https://aws.amazon.com/>
- [https://aws.amazon.com/what-is-aws/?nc1=f\\_cc](https://aws.amazon.com/what-is-aws/?nc1=f_cc)



# **Code Refactoring Techniques**

**Modern Topics in Information Technology**  
**4<sup>th</sup> Year – Semester 1**  
**Lecture 5**  
**By Udara Samaratunge**

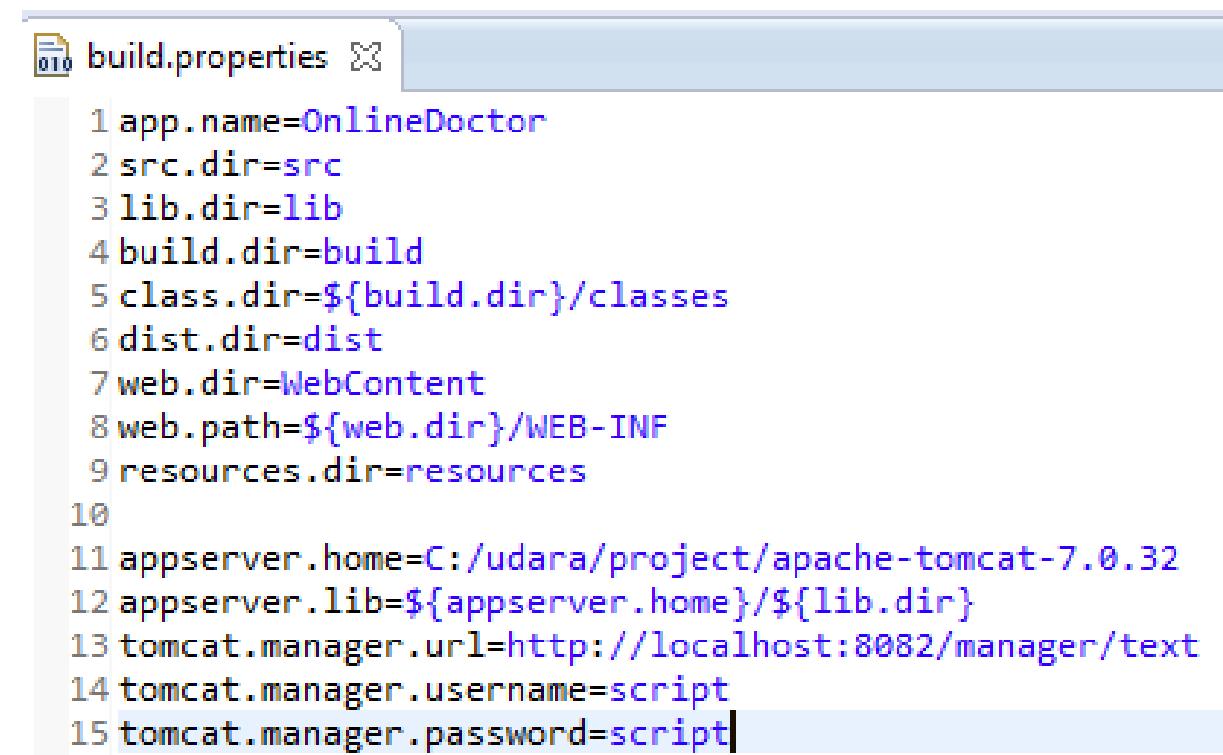
# Usage of Properties

- **.properties** is a **file extension** for **files** mainly used in Java related technologies
- This is used to store configurable parameters of an application.
- They can also be used for storing strings for Internationalization and localization.
- Known as Property Resource Bundles.
- Properties are configuration values managed as key/value pairs.
- In each pair, the key and value are both **String** values.

# Usage of Properties

- Properties extends [java.util.Hashtable](#). Some of the methods inherited from Hash Table support the following actions:

- Testing to see if a particular key or value is in the Properties object
- Getting the current number of key/value pairs
- Removing a key and its value
- Adding a key/value pair to the Properties list
- Enumerating over the values or the keys
- Retrieving a value by its key
- Finding out if the Properties object is empty.



A screenshot of a code editor window titled "build.properties". The file contains the following configuration properties:

```
1 app.name=OnlineDoctor
2 src.dir=src
3 lib.dir=lib
4 build.dir=build
5 class.dir=${build.dir}/classes
6 dist.dir=dist
7 web.dir=WebContent
8 web.path=${web.dir}/WEB-INF
9 resources.dir=resources
10
11 appserver.home=C:/udara/project/apache-tomcat-7.0.32
12 appserver.lib=${appserver.home}/${lib.dir}
13 tomcat.manager.url=http://localhost:8082/manager/text
14 tomcat.manager.username=script
15 tomcat.manager.password=script|
```

# Read property file

```
/**  
 * Read a property file and its key value content  
 */  
public static void readProperty(){  
  
    FileInputStream fIstream = null;  
    Properties property = new Properties();  
  
    try {  
        fIstream = new FileInputStream(FILE_PATH);  
        property.load(fIstream);  
        System.out.println(property.getProperty(SERVER_URL));  
        System.out.println(property.getProperty(SERVER_USERNAME));  
        System.out.println(property.getProperty(SERVER_PASSWORD));  
  
    } catch (FileNotFoundException e) {  
        e.printStackTrace();  
    } catch (IOException e) {  
        e.printStackTrace();  
    } finally{  
        if(fIstream != null){  
            try {  
                fIstream.close();  
            } catch (IOException e) {  
                e.printStackTrace();  
            }  
        }  
    }  
}  
  
//Constant for Property file path  
public static final String FILE_PATH = "src\\Config.properties";  
//Constant to read Property file for server url  
public static final String SERVER_URL = "tomcat.manager.url";  
//Constant to read Property file for server username  
public static final String SERVER_USERNAME = "tomcat.manager.username";  
//Constant to read Property file for server password  
public static final String SERVER_PASSWORD = "tomcat.manager.password";
```

## Out put

```
<terminated> ReadProperty [Java Application] C:\Program Files\Java\  
http://localhost:8082/manager/text  
admin  
admin123
```

# Best Practices

```
public class ReadProperty {  
  
    //Constant for Property file path  
    public static final String FILE_NAME = "Config.properties"; ← File Name  
    //Constant to read Property file for server url  
    public static final String SERVER_URL = "tomcat.manager.url";  
    //Constant to read Property file for server username  
    public static final String SERVER_USERNAME = "tomcat.manager.username";  
    //Constant to read Property file for server password  
    public static final String SERVER_PASSWORD = "tomcat.manager.password";  
  
    /**  
     * Read a property file and its key value content  
     */  
    public static void readProperty(){  
  
        Properties property = new Properties();  
  
        try {  
            property.load(ReadProperty.class.getClassLoader().getResourceAsStream(FILE_NAME)); ← Use InputStream and access through class path  
            System.out.println(property.getProperty(SERVER_URL));  
            System.out.println(property.getProperty(SERVER_USERNAME));  
            System.out.println(property.getProperty(SERVER_PASSWORD));  
  
        } catch (FileNotFoundException e) {  
            e.printStackTrace();  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

# Generate property file

```
/**  
 * Create property file to store build.xml configurations  
 */  
  
public static void createProperty() {  
    try {  
        Properties properties = new Properties();  
        properties.setProperty(APP_NAME, "OnlineDoctor");  
        properties.setProperty(SOURCE_DIR, "src");  
        properties.setProperty(BUILD_DIR, "build");  
        properties.setProperty(WEB_DIR, "WebContent");  
        properties.setProperty(WEB_PATH, "${web.dir}/WEB-INF");  
        properties.save(new FileOutputStream(PROPERTY_FILE_NAME),  
                      COMMENT);  
    } catch (FileNotFoundException e) {  
        e.printStackTrace();  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
}
```

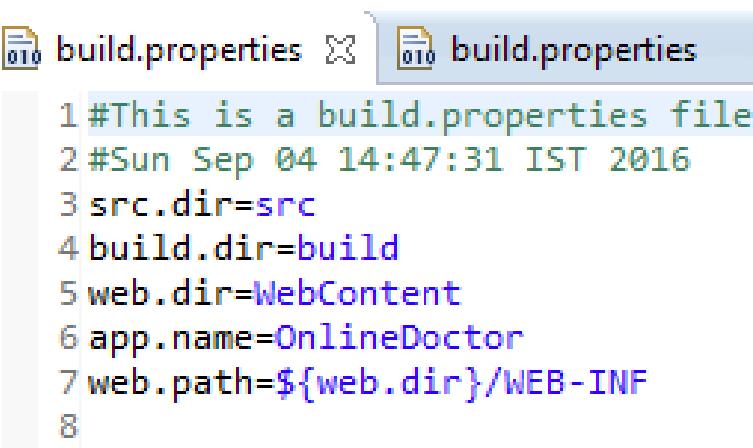
Don't use **Deprecated** methods.  
Instead search the most suitable  
method

Instead use **store()**  
method

# Best Practice

```
/*
 * Create property file to store build.xml configurations
 */
public static void createProperty() {
    try {
        Properties properties = new Properties();
        properties.setProperty(APP_NAME, "OnlineDoctor");
        properties.setProperty(SOURCE_DIR, "src");
        properties.setProperty(BUILD_DIR, "build");
        properties.setProperty(WEB_DIR, "WebContent");
        properties.setProperty(WEB_PATH, "${web.dir}/WEB-INF");
        properties.store(new FileOutputStream(PROPERTY_FILE_NAME),
                       COMMENT);
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

```
// Constant to write Property file name
public static final String PROPERTY_FILE_NAME = "build.properties";
// Constant to write Property file for Application name
public static final String APP_NAME = "app.name";
// Constant to write Property file for source location
public static final String SOURCE_DIR = "src.dir";
// Constant to write Property file for build directory
public static final String BUILD_DIR = "build.dir";
// Constant to write Property file for web content
public static final String WEB_DIR = "web.dir";
// Constant to write Property file for web path
public static final String WEB_PATH = "web.path";
//Constant to write comment for build.properties
public static final String COMMENT = "This is a build.properties file";
```



```
build.properties build.properties
#This is a build.properties file
#Sun Sep 04 14:47:31 IST 2016
src.dir=src
build.dir=build
web.dir=WebContent
app.name=OnlineDoctor
web.path=${web.dir}/WEB-INF
8
```

```
public class PropertyToXML {  
  
    private static final String INSTITIUTE = "SLIIT";  
    private static final String NAME = "Udara";  
    private static final String MODULE = "MTIT";  
    private static final String YEAR = "4th Year";  
    private static final String FIELD = "SE";  
    private static final String COMMENT = "This is a XML file generated through properties";  
    private static final String PROPERTY_FILE_NAME = "Properties.xml";  
  
    /**  
     * Create property file to store Properties.xml configurations  
     */  
    public static void createProperty() {  
        try {  
            Properties properties = new Properties();  
            properties.setProperty("INSTITIUTE", INSTITIUTE);  
            properties.setProperty("NAME", NAME);  
            properties.setProperty("MODULE", MODULE);  
            properties.setProperty("YEAR", YEAR);  
            properties.setProperty("FIELD", FIELD);  
            properties.storeToXML(new FileOutputStream(PROPERTY_FILE_NAME),  
                COMMENT);  
        } catch (FileNotFoundException e) {  
            e.printStackTrace();  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
  
    public static void main(String[] args) {  
        createProperty();  
    }  
}
```

# Property to XML File

Properties.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>  
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">  
<properties>  
    <comment>This is a XML file generated through properties</comment>  
    <entry key="MODULE">MTIT</entry>  
    <entry key="INSTITIUTE">SLIIT</entry>  
    <entry key="NAME">Udara</entry>  
    <entry key="FIELD">SE</entry>  
    <entry key="YEAR">4th Year</entry>  
</properties>
```

# Pass parameters for Property file

- Use `java.text.MessageFormat`
- `MessageFormat` provides a means to produce concatenated messages in a language-neutral way. Use this to construct messages displayed for end users.
- `MessageFormat` takes a set of objects, formats them, then inserts the formatted strings into the pattern at the appropriate places.
- **`String java.text.MessageFormat.format(String pattern, Object... arguments)`**
- You can pass any number of arguments for the property file using `MessageFormat` and you have to pass the String Pattern with specifying indexes in curly braces to display passed parameters. Eg:- `{0}, {1}, {2}...etc`

configuration.properties

```
1 message=Error code is {0}. And Error message is {1}
2 message2=My name is {0} and works in {1}. My subject is {2}.
```

Create  
Configuration.properties

```
private static final String ERROR_CODE = "4008";
private static final String ERROR_MESSAGE = "Custom Error Message";
private static final String MESSAGE_KEY = "message";
private static final String MESSAGE_KEY_2 = "message2";
private static final String NAME = "Udara";
private static final String PLACE = "SLIIT";
private static final String SUBJECT = "MTIT";

private static final String PROPERTY_FILE = "configuration.properties";
private static Properties properties = null;

static{
    properties = new Properties();
    try {
        properties.load(PropertyParam.class.getResourceAsStream(PROPERTY_FILE));
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Load  
Configuration.properties

```
String ErrorMsg = MessageFormat.format((String)properties.get(MESSAGE_KEY), ERROR_CODE, ERROR_MESSAGE);
System.out.println(ErrorMsg);
String MyDetailsMsg = MessageFormat.format((String)properties.get(MESSAGE_KEY_2), NAME, PLACE, SUBJECT);
System.out.println(MyDetailsMsg);
```



```
Error code is 4008. And Error message is Custom Error Message
My name is Udara and works in SLIIT. My subject is MTIT.
```

→ **Response**

# **XML Manipulations**

# Embed Queries into XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
BiharGlass HQL queries
-->
<Querys>

    <!-- Select all customers Query -->
<query id="c1">
    <![CDATA[
        select customer from Customer as customer
    ]]>
</query>

    <!-- Select a particular customer Query -->
<query id="c2">
    <![CDATA[
        from Customer as customer where customer.customerId = ?
    ]]>
</query>

    <!-- Select all customer orders Query -->
<query id="co1">
    <![CDATA[
        select orders from CustomerOrder as orders
    ]]>
</query>

    <!-- Select a particular customer order Query -->
<query id="co2">
    <![CDATA[
        from CustomerOrder as orders where orders.customerOrderId = ?
    ]]>
</query>

    <!-- Confirm allocations related to MO Query -->
<query id="al4">
    <![CDATA[
        update Allocation a
        set a.status = ?
        where a.referenceOrderId = ?
    ]]>
</query>

    <!-- Get all material line demand query -->
<query id="ds1">
    <![CDATA[
        select sum(li.materialQuantity)
        from ManufacturingOrderLine as li
        where li.status = ? and li.material.materialId = ?
        group by li.material.materialId
    ]]>
</query>

    <!-- Get all remaining quantity of inventory query -->
<query id="ds2">
    <![CDATA[
        select sum(i.remainingQty)
        from Inventory as i
        where i.materialId = ?
        group by i.materialId
    ]]>
</query>
```

# HQL Queries in a XML

```
228 <!-- Select allocated quantity from inventory Query -->
229 <query id="in2">
230   <![CDATA[
231     select inventory.receivedQuantity from Inventory as inventory where inventory.lotNumber = ?
232   ]]>
233 </query>
234
235 <!-- Select list of items Lot wise Query -->
236 <query id="in3">
237   <![CDATA[
238     select inventory from Inventory as inventory
239   ]]>
240 </query>
241
242 <!-- Select list of elements in item wise Query -->
243 <query id="in4">
244   <![CDATA[
245     select new List(inventory.materialId, sum(inventory.receivedQuantity), inventory.remainingQty,
246                   sum(inventory.allocatedQuantity), inventory.receiptDate, inventory.allocatable, inventory.referenceOrderId)
247     from Inventory as inventory group by inventory.materialId
248   ]]>
249 </query>
250
251 <!-- Get item by ID Query -->
252 <query id="in5">
253   <![CDATA[
254     select inventory from Inventory as inventory where inventory.lotNumber = ?
255   ]]>
256 </query>
```

# Read Elements in XML query

```
public class QueryUtil {  
    /*  
     * This method read the SPARQLQuery.xml file and retrieve the query by query id.  
     */  
    public static String queryByID(String id)  
    {  
        NodeList nodeList;  
        Element element = null;  
        try  
        {  
            nodeList = DocumentBuilderFactory  
                .newInstance()  
                .newDocumentBuilder()  
                .parse(new File(System.getProperty("catalina.base") + OntologyConstant.SPARQL_FILE))  
                .getElementsByTagName("query");  
  
            for (int x = 0; x < nodeList.getLength(); x++) {  
                element = (Element)nodeList.item(x);  
                if(element.getAttribute("id").equals(id))break;  
            }  
        }  
        catch (Exception ex){  
            ex.printStackTrace();  
        }  
        return element.getTextContent();  
    }  
}
```

```
<Queries>  
    <!-- Create table employee Query -->  
    <query id="create_employee_table">  
        <![CDATA[  
            CREATE TABLE employees(  
                EmployeeID varchar(10) not null,  
                Fullname varchar(25),  
                Address varchar(50),  
                Faculty varchar(20),  
                Department varchar(25),  
                Designation varchar(20),  
                primary key (EmployeeID)  
            )  
        ]]>  
    </query>  
    <!-- DROP TABLE EMPLOYEES Query -->  
    <query id="drop_table">  
        <![CDATA[  
            DROP TABLE IF EXISTS employees  
        ]]>  
    </query>
```

```
/*
 * Get employee by ID query will be retrieved from
 * EmployeeQuery.xml
 */
PreparedStatement = connection.prepareStatement(QueryUtil
    .queryByID(CommonConstants.QUERY_ID_GET_EMPLOYEE));
PreparedStatement.setString(CommonConstants.COLUMN_INDEX_ONE, employeeID);
```

**Employee ID will be assed as a parameter for xml element**

```
<!-- Select a particular employee by id Query -->
<query id="employee_by_id">
    <![CDATA[
        select * from employees where employees.EmployeeID = ?
    ]]>
</query>
```

# Inject Parameter to XML file

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- JSON file -->
<Requests>
    <request id="createUser">
        <![CDATA[
            {
                "user": {
                    "name": "Udara Samaratunge",
                    "email": "%1$s"
                }
            }
        ]]>
    </request>
</Requests>
```

```
public class RequestUtil {

    private static final String REQUEST_FILE_PATH = "src/RestPostRequests.xml";
    private static final String REQUEST = "request";
    private static final String ID = "id";

    /**
     * This method read the RestPostRequests.xml file and retrieve the query by
     * query id.
     *
     * @param type
     * @return
     */
    public static String requestByID(String id) {

        NodeList nodeList;
        Element element = null;

        try {
            nodeList = DocumentBuilderFactory.newInstance().newDocumentBuilder()
                .parse(new File(REQUEST_FILE_PATH)).getElementsByName(REQUEST);

            for (int x = 0; x < nodeList.getLength(); x++) {
                element = (Element) nodeList.item(x);
                if (element.getAttribute(ID).equals(id))
                    break;
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        return element.getTextContent();
    }
}
```

```
private static final String CREATE_USER = "createUser";
private static final String EMAIL = "udara.s@sliit.lk";
private static final String PROPERTY_FILE_NAME = "Properties.xml";

/**
 * Add parameter to XML file
 */
private static void addParamToXML(){

    System.out.println("=====Before=====");
    System.out.println(RequestUtil.requestByID(CREATE_USER).trim());
    System.out.println("=====After=====");
    String message = String.format(RequestUtil.requestByID(CREATE_USER), EMAIL).trim();
    System.out.println(message);
}
```

## Response

```
<terminated> PropertyToXML [Java Application] C:\Program Files\Java\jdk1
=====
Before=====
{
    "user": {
        "name": "Udara Samaratunge",
        "email": "%1$s"
    }
}
=====
After=====
{
    "user": {
        "name": "Udara Samaratunge",
        "email": "udara.s@sliit.lk"
    }
}
```

```
private static void buildXML() throws Exception {  
  
    Document doc = DocumentBuilderFactory.newInstance()  
        .newDocumentBuilder().newDocument();  
  
    Element rootElement = doc.createElement("cars");  
    doc.appendChild(rootElement);  
    Element supercar = doc.createElement("supercars");  
    rootElement.appendChild(supercar);  
  
    // setting attribute to element  
    Attr attr = doc.createAttribute("company");  
    attr.setValue("Ferrari");  
    supercar.setAttributeNode(attr);  
  
    // carname element  
    Element carname = doc.createElement("carname");  
    Attr attrType = doc.createAttribute("type");  
    attrType.setValue("formula one");  
    carname.setAttributeNode(attrType);  
    carname.appendChild(doc.createTextNode("Ferrari 101"));  
    supercar.appendChild(carname);  
  
    Element carname1 = doc.createElement("carname");  
    Attr attrType1 = doc.createAttribute("type");  
    attrType1.setValue("sports");  
    carname1.setAttributeNode(attrType1);  
    carname1.appendChild(doc.createTextNode("Ferrari 202"));  
    supercar.appendChild(carname1);  
  
    // write the content into xml file  
    Transformer transformer = TransformerFactory.newInstance().newTransformer();  
    DOMSource source = new DOMSource(doc);  
    transformer.transform(source, new StreamResult(new File("cars.xml")));  
    transformer.transform(source, new StreamResult(System.out));  
}
```

# Generate XML file

Response

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>  
<cars>  
    <supercars company="Ferrari">  
        <carname type="formula one">Ferrari 101</carname>  
        <carname type="sports">Ferrari 202</carname>  
    </supercars>  
</cars>
```

# **Separation of Logic and refactor considering cohesion and coupling**

```
private static void buildXML() throws Exception {  
  
    Document doc = DocumentBuilderFactory.newInstance()  
        .newDocumentBuilder().newDocument();  
  
    Element rootElement = doc.createElement("cars");  
    doc.appendChild(rootElement);  
    Element supercar = doc.createElement("superCars");  
    rootElement.appendChild(supercar);  
  
    // setting attribute to element  
    Attr attr = doc.createAttribute("company");  
    attr.setValue("Ferrari");  
    supercar.setAttributeNode(attr);  
  
    // carname element  
    Element carname = doc.createElement("carname");  
    Attr attrType = doc.createAttribute("type");  
    attrType.setValue("formula one");  
    carname.setAttributeNode(attrType);  
    carname.appendChild(doc.createTextNode("Ferrari 101"));  
    supercar.appendChild(carname);  
  
    Element carname1 = doc.createElement("carname");  
    Attr attrType1 = doc.createAttribute("type");  
    attrType1.setValue("sports");  
    carname1.setAttributeNode(attrType1);  
    carname1.appendChild(doc.createTextNode("Ferrari 202"));  
    supercar.appendChild(carname1);  
  
    // write the content into xml file  
    Transformer transformer = TransformerFactory.newInstance().newTransformer();  
    DOMSource source = new DOMSource(doc);  
    transformer.transform(source, new StreamResult(new File("cars.xml")));  
    transformer.transform(source, new StreamResult(System.out));  
}
```

# What is the Problem ???

Logics were not separated

Cohesiveness of the method?

```
private static void buildXML() throws Exception {  
  
    Document doc = DocumentBuilderFactory.newInstance()  
        .newDocumentBuilder().newDocument();  
  
    Element rootElement = doc.createElement("cars");  
    doc.appendChild(rootElement);  
    Element supercar = doc.createElement("supercars");  
    rootElement.appendChild(supercar);  
  
    // setting attribute to element  
    Attr attr = doc.createAttribute("company");  
    attr.setValue("Ferrari");  
    supercar.setAttributeNode(attr);  
  
    // carname element  
    Element carname = doc.createElement("carname");  
    Attr attrType = doc.createAttribute("type");  
    attrType.setValue("formula one");  
    carname.setAttributeNode(attrType);  
    carname.appendChild(doc.createTextNode("Ferrari 101"));  
    supercar.appendChild(carname);  
  
    Element carname1 = doc.createElement("carname");  
    Attr attrType1 = doc.createAttribute("type");  
    attrType1.setValue("sports");  
    carname1.setAttributeNode(attrType1);  
    carname1.appendChild(doc.createTextNode("Ferrari 202"));  
    supercar.appendChild(carname1);  
  
    // write the content into xml file  
    Transformer transformer = TransformerFactory.newInstance().newTransformer();  
    DOMSource source = new DOMSource(doc);  
    transformer.transform(source, new StreamResult(new File("cars.xml")));  
    transformer.transform(source, new StreamResult(System.out));  
}
```

# Identify Logics to be separated

1. Create XML document builder
2. Create Element
3. Add name to element
4. Create Attributes
5. Add attributes to Element
6. Transform response to XML

# Separation of Logics

```
/*
 * Create Instance of Document builder
 *
 * @return
 * @throws ParserConfigurationException
 */
public Document buildDocument() throws ParserConfigurationException {
    return DocumentBuilderFactory.newInstance().newDocumentBuilder()
        .newDocument();
}

/**
 * Create an XML Element
 *
 * @param doc
 * @param elementName
 * @return
 */
public Element createElement(Document doc, String elementName) {
    return doc.createElement(elementName);
}

/**
 * Create attribute for the element
 *
 * @param doc
 * @param type
 * @param value
 * @return
 */
public Attr createAttribute(Document doc, String type, String value) {
    Attr attribute = doc.createAttribute(type);
    attribute.setValue(value);
    return attribute;
}
```

1. Create XML instance

2. Create Elements

3. Create Attributes

```
/*
 * Create Node name for the element and Append
 *
 * @param doc
 * @param element
 * @param textNode
 * @return
 */
public Element appendTextNode(Document doc, Element element, String textNode) {
    element.appendChild(doc.createTextNode(textNode));
    return element;
}

/**
 * Set attributes for elements
 *
 * @param element
 * @param attribute
 * @return
 */
public Element setAttributeForElement(Element element, Attr attribute) {
    element.setAttributeNode(attribute);
    return element;
}

/*
 * Write the content into xml file
 *
 * @param doc
 * @throws TransformerFactoryConfigurationError
 * @throws TransformerException
 */
public void transformToXML(Document doc)
    throws TransformerFactoryConfigurationError, TransformerException {
    Transformer transformer = TransformerFactory.newInstance().newTransformer();
    DOMSource source = new DOMSource(doc);
    transformer.transform(source, new StreamResult(new File("cars.xml")));
    // Output to console for testing
    transformer.transform(source, new StreamResult(System.out));
}
```

#### 4. Add name to element

#### 5. Add attributes to Element

#### 6. Transform response to XML

Build XML file invoking implemented methods and append child elements accordingly

```
/**  
 * Build XML file  
 *  
 * @throws ParserConfigurationException  
 * @throws TransformerException  
 * @throws TransformerFactoryConfigurationError  
 * @throws Exception  
 */  
  
public void buildXMLfile() throws ParserConfigurationException,  
    TransformerFactoryConfigurationError, TransformerException {  
  
    Document doc = buildDocument();  
    Element supercar = setAttributeForElement(  
        createElement(doc, "supercars"),  
        createAttribute(doc, "company", "Ferrari"));  
    supercar.appendChild(appendTextNode(  
        doc,  
        setAttributeForElement(createElement(doc, "classname"),  
            createAttribute(doc, "type", "formula one")),  
        "Ferrari 101"));  
    supercar.appendChild(appendTextNode(  
        doc,  
        setAttributeForElement(createElement(doc, "classname"),  
            createAttribute(doc, "type", "sports")), "Ferrari 202"));  
    doc.appendChild(createElement(doc, "cars")).appendChild(supercar);  
    // Display transformed output  
    transformToXML(doc);  
}
```

## Response

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>  
<cars>  
    <supercars company="Ferrari">  
        <classname type="formula one">Ferrari 101</classname>  
        <classname type="sports">Ferrari 202</classname>  
    </supercars>  
</cars>
```

# Usage of Generics

```
public abstract class DBDriverBase<T> {  
  
    /**  
     * Initializes a session from a SessionFactoryUtil class  
     */  
    private Session session = SessionFactoryUtil.getSessionFactory().openSession();  
  
    /**  
     * Method gets a T object and saves it in the DB using Hibernate  
     * @param T object  
     * @return Boolean  
     * @exception throws {@link RuntimeException} and {@link HibernateException}  
     */  
  
    public boolean save(T object){  
        Transaction tx = null;  
        try{  
            tx = session.beginTransaction();  
            session.save(object);  
            tx.commit();  
            return true;  
        }catch(RuntimeException ex){  
            if(tx != null && tx.isActive())  
            {  
                try{  
                    tx.rollback();  
                }catch(HibernateException he){  
                    System.err.println("Error rolling back transaction");  
                    he.printStackTrace();  
                }  
            }  
            System.err.println("Error in saving the object in the database");  
            ex.printStackTrace();  
            return false;  
        }  
    }  
}
```

- Implementation of **Generic templates** is good practice in implementation



# Bad Practice

```
public class AdminPermissionrequestDBDriver extends DBDriverBase<AdminPermissionrequest> {  
  
    Session session = SessionFactoryUtil.getSessionFactory().openSession();  
  
    /** * Add new Permissions to the Database  
     * @param rpObj is a RolePermission type object that contains data about new permission  
     * @return a boolean value. Returns true if new permission inserted successfully else returns false  
     */  
  
    public boolean inserPermissionRequest(AdminPermissionrequest rpObj){  
  
        Transaction tx = null;  
        try{  
            tx = session.beginTransaction();  
            session.save(rpObj);  
            tx.commit();  
            return true;  
        }catch(RuntimeException e){  
            if(tx != null && tx.isActive())  
            {  
                try{  
                    tx.rollback();  
                }catch(HibernateException he){  
                    System.err.println("Error rolling back transaction");  
                }  
                throw ex;  
            }  
        }  
        return false;  
    }  
}
```

- Generic template NOT USED properly



```
public class AdminPermissionrequestDBDriver extends DBDriverBase<AdminPermissionrequest> {  
  
    Session session = SessionFactoryUtil.getSessionFactory().openSession();  
  
    /** * Add new Permissions to the Database  
     * @param rpObj is a RolePermission type object that contains data about new permission  
     * @return a boolean value. Returns true if new permission inserted successfully else returns false  
     */  
  
    public boolean inserPermissionRequest(AdminPermissionrequest rpObj){  
  
        return save(rpObj);  
    }  
}
```



# Required Code Refactoring

```
public class ReceiptDBDriver extends DBDriverBase<HisFinReceipt> {  
  
    Session session = SessionFactoryUtil.getSessionFactory().getCurrentSession();  
  
    public boolean addReceipt(HisFinReceipt receipt){  
        if(save(receipt))  
            return true;  
        else  
            return false;  
    }  
  
    public boolean updateReceipt(HisFinReceipt receipt){  
        if(update(receipt))  
            return true;  
        else  
            return false;  
    }  
  
    public boolean deleteReceipt(HisFinReceipt receipt){  
        if(delete(receipt))  
            return true;  
        else  
            return false;  
    }  
}
```



```
public class ReceiptDBDriver extends DBDriverBase<HisFinReceipt> {  
  
    Session session = SessionFactoryUtil.getSessionFactory().getCurrentSession();  
  
    public boolean addReceipt(HisFinReceipt receipt){  
        return save(receipt);  
    }  
  
    public boolean updateReceipt(HisFinReceipt receipt){  
        return update(receipt);  
    }  
  
    public boolean deleteReceipt(HisFinReceipt receipt){  
        return delete(receipt);  
    }  
}
```



- Understood the generic design concept.
- But code was not refactored.

# Generic Implementation

```
GenericMap<String, Employee> employeeMap = new GenericMap<String, Employee>();
employeeMap.put("3000", new Employee("Ann", 20));
employeeMap.put("3001", new Employee("Sam", 25));
employeeMap.put("3002", new Employee("Jon", 30));
employeeMap.put("3003", new Employee("Jim", 35));
employeeMap.put("3004", new Employee("Tom", 40));

employeeMap.displayMap(employeeMap);
System.out.println();

GenericMap<Integer, Integer> graphCoordinates = new GenericMap<Integer, Integer>();
graphCoordinates.put(2, 3);
graphCoordinates.put(10, 20);
graphCoordinates.put(-20, 300);
graphCoordinates.put(0, 0);
graphCoordinates.put(0, -1);

graphCoordinates.displayMap(graphCoordinates);
```

```
class Employee{

    public String name;

    public int age;

    public Employee(String name, int age) {
        this.name = name;
        this.age = age;
    }

    @Override
    public String toString() {
        return "Name = " + this.name + "Age = " + this.age;
    }
}
```

## Outputs

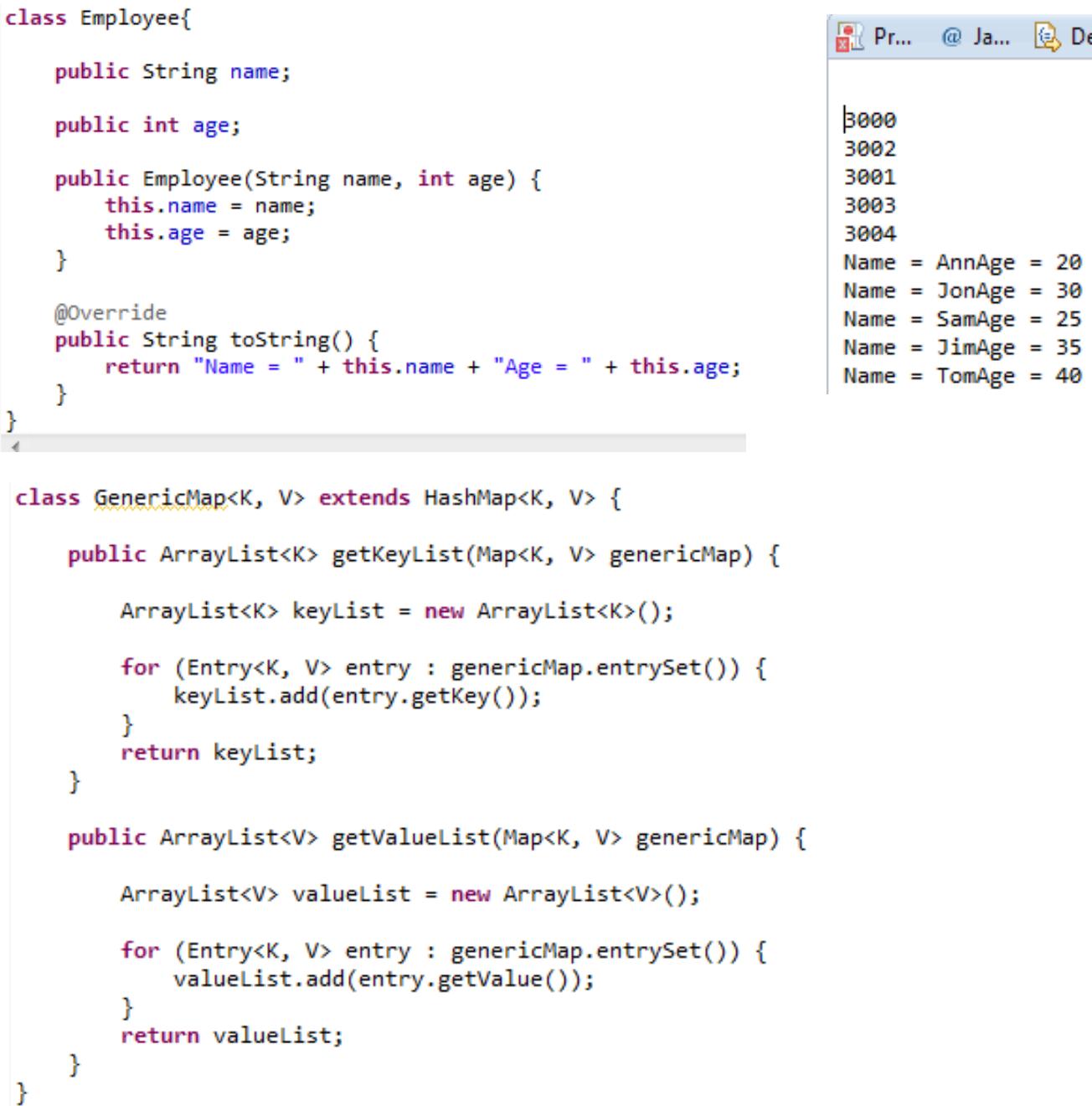
```
3000 Name = AnnAge = 20
3002 Name = JonAge = 30
3001 Name = SamAge = 25
3003 Name = JimAge = 35
3004 Name = TomAge = 40
0 -1
2 3
10 20
-20 300
```

# Generic Implementation

```
GenericMap<String, Employee> employeeMap = new GenericMap<String, Employee>();
employeeMap.put("3000", new Employee("Ann", 20));
employeeMap.put("3001", new Employee("Sam", 25));
employeeMap.put("3002", new Employee("Jon", 30));
employeeMap.put("3003", new Employee("Jim", 35));
employeeMap.put("3004", new Employee("Tom", 40));

GenericMap<Integer, Integer> graphCoordinates = new GenericMap<Integer, Integer>();
graphCoordinates.put(2, 3);
graphCoordinates.put(10, 20);
graphCoordinates.put(-20, 300);
graphCoordinates.put(0, 0);
graphCoordinates.put(0, -1);

for (String key : employeeMap.getKeyList(employeeMap)) {
    System.out.println(key);
}
for (Employee value : employeeMap.getValueList(employeeMap)) {
    System.out.println(value);
}
```



The screenshot shows an IDE interface with a code editor and a right-hand panel displaying output or results.

**Code Editor Content:**

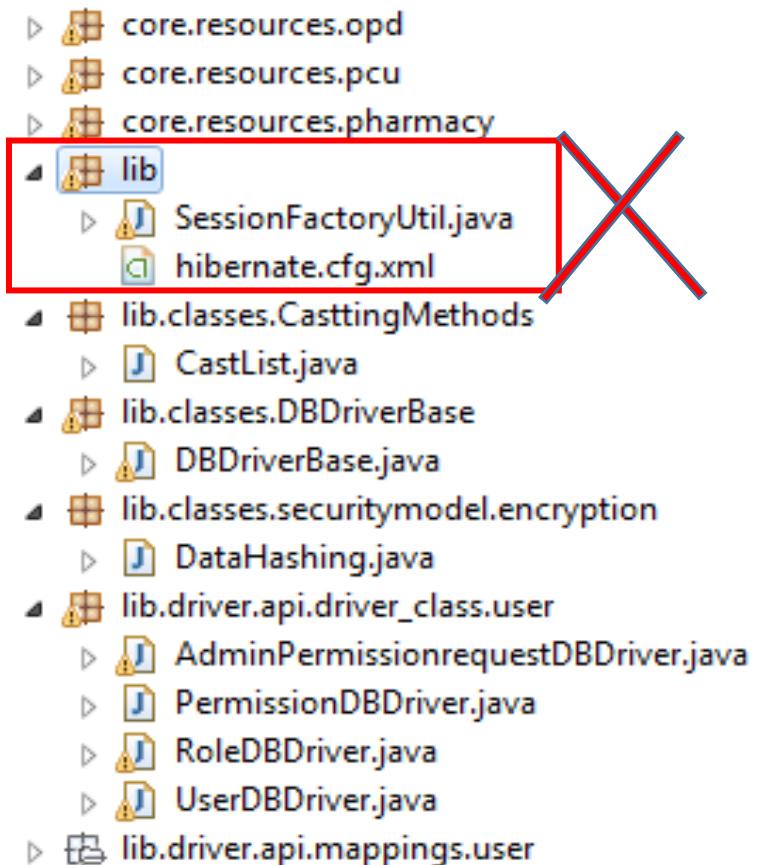
```
class Employee{  
    public String name;  
    public int age;  
    public Employee(String name, int age) {  
        this.name = name;  
        this.age = age;  
    }  
    @Override  
    public String toString() {  
        return "Name = " + this.name + "Age = " + this.age;  
    }  
}  
  
class GenericMap<K, V> extends HashMap<K, V> {  
    public ArrayList<K> getKeyList(Map<K, V> genericMap) {  
        ArrayList<K> keyList = new ArrayList<K>();  
        for (Entry<K, V> entry : genericMap.entrySet()) {  
            keyList.add(entry.getKey());  
        }  
        return keyList;  
    }  
    public ArrayList<V> getValueList(Map<K, V> genericMap) {  
        ArrayList<V> valueList = new ArrayList<V>();  
        for (Entry<K, V> entry : genericMap.entrySet()) {  
            valueList.add(entry.getValue());  
        }  
        return valueList;  
    }  
}
```

**Output Panel Content:**

Key	Value
3000	Name = AnnAge = 20
3002	Name = JonAge = 30
3001	Name = SamAge = 25
3003	Name = JimAge = 35
3004	Name = TomAge = 40

# Tips for development

- DataHashing secure class should be a utility. (Move to Util Package)
- Don't use "lib" as package name here.
- You should have **proper packaging and naming conventions**.
- All naming conventions should be documented and everybody should follow it.
- **Classes, methods, Comments and other implementation should follow common template.**
- **Template should be documented & shared in the repository.**
- Learn how to use code repository.
- Use merge tools to merge large source files.



# Implement Utilities

```
/*
 * This method generate Unique Lot no for
 * Insertion
 * @param poID
 * @return
 */
public static String generateLotNo(String poID, ArrayList<String> idList) {
    String lotNo = poID;
    ArrayList<Integer> subIdList = new ArrayList<Integer>();
    int postfix = 0;

    if (idList.isEmpty()) {
        postfix++;
        lotNo = lotNo + DASH_ZERO + postfix;
    }
    else {
        for (String lotID : idList) {
            if (lotID.split(DASH)[0].equals(poID)) {
                subIdList.add(Integer.parseInt(lotID.split(DASH)[1]));
            }
            else{
                subIdList.add(0);
            }
        }
        postfix = Collections.max(subIdList) + 1;
        lotNo = (postfix <= MAX_DIGIT) ? lotNo + DASH_ZERO + postfix
            : lotNo + DASH + postfix;
    }
    return lotNo;
}

    /**
     * Read Facility-configuration.xml file and get all available facilities
     * @return
     */
    public static ArrayList<String> getFacilities() {
        ArrayList<String> facilities = new ArrayList<String>();
        NodeList nodeList;
        try {
            // Once deploy to Tomcat server use below configuration for FACILITY
            // System.getProperty("catalina.base")+"/webapps/BiharGlassApp/WEB-INF
            Document xmlDocument = DocumentBuilderFactory.newInstance()
                .newDocumentBuilder().parse(new File(FACILITY_PATH));
            XPath xPath = XPathFactory.newInstance().newXPath();
            nodeList = (NodeList) xPath.compile(XPATH_EXPRESSION).evaluate(
                xmlDocument, XPathConstants.NODESET);

            for (int i = 0; i < nodeList.getLength(); i++) {
                facilities.add(nodeList.item(i).getFirstChild().getNodeValue());
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        return facilities;
    }
}
```

The End



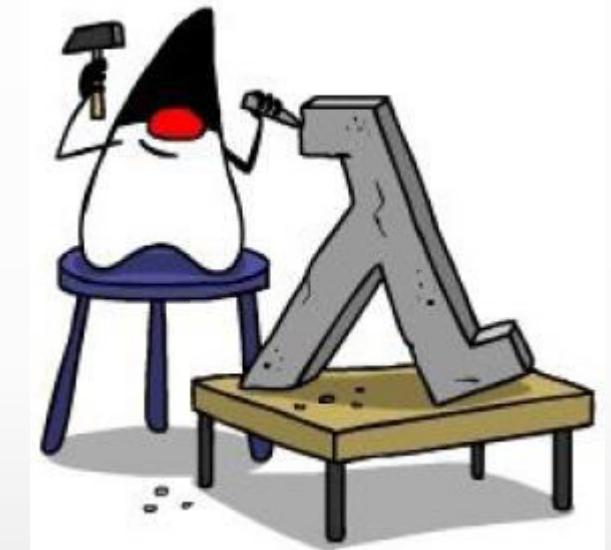
# Lambda Expressions

**Modern Topics in IT(MTIT)**  
**4<sup>th</sup> Year – Semester 1**  
**By Udara Samaratunge**

# Java 8 Lambda Expressions

---

- Introduction
- Anonymous classes
- Syntax of Java 8 lambda expressions
- Functional interfaces
- Default methods
- Method references
- Benefits of Lambda Expression



# Introduce Lambda Expressions

---

Mark Reinhold, Oracle's Chief Architect, describes Lambda expressions as the single largest upgrade to the programming model ever — larger even than generics.



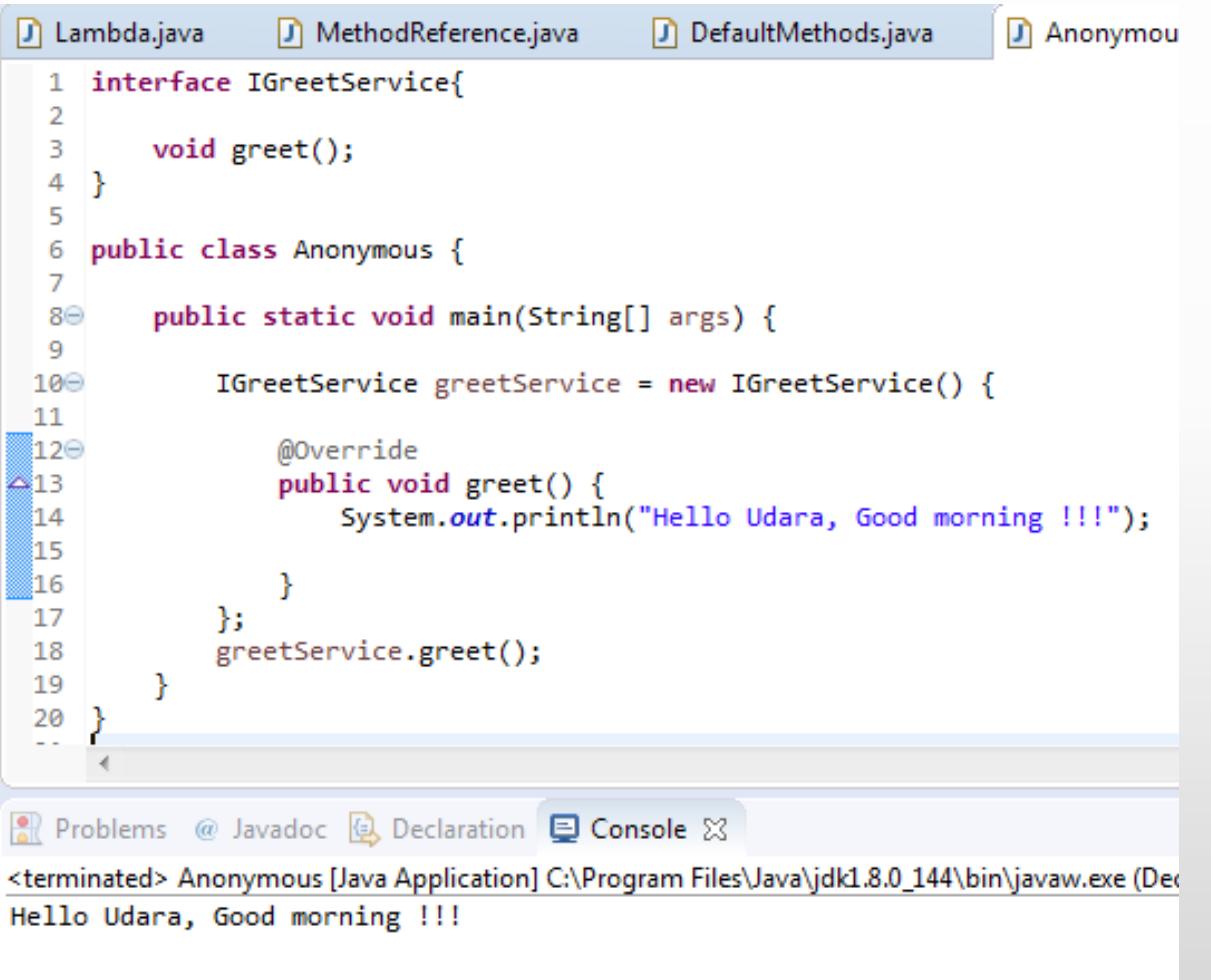
# Why Java need Lambda?

---

- One **issue with anonymous classes** is that if the implementation of your anonymous class is very simple, such as an interface that contains only one method, **then the syntax of anonymous classes may seem unwieldy and unclear.**
- In these cases, you're usually trying to **pass functionality as an argument to another method**, such as what action should be taken when someone clicks a button.
- Lambda expressions enable you to do this, to treat functionality as **method argument, or code as data.**

# Anonymous Classes

- Enable you to declare and instantiate a class at the same time.
- They are like local classes except that they do not have a name.
- Use them if you need to use a local class only once.



The screenshot shows a Java code editor with several tabs at the top: Lambda.java, MethodReference.java, DefaultMethods.java, and Anonymous.java. The Anonymous.java tab is active. The code defines an interface IGreetService with a greet() method, and a class Anonymous that implements it. The main() method creates an instance of IGreetService and prints "Hello Udara, Good morning !!!". The code is as follows:

```
1 interface IGreetService{
2
3     void greet();
4 }
5
6 public class Anonymous {
7
8     public static void main(String[] args) {
9
10        IGreetService greetService = new IGreetService() {
11
12            @Override
13            public void greet() {
14                System.out.println("Hello Udara, Good morning !!!");
15            }
16        };
17        greetService.greet();
18    }
19 }
20 }
```

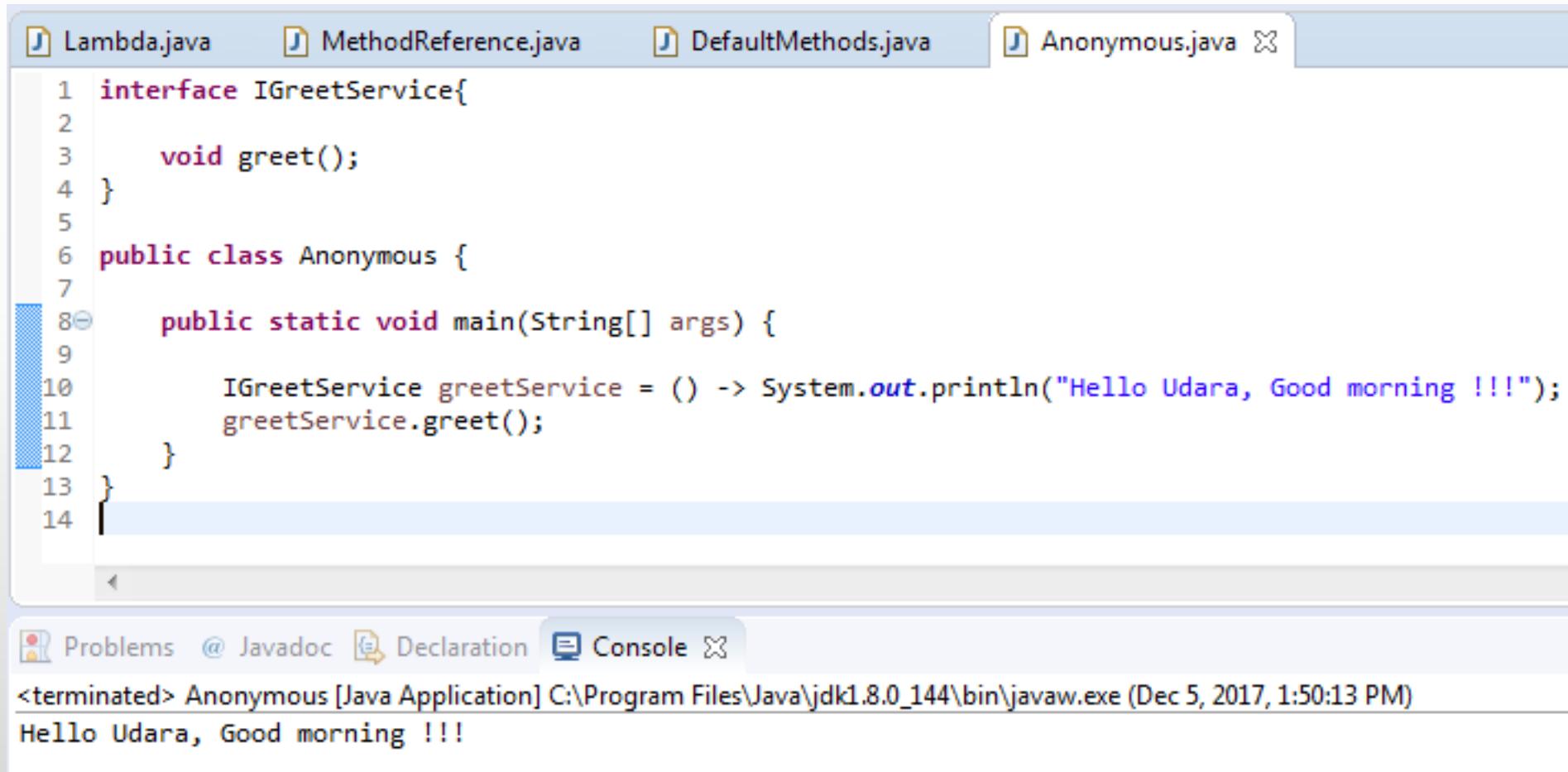
At the bottom, the IDE's status bar shows the output: <terminated> Anonymous [Java Application] C:\Program Files\Java\jdk1.8.0\_144\bin\javaw.exe (Dec 14, 2017, 11:25:22 PM)  
Hello Udara, Good morning !!!

# Anonymous Class

---

1. The new operator
2. The name of an interface to implement or a class to extend. In this example the anonymous class is implementing the interface HelloWorld.
3. Parentheses that contain the arguments to a constructor, just like a normal class instance creation expression. Note: When you implement an interface, there is no constructor, so you use an empty pair of parentheses, as in this example.
4. A body, which is a class declaration body. More specifically, in the body, method declarations are allowed but statements are not.

# Anonymous Class



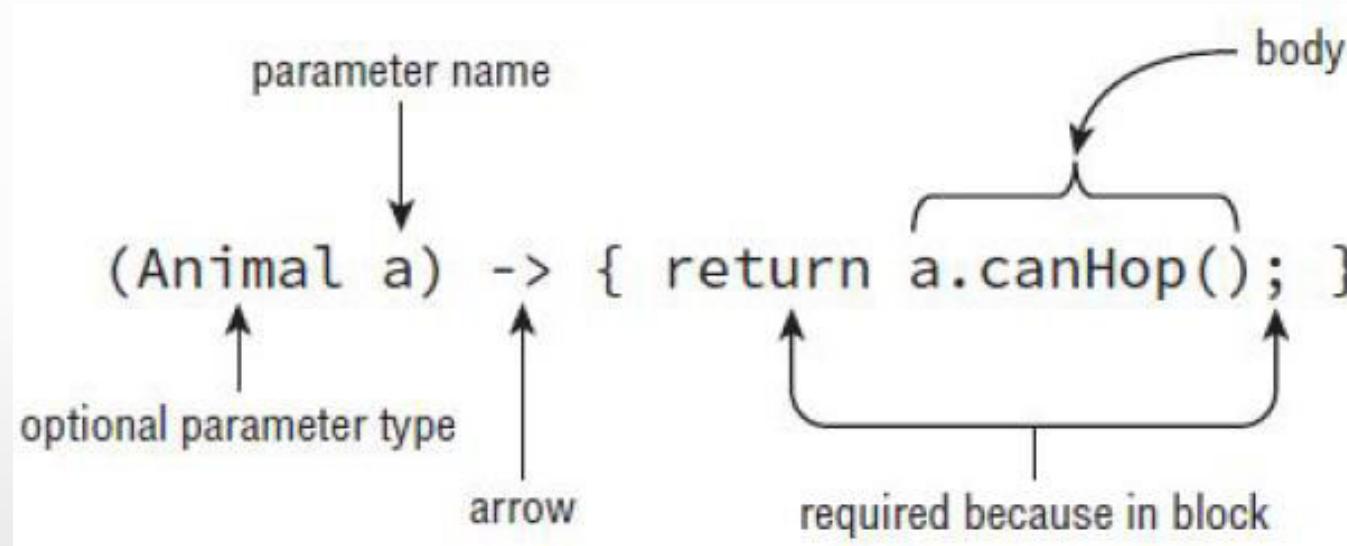
The screenshot shows an IDE interface with the following details:

- File Tab:** The tab bar includes "Lambda.java", "MethodReference.java", "DefaultMethods.java", and "Anonymous.java" (which is currently active).
- Code Editor:** The main window displays Java code:

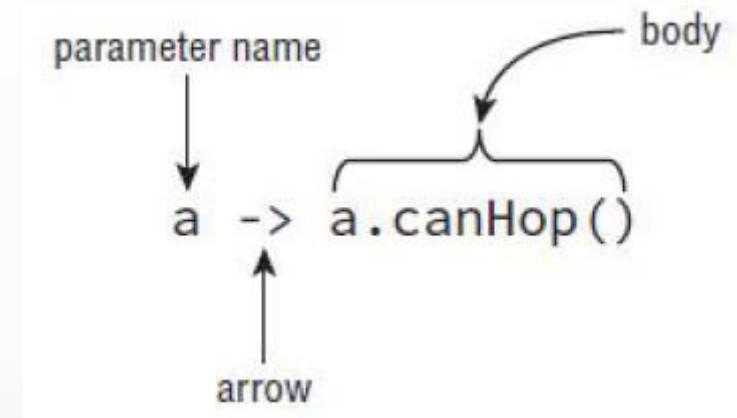
```
1 interface IGreetService{  
2     void greet();  
3 }  
4  
5 public class Anonymous {  
6     public static void main(String[] args) {  
7         IGreetService greetService = () -> System.out.println("Hello Udara, Good morning !!!");  
8         greetService.greet();  
9     }  
10    }  
11 }
```
- Console Tab:** The "Console" tab is selected, showing the output of the program:

```
<terminated> Anonymous [Java Application] C:\Program Files\Java\jdk1.8.0_144\bin\javaw.exe (Dec 5, 2017, 1:50:13 PM)  
Hello Udara, Good morning !!!
```

# Simple Lambda Expression Syntax - Complete



# Simple Lambda Expression Syntax - Simplified



## Simplifying Rules

1. The **parenthesis can only be omitted** if there is a **single parameter** and its type is not explicitly stated
2. Can **omit braces** when there is only **one statement** in the body
3. Does **not require to type return** or semicolon when braces are omitted

# Lambda Syntaxes

---

```
() -> {}           // No parameters; void result

() -> 42           // No parameters, expression body
() -> null          // No parameters, expression body
() -> { return 42; }    // No parameters, block body with return
() -> { System.gc(); }   // No parameters, void block body
```

# Lambda Syntax

```
(int x) -> x+1          // Single declared-type argument
(int x) -> { return x+1; } // same as above
(x) -> x+1              // Single inferred-type argument, same as below
x -> x+1                // Parenthesis optional for single inferred-type case

(String s) -> s.length()  // Single declared-type argument
(Thread t) -> { t.start(); } // Single declared-type argument
s -> s.length()           // Single inferred-type argument
t -> { t.start(); }        // Single inferred-type argument

(int x, int y) -> x+y    // Multiple declared-type parameters
(x,y) -> x+y             // Multiple inferred-type parameters
(x, final y) -> x+y     // Illegal: can't modify inferred-type parameters
(x, int y) -> x+y       // Illegal: can't mix inferred and declared types
```

# Examples of Lambda Syntax

---

```
(int x, int y) -> { return x + y; }
    //Returns the sum of two integers

(x, y) -> x + y
/*
 * Returns the sum of two numbers types of parameters and return type
 * are inferred body is a single expression; braces omitted
 */
n -> n % 2 == 0
    //returns true if n is even (type is inferred)

() -> 42
/*
 * Constant function, no parameters returns the answer to the ultimate
 * question of life, the universe, and everything
 */
```

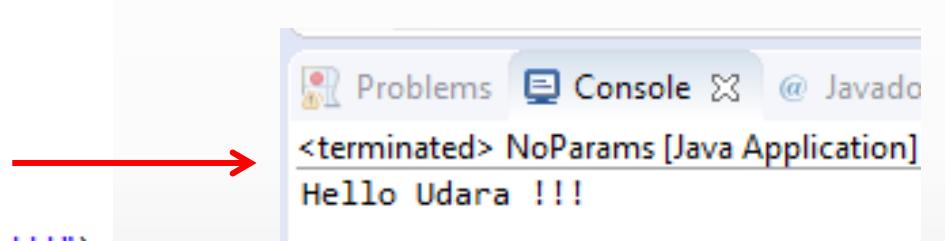
# More Examples of Java 8 Lambdas

---

- A Java 8 lambda is basically a method in Java without a declaration usually written as (parameters) -> { body }. Examples,
  1. `(int x, int y) -> { return x + y; }`
  2. `x -> x * x`
  3. `() -> x`
- A lambda can **have zero or more parameters** separated by commas and their type can be explicitly declared or inferred from the context.
- Parenthesis are not needed around a single parameter.
- `()` is used to denote **zero parameters**.
- The body can contain **zero or more statements**.
- Braces are not needed around a single-statement body.

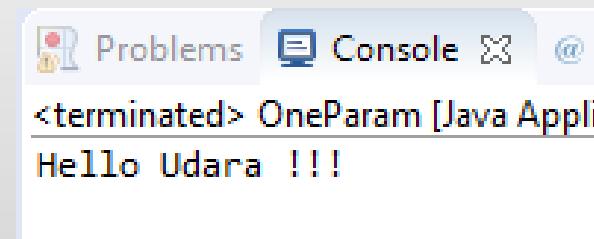
# Java Lambda Expression with no Parameters

```
interface ICommunicable{  
    void sayHello();  
}  
  
public class NoParams {  
    public static void main(String[] args) {  
        //Java Lambda Expression with no Parameters  
        ICommunicable iCommunicable = () -> System.out.println("Hello Udara !!!");  
        iCommunicable.sayHello();  
    }  
}
```



# Java Lambda Expression with one Parameter

```
interface ICommunicable2{  
    void sayHello(String name);  
}  
  
public class OneParam {  
    public static void main(String[] args) {  
        //Java Lambda Expression with one Parameters  
        ICommunicable2 iCommunicable2 = (name) -> System.out.println("Hello " + name + " !!!");  
        iCommunicable2.sayHello("Udara");  
    }  
}
```



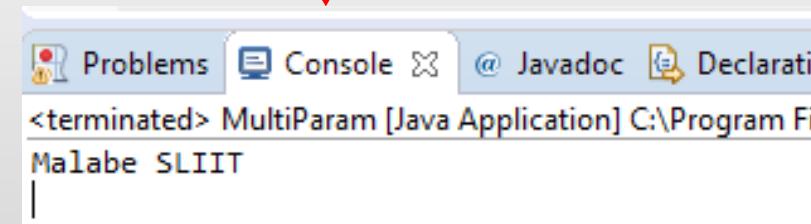
The terminal window shows the output of the Java application. It has tabs for 'Problems', 'Console' (which is selected), and '@'. The text area displays: '<terminated> OneParam [Java Appli' followed by 'Hello Udara !!!'.

# Java Lambda Expression with multiple Parameters

```
package lambda;

interface IConcatenate{
    void concat(String branch, String institute);
}

public class MultiParam {
    public static void main(String[] args) {
        //Java Lambda Expression with no Parameters
        IConcatenate iConcatenate = (branch, institute) -> System.out.println(branch + institute);
        iConcatenate.concat("Malabe ", "SLIIT");
    }
}
```

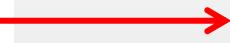


The screenshot shows the Eclipse IDE's Console tab. The tab bar includes 'Problems', 'Console' (which is selected), 'Javadoc', and 'Declarations'. The console output window displays the message: '<terminated> MultiParam [Java Application] C:\Program File Malabe SLIIT |'

# Lambda Syntax

```
public class Example2 {  
  
    interface ICalcculater{  
  
        int add(int no1, int no2);  
    }  
  
    public static void main(String[] args) {  
  
        ICalcculater iCalcculater = (x, y) -> (x + y);  
  
        System.out.println(iCalcculater.add(100, 200));  
    }  
}
```

Response



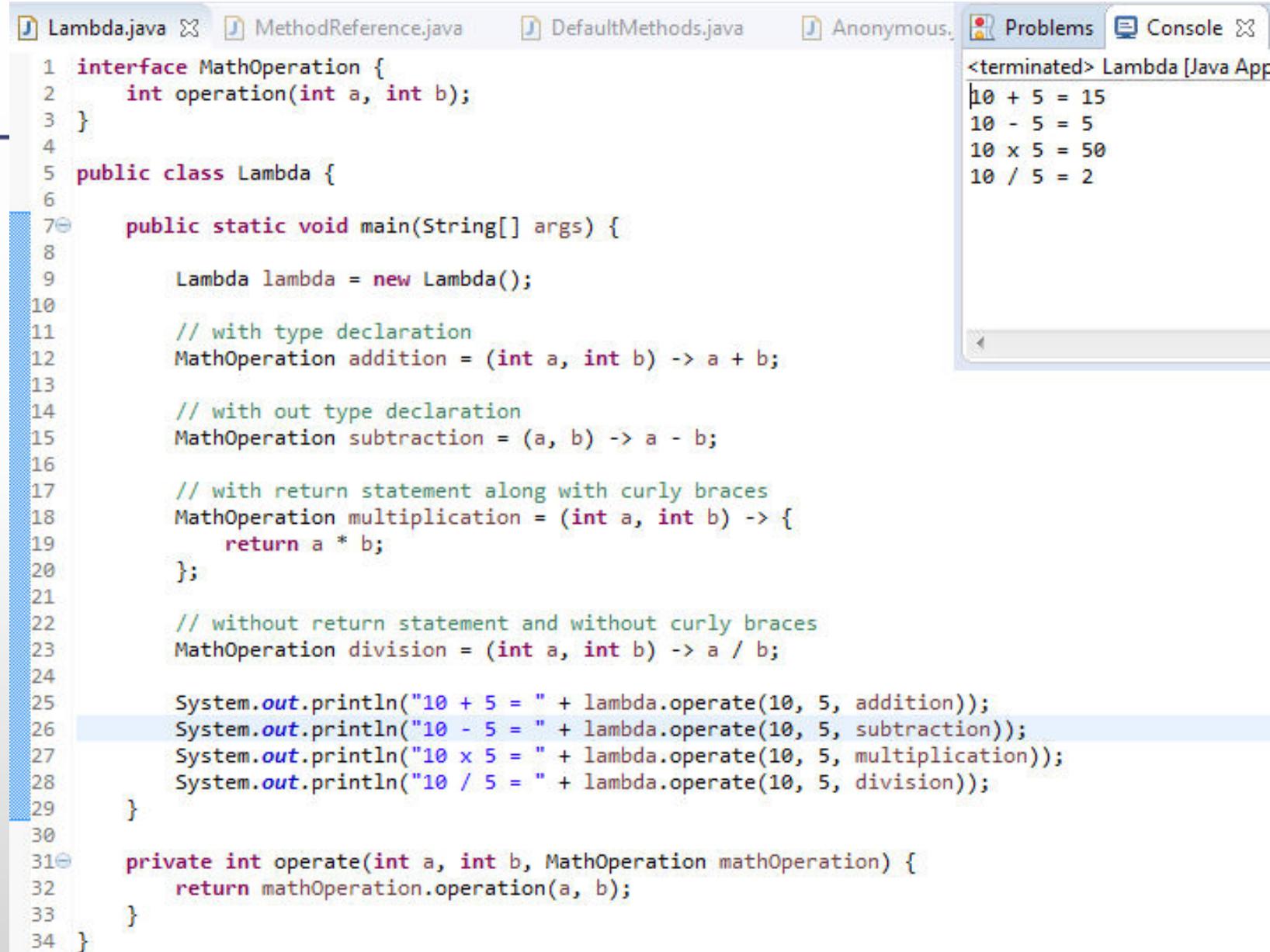
```
Problems Console @ Javadoc Declaration  
<terminated> FunctionalInterface [Java Application] C:\Program  
300
```

# Lambda Syntax

---

- **Optional type declaration** – No need to declare the type of a parameter. The compiler can inference the same from the value of the parameter.
- **Optional parenthesis around parameter** – No need to declare a single parameter in parenthesis. For multiple parameters, parentheses are required.
- **Optional curly braces** – No need to use curly braces in expression body if the body contains a single statement.
- **Optional return keyword** – The compiler automatically returns the value if the body has a single expression to return the value. Curly braces are required to indicate that expression returns a value.

# Lambda Syntax



The screenshot shows an IDE interface with the following details:

- Project Structure:** Shows tabs for "Lambda.java", "MethodReference.java", "DefaultMethods.java", and "Anonymous.java".
- Code Editor:** Displays the Java code for "Lambda.java". The code defines an interface "MathOperation" with a method "operation(int a, int b)". It then creates a class "Lambda" with a main method. Inside the main method, it creates a lambda expression "addition = (int a, int b) -> a + b;". It also shows examples of lambda expressions without type declarations, with return statements and curly braces, and without return statements and curly braces. Finally, it prints the results of these operations.
- Output Console:** Shows the terminal output:

```
<terminated> Lambda [Java Application]
10 + 5 = 15
10 - 5 = 5
10 x 5 = 50
10 / 5 = 2
```

# Lambda Variable Binding

Without using Lambda Expressions

```
package variable;

public class Binding {

    public static void main(String[] args) {
        Binding binding = new Binding();
        binding.method();
    }

    void method() {
        final int count = 16;
        Runnable r = new Runnable() {
            public void run() {
                System.out.println("count: " + count);
            }
        };
        Thread t = new Thread(r);
        t.start();
        count++; // error: count| is final
    }
}
```

Final variable can not be reassigned  
cont++ implies count = count + 1

→ Compile Error

# Lambda Variable Binding

Using Lambda Expressions

```
package variable;

public class Binding {

    public static void main(String[] args) {
        Binding binding = new Binding();
        binding.method();
    }

    void method() {
        final int count = 16;
        Runnable r = () -> {System.out.println("count: " + count);}
        Thread t = new Thread(r);
        t.start();
        count++; // error: count is final
    }
}
```

Final variable can not be reassigned  
cont++ implies count = count + 1

→ **Still Compile Error**

What happens if you remove final keyword?

# Lambda Variable Binding

## Using Lambda Expressions

```
package variable;

public class Binding {

    public static void main(String[] args) {
        Binding binding = new Binding();
        binding.method();
    }

    void method() {
        int count = 16;
        Runnable r = () -> {System.out.println("count: " + count);};
        Thread t = new Thread(r);
        t.start();
        count++; // error: count is final
    }
}
```

**Still Compile Error**



No **final keyword** for the variable declarations



- The difference is that the count variable **need not be declared as final** in the **enclosing context**.
- The **compiler automatically treats it as final** as soon as it is used **inside a lambda expression**
- In other words, variables from the enclosing context that are used inside a lambda expression are **implicitly final (or effectively final)**.

---

# Functional Interfaces

# Functional Interfaces

---

- Is an interface
- **Functional Interfaces** can only be used with **Lambda Expressions**
- Must have **only one abstract method**  
In JDK 7 this would mean only one method (like ActionListener)
- JDK 8 introduced **default methods**  
Adding multiple inheritance of types to Java These are, by definition, not abstract
- **Functional Interface** can have only **one abstract methods** but **0 or more default methods** or **static methods**
- **@FunctionalInterface** to have the compiler check

# Is This A Functional Interface?



```
Anonymous.java  Annotate.java  Example1.java  FunctionalInterface.java
1 interface IGreetService{
2
3     void greet();
4     void sayGoodBye(String bve);
5 }
6
7 public class Anonymous {
8
9     public static void main(String[] args) {
10
11         IGreetService greetService = () -> System.out.println("Hello Udara, Good morning !!!");
12         greetService.greet();
13     }
14 }
15
```

**Functional Interface  
should have only one  
abstract method**

# Is This A Functional Interface?

```
interface ICalculator{  
  
    int add(int no1, int no2);  
  
    default int min(int no1, int no2){  
        return (no1 - no2);  
    }  
  
    default int mul(int no1, int no2){  
        return (no1 * no2);  
    }  
  
    default double div(int no1, int no2){  
        return (no1 / no2);  
    }  
}
```



But you can include  
multiple default  
methods

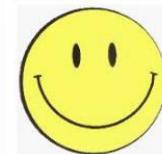


# Is This A Functional Interface?

```
interface IGreetService{  
    void greet();  
  
    static void sayGoodBye(){  
        System.out.println("Hi Good Bye");  
    }  
}
```



You can include  
multiple default  
static methods



```
11 public class Anonymous {  
12  
13     public static void main(String[] args) {  
14  
15         IGreetService greetService = () -> System.out.println("Hello Udara, Good morning !!!");  
16         greetService.greet();  
17         IGreetService.sayGoodBye();  
18     }  
19 }
```

Problems Console @ Javadoc Declaration

<terminated> Anonymous [Java Application] C:\Program Files\Java\jdk1.8.0\_144\bin\javaw.exe (Jan 25, 2018, 10:09:53 AM)

Hello Udara, Good morning !!!  
Hi Good Bye

# Functional Interface

```
interface IGreetService{  
    void greet();  
}
```



```
@FunctionalInterface  
interface IGreetService{  
    void greet();  
}
```



```
@FunctionalInterface  
interface IGreetService{  
    void sayGoodMorning();  
  
    default void sayGoodAfterNoon(){  
        System.out.println("Hi Udara, Good After Noon !!!");  
    }  
  
    static void sayGoodEvening(){  
        System.out.println("Hi Udara, Good Evening !!!");  
    }  
}
```



@FunctionalInterface annotation is useful for compilation time checking of your code

But you can use lambdas without this annotation as well as you can override methods without @Override annotation

# Functional Interface

```
@FunctionalInterface
interface IGreetService{

    void sayGoodMorning();

    default void sayGoodAfterNoon(){
        System.out.println("Hi Udara, Good After Noon !!!");
    }

    static void sayGoodEvening(){
        System.out.println("Hi Udara, Good Evening !!!");
    }
}
```



```
16 public class Anonymous {
17
18    public static void main(String[] args) {
19
20        IGreetService greetService = () -> System.out.println("Hi Udara, Good morning !!!");
21        greetService.sayGoodMorning();
22        greetService.sayGoodAfterNoon();
23        IGreetService.sayGoodEvening(); → Access static method
24    }
25 }
```

```
Problems Console X @ Javadoc Declar
<terminated> Anonymous [Java Application] C:\Program
Hi Udara, Good morning !!!
Hi Udara, Good After Noon !!!
Hi Udara, Good Evening !!!
```

# Is This A Functional Interface?

```
@FunctionalInterface
public interface Predicate<T> {
    default Predicate<T> and(Predicate<? super T> p) {...};
    default Predicate<T> negate() {...};
    default Predicate<T> or(Predicate<? super T> p) {...};
    static <T> Predicate<T> isEqual(Object target) {...};
    boolean test(T t);
}
```

Yes. There is still  
only one abstract  
method

```
@FunctionalInterface
public interface Runnable {
    public abstract void run();
}
```

Yes. There is only  
one abstract  
method

# Is This A Functional Interface?

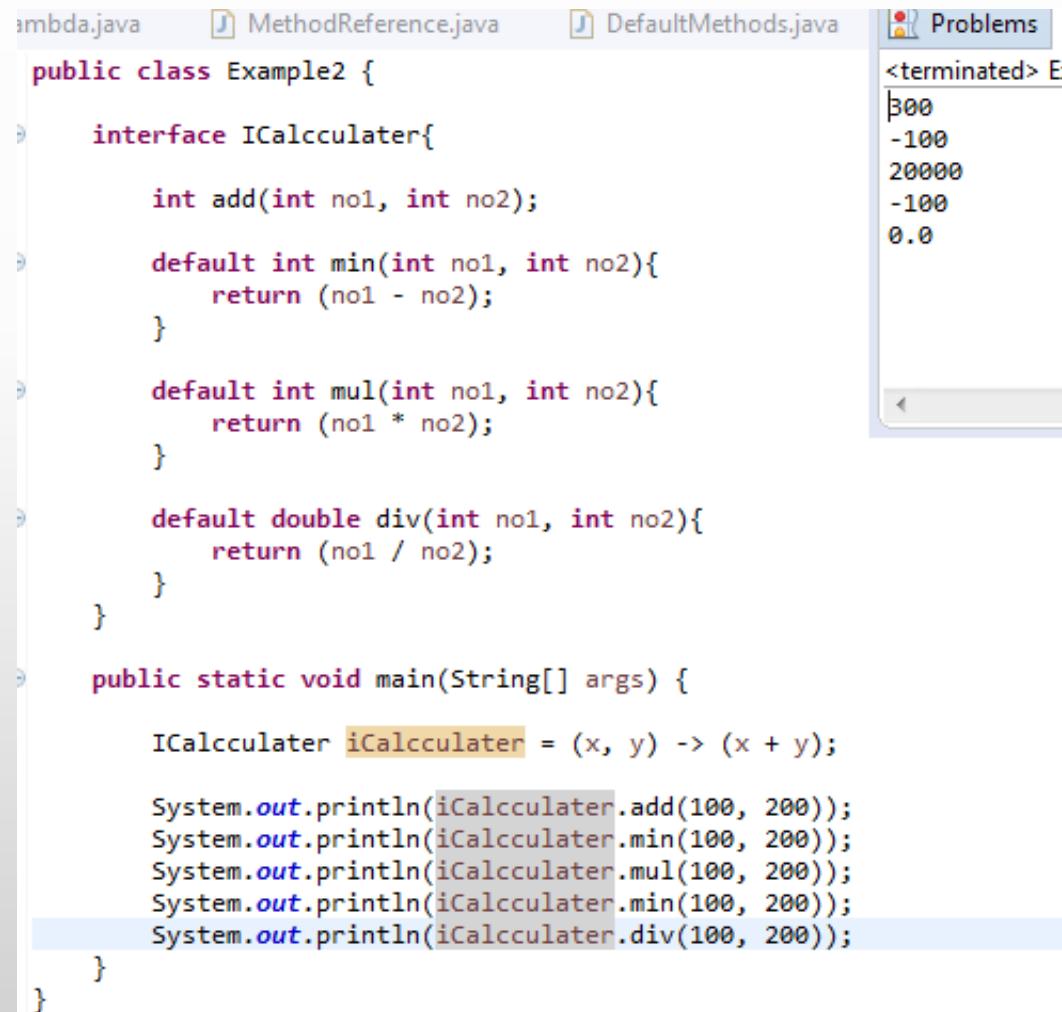
```
@FunctionalInterface
public interface Comparator {
    // Static and default methods elided
    int compare(T o1, T o2);
    boolean equals(Object obj);
}
```

The equals(`Object`) method is implicit from the Object class

Therefore only one abstract method

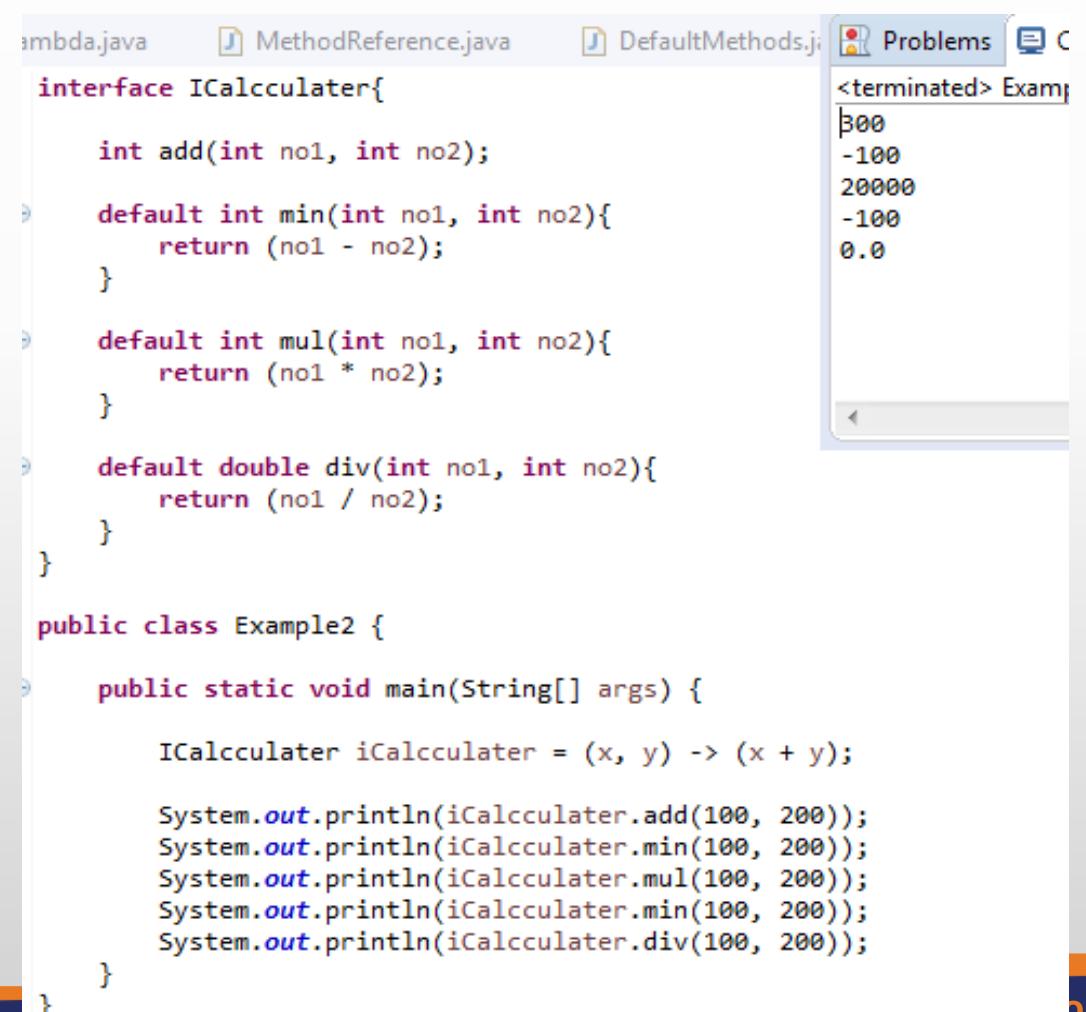
# Lambda Expressions with Functional Interface

## Interface within the class



```
lambda.java  MethodReference.java  DefaultMethods.java  Problems
public class Example2 {
    interface ICalcculater{
        int add(int no1, int no2);
        default int min(int no1, int no2){
            return (no1 - no2);
        }
        default int mul(int no1, int no2){
            return (no1 * no2);
        }
        default double div(int no1, int no2){
            return (no1 / no2);
        }
    }
    public static void main(String[] args) {
        ICalcculater iCalcculater = (x, y) -> (x + y);
        System.out.println(iCalcculater.add(100, 200));
        System.out.println(iCalcculater.min(100, 200));
        System.out.println(iCalcculater.mul(100, 200));
        System.out.println(iCalcculater.min(100, 200));
        System.out.println(iCalcculater.div(100, 200));
    }
}
```

## Interface outside the class



```
lambda.java  MethodReference.java  DefaultMethods.java  Problems
interface ICalcculater{
    int add(int no1, int no2);
    default int min(int no1, int no2){
        return (no1 - no2);
    }
    default int mul(int no1, int no2){
        return (no1 * no2);
    }
    default double div(int no1, int no2){
        return (no1 / no2);
    }
}
public class Example2 {
    public static void main(String[] args) {
        ICalcculater iCalcculater = (x, y) -> (x + y);
        System.out.println(iCalcculater.add(100, 200));
        System.out.println(iCalcculater.min(100, 200));
        System.out.println(iCalcculater.mul(100, 200));
        System.out.println(iCalcculater.min(100, 200));
        System.out.println(iCalcculater.div(100, 200));
    }
}
```

---

# Method References

# Method References

---

## What is the Method References?

**Method reference is the shorthand notation of a lambda expression to call a method**

If your lambda expression is like this:

```
str -> System.out.println(str)
```

then you can replace it with a method reference like this:

```
System.out::println
```

The **:: operator** is used in method reference to separate the class or object from the method name

# What is the difference ?

Anonymous  
Inner  
classes

```
File[] files = myDir.listFiles(  
    new FileFilter() {  
        public boolean accept(File f) { return f.isFile(); }  
    }  
) ;
```

Lambda  
Expressions

```
File[] files = myDir.listFiles(  
    (File f) -> { return f.isFile(); }  
) ;
```

Method  
References

```
File[] files = myDir.listFiles( File::isFile );
```

# Method References

---

## Four Types of Method References

- 1) Method reference to an **instance method** of an object – `object::instanceMethod`
- 2) Method reference to a **static method** of a class – `Class::staticMethod`
- 3) Method reference to an **instance method** of an arbitrary object of a particular type – `Class::instanceMethod`
- 4) Method reference to a **constructor** – `Class::new`

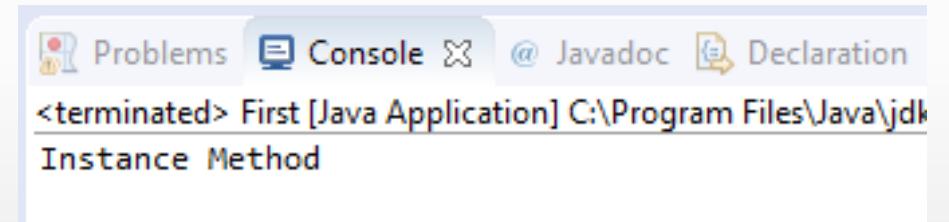
# 1. Method Reference to an instance method of an object

```
package method.reference;

interface IReference {
    void display();
}

public class First {
    public void myMethod() {
        System.out.println("Instance Method");
    }

    public static void main(String[] args) {
        First first = new First();
        IReference iReference = first::myMethod;
        iReference.display();
    }
}
```



Problems Console @ Javadoc Declaration  
<terminated> First [Java Application] C:\Program Files\Java\jdk...  
Instance Method

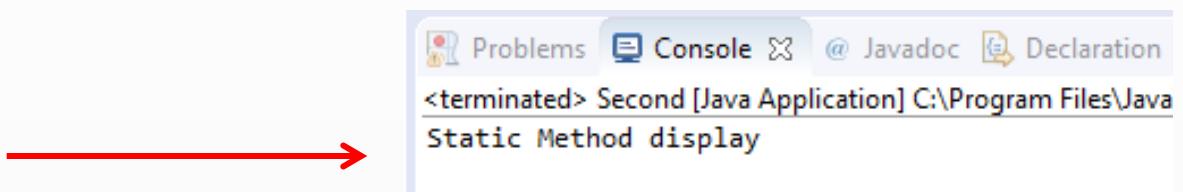
## 2. Method Reference to an instance method of an object

```
interface IReference {
    void display();
}

public class Second {

    public static void staticMethod() {
        System.out.println("Static Method display");
    }

    public static void main(String[] args) {
        IReference iReference = Second::staticMethod;
        iReference.display();
    }
}
```



Static method reference

### 3. Method reference to an instance method of an arbitrary object of a particular type

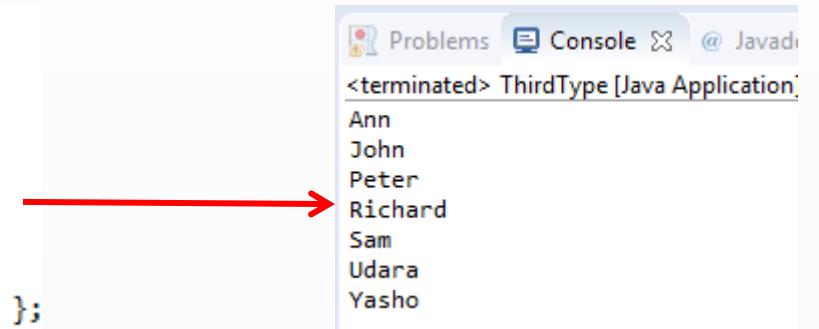
```
package method.reference;

import java.util.Arrays;

public class ThirdType {
    public static void main(String[] args) {
        String[] stringArray = { "Sam", "Richard", "Ann", "Udara", "John", "Peter", "Yasho" };

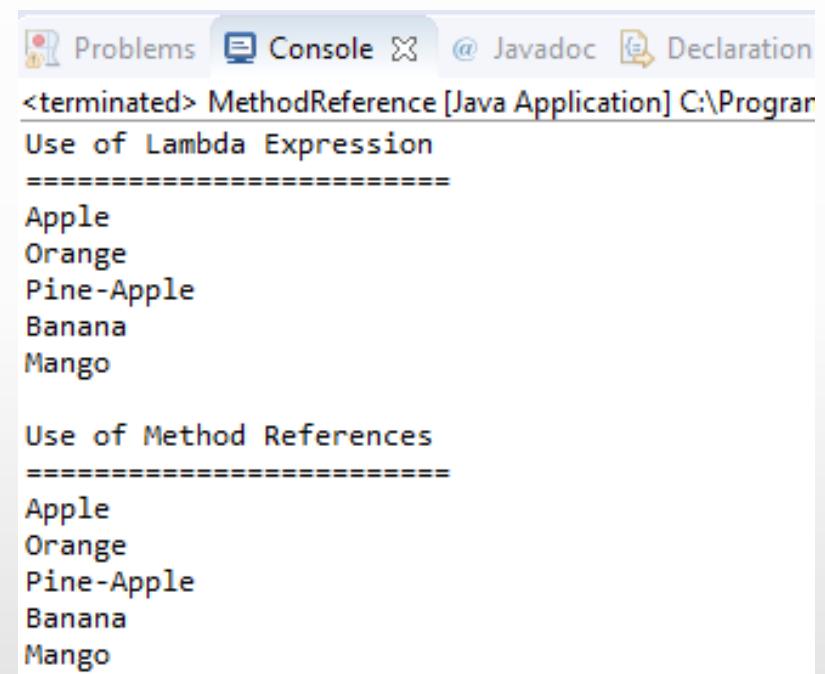
        /*
         * Method reference to an instance method of an arbitrary object of a
         * particular type
         */
        Arrays.sort(stringArray, String::compareToIgnoreCase);

        for (String str : stringArray) {
            System.out.println(str);
        }
    }
}
```



### 3. Method reference with and without examples

```
public class MethodReference {  
  
    public static void main(String[] args) {  
  
        List<String> fruities = new ArrayList<String>();  
        fruities.add("Apple");  
        fruities.add("Orange");  
        fruities.add("Pine-Apple");  
        fruities.add("Banana");  
        fruities.add("Mango");  
        System.out.println("Use of Lambda Expression");  
        System.out.println("=====");  
  
        fruities.forEach(  
            //Use of Lambda Expression  
            (fruit) -> System.out.println(fruit)  
        );  
  
        System.out.println("\nUse of Method References");  
        System.out.println("=====");  
        //Use of Method References  
        fruities.forEach(System.out::println);  
    }  
}
```



Problems Console @ Javadoc Declaration

<terminated> MethodReference [Java Application] C:\Program

Use of Lambda Expression

=====

Apple  
Orange  
Pine-Apple  
Banana  
Mango

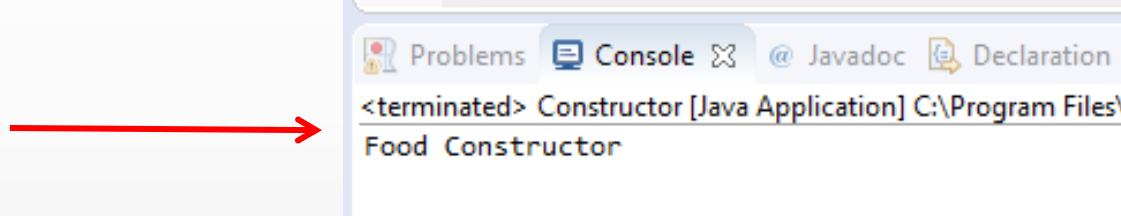
Use of Method References

=====

Apple  
Orange  
Pine-Apple  
Banana  
Mango

## 4. Method Reference to a Constructor

```
@FunctionalInterface  
interface IFoodService{  
    void display();  
}  
  
class Food{  
    public Food(){  
        System.out.print("Food Constructor");  
    }  
}  
  
public class Constructor {  
  
    public static void main(String[] args) {  
        //Method reference to a constructor  
        IFoodService iFoodService = Food::new;  
        iFoodService.display();  
    }  
}
```



→ Use new key word with :: operator to invoke constructor of the class

# Summary of Method References

Method Reference Type	Syntax	Example
static	ClassName::StaticMethodName	String::valueOf
constructor	ClassName::new	ArrayList::new
specific object instance	objectReference::MethodName	x::toString
arbitrary object of a given type	ClassName::InstanceMethodName	Object::toString

# Summary of Method References & Lambda Expressions

---

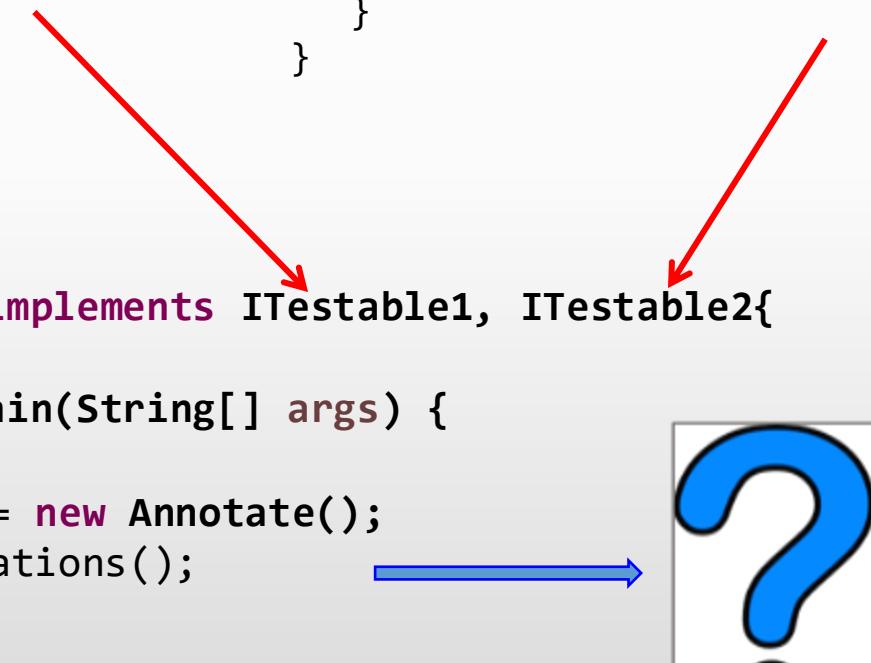
Type	Syntax	Method Reference	Lambda expression
Reference to a static method	<i>Class::staticMethod</i>	<i>String::valueOf</i>	<i>s -&gt; String.valueOf(s)</i>
Reference to an instance method of a particular object	<i>instance::instanceMethod</i>	<i>s::toString</i>	<i>O -&gt; "string".toString()</i>
Reference to an instance method of an arbitrary object of a particular type	<i>Class:instanceMethod</i>	<i>String::toString</i>	<i>s -&gt; s.toString()</i>
Reference to a constructor	<i>Class::new</i>	<i>String::new</i>	<i>O -&gt; new String()</i>

# Multiple Inheritance of Ambiguities

```
interface ITestable1{
    public default void readAnnotations(){
        System.out.println("Read Annotations 01");
    }
}

interface ITestable2{
    public default void readAnnotations(){
        System.out.println("Read Annotations 02");
    }
}

public class Annotate implements ITestable1, ITestable2{
    public static void main(String[] args) {
        Annotate diff = new Annotate();
        diff.readAnnotations();
    }
}
```



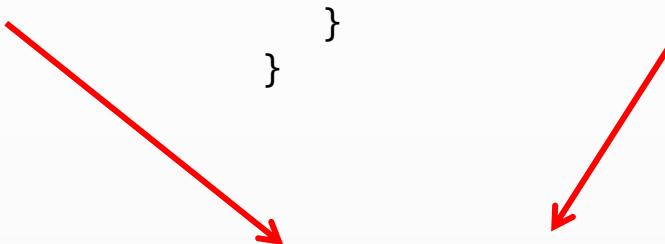
# Solution for Multiple Inheritance

```
interface ITestable1{
    public default void readAnnotations(){
        System.out.println("Read Annotations 01");
    }
}

interface ITestable2{
    public default void readAnnotations(){
        System.out.println("Read Annotations 02");
    }
}

public class Annotate implements ITestable1, ITestable2{
    public static void main(String[] args) {
        Annotate diff = new Annotate();
        diff.readAnnotations();
    }

    @Override
    public void readAnnotations() {
        ITestable1.super.readAnnotations();
        ITestable2.super.readAnnotations();
    }
}
```

A diagram illustrating multiple inheritance. Two red arrows point from the class definition 'Annotate' to the interface definitions 'ITestable1' and 'ITestable2' respectively.

# Apply Lambda Expressions – List iteration

```
package lambda;

import java.util.Arrays;
import java.util.List;

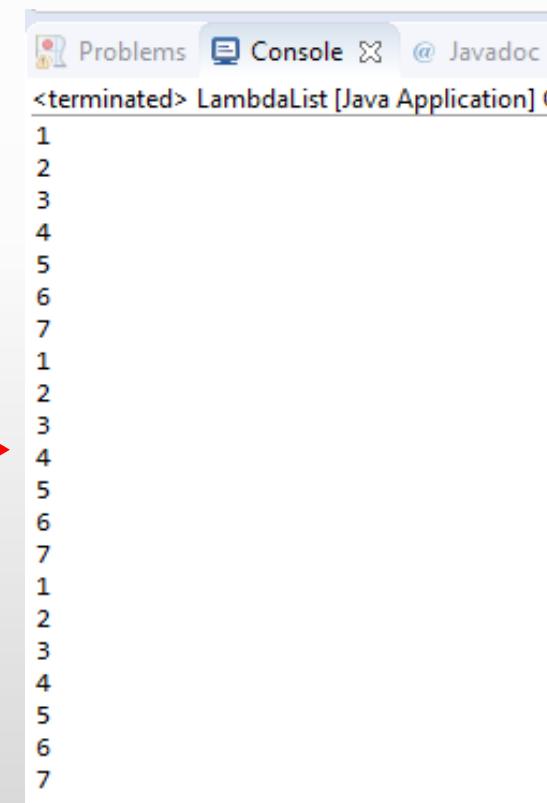
public class LambdaList {

    public static void main(String[] args) {

        // Old way:
        List<Integer> list = Arrays.asList(1, 2, 3, 4, 5, 6, 7);
        for (Integer n : list) {
            System.out.println(n);
        }

        // New way:
        List<Integer> list2 = Arrays.asList(1, 2, 3, 4, 5, 6, 7);
        list2.forEach(n -> System.out.println(n));

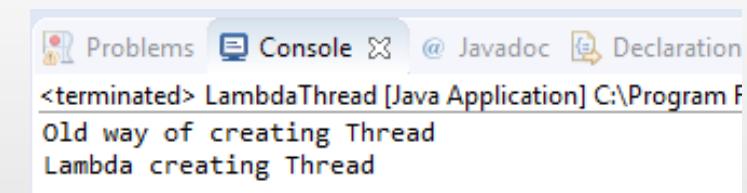
        // or we can use :: double colon operator in Java 8
        list.forEach(System.out::println);
    }
}
```



```
1
2
3
4
5
6
7
1
2
3
4
5
6
7
1
2
3
4
5
6
7
```

# Apply Lambda Expressions for Threads

```
public class LambdaThread {  
  
    public static void main(String[] args) {  
  
        //Old way of creating Threads  
        new Thread(  
            new Runnable() {  
  
                @Override  
                public void run() {  
                    System.out.println("Old way of creating Thread");  
  
                }  
            }).start();  
  
        //New way using Lambda Expressions  
        new Thread( () -> System.out.println("Lambda creating Thread")).start();  
  
    }  
}
```



Problems Console @ Javadoc Declaration  
<terminated> LambdaThread [Java Application] C:\Program F  
Old way of creating Thread  
Lambda creating Thread

# Benefits of Lambdas in Java 8

---

- Enabling functional programming
- Writing leaner more compact code
- Facilitating parallel programming
- Developing more generic, flexible and reusable APIs
- Being able to pass behaviors as well as data to functions

# References

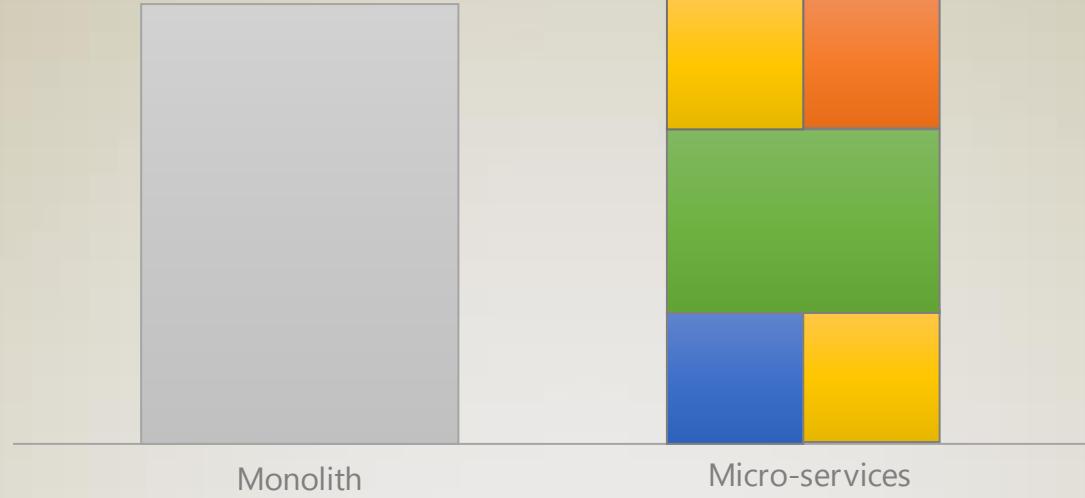
---

- The Java Tutorials, <http://docs.oracle.com/javase/tutorial/java/index.html>
- Lambda Expressions,  
<http://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html>
- Adib Saikali, Java 8 Lambda Expressions and Streams,  
[www.youtube.com/watch?v=8pDm\\_kH4YKY](https://www.youtube.com/watch?v=8pDm_kH4YKY)
- Brian Goetz, Lambdas in Java: A peek under the hood.  
<https://www.youtube.com/watch?v=MLksirK9nnE>

---

# The End





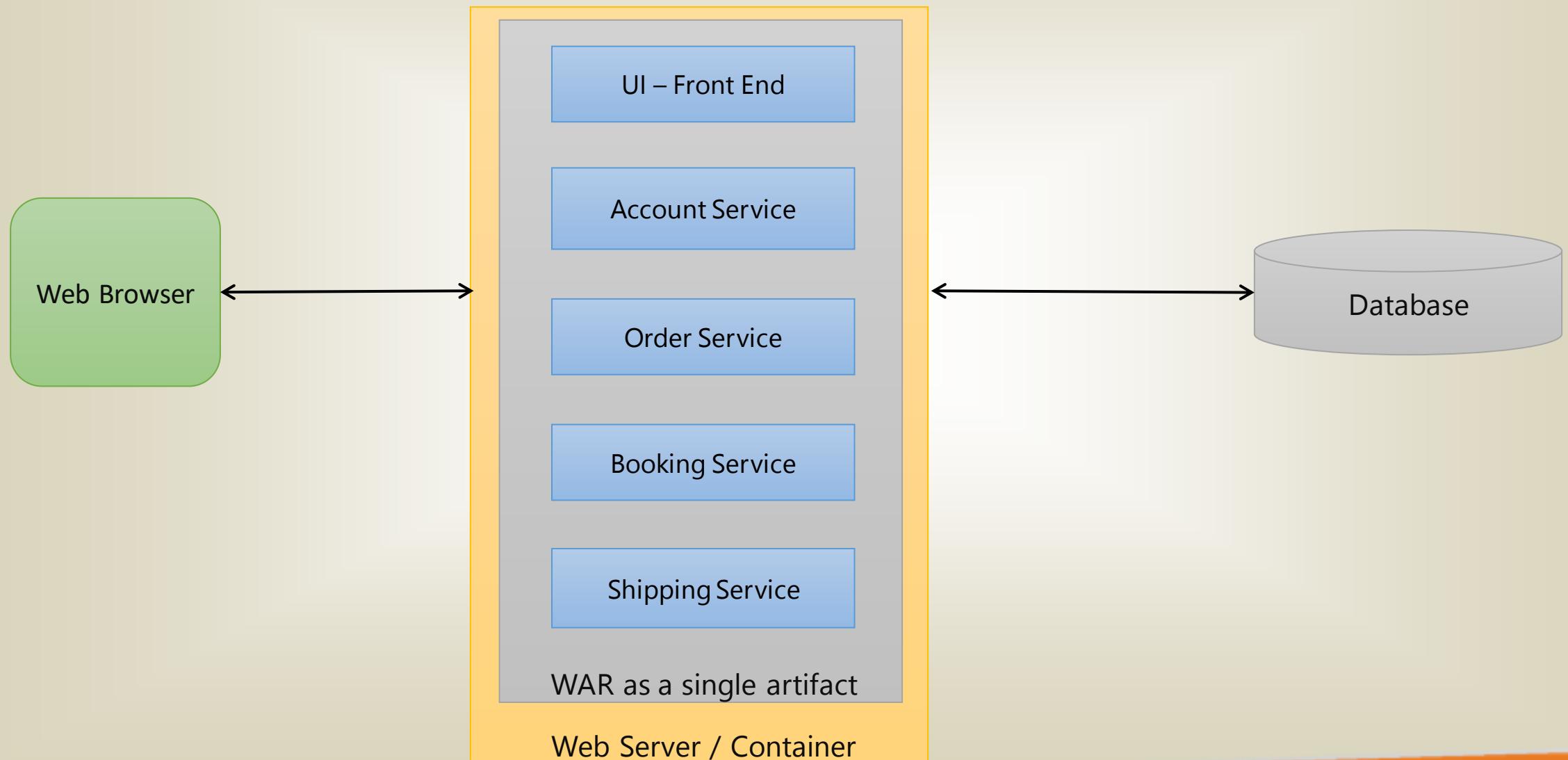
# Micro-Services Session I

Modern Topics in IT(MTIT)  
4<sup>th</sup> Year – Semester 1  
By Isuru Jayakantha

# Evolution from Monoliths to Microservices

- Monolith means composed all in one piece.
- A **monolithic architecture** is one in which a software application is designed to work as a single, self-contained unit.
- The components within a monolithic architecture are interconnected and interdependent, resulting in **tightly coupled** code.
- Presentation, Business logic, Data access, Application integration layers are compacted in a single artifact.

# Monolithic Web Application Example



# Benefits of Monolithic Architecture

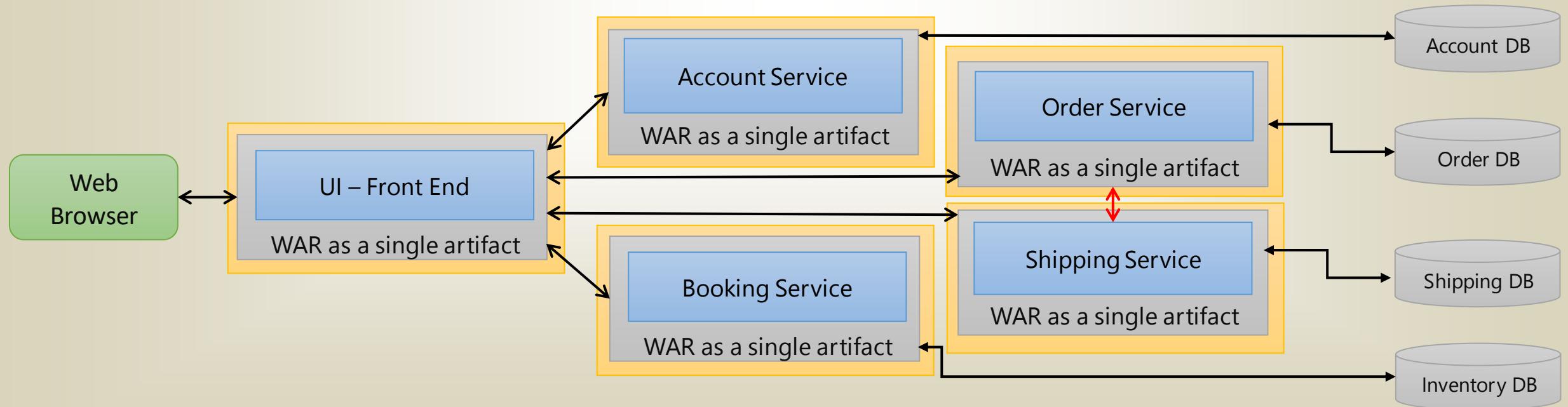
- Simple to develop – Easier to develop at the beginning of the project.
- Simple to test – **End to end testing** is much more easier.
- Simple to deploy – Only required to copy single artifact to the server.
- Simple to scale – **scale horizontally** by running multiple copies behind a load balancer.

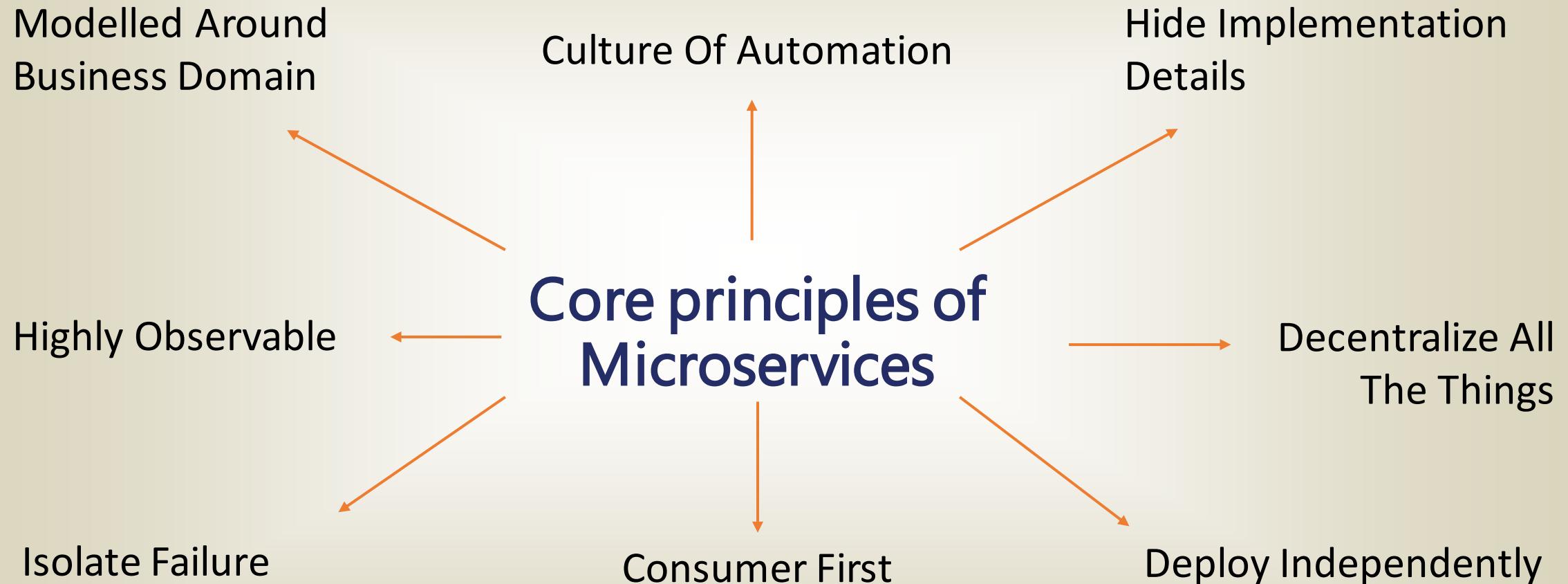
# Drawbacks(අනුජාතු) of Monolithic Architecture

- Maintenance - If Application is too large and complex to understand entirely, it is challenging to make changes fast and correctly.
- Deployment - The **size of the application** can slow down the start-up time.
- Monolithic applications can also be challenging to scale when different modules have conflicting resource requirements (i.e. when a certain module is required to scale).
- **Redeployment** is required for the entire application on each update.
- Difficulty to adopting new and advance technologies.

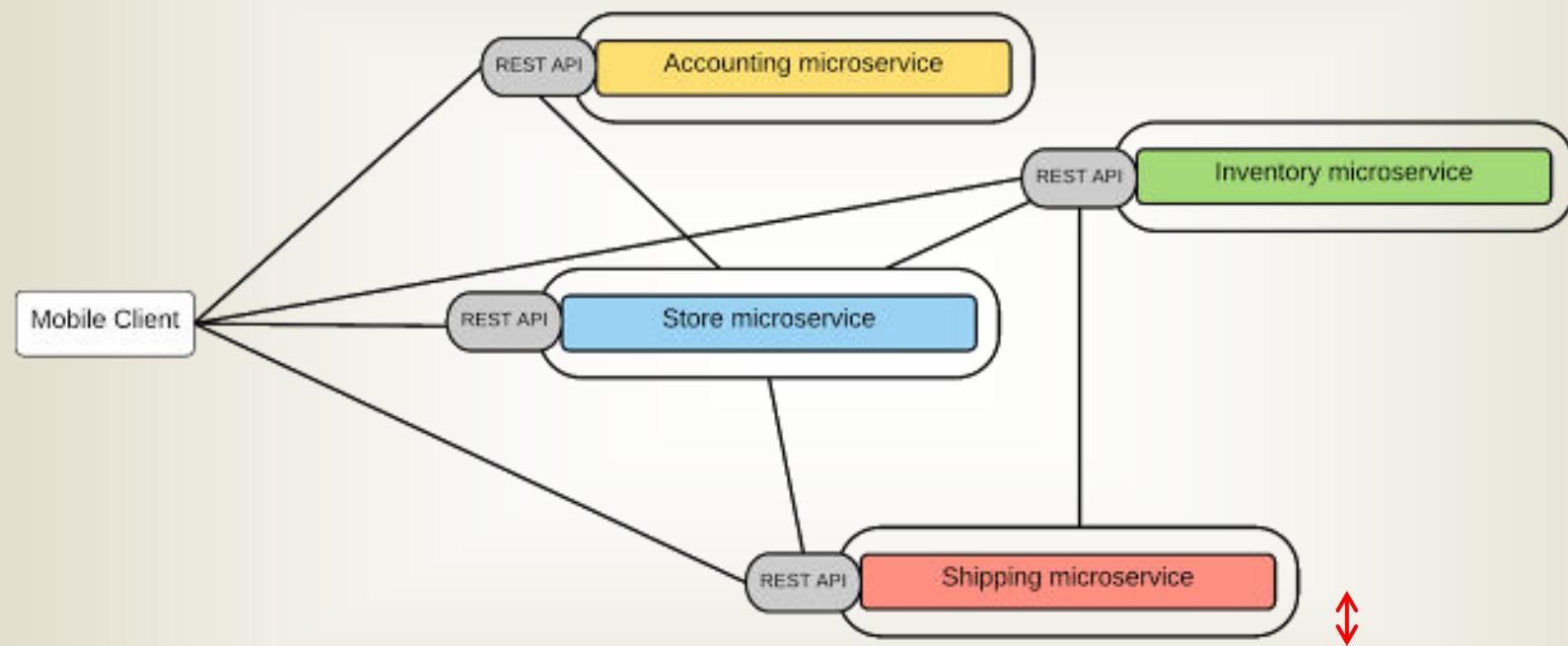
# Core principles of Microservices

- Microservices, aka Microservice Architecture, is an architectural style that structures an application as a collection of **small autonomous services**, modeled around a business domain.



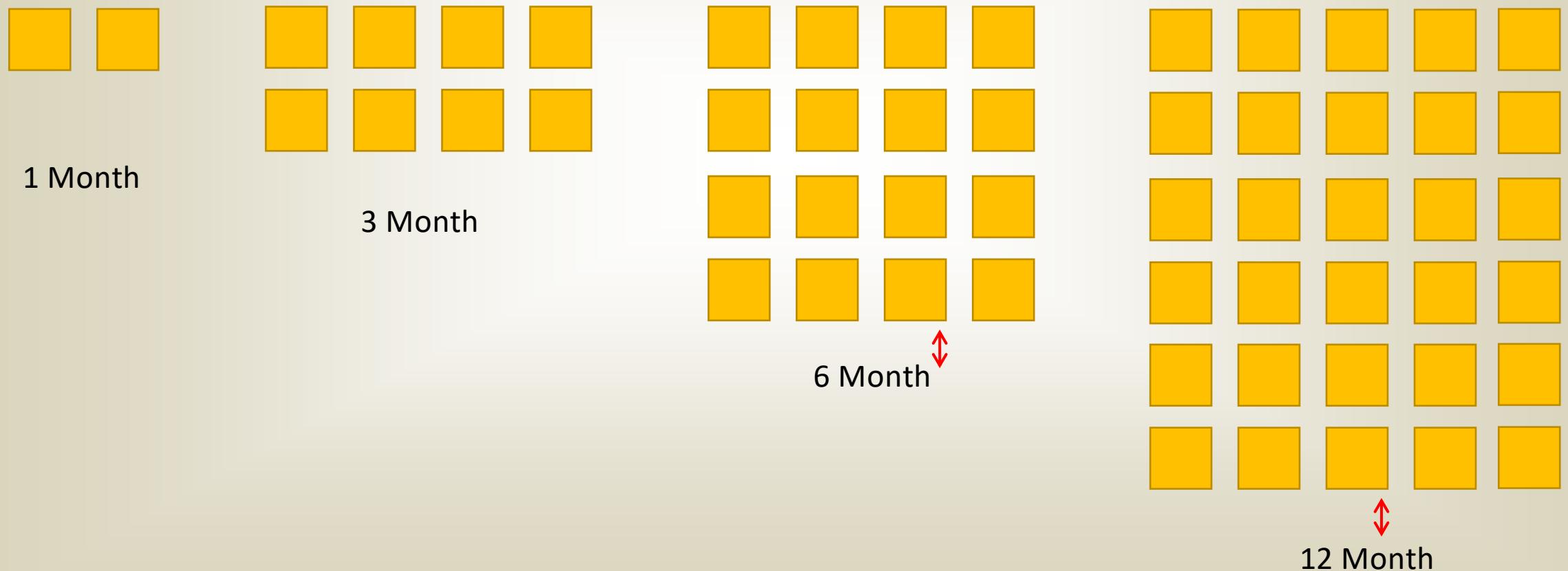


# Modelled Around Business Domain



**Discussion Points :** List down Domain Driven Modules for a Taxi and Cab Booking Platform & Online Food Ordering Platform.

# Culture Of Automation



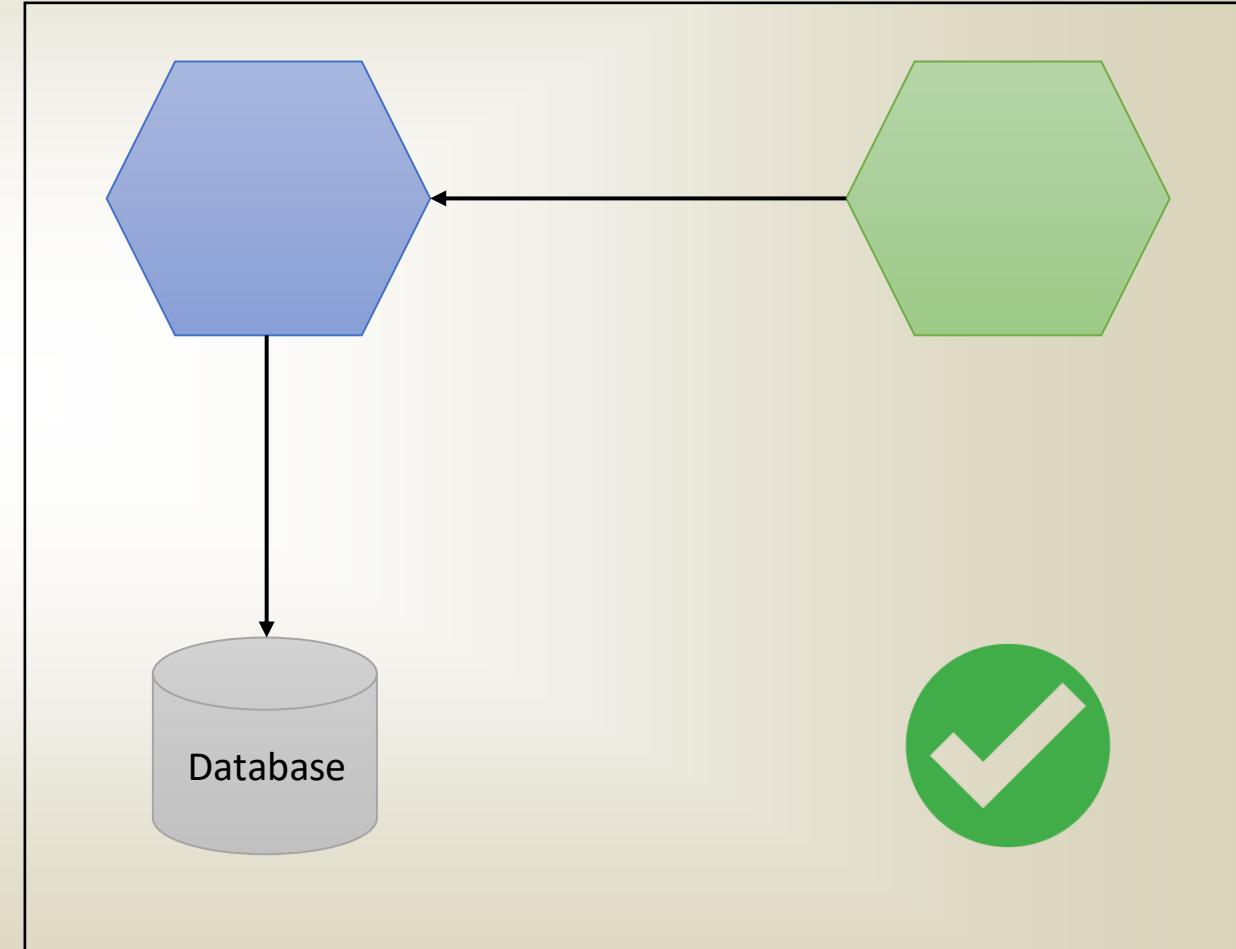
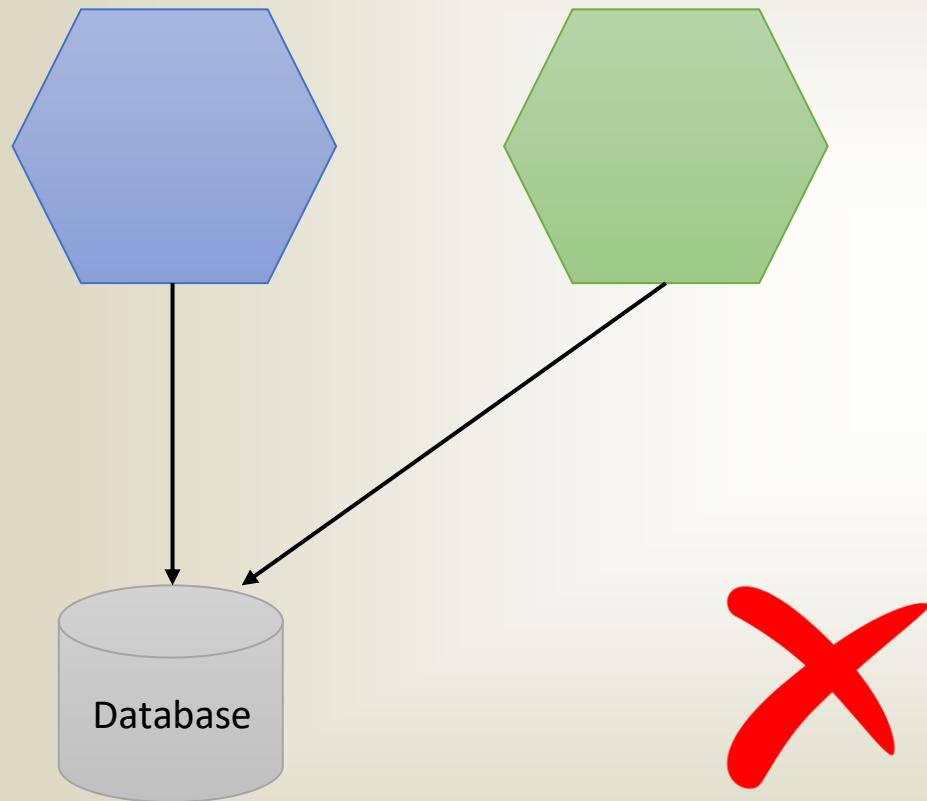
# Culture Of Automation

Infrastructure Automation

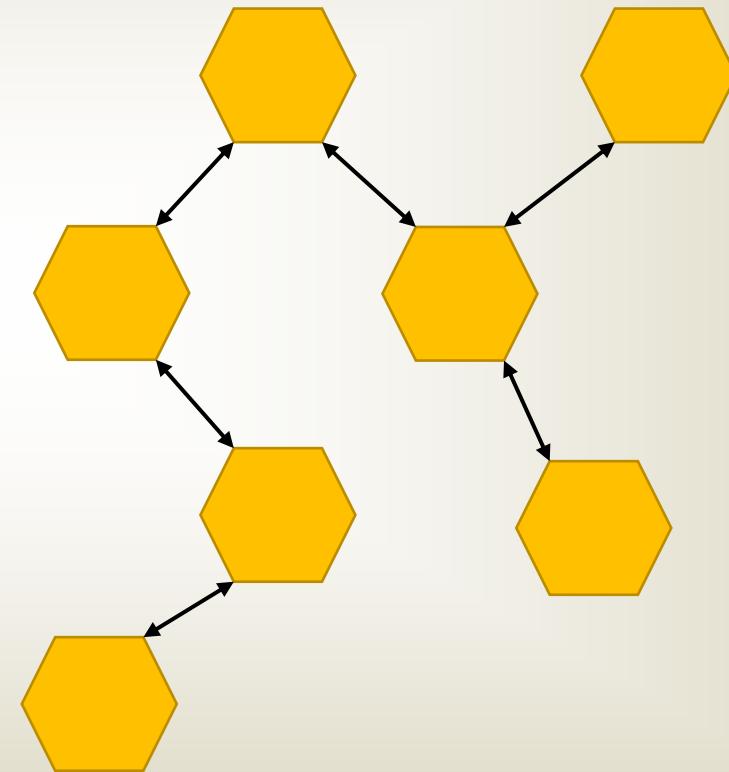
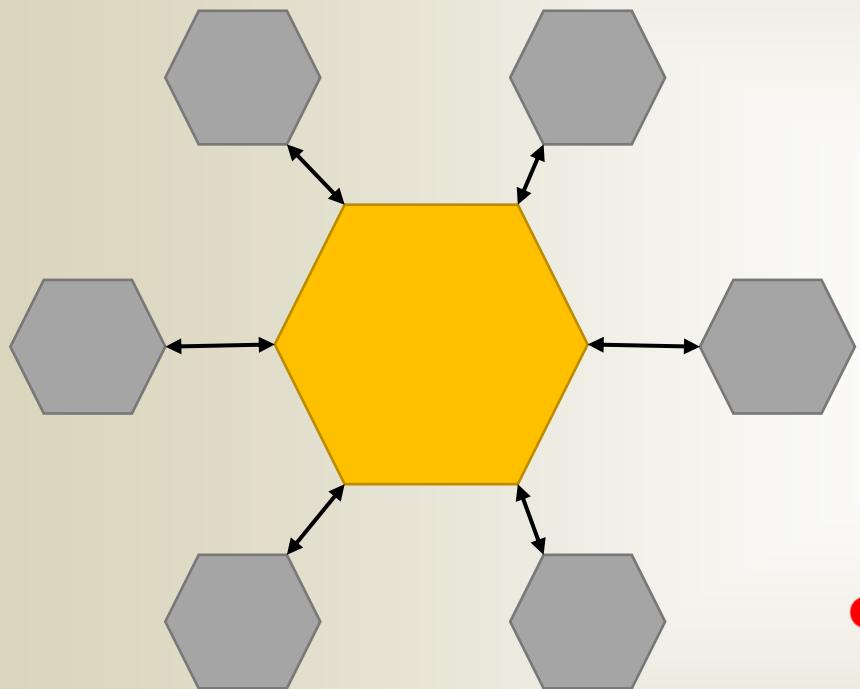
Automation Testing

Continuous Delivery

# Hide Implementation Details

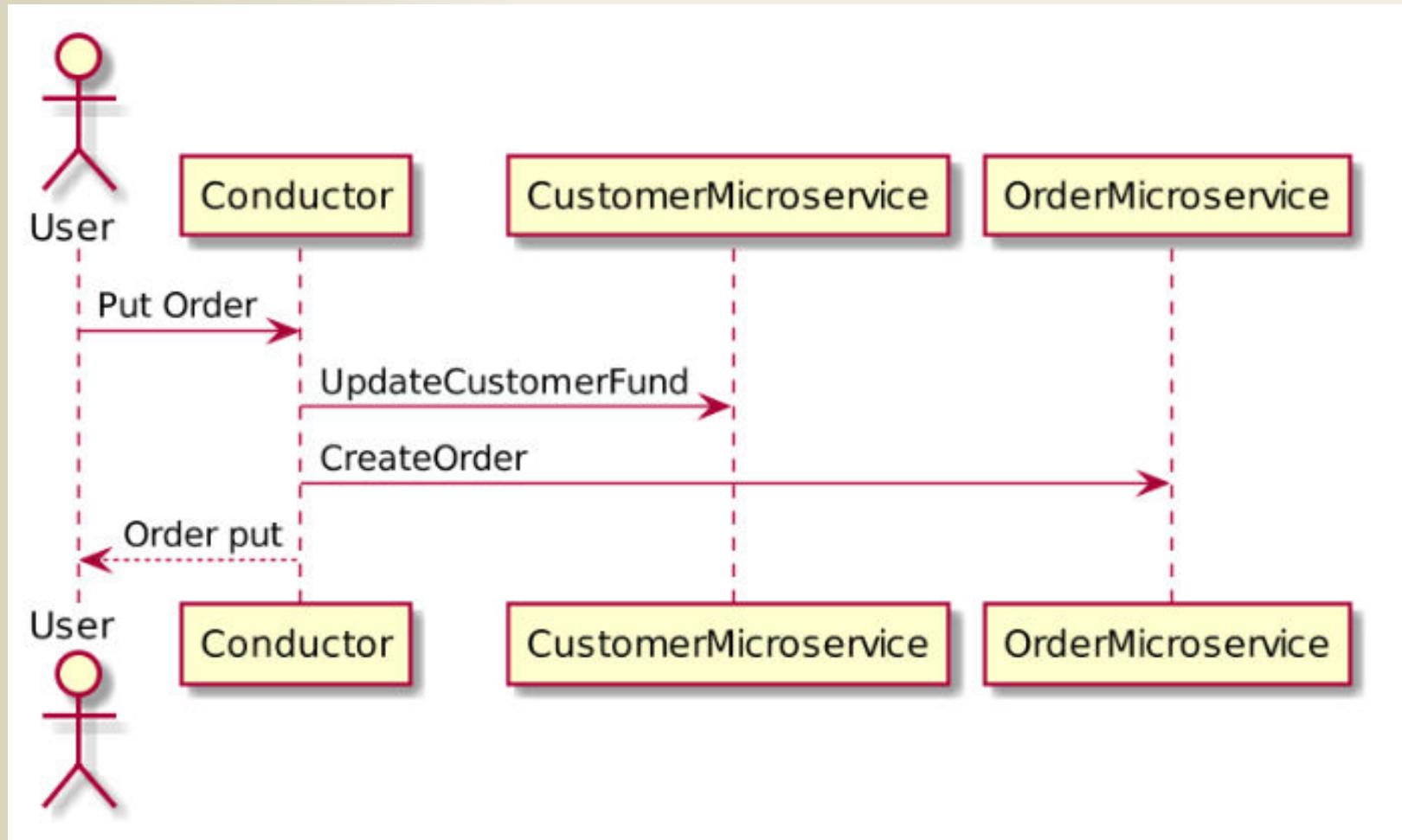


# Decentralize All The Things



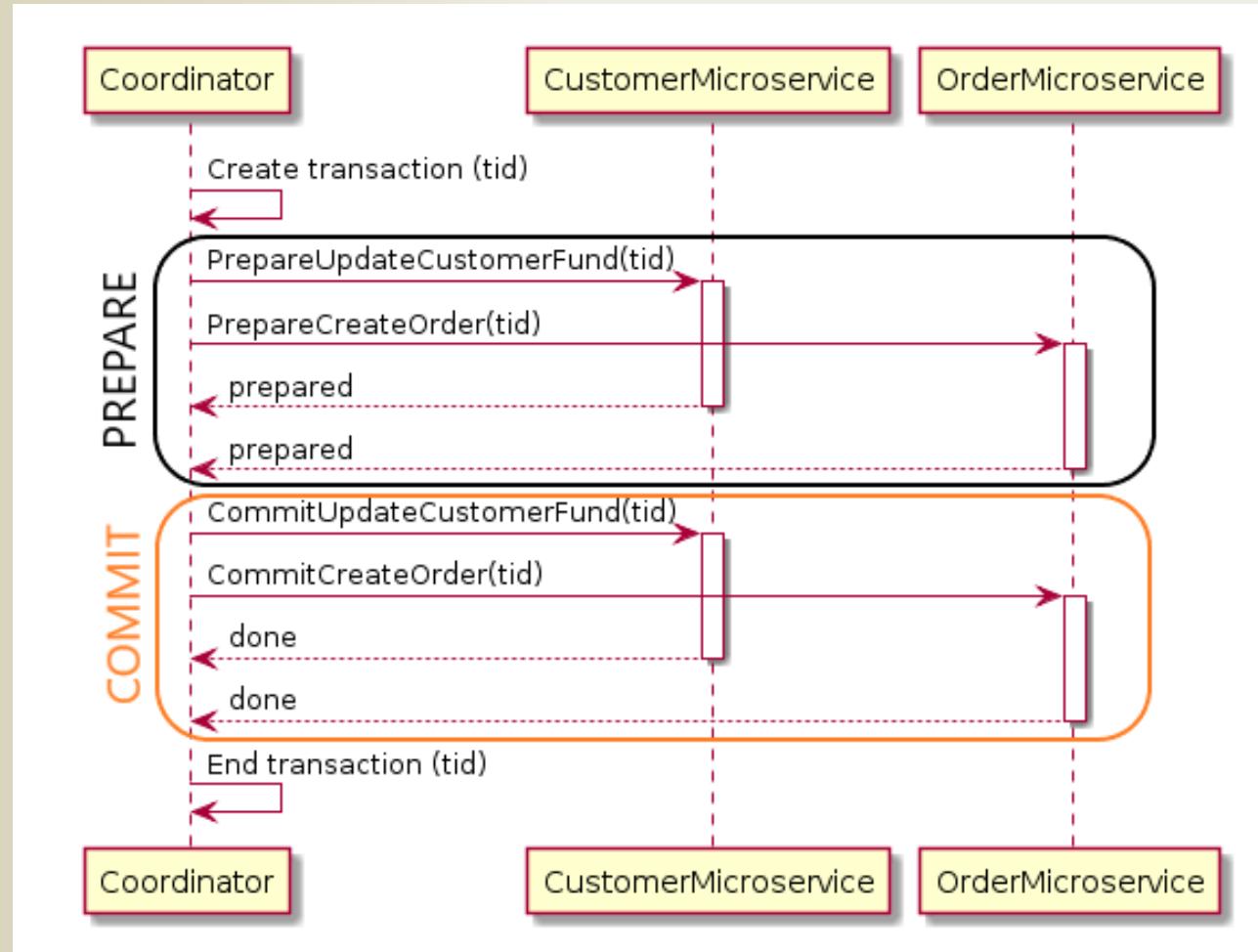
# Decentralize All The Things

Ensuring data consistency across services



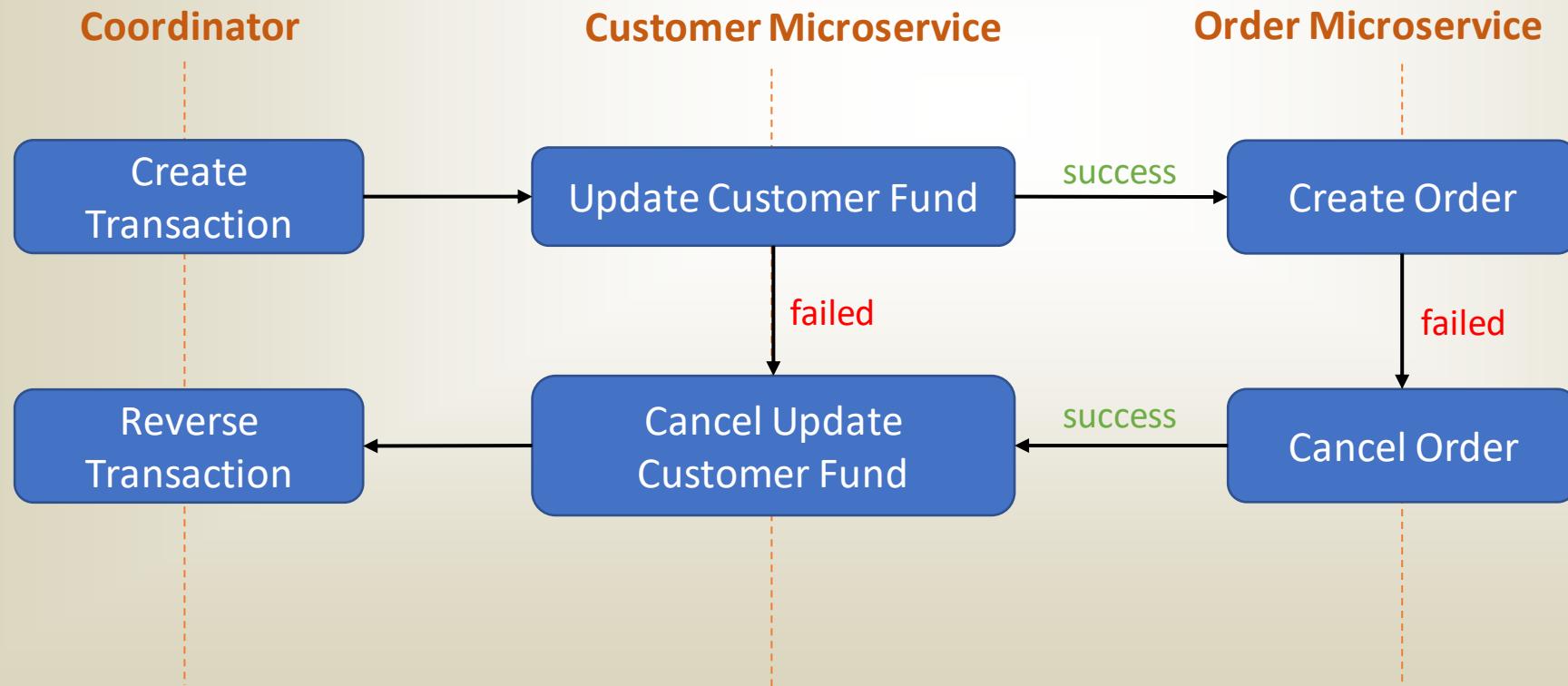
# Two-Phase Commit

2PC consists of two stages, one is “PREPARE” and “COMMIT”.

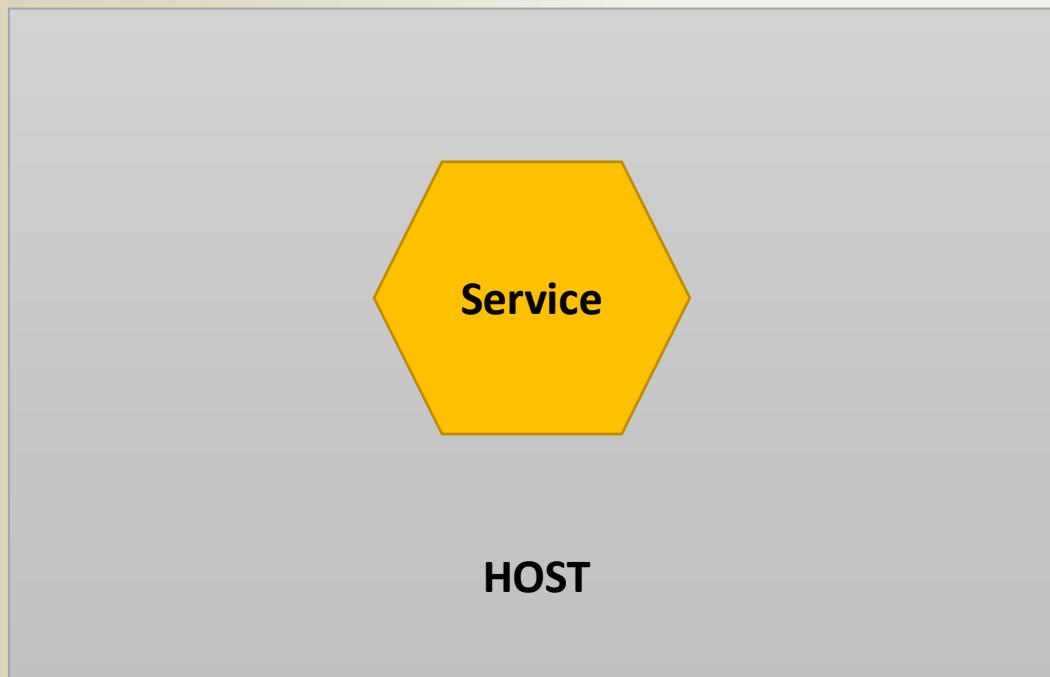


# SAGA

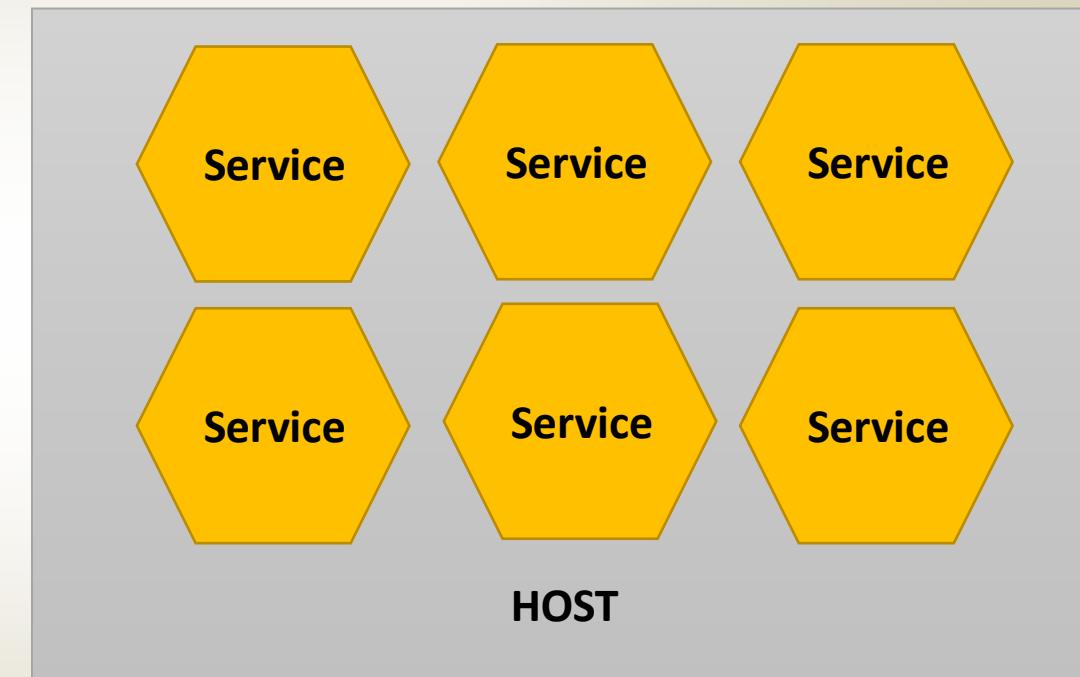
SAGA commits multiple compensatory transactions at different stages ensuring to rollback when required.



# Deploy Independently



**Service instance per host**

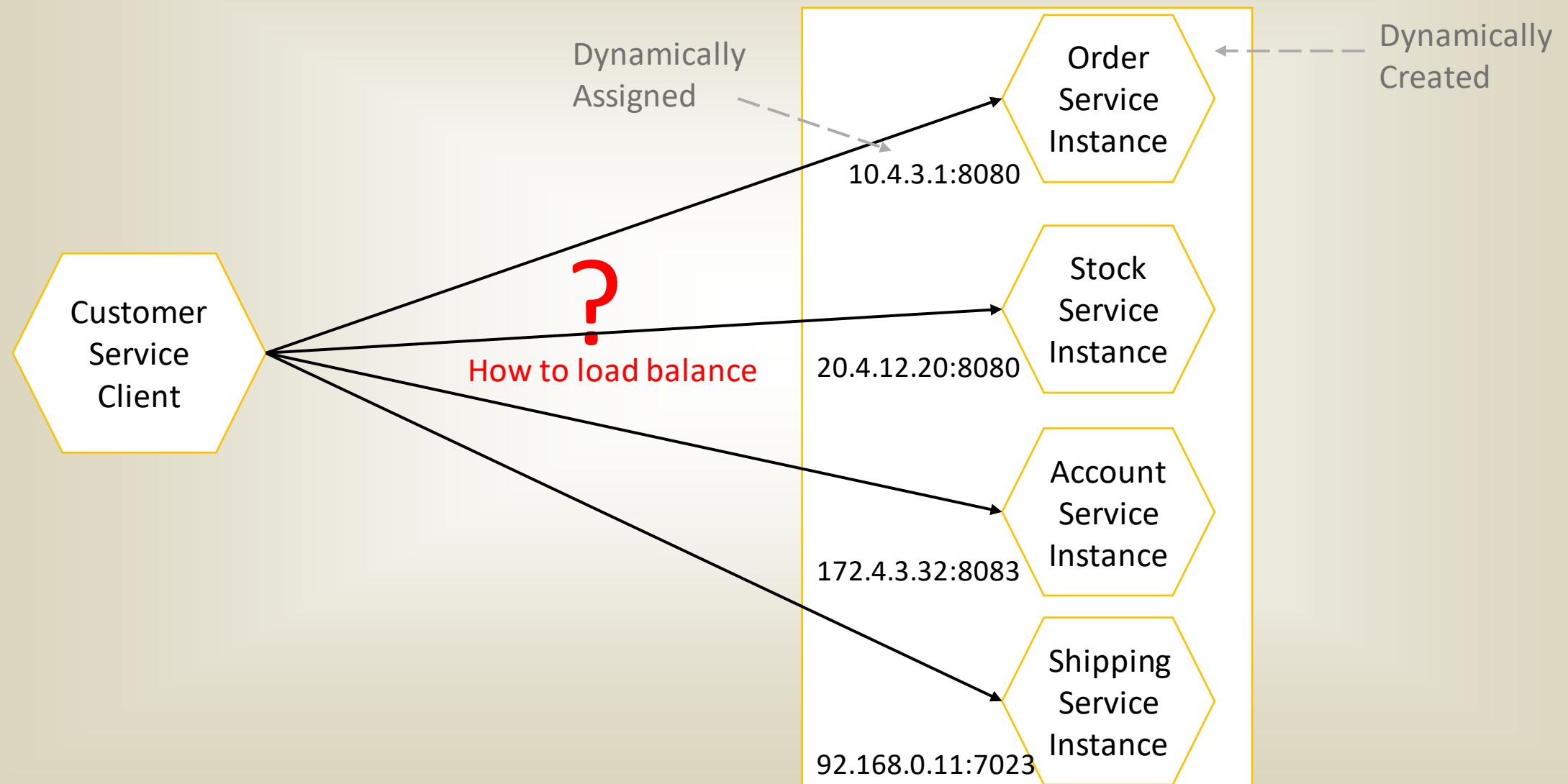


**Multiple service instances per host**

# Consumer First

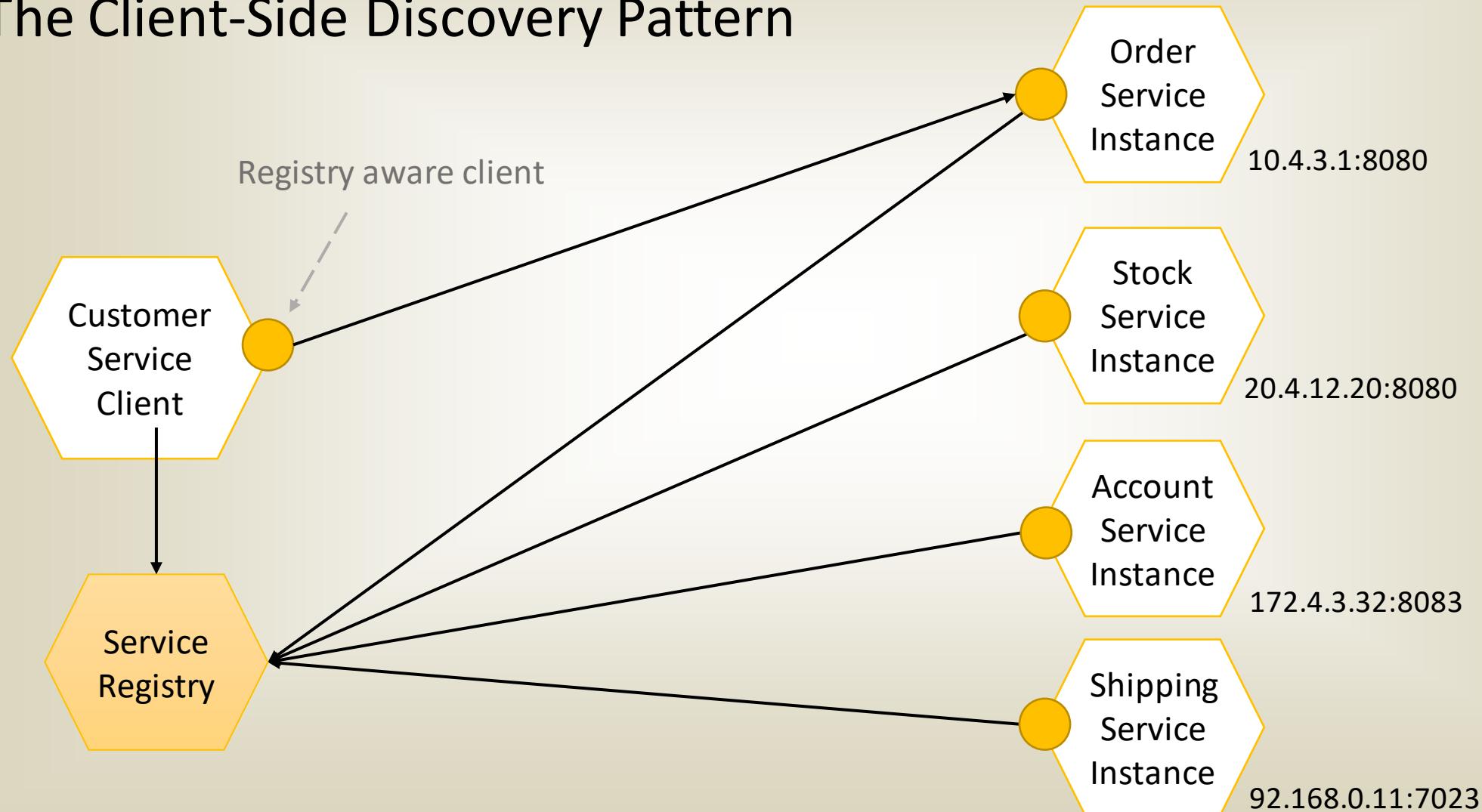
- Identify Consumers
- Method of Communication
- Documentation
- Standards and Best Practices
- Service Discovery

# Service Discovery : The Problem Context



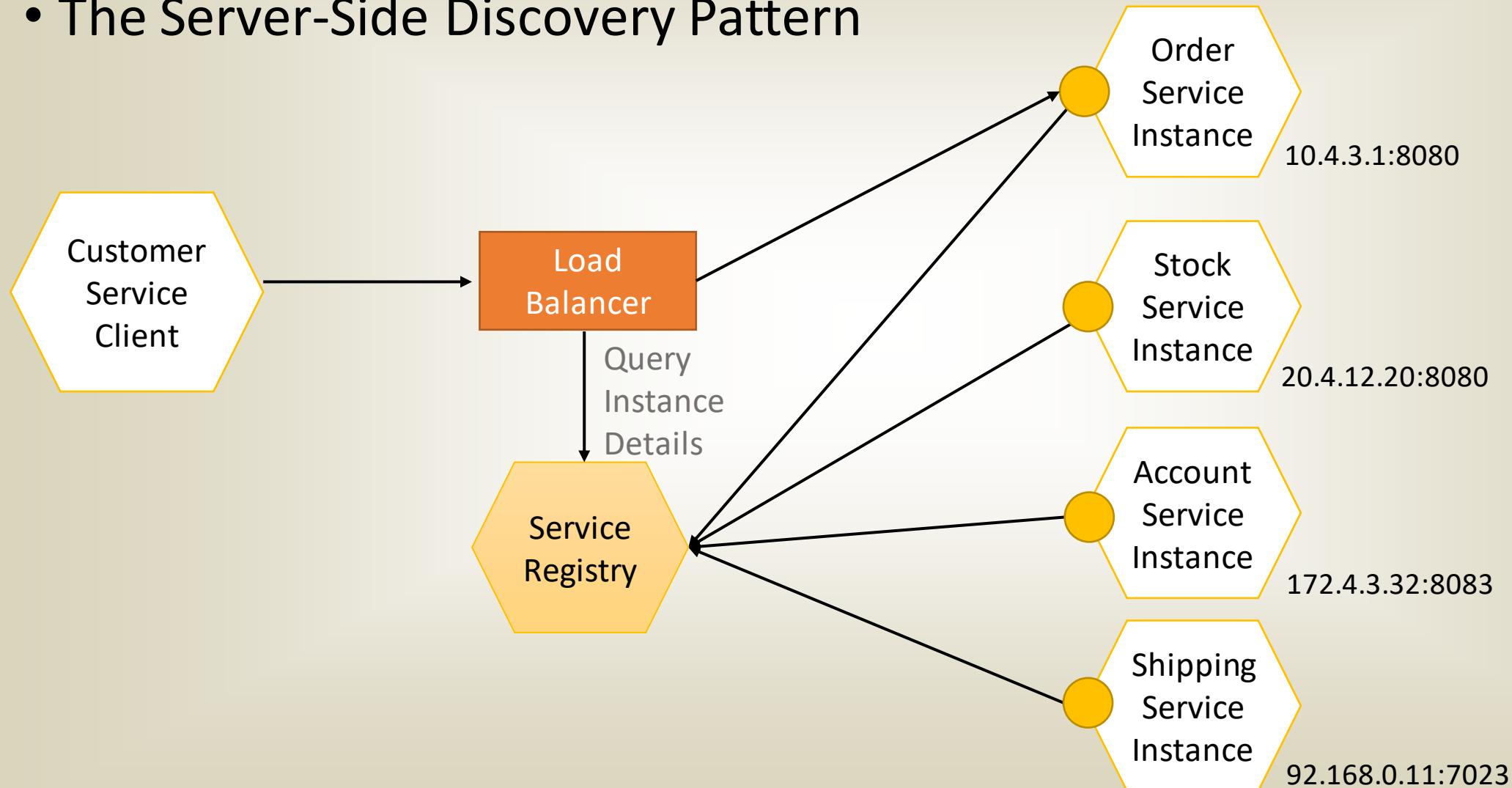
# Service Discovery Patterns

- The Client-Side Discovery Pattern



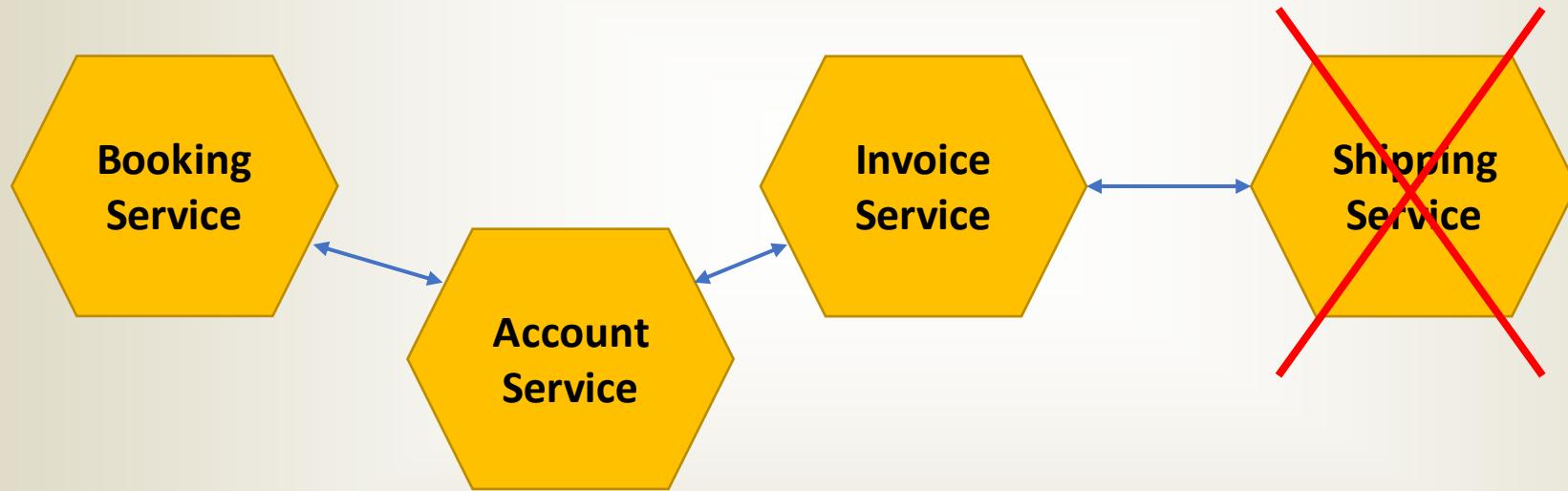
# Service Discovery Patterns

- The Server-Side Discovery Pattern



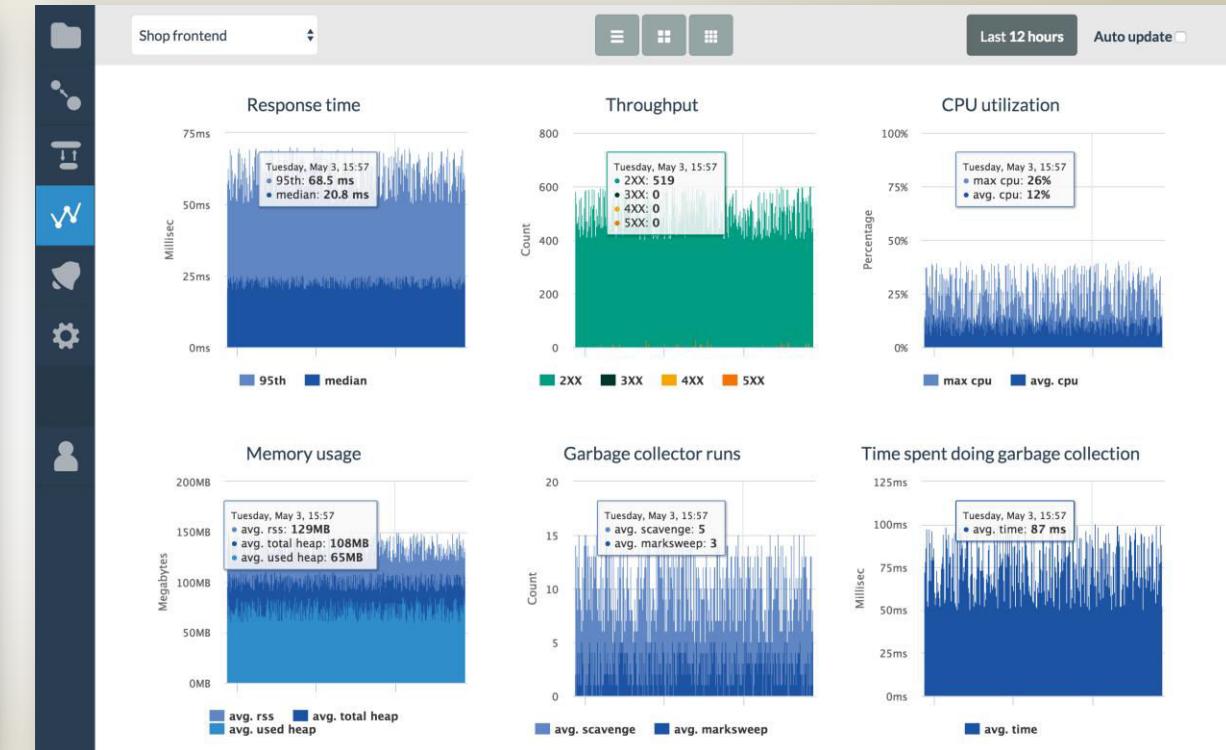
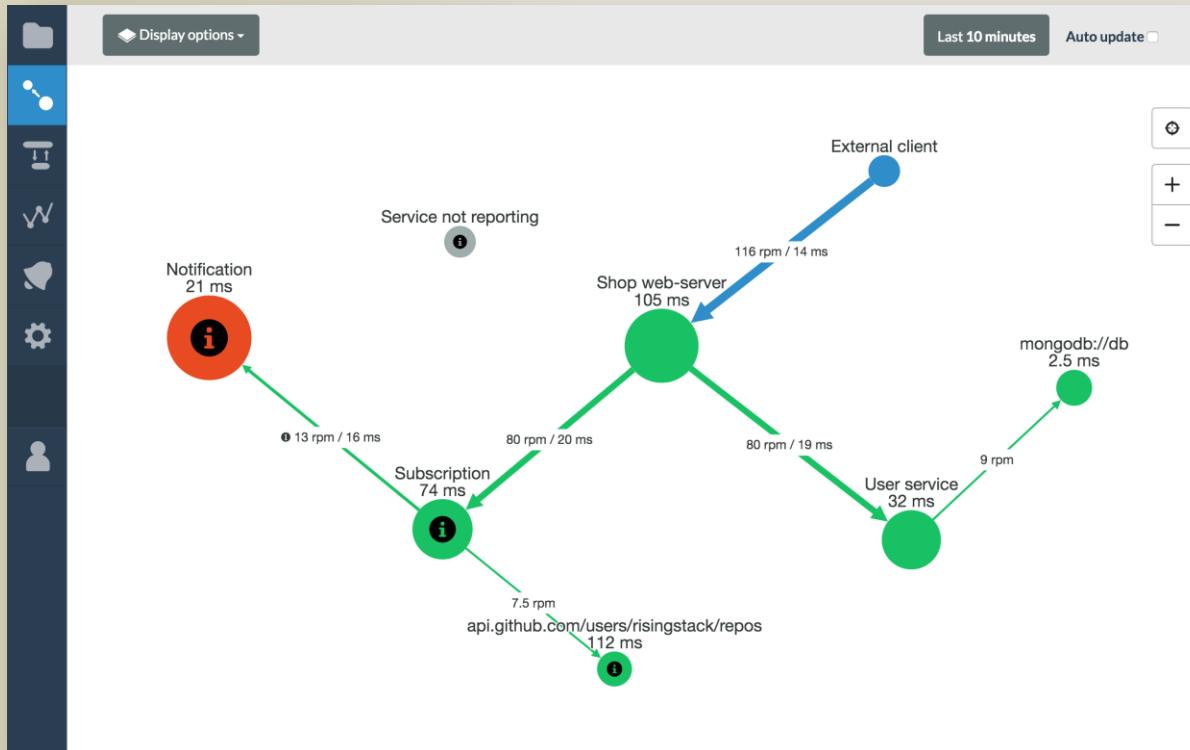
# Isolate Failure

One of the best advantages of a microservices architecture is that you can isolate failures and achieve graceful service degradation as components fail separately.



**Discussion Points:** When one service synchronously invokes another there is always the possibility that the other service is unavailable or is exhibiting such high latency it is essentially unusable. This might lead to resource exhaustion, which would make the calling service unable to handle other requests. How to prevent a network or service failure from cascading to other services?

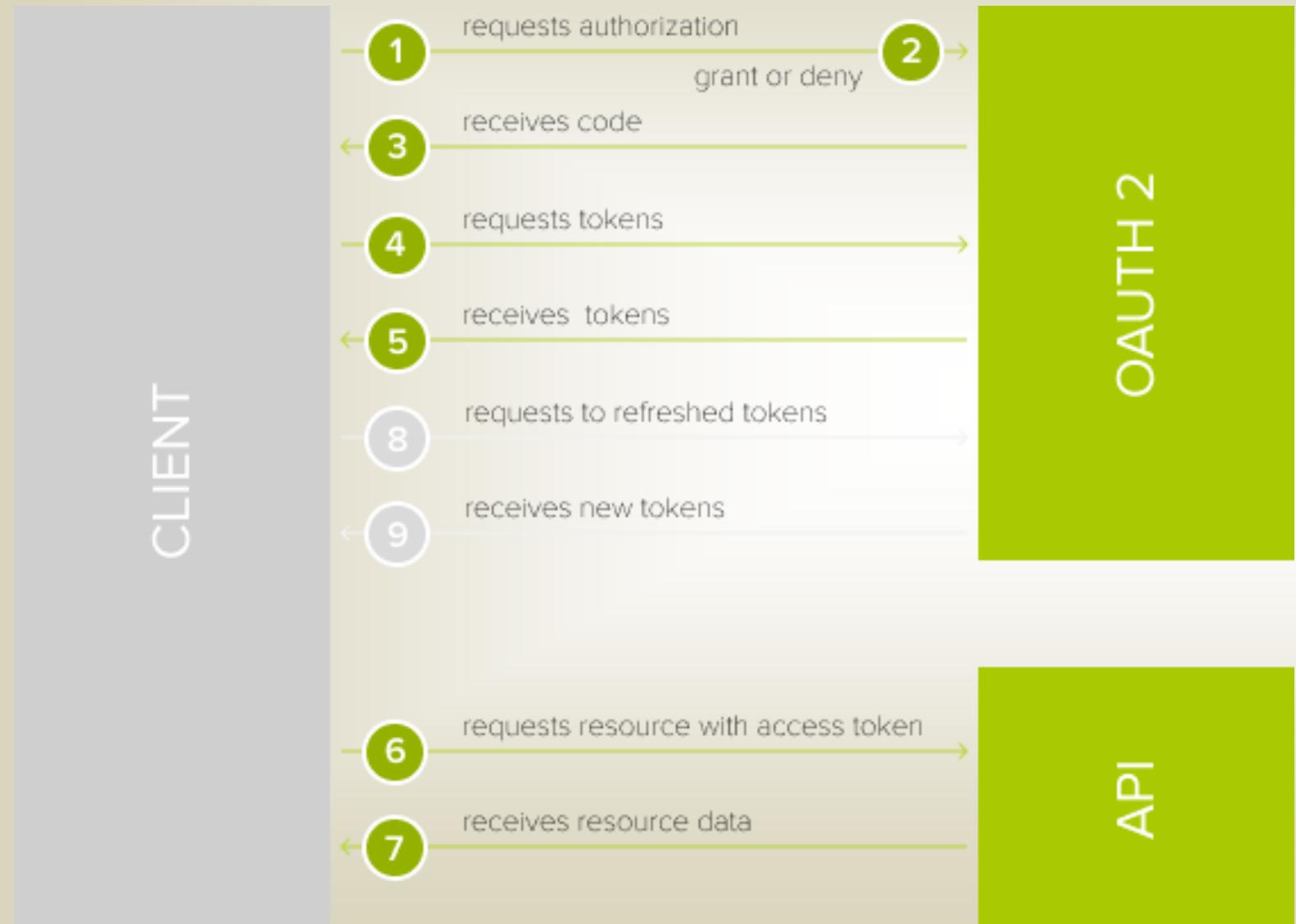
# Highly Observable



**Discussion Points:** How to uniquely identify each 'customer order', or the equivalent in your system when request travels through multiple microservices ?

# Lab Session - I

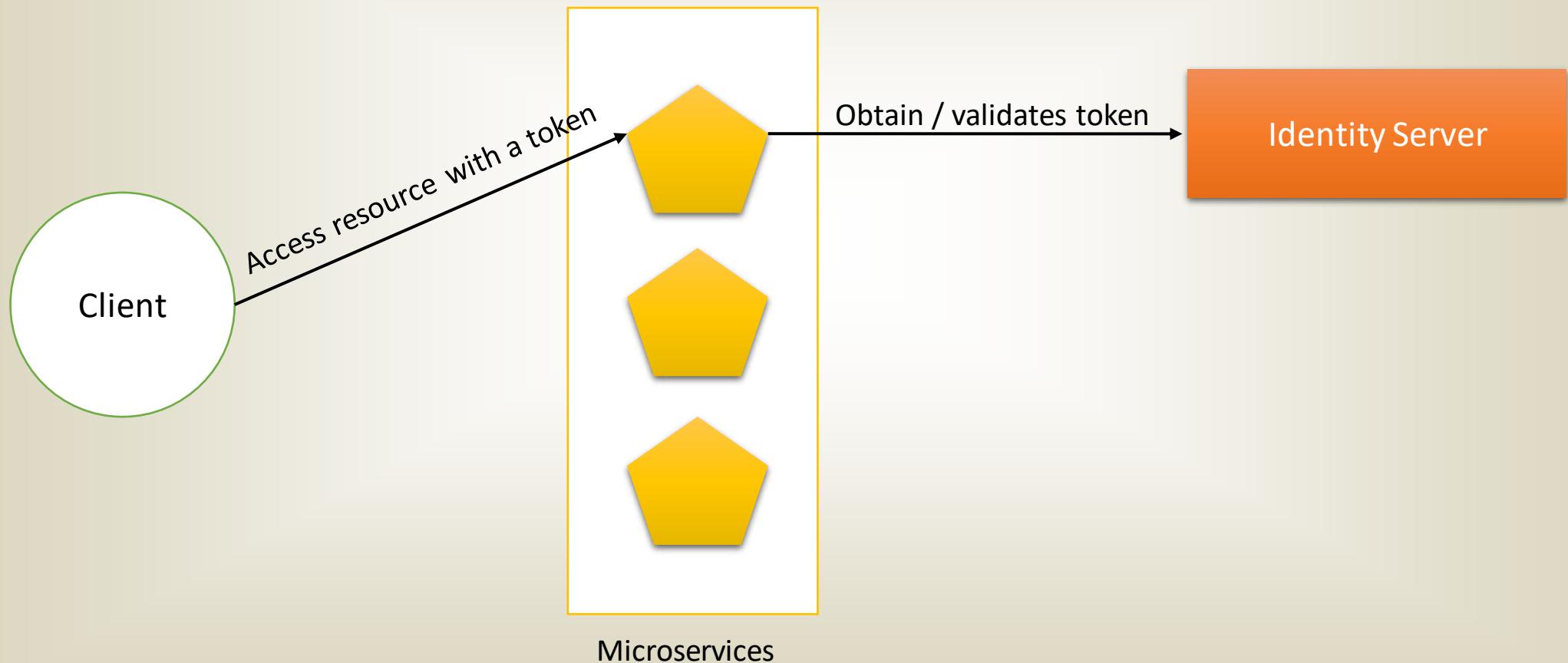
# OAUTH 2



# Securing Microservices

- Use TLS-protocols for all APIs
- Profile All APIs
- Use OpenID or OAuth 2.0, etc.
- Encrypt Sensitive Data
- Protect APIs From Denial-Of-Service-Attacks
- Rate Limiting (Throttling)
- The token expiry times should be as short as possible

# Securing Microservices



# Lab Session - II

# Principles of Microservice Scalability and Performance

- **Portability**

Ability to deploy quickly.

- **Elasticity**

Ability to locate both a single and multiple microservices on a single host.

- **Availability**

Ability to be replicated and spread across data-centers and geographical distances.

- **Robustness**

Implement some fault-tolerant mechanisms that can detect possible failures in microservices they depend on in order to prevent cascading failures.

# Best Practices for Designing a Microservices Architecture

- Create a Separate Data Store for Each Microservice  
Teams can work independently.
- Keep Code at a Similar Level of Maturity  
Maintain versioning for changes.
- Deploy in Containers  
One tool to deploy everything.
- Document API's
- Asynchronous Messaging
- Applying domain-driven design principles
- Use Service registry  
Registration, Heartbeats, Service discovery, De-registration.
- Integration Testing is a must

# References

- Software Architect's Handbook by Joseph Ingino
- Production-Ready Microservices by Susan J. Fowler
- Hands-On Microservices - Monitoring and Testing By Dinesh Rajput
- Microservices Best Practices for Java By Michael Hofmann, Erin Schnabel and Katherine Stanley



# Modular Development

# Open Service Gateway Initiatives

**Modern Topics in Information Technology**

**4<sup>th</sup> Year – Semester 1**

**Lecture 1**

**By Udara Samaratunge**

# What is OSGi?

- OSGi ([Open Services Gateway initiative](#))
- OSGi is a **framework** which allows [modular development](#) of applications using java.
- A Java framework for developing (remotely) deployed service applications, that require:
  - Reliability
  - Large scale distribution
  - Wide range of devices
  - Collaborative
- Created through a collaboration of industry leaders
  - IBM, Ericsson, Nokia, Sony, Telcordia, Samsung, ProSyst,
  - Gatespace, BenQ, Nortel, Oracle, Sybase, Espial, and many more

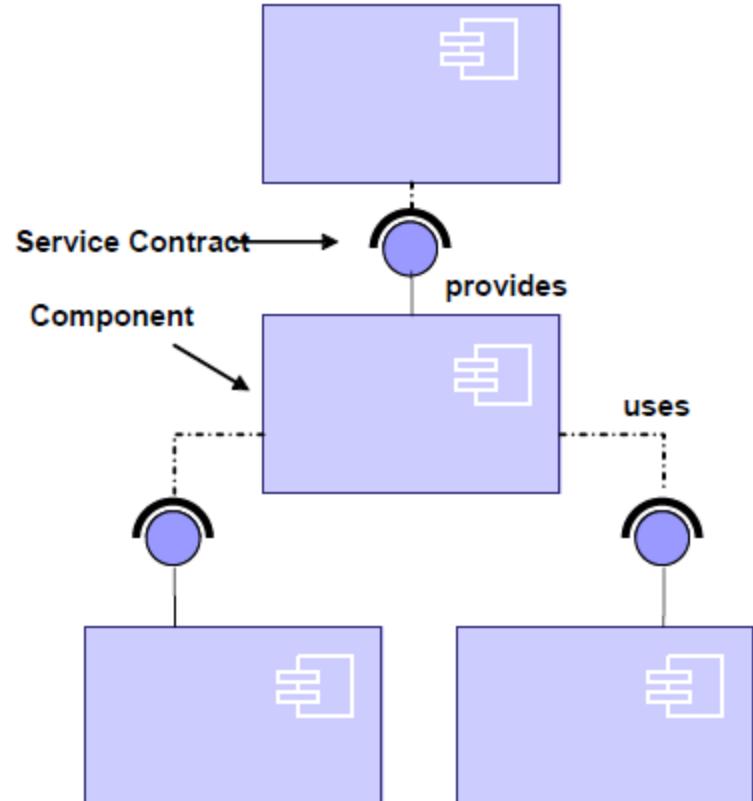
- OSGi containers allow you to **break your application into individual modules**. (are jar files with additional meta information and called **bundles** in OSGi terminology)
- Manage the **cross-dependencies** between modules.
- An OSGi framework then offers you dynamic loading/unloading, configuration and control of these bundles - without requiring restarts.
- Major Framework vendors are
  - ProSyst,
  - Gate space Telematics, and
  - IBM
  - Siemens
  - Espial
- Open source implementations
  - Apache Felix
  - Eclipse Equinox
  - Gate space Knopflerfish

# Why OSGi Service Platform use?

- What problems does the OSGi Service Platform address?
  - The limited (binary) software **portability problem**
  - The **complexity of building** heterogeneous software systems
    - Supporting the myriad of **configuration, variations, and customizations** required by today's devices
  - Managing the software on the device

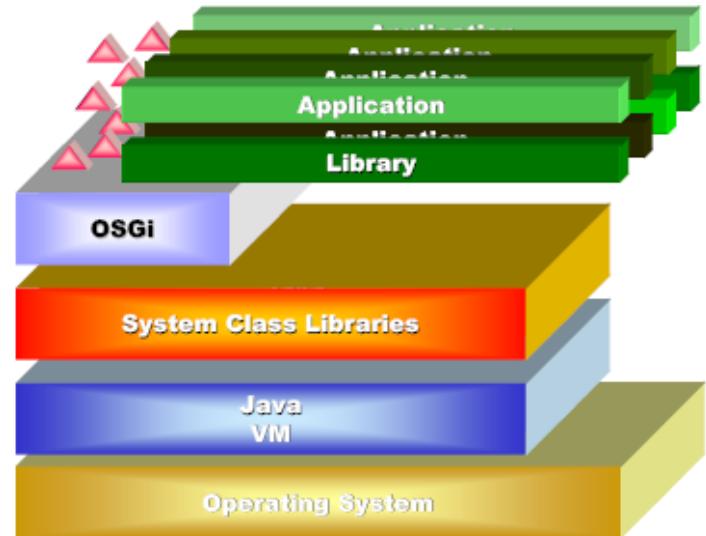
# The OSGi Platform vs Service Oriented Architectures (SOA)

- Separate the **contract** from the implementation
- Allows alternate implementations
- **Dynamically discover** and **bind** available implementations
- Binding based on **contract** (interface definitions)
- Components are **reusable**
- Components are not coupled to implementation details of other components, only their independent interfaces have to be known

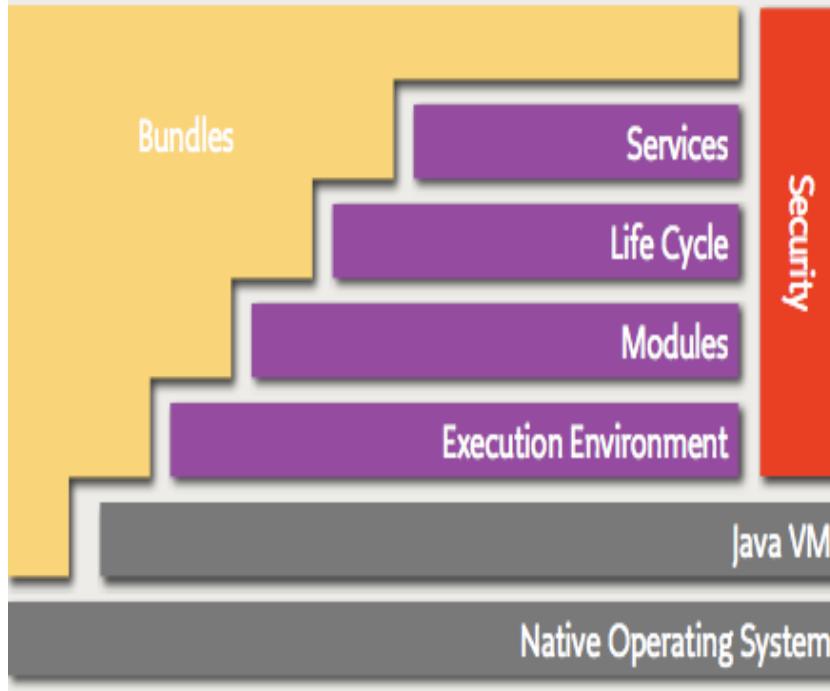


# The OSGi Framework Architecture

- Allows applications to share a single Java VM.
- Handles all class loading in a much better defined way than standard Java.
  - Versioning!
- Gives isolation/security between applications.
- Mediates between communication & collaborations between applications.
- Provides life cycle management (install, start, stop, update, etc).
- Policy free
  - Policies are provided by bundles



# OSGi Bundle Architecture

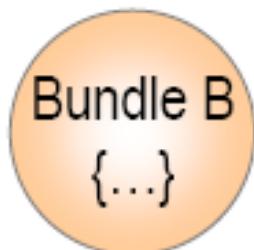
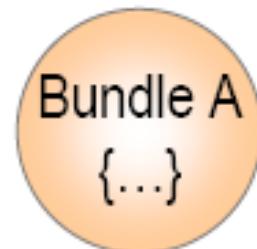


- The following list contains a short definition of the terms:
  - **Bundles** - Bundles are the OSGi components made by the developers.
  - **Services** - The services layer connects bundles in a dynamic way by offering a *publish-find-bind model* for *plain old Java objects*.
  - **Life-Cycle** - The API to install, start, stop, update, and uninstall bundles.
  - **Modules** - The layer that defines how a bundle can import and export code.
  - **Security** - The layer that handles the security aspects.
  - **Execution Environment** - Defines what methods and classes are available in a specific platform.

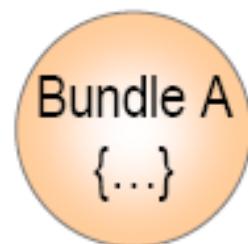
# What is a Bundle?

- In Java terms, a **bundle** is a plain old JAR file.
- In standard Java everything in a JAR is completely visible to all other JARs.
- But OSGi **hides everything** in that JAR unless explicitly **exported**.
- **Reason for hiding** is to **maintain multiple versions** of the same library.
- **By default**, there is no sharing.

- A *bundle* is the deliverable application
  - Like a Windows EXE file
  - Content is a JAR file
- A bundle registers zero or more services
  - A service is specified in a Java interface and may be implemented by multiple bundles
  - Services are bound to the bundle life-cycle
- Searches can be used to find services registered by other bundles
  - Query language

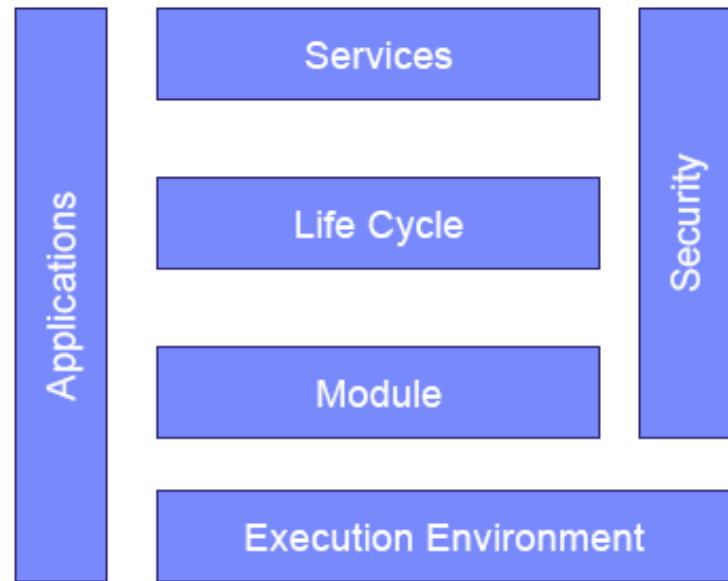


- A Bundle contains (normally in a JAR file):
  - Manifest (bundle meta data)
  - Code (classes in packages)
  - Resources (other files in the JAR file)
- The Framework:
  - Reads the bundle's manifest
  - Installs the code and resources
  - Resolves dependencies
  - Controls the bundle life cycle
- During Runtime:
  - Calls the Bundle Activator to start the bundle
  - Manages java class path for the bundle as a network of class loaders
  - Handles the service dependencies
  - Calls the Bundle Activator to stop the bundle
  - Cleans up after the bundle



# OSGi Service Platform Layering

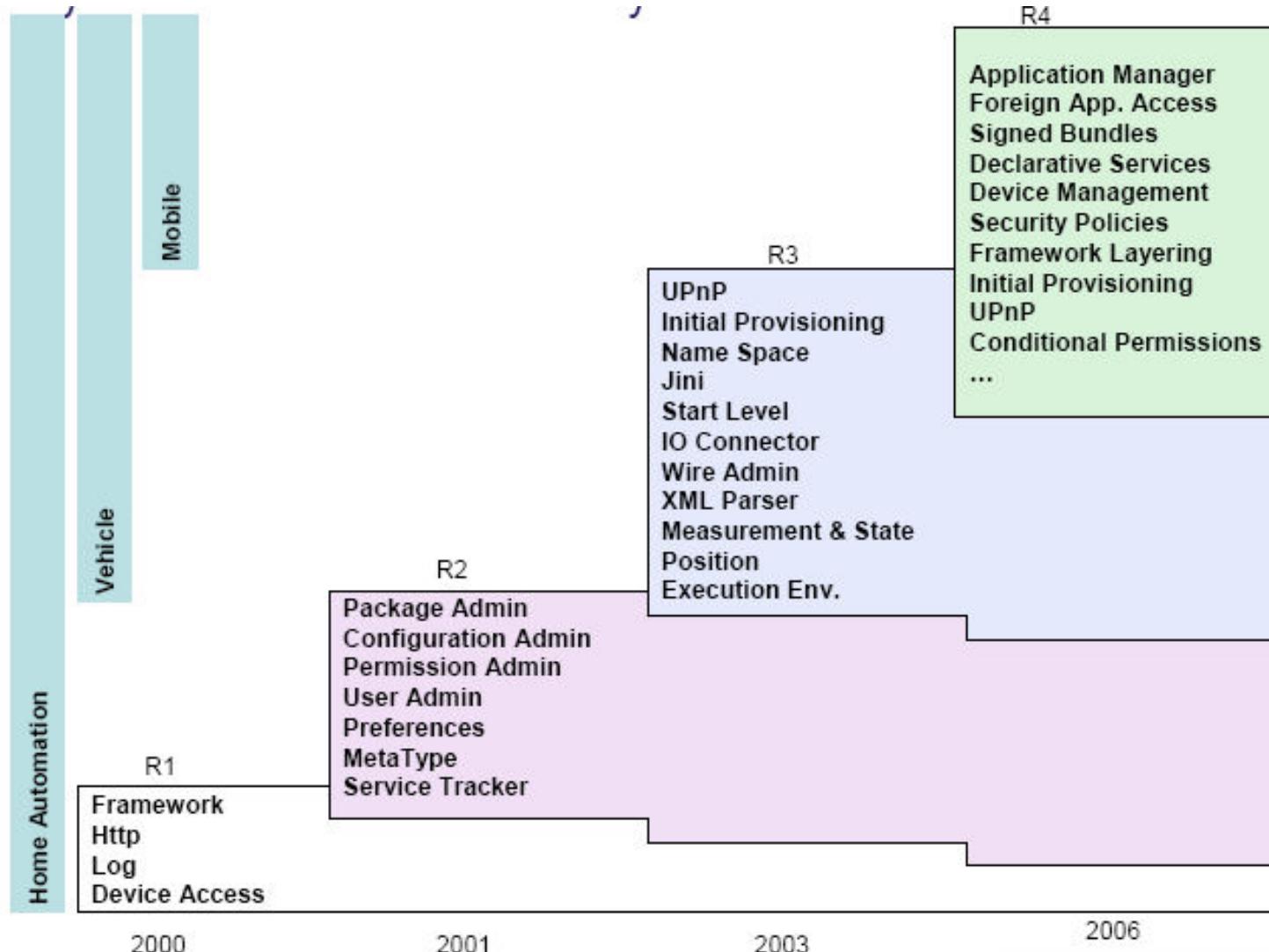
- The OSGi Service Platform is divided in a number of layers.
- Execution Environment provides a defined context for applications.
- The Module layer provides class loading and packaging specifications.
- The Services layer provides a collaboration model.
- The extensive Security layer is embedded in all layers.



# OSGi Service Layer

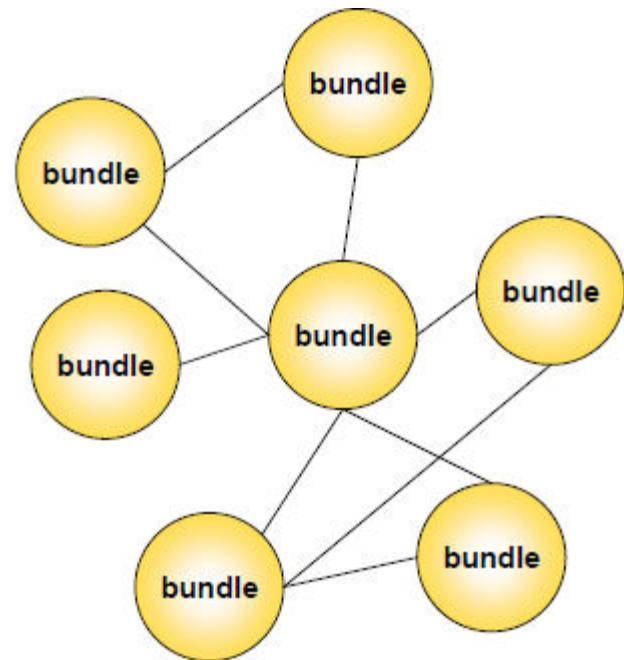
- Provides an inside-VM service model
  - Discover (and get notified about) services based on their interface or properties, no protocol required
  - Bind to one or more services by
    - program control,
    - default rules, or
    - deployment configuration
- Service Oriented Architectures (SOA) Confusion
  - Web services bind and discover over the net
  - The OSGi Service Platform binds and discovers inside a Java VM
- The OSGi Alliance provides many standardized services
- OSGi defines a standard set of services
  - Other organizations can define more (AMI-C, Ertico, JCP)

# The OSGi Service Layer Evolution



# OSGi Module Layer

- Packaging of applications and libraries in Bundles
  - Java has significant deployment issues
- Class Loading modularization
  - Java provides the Class Path as an ordered search list, which makes it hard to control multiple applications
- Protection
  - Java can not protect certain packages and classes from others.
- Versioning
  - Java can not handle multiple versions of the same package in a VM



# The Security Layer

- The OSGi Service Platform can be configured to be one of the most secure execution environments
- The security model is **dynamic**, unlike normal Java
- Fully under control of the operator
  - Fully controlled systems
  - Walled gardens
  - Fully open systems
- Services can require Service Permission if security is enabled
  - Fully under operator control
- A number of extra rules to increase the overall system security

# Practical Example

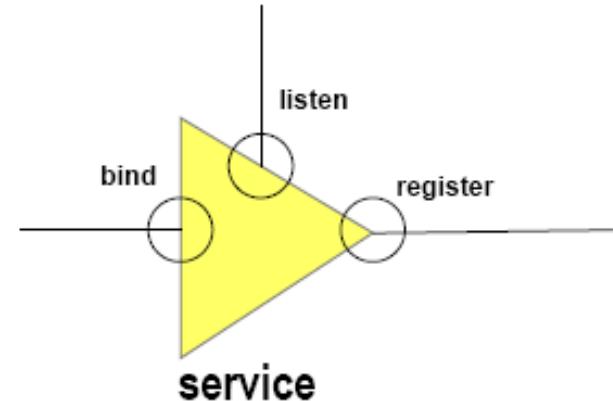
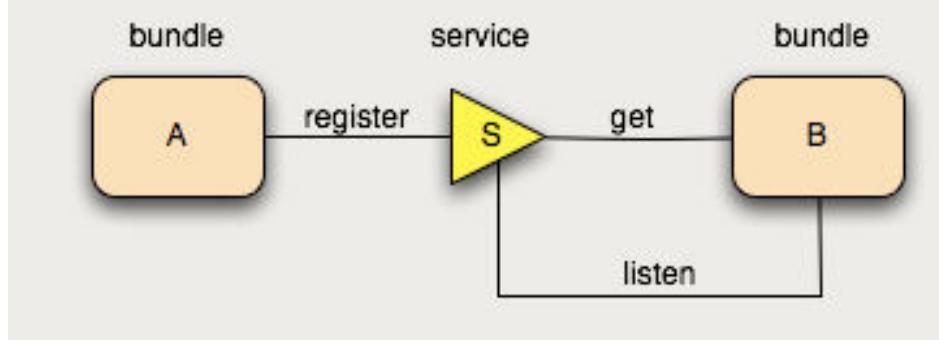
Eg :- I have an application that is interacting with underlying MySQL database and after few month I found that MySQL team has fixed a major bug in their new version of mysql-connector library release so in order to incorporate this new library in my traditional application I have to **stop my application and re-package it (or just replace the older one)**

But, with OSGI we **don't need to stop the whole application** because **everything is exposed** either as a component or as a service therefore we **just need to install new component/service in OSGI container**.

when the **services/components** are updated in OSGI container there are **various event listeners** that propagate the service/component update event to service/component consumers.

And accordingly consumers adapts themselves to use new version of web service (on the **consumer side we need to listen for various events** so that consumers can decide whether to respond for change or not.

# OSGi Services



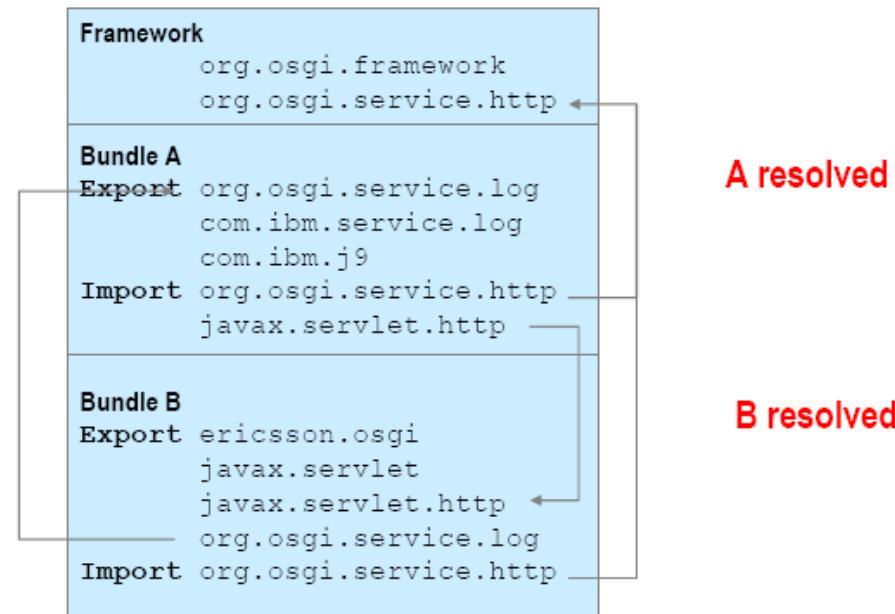
- The Framework **Service Registry** is available to all bundles to collaborate with other bundles.
- Different bundles (from different vendors) can implement the same interface
  - Implementation is not visible to users
  - Allows operator to replace implementations without disrupting service
- A bundle can create an object and register it with the **OSGi service registry** under one or more interfaces.

- Other bundles can go to the **registry** and list all objects that are registered under a specific interfaces or class.
- A bundle can therefore *register a service*, it can *get a service*, and it can *listen for a service* to appear or disappear.
- **Multiple bundles** can register objects under the same interface or class with the same name.
- Services are associated with **properties**
  - Powerful query language to find appropriate service
  - Bundles can update the properties of a service dynamically
- A **bundle** can **decide** to withdraw its service **from the registry** while other bundles are still using this service

- A bundle can use a service (bind to) with any cardinality
  - 1..1, 0..1, 0..n
- A service can be discovered dynamically
  - Active search with query filter
  - Listener interface
- Services are dynamic
  - A bundle can decide to withdraw its service from the registry while other bundles are still using this service. Bundles using such a service must then ensure that they no longer use the service object and drop any references.
- Services can go away at any time! This is very dynamic!

# Bundle Deployment

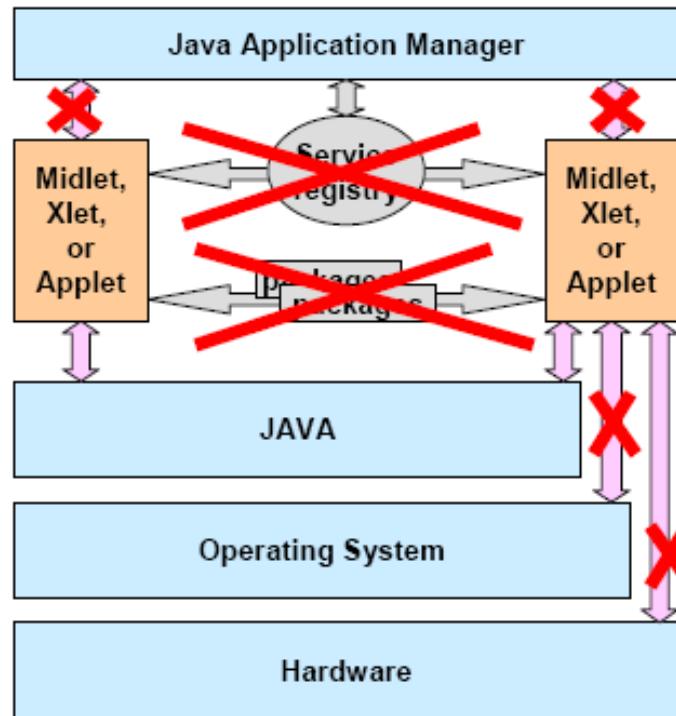
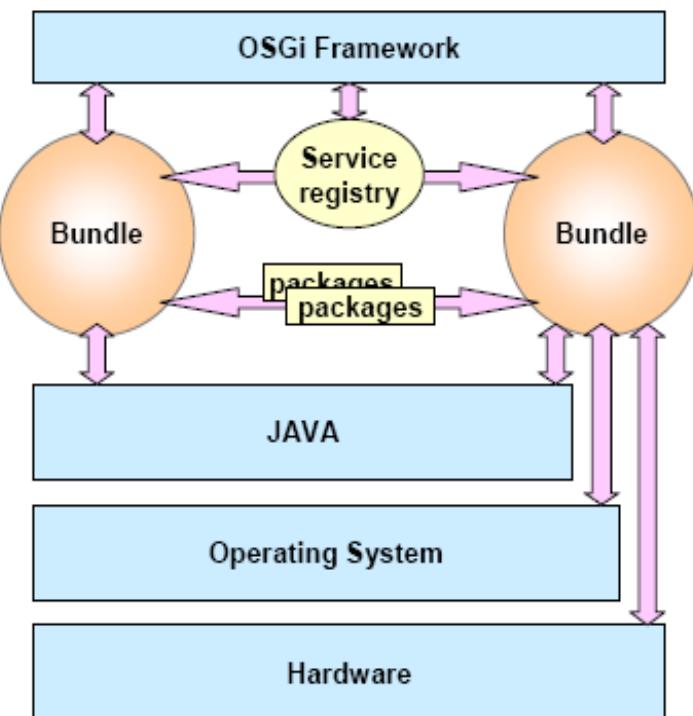
- Bundles are deployed on an **OSGi framework**, the bundle runtime environment.
- This is not a container like Java Application Servers. It is a *collaborative environment*.
- Bundles run in the same VM and can actually share code.
- The framework uses the **explicit imports** and **exports** to wire up the bundles so they do not have to concern themselves with class loading.



# Component interaction and collaboration

- OSGi is more than an Applet, MIDlet, Xlet runner
- Bundles can collaborate through:
  - service objects
  - package sharing
- A dynamic registry allows a bundle to find and track service objects
- Framework fully manages this collaboration
  - Dependencies, security

# Component interaction and collaboration



No management bundles

No collaboration

No package management  
(versions!)

No native code

# Functionalities supported by OSGi

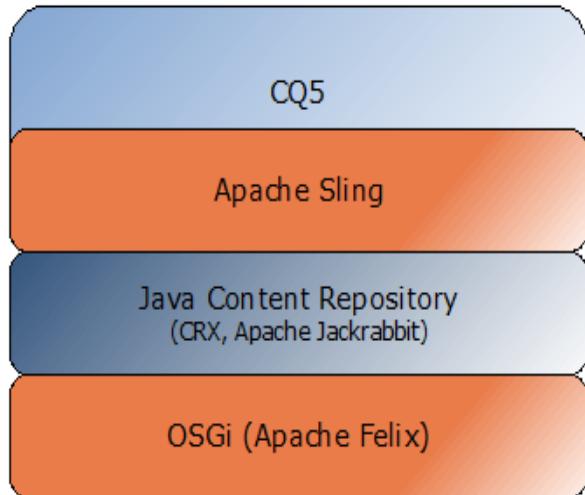
- 1) Reduces the complexity of the system.
- 2) Managing service/component dependencies.
- 3) Makes the components loosely-coupled and easy to manage.
- 4) Increases the performance of the system.

# OSGi increase performance.

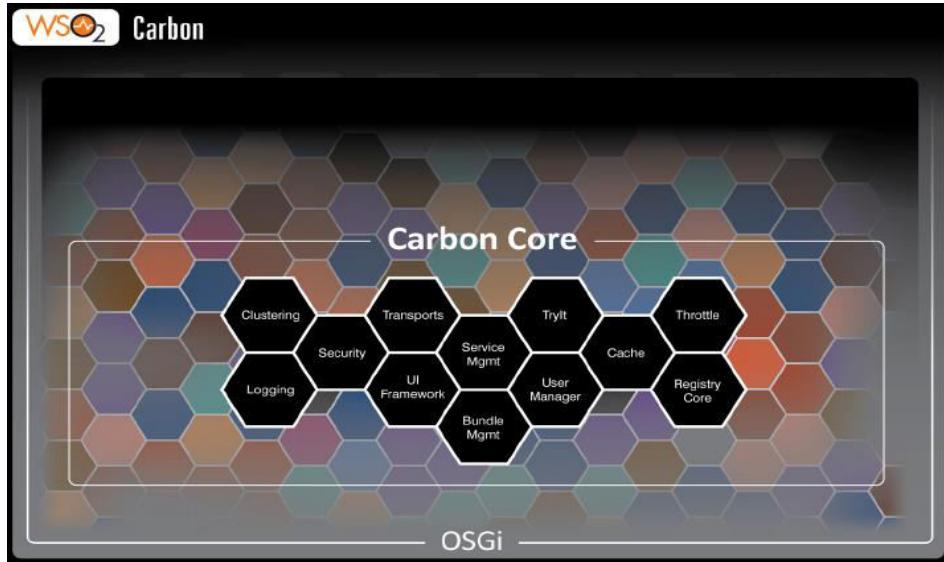
- Consider a scenario where you have a large application which uses a logging framework.
- This **logging framework** can be deployed as an OSGi Bundle, which **can be managed independently**.
- 
- Therefore, it can be started when required by our application and can be stopped when not in use.
- Also the OSGi container makes these **bundles available as services**, which can be subscribed by other parts of application.

# Custom products uses OSGi

Adobe CQ (CMS app)



- CQ5, uses the Apache Felix implementation of OSGi.
- Apache Felix is a open-source project to implement the OSGi R4 Service Platform.
- That includes
  - OSGi framework and standard services.
  - OSGi-related technologies.

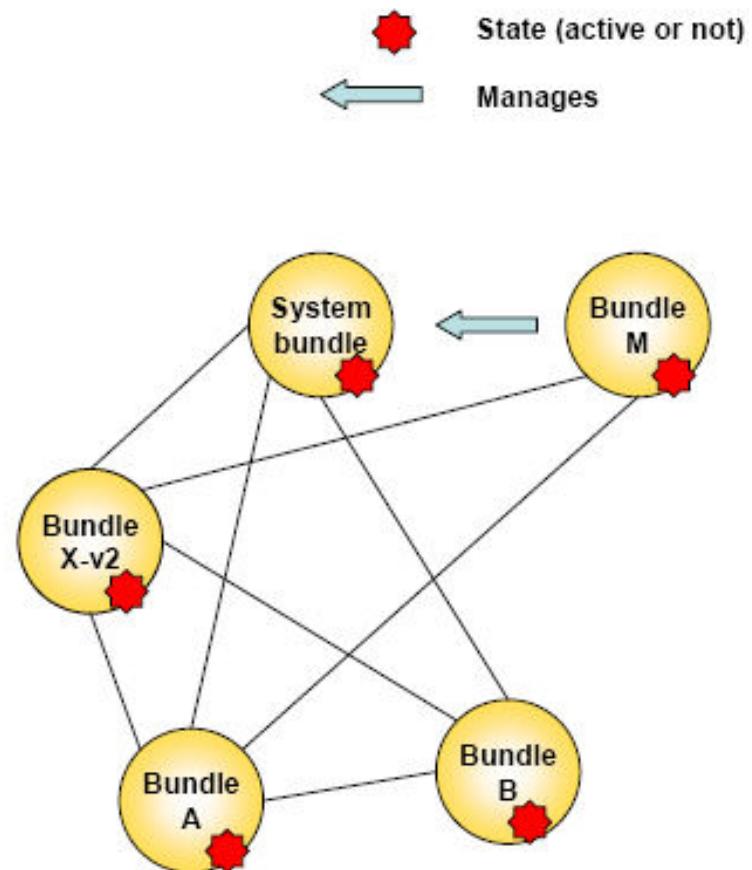


## What is a Carbon Component

- A set of OSGi Bundles.
- Lives in the Carbon Framework. Hence should conform to rules define in the Carbon Framework.
- Develop the Carbon component
  - . Back-end component (BE OSGi bundles)
  - . Front-end component (FE OSGi bundles)
  - . Common bundles, if any

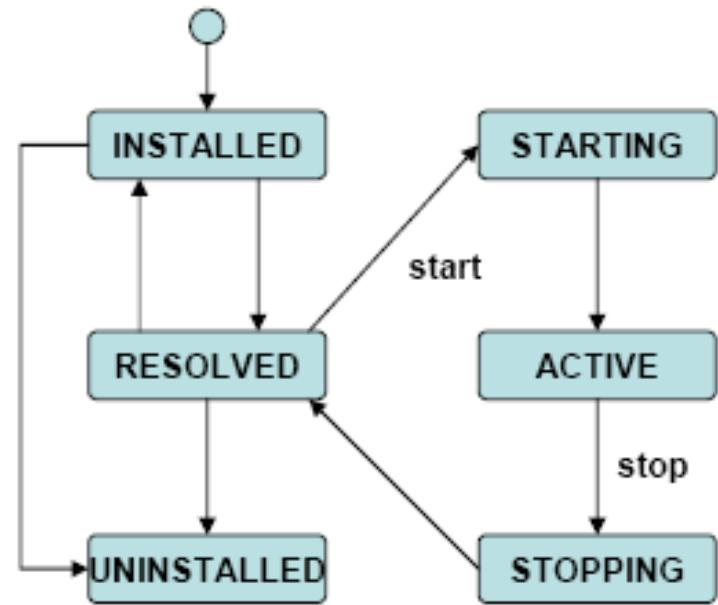
# Layers: OSGi Life Cycle Layer

- System Bundle represents the OSGi Framework
- Provides an API for managing bundles
  - Install
  - Resolve
  - Start
  - Stop
  - Refresh
  - Update
  - Uninstall



# Layers: OSGi Life Cycle Layer

- Bundle is started by the *BundleActivator* class
- Header in the JAR manifest file refer to this class
- *BundleActivator* interface has 2 methods
  - Start: Initialize and return immediate
  - Stop: Cleanup
- The *BundleActivator* gets a *Bundle Context* that provides access to the OSGi Framework functions.
- The Framework provides the Start Level service to control the start/stop of groups of applications.



# OSGI life cycle methods

## org.osgi.framework

### BundleActivator

- start(BundleContext) : void
- stop(BundleContext) : void

## States of bundles

- Installed – finish bundle installation
- Active – start the bundle
- Resolved – stop the bundle

## org.osgi.framework

### BundleContext

- addBundleListener(BundleListener) : void
- addFrameworkListener(FrameworkListener) : void
- addServiceListener(ServiceListener) : void
- addServiceListener(ServiceListener, String) : void
- createFilter(String) : Filter
- getAllServiceReferences(String, String) : ServiceReference
- getBundle() : Bundle
- getBundle(long) : Bundle
- getBundles() : Bundle[]
- getDataFile(String) : File
- getProperty(String) : String
- getService(ServiceReference) : Object
- getServiceReference(String) : ServiceReference
- getServiceReferences(String, String) : ServiceReference
- installBundle(String) : Bundle
- installBundle(String, InputStream) : Bundle
- registerService(String[], Object, Dictionary) : ServiceReference
- registerService(String, Object, Dictionary) : ServiceReference
- removeBundleListener(BundleListener) : void
- removeFrameworkListener(FrameworkListener) : void
- removeServiceListener(ServiceListener) : void
- ungetService(ServiceReference) : boolean

```
51 public interface BundleActivator {
```

Called when this bundle is started so the Framework can perform the bundle-specific activities necessary to start this bundle. This method can be used to register services or to allocate any resources that this bundle needs.

This method must complete and return to its caller in a timely manner.

**Parameters:**

context The execution context of the bundle being started.

**Throws:**

[java.lang.Exception](#) If this method throws an exception, this bundle is marked as stopped and the Framework will remove this bundle's listeners, unregister all services registered by this bundle, and release all services used by this bundle.

```
66
```

```
67 public void start(BundleContext context) throws Exception;
```

Called when this bundle is stopped so the Framework can perform the bundle-specific activities necessary to stop the bundle. In general, this method should undo the work that the `BundleActivator.start` method started. There should be no active threads that were started by this bundle when this bundle returns. A stopped bundle must not call any Framework objects.

This method must complete and return to its caller in a timely manner.

**Parameters:**

context The execution context of the bundle being stopped.

**Throws:**

[java.lang.Exception](#) If this method throws an exception, the bundle is still marked as stopped, and the Framework will remove the bundle's listeners, unregister all services registered by the bundle, and release all services used by the bundle.

```
85
```

```
86 public void stop(BundleContext context) throws Exception;
```

```
87 }
```

Registers the specified service object with the specified properties under the specified class name with the Framework.

This method is otherwise identical to [registerService\(java.lang.String\[\], java.lang.Object, java.util.Dictionary\)](#) and is provided as a convenience when service will only be registered under a single class name. Note that even in this case the value of the service's [Constants.OBJECTCLASS](#) property will be an array of string, rather than just a single string.

#### Parameters:

`clazz` The class name under which the service can be located.

`service` The service object or a `ServiceFactory` object.

`properties` The [properties for this service](#).

#### Returns:



A `ServiceRegistration` object for use by the bundle registering the service to update the service's properties or to unregister the service.

#### Throws:

[java.lang.IllegalStateException](#) If this `BundleContext` is no longer valid.

#### See also:

[registerService\(java.lang.String\[\], java.lang.Object, java.util.Dictionary\)](#)

461

462     public [ServiceRegistration](#) [registerService\(String clazz, Object service, Dictionary properties\);](#)

# Manipulating Services

- The *Bundle Context* provides the methods to manipulate the service registry
- Services registrations are handled by *Service Registration* objects
  - They can be used to unregister a service or modify its properties
- *Service Reference* objects give access to the service as well as to the service's properties
- Access to service objects is through the *getService* method. These services should be returned with the *ungetService* method.

```
ServiceRegistration registerService(  
    String clazz,  
    Object srvc,  
    Dictionary props)  
  
ServiceReference[]  
getServiceReferences(  
    String clazz,  
    String filter)  
  
Object getService(  
    ServiceReference reference)  
  
boolean ungetService(  
    ServiceReference reference);
```

# References

- <http://www.osgi.org/Main/HomePage>
- <http://www.osgi.org/Technology/WhatIsOSGi>
- <http://en.wikipedia.org/wiki/OSGi>
- <http://grepcode.com/file/repository.grepcode.com/java/eclipse.org/3.5/org.eclipse/osgi/3.5.0/org/osgi/framework/BundleContext.java#BundleContext.getBundle%28long%29>
- [\*http://felix.apache.org/site/apache-felix-framework-usage-documentation.html\*](http://felix.apache.org/site/apache-felix-framework-usage-documentation.html)

# Stereo Vision, Stereoscopy & Virtual Reality

By

**Aruna Ishara Gamage**

Lecturer, Software Engineering/Multimedia,  
Department of Computer Science & Software Engineering,  
Faculty of Computing, SLIIT

[ishara.g@sliit.lk](mailto:ishara.g@sliit.lk)



# Virtual Reality

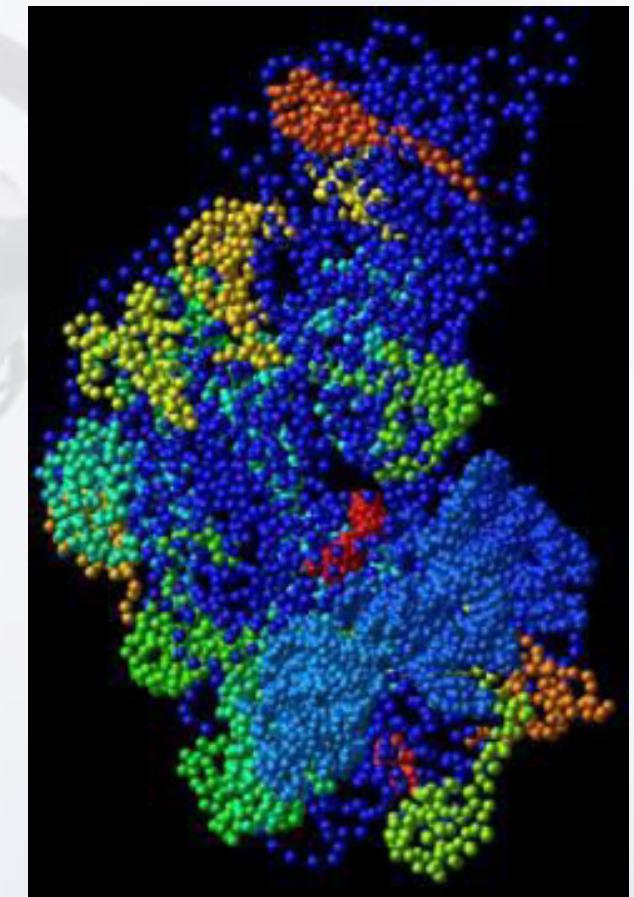
What makes the sense of realism(යලාර්ථමාදය)?

**“Feeling of the existence of a non-existing object or environment”**



# Stereo Vision

- A large part of our brain is devoted to understanding visual cues.
- Depth information can help us to understand spatial relationships in a complex data set



# Binocular Vision

- Our two eyes produce a single image in the brain:  
a “Cyclopean image”
- Creatures with binocular vision generally have  
forward-facing eyes that move together



# Binocular Vision

- Animals that tend to get chased find it useful to have a **panoramic view of the world**(හංා යැමට නැඹුරුවන සතුන් ලේකය පිළිබඳ පරිදුරුණීය දාශ්ච්වලයක් තිබේම ජරෙයෝගනවත් වේ)
  - A rabbits have almost 360 field of vision
- Animals that do the chasing need to be able to judge distance to their prey accurately(හංා යාමේ යොදෙන සතුන්ට තම ගොදුරට ඇති දුර නිවැරදිව විනිශ්චය කිරීමට හැකියාව තිබිය යුතුය)
- Predators tend to move fast, if they are inaccurate in distance estimates they will starve or injure themselves
- Tree apes need to be able to judge the distance to the next branch accurately

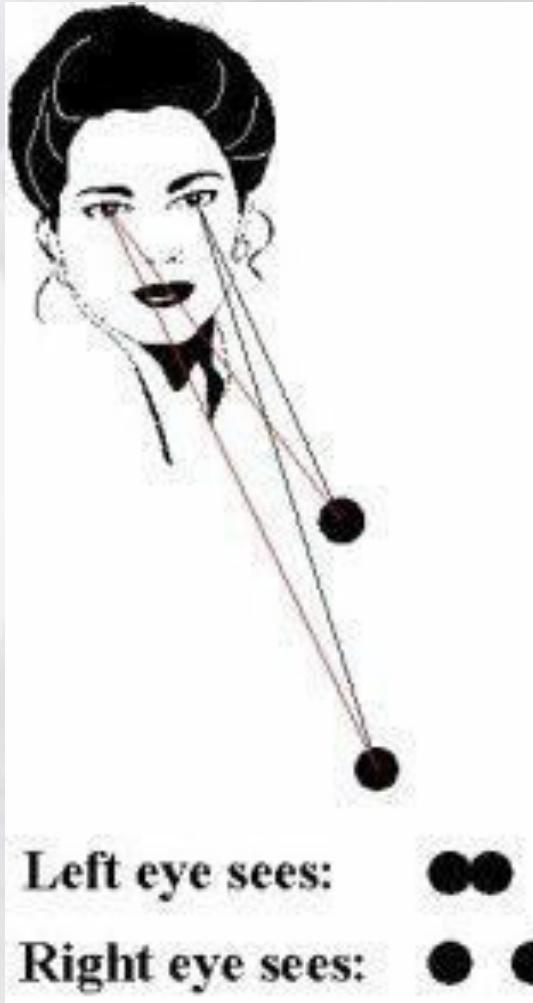


# IPD: (*Human*) Inter-Pupillary Distance

- Need separation between our eyes to see in stereo. (ස්ටීරියෝඩ් වලින් බැලීමට අපට අපගේ ඇස් අතර වෙන්වීමක් අවශ්‍යයි.)
- Different sources provide different values for this number.
- **Mean adult IPD** is around 63 mm.
- The **vast majority** of adults have IPDs in the range 50-75 mm.
- The wider range of 45-80 mm is likely to include (almost) all adults
- Minimum IPD for children (down to five years old) is around 40 mm



# Binocular Vision



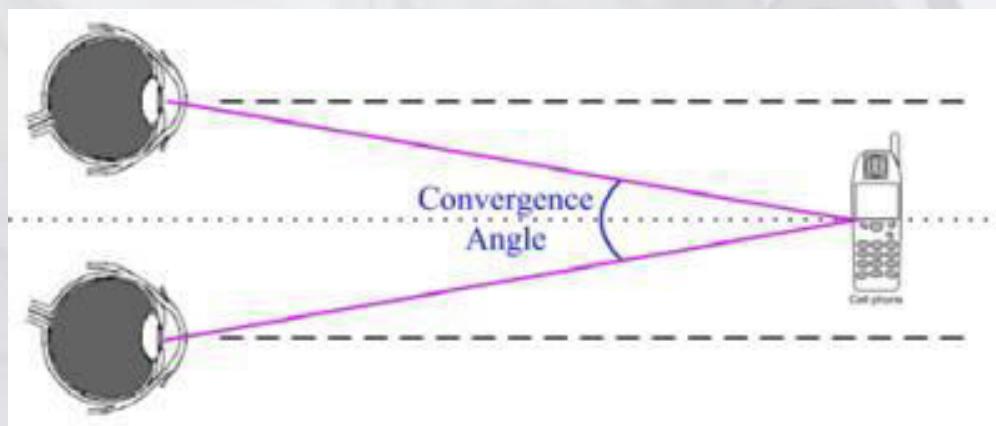
- Our eyes are separated by about 6.5 cm so our retinas each get a slightly different view of the world.
- The right actually sees more distance between the objects.  
*(as well as slightly different parts of the surfaces)*



# Binocular Vision

## Components of Stereo Vision:

- **CONVERGENCE** of the eyes (achieved by the eye muscles)
- **ACCOMODATION** (focus) of the lens to provide sharp images on the retinas



# Binocular Vision

The following table shows convergence angles for different distances given for 6.3 cm IPD.

Distance	Convergence Angle (deg)
10 cm	35.0
30 cm	12.0
60 cm	6.0
100 cm	3.6
200 cm	1.8
500 cm	0.72
1000 cm	0.36

**Convergence Angle is Inversely Proportional to Stereo Effect.**



# Seeing in 3D

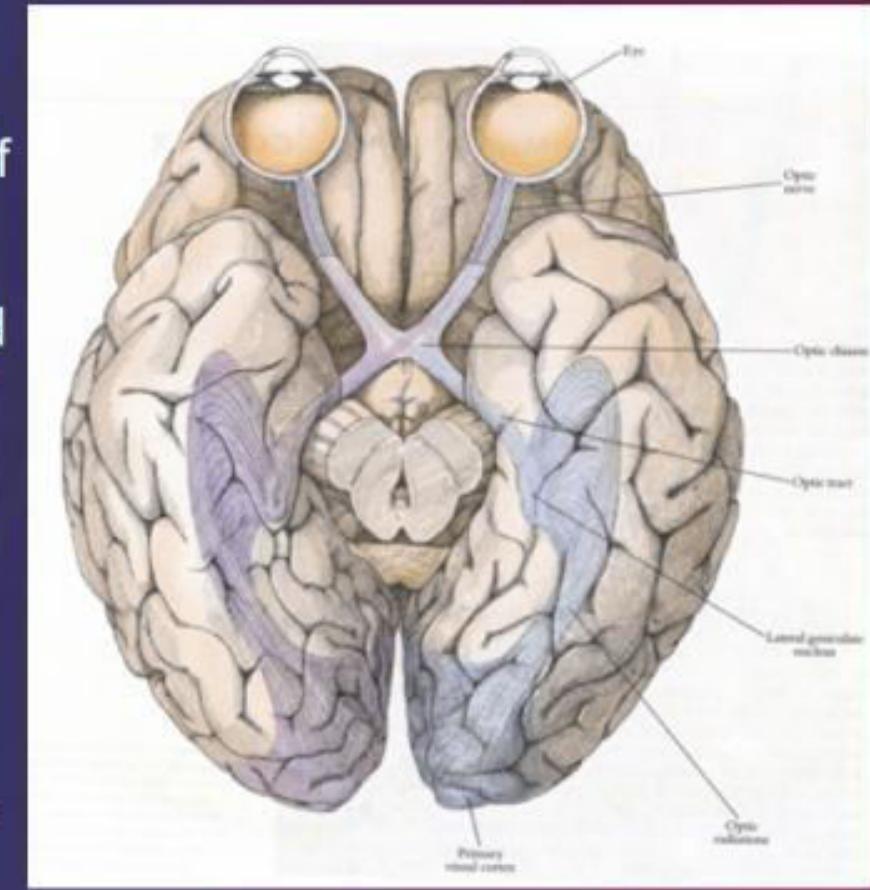
- If retinal(දූජ්ට්‍රි විතානය) images are very different the eyes try to adjust to make them more similar.
- Once the brain has fused the images into one object, the small differences on the retina are interpreted as the **3rd Dimension**.
- Our brain interprets the two views as a scene with depth and does a great job of judging distances from us, up to about 20 feet.  
**(diminishes but works up to ~200 m)** (අපගේ මොලය මෙම දුර්ගන දෙක ගැහුරින් දුර්ගනයක් ලෙස අර්ථකථනය කරන අතර අඩු 20 ක් පමණ දුරින් අපෙන් දුර විනිශ්චය කිරීමේ විශාල කාර්යයක් කරයි. (අඩු නමුත් 200 m දක්වා ක්රියා කරයි)



# Seeing in 3D

## Binocular Vision

- image from left eye is sent to the right half of the brain
- the brain deciphers retinal differences and interprets them as 3D information
- the interpretation can take time
- with practice the computations can go faster (they get "hardwired")
- image from "Eye Brain and Vision" by David H. Hubel



# Non-Stereo Depth Cues

## Monocular cues

**Occlusion** - near objects block the view of distant objects.

**Apparent size** - if two objects are actually the same size, but one appears smaller, then the small one is farther away than the larger.

**Motion parallax** - near objects appear move faster than distant objects.

**Perspective** - parallel lines converge in the distance(සමාන්තර රේඛා කුරින් අභිසාරී වේ).



# Non-Stereo Depth Cues

## Monocular cues

**Texture** - becomes finer(සිහින්) with the distance.

**Colour change** - colours become more blue with the distance.

**Haze** - objects become fuzzy with the distance.

**Accommodation** - our brain knows how hard our eyes are working to provide continues focus.



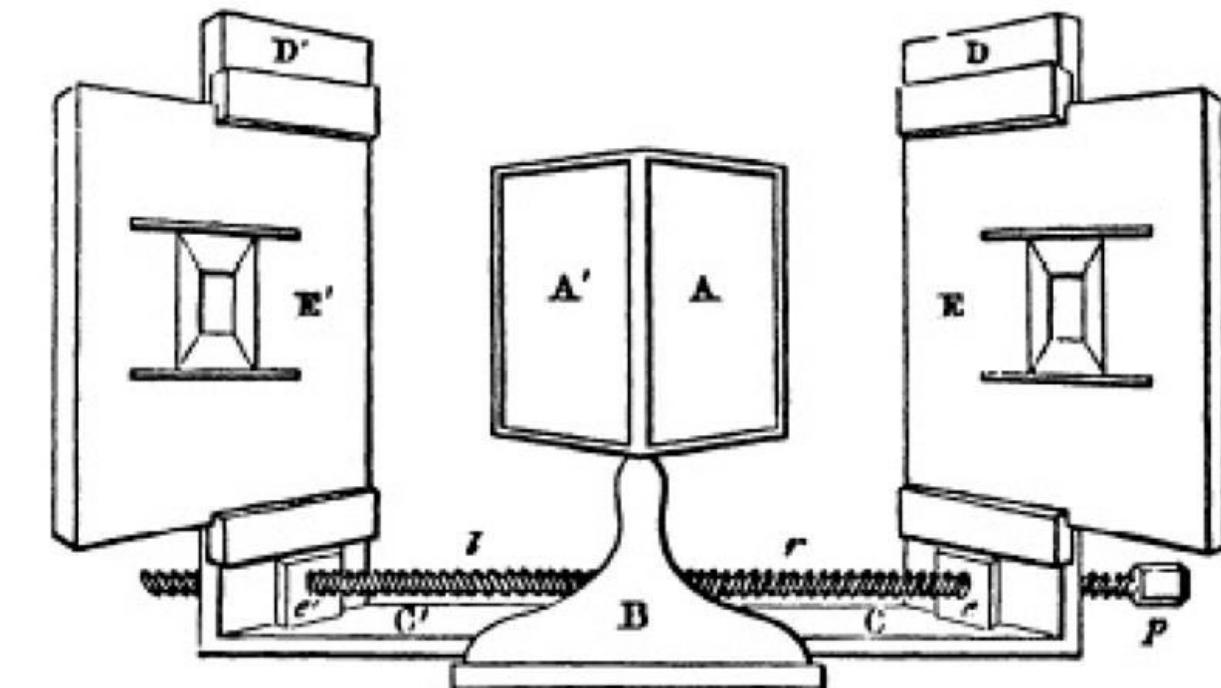
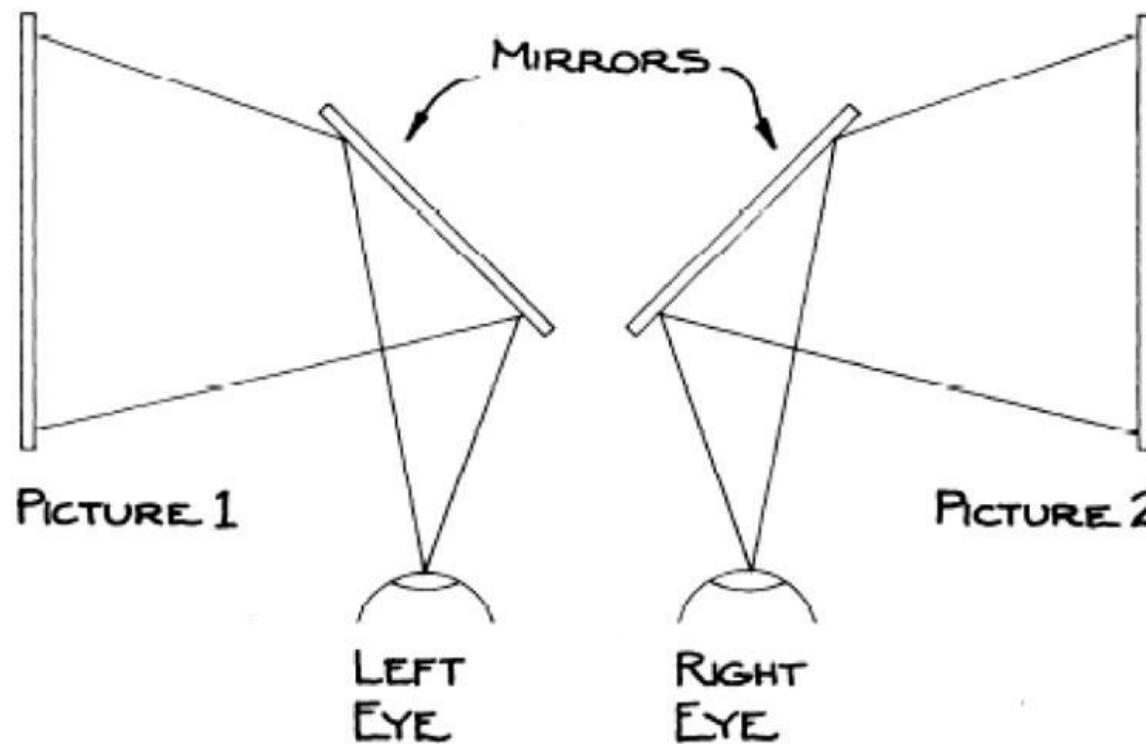
# Stereoscopy

- A technique to create the illusion of depth in a photograph, movie, or other two-dimensional image. (ජායාරූපයක, වින්රපටයක හෝ වෙනත් දේශීමාන රුපවල ගැඹුර පිළිබඳ මින්යාව නිර්මාණය කිරීමේ තාක්ෂණයකි.)
- Need to present a slightly(තරමක්) different image to each eye.
- Our brain interprets the two views as a scene with depth and does a great job of judging distances from us, up to about 20 feet. (diminishes but works up to ~200 m).
- **Stereoscopic Viewer invented by Sir Charles Wheatstone in 1838.**



# Stereoscopy

Stereoscopic Viewer by Sir Charles Wheatstone (1838)  
- Note: No PHOTOGRAPHY yet!



# Stereoscopy

**Stereoscopic Viewer - Upgraded (1905 ~ 2000's)**



# Stereoscopy

- To create depth perception in the brain provide to the eyes of the viewer with two different images.
- Two perspectives of the same object.
- Minor deviation similar to the perspectives that both eyes naturally receive in binocular vision.



# Stereoscopy: a simple test?

- Hold a pencil in front of your nose and look into the distance.
- Do you see 2 pencils?
- Some people don't see 2 pencils because their brains suppress information from one eye.
- The suppressed eye can shift from left to right.
- There might be permanent suppression of one eye.
- people with one eye suppressed won't be able to see in 3D.



# Cross-eyed Stereo

- Look at the following stereo image pair by slightly crossing your eyes.
- Left eye sees the right image and right eye sees the left image





# Stereoscopic 3D Video Content



- A common term: **3D movies!**
- **Need special eye ware (3D glasses)**
  - Active Stereo
    - (battery Powered) Chemical Shutter Glasses
  - Passive Stereo
    - Anaglyph 3D
    - Polarized 3D
      - Linear Polarized
      - Circular Polarized
- **Need special projection systems.**
  - Need to be able to project frames alternatively, representing Left-eye and Right-eye image components.



# 3D Movies: Feeling of Realism

Feeling of realism is generated by:

- Creating a **visual illusion of depth** by cheating our brain.
- Calibrated **surround sound effects** representing the visual environment.

Making us believe that we are one with what we see...  
or... what we see actually exist in front of us!

*Which is something that  
really isn't there!*



# Virtual Reality

Virtual Reality (VR) is the illusion of a

- Three-Dimensional
- Interactive,

Computer-Generated Reality where

- Sight
- Sound, and sometimes even
- Touch

are simulated to create

- Pictures
- Sounds,

and objects that actually seen and feel real.



# VR Headsets

## Cost Effective (Budget) Solutions

- Google Cardboard
- VR-Box



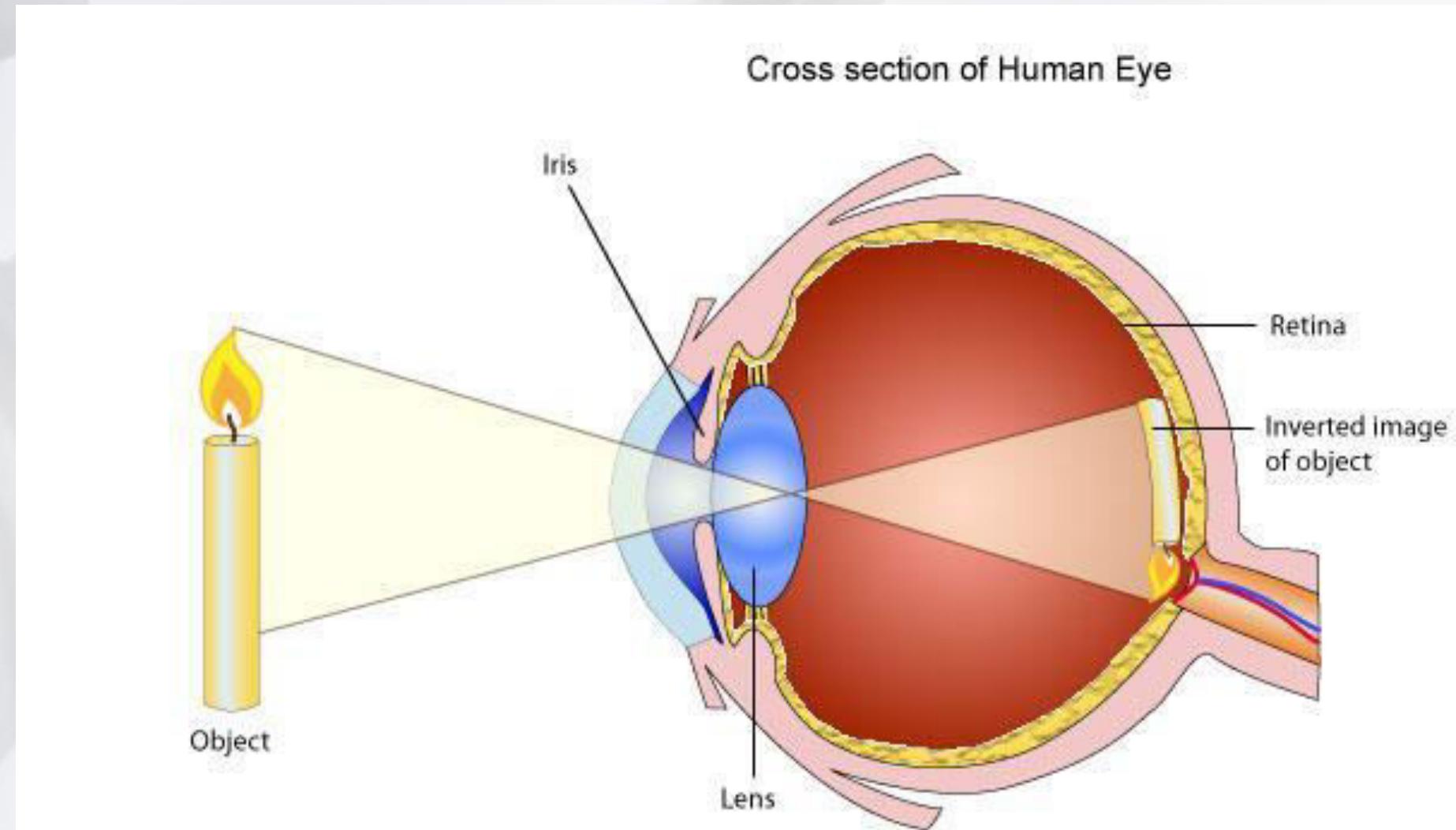
## High End Solutions

- Samsung Gear VR
- Oculus Rift
- HTC Vive

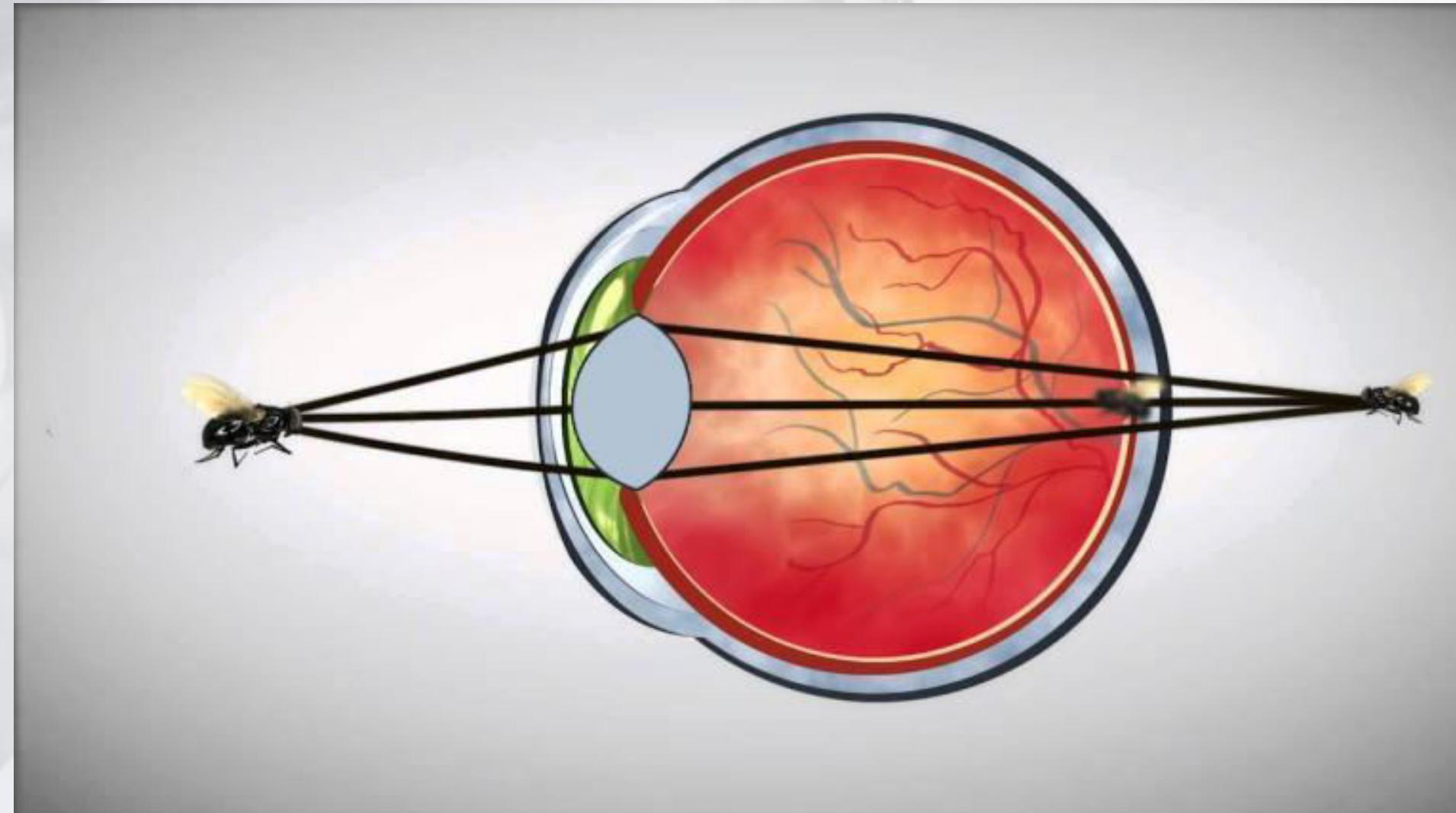


*Remember, all products use the same basic, simple elements to bring in the Virtual Reality. Only the technological ways of presenting the Visual content is different.*

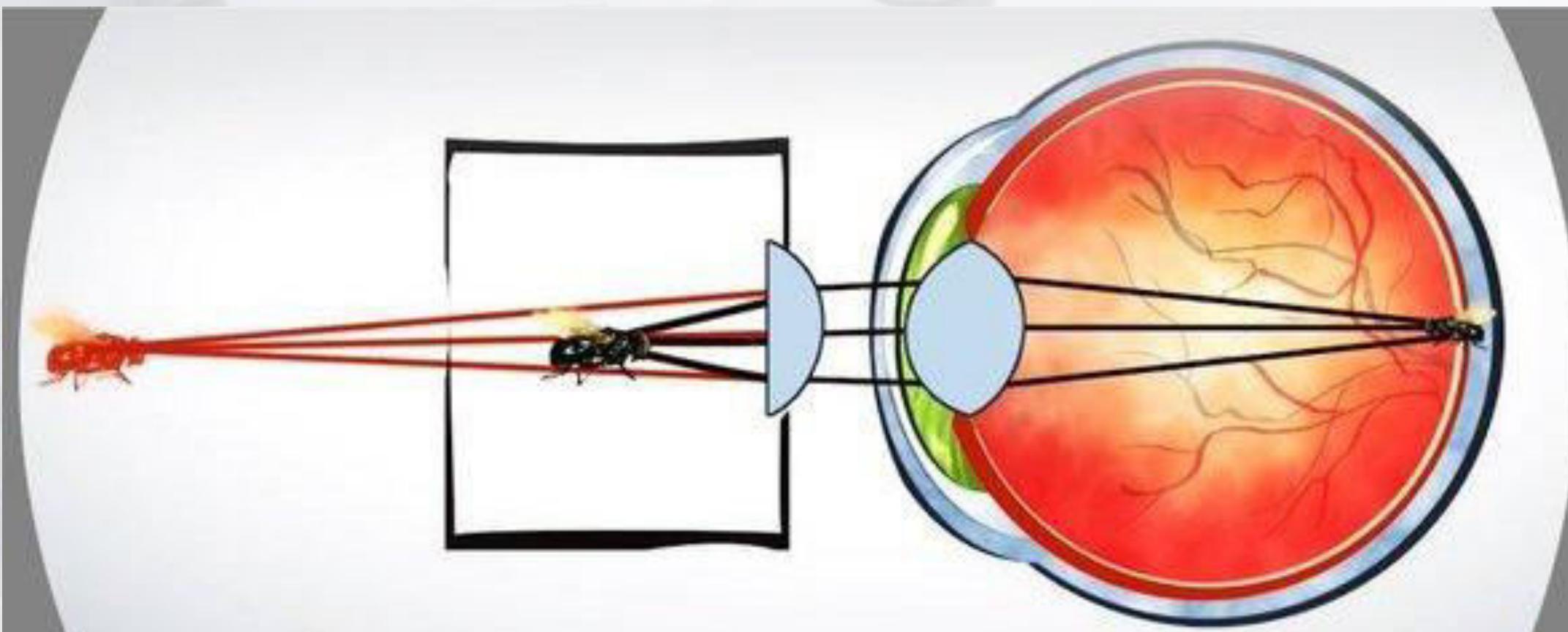
# VR Headsets: How they work?



# VR Headsets: How they work?



# VR Headsets: How they work?



# VR Headsets: How they work?

## Functionality of the VR Special lenses

Correct the angle of light that is coming into eye and make a forced perspective of the object being farther away than it really is.



# VR Box - Component Breakdown

**43 Sets of Precision Parts**

Original 43 sets of precision components, each set of accessories after a few procedures debugging combinations to ensure accuracy, 59 assembly process, harsh to the extreme technical requirements, so that every fan experiences the most perfect products.



**43**  
Precision parts

**59**  
military technology



# Virtual Reality: Major Concepts

- VR must allow the user to **view the environment** from any point and at **any angle**.  
*(360 degree view)*
- VR must allow the user to **interact with objects** in the environment.



# Again - Virtual Reality - Concept Baseline

What makes the sense of realism?

**“Feeling of the existence of a  
non-existing object or environment”**



# Any Questions?

**Aruna Ishara Gamage**

Lecturer, Software Engineering/Multimedia,  
Department of Computer Science & Software Engineering,  
Faculty of Computing,

SLIIT

[ishara.g@sliit.lk](mailto:ishara.g@sliit.lk)

071 44 38 449

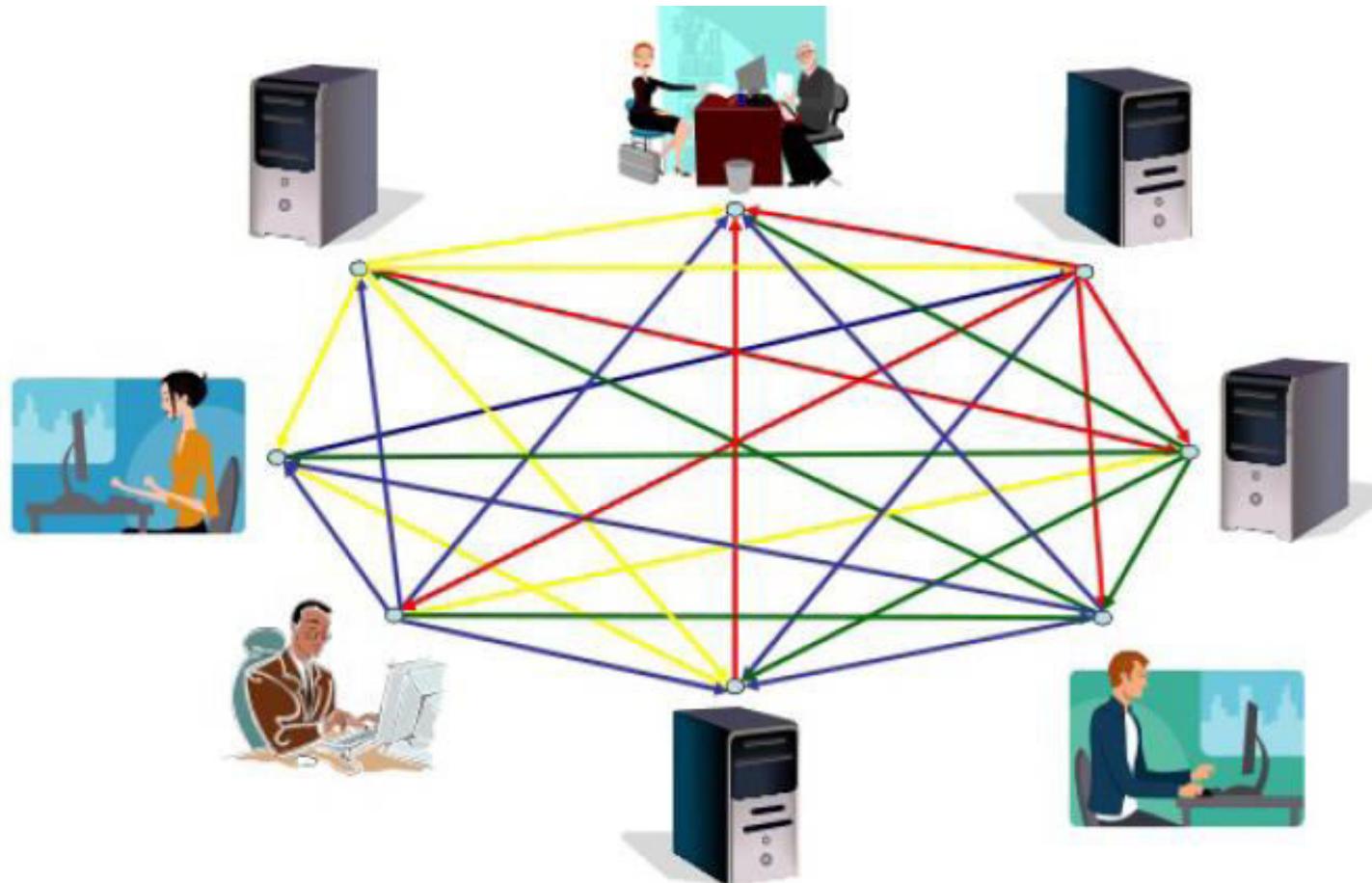




# WSO2 Enterprise Service Bus

**Modern Topics in Information Technology  
4<sup>th</sup> Year – Semester 1  
By Udara Samaratunge**

# Service Oriented Architecture - SOA



# SOA Evolution

- Main Frames – Used Tapes to transfer files
- Later lower level socket based communication was used
- Then came, Network File System (NFS) and File Transfer Protocols (FTP)
- Remote Procedure Calls (RPCs) got matured along with the improved of server hardware
- CORBA came in but the advent of Java resulted the demise of CORBA
- DCOM came in but its proprietary nature resulted its demise
- SOAP relies on XML as the payload, which has got much higher degree of interoperability between programming languages.

# Problems related to RPC

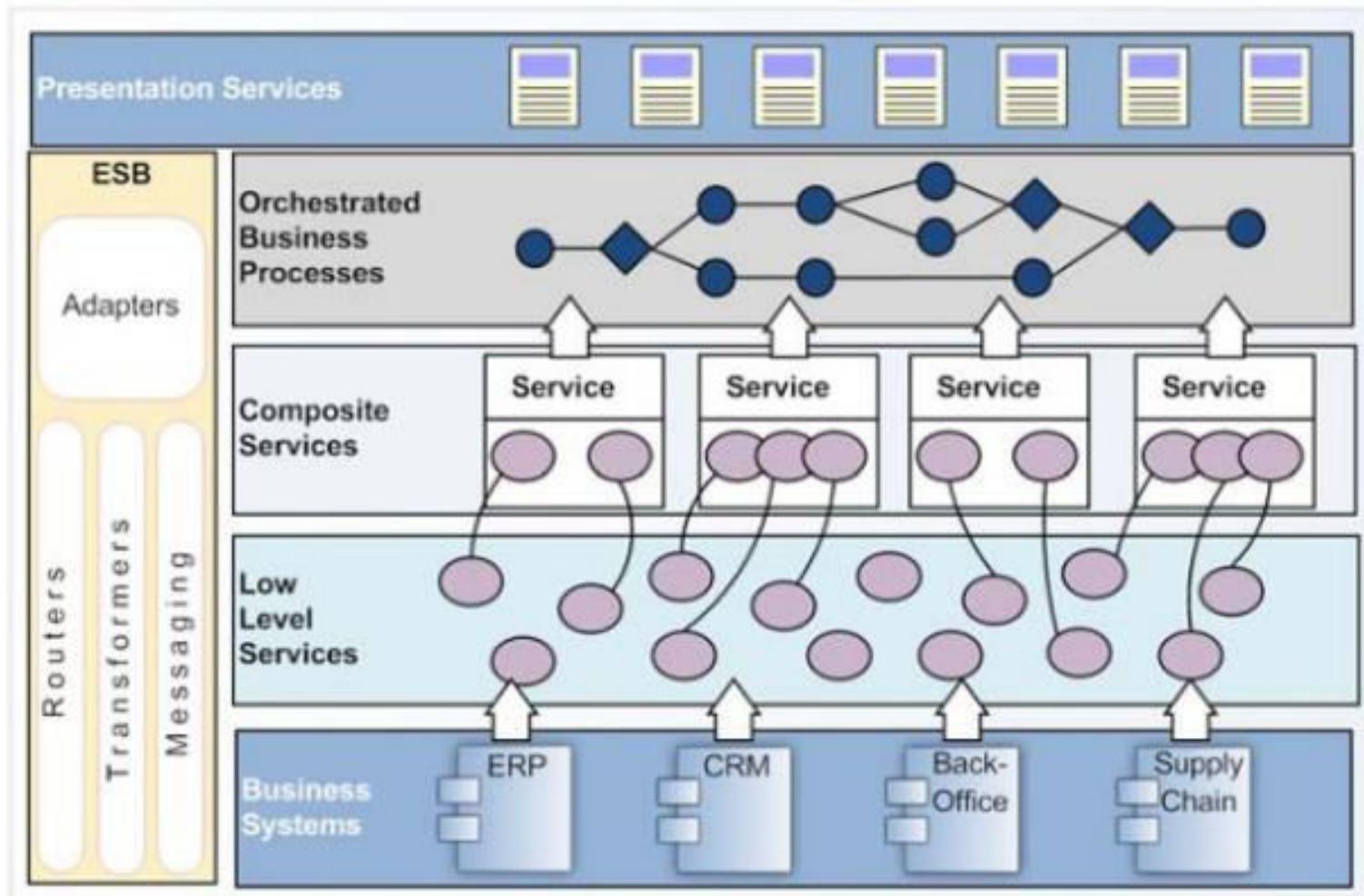
- **Tight coupling** between local and remote systems requires significant bandwidth demands
- **Interoperability issues** – Mainly due to incompatible data types in different languages

*SOAP's message style can overcome above issues. SOA was developed keeping "loose coupling" and the "interoperability" in mind*

# Why SOA?

- CORBA, EJB, DCOM introduced a **highly coupled RPC**. [*Unlike SOA*]
- EJB and DCOM were **tied to specific platforms** and not at all inter-operable. [*Unlike SOA*]
- EJB, DCOM and CORBA were more relied on **commercial oriented products**. [*Unlike SOA*]
- **SOA** can be implemented using a **completed “Open Source” Stack**.
- **SOA** relies on **XML** as the *underlying data representation*, unlike the others, which used *proprietary binary-based objects*
- Unlike CORBA, EJB or DCOM, **SOA** is **more than a RPC technology**, It is a,
  - Governance
  - SLAs (Service Level Agreements)
  - Meta-data Definitions/ Registries

# The SOA Environment

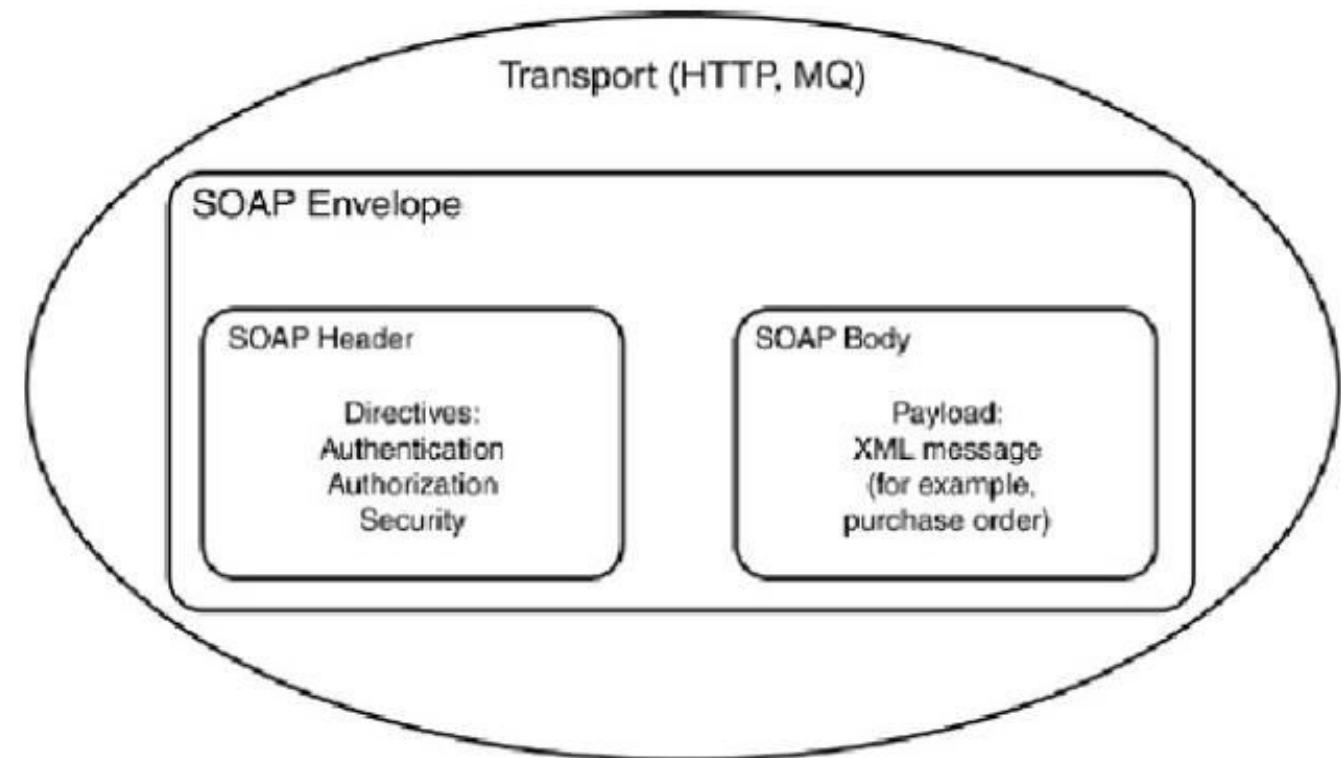


# The characteristics of SOA

- The service **Interface/Contract**
- A service must have well defined interface contract.
- This contract should identify,
  - The “**operations**” that are available **through the service**
  - “**Data Requirements**” for any “**Exchanged Information**”
- **WSDL (Web Service Description Language)** is a good example for a **service contract**
- Web services – Related technologies = **XML / XML Schema**
- **Web service** use **SOAP** as **communication Protocol**. [**Simple Object Access Protocol**]
- **SOAP** runs on **HTTP protocol** & uses **default port 80**. **Message** structured as **XML**.

# XML Messaging SOAP

- The SOAP envelop is just a container to hold XML data.
- **SOAP envelope**
  - **SOAP Header** – Contains information related to the message and its security
  - **SOAP Body** – Contains the **message payload**
- Requests are encoded in XML and sent via HTTP POST
- Most **firewalls allow HTTP** traffic. This allows **XML-RPC** or **SOAP** messages to be used as **HTTP messages**.



# WSDL

- Three Sections

- **What Section** – Input and Output messages (<wsdl:types>, <wsdl:message>)
- **How Section** – How messages should be packaged (bind) to different protocols in the SOAP envelop and how to transfer it (<wsdl:binding>)
- **Where Section** – The endpoint details (<wsdl:service>)

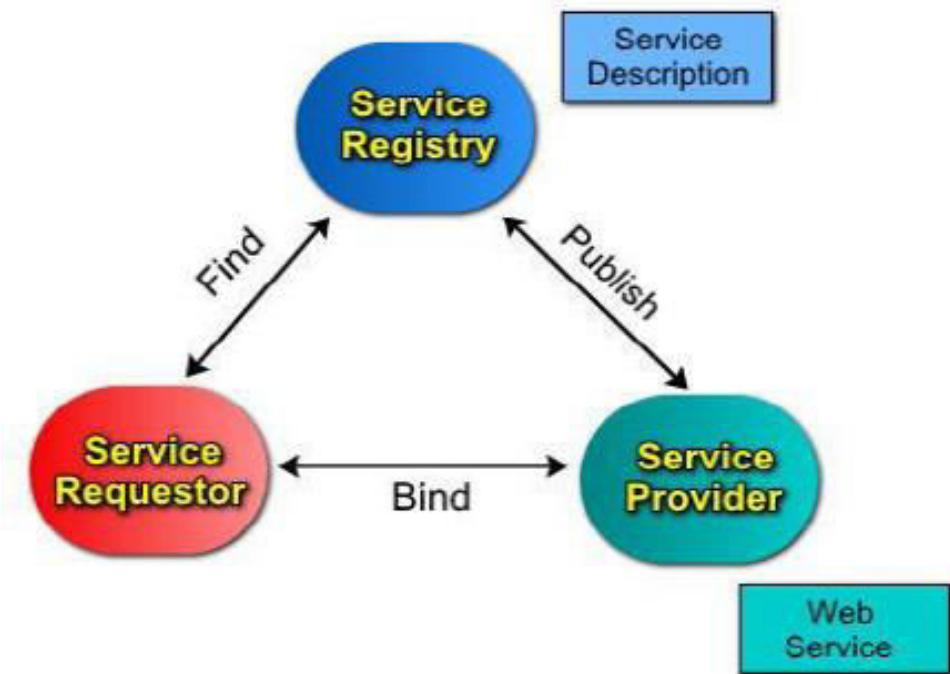
```
-<wsdl:definitions targetNamespace="http://ead">
  <wsdl:documentation> Please Type your service description here </wsdl:documentation>
+<wsdl:types></wsdl:types>
+<wsdl:message name="farenhit2celciusRequest"></wsdl:message>
+<wsdl:message name="farenhit2celciusResponse"></wsdl:message>
+<wsdl:message name="celcius2farenhitRequest"></wsdl:message>
+<wsdl:message name="celcius2farenhitResponse"></wsdl:message>
-<wsdl:portType name="TempWSPortType">
  -<wsdl:operation name="farenhit2celcius">
    <wsdl:input message="ns:farenhit2celciusRequest" wsaw:Action="urn:farenhit2celcius"/>
    <wsdl:output message="ns:farenhit2celciusResponse" wsaw:Action="urn:farenhit2celciusResponse"/>
  </wsdl:operation>
  -<wsdl:operation name="celcius2farenhit">
    <wsdl:input message="ns:celcius2farenhitRequest" wsaw:Action="urn:celcius2farenhit"/>
    <wsdl:output message="ns:celcius2farenhitResponse" wsaw:Action="urn:celcius2farenhitResponse"/>
  </wsdl:operation>
</wsdl:portType>
+<wsdl:binding name="TempWSSoap11Binding" type="ns:TempWSPortType"></wsdl:binding>
+<wsdl:binding name="TempWSSoap12Binding" type="ns:TempWSPortType"></wsdl:binding>
+<wsdl:binding name="TempWSHttpBinding" type="ns:TempWSPortType"></wsdl:binding>
-<wsdl:service name="TempWS">
  -<wsdl:port name="TempWSHttpSoap11Endpoint" binding="ns:TempWSSoap11Binding">
    <soap:address location="http://192.168.2.2:8080/axis2/services/TempWS.TempWSHttpSoap11Endpoint"/>
  </wsdl:port>
  -<wsdl:port name="TempWSHttpSoap12Endpoint" binding="ns:TempWSSoap12Binding">
    <soap12:address location="http://192.168.2.2:8080/axis2/services/TempWS.TempWSHttpSoap12Endpoint"/>
  </wsdl:port>
  -<wsdl:port name="TempWSHttpEndpoint" binding="ns:TempWSHttpBinding">
    <http:address location="http://192.168.2.2:8080/axis2/services/TempWS.TempWSHttpEndpoint"/>
  </wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

The diagram illustrates the three sections of WSDL by grouping the code with curly braces:

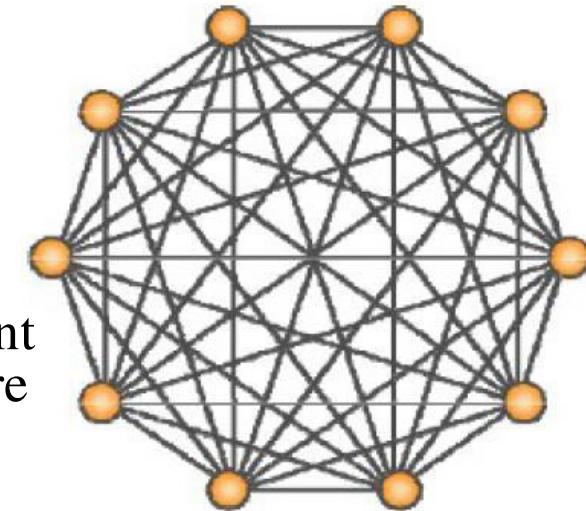
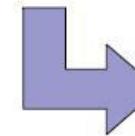
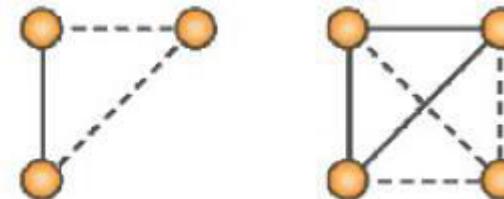
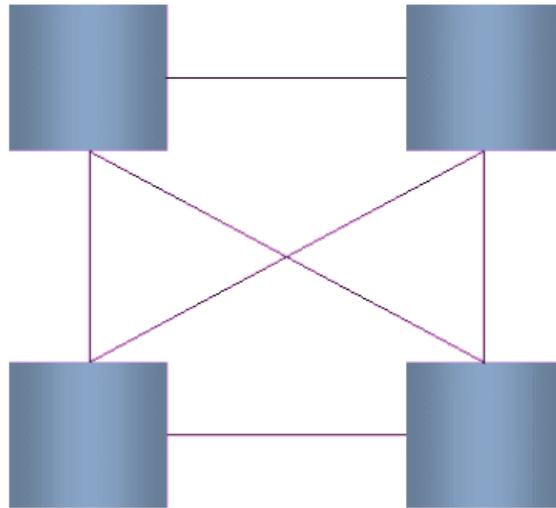
- What:** Groups the first four lines of the WSDL code: <wsdl:definitions>, <wsdl:documentation>, <wsdl:types>, and <wsdl:message> elements.
- How:** Groups the <wsdl:portType> element and all three <wsdl:binding> elements below it.
- Where:** Groups the <wsdl:service> element and its three <wsdl:port> elements.

# Web Services Model

- UDDI is an XMLbased standard for describing, publishing, and finding Web services
- UDDI = (Universal Description, Discovery and Integration)
- UDDI uses WSDL to describe interfaces to web services
- **Approaches of writing web service**
  - **Bottom-Up/Code First Approach** [Implement web service method first]
  - **Top-Down/Contract First Approach** [Write WSDL first then generate Stub classes & Skeleton classes]
- If you generate web services for different plat-forms (Java or .NET) which method is most suitable?
- **Service Provider:** The provider of web service
- **Service Requester:** The web service consumer
- **Service Registry:** The central directory of services



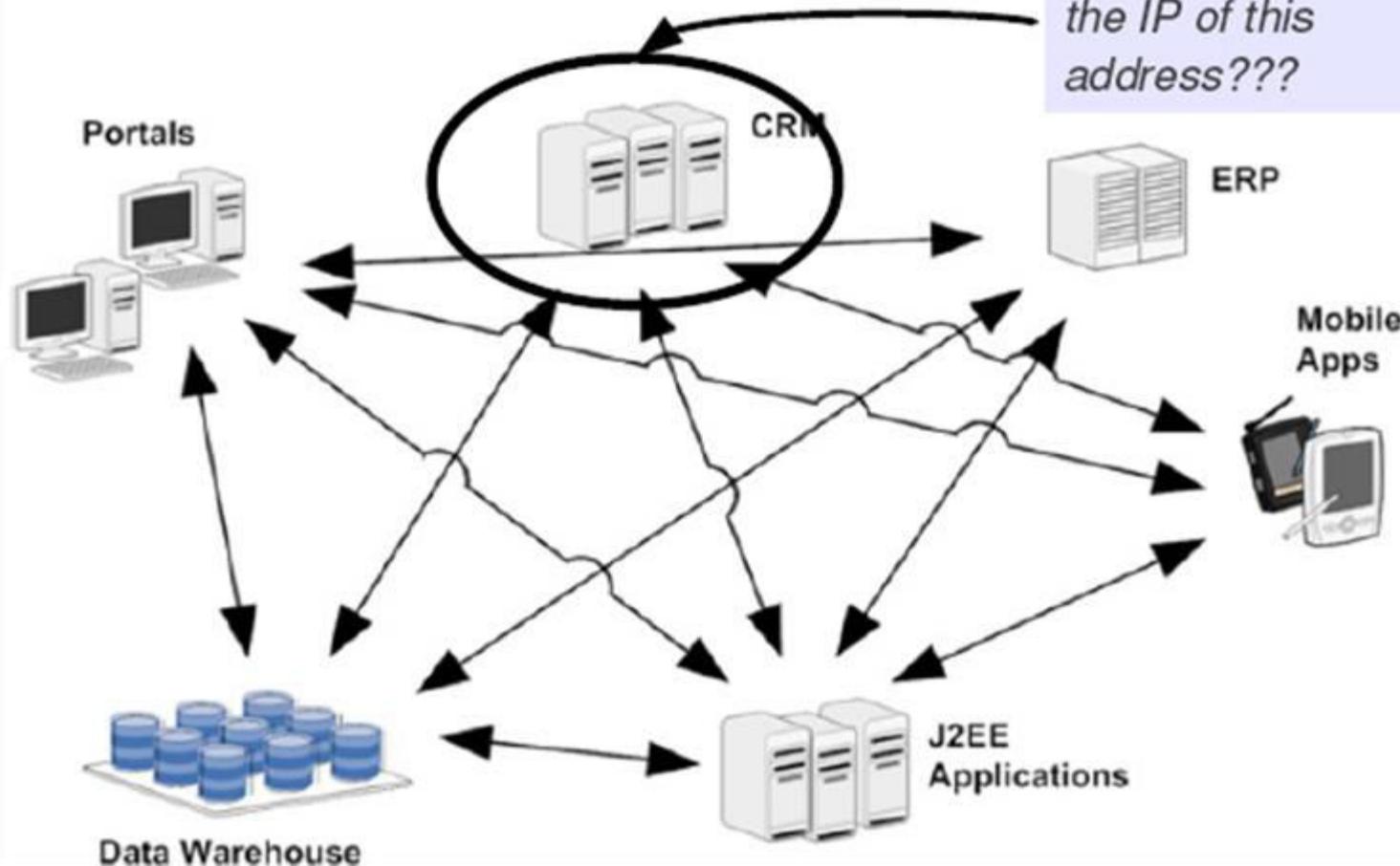
# Point-to-Point Integration



- Specifically, linking every component to every other component will require **N(N-1)/2 physical connections**
- N = Total Number of Components in the Network
- E.g: If there are 10 components in the network,
- Total number of physical connections =  $10(10-1)/2 = 45$

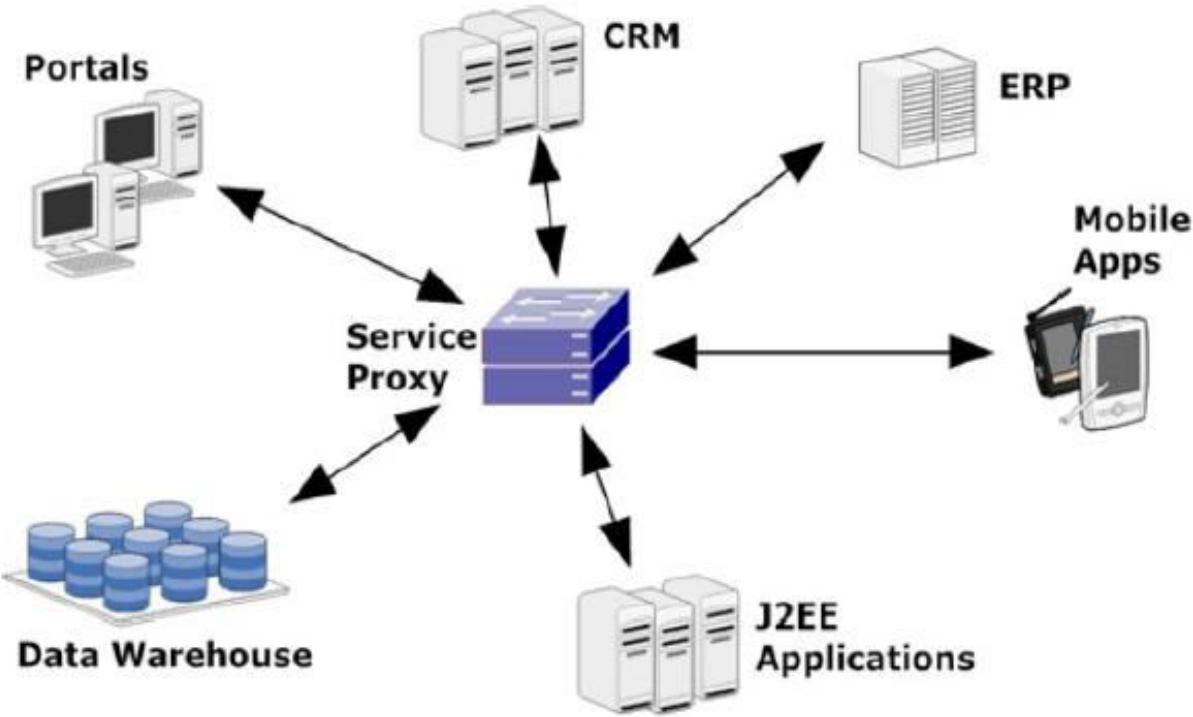
# Why ESB?

## The Service Transparency



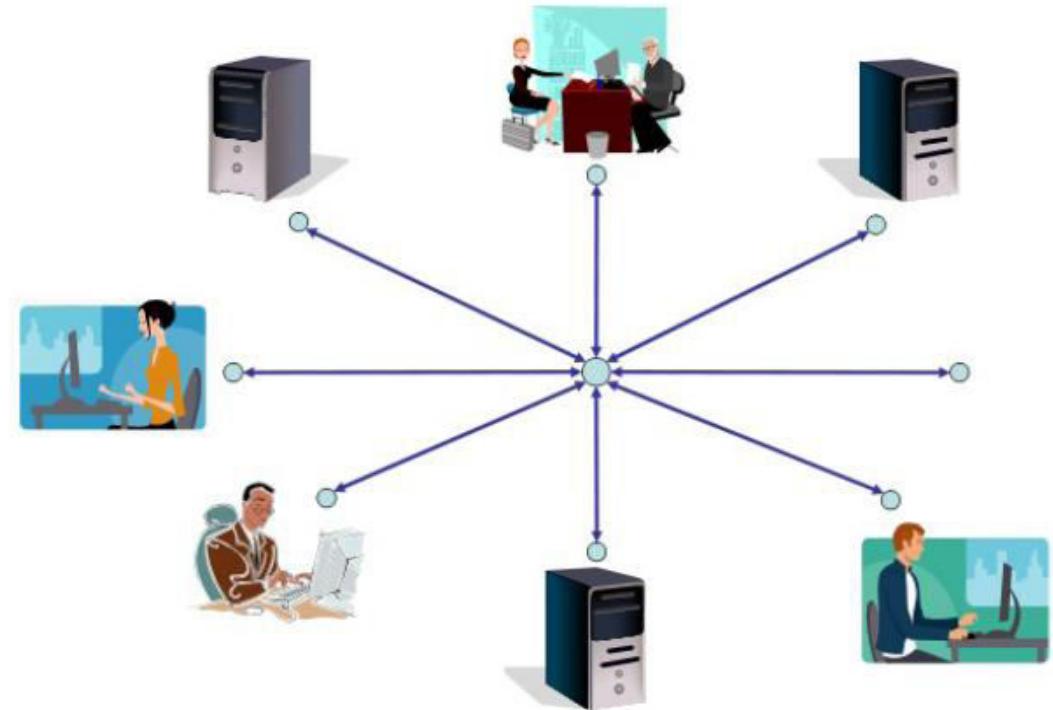
# Solution for the Issue

## The Service Transparency



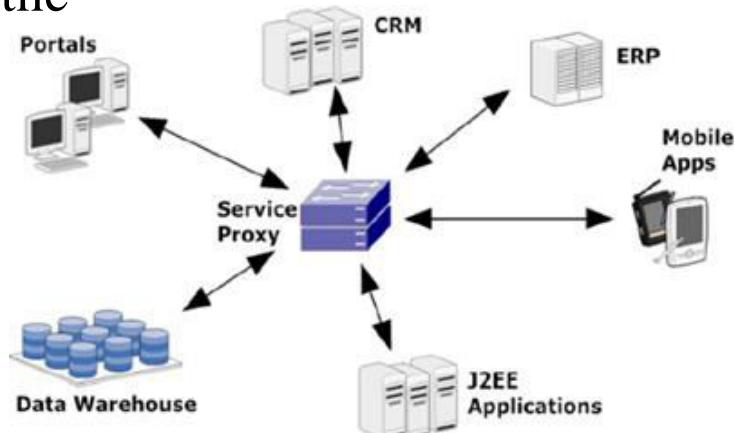
## Hub-Spoke Model

A more centralized approach to the previous point-to-point approach



# Solution from ESB

- An **ESB** or a **Service Proxy** can be the solution to the mess created by the point-to-point approach. **ESB** = Enterprise Service Bus
- All service calls are directed to the **proxy** or **gateway**, which in turn, forwards the message to the appropriate endpoint destination.
- If an **endpoint is changed**, only the **proxy configuration** will be required to be **changed**.
- Each **component** communicates with **Proxy**. **Proxy** should know how to send the **message for destination**.
- Now **component free** from maintaining IP addresses of destination. That **responsibility delegated to 3<sup>rd</sup> party module called ESB**.
- **ESB mediate the message for exact endpoint based on proxy details.**



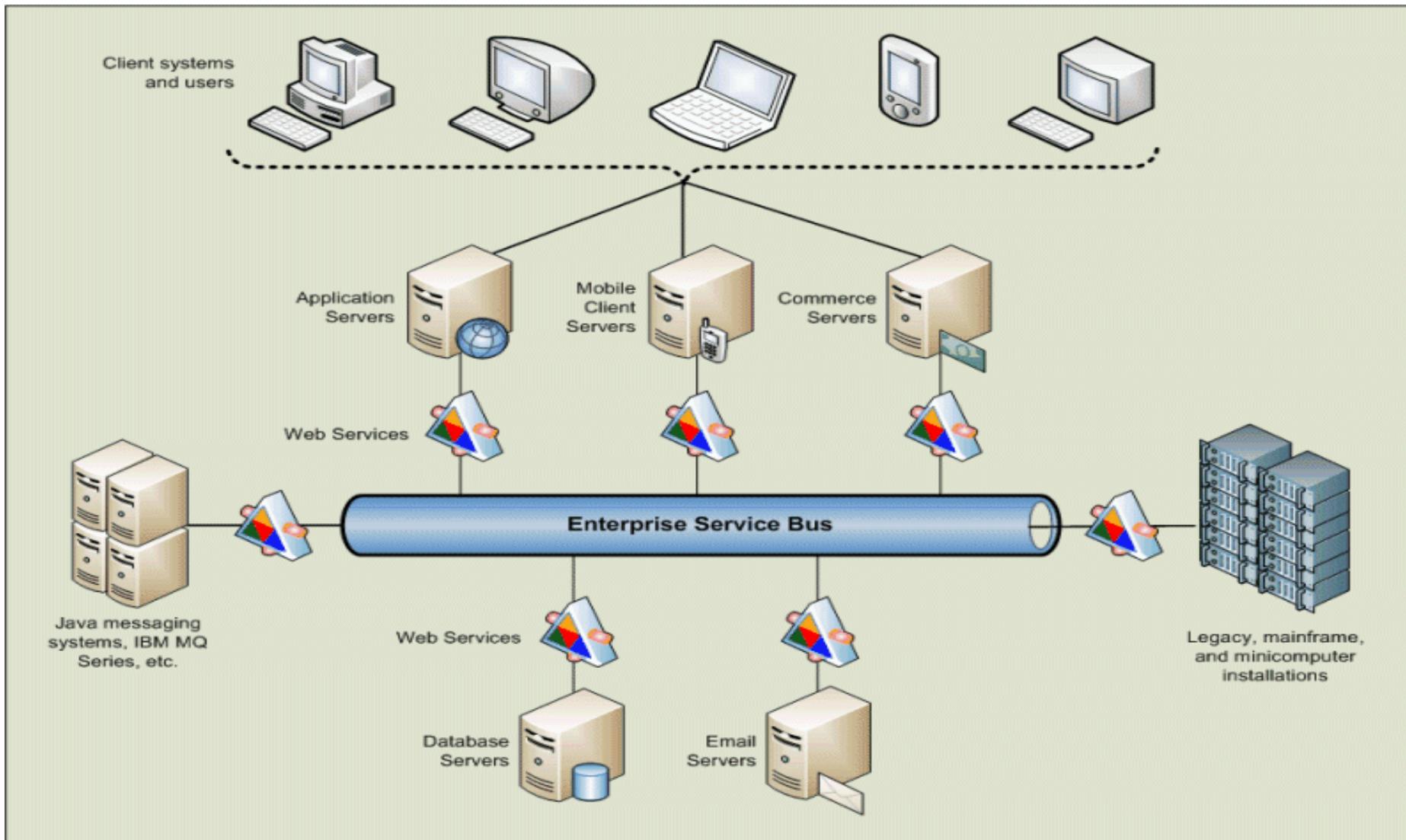
# What is an ESB?



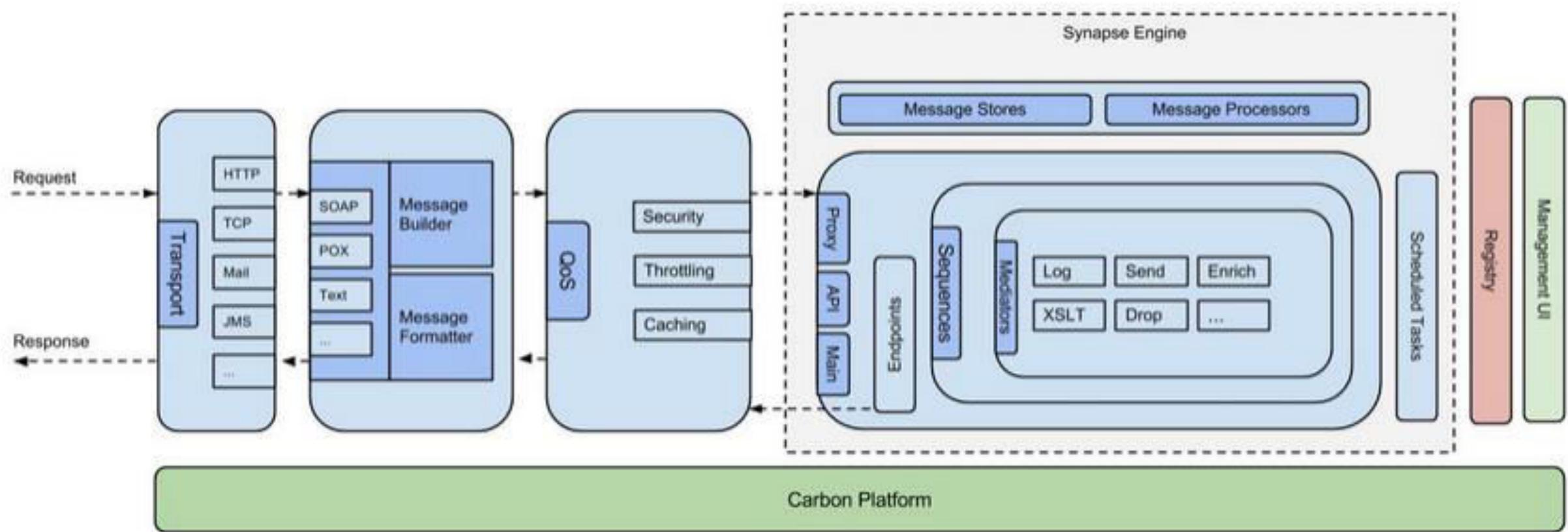
**ESB** “is a software architecture model used for designing and implementing the interaction and communication between **mutually interacting** software applications in **Service Oriented Architecture**”

- Promotes **asynchronous message mediation**
- Message **identification and routing** between applications and services.
- Allows messages to flow across **different transport protocols**
- **Transforming** of messages
- Allows **secure, reliable** communications
- Extensible architecture (based on pluggable components)

# What is an ESB ? Cont....

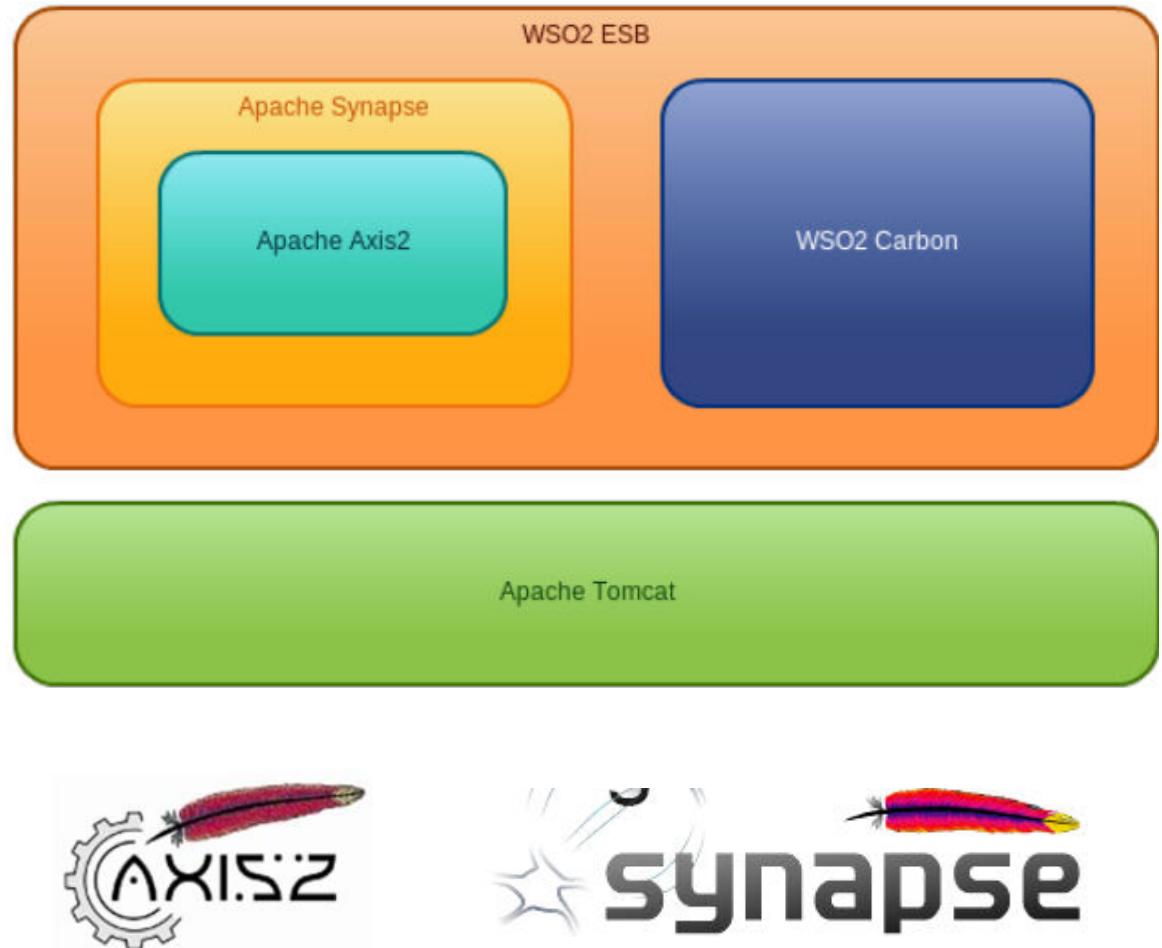


# WSO2 ESB Architecture



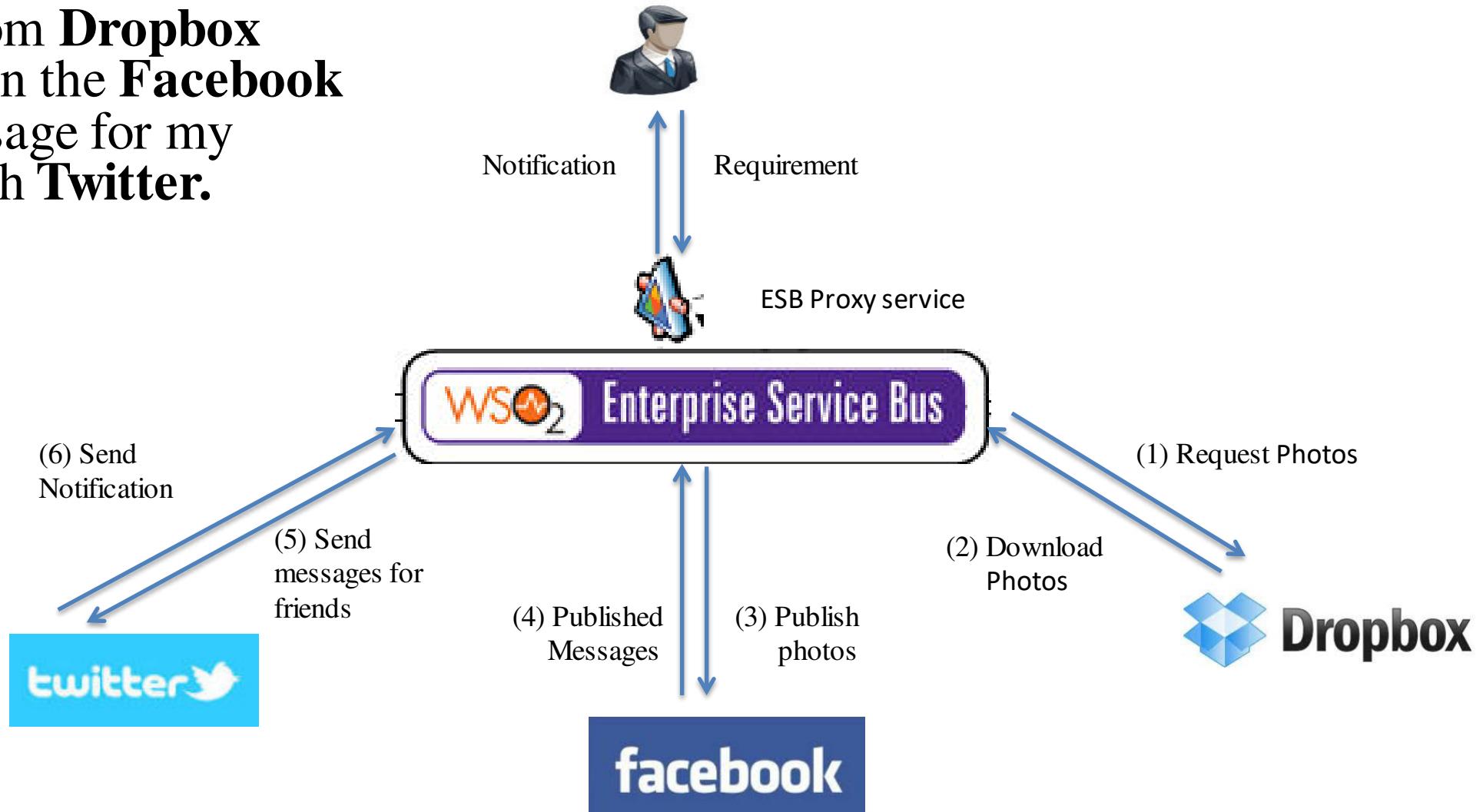
# Technology Stack

- WSO2's core product, Middleware platform.
- A dynamic component model built for Java
- Components can be **started, stopped, installed, uninstalled** etc without reboot
- Built on **OSGi concepts**
- Powers SOA capabilities
- EVERYTHING that **WSO2 builds is based on Carbon**
- **Apache Synapse:**
  - Based on Axis2/Java
  - Acts a mediation library for different protocols
  - Supports different protocols through SOAP based Proxy Services



# ESB Business Scenario.

- Get photos from **Dropbox** publish them in the **Facebook** and send message for my friends through **Twitter**.

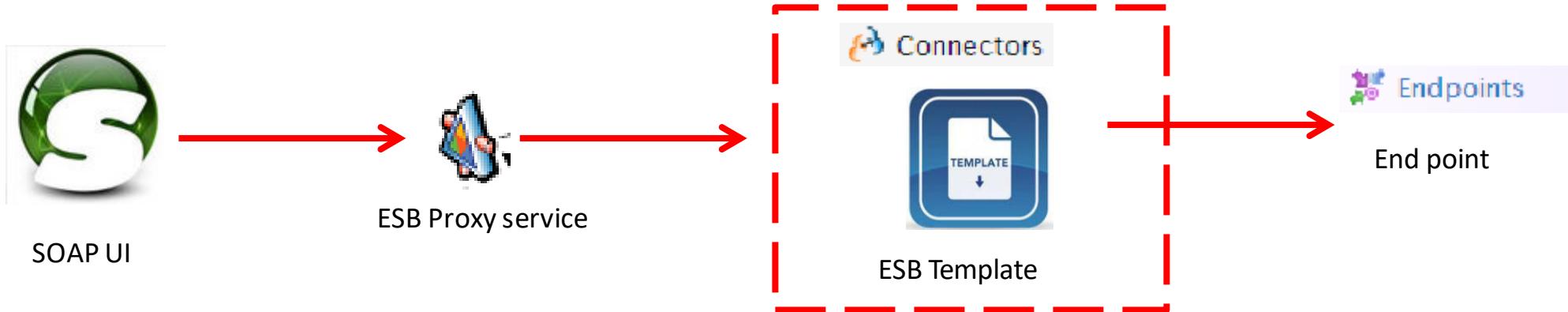


# Apache Synapse

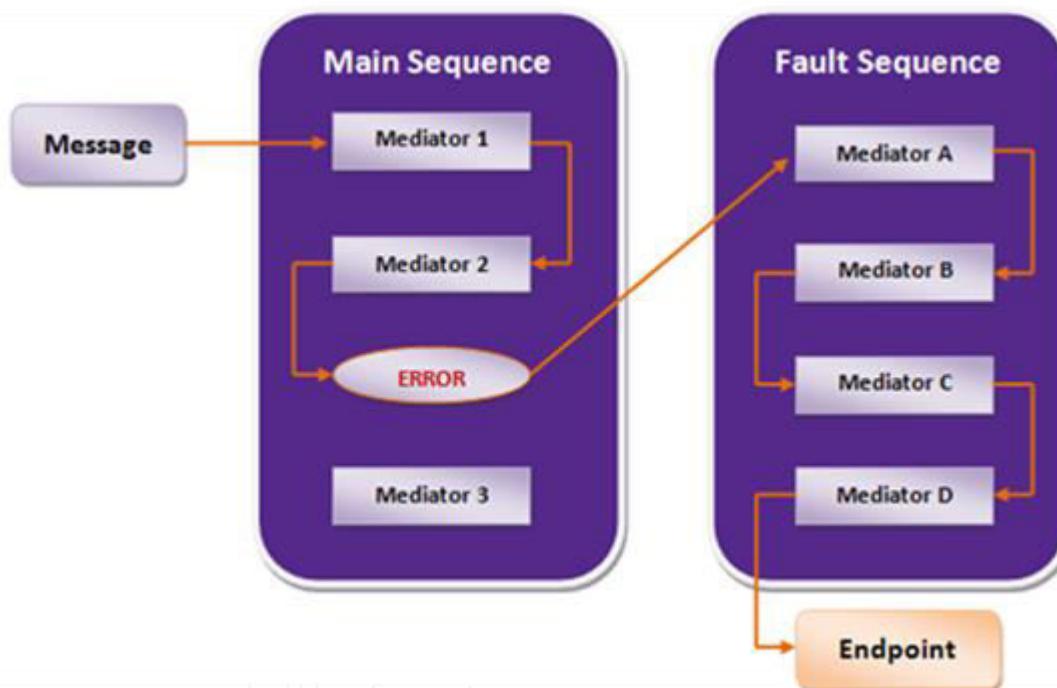
Apache Synapse “*is a lightweight and high-performance ESB with a powerful mediation engine.*”

- Default transport – **HTTP-NIO** (configurable pool of non-blocking worker threads)
- Support for any request types **XML**, **SOAP**, plain text, binary, **JSON** and etc.
- Supports any protocol **HTTP**, **HTTPS**, **Mail (POP3, IMAP, SMTP)**, **JMS**, **TCP**, **UDP**, **VFS**, **SMS**, **XMPP** and **FIX**
- Non-blocking **HTTP/HTTPS** transports
- Support for **WS-\*** standards (**WS-Addressing**, **WS-Security** and **WS-Reliable Messaging**)

# Flow of request



In case an error occurs in the main sequence while processing, the message goes to the fault sequence.



# Sample Request types

## SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
    xmlns:urn="urn:wso2.connector.googledrive.getfile">  
    <soapenv:Body>  
        <root>  
            <urn:fileId>1QL5LZOm9m4-h1lehXgU6gN4Kjovf8o3sqPKvw2RN40k</urn:fileId>  
            <urn:updateViewedDate>false</urn:updateViewedDate>  
            <urn:useServiceAccount>false</urn:useServiceAccount>  
            <urn:clientId>521684679704.apps.googleusercontent.com</urn:clientId>  
            <urn:clientSecret>AOZtcdakFwcnx1BuIavjtMEX</urn:clientSecret>  
            <urn:accessToken></urn:accessToken>  
            <urn:refreshToken>1/khM2ZQlpe_1PSp8WI</urn:refreshToken>  
            <urn:serviceAccountEmail>  
                757865184057@developer.gserviceaccount.com</urn:serviceAccountEmail>  
            <urn:fields>alternateLink,labels</urn:fields>  
        </root>  
    </soapenv:Body>  
</soapenv:Envelope>
```

## JSON Request

```
{  
    "accessToken": "AQV068KoSLBPT8U",  
    "apiUrl": "https://api.linkedin.com",  
    "publicUrl": "http://www.linkedin.com/pub/wso2connector-abdera/87/998/935",  
    "memberId": "12323"  
}
```

## POX Request

```
<createExpense>  
    <arbitraryPassword></arbitraryPassword>  
    <apiUrl>https://sansu.freshbooks.com</apiUrl>  
    <authenticationToken>  
        c361a63c7456519412fa8051ea605a6d</authenticationToken>  
    <staffId>1</staffId>  
    <status>1</status>  
    <vendor>Sun Tzu Auto</vendor>  
    <categoryId>5</categoryId>  
    <projectId>19410</projectId>  
    <date>2014-05-22</date>  
    <clientId>99962</clientId>  
    <compoundTax></compoundTax>  
    <amount>2000</amount>  
    <tax1Name>VAT</tax1Name>  
    <tax1Amount>200</tax1Amount>  
    <tax1Percent>10</tax1Percent>  
    <tax2Name>GST</tax2Name>  
    <tax2Amount>200</tax2Amount>  
    <tax2Percent>10</tax2Percent>  
    <notes>Expense Test</notes>  
    <compoundTax>true</compoundTax>  
</createExpense>
```

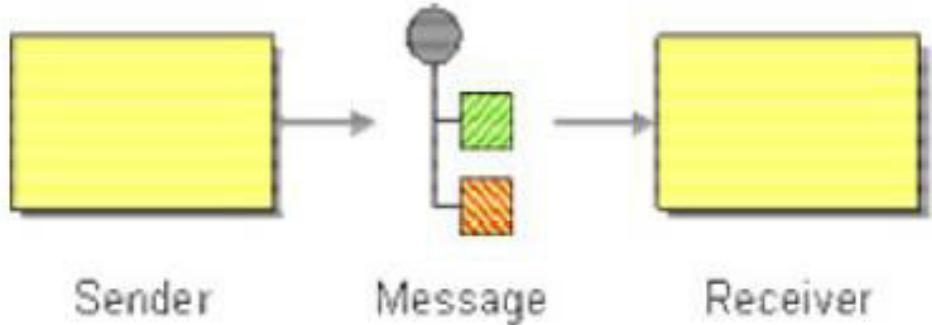
# Message Mediation

- A **mediator** is **the basic, full-powered message processing unit** in the ESB.
- A mediator can **take a message**, carry out some predefined actions on it, and output the modified message.
- Usually, a **mediator** is configured using **XML**.
- Different **mediators** have their **own XML configurations**.
- At the **run-time**, a **message** is injected in to the mediator with the **ESB run-time information**.
- Then this mediator can do virtually anything with the message.
- A user can write a mediator and put it into the ESB.

# Messaging

- **Messaging** is a form of *loosely coupled* distributed communication.
- ‘**Communication**’ can be understood as an exchange of messages between software components.
- **Message-oriented Middleware (MOM)** attempt to relax tightly coupled communication (such as TCP network sockets, CORBA or RMI) by the introduction of an “intermediary component”.
- **MOMs** allow software components to communicate ‘**indirectly**’ with each other.
- **Benefits of Messaging** include message senders **not needing** to have precise **knowledge of their receivers**.
- Thus any data that is to be transmitted via a messaging system must be converted into one or more **messages**

# Message Structure

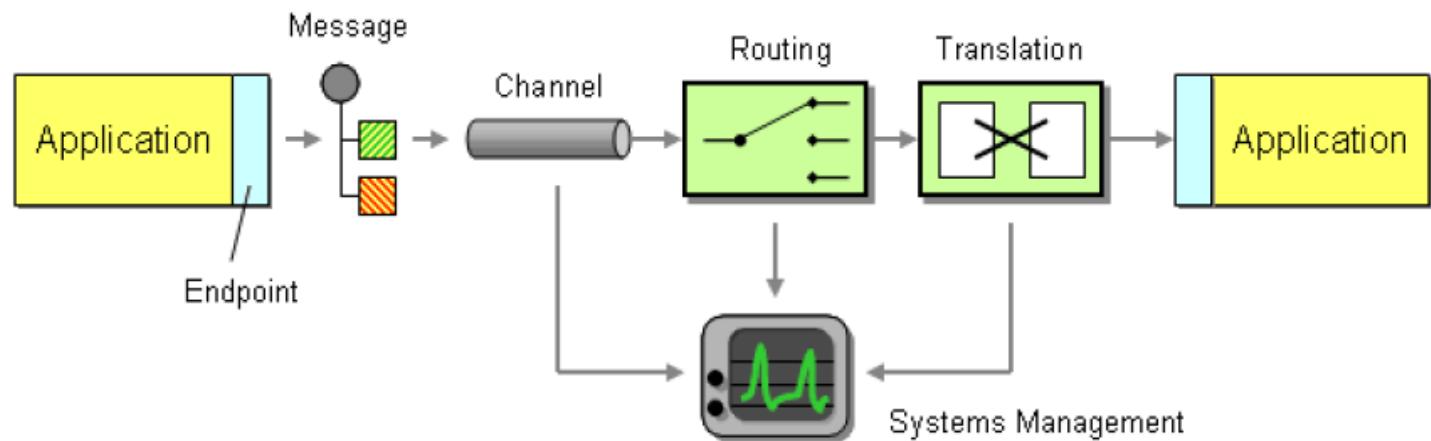


- Message consist of two basic parts.
- **Header**
  - Describes the **data being transmitted, its origin, its destination, MessageID, Timestamp, Priority, Delivery mode, Message type** and etc.
- **Body**
  - The data being transmitted; generally ignored by the messaging system and simply transmitted as-is.
- There are different kind of messages.
  - JMS Message
  - .NET Message
  - SOAP Message
- **Messaging play major role in Enterprise Application Integration.**

# Enterprise Integration Patterns

There are 7 root patterns

- 1) Messaging
- 2) Message Channel
- 3) Message
- 4) Pipes and Filters
- 5) Message Router
- 6) Message Translator
- 7) Message Endpoint



# Enterprise Integration Patterns - Messaging

- **Messaging**

- ✓ Is a technology that enables **high speed**, “**asynchronous**”, program-to-program communication with **reliable delivery**.
- ✓ Message itself is simply some sort of data structure such as a **string**, a byte array, a record, or an object



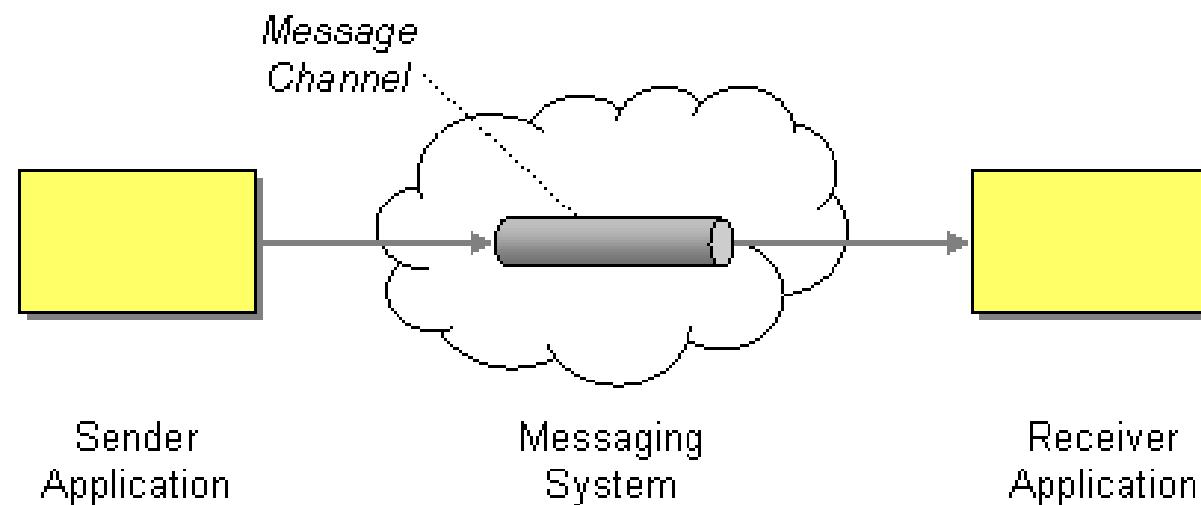
- **Messages**

- ✓ Programs communicate by sending *packets of data* called “**messages**” to each other.
- ✓ First process **marshals** the data into a byte stream and copy it from the first process to the second process.
- ✓ The second process **unmarshal** the data back to its original form.

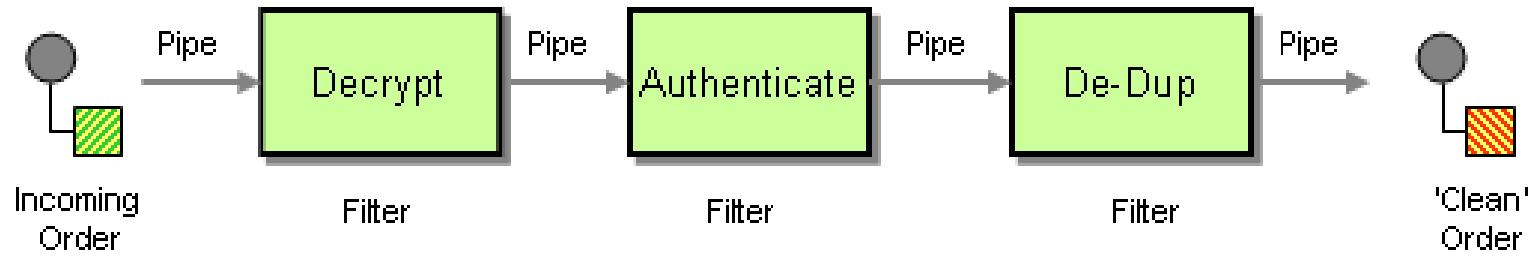
# Enterprise Integration Patterns – Message channel

- **Message Channel**

- ✓ Messaging applications transmit data through a **Message Channel**, a virtual pipe that connects a sender to a receiver
- ✓ The media that can move data from one application to the other

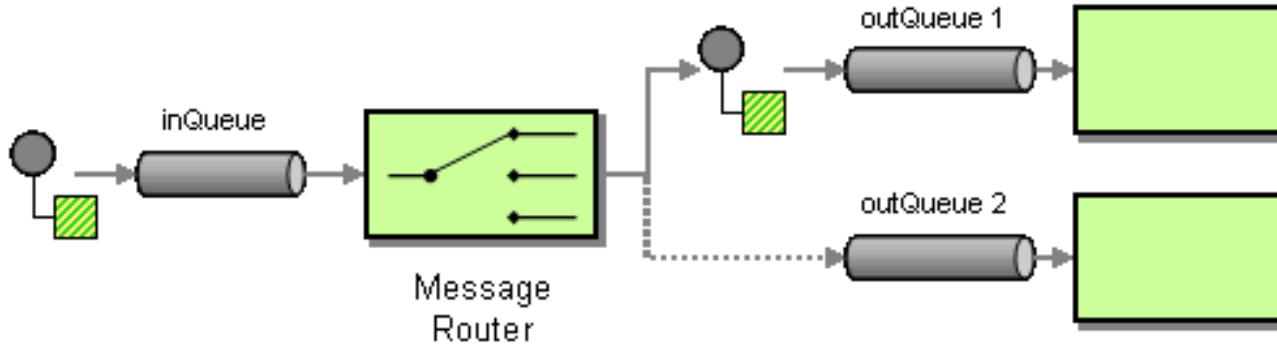


# Enterprise Integration Patterns – Pipes and Filters



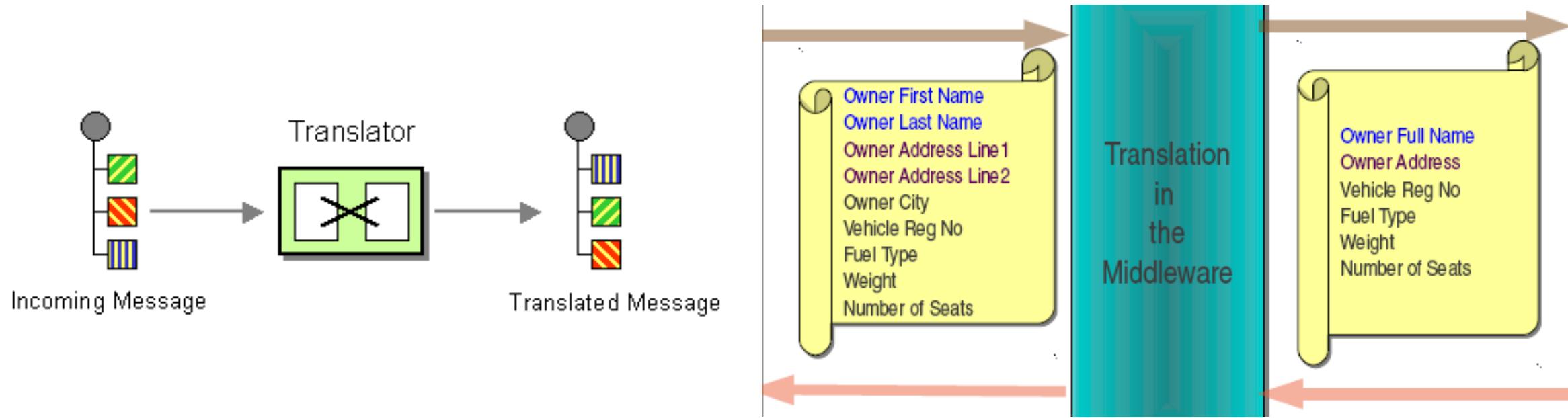
- Use the **Pipes and Filters** architectural style to **divide a larger processing task into a sequence of smaller, independent processing steps** (Filters)
- That are connected by channels (Pipes).
- Each filter exposes a very simple interface.
- The **connection between filter** and pipe is sometimes called **port**. In the basic form, each filter component has **one input port** and **one output port**.
- We can add new filters, omit existing ones or rearrange them into a new sequence.

# Message Router Pattern



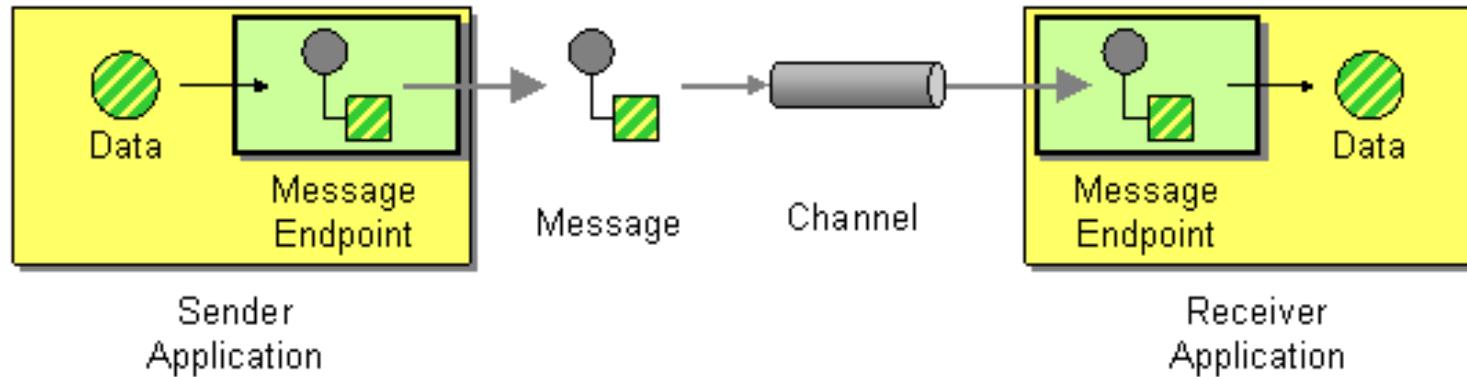
- Consumes a Message from **one Message Channel** and republishes it to a different Message Channel depending on a **set of conditions**.
- A key property of the **Message Router** is that **it does not modify the message contents**.
- If **new message types** are defined,
  - new processing components are added,
  - or the routing rules change,
  - we need to change only the Message Router logic and all other components remain unaffected

# Message Translator Pattern



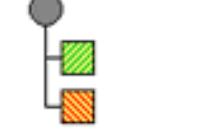
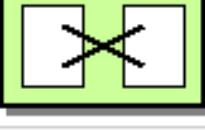
- The **Message Translator** is the messaging equivalent of the **Adapter pattern**.
- An **adapter** converts the **interface** of a component into a **another interface** so it can be used in a different context.
- **XSLT** Mediators can be used for message transformation.

# Message Endpoint Pattern



- Connect an application to a messaging channel using a **Message Endpoint**.
- A client of the messaging system that the application can then use to send or receive messages.
- It is the **endpoint** that receives a message, **extracts the contents, and gives them to the application** in a meaningful way.

# In a nutshell

	<b>Message Channel</b>	How does one application communicate with another using messaging?
	<b>Message</b>	How can two applications connected by a message channel exchange a piece of information?
	<b>Pipes and Filters</b>	How can we perform complex processing on a message while maintaining independence and flexibility?
	<b>Message Router</b>	How can you decouple individual processing steps so that messages can be passed to different filters depending on a set of conditions?
	<b>Message Translator</b>	How can systems using different data formats communicate with each other using messaging?
	<b>Message Endpoint</b>	How does an application connect to a messaging channel to send and receive messages?

# Proxy Services

- **Pass Through Proxy** - Forwards messages to the endpoint without performing any processing on them. This proxy service is useful as a catch-all, so that messages that do not meet the criteria to be handled by other proxy services are simply forwarded to the endpoint.
- **Secure Proxy** - Uses WS-Security to process incoming requests and forward them to an unsecured backend service.
- **WSDL Based Proxy** - A proxy service that is created from the remotely hosted WSDL of an existing web service. The endpoint information is extracted from the WSDL.
- **Logging Proxy** - Logs all the incoming requests and forwards them to a given endpoint. It can also log responses from the backend service before routing them to the client.
- **Transformer Proxy** - Transforms all the incoming requests using XSLT and then forwards them to a given endpoint. It can also transform responses from the backend service.
- **Custom Proxy** - A custom proxy service in which you customize the sequences, endpoints, transports, and other QoS settings.

# Custom proxy service.

The screenshot shows a web browser window with the URL <https://172.22.176.19:9443/carbon/proxyservices/source.jsp?anonEpAction=Create&header=Modify&origin>. The browser's address bar is redacted. The page content is a code editor displaying XML configuration for a proxy service. The code includes properties for region, secret access key, return consumed capacity, access key ID, version, expected values, table name, item, and return values. It also defines methods for initializing Amazon DynamoDB and putting an item, and includes a filter for axis2:HTTP\_SC. The code is numbered from 7 to 41.

```
7     startOnLoad="true">
8     <target>
9       <inSequence onError="faultHandlerSeq">
10      <property name="region" expression="json-eval($.region)"/>
11      <property name="secretAccessKey" expression="json-eval($.secretAccessKey)"/>
12      <property name="returnConsumedCapacity"
13        expression="json-eval($.returnConsumedCapacity)"/>
14      <property name="accessKeyId" expression="json-eval($.accessKeyId)"/>
15      <property name="version" expression="json-eval($.version)"/>
16      <property name="expected" expression="json-eval($.expected)"/>
17      <property name="tableName" expression="json-eval($.tableName)"/>
18      <property name="item" expression="json-eval($.item)"/>
19      <property name="returnValues" expression="json-eval($.returnValues)"/>
20      <property name="returnItemCollectionMetrics"
21        expression="json-eval($.returnItemCollectionMetrics)"/>
22      <property name="conditionalOperator"
23        expression="json-eval($.conditionalOperator)"/>
24     <amazondynamodb.init>
25       <region>{$ctx:region}</region>
26       <secretAccessKey>{$ctx:secretAccessKey}</secretAccessKey>
27       <returnConsumedCapacity>{$ctx:returnConsumedCapacity}</returnConsumedCapacity>
28       <accessKeyId>{$ctx:accessKeyId}</accessKeyId>
29       <version>{$ctx:version}</version>
30     </amazondynamodb.init>
31     <amazondynamodb.putItem>
32       <expected>{$ctx:expected}</expected>
33       <tableName>{$ctx:tableName}</tableName>
34       <item>{$ctx:item}</item>
35       <returnValues>{$ctx:returnValues}</returnValues>
36       <returnItemCollectionMetrics>{$ctx:returnItemCollectionMetrics}</returnItemCollectionMetrics>
37       <conditionalOperator>{$ctx:conditionalOperator}</conditionalOperator>
38     </amazondynamodb.putItem>
39     <filter source="$axis2:HTTP_SC" regex="^[^2][0-9][0-9]">
40       <then>
41         <switch source="$axis2:HTTP_SC">
```

# Sample proxy services

The screenshot shows the WSO2 Enterprise Service Bus Management Console interface. The top navigation bar includes the WSO2 logo, 'Enterprise Service Bus', 'Management Con...', 'Signed-in as: admin@carbon.super | Sign-out | Docs |', and a help icon. The left sidebar has tabs for Main (selected), Monitor, Configure, and Tools. Under Main, the 'Manage' section is expanded, showing 'Services' (List, Add, Proxy Service selected), 'Service Bus' (Sequences, Scheduled Tasks, Templates, Endpoints), 'Local Entries', 'Message Processors', 'Message Stores', 'Priority Executors', 'APIs', 'Source View', 'Connectors' (List, Add), and 'Secure Vault Tool'. The main content area shows a heading 'Deployed Services' with the sub-heading 'Home > Manage > Services > List'. It displays '19 active services. 19 deployed service group(s.)'. Below this is a search bar with 'Service Type ALL' dropdown, a service input field, and a magnifying glass icon. Navigation links include '<< first', '<prev', '1', '2', 'next >', and 'last >>'. Below the search bar are links for 'Select all in this page', 'Select all in all pages', and 'Select none'. A 'Delete' button with a trash icon is also present. The 'Services' table lists five entries:

	Service Name	Type	Security	WSDL1.1	WSDL2.0	Action	Design View	Source Vi
1	amazondynamodb_putitem	proxy	Unsecured	WSDL1.1	WSDL2.0	Try this service		
2	amazonses_listIdentities	proxy	Unsecured	WSDL1.1	WSDL2.0	Try this service		
3	bugherd_addTaskComment	proxy	Unsecured	WSDL1.1	WSDL2.0	Try this service		
4	bugherd_createProjectTask	proxy	Unsecured	WSDL1.1	WSDL2.0	Try this service		
5	bugherd_deleteTaskAttachment	proxy	Unsecured	WSDL1.1	WSDL2.0	Try this service		

# Connector Template Implementation

```
<template name="updateProjectTask" xmlns="http://ws.apache.org/ns/synapse">
    <parameter name="taskId" description="ID of the Project Task to be updated."/>
    <parameter name="task" description="The task JSON object to be sent to the API."/>
    <sequence>
        <property name="uri.var.task" expression="$func:task"/>
        <property name="uri.var.taskId" expression="$func:taskId"/>

        <payloadFactory media-type="json">
            <format>
                {"task":$1}
            </format>
            <args>
                <arg expression="get-property('uri.var.task')"/>
            </args>
        </payloadFactory>

        <property name="DISABLE_CHUNKING" value="true" scope="axis2"/>

        <call>
            <endpoint>
                <http method="put" uri-template="{uri.var.apiUrl}/api_v2/projects/{uri.var.projectId}/tasks/{uri.var.taskId}.json"/>
            </endpoint>
        </call>

        <!-- Remove custom Headers from the Response -->
        <header name="Via" scope="transport" action="remove"/>
        <header name="ETag" scope="transport" action="remove"/>
        <header name="X-Runtime" scope="transport" action="remove"/>
        <header name="X-Powered-By" scope="transport" action="remove"/>
        <header name="X-Rack-Cache" scope="transport" action="remove"/>
        <header name="X-Request-Id" scope="transport" action="remove"/>
        <header name="X-Frame-Options" scope="transport" action="remove"/>
        <header name="X-UA-Compatible" scope="transport" action="remove"/>
        <header name="X-XSS-Protection" scope="transport" action="remove"/>
```

# ESB Connector

➤ Aggregation of mediators => is called **sequence**

➤ Aggregation of **sequences** => is called **template**

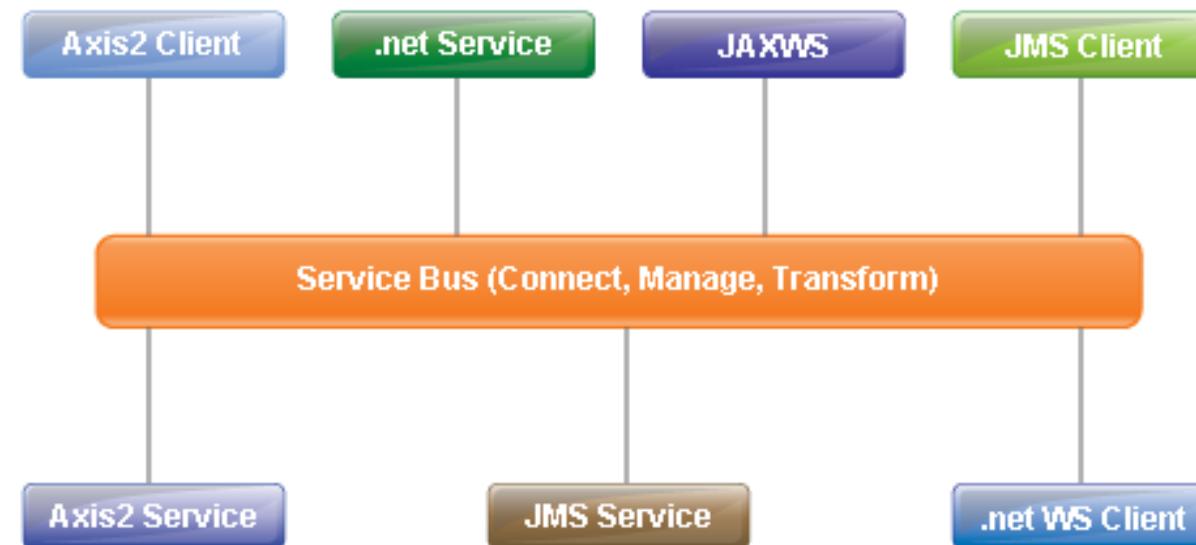
➤ Aggregation of **templates** => is called **connector**

The screenshot shows the WSO2 ESB Manager interface. On the left, there is a sidebar with a purple header 'Manage' containing various service-related options: Services (selected), List, Add, Proxy Service, Service Bus, Sequences, Scheduled Tasks, Templates, Endpoints, Local Entries, and Message Processors. The 'List' option is highlighted. The main area has a white header 'Connectors' and a blue sub-header 'Connector List'. Below this is a table with the following data:

Library Name	Package	Description	Status	Actions	
bugherd	org.wso2.carbon.connector	wso2 connector library for BugHerd	Enabled		
amazonses	org.wso2.carbon.connector	wso2 connector library for AmazonSES	Enabled		
amazondynamodb	org.wso2.carbon.connector	Amazon Dynamo DB connector library	Enabled		
magento	org.wso2.carbon.connector	Magento connector library	Enabled		

# Service Bus and Mediation

- Service Bus is the **common communication channel** that is used by all the parties in messaging.
- Mediation is the process of facilitated communication between parties by providing intermediary conflict resolutions.



# References

- <https://docs.wso2.com/display/ESB480/Using+a+Connector>
- <https://docs.wso2.com/display/ESB480/JIRA+Connector>
- <https://docs.wso2.com/display/ESB480/Managing+Connectors+in+Your+ESB+Instance>
- <https://docs.wso2.com/display/ESB480/Mediators>