17-723: Designing Large-scale Software Systems

Design for Interoperability

Tobias Dürschmid

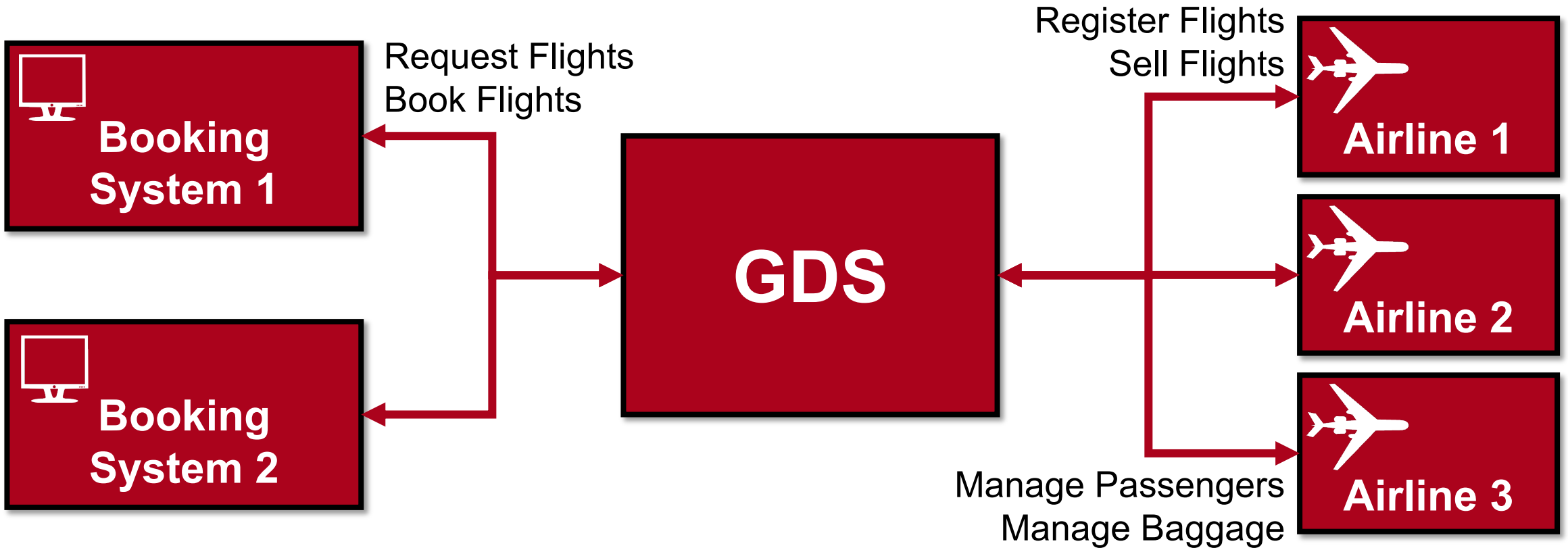
Carnegie Mellon University

Learning Objectives

- Describe interoperability and its importance as a quality attribute
- **Generate** design options for interoperability by applying appropriate designing principles
- **Communicate** the interoperability of design options via appropriate design abstractions of interface descriptions
- **Evaluate** the interoperability of a given design option for a given context
- Describe common trade-offs between interoperability and other quality attributes

Carnegie Mellon University

Case Study: Global Distribution System (GDS)



Carnegie Mellon University

Why should I care about **Interoperability**?

Definition of Interoperability

Interoperability does not exist in isolation, but only with respect to other systems

The degree to which **two or more systems** (or components) can usefully exchange meaningful information in a particular context.

Carnegie Mellon University

Why should I care about **Interoperability**?

Your System is NOT a Lonely Island

Carnegie Mellon University

Lesson Learned: **Interoperability Lets You Use Services**

Rather than implementing the functionality of external services yourself, you can **use existing service providers**

E.g.: Authentication, Cloud Storage, Payment services, Content Delivery, Analytics & Monitoring, Google Maps, …

Carnegie Mellon University

Lesson Learned: **Interoperability Can Improve Usability**

Users can bring their data from another system

or **transfer data** from your system into another system

(e.g., electronic patient records in medical systems)

Carnegie Mellon University

Lesson Learned: **Interoperability Supports Cross-Platform Solutions**

Many systems need to interface with **separately developed Systems** (e.g., Mobile Apps, IoT systems, Microservices, …).

Interoperability simplifies communication between them.

Carnegie Mellon University

In-Class Activity: Describe Interoperability Requirements for GDS

How to Describe Interoperability Requirements?

Scenario

1. The **Systems** that should Collaborate
2. The Types of **Data** they Should Exchange

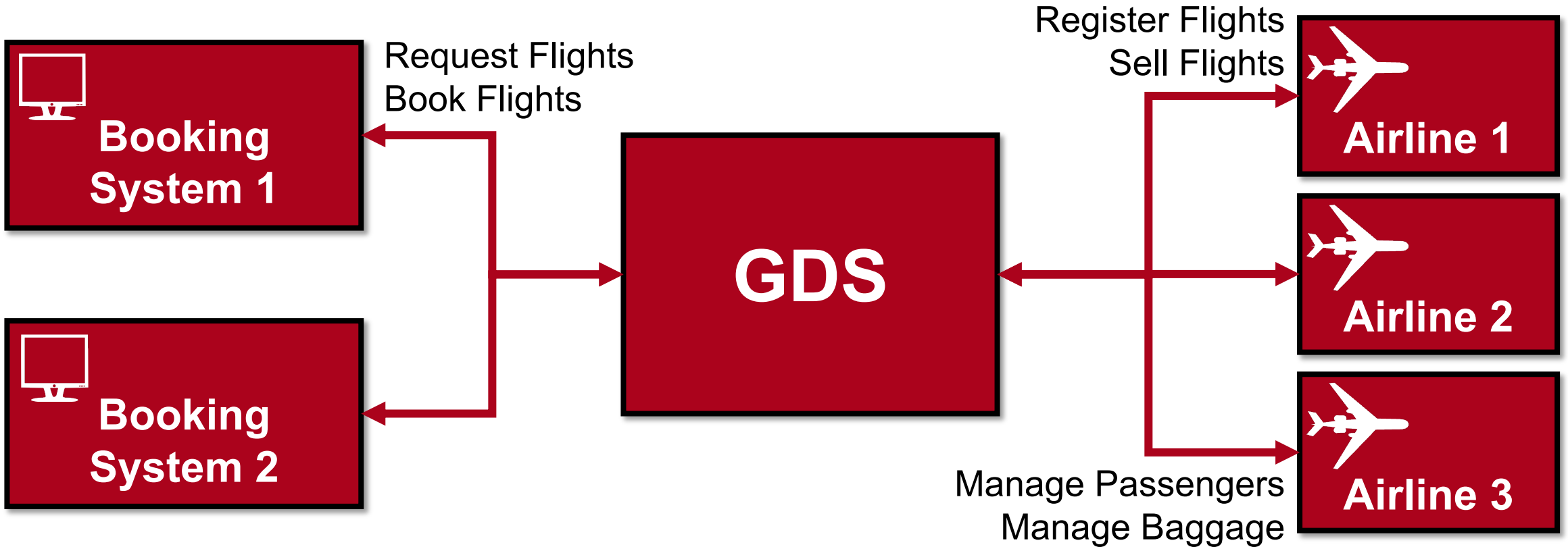
Measure

The **Percentage** of **Data** that has been **Exchanged Correctly**

Carnegie Mellon University

In-Class Activity: Invent a Design Principle to Design Systems for Interoperability

What makes GDS Interoperable?



Carnegie Mellon University

Design Principle for Interoperability: **Create Shared Interfaces / Data Formats**

1. **List all data** that needs to be exchanged

2. Define an **interface** / **data format** that supports all data

3. Implement **serialization** & **deserialization**

Carnegie Mellon University

Design Principle for Interoperability: **Create Shared Interfaces / Data Formats**

Build **Abstractions**

The interface / document format should **not expose any implementation details** of systems / components.

Ensure **Language- and Platform-Independence**

The format should be supported by all programming languages, operating systems, and devices.

Carnegie Mellon University

Common Technique to Implement Shared Interfaces: **REST APIs**

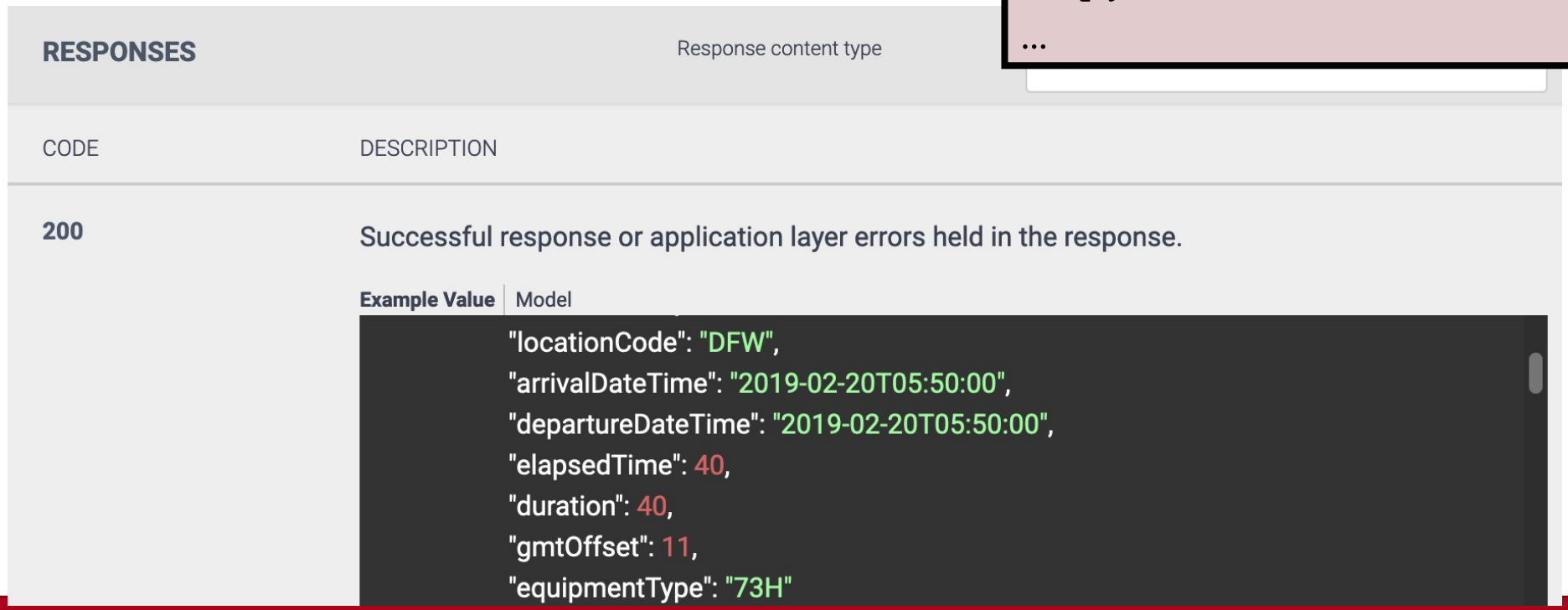
- REST (representational state transfer) is a stateless protocol to exchange data in client-server systems via HTTP / HTTPS:
  - `POST` – Creates a Resource
  - `GET` – Reads a Resource
  - `PUT` – Updates a Resource
  - `DELETE` – Deletes a Resource
- Resources are often described via XML, YAML, JSON, or HTML

Carnegie Mellon University

Example REST API: GDS API

- Request: **`POST /getFlights`**

```
…
"travelPreferences": {
    "flightType": "Direct",
    "maximumStopsQuantity": 1
},
"itineraryParts": [ {
    "departureAirportCode": "PIT",
    "arrivalAirportCode": "DFW",
… ]}
…
```

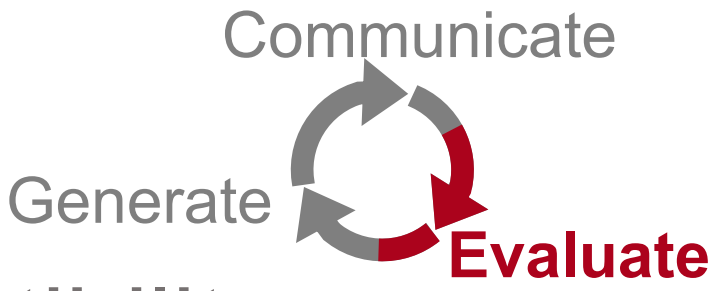


Carnegie Mellon University

Alternative Techniques to Implement Shared Interfaces (besides REST)

RPC (Remote Procedure Call)	Calling a function on a remote server as if it were local. Can be stateful. **Functions** and **actions** beyond CRUD. Good for **complex calculations.** Can use multiple document formats.
SOAP (Simple Object Access Protocol)	Can be stateful. More **complex**. Sometimes **slower**. Has more **security** features. Has built-in **error handling**. Good for **distributed enterprise environments**. Uses XML.
GraphQL	Server-side schema defines types, enabling checking of data structure conformance. Good for **large**, **complex**, and **interrelated** data sources. Uses JSON.

Carnegie Mellon University



Common Technique to Test Compatibility: **XML / JSON / YAML Schema**

- A schema **describes the structure** of document
- Schemas list attributes, their possible values, and complex types (e.g., sequences, recursive types) for a document
- **Validation** of a document against a schema can be done **automatically** to **test compatibility at run-time**

Carnegie Mellon University

Example JSON Schema

```
"properties": {
        "departureAirportCode": {
                "type": "string"
                "pattern": "^[A-Z]{3}$"
        },
        "price": {
                "type": "number",
                "minimum": 0,
                "exclusiveMinimum": true
        }
},
"required": ["departureAirportCode", "price"]
```

Constraints on the Type of a Property

Constraints on the Values of a Property

Constraints on Document Structure

Carnegie Mellon University

Question: How could they have prevented this bug?

Case Study
Mars Climate Orbiter

Spacecraft Lost Due to Lack of *Semantic Interoperability* (Shared Interpretation of Shared Data)

Flight System Software
Developed by NASA JPL

Expected commands in *N (SI units)*

Command Interface

Ground Software
Supplied by Lockheed Martin (US-based sub-contractor)

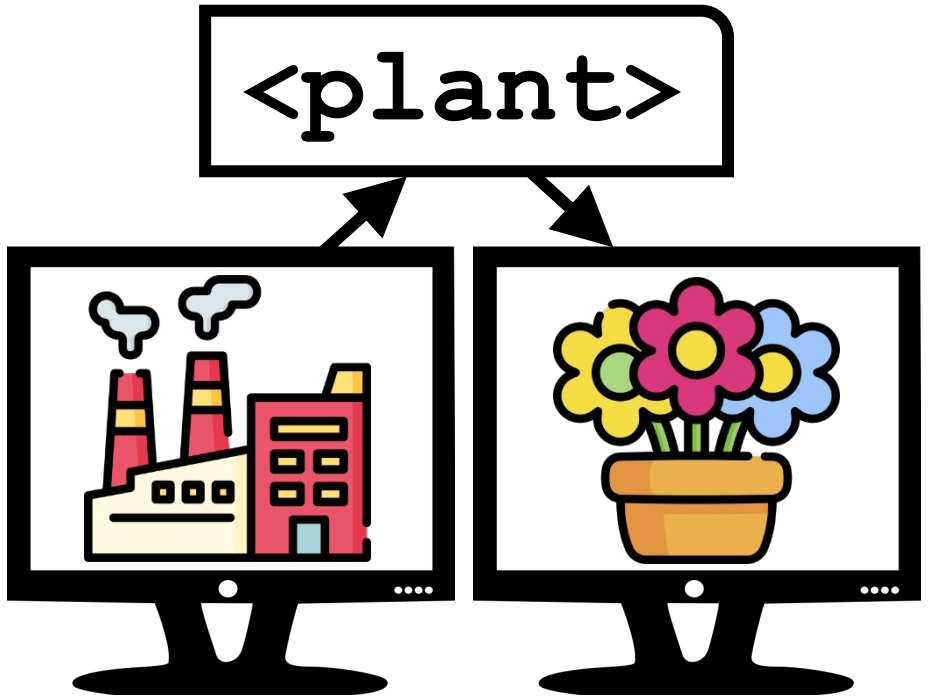
Sent commands in *lbf (US Customary units)*

Carnegie Mellon University

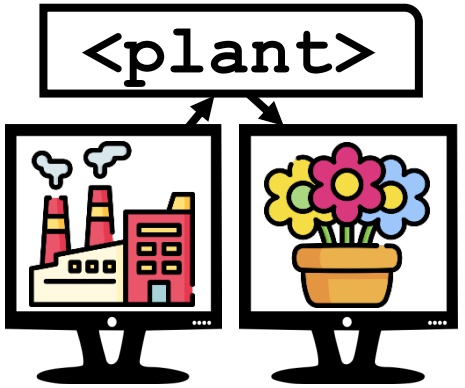
In-Class Activity: Invent a Design Principle for Semantic Interoperability

Lesson Learned:
Syntactic Interoperability is Not Enough

Data exchanged between systems / components must be **interpreted** in the same way by all systems / components.



Carnegie Mellon University



Design Principle for Interoperability: **Define the Semantics of Shared Data**

- **Document Interfaces** and their **Semantics** (e.g., What units are implied? Does price include tax? MM/DD/YY or DD/MM/YY? What coordinate system is used a reference frame?)
- **Use Shared Dictionary of Items** or **Agree on Vocabulary** (e.g., doughnut or donut? 单丛茶 or单枞茶?)
- Write **Integration Tests** for the Systems

In-Class Activity: Describe The Semantic View of GDS Flight Booking API

Interface Descriptions

Syntactic View	Describe document format, the actions that can be performed, their parameters, and outputs.
Semantic View	Describe the purpose / meaning of the resource / action: • **Side-effects**: Changes to the state of a resource or environment • **Usage restrictions**: Who can perform this action? • **Error Handling**: What errors can occur and why? • **Examples**: Examples of outputs for a given input

Carnegie Mellon University

Example: Semantic View of GDS Booking

- **Purpose**: The airline confirms a requested booking
- **Side-effects**: money is charged, seat is marked as sold, can be canceled within 1 hour
- **Usage restrictions**: Authorized booking systems
- **Errors**: Invalid Format, Unauthorized, too many requests, …

See here: https://developer.sabre.com/docs/rest_apis/trip/orders/booking_management

Carnegie Mellon University

Guidelines on Interface Documentation

- Focus on how **elements interact** and their **externally visible effects**, not their implementation
- To support **changeability** of the implementation, expose only **what is needed** to use the interface (See **Information Hiding**)
- Keep the documentation **minimal** and **use-case oriented** to increase **readability**

In-Class Activity: Describe Pros and Cons of the GDS Interface Documentation

GDS Interface Documentation

POST `/getFlights`

Creates a list of itineraries with the given search criteria during exchange. The response contains a list of itineraries with fare offerings and detailed price breakdown.

PARAMETERS CANCEL

NAME	DESCRIPTION
X-Request-ID `string` *(header)*	The unique ID of the request to track flow level transactions. X-Request-ID
AirSearchRequest `object`	Contains the required parameters to search for the itineraries. **Edit Value** | Model

In-Class Activity: Describe Pros and Cons of the GDS Interface Documentation

GDS Interface Documentation

OnTimePerformance {

description: Contains the statistical data on how often the flight is on time.

level — string
pattern: ^[0-9]{1}$
example: 1

The numeric value (0-9) associated with the level of on time performance of the flight.

***Semantic Interoperability* Via Shared Dictionary**

AirportCode string

pattern: ^[A-Z]{3}$

example: DFW

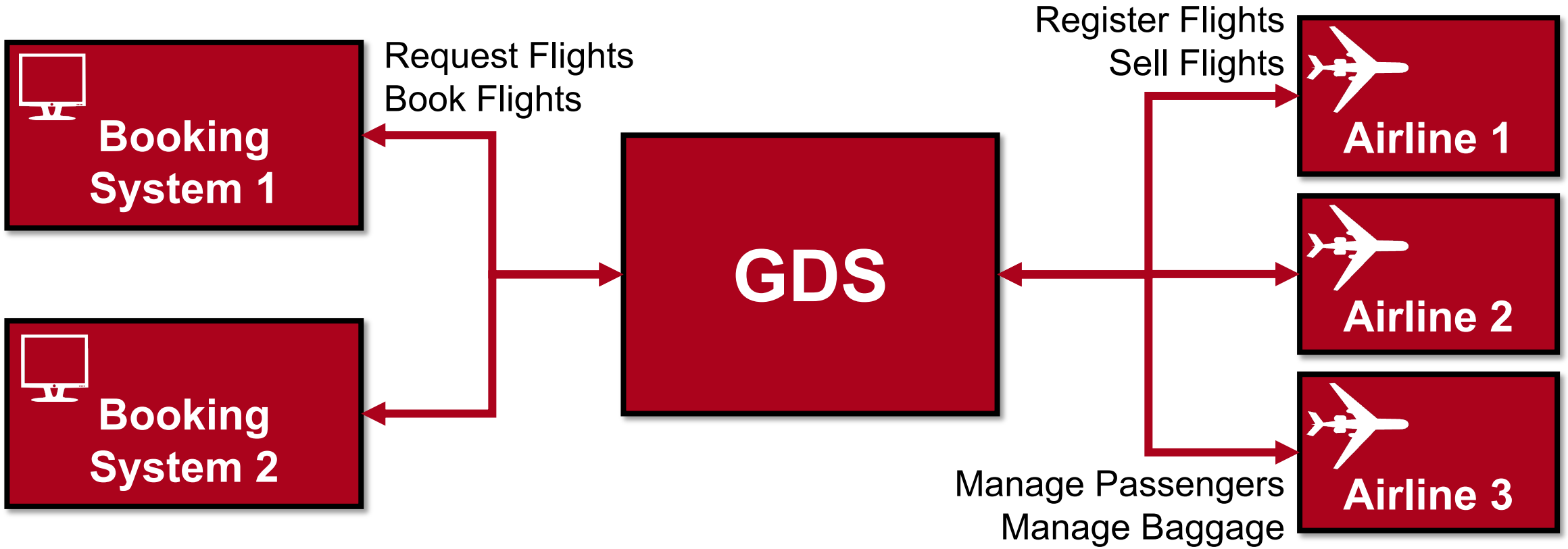
The three-letter IATA code of the airport.

Semantics of Level Are Unclear

Carnegie Mellon University

In-Class Activity: Generate Design Options to Increase the Changeability of GDS Booking Options

What makes GDS **Less Changeable**?



Carnegie Mellon University

In-Class Activity: What Disadvantage does this have over the existing GDS?

Making GDS More Changeable

Harder to Implement

Extensible Interfaces:

- Offers can contain a dynamically-defined add-ons
- Add-on: **`(price, name, description, id)`**
  - **`price(int)`**: The price in cents (excl. tax) additionally charged when this add-on is selected
  - **`name(str)`**: The name of the add-on as shown to the user (in UTF-8)
  - **`description(str)`**: A short description shown to the user in order to decide if they want to purchase the add-on (in UTF-8)
  - **`id(str)`**: Unique identifier of this add-on starting with the flight number (in ASCII)
- A list of add-ons is added to a flight listing. The booking API needs to add an optional list of add-on **`id`**s to identify requested add-ons.

Carnegie Mellon University

Lesson Learned: **Interoperability Often Conflicts with Changeability**

- Shared interfaces and data format mean changes have to be implemented **in all** cooperating systems
- We **cannot localize** the interface change to a single system, so extensions / changes require a new version of the interface, which breaks interoperability

Carnegie Mellon University

How to Evaluate **Practical Interoperability**?

Measure the **Effort to Implement** the Interface in all Systems / Components	Measure the **Variability** Allowed by the Interface / Format
A **more complex** interface / data format is less likely to be adopted by many systems due to higher implementation cost.	An interface / data format that supports **more use cases** and potential **extensions** is more likely to be adopted.

Carnegie Mellon University

Question: Does this only work for Syntactic Interoperability?

Design Pattern for Interoperability: Use *Adapters* to Connect Interfaces

Problem: Two systems use **different interfaces** (e.g., different data formats, different protocols, …)

Solution: Create an **adapter component** that **translates** between the two interfaces.

Carnegie Mellon University

Design Principles From This Lecture Apply!

Interoperability for Microservices

Microservices can be seen as **interoperating components**.

They can be written in **different programming languages** and **exchange data** via REST APIs / SOAP APIs / RPC.

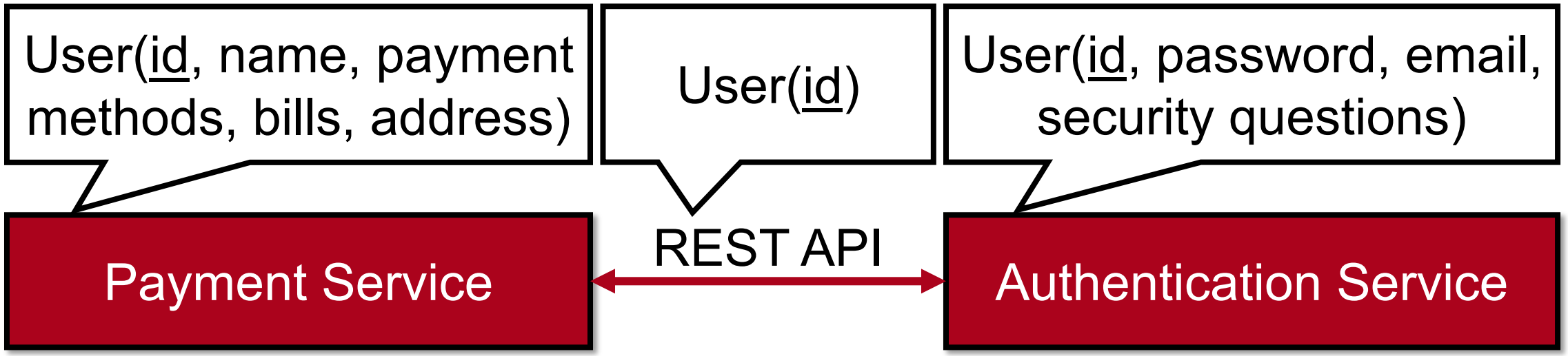
Orchestration of microservices can require complex interactions.

Carnegie Mellon University

Often improves **Changeability**

Bounded Context

Each microservice has its **own data model** of the entities it is handling to separate concerns. Interfaces should be minimal.

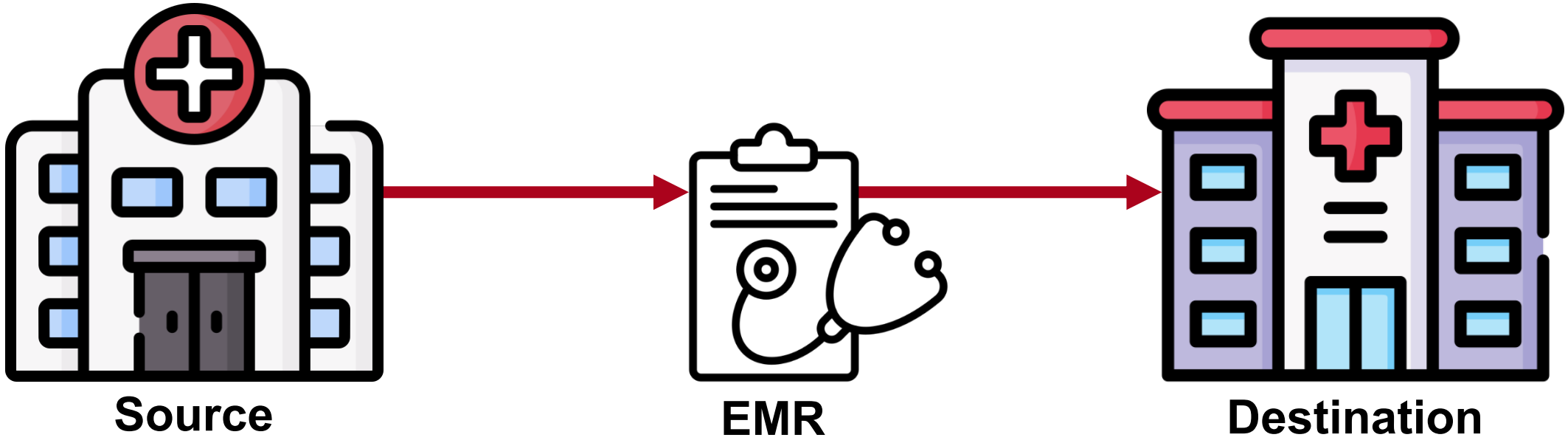


In-Class Activity: Specify Interoperability Requirements

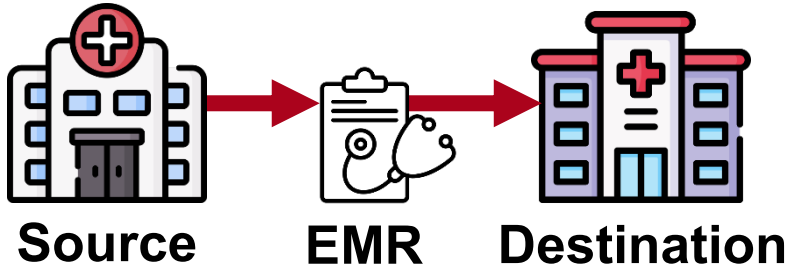
In-Class Activity: How can this data be transferred between hospitals?

Interoperability for Healthcare Systems

Patients want to get their data (**EMR** = electronic medical record) from one hospital and bring it to another hospital



In-Class Activity: What kind of data needs to be exchanged (Syntax)?



Interoperability for Healthcare Systems

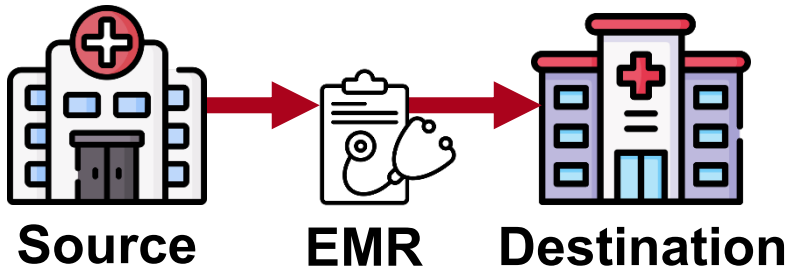
Central Network (Like GDS)

- \+ Better Usability / Flexibility
- \- Higher Development Cost
- \- Hospitals Need to Share EMR Interface

Manual Data Transfer (Prints, CDs)

- \+ Fewer Security Concerns
- ± Keep Patients
- \- Higher Labor Cost for Data Input

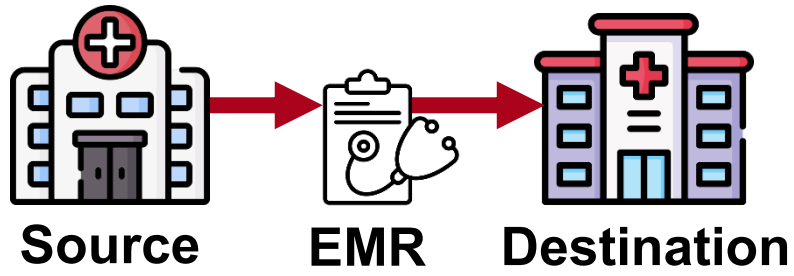
In-Class Activity: How to Ensure Semantic Interoperability of EHRs?



Interoperability for Healthcare Systems

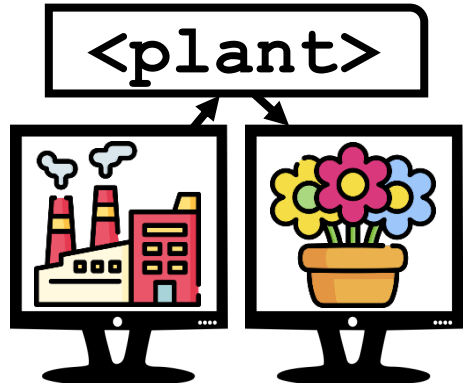
Patient Demographics	Name, date of birth, gender, contact details
Medical History	Past medical conditions, surgeries, hospitalizations, allergies, immunizations, … **Format: List of structured documents**
Test Results	Results of various tests such as vitals, blood tests, imaging scans, biopsies, … **Format: images, tables, free text**

Carnegie Mellon University



Interoperability for Healthcare Systems

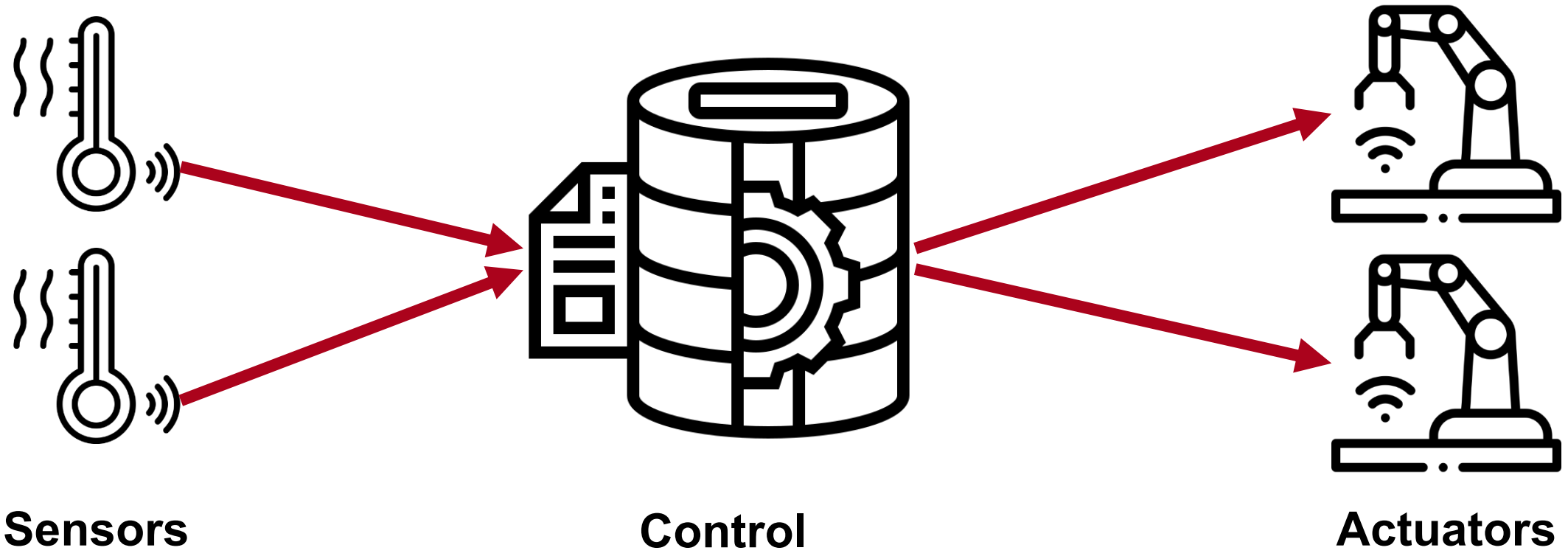
Standardized Terminologies	SNOMED CT (Systematized Nomenclature of Medicine Clinical Terms)
Semantic Data Models	RDF (Resource Description Framework) OWL (Web Ontology Language)
Adapters	Data mapping and transformation between different formats / standards / units



Carnegie Mellon University

In-Class Activity: What do you need to make smart factories interoperable?

Interoperability for Industry 4.0 Systems



Carnegie Mellon University

Please Complete the Exit Ticket in Canvas!

Question 1 1 pts

Please describe **two design principle** for Design for Interoperability. One for **syntactic** interoperability **and** one for **semantic** interoperability.

Question 2 1 pts

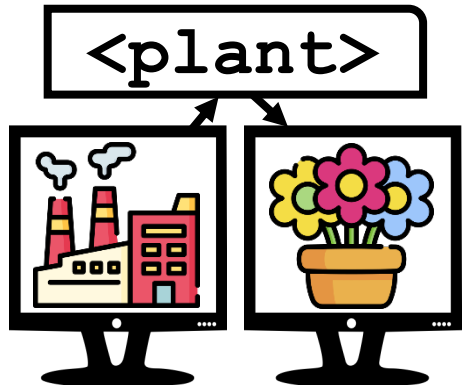
Please describe **why** interoperability often conflicts with changeability (1-2 sentences).

Question 3 1 pts

Please leave any questions that you have about today's materials and things that are still unclear or confusing to you (if none, simply write N/A).

Carnegie Mellon University

Summary



- Interoperability Is Important to **Offer / Use Services**
- To build Interoperable Systems, **Create Shared Interfaces / Data Formats**
- Syntactic Interoperability is Not Enough → **Semantic Interoperability**
- Define the Semantics via **Interface Documentation** and **Vocabulary**
- Interoperability Often **Conflicts with Changeability**
- Use ***Adapters*** to Connect Interfaces

Credits: These slide use images from Flaticon.com (Creators: Freepik, Eucalyp, LAFS, vectorslab, Vector Stall) and free-vectors.net