17-423/723: Designing Large-scale Software Systems

Design for Interoperability

Feb 21, 2025

Learning Goals

- Describe the importance of interoperability as a quality attribute of a software system

- Describe the difference between syntactic vs. semantic interoperability

- Apply design principles for achieving semantic interoperability

Content based on slides by Tobias Dürschmid

What is Interoperability?

- A quality attribute describing how well a system communicates and integrates with other systems

Apple has claimed that it continues to use Lightning because replacing it would supposedly produce "an unprecedented amount of electronic waste". Some reviewers...have posited that it is simply because Apple wants to continue profiting from its proprietary chargers and accessories.
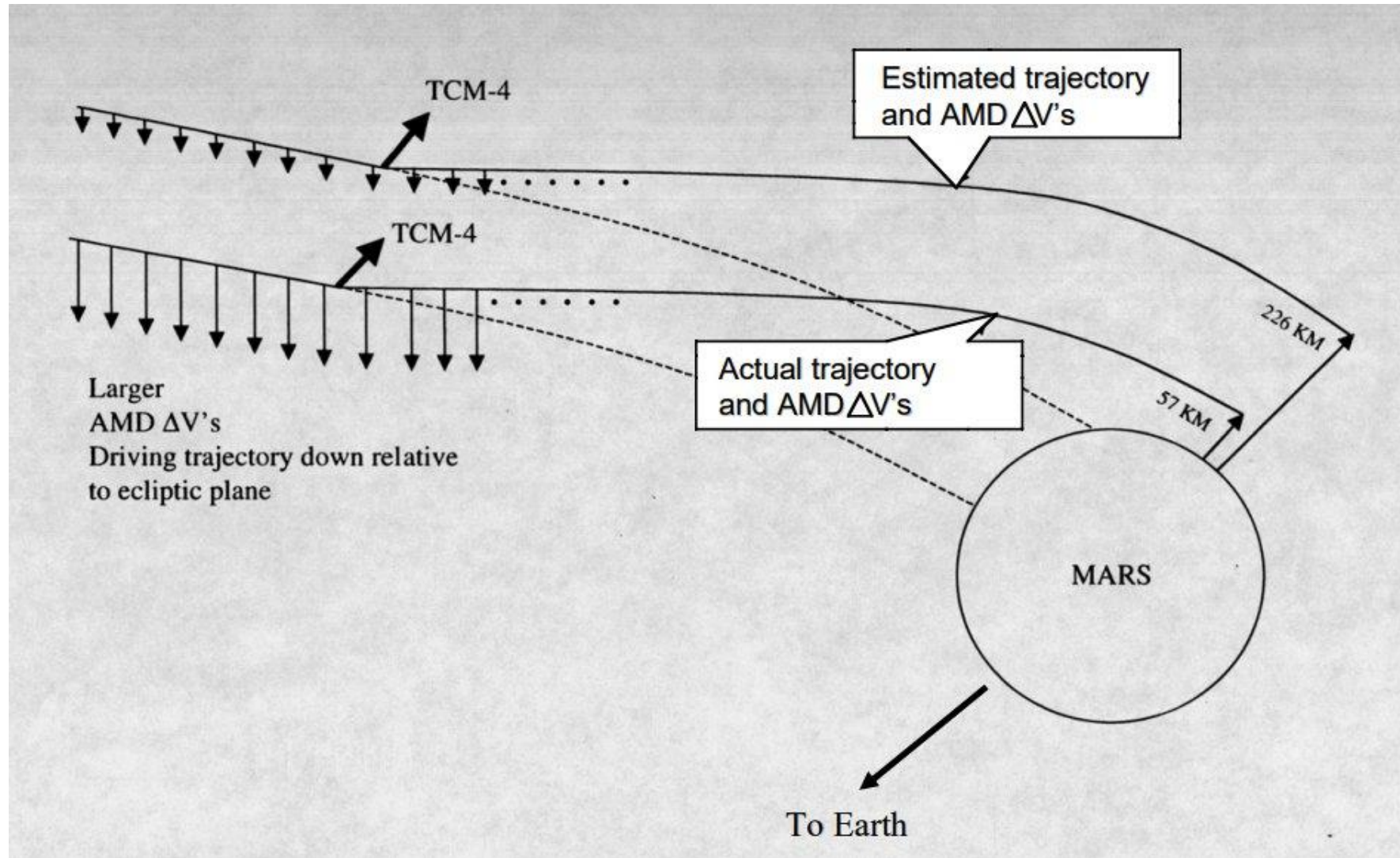
https://en.wikipedia.org/wiki/Lightning_(connector)
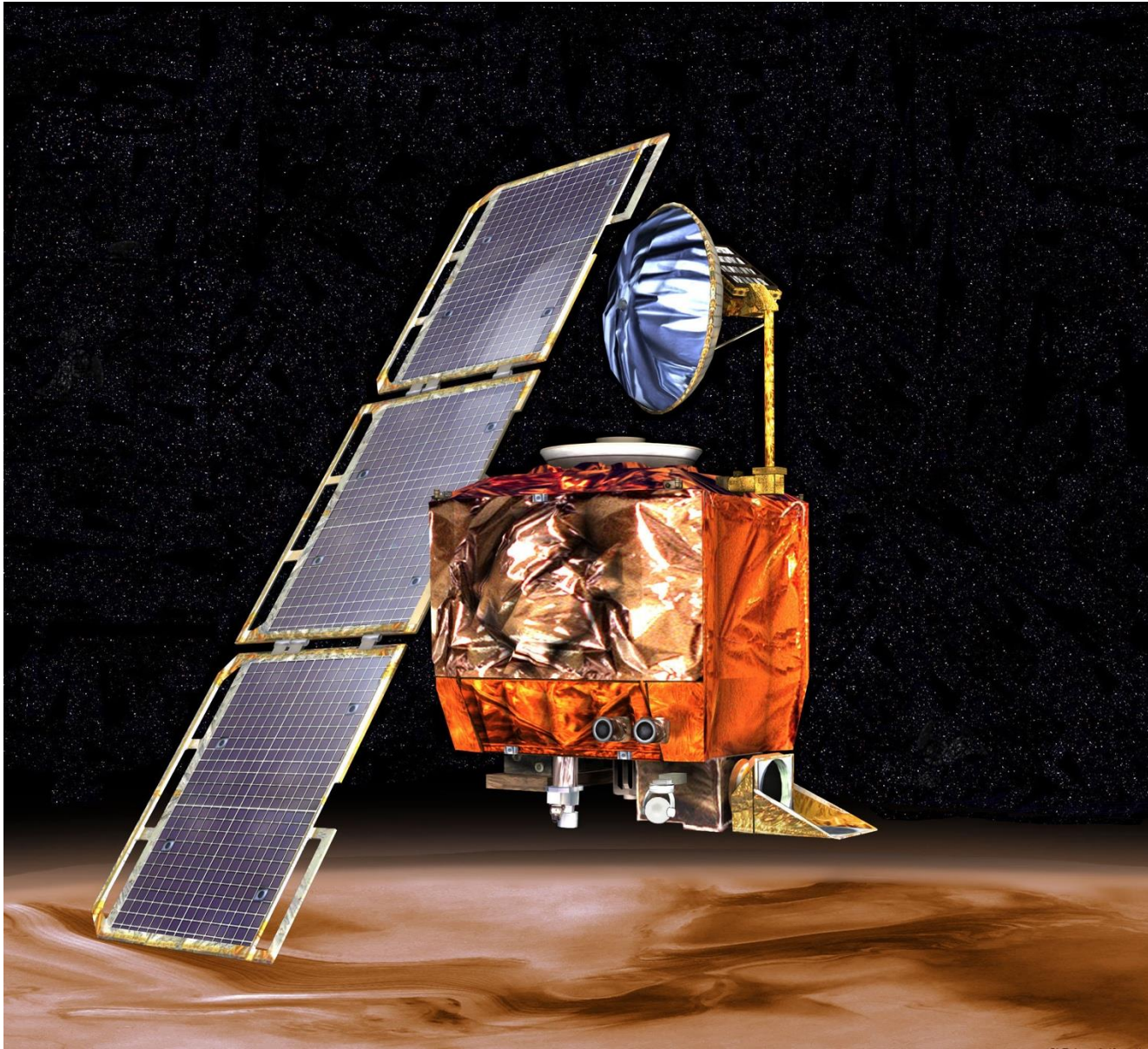
Mars Climate Orbiter
- Destroyed as it entered the atmosphere
- $327.6 million loss

Trajectory Calculation
- A third-party component (by Lockheed Martin) used pound-force/seconds (lbf/s)
- NASA assumed Newton/second (N/s)!
- This discrepancy remained undetected prior to launch

100+ miles discrepancy in actual vs. estimated trajectories

BOTCHED OPERATION

Death By 1,000 Clicks: Where Electronic Health Records Went Wrong

The U.S. government claimed that turning American medical charts into electronic records would make health care better, safer and cheaper. Ten years and $36 billion later, the system is an unholy mess. Inside a digital revolution that took a bad turn.

By **Fred Schulte** and **Erika Fry, Fortune** • MARCH 18, 2019

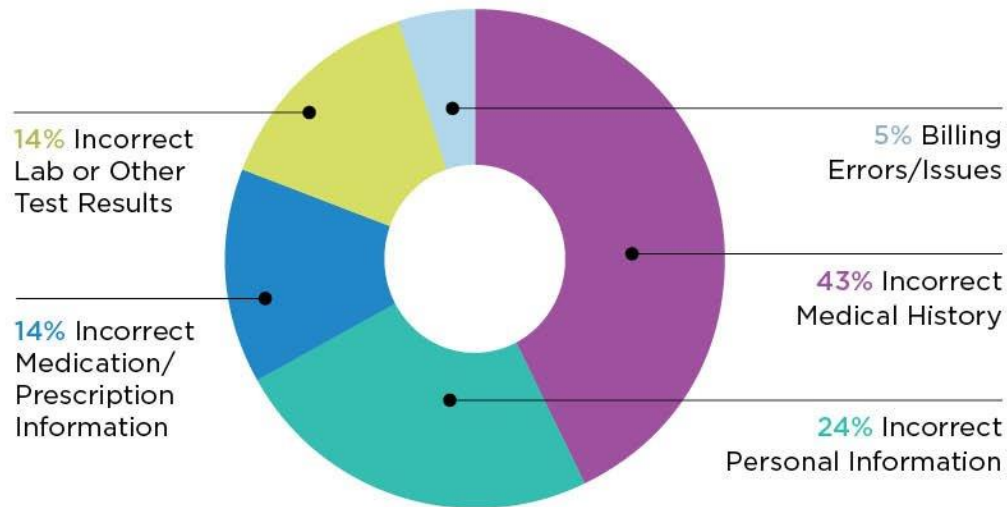https://kffhealthnews.org/news/death-by-a-thousand-clicks/

Broken Records

One in five people surveyed this year by the Kaiser Family Foundation has found a mistake in their EHR. Of those, nearly half have incorrect medical histories.

21%

OF PATIENTS FOUND AN ERROR IN THEIR EHR.

Type of Error Noticed in the Medical Record

14% Incorrect Lab or Other Test Results

14% Incorrect Medication/ Prescription Information

5% Billing Errors/Issues

43% Incorrect Medical History

24% Incorrect Personal Information

Source: Nicolas Rapp/Fortune

Electronic Health Records (EHR)
- Many different EHR systems, but lack of data sharing
- Manual entry by the nurse often required to transfer patient data between hospitals, pharmacies, etc.,
- **Data entry errors**: A major source of medical incidents!

Why Interoperability?

- **Facilitate reuse**: Allow a system to use existing services instead of implementing their own (e.g., authentication, cloud storage, payment services…)

- **Improve usability**: Allow users to bring/transfer their own data from one system to another (e.g., export Google Docs to Microsoft Word)

- **Simplify integration**: Allow independently developed systems to interact without ad-hoc integration effort; reduce the likelihood of errors during integration

Data Formats & Protocols

- Interoperability involves **exchange of information** between systems developed by multiple teams or organizations

- Requires a **shared data format** and a **common protocol**

- **Data format**: How is the data structured?
  - JSON, XML, CSV, YAML,…

- **Protocols**: How is the data sent/received?
  - HTTP/HTTPS: Web-based communication
  - gRPC: Microservices; highly efficient but less general than HTTP
  - MQTT: Messaging for IoT devices
  - SMTP/IMAP/POP3: E-mail clients.
  - …

Data Schema

- Defines the structure, types, and constraints over data elements
- Enables data validation by enforcing the schema & detecting errors in input/output data

```json
{
  "$schema": "http://json-schema.org/draft-07/schema",
  "type": "object",
  "properties": {
    "user_id": { "type": "integer" },
    "name": { "type": "string" },
    "email": { "type": "string", "format": "email" },
    "wage": { "type": "integer", "minimum": 20 }
  },
  "required": ["user_id", "name", "email"]
}
```

Representational State Transfer (REST)

- An approach for designing web applications

- Emphasis on **uniform interfaces**:
  - Standard usage for HTTP methods (GET, POST, PUT, DELETE); status codes for the request outcome
  - Naming convention for URLs, based on *resource identifiers*

https://api.bookstore.com   Base URL

GET https://api.bookstore.com/books/35   Request

{ "id": 35,   Response
  "title": "The Great Gatsby",
  "author": "F. Scott Fitzgerald",
  "year": 1925 }

Before REST

- "The Wild West" of the Web
- Many different protocols, patterns, ad-hoc conventions for APIs
  - e.g., /getUser?id=123, vs. /users/123
  - Inconsistent use of HTTP methods (e.g., POST for everything)
- Clients must adapt to specific individual API styles; no universal standard!

Representational State Transfer (REST)

- An approach for designing web applications
- Emphasis on **uniform interfaces**:
  - Standard usage for HTTP methods (GET, POST, PUT, DELETE); status codes for the request outcome
  - Naming convention for URLs, based on *resource identifiers*
- (Also: Stateless APIs - i.e., no server-side sessions)
- Once widely adopted, REST significantly improved the interoperability of web applications
  - Web apps, mobile apps, IoT devices, etc.,
- Clients can assume that REST APIs are structured the same way; no need for API-specific convention!

# Example: Flight Information System

- **Q. What could go wrong with multiple systems interpreting this data?**

```
<flight>
  <flight_number>BA150</flight_number>
  <departure_date>03/07/2025<departure_date>
  <departure_time>15:30</departure_time>
  <arrival_time>18:45</arrival_time>
  <arrival_date>03/07/2025<arrival_date>
  <departure_airport>LHR</departure_airport>
  <arrival_airport>JFK</arrival_airport>
  <baggage_allowance>23</baggage_allowance>
</flight>
```
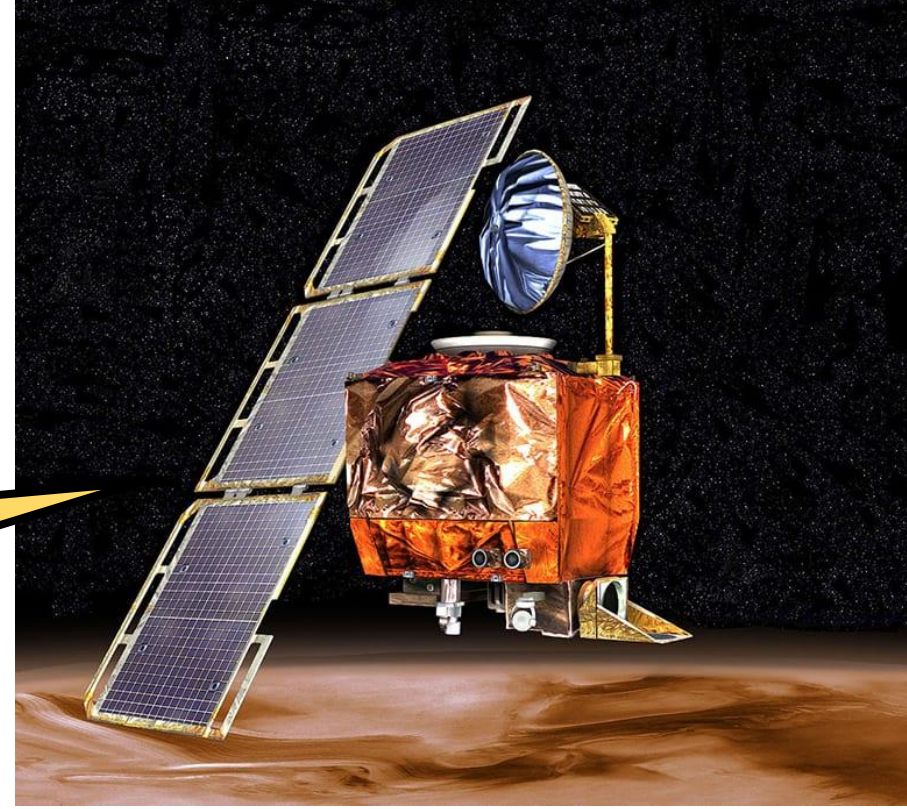
Syntactic vs. Semantic Interoperability

- **Syntactic** interoperability: Multiple systems exchange data over a **shared format** & a **protocol**

- **Semantic** interoperability: Multiple systems exchange and assign a **common interpretation** to data

- In most cases, syntactic interoperability is not enough for intended system functioning; we also want semantic interoperability!

`<plant>`
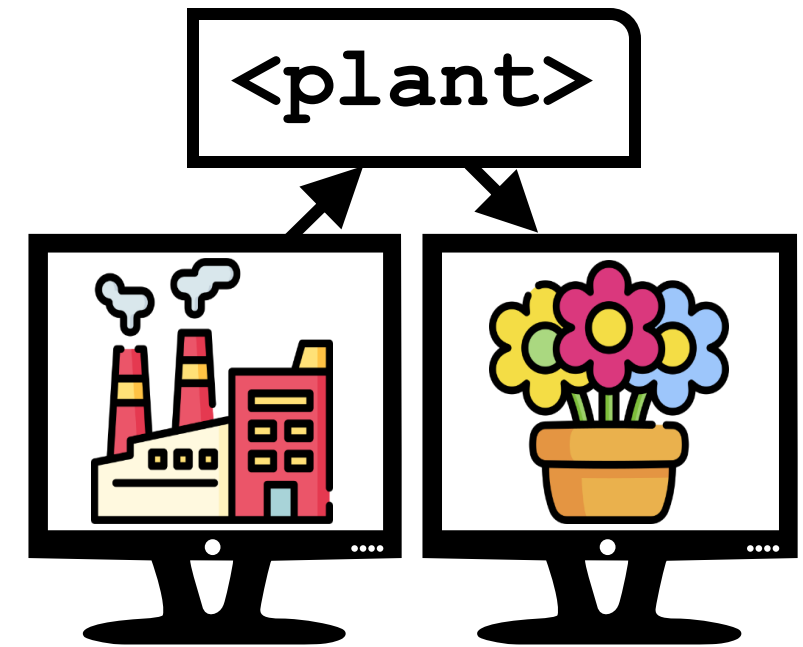
Recall: Mars Climate Orbiter

Spacecraft lost due to lack of *semantic interoperability!*

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)

Semantic Interoperability: Design Principles

- Develop a **shared ontology** of data elements
- Support **backward compatibility**
- Use an existing, **open standard** if possible

Example: Public Transport Dataset

Adrian works for the Transport Agency of MyCity and oversees publishing data about public transport. Adrian wants to publish this data for different types of data consumers such as developers interested on creating applications and for software agents. It is important that both humans and software agents can easily understand and process the data, which should be kept up to date and be easily discoverable on the Web.
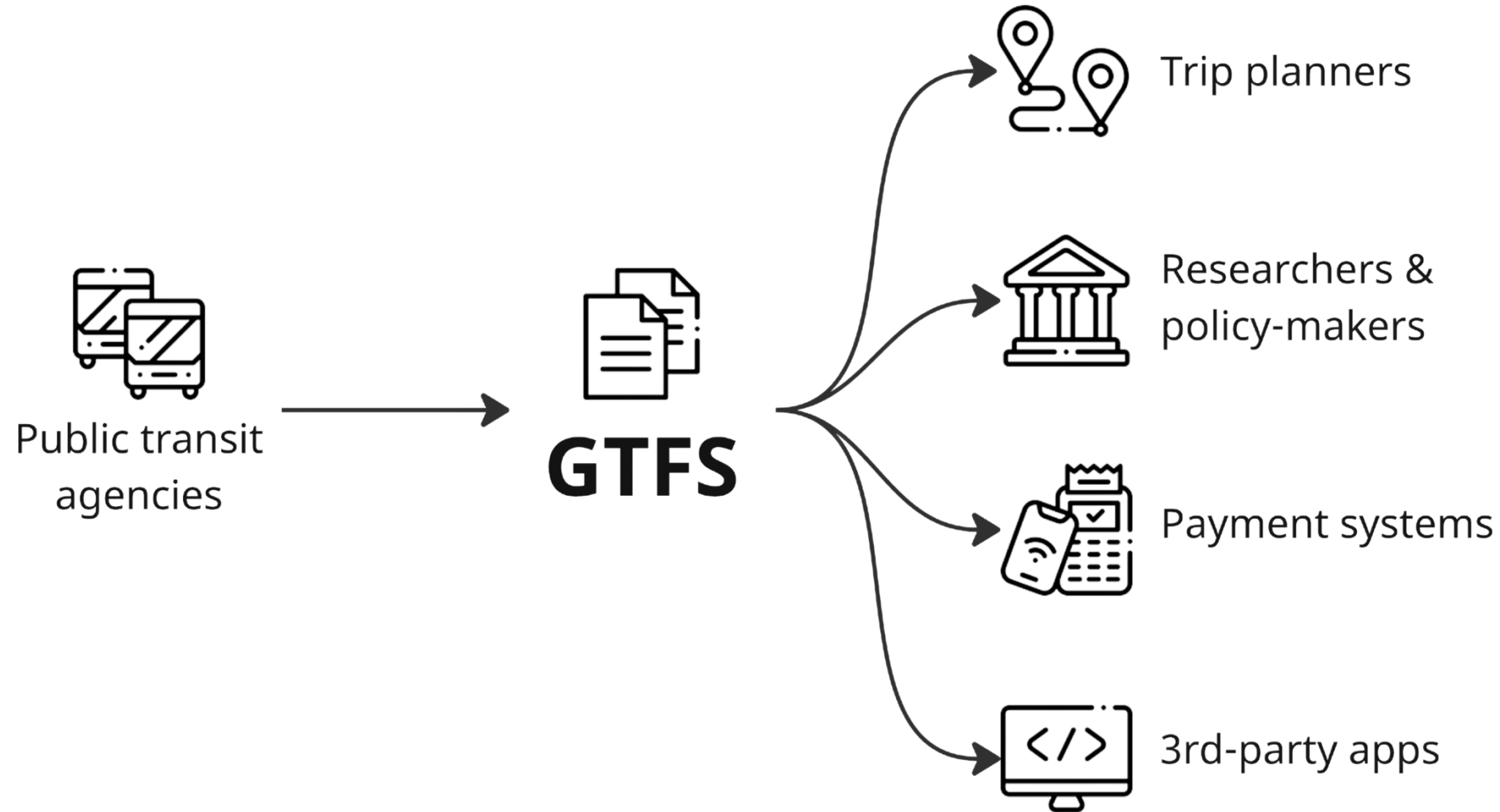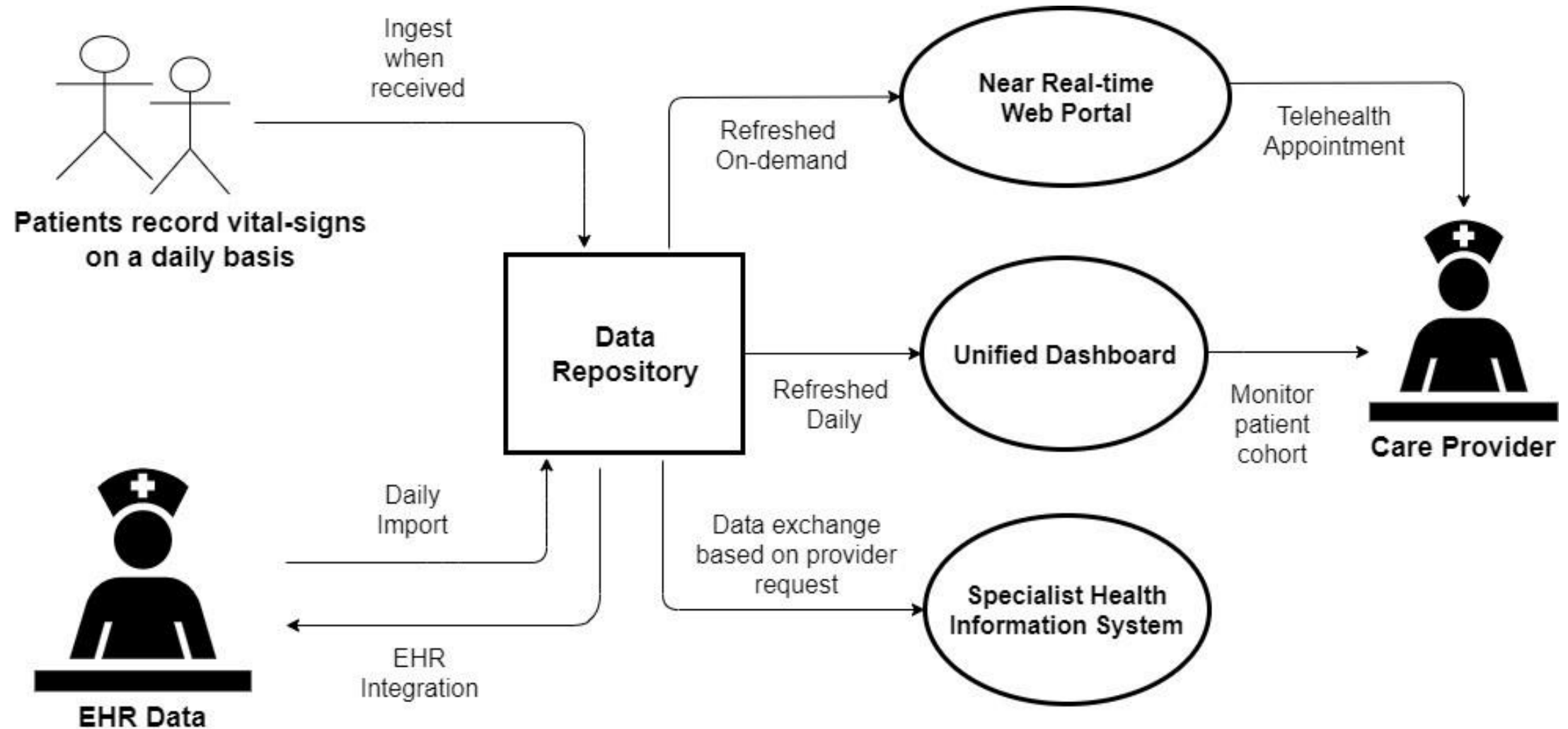
Develop a Shared Data Ontology

- An **ontology** defines concepts, their relationships, and constraints in an application area of interest
  - Sometimes also called a **vocabulary**
- **Example**: Ontology for public transit data
- Consider potential consumers of the ontology (users or applications), use cases, and data needed to support them

Public Transit Data: Who are the Consumers?

Public transit agencies → GTFS → Trip planners, Researchers & policy-makers, Payment systems, 3rd-party apps

# Electronic Health Records: Who are the Consumers?

Develop a Shared Data Ontology

- An **ontology** defines concepts, their relationships, and constraints in an application area of interest
  - Sometimes also called a **vocabulary**
- **Example**: Ontology for public transit data
- Consider potential consumers of the ontology (users or applications) and their purposes
- Maintain a consistent, human-readable naming convention for concepts & relationships
  - **Bad***:* usr_inf (confusing)
  - **Better***:* hasUserInformation (clear & readable)

Develop a Shared Data Ontology

- An **ontology** defines concepts, their relationships, and constraints in an application area of interest
  - Sometimes also called a **vocabulary**
- **Example**: Ontology for public transit data
- Consider potential consumers of the ontology (users or applications) and their purposes
- Maintain a consistent, human-readable naming convention for concepts & relationships
- Use a **data model** to specify and communicate your ontology to others (recall the lecture on design abstractions)
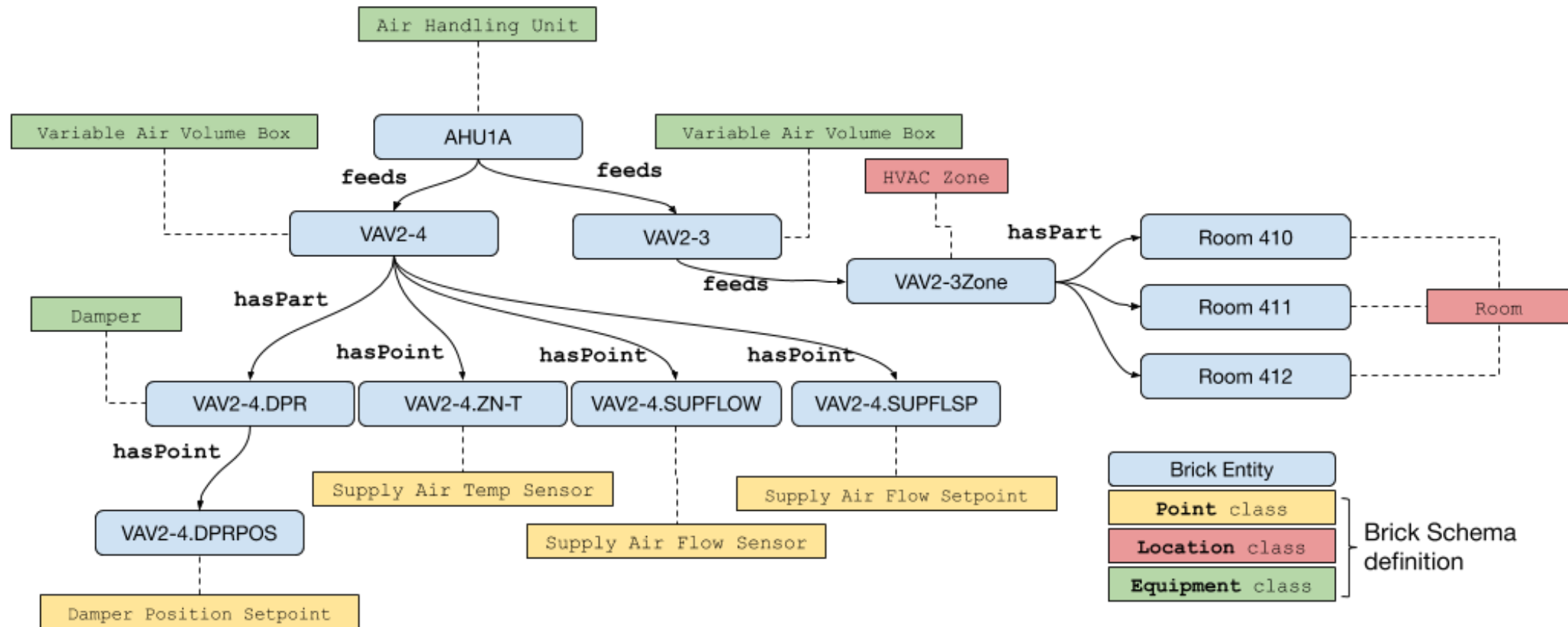
Data Model for Public Transit Data
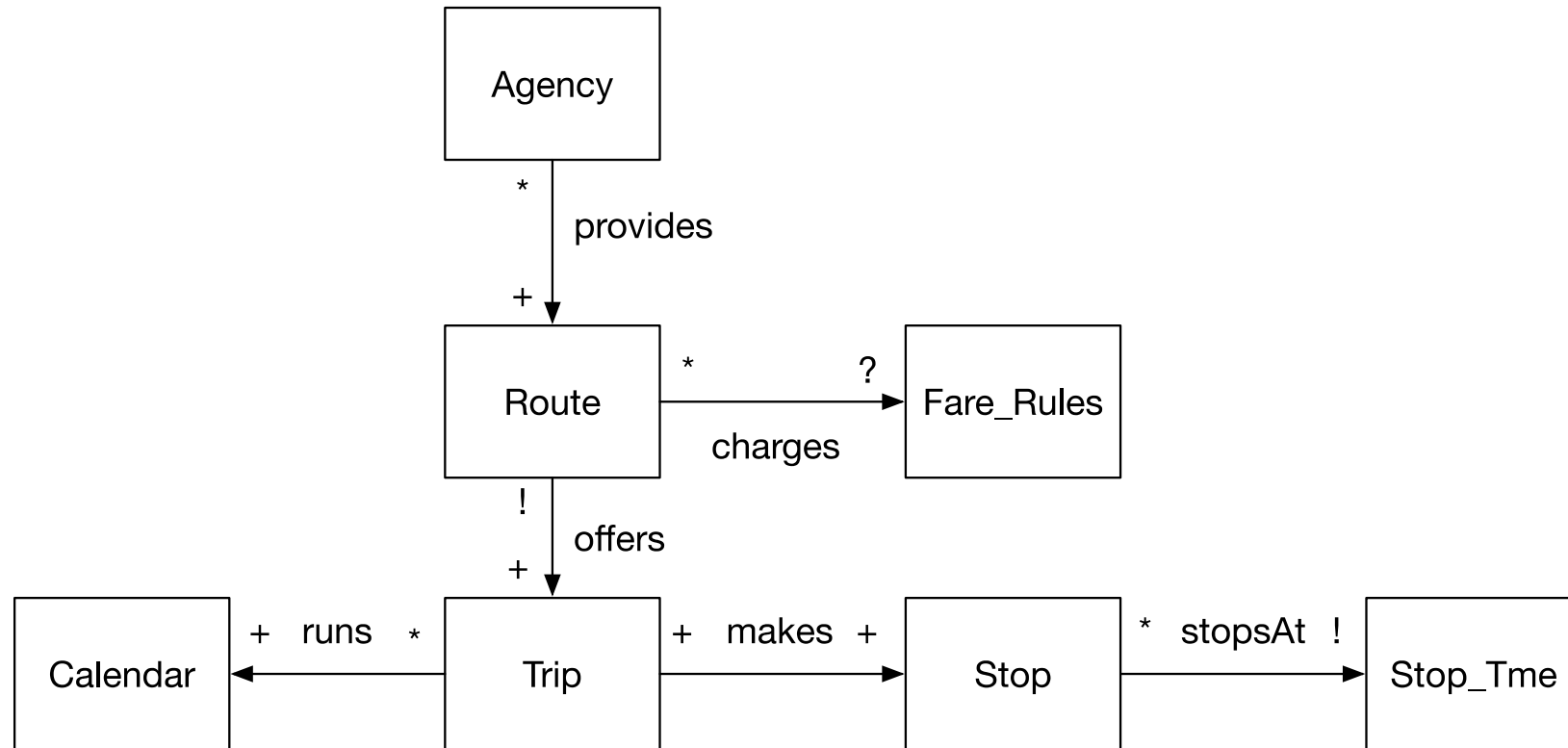
- Augment the data model with a textual description of data elements and relations

Another Example: Brick Ontology for Buildings

https://ontology.brickschema.org/
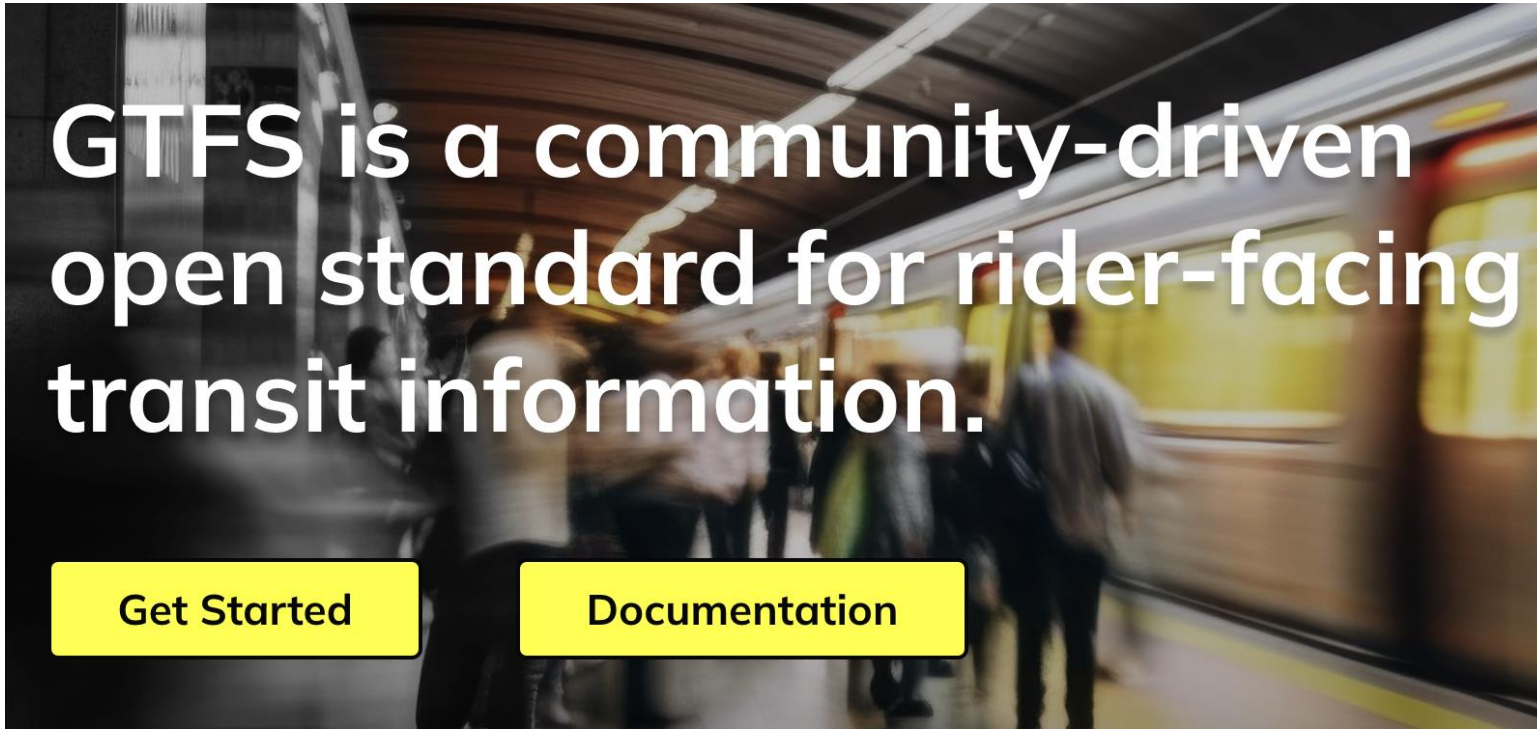
Support Backward Compatibility

- The ontology that you've developed will likely change over time
- **Backward compatibility**: Can the existing systems continue to use your ontology?
- Consider: "Can this change break the client's code?"
  - Adding a new data field – probably OK
  - Removing/renaming, changing the meaning of a field – will likely break!
- Use API versioning to allow clients to transition between versions
  - https://api.bookstore.com/books -> /books/v1/…, /books/v2/…
- **Deprecate** instead of removing data

```
HTTP/1.1 200 OK
Deprecation: true
Warning: "The 'username' field will be removed in API v3.0. Use 'email' instead."
```

Use an Existing Open Standard

GTFS is a community-driven open standard for rider-facing transit information.

Get Started   **Documentation**

https://gtfs.org/

- If available, adapt a well-established, open standard
- Do not re-invent the wheel! It will cause more integration work later for your team & others

Interoperability vs. Changeability

- **Q. What is the relationship between interoperability vs. changeability?**

Interoperability vs. Changeability

- **Recall**: Interface segregation principle
  - *An interface should not force clients to depend on unnecessary details*
- *Interface pollution* is a common risk of interoperability
  - To be interoperable, a data schema/ontology may include more data elements than needed by a single system
  - Tends to result in a bloated ontology; multiple, *partial* implementations of the schema (e.g., Google Transit implementation of GTFS)
- Another risk: Increased dependencies between systems
  - If a data schema/ontology changes, all systems that depend on it may be forced to change
  - Supporting backward compatibility is crucial for changeability!

Interoperability: Takeaways

- Interoperability allows multiple systems to communicate and integrate with each other

- Syntactic interoperability is the bare minimum; semantic interoperability is often what is needed

- Interoperability can negatively impact changeability

- Not all systems may need to be interoperable! Like other qualities, consider how crucial interoperability is to the successful of your product (recall: **risk-driven design**!)

Team Project

- In a future milestone, you will develop a service that will be used by multiple applications

- You will be asked to design an interoperable API with a well-defined data ontology

- We will come back to the topic of interoperability in a few weeks!

Summary

- Exit ticket!