# 17-423/723: Software System Design

## Design for Interoperability

Feb 18, 2026

# 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

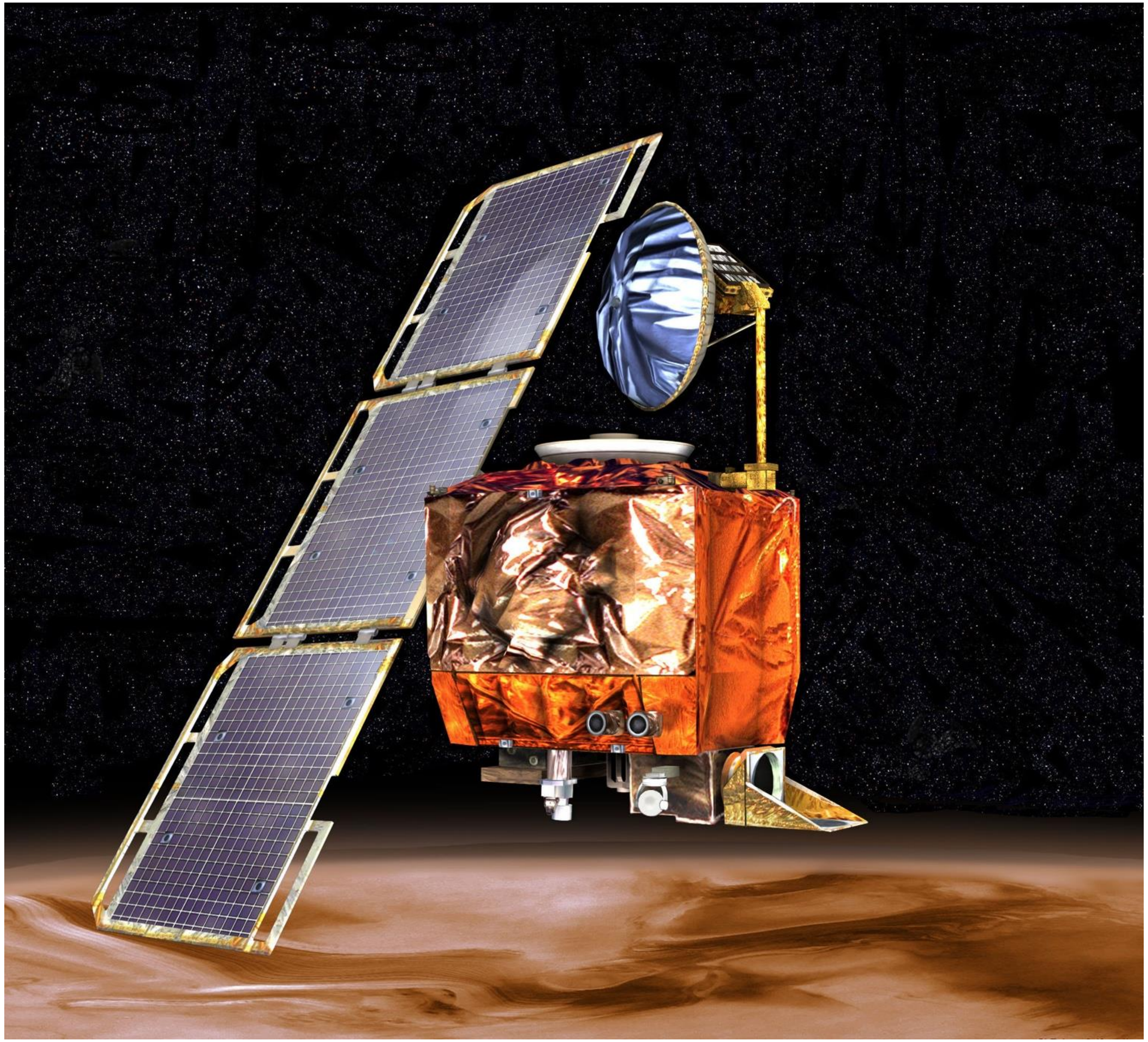
Content partly based on a lecture by Tobias Dürschmid

# 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)

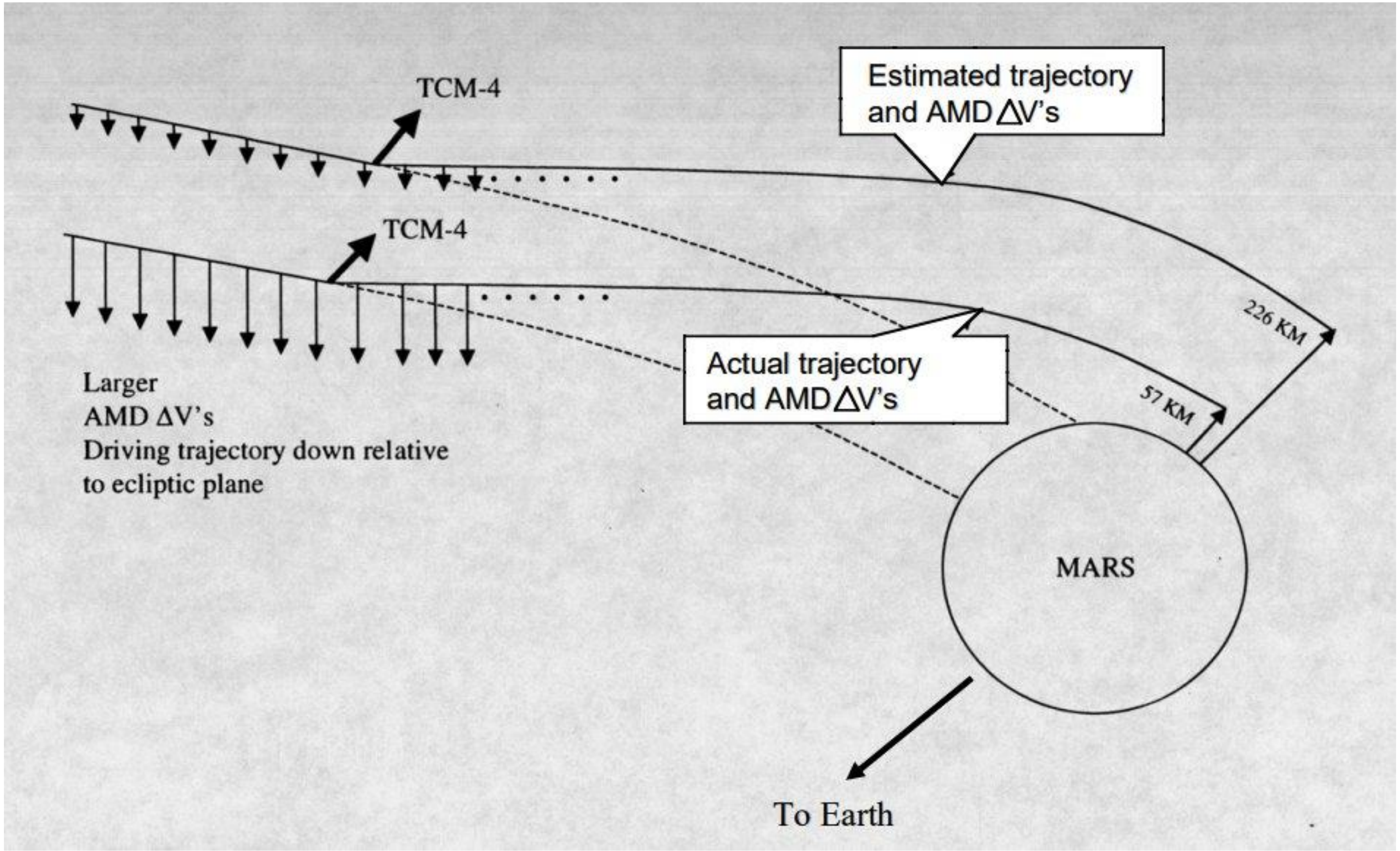


## Mars Climate Orbiter

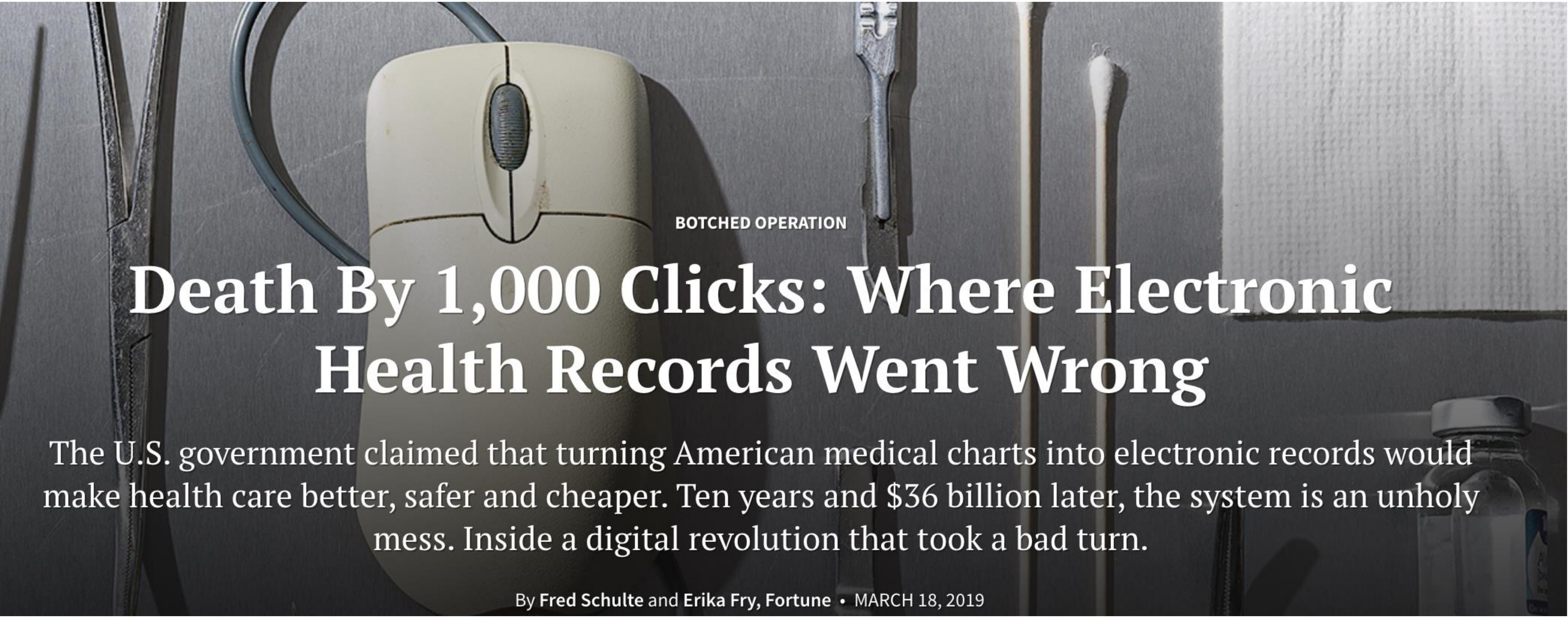
- Launched on Dec 1998, lost on Sept 1999
- Destroyed as it entered the atmosphere
- $327.6 million loss

## 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


TCM-4
Estimated trajectory and AMD ΔV's
TCM-4
226 KM
Actual trajectory and AMD ΔV's
57 KM
Larger AMD ΔV's Driving trajectory down relative to ecliptic plane
MARS
To Earth

100+ miles discrepancy in actual vs. estimated trajectories



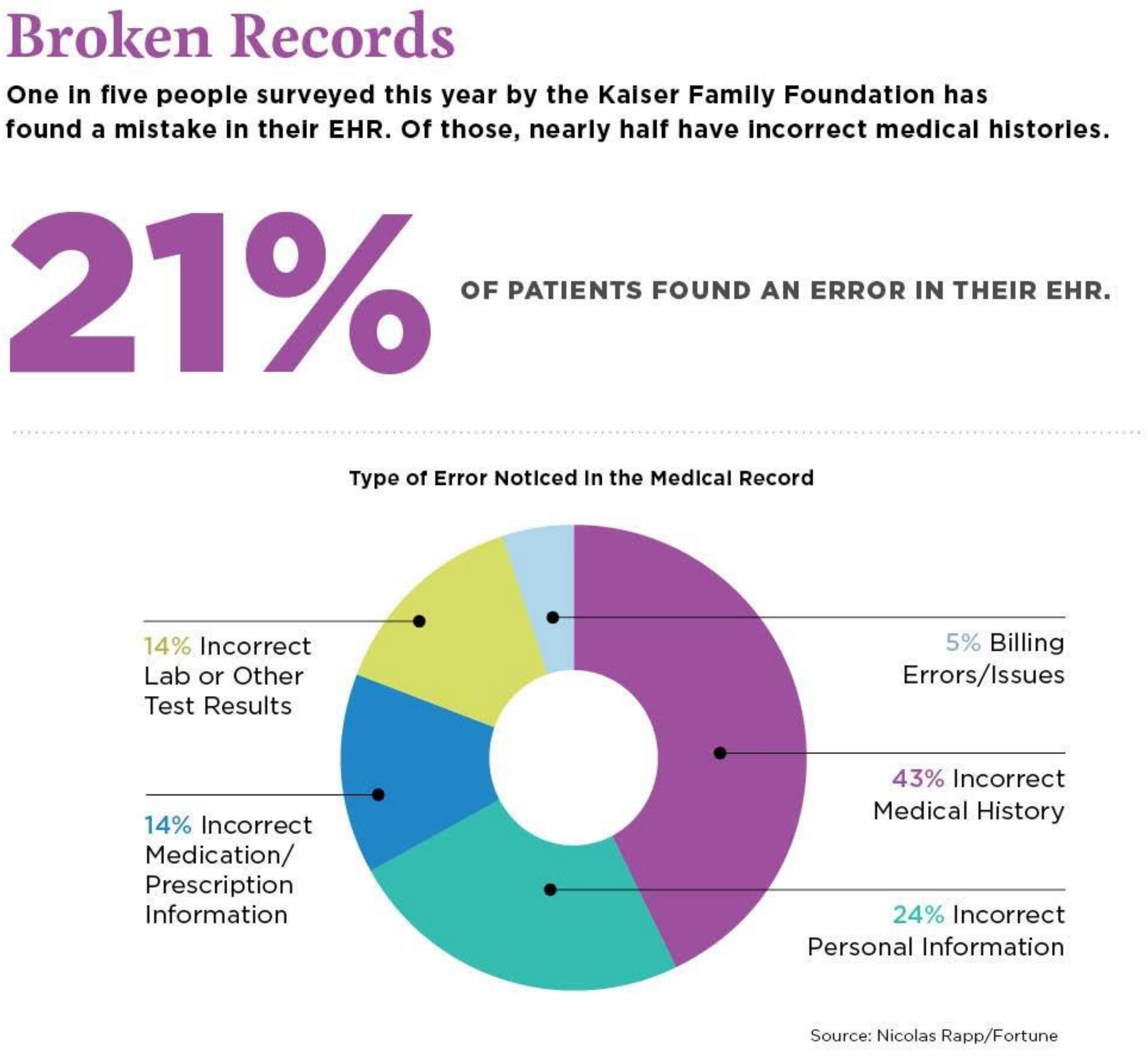
BOTCHED OPERATION

# 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

Source: Nicolas Rapp/Fortune

## Electronic Health Records (EHR)

- Many different EHR systems, but lack common data format and integration platform
- Manual entry by the nurse is 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

# Interoperability QA: Examples

- When the internal CRM system sends a customer update, the Salesforce connector must consume and process the data within 500 ms
- The Chemical Tracking System must be able to import any chemical structure encoded using the **SMILES** notation with 100% accuracy
- Every 60 minutes, the Payroll System must synchronize employee records with the Cost Accounting software, ensuring zero data loss during transmission
- Version 2.0 of the flight booking API must support requests from clients using Version 1.0 for at least 12 months, allowing for phased upgrade

# 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

```
{
  "$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"]
}
```

# Example: Representational State Transfer (REST)

- An approach for designing web APIs
- 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

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

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

Base URL

Request

Response

# 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!

Welcome to Netscape!
File Edit View Go Bookmarks Options Directory Help
Back Forward Home Reload Images Open Find Stop
Location:
Guided Tour What's New Questions Net Search Net Directory Newsgroups
WELCOME TO NETSCAPE

http://home.mcom.com/home/welcome.html

## Welcome to Netscape

You have just embarked on a journey across the Internet, and Netscape is your vehicle. This welcome page will help you get started on your use of Netscape and your exploration of the Internet.

Document: Received 29472 of 29646 bytes

# 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!

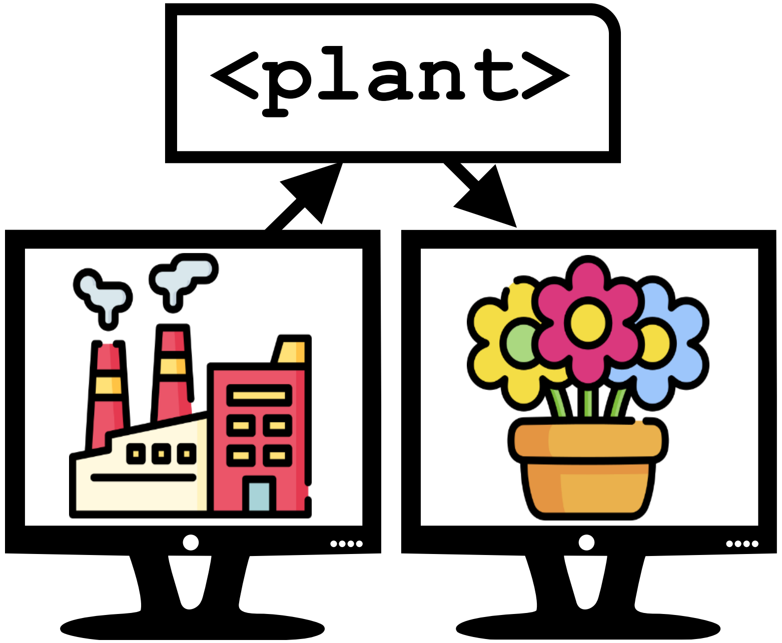
# Example: Package Delivery System

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

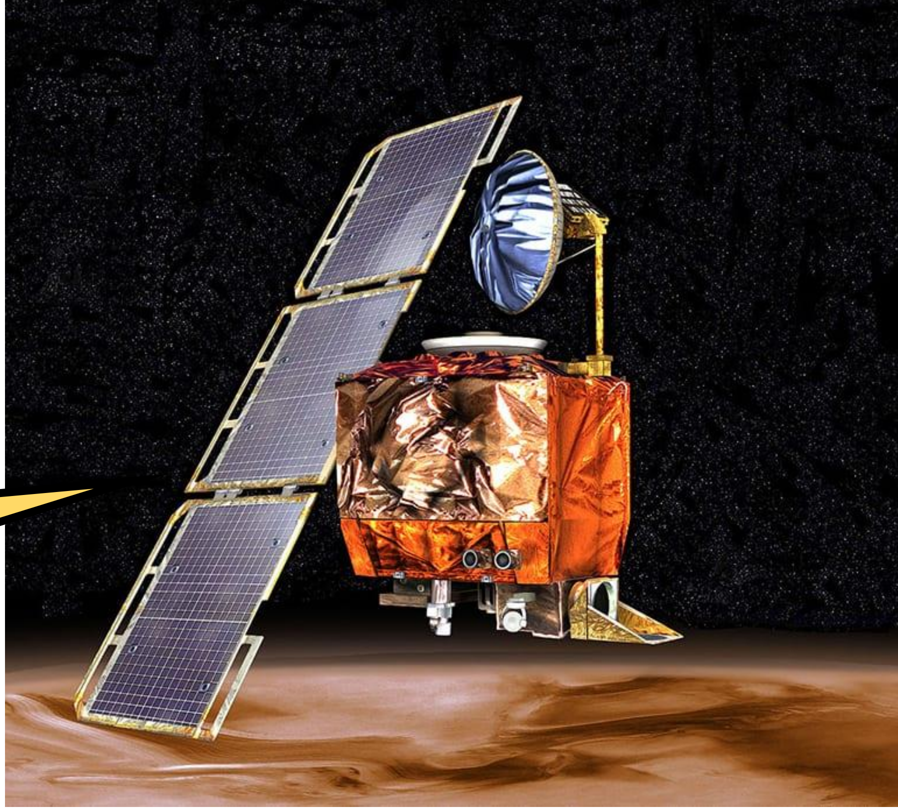
```
<shipment>
<tracking_number>1Z999AA10123456784</tracking_number>
  <weight>5.2</weight>
  <delivery_date>03/07/2025</delivery_date>
  <delivery_time>14:30</delivery_time>
  <recipient>John Smith, Gates 5000</recipient>
</shipment>
```

# 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** (i.e., meaning) to data
- In most cases, syntactic interoperability is not enough for intended system functioning; we also want semantic interoperability!



# 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**
- Handle **non-conformant inputs**
- Use an existing, **open standard** if possible

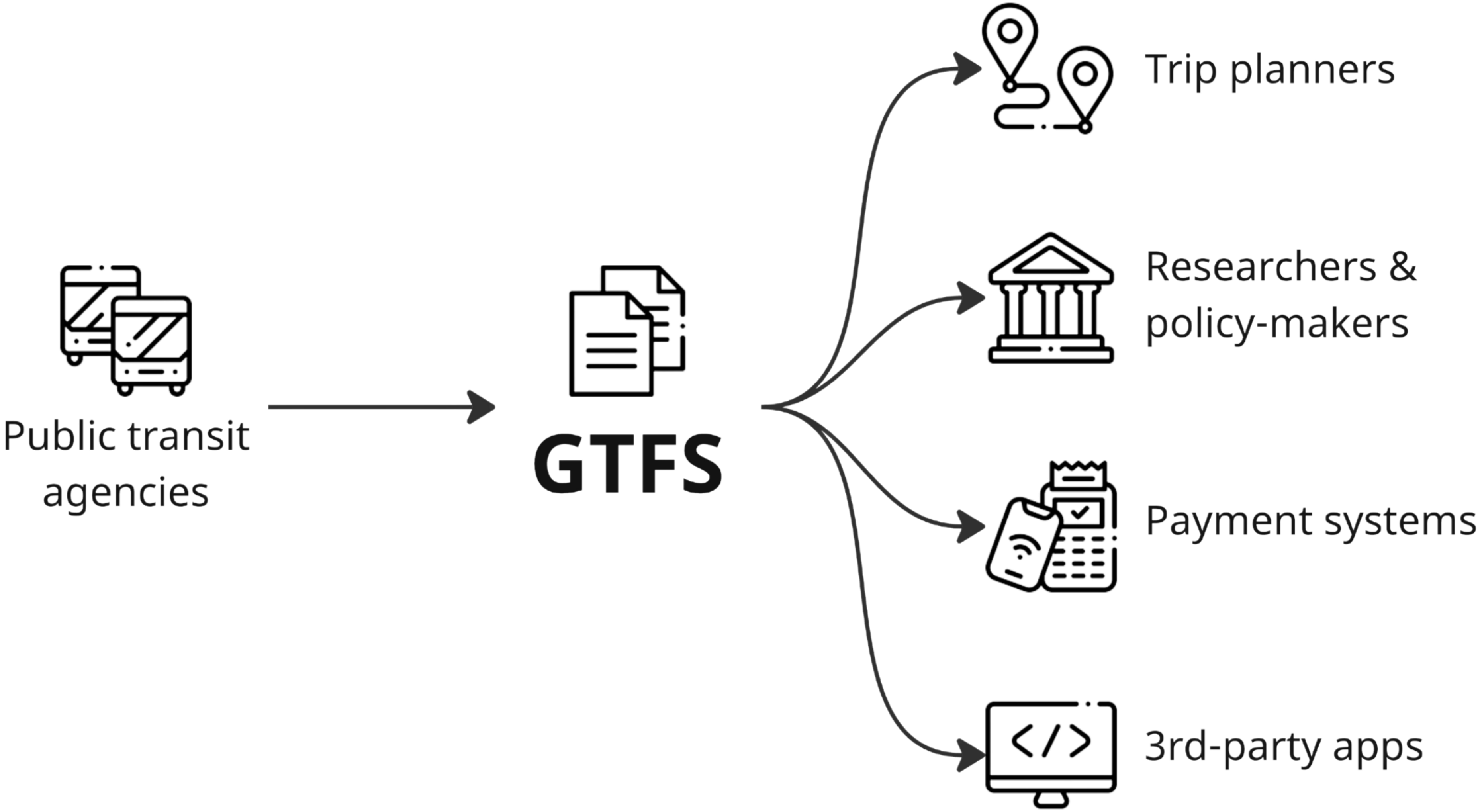
# Example: Public Transport Dataset

*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
  - Also called a **vocabulary**
- **Example**: Ontology for public transit data
- Consider potential consumers of the ontology (users, components or external applications), use cases, and data needed to support them

# Public Transit Data: Who are the Consumers?



Public transit agencies

GTFS

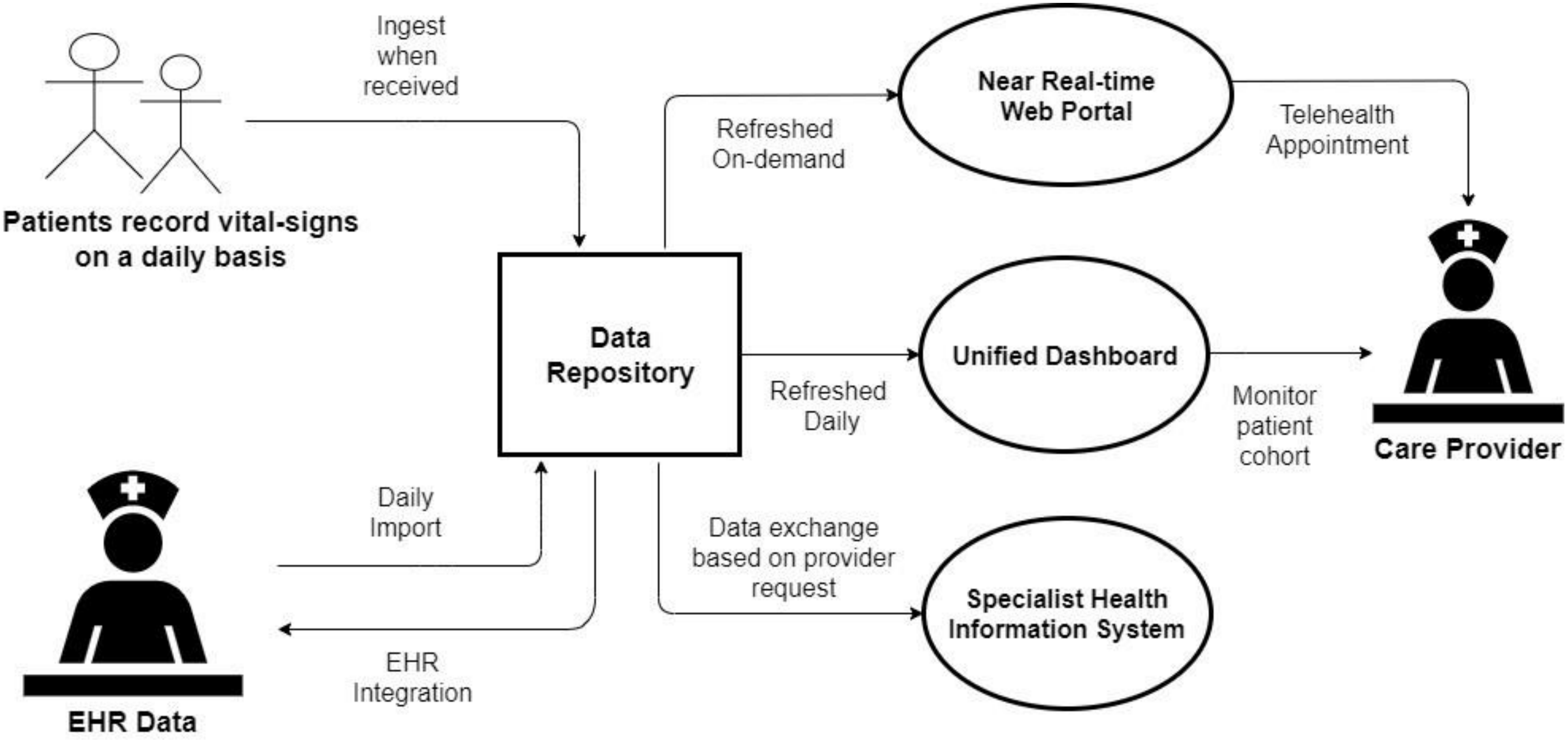
Trip planners

Researchers & policy-makers

Payment systems

3rd-party apps

# Electronic Health Records: Who are the Consumers?



Patients record vital-signs on a daily basis

Ingest when received

Data Repository

Refreshed On-demand

Near Real-time Web Portal

Telehealth Appointment

Refreshed Daily

Unified Dashboard

Monitor patient cohort

Care Provider

Data exchange based on provider request

Specialist Health Information System

Daily Import

EHR Integration

EHR Data

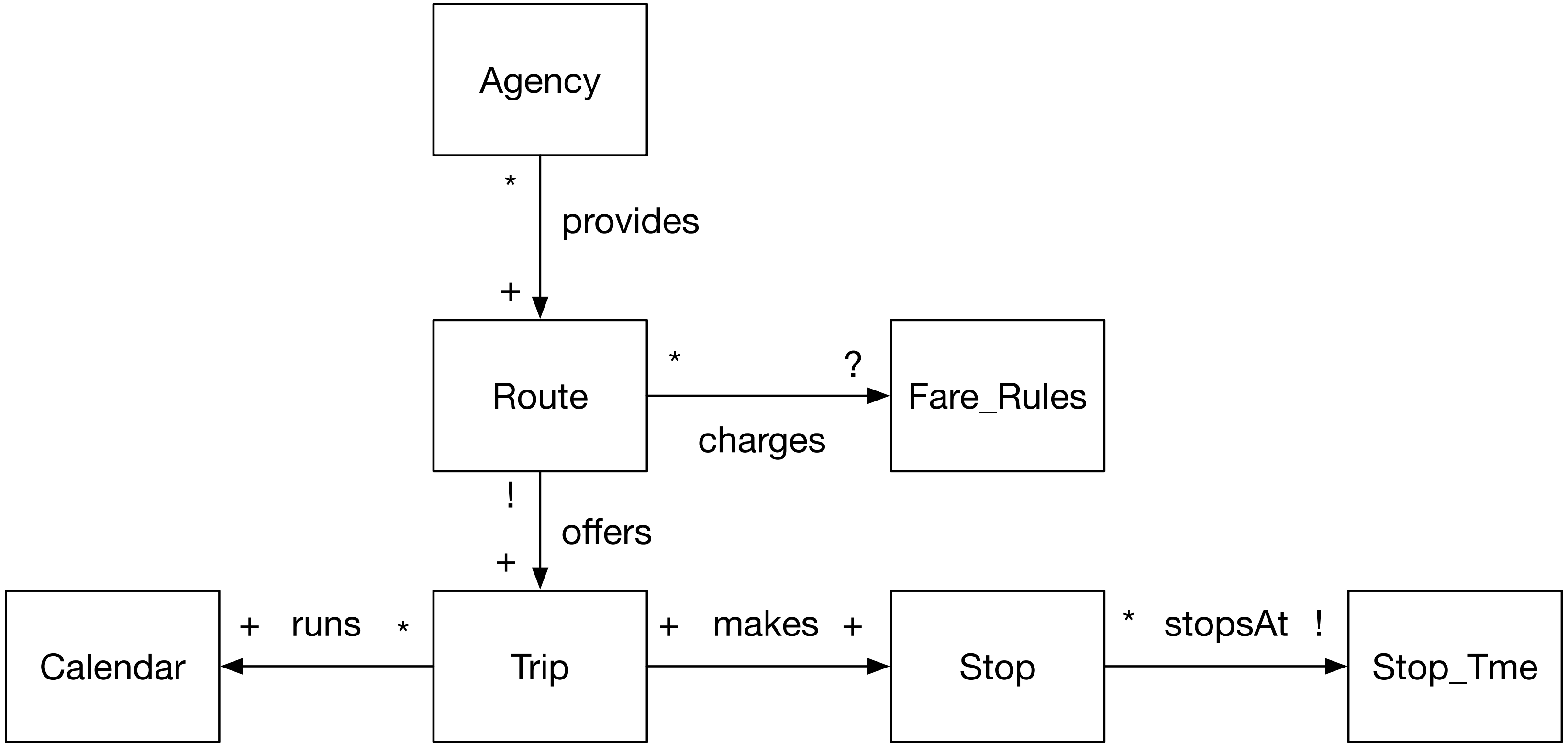
# Develop a Shared Data Ontology

- An **ontology** defines concepts, their relationships, and constraints in an application area of interest
  - Also called a **vocabulary**
- **Example**: Ontology for public transit data
- Consider potential consumers of the ontology (users, components or external 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
  - Also called a **vocabulary**
- **Example**: Ontology for public transit data
- Consider potential consumers of the ontology (users, components or external 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



- The data model should be accompanied with a textual description of data types and relations

# Describing the Meaning of a Data Model

**Data Type:** Trip ✗

A trip on a route that a vehicle follows.

**Data Type:** Trip ✓

A single vehicle following a specific route at a specific time with a specific sequence of stops. Each trip has a unique identifier and represents one execution of a route. For example, the 61C bus departing Forbes & Murray at 9:15 AM is one trip; the same bus at 9:45 AM is a different trip.

# Describing the Meaning of a Data Model

**Relation:** Route **offers** Trip ✓

A route offers one or more trips available throughout the day. Each trip is a specific instance of a vehicle operating on that route at a particular time.

**Constraints:**

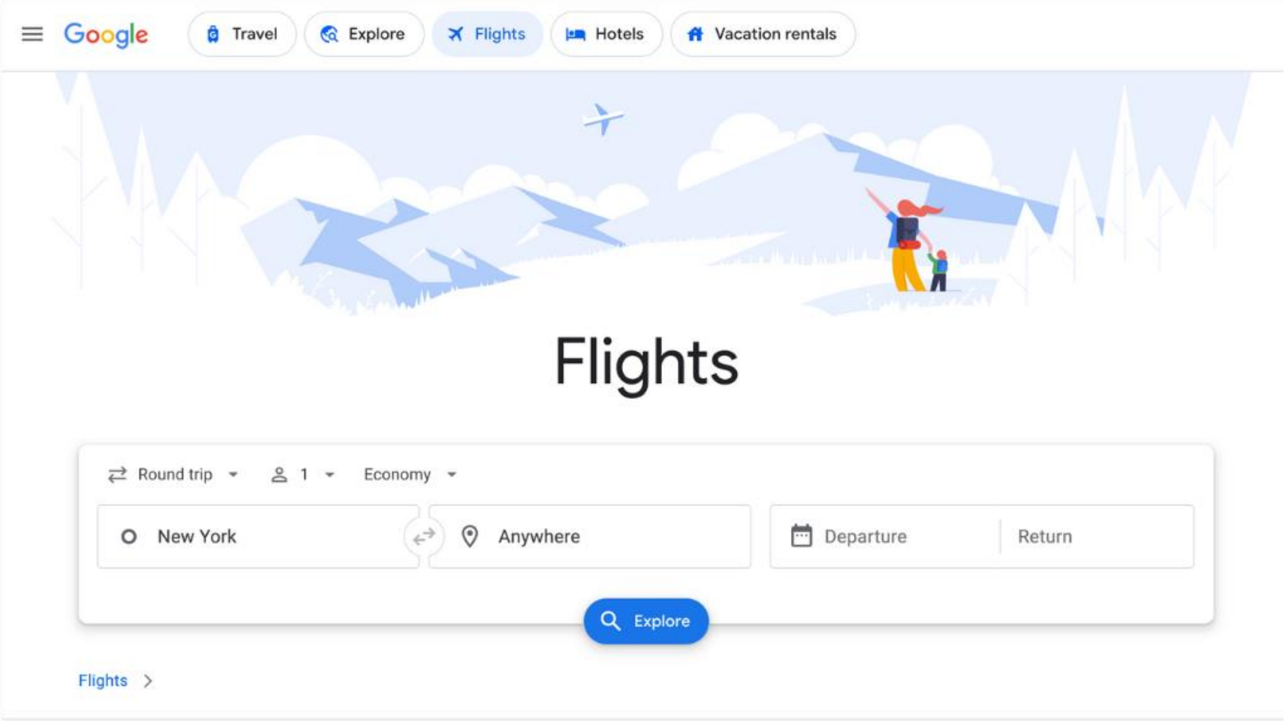
- Each route must offer at least one trip; a route typically offers many trips per day.
- Every trip must be associated with exactly one route; a trip cannot operate on multiple routes simultaneously.
- Trips for the same route may have different stop sequences on different days (e.g., weekend service skips some stops).

**Examples**:

- Route "61C" offers 72 trips on a typical weekday
- Route "Night Owl" offers 4 trips per night (11 PM, 1 AM, 3 AM, 5 AM)

# Exercise: Ontology for Flight Reservations

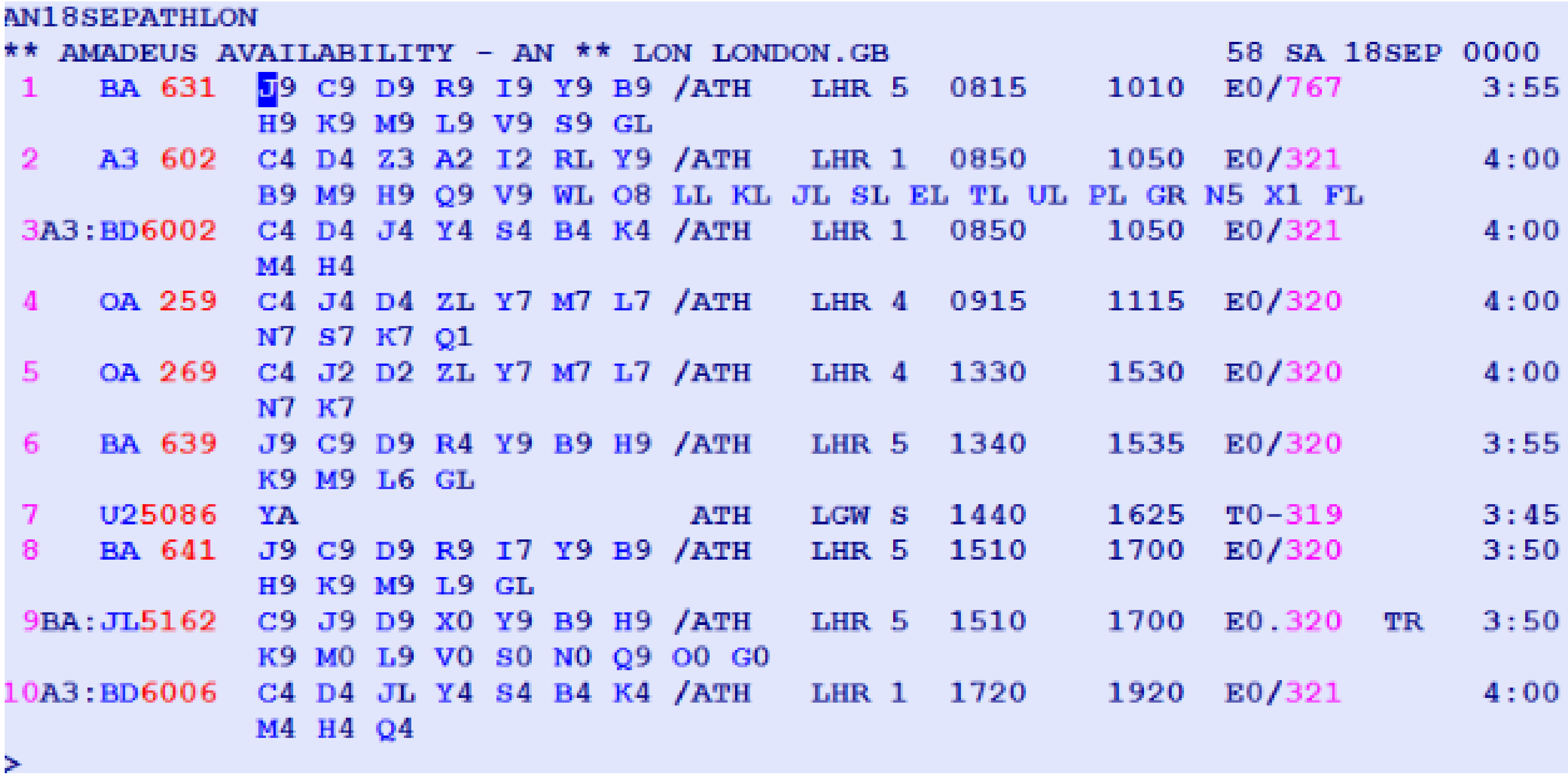
- Suppose that you are responsible for designing a global flight reservation system that enables clients to search, book, and modify flights across multiple airlines.



# Exercise: Ontology for Flight Reservations

- Suppose that you are responsible for designing a global flight reservation system that enables clients to search, book, and modify flights across multiple airlines.
- Design an ontology to support the system. Consider:
  1. Who are potential consumers of the ontology?
  2. What are the key elements of the ontology (e.g., data types and relationships between them)?
  3. What are possible semantic interoperability issues (e.g., ambiguities)?
- Sketch a data model that represents the ontology

# Global Distribution System (GDS)



```
AN18SEPATHLON
** AMADEUS AVAILABILITY - AN ** LON LONDON.GB             58 SA 18SEP 0000
 1    BA 631   J9 C9 D9 R9 I9 Y9 B9 /ATH   LHR 5  0815    1010  E0/767        3:55
               H9 K9 M9 L9 V9 S9 GL
 2    A3 602   C4 D4 Z3 A2 I2 RL Y9 /ATH   LHR 1  0850    1050  E0/321        4:00
               B9 M9 H9 Q9 V9 WL O8 LL KL JL SL EL TL UL PL GR N5 X1 FL
 3A3:BD6002   C4 D4 J4 Y4 S4 B4 K4 /ATH   LHR 1  0850    1050  E0/321        4:00
               M4 H4
 4    OA 259   C4 J4 D4 ZL Y7 M7 L7 /ATH   LHR 4  0915    1115  E0/320        4:00
               N7 S7 K7 Q1
 5    OA 269   C4 J2 D2 ZL Y7 M7 L7 /ATH   LHR 4  1330    1530  E0/320        4:00
               N7 K7
 6    BA 639   J9 C9 D9 R4 Y9 B9 H9 /ATH   LHR 5  1340    1535  E0/320        3:55
               K9 M9 L6 GL
 7    U25086   YA                    ATH   LGW S  1440    1625  T0-319        3:45
 8    BA 641   J9 C9 D9 R9 I7 Y9 B9 /ATH   LHR 5  1510    1700  E0/320        3:50
               H9 K9 M9 L9 GL
 9BA:JL5162   C9 J9 D9 X0 Y9 B9 H9 /ATH   LHR 5  1510    1700  E0.320   TR   3:50
               K9 M0 L9 V0 S0 N0 Q9 O0 G0
10A3:BD6006   C4 D4 JL Y4 S4 B4 K4 /ATH   LHR 1  1720    1920  E0/321        4:00
               M4 H4 Q4
>
```

- Originally developed in 1960s; still used by most airlines and travel agents for flight transactions

# Global Distribution System (GDS)

WENDOVER
01 WENDOVER/SAM*
LAST TKT DTE 08FEB23 - DATE OF ORIGIN
AL FLGT BK DATE TIME
How Bad Technology Holds Airlines Back

Link to article

*For example, Air New Zealand's SkyCouch product—transformable economy seats into lie-flat beds—cannot be adequately managed through the GDS. These seats appear perpetually occupied, forcing customers to book directly through Air New Zealand's website.*

# Support Backward Compatibility

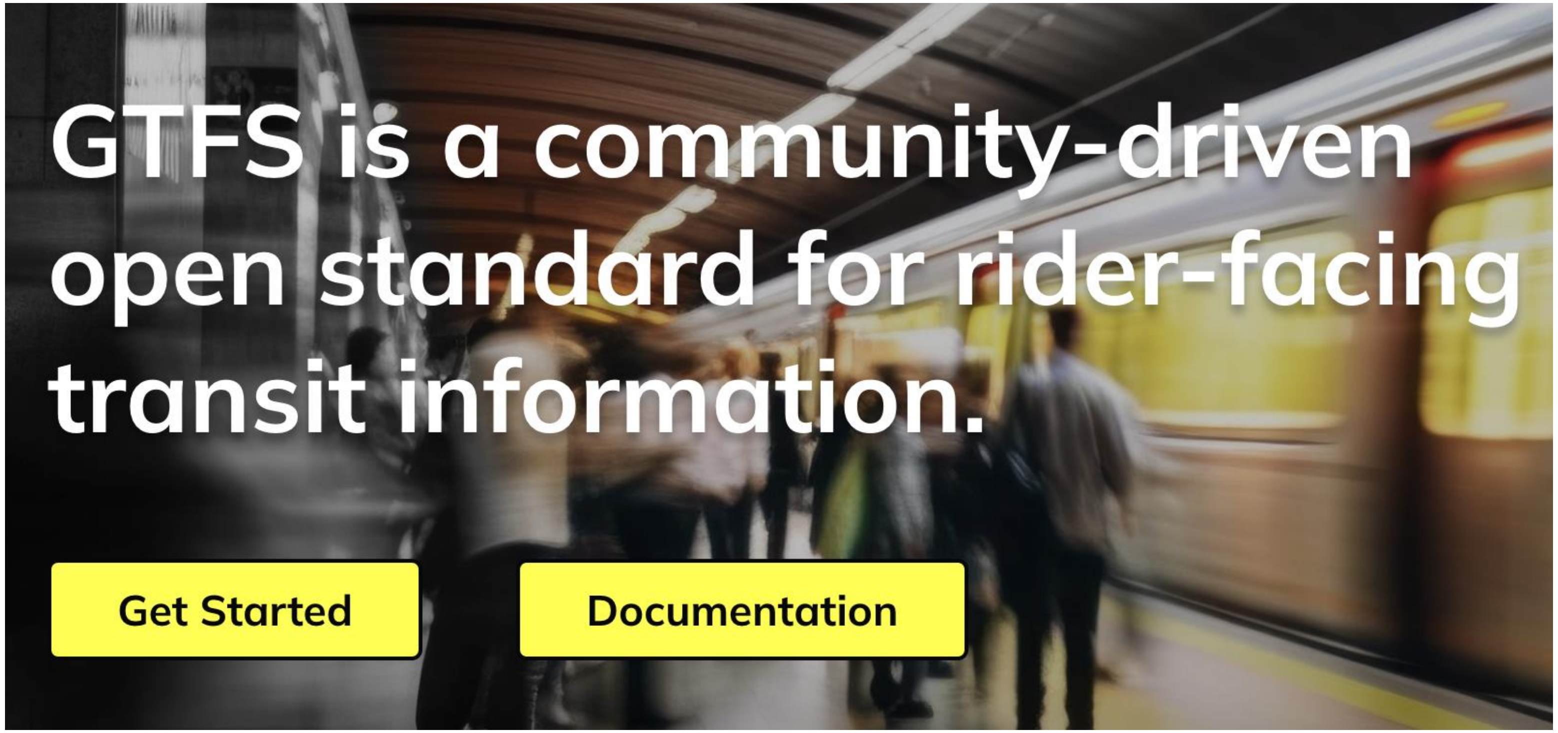
- The ontology that you've developed will likely change over time
- **Backward compatibility**: Can the existing systems continue to use your ontology?
- Consider: "Can this change break the client's code?"
  - Adding a new data field – probably OK
  - Removing/renaming, changing the meaning of a field – 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."
```

# Handle Non-Conformant Inputs

- Postel’s Law: *“Be conservative in what you do, be liberal in what you accept from others”* (Jon Postel, 1981)
    - Also called the Robustness Principle
- When sending data to others, strictly follow the specification
- When accepting data from others, be tolerant of:
    - Missing or unknown values for optional fields
    - Additional metadata
    - Different ordering of fields
    - Minor deviations from expected values
        - Examples: “USD ” instead of “USD” (extra white spaces), “canceled” instead of “cancelled”, etc.,
- **Q. What are benefits & downsides with this approach?**

# Use an Existing Open Standard



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

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!