A Middleware for Automatic Composition and Mediation in IoT Systems

Abdessalam Elhabbash, Yehia Elkhatib, Georgios Bouloukakis, Maria Salama

Lancaster University

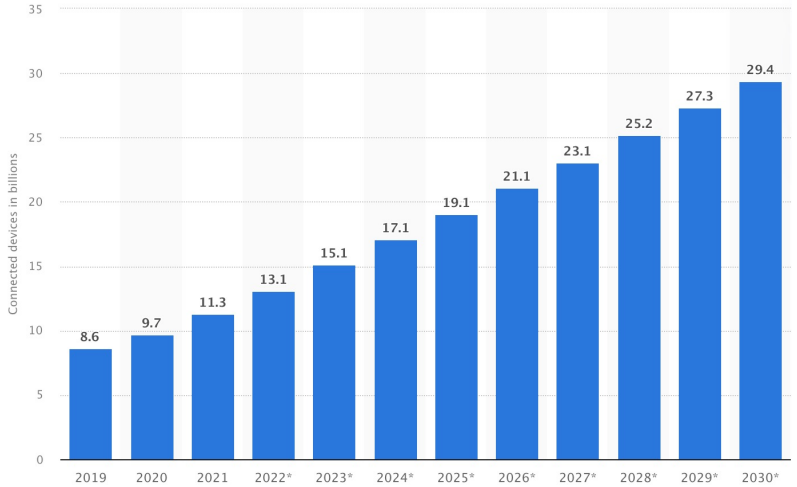
University of Glasgow

TELECOM SudParis

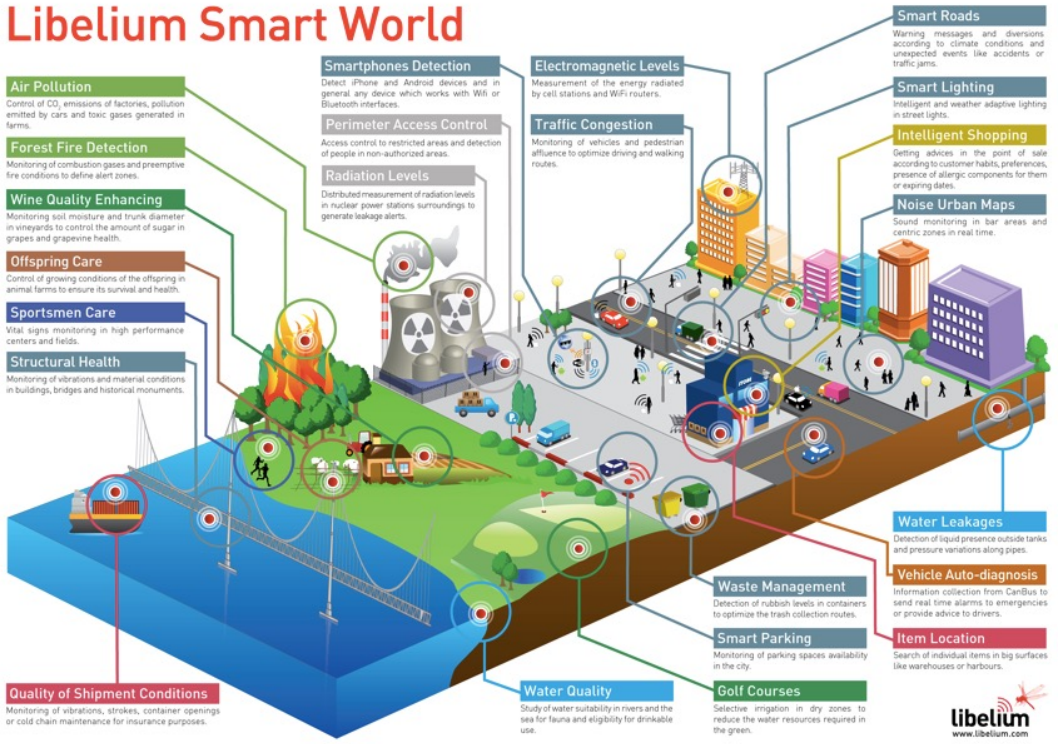
ABC
ADAPTIVE BROKERAGE FOR THE CLOUD

The Rise in IoT Uptake

- Technological advances
  - system-on-chip design and manufacturing
  - devices are highly available, affordable, small, power-efficient
  - network connectivity: LTE, 5G, WiFi, Zigbee, LoRaWAN
- Great progress in deploying applications over various fabrics
  - in situ sensing and actuating devices
- Lots of interesting verticals



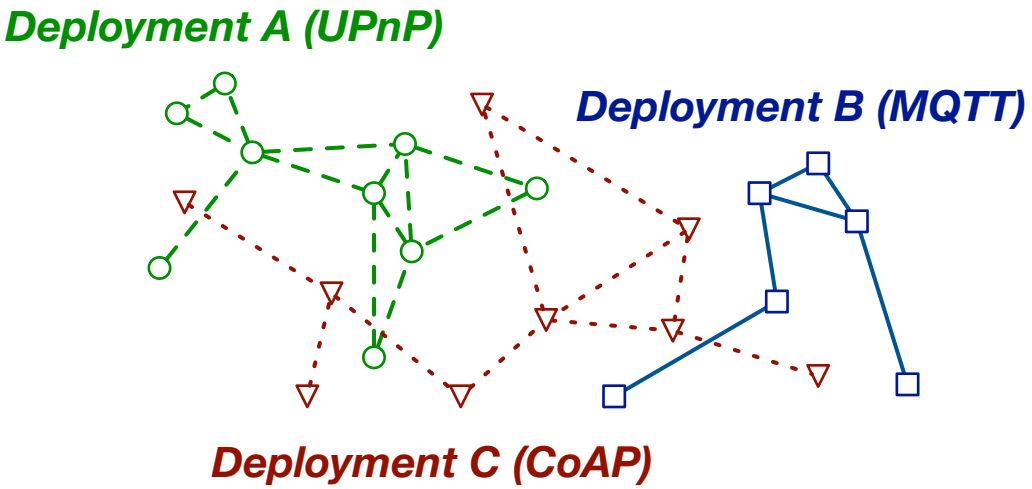
https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/



https://www.libelium.com/libeliumworld/top_50_iot_sensor_applications_ranking/

Problem Statement

- A variety of constituent systems
  - independently deployed
  - use proprietary APIs + data formats
  - co-exist in shared physical spaces
- Challenge:
  - heterogeneity makes it difficult to design, maintain and adapt *integrated IoT systems*
  - system developers are overwhelmed with the amount of knowledge they need to acquire
  - difficult to future-proof applications for opportunistic interaction with other systems

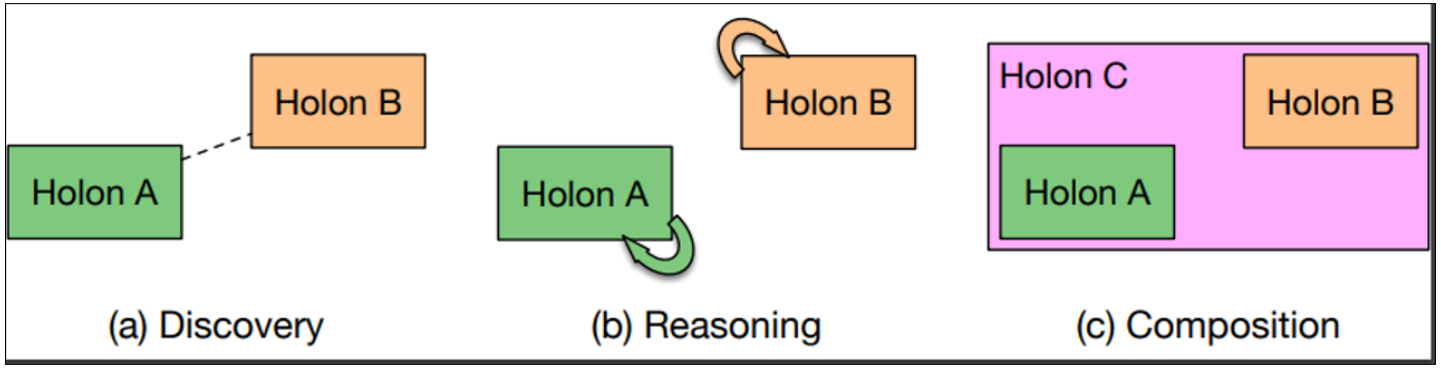


Research Focus

- Aim:
  - allow IoT development at a high-level as an *abstract workflow*
  - enable automated opportunistic construction and mediation in IoT contexts
- Objectives:
  - design development tools for high-level application logic
  - identify independent systems with their unique characteristics
  - recognize the need to compose with other systems
  - detect the need for mediation logistics
  - set in action any support roles needed to actuate mediation

Holon?

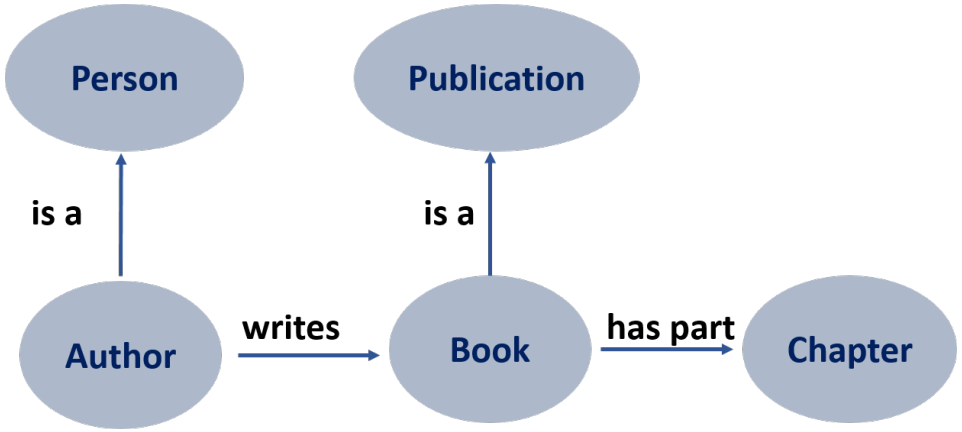
- A *holon* is a self-describing system that appears as a whole when viewed from above whilst potentially comprising multiple subsystems when viewed from below
- A holon is described using an ontology that can be
  - advertised
  - discovered
  - used to reason about opportunistic composition



Gordon Blair, Yérom-David Bromberg, Geoff Coulson, Yehia Elkhatib, Laurent Réveillère, Heverson B. Ribeiro, Etienne Rivière, and François Taïani, “Holons: Towards a Systematic Approach to Composing Systems of Systems”, Workshop on Adaptive and Reflective Middleware, 2015.

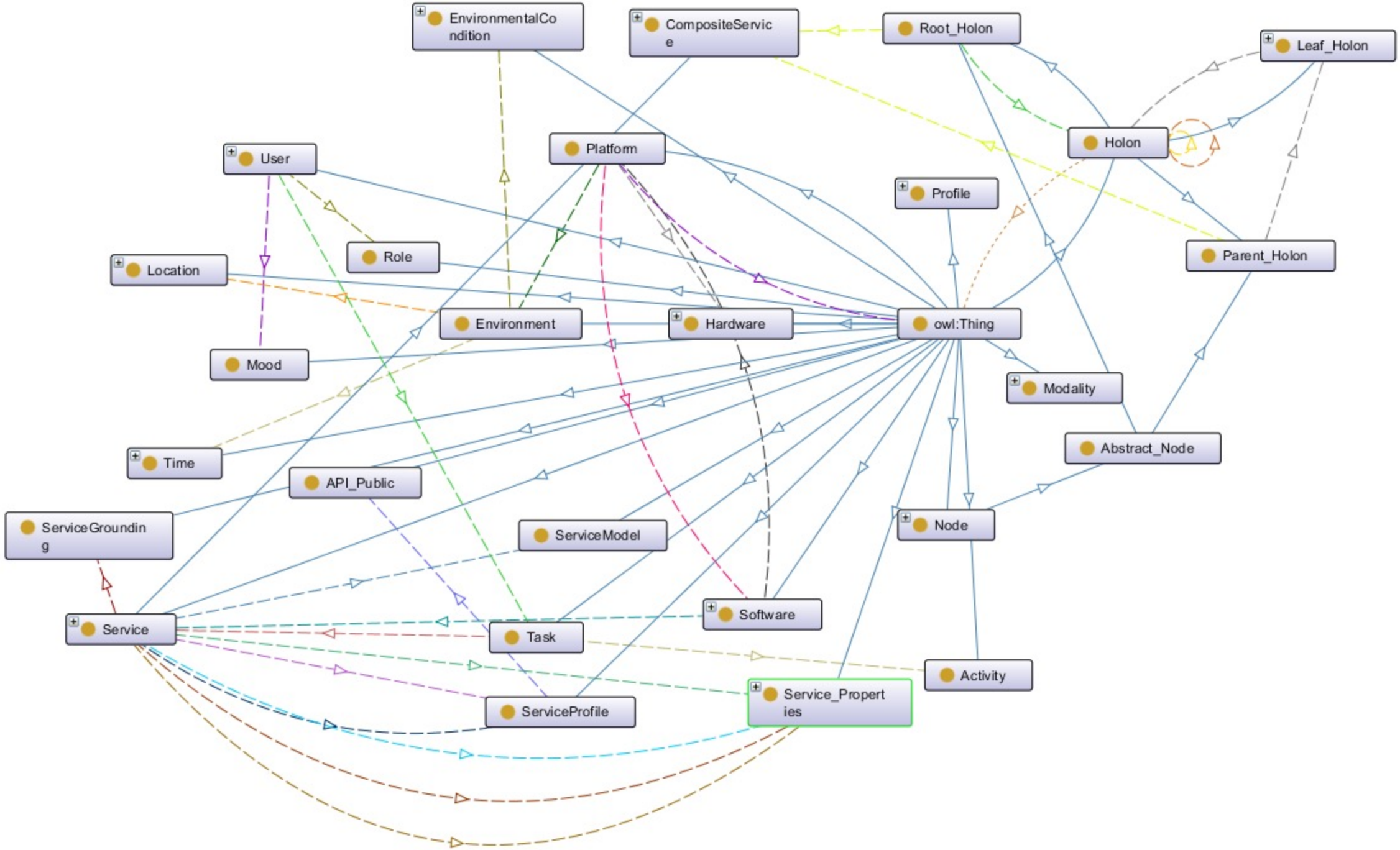
What is an ontology?

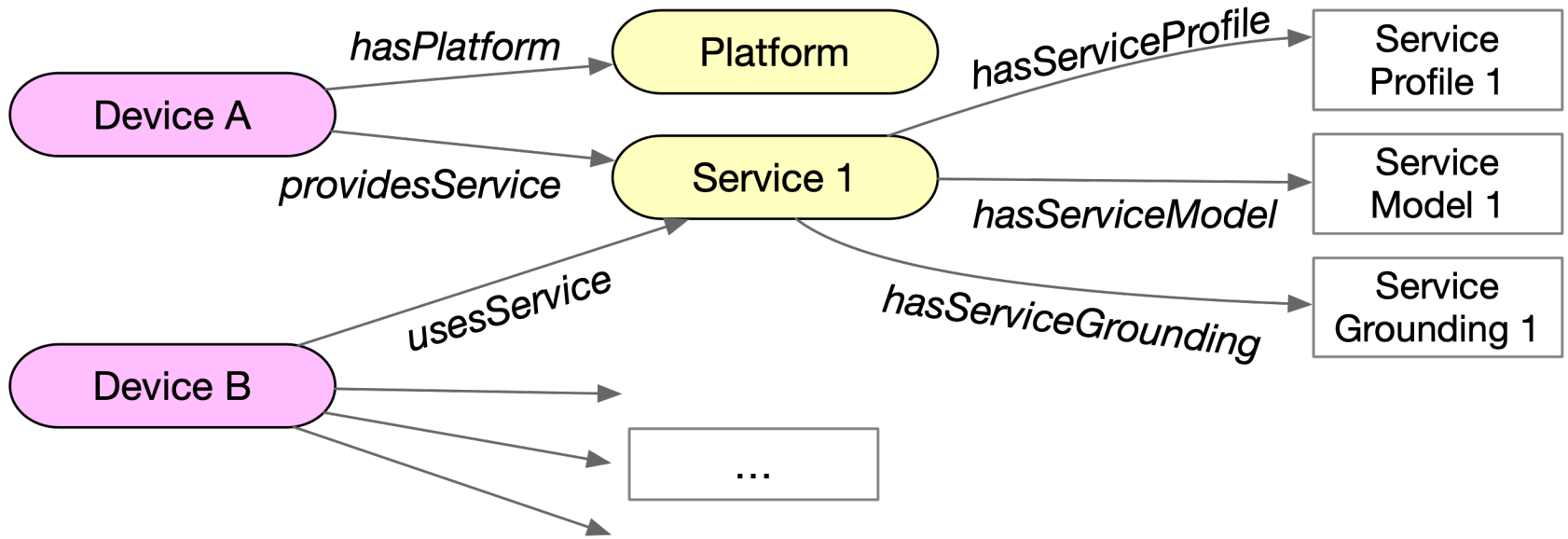
- An ontology is an abstract model of the world, or some domain
- An ontology introduces a vocabulary that is relevant to the domain
  - Explicitly defines concepts, properties, relations, and constraints found in the domain

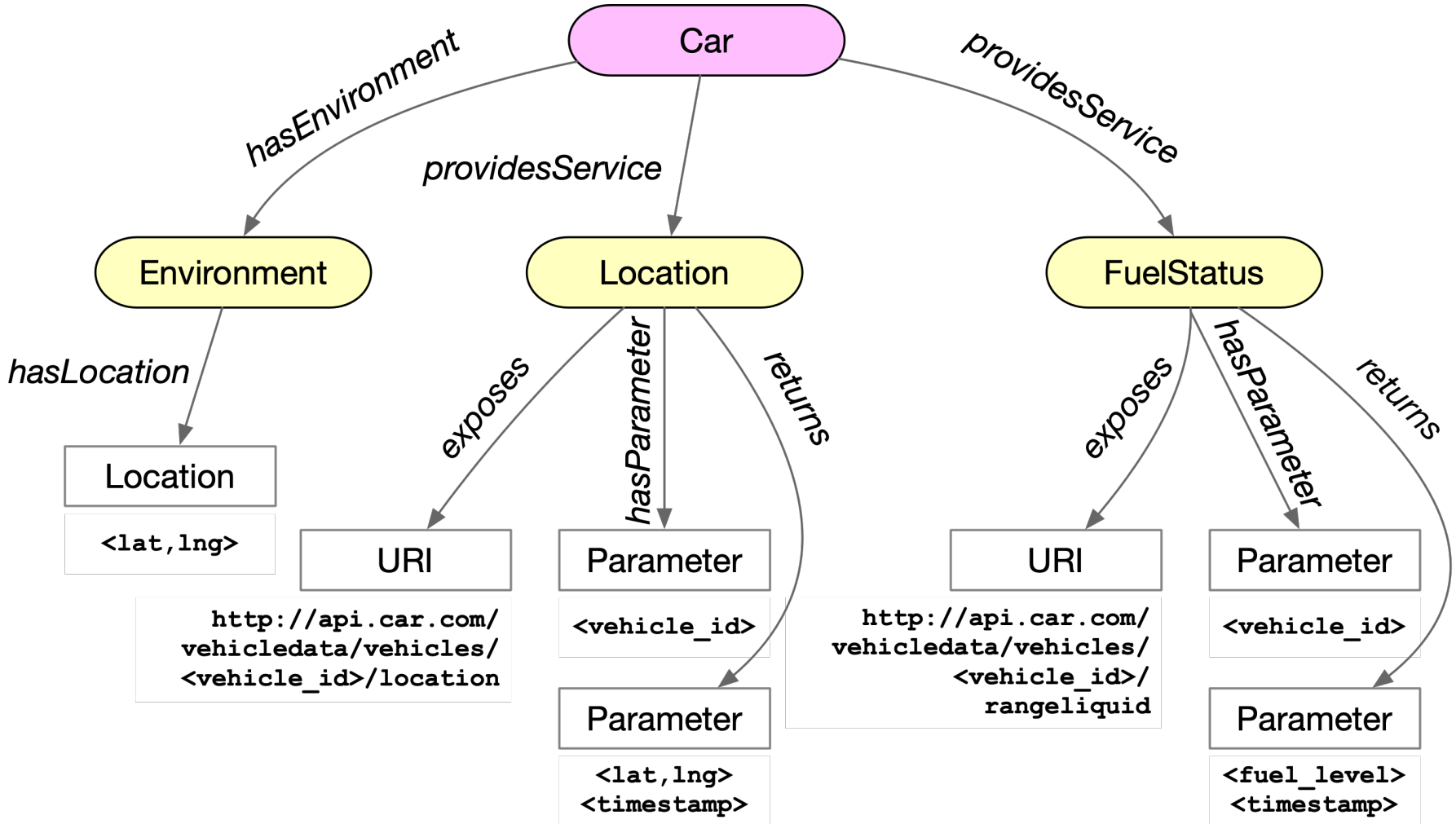


Specifying a Holon – Upper classes

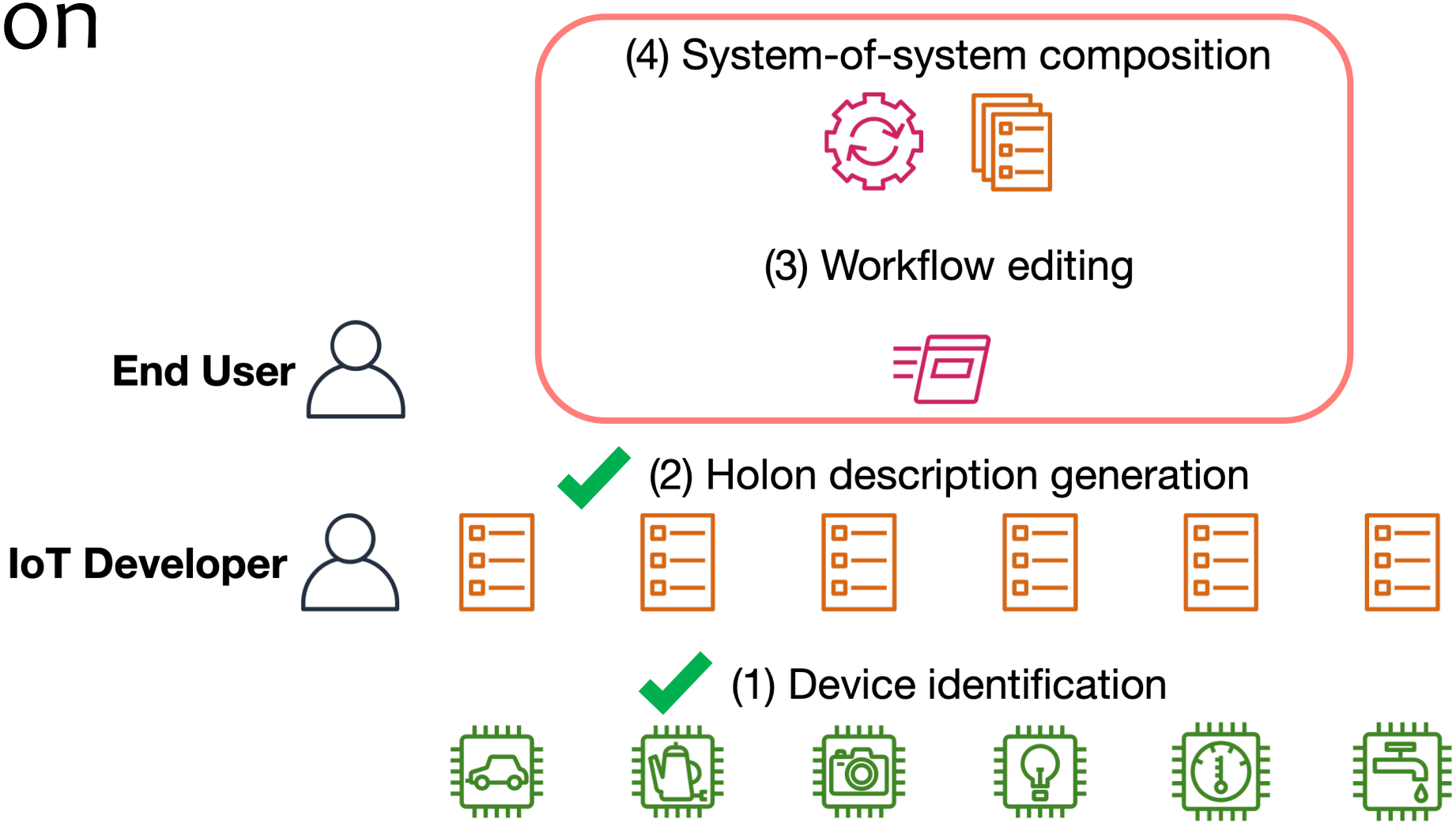
- HOLON:
  - has parent (**holon**); has children (**holons**); supports **services**
  - Membership **properties**: has Children, is Member
  - Physical Properties: power level; location; mobile; firmware; …
- SERVICE:
  - requires properties; desires properties; guarantees properties; may provide properties; exposes API
- PROPERTY: *(domain-specific)*
  - e.g. smart home: sensing temperature, humidity, motion; data storage; controlling climate, doors, windows, curtains; …


EnvironmentalCondition
CompositeService
Root_Holon
Leaf_Holon
Holon
Platform
User
Profile
Parent_Holon
Location
Role
Environment
Hardware
owl:Thing
Mood
Modality
Time
Abstract_Node
API_Public
ServiceGrounding
ServiceModel
Node
Software
Service
Task
Activity
Service_Properties
ServiceProfile


Device A
Device B
hasPlatform
providesService
usesService
Platform
Service 1
...
hasServiceProfile
hasServiceModel
hasServiceGrounding
Service Profile 1
Service Model 1
Service Grounding 1


Car
hasEnvironment
providesService
providesService
Environment
Location
FuelStatus
hasLocation
exposes
hasParameter
returns
exposes
hasParameter
returns
Location
<lat,lng>
URI
http://api.car.com/
vehicledata/vehicles/
<vehicle_id>/location
Parameter
<vehicle_id>
Parameter
<lat,lng>
<timestamp>
URI
http://api.car.com/
vehicledata/vehicles/
<vehicle_id>/
rangeliquid
Parameter
<vehicle_id>
Parameter
<fuel_level>
<timestamp>

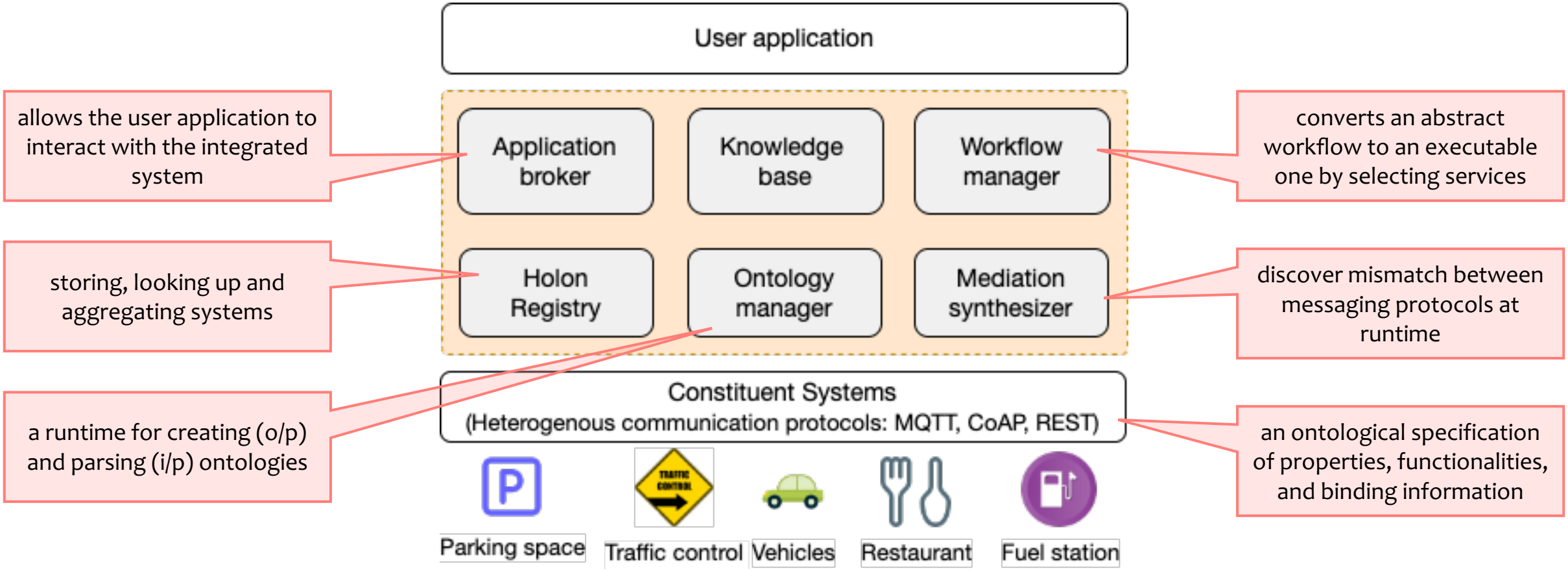
Vision


(4) System-of-system composition
(3) Workflow editing
End User
(2) Holon description generation
IoT Developer
(1) Device identification

Approach

- Apply the concept of ***holons*** to represent systems recursively
- Build an architecture for automated composition of holons
  - discover other systems and their services
  - opportunistically integrate with them to satisfy a user request
  - detect the need for mediation
  - create mediators at runtime for direct interaction with other systems
- Evaluate using a real-world Internet of Vehicles (IoV) scenario
  - productivity of and value added to developers through a user study

Hetero-Genius Architecture

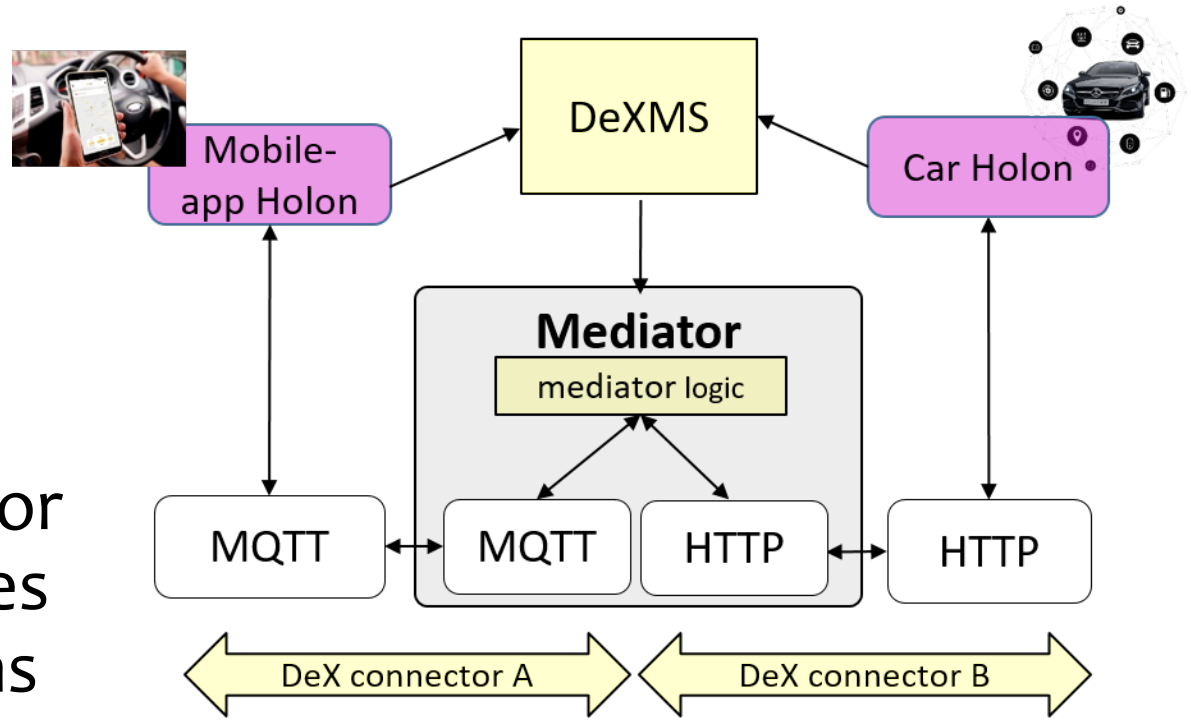


Concertizing an Abstract Workflow

- Find physical systems that provide the required services
- AB queries the HR for descriptions of suitable systems
- AB verifies the response to match:
  - the annotations of the required services (specified in the workflow), with
  - the annotations of the provided services (specified in the system's holon)
- Return the binding information of each service to the user application, along with messaging info (API, protocol)

Addressing Interoperability

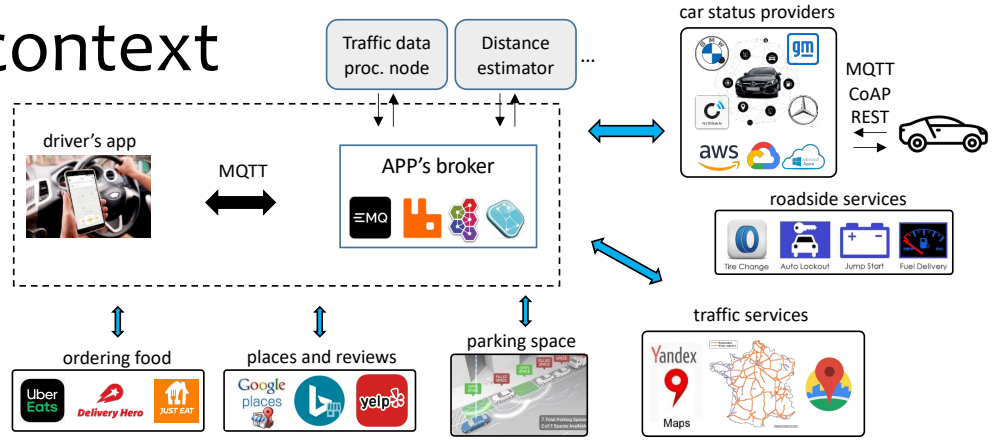
- Systems have different protocols, data-serialization formats, and interaction paradigms
- AB can detect this from holon descriptions
- We use the Data eXchange Mediator Synthesizer (DeXMS), which creates bridges between divergent systems



Georgios Bouloukakis, Nikolaos Georgantas, Patient Ntumba, and Valérie Issarny, “Automated Synthesis of Mediators for Middleware-layer Protocol Inter- operability in the IoT”, Future Generation Computer Systems 101 (2019), 1271 – 1294, 2019.

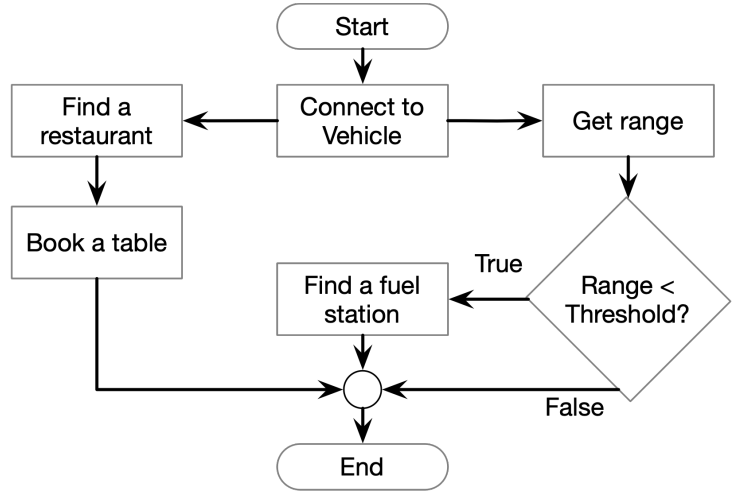
Evaluation – Overview

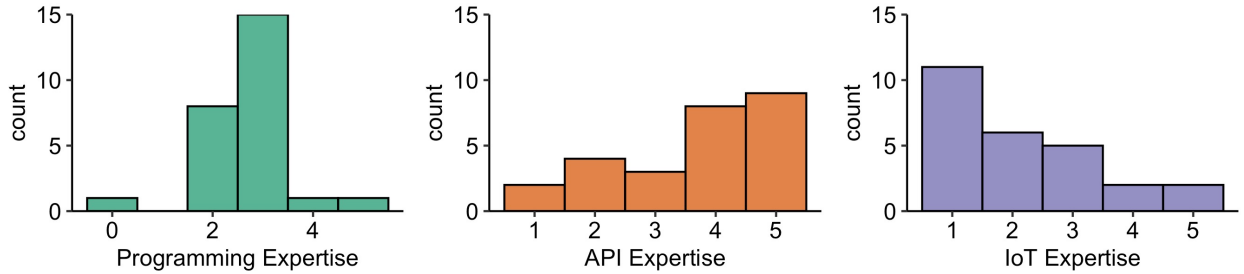
- Aim to measure:
  - Productivity – time required to develop application logic
  - Correctness – adherence to usage of correct APIs
- Strategy: user experiments from an IoV context
  - stop for food (and fuel?) on a car journey
- 4 systems:
  - car – provides **`fuelStatus`**, **`location`**
  - fuel station – provides **`location`**
  - restaurant – provides **`menu`**, **`details`** (location, contact details, etc)
  - traffic control unit (TCU) – provides **`trafficStatus`**



Evaluation – Procedure

- Simple workflow (round robin allocation)
  - classic – use native APIs to implement functionality
  - using holons – learn annotations and implement
- Duration: up to 1 hour (+5 minutes intro)
- Reward: £15 Amazon voucher
- Recruitment: 26 participants
  - researchers, students, developers at campuses, local startups, incubators
- Self-reported expertise levels:





Sample Response – Classical

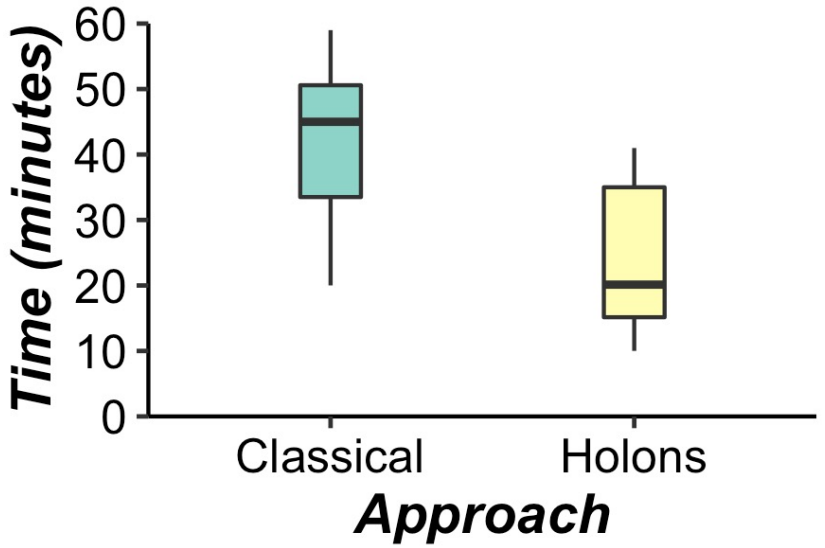
```
String vehicleId = "123";
int fuelThreshold = 111;
Location locationToNavigate; //assume Location data format exists
String token = ""; //authentication todo
String resultFuel = call("https://api.mercedes-benz.com/vehicledata/v2/vehicles/vehicleId/resources/rangeliquid", token);
int factFuel = JsonParser.parse(resultFuel).get("rangeLiquid").get("value");
if (factFuel <= fuelThreshold) {
    String location = "" //find location todo
    String resultStations = call("https://api/alt-fuel-stations/v1/nearest.format?parameters", location);
    double factLatitude = JsonParser.parse(resultStations).get("latitude");
    double factLongitude = JsonParser.parse(resultStations).get("longitude");
    locationToNavigate = find(factLatitude, factLongitude); //assume find function exists
}
String resultRestaurants = call("https://foodbukka.herokuapp.com/api/v1/restaurant");
String[] factResuatrants = JsonParser.parse(resultrestaurants).get("businessName");
String selectedRestaurant;
if (selectedResuarant.equals("Foodfusion") {
    String FoodfusionToken = "";
    String restaurant = "Foodfusion";
    int noOfPersons = 1;
    Date bookingFrom = Date.now;
    String resultFoodfusion = call("https://foodfusion/booking", FoodfusionToken, restaurant, noOfPersons, bookingFrom);
    String factFoodfusion = JsonParser.parse(resultFoodfusion).get("statusCode");
    if (!"OK".equals(factFoodfusion)) {
        //repeat booking
    } else {
        //locationToNavigate = this restaurant
    }
} else if (selectedResuarant.equals("Hathaway") {
    String name = "John";
    String number = 1;
    String email = "john@gmail.com";
    Date date = Date.now;
    String resultHathaway = call("https://api.hathawayfood.com/booktable", name, number, email, date);
    String factHathaway = JsonParser.parse(resultHathaway).get("statusCode");
    if (!"OK".equals(factHathaway)) {
        //repeat booking
    } else {
        //locationToNavigate = this restaurant
    }
}
//navigate to locationToNavigate
```

Sample Response – Holon

```
String vehicleResult = call("http://holons/registry/getHolon?holonID");
vehicle= Holon.parse(vehicleResult);
connection = vehicle.call(AUTHENTICATE);
fuel= vehicle.call(CAR_RANGE);
threshold = 5;

if(fuel < threshold ) {
    String stationResult= call("http://holons/registry/getHolon?type=station&longitude=longitudeValue&latitude=latitudeValue)";
    fuelStation= Holon.parse(stationResult);
    echo "Fuel Station";
}

String restaurantResult= call("http://holons/registry/getHolon?type=restaurant&longitude=longitudeValue&latitude=latitudeValue");
restaurant= Holon.parse(restaurantResult);

restaurant.call(BOOK_TABLE);
```

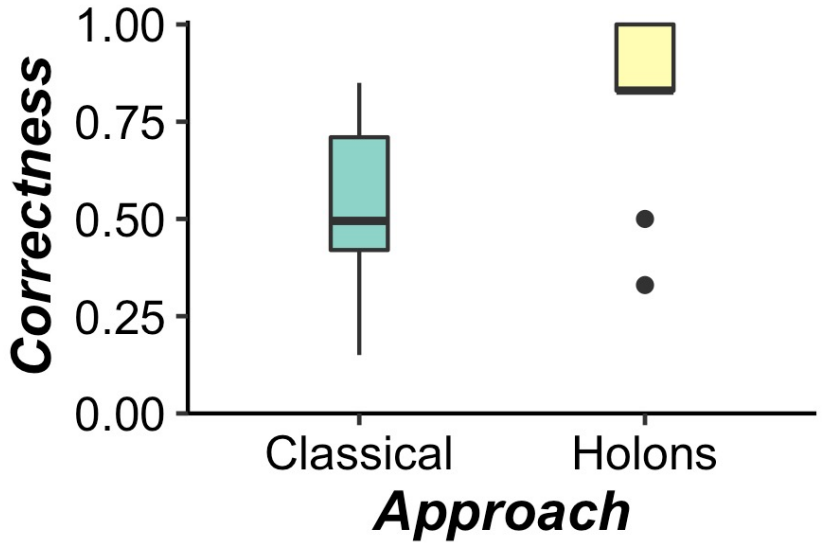
Results – Productivity

- Average / median time to complete task
  - classic – 43.82 / 45.30
  - holons – 23.25 / 20.13
- Level of improvement varies according to the level of programming expertise
- Overall, we improve productivity by 47%



Results – Correctness

- We manually inspect participant code for
  - correct usage of the required APIs
  - adherence to realizing the given workflow
- Calculate percentage score
  - one point for each correct API
  - one point per milestone code statement
- Average / median % correctness score
  - classic – 54.25 / 49.50
  - holons – 84.35 / 83.00
- Overall, we improve correctness by 55%



FAQ/Cs

- You can't have "one ontology to rule them all"
  - just one means, with a tradeoff between generality and usefulness
  - overhead is not insurmountable as evidenced by our experiments
  - we do not expect manufacturers to change anything
- Developer overhead still too high, e.g. end users
  - developed an approach using graphical programming techniques

    Zhuo Wang, Yehia Elkhatib, and Abdessalam Elhabbash, "HolonCraft – An Architecture for Dynamic Construction of Smart Home Workflows", Conf on Future Internet of Things and Cloud (FiCloud), IEEE, August 2022.
  - developed NLP method to automatically generate Holon descriptions

    Ziyu Zhang, Yehia Elkhatib, Abdessalam Elhabbash, "An NLP Method for Automated Generation of Smart Home Ontological System Description for Reasoning and Composition", *(in progress)*

Conclusion and Next Steps

- Ontology-based architecture for composition, reasoning, mediation
- Evaluated efficacy in a IoV user study
- Next:
  - Quantitative evaluation to investigate effectiveness (latency) and computational overhead (CPU, memory, messages) at different scales
  - Tools for generalising to other IoT ecosystems
    - wildfire detection, disaster rescue, dynamic cluster management

Thanks for your attention!

https://yelkhatib.github.io/


Abdessalam


Yehia


Georgios


Maria