Use, Create, and Promote open source!

Gael Blondelle, Brussels, 2024

There is only one definition of Free Software (since 1984) and Open Source is synonymous!

Freedom 0 - to **run** the program, for any purpose

Freedom 1 -to **study** how the program works, and change it to make it do what you wish

Freedom 2 - to **redistribute** copies

Freedom 3 - to distribute copies of your **modified** versions to others

Open Source is running the world!

> **Today, you can't develop software without doing open source!**
>
> — Mercedes Benz

81%

% companies consuming open source in products or services

44%

% firms contributing to upstream open source projects

80-90%

Open source makes up 80-90% of applications

Sources: TODO Group, Forrester

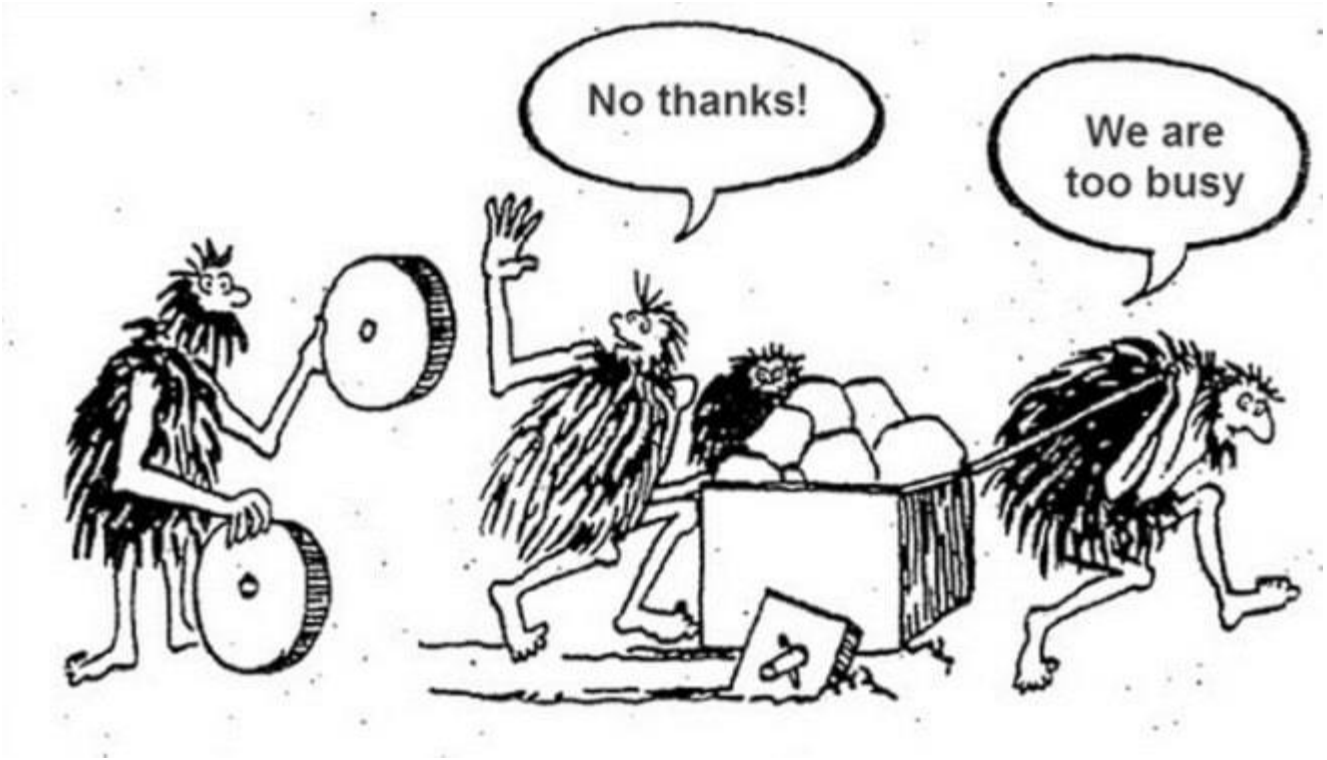
ECLIPSE FOUNDATION

Abandonware is not an option!

My reactions to Abandonware!

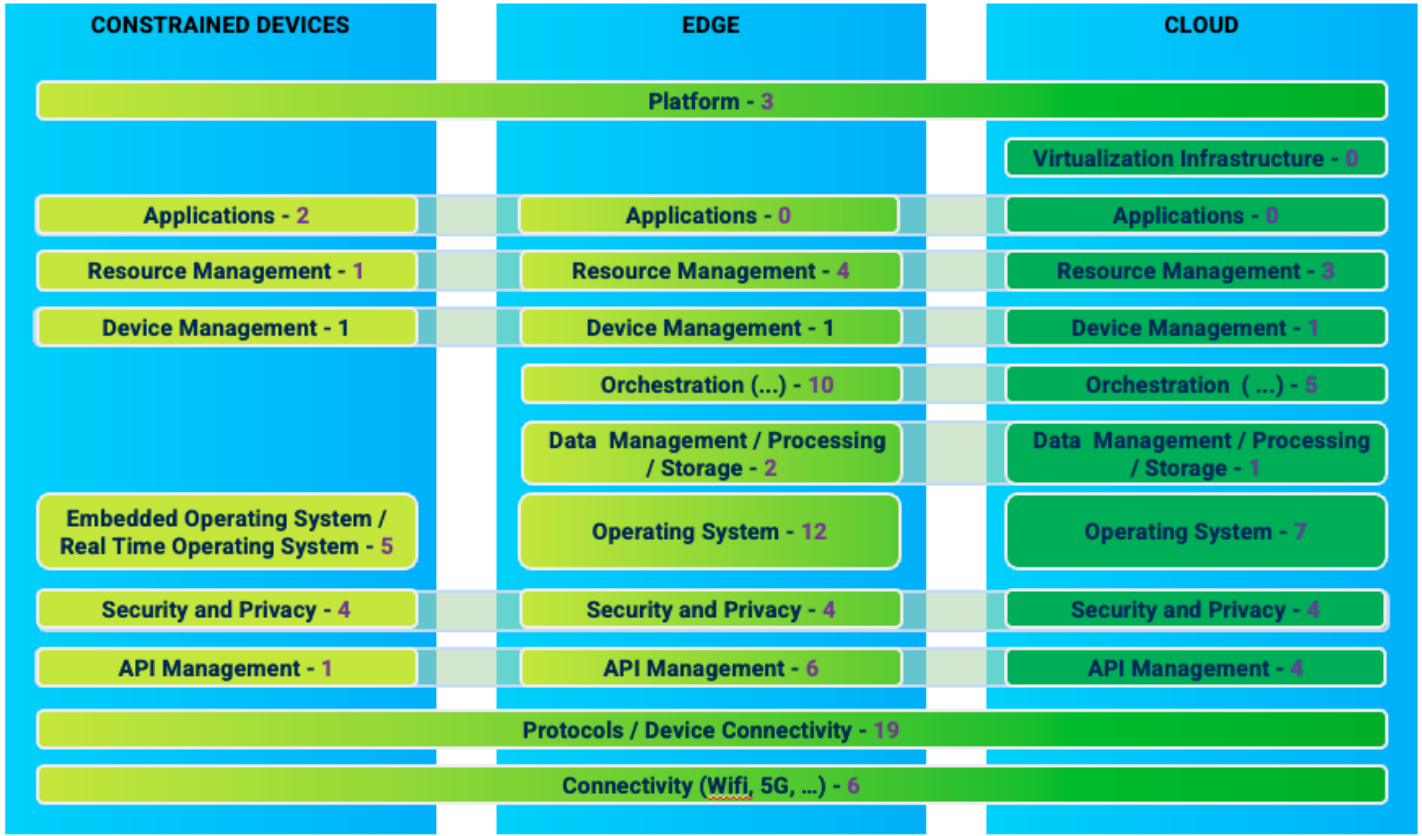
Too bad, that's a lost opportunity

Reuse instead of reinventing the wheel : eg MQTT

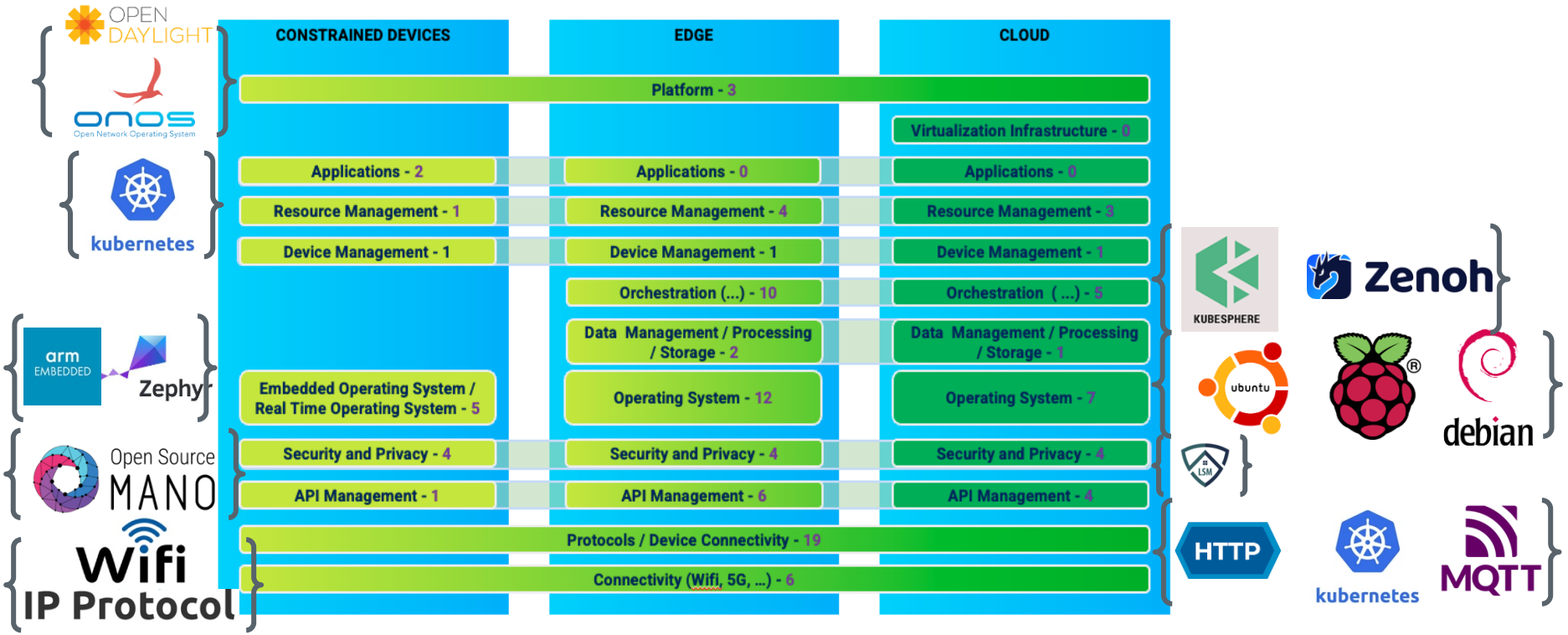


… be faster in reach a good base and innovate on it!

Open Source Stack - Swarm Intelligence view



Open Source Stack - Swarm Intelligence view



Open source is Mandatory for Swarm computing

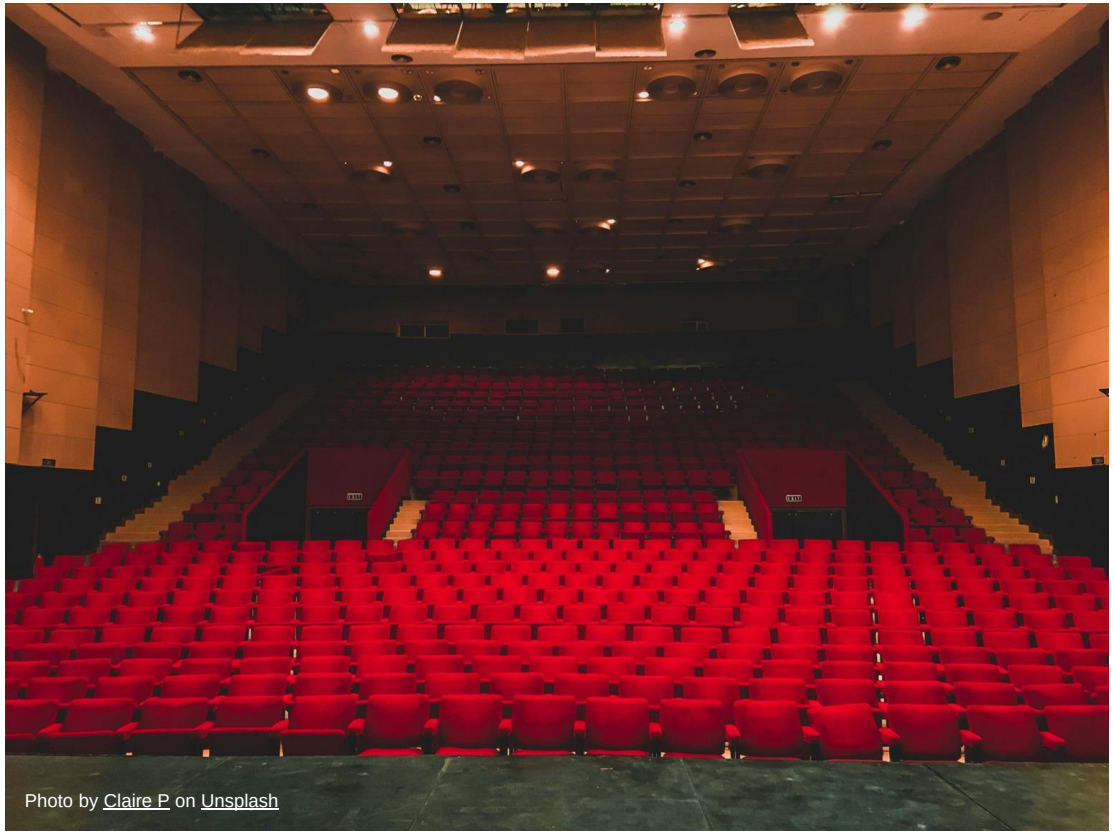
Scalability

Interoperability

Be noticed

Your project community is the one you create!

Your project community is the one you create!

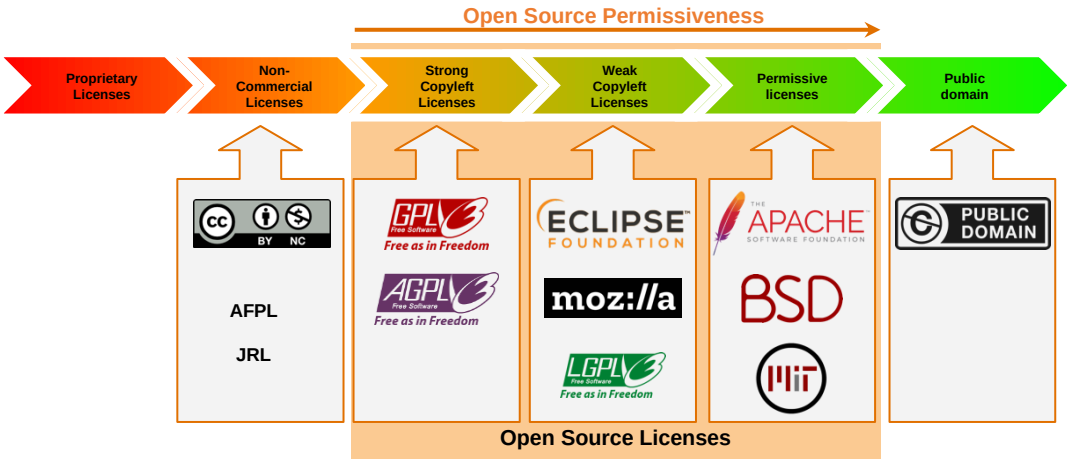


Start the party and they may come!

Open Source is about Intellectual Property management

Select an OSS license for the project

- **Select a popular, with a strong community license**
  - https://opensource.org/licenses
- **Avoid exotic licenses, usually rejected by industry**



Start open sourcing as early as possible

Photo by Aron Visuals on Unsplash

Open the kitchen! Work in public!

Rally partners! Collaborate in open source!

OCX 24

OPEN COMMUNITY EXPERIENCE

OCX KEYNOTE SPEAKER

We Build Software in the Open to Build Trust

SARAH NOVOTNY

Open Source Champion

22 - 24 October 2024

Mainz, Germany

Register at OCXconf.org

#OCX24

eSAAM 2024 on Data Spaces

OCX 24 OPEN COMMUNITY EXPERIENCE

4th Eclipse Security, Artificial Intelligence, Architecture, and Modelling Conference on DATA SPACES

Mainz, Germany | 22 October 2024

Co-Organisers

ECLIPSE FOUNDATION

ITI INVESTIGATE TO INNOVATE

HELLENIC REPUBLIC UNIVERSITY OF MACEDONIA

CERTH CENTRE FOR RESEARCH & TECHNOLOGY HELLAS

OpenSwarm

Coaty
A Framework for Collaborative IoT

Siemens Technology

Presented by Danny Hughes (KU Leuven)

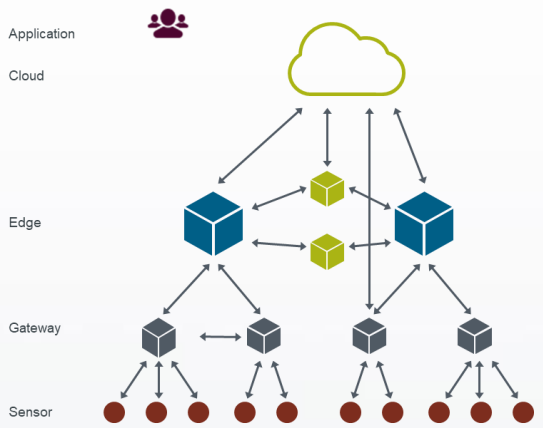
www.openswarm.eu

Outline

- Motivation and Trends
- The Coaty Framework
  - Overview
  - Communication Foundations
  - Evolution
- Exemplary OpenSwarm Use Case
- Summary

Motivation

- Market shows a clear trend towards systems collaborating independently and autonomously as self-organizing system of systems
- Demand of collaborative smart autonomous systems identified across all major industrial domains
- A decentralized collaborative framework is considered to be a cornerstone for Swarm Applications



Cloud-centric Hierarchical IoT network

Decentralized Non-hierarchical IoT network

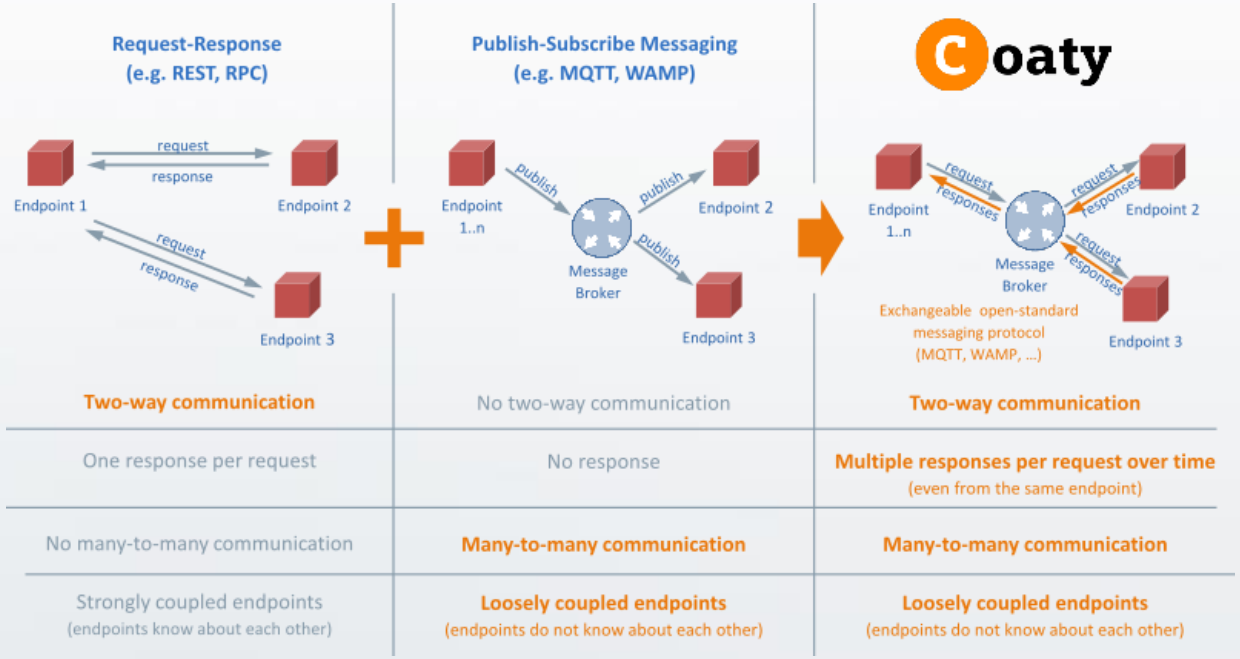
"Coaty enables powerful any-to-any collaboration between your autonomously acting IoT devices, people, and services in ever-changing scenarios."

Coaty Framework

- Implementation of interaction and communication foundation for smart autonomous systems in distributed, decentralized applications.
- Provides software framework for data-centric agent interaction with loosely coupled systems, any-to-any communication, and smooth handling of asynchronous events.
- Provides collaboration capabilities in a middleware layer on top of transport protocols and OS layer / stacks
- Applicable to the full scale of potential deployments of agents (e.g., Cloud, Edge, Wearables).
- Open Source framework powered by Siemens (https://coaty.io)

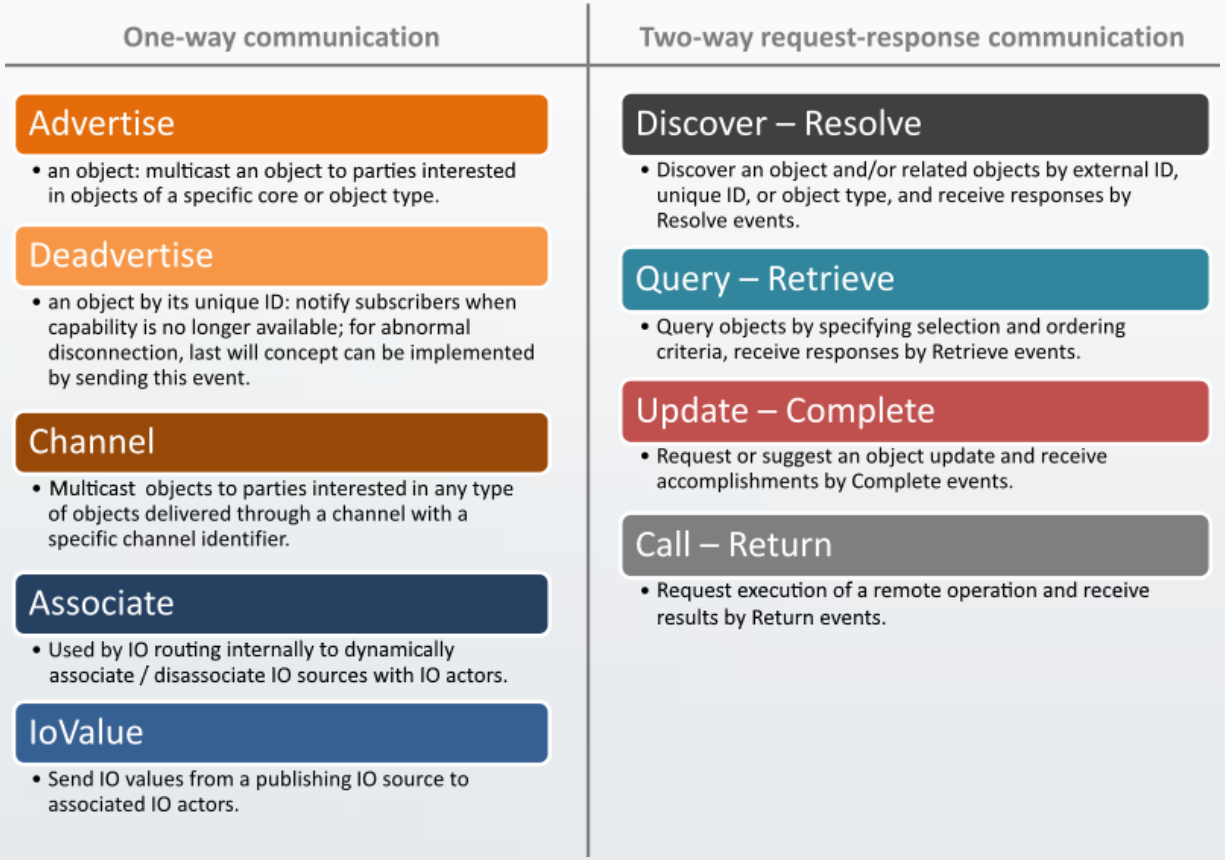
Coaty Communication Foundation

- Data-centric communication over interchangeable messaging protocols
- Combines classic request-response and classic Pub/Sub communication patterns
- Supports loose coupling of decentrally organized components
- Provides routed one-way/two-way and one-to-many/many-to-one communication flows



Coaty Communication Patterns

- Offers a minimal, yet complete core set of patterns for distributing data in motion, data in use, data at rest
- Provides generic abstraction layer on top of interchangeable messaging protocols, avoiding vendor lock-in
- Routing of information flows by dynamic subscriptions based on context.
- Spatial filtering of information flows by location and proximity aware subscriptions.



Coaty Evolution

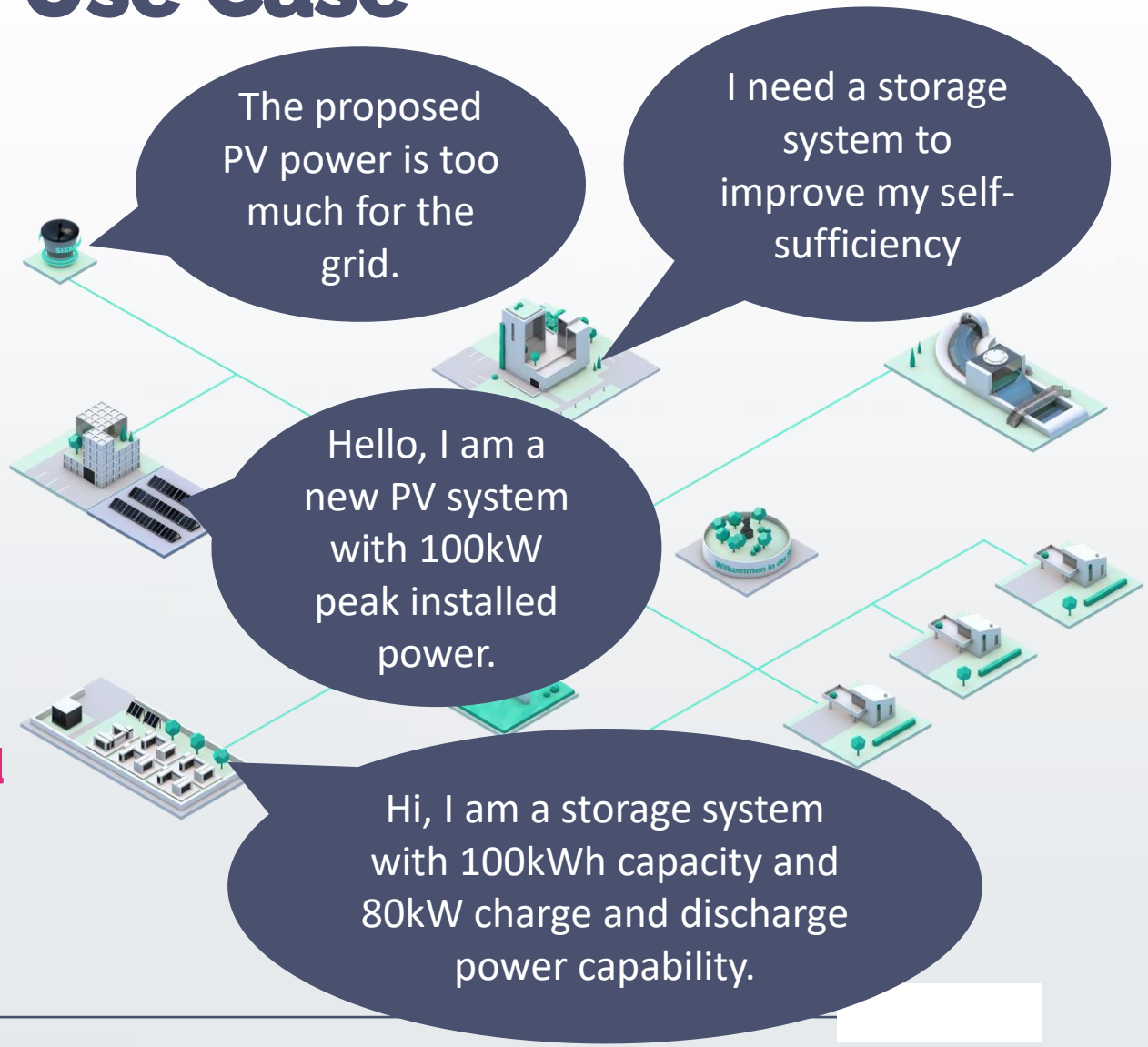
- Coaty has evolved to be “Run anywhere”, providing support for different deployment strategies
  - As a lightweight ***sidecar*** binary integrated with applications through modern open-standard APIs based on gRPC and gRPC Web
  - As a lightweight ***library*** distributed as a Golang module with packages to be imported directly into an application component
- Sidecars are very lightweight and work across edge, cloud, on-premise, and hybrid environments, either as processes or containerized.
- Supports observability, i.e. the ability to measure and infer its internal state by analyzing OpenTelemetry traces, logs, and metrics that it generates and exports.
- The evolved Coaty framework, dubbed Data Distribution Agent (DDA), is available as Open Source in https://github.com/coatyio/dda.

Exemplary OpenSwarm Use Case

Renewable Energy Community (REC)

- Energy shared or sold locally
- Potential for “grid friendly” behavior utilizing active components like
  - Storage systems
  - E-Cars
  - Active loads

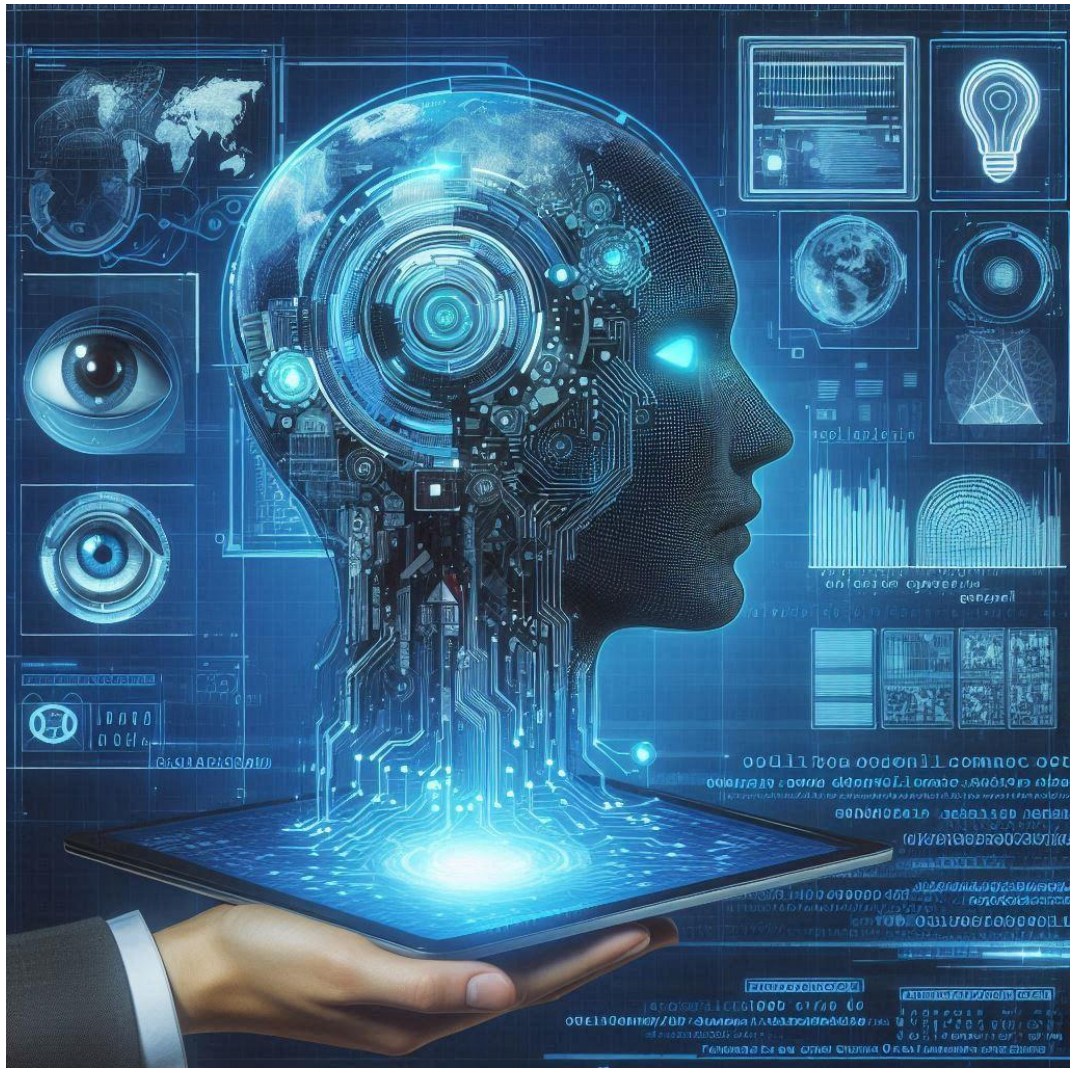
Coaty acts as the enabler for decentralized communication among the different components.



Summary

- Coaty Removes the complexity in developing communication technologies for distributed systems removing developers from the burden of in-depth knowledge of messaging protocols and communication networks.
- Coaty has evolved towards supporting different deployments from sidecar to standalone deployments suitable from the cloud to small footprint components on constrained devices.
- In addition to many success cases from the past, in OpenSwarm we take Coaty into a whole new set of Application Domains.

Chunks and Rules for Cognitive Control



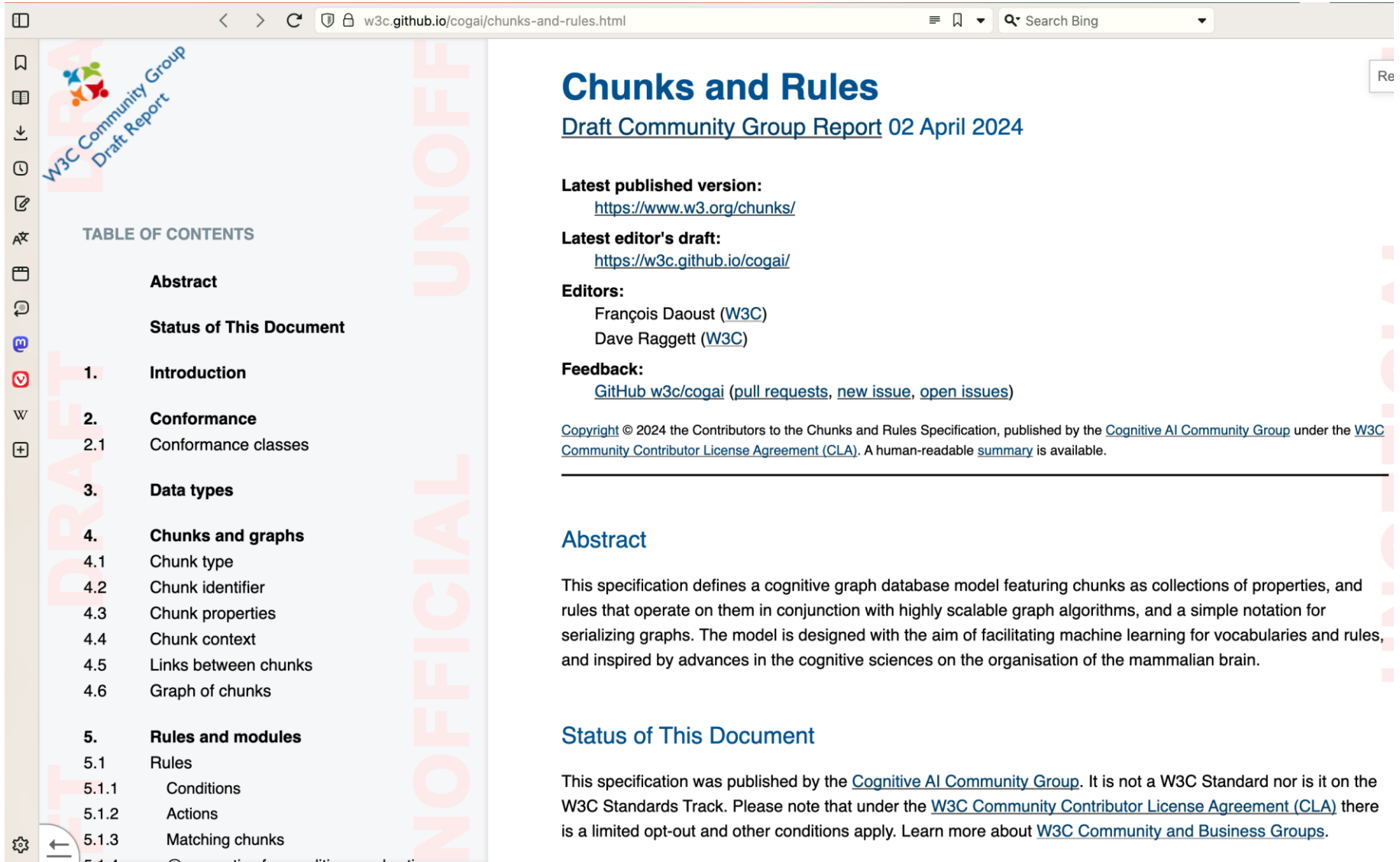
Cognitive Approach to Low-Code Control

- *Low-code* is an approach to application development that simplifies the process of automating workflows and building applications
- Some low-code platforms use visual drag-and-drop elements and prebuilt components along with scripting
- Empowering professional developers and business users to create applications more efficiently
- Cognitive approach mimics how humans execute tasks, drawing upon decades of work in the cognitive sciences
- Behaviour is described using facts + rules

- Enabling application developers to use a low-code cognitive approach to specifying real-time behaviour
- Event-driven concurrent threads of behaviour using APIs exposed by resources as described in taxonomies
- Easy to learn, convenient syntax for chunks* and condition-action rules
  - W3C Cognitive AI CG's Chunks & Rules specification
- Mature JavaScript library
- Extension to distributed agents, e.g. swarms using asynchronous message exchange

* Chunks are sets of name/value pairs

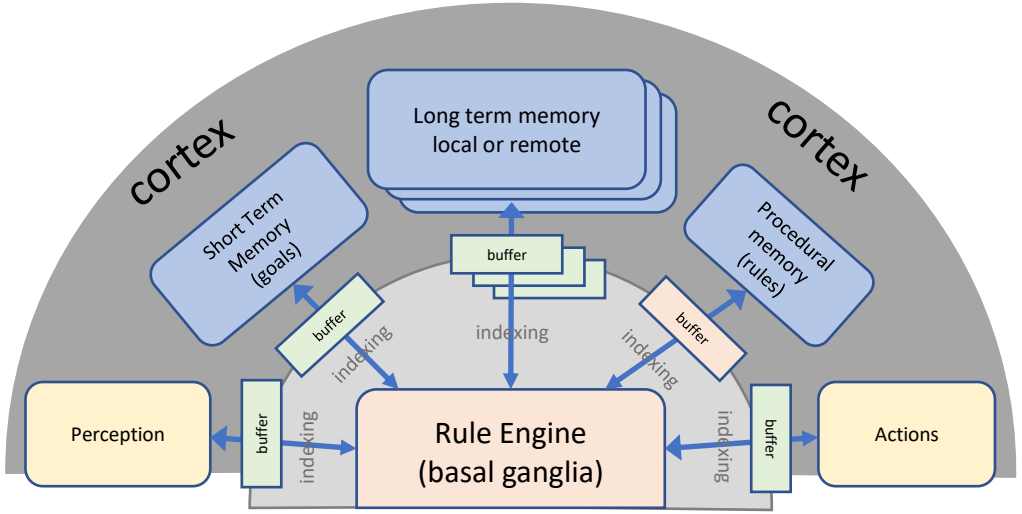
Formal Specification from Cognitive AI CG



Cognitive Architecture

- Inspired by John Anderson's ACT-R
- Mimics characteristics of human cognition and memory, including spreading activation and the forgetting curve
- Asynchronous operations that enable distributed cognition
- Perception builds live models of the environment including events that trigger corresponding behaviours
- Actions expressed as intents to be realised as appropriate
  - *intent*: an aim, purpose, goal or objective
- Reasoning is decoupled from real-time control over external actions, e.g. a robot arm

Cognition – Sequential Rule Engine

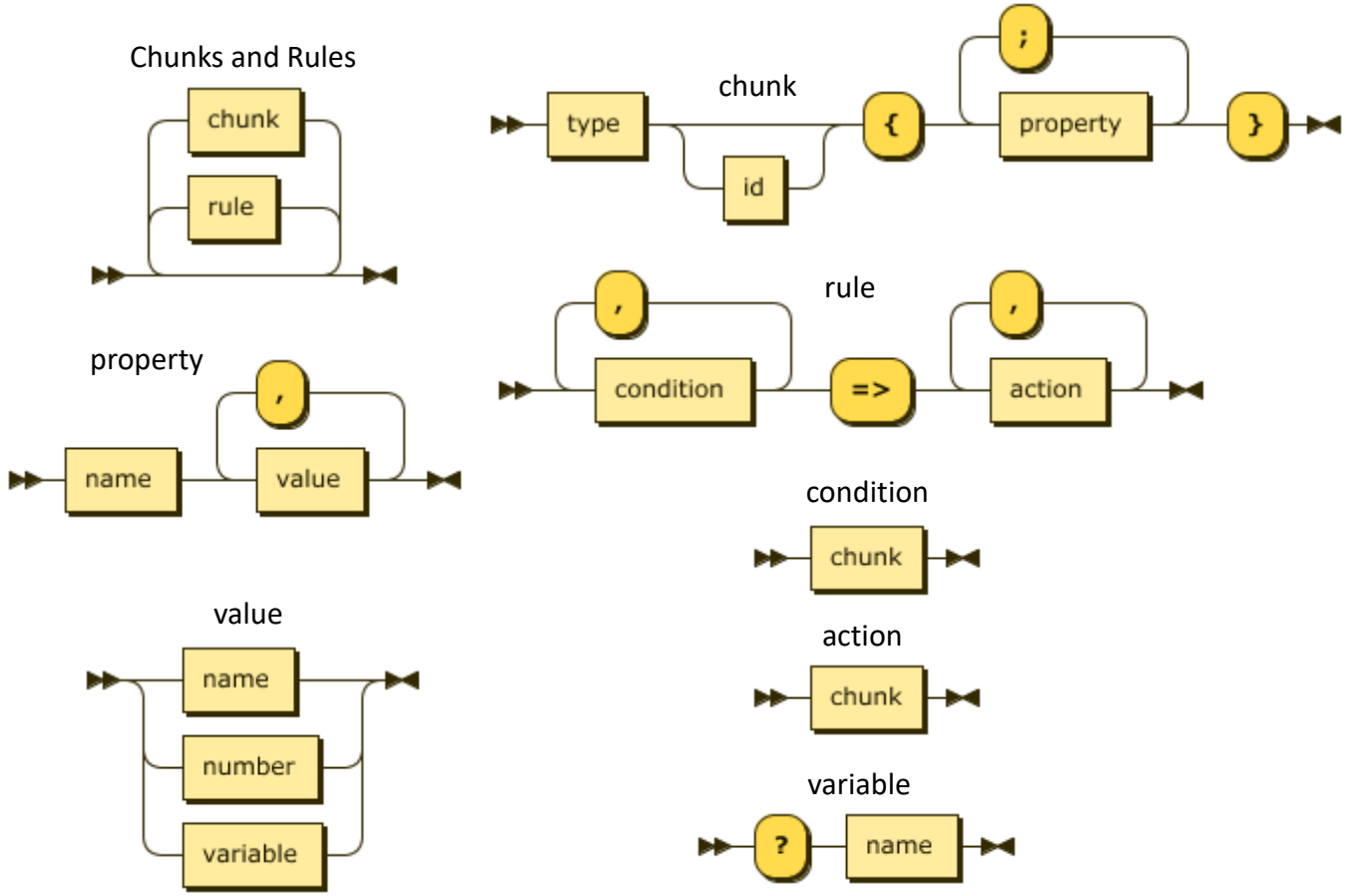


- The cortex holds a set of cognitive modules, each of which is associated with a module buffer that holds a single chunk
- Predefined asynchronous operations on buffers in analogy with REST

Chunks and Rules

web-based demos for smart homes and factories

- Chunks are sets of properties
  - Name/value pairs that correspond to a set of RDF triples with same subject
- Rule conditions and actions that specify which cognitive module buffer they apply to
- Variables are scoped to the rule they appear in
- Actions either directly update the buffer or invoke operations on the buffer's module, which asynchronously updates the buffer
- Extensible suite of cortical operations inspired by REST

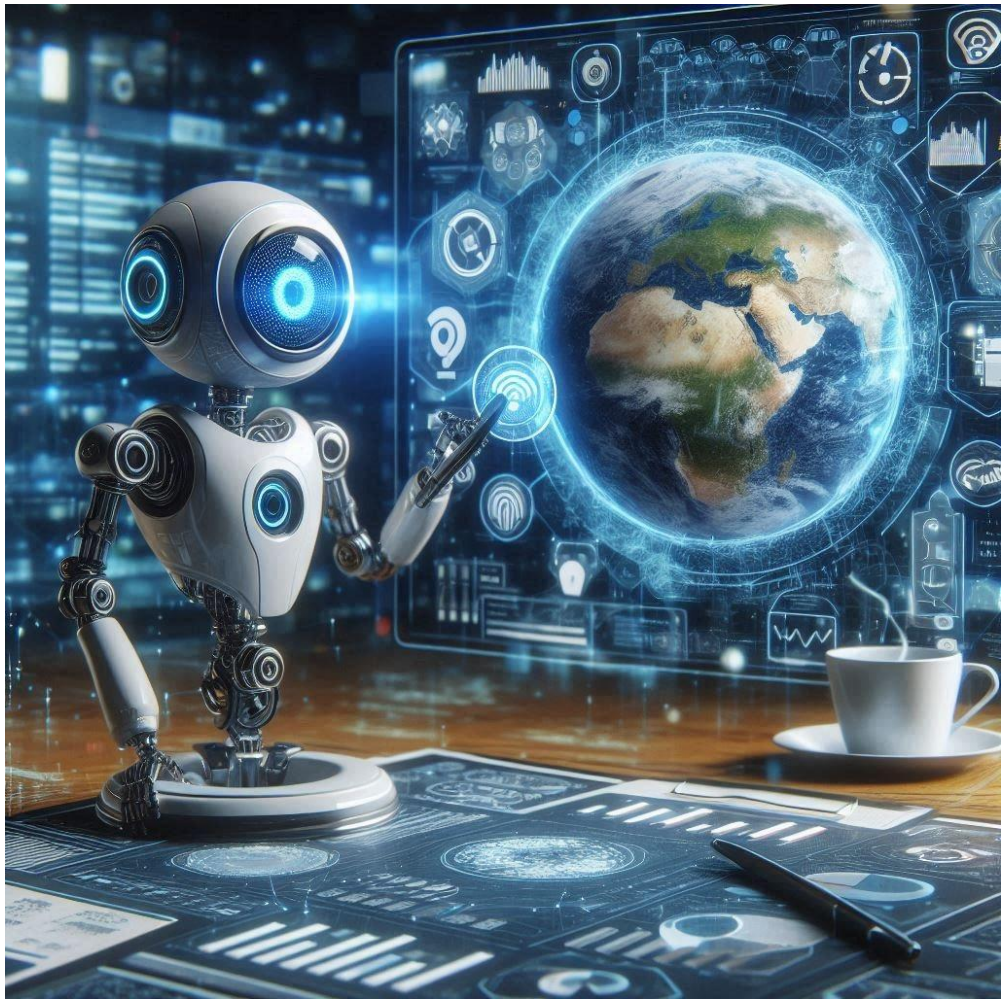


names beginning with "@" are reserved, e.g. @do for actions

See [W3C Cognitive AI Community Group]

Chunk Rules for Digital Twins

- Nephele's Virtual Objects are related to digital twins for devices, processes and even people*
- Chunk rule actions can be used to invoke the affordances exposed by digital twins
- Some glue code is needed to handle the data formats and protocols
- Complex results involve using the predefined suite of operations over chunk graphs given that module buffers are limited to single chunks

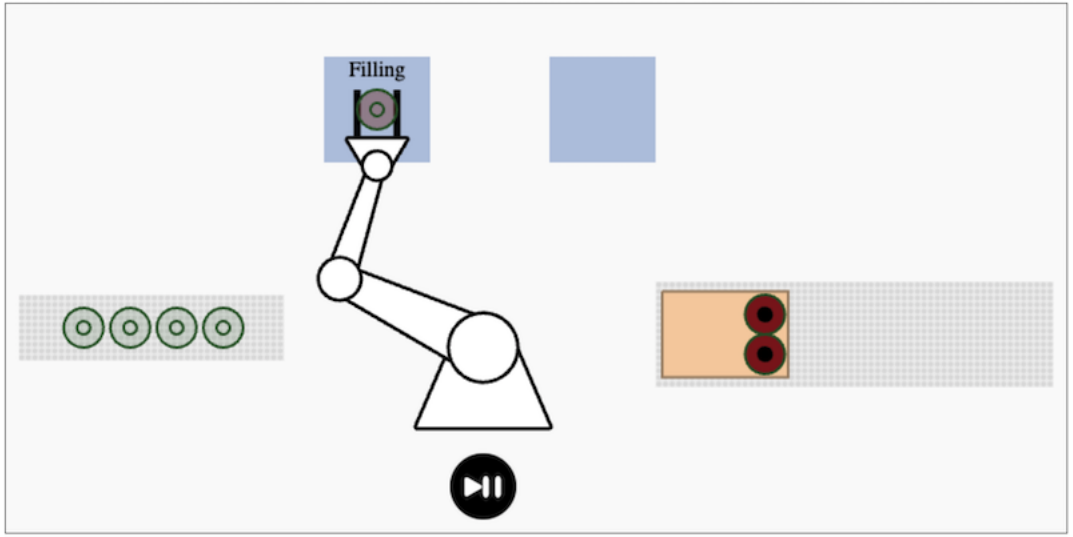


* Digital twins for use in healthcare applications, and for virtual devices as abstractions over multiple physical devices (i.e. composite virtual objects)

Chunks and Rules

- Mature JavaScript library for use in webpages or with NodeJS
- Application script declares additional operations, e.g. for robot control, layered above ROS operations
- These are implemented in JavaScript and can use real-time clock as well as networking for external messaging
- ERCIM can help with this
- Contact Dave Raggett <dsr@w3.org>

Note: rules can use variables for dynamic parameters where an object location is determined at run-time.



Factory demo: filling, capping and packing bottles of wine with real-time control over conveyor belts, filling and capping machines, and a robot arm

```
# move robot arm into position to grasp empty bottle
after {step 1} =>
   robot {@do move; x -170; y -75; angle -180; gap 30; step 2}

# grasp bottle and move it to the filling station
after {step 2} =>
   goal {@do clear},
   robot {@do grasp},
   robot {@do move; x -80; y -240; angle -90; gap 30; step 3}
```

See also smart homes demo

Robot Operating System (ROS)

ROS diffbot

- ROS is an open source software framework for robots
  - Linux, Windows, MacOS
- Strong developer community
- Message based with hardware abstraction
  - Topic based streams
  - Services with request/response
  - Nodes for message exchange
  - Shared database for parameters
- **Chunks & Rules** are a good fit for controlling ROS robots
- Using ROS topic streams to update chunk models of robots and their environment
- Using Chunk Rules to involve ROS services
  - Delegation for planning and execution
- Existing JavaScript libraries for integration with ROS

Iterative Refinement

- Cognitive rules can respond in milliseconds*, and can be complemented by faster reactions using simple reflex responses implemented at a lower level
- Application development is a collaboration between people maintaining the low-code description of high level behaviour and system programmers responsible for the glue code for the digital twins, i.e. Nephele (composite) VOs
- Development starts using a simple approach and iteratively refines it as new requirements come to light, e.g. when something unexpected occurs at run-time and needs to be handled
- That may further necessitate changes to the digital twins, e.g. to sense error conditions
- In robot use case: errors such as a bottle falling over, being only partially filled, or badly capped

* Rule execution is fast as time consuming operations are handled asynchronously

Questions?