

# W3C and digital twins for industrial systems

Dave Raggett <[dsr@w3.org](mailto:dsr@w3.org)> ERCIM

15 May 2025



Grateful acknowledgement for support from the [Nephele Project](#) by the European Union's Horizon Europe research and innovation programme under grant agreement No 101070487.



# Overview

- Digital Twins
- Web of Things
- RDF 1.2 and related standards
- NGS-LD and RDF
- Web-based monitoring, orchestration and simulation
- Chunks & Rules for low-code adaptive control
- Extension to swarm computing
- Where next with AI?

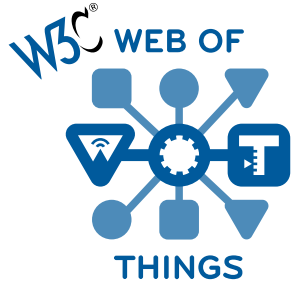


# Digital Twins

- Digital Twins can be defined as computer models of physical systems and processes
- Digital Twins can be used for:
  - monitoring current state of their physical twins
  - controlling their physical twins
  - diagnosing and fixing faults using causal models
  - planning and simulation
  - optimisation based upon applying machine learning to recorded data



A robot arm with its digital twin



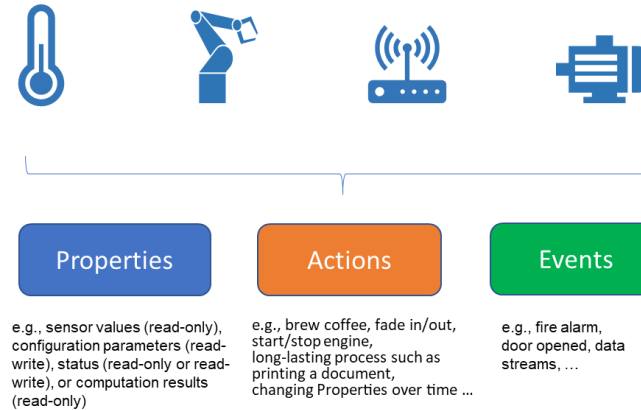
# Web of Things

Protocol Agnostic

<https://www.w3.org/WoT/>

- W3C Web of Things (WoT) simplifies application development by decoupling applications from the details of the underlying protocols
- Digital Twins are modelled as objects with properties, actions and events
- RDF (JSON-LD) is used for declarative descriptions of the object model, the semantics, communication protocols and security settings
- Applicable to modelling locally connected devices and devices at the edge, as well as digital twins hosted in the cloud

Semantic Interoperability = shared understanding



- W3C Web of Things activity

- WoT architecture
- WoT Thing Descriptions
- WoT Scripting API
- WoT Discovery
- WoT Bindings Templates
- WoT Security and Privacy
- WoT Working Group, Interest Group and Community Group

## Working Group Chairs



Sebastian Käbisch  
Siemens



Michael Koster  
Invited Expert



Michael McCool  
Intel



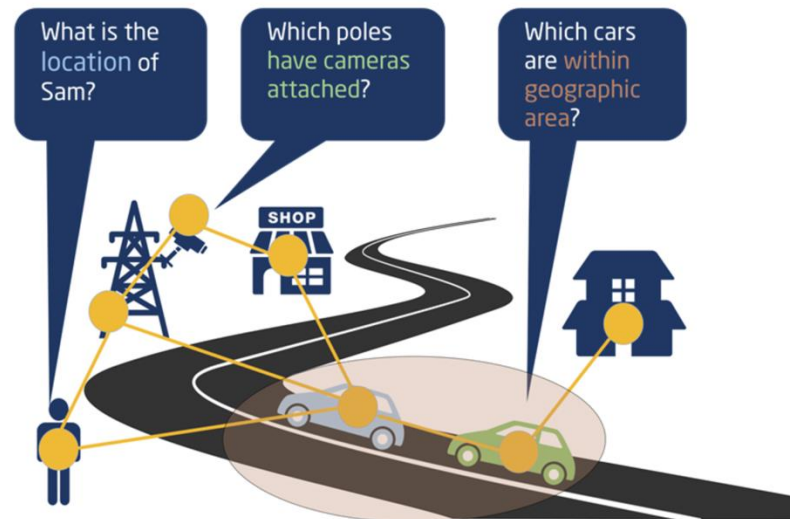
# RDF 1.2 and related standards

- RDF is W3C's suite of standards for the Semantic Web
  - [Scientific American article](#) in May 2001
- Knowledge Graphs modelled in terms of triples:  $\langle \textit{Subject}, \textit{Predicate}, \textit{Object} \rangle^*$
- Global identifiers with IRIs as the basis for standardised vocabulary terms
- RDF Semantics are independent of the serialization, e.g.
  - RDF/XML, Turtle, N-Triples, JSON-LD, N3
- Queries with SPARQL
- Ontologies with RDF schemas and OWL
- Shape constraints with SHACL
- RDF first draft released in 1997
- RDF 1.0 released in 1999
- RDF 1.1 released in 2014
- W3C is now finalising [RDF 1.2](#) which is designed to simplify expressing annotations to triples
  - Use of RDF triples as a triple term in the object position of another triple
  - Backwards compatible with earlier versions of RDF
- RDF 1.2 is motivated in part by the success of labelled property graphs
  - Properties for graph Vertices *and* Edges
- Basis for semantic interoperability for information held in a variety of different formats
  - RDBMS, LPG, CSV, PDF, Spreadsheets, ...

\* There is also a four-place version with an extra parameter for a named graph

# NGSI-LD and RDF

- NGSI-LD is a standard for publishing, querying and subscribing to (IoT) context information via REST protocol

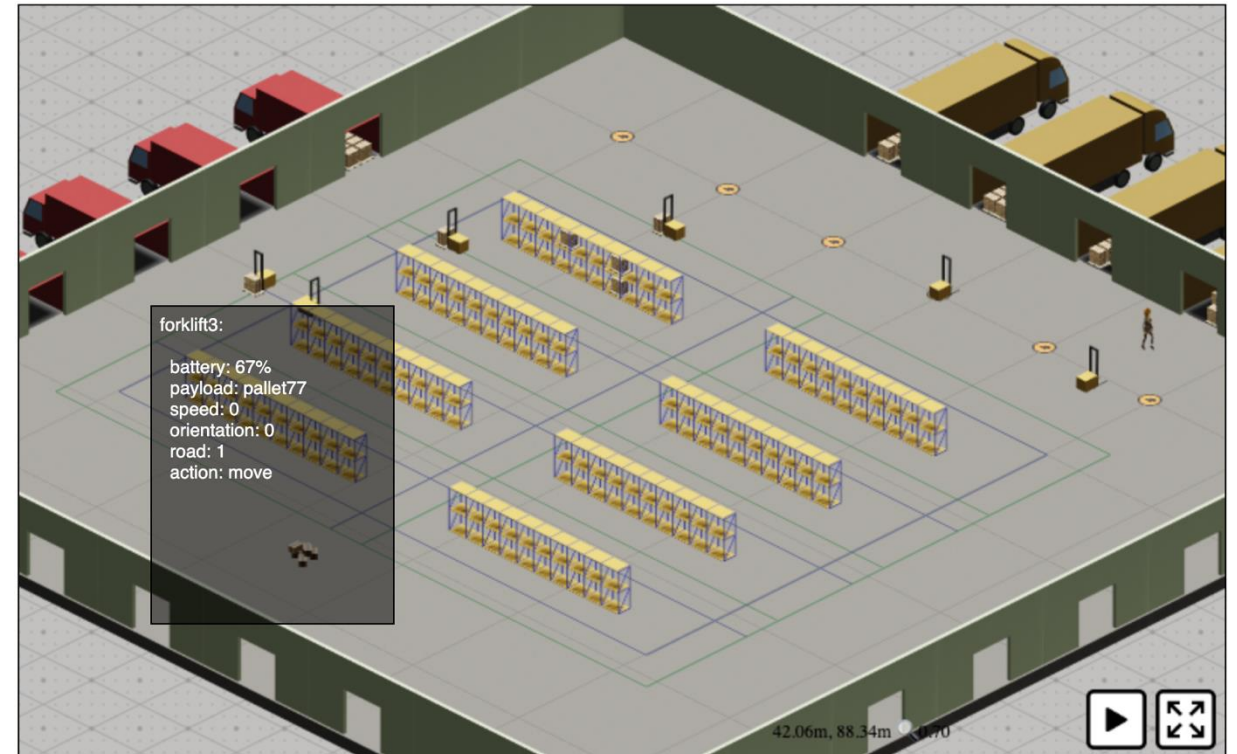


Courtesy of [ngsild.org](http://ngsild.org)

- Standardised by ETSI through [ISG CIM](#), inspired by OMA's NGSI specifications and the FIWARE community
- Semantic interoperability using JSON-LD, but originated using Labelled Property Graphs (LPG)
- W3C and ETSI holding regular liaison meetings to drive alignment with W3C's Web of Things
- Extending WoT Thing Descriptions to support NGSI-LD information servers

# Web-based monitoring, orchestration and simulation

- W3C standards for web browsers enable web-based monitoring, orchestration and simulation
  - HTML5, WebGPU, WebXR, WebNN, ...
- Use web sockets to stream state updates to the web page
  - Local prediction for smooth animation
- 2.5D isometric rendering akin to SimCity with offscreen 3D rendering for robot arms, etc.
- Situation Reports describing what's happening, departures from the plan, and where attention is needed



[SimSwarm](#): a Web-based simulation of a smart warehouse

# Cognitive Approaches to Low-Code control

Robots are current programmed to repeat simple actions over and over again. For greater resilience we need more flexible behaviour that dynamically adapts to changes in the context.

Inspired by decades of work in the cognitive sciences, we can use facts and rules to model agents in terms of event-driven concurrent threads of behaviour. Reasoning is decoupled from real-time control, mimicking the human brain where the conscious mind delegates actions to the cortico-cerebellar circuit that manages large numbers of muscle activations.

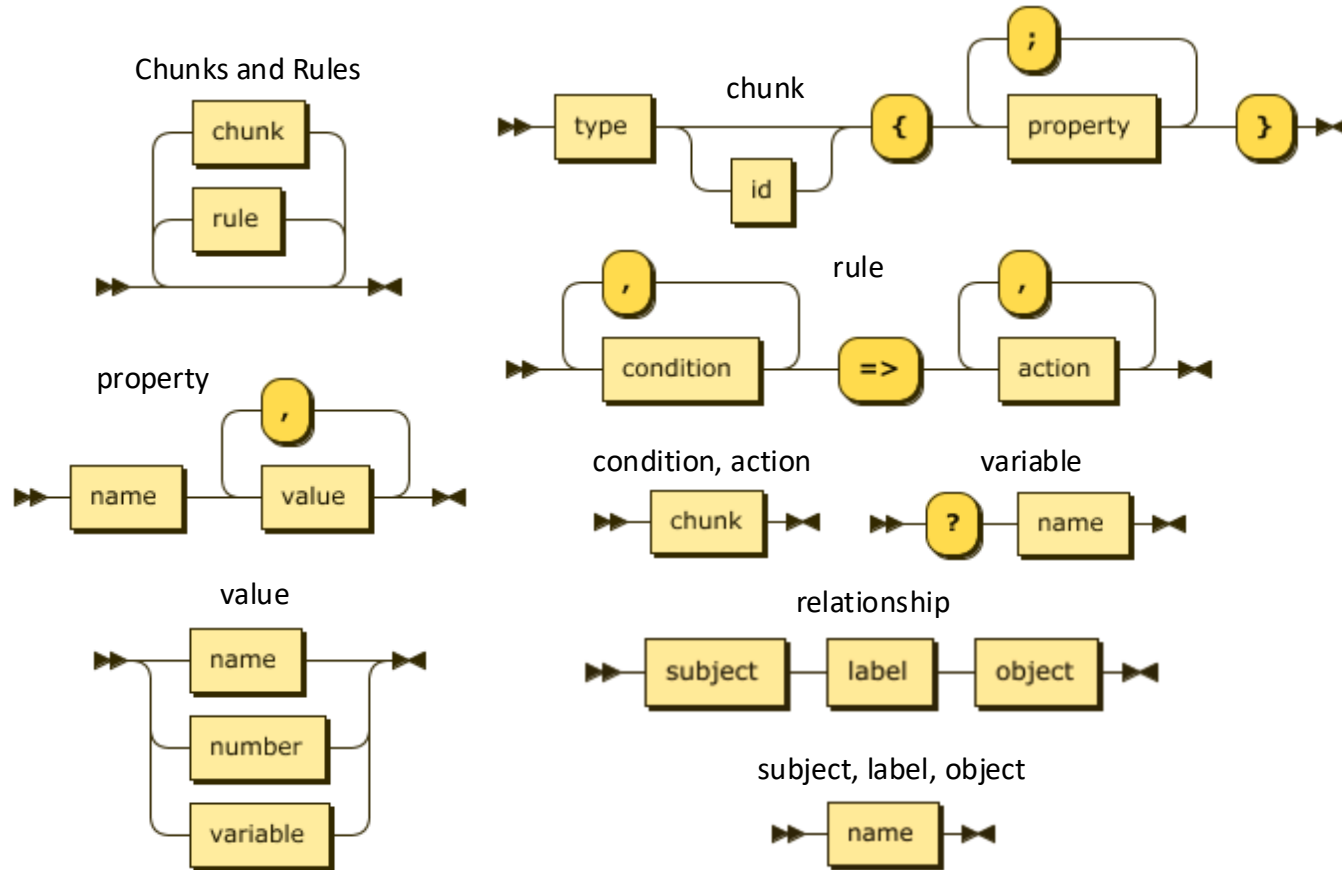
I'd like to acknowledge John R. Anderson, who is a pioneer in the cognitive sciences, and well known for his work at CMU on ACT-R, a computational theory about the operation of the mind for use in practical tests with human subjects.



# Chunks and Rules\*

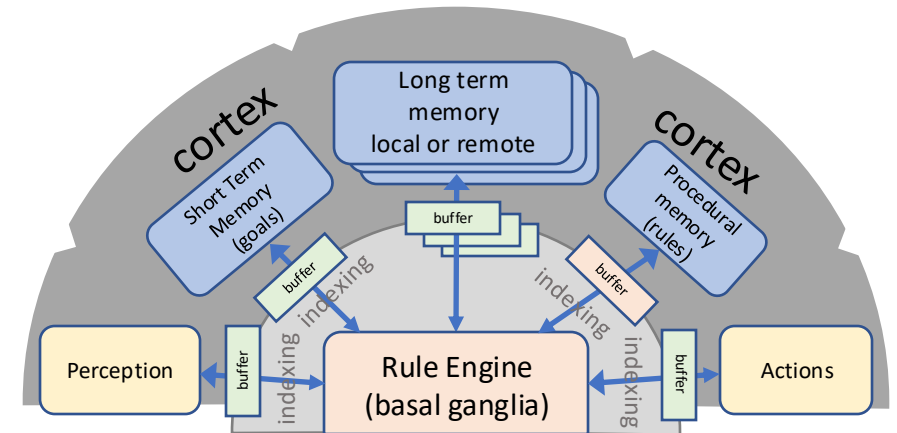
higher level than RDF

Simple syntax as railroad diagrams



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

## Cognition – Sequential Rule Engine



Cognitive Buffers hold single chunks  
Analogy with HTTP request-response model

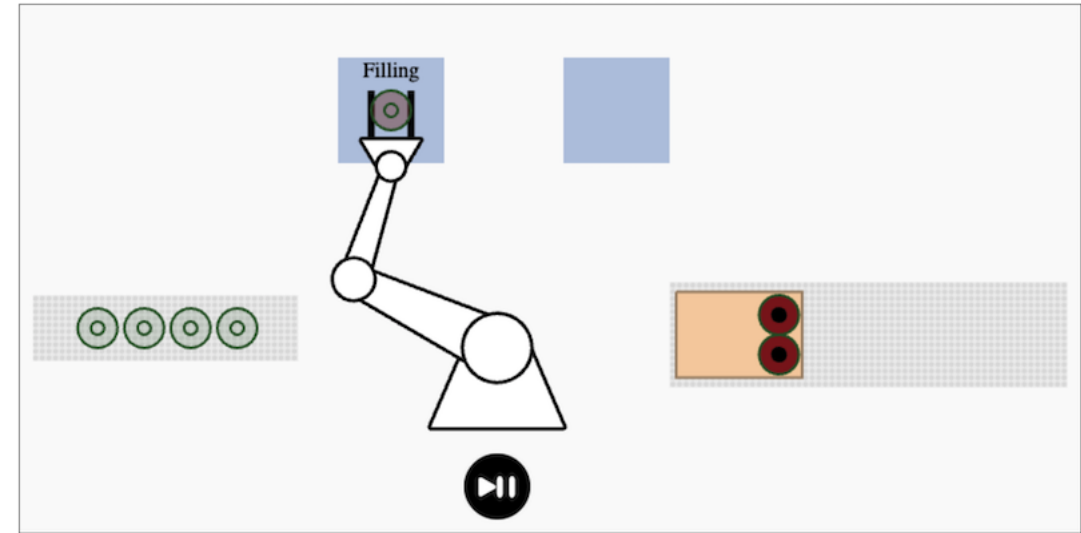
- Inspired by John Anderson's ACT-R and decades of cognitive science research at CMU and elsewhere
- Mimics characteristics of human cognition and memory, including spreading activation and the forgetting curve
- Rule conditions and actions specify which cognitive module buffer they apply to
- Variables are scoped to the rule they appear in
- Actions either directly update the buffer or invoke operations on the buffer's cortical module, which asynchronously updates the buffer
- Predefined suite of built-in cortical operations
- Reasoning decoupled from real-time control over external actions, e.g. a robot arm

\* From the [W3C Cognitive AI Community Group](#). See the [chunks & rules specification](#).

# Chunks and Rules

- Mature [JavaScript library](#) for use in webpages or with NodeJS
  - Along with specification and test suite
  - Fast execution as actions are handled asynchronously
- Application script declares custom operations, e.g. for robot control, layered above ROS operations
  - Implemented in JavaScript using real-time clock as well as networking for external messaging
- Iterative refinement as new requirements come to light, e.g. in unusual situations like faults, or where humans have intervened in an unexpected way
- Stochastic behaviour where needed
  - If several rules match the chunk buffers, then one of them will be randomly selected for execution
  - Rules with same conditions and different actions

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](#)

# Extension to swarm computing

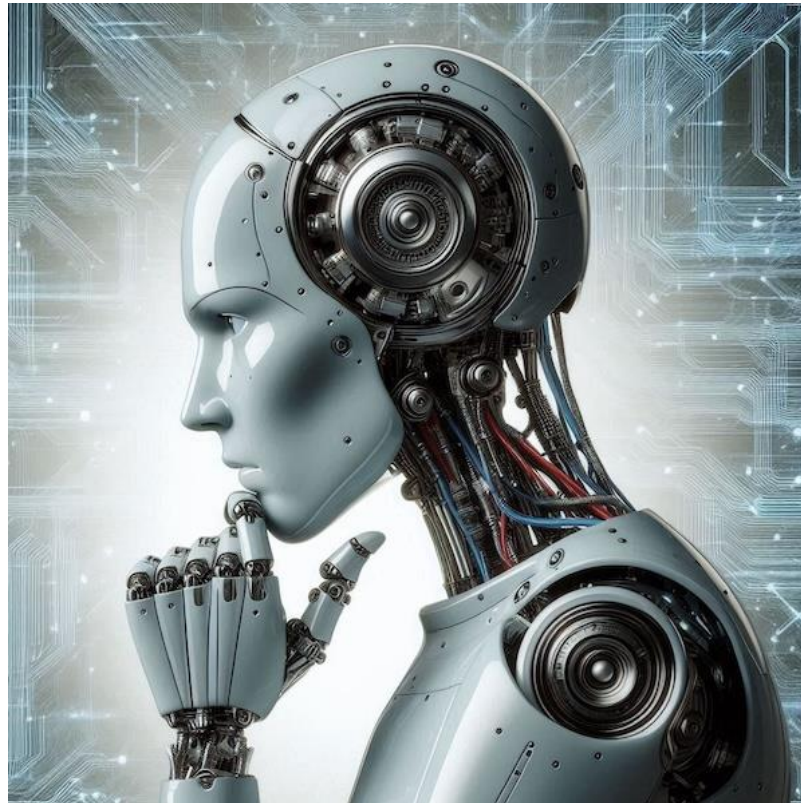
- Each cognitive agent can control multiple devices
  - In the bottling demo, the agent controls two conveyor belts, the robot arm, as well as the filling and capping machines
- But on a larger scale, we need multiple cognitive agents working together to control many devices
- Agents can send each other asynchronous messages as chunks
  - Treat as events to trigger behaviour, or use to update the agent's model of the world
  - Named agents or groups of agents
  - Named topics for interested agents
- Agents send arbitrary chunks using *@message* to identify the recipient, e.g. as in the following rule action

```
# tell agent4 to start task t42
start {@message agent4; task t42}
```
- Messages can be sent with *@topic* to subscribers of named topics, e.g.

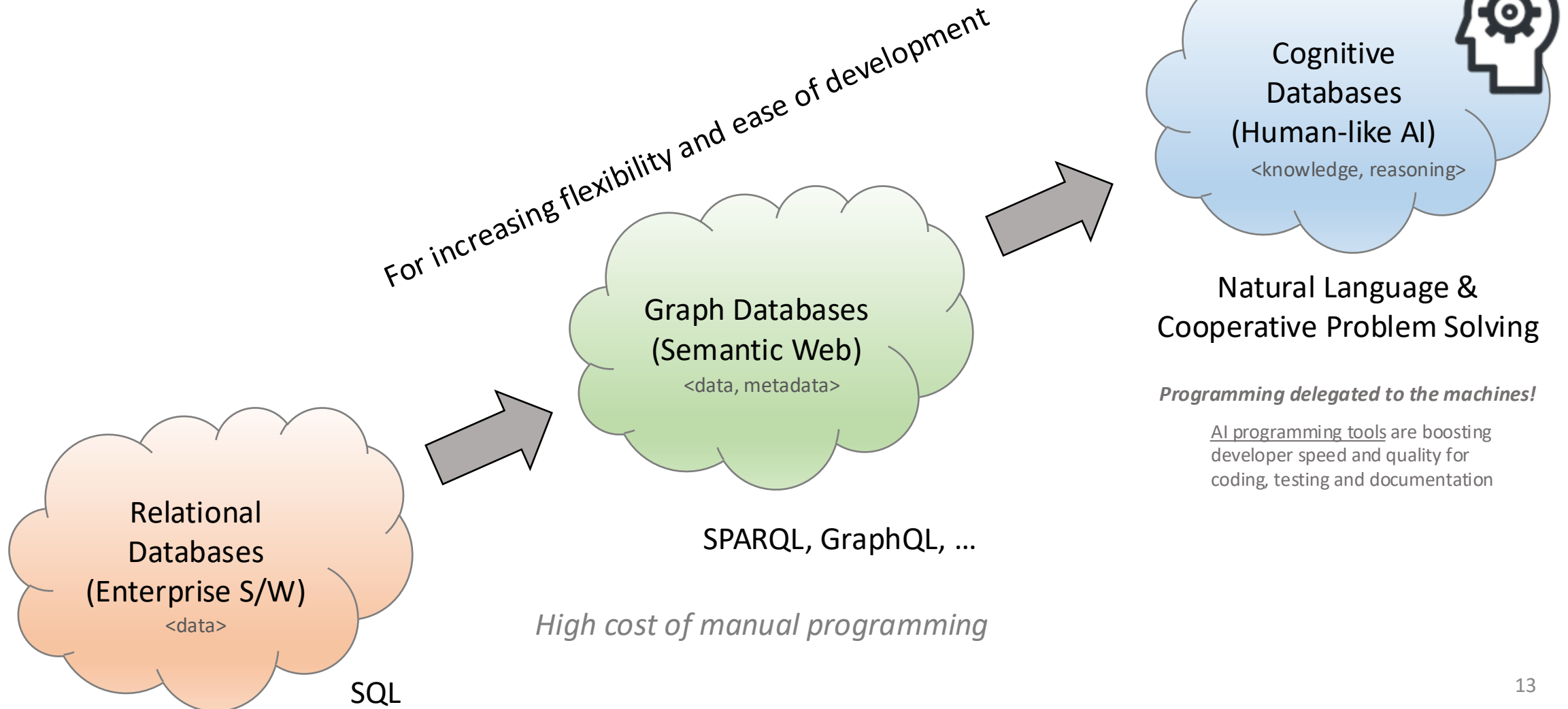
```
# send stop message on topic12
stop {@topic topic12}
```
- Agents can subscribe to a topic with *@do subscribe*, e.g.

```
# subscribe to topic12
listen {@subscribe topic12}
```

# Where next with AI?



# Evolution in ICT Systems



# Quantum Leap Forward from Generative AI to Sentient AI

- Generative AI is now being applied to implement agents
  - e.g. using [langchain](#) framework together with PyTorch
- Enables high level control, e.g. with spoken or written directions in natural language
- Can be used for high level instructions for supervising simpler robot control systems
  - Overcoming high cost of manually programming robot behaviour
- **Generative AI** is powerful but has known limitations, e.g.
  - Lacks ability to acquire new skills once deployed, and is prone to making things up (hallucination)
- **Sentient AI** adds *episodic memory*
  - Continual learning through continual prediction and continual reasoning
  - Agents with awareness of their environment, goals and performance, along with the means to remember and reflect on their past experiences
  - Research is now needed on the technical challenges to realise this breakthrough

*Neural networks have shown the huge power of statistical approaches to AI*  
Symbols as a crude approximation to a more profound statistical model of knowledge



**Comments and Questions?**