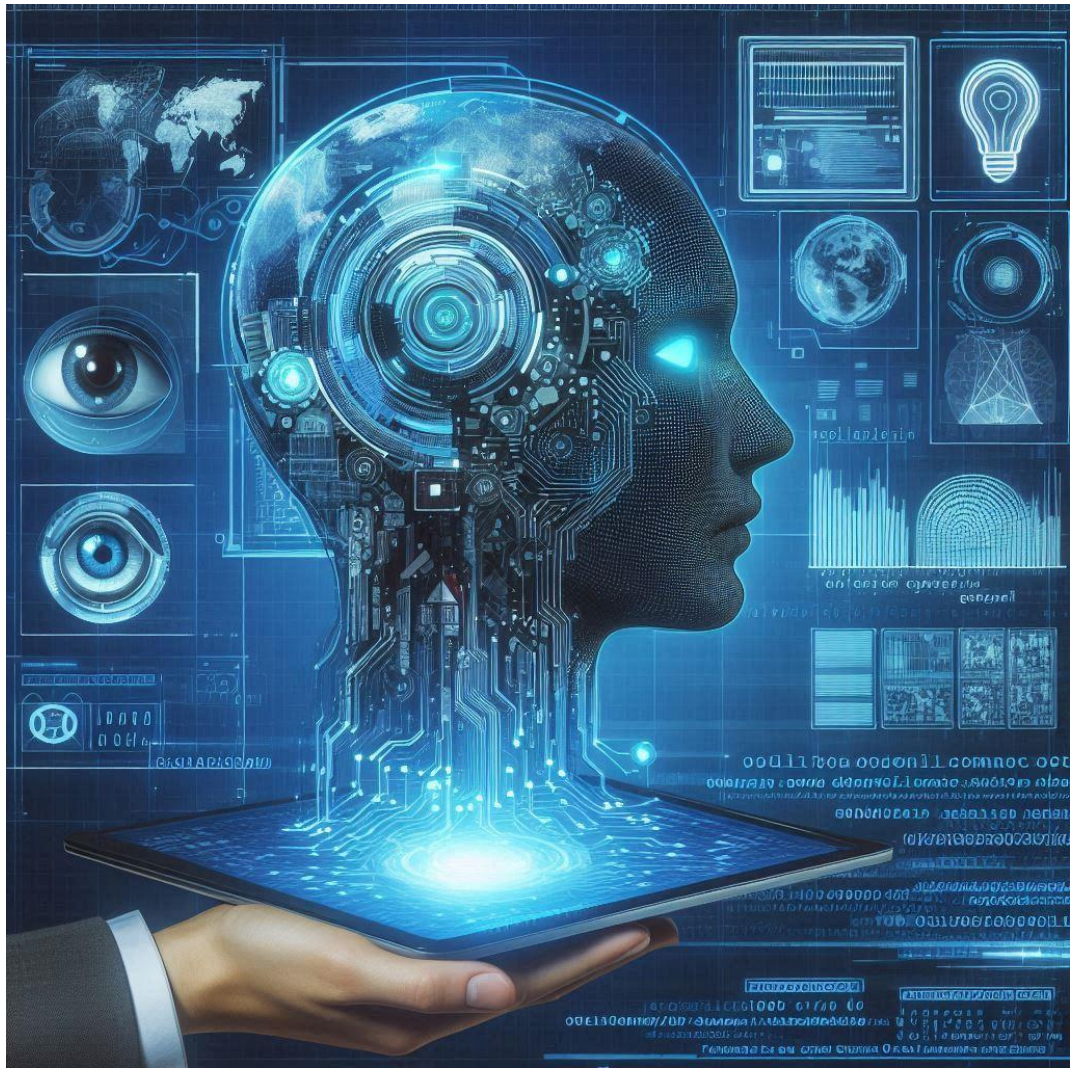
Chunks and Rules for Cognitive Control



Grateful acknowledgement of support from the Nephele and SmartEdge projects under the EU Horizon Europe programme

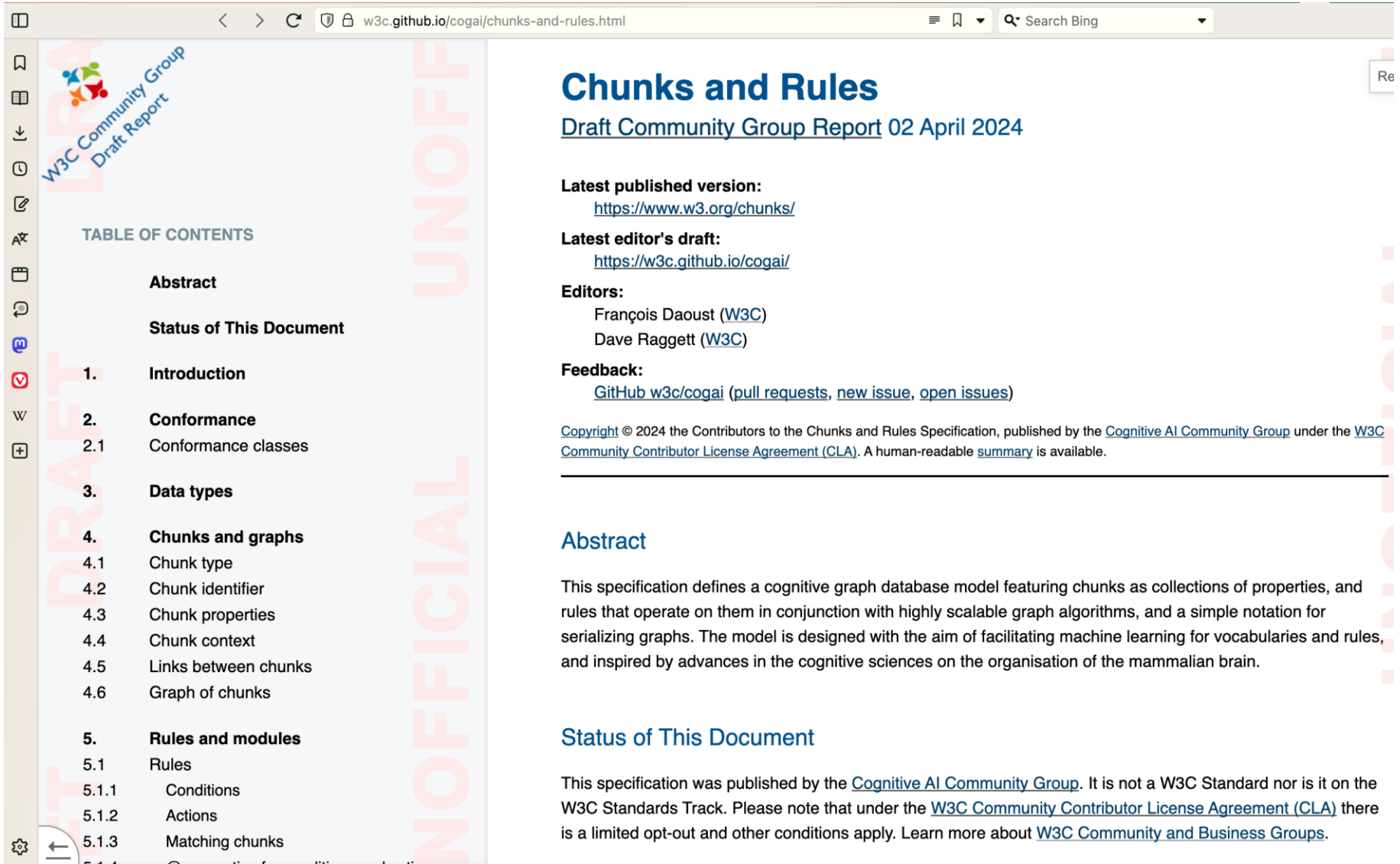
Cognitive Approach to Low-Code Control

- *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
  - Enabling resilience and adaptability
- 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

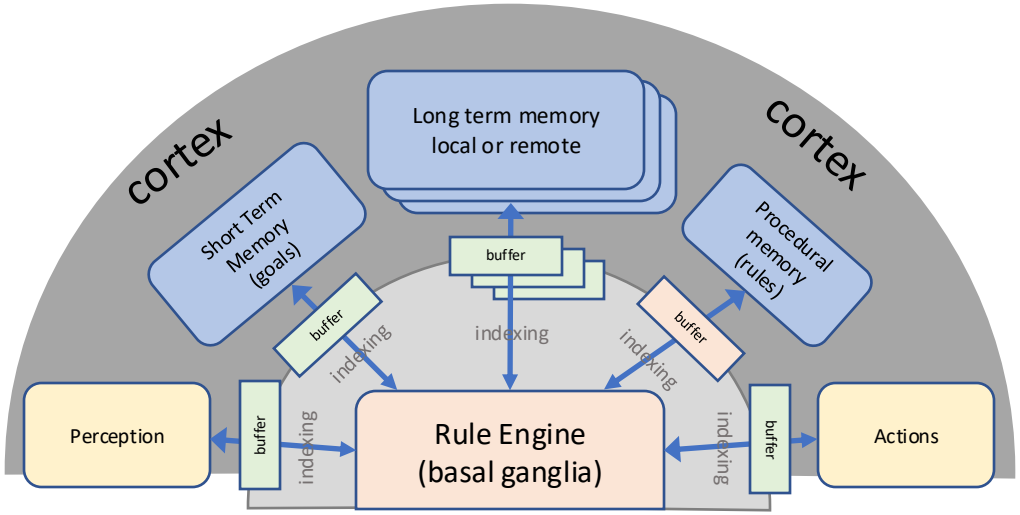
Formal Specification from Cognitive AI CG



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

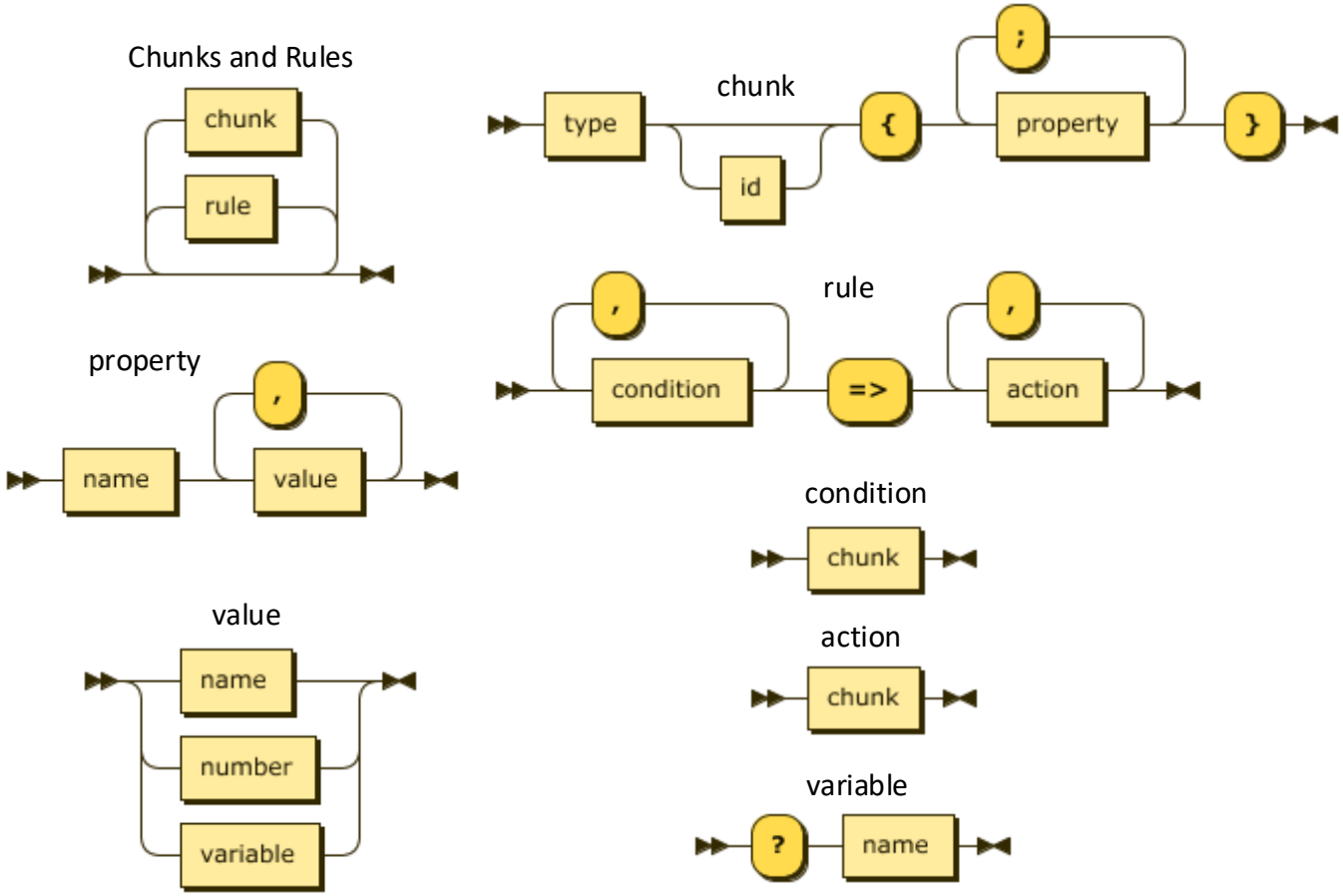


- The cortex holds a set of cognitive modules, each of which is associated with a module buffer that holds a single chunk
- Predefined asynchronous operations on buffers in analogy with REST

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

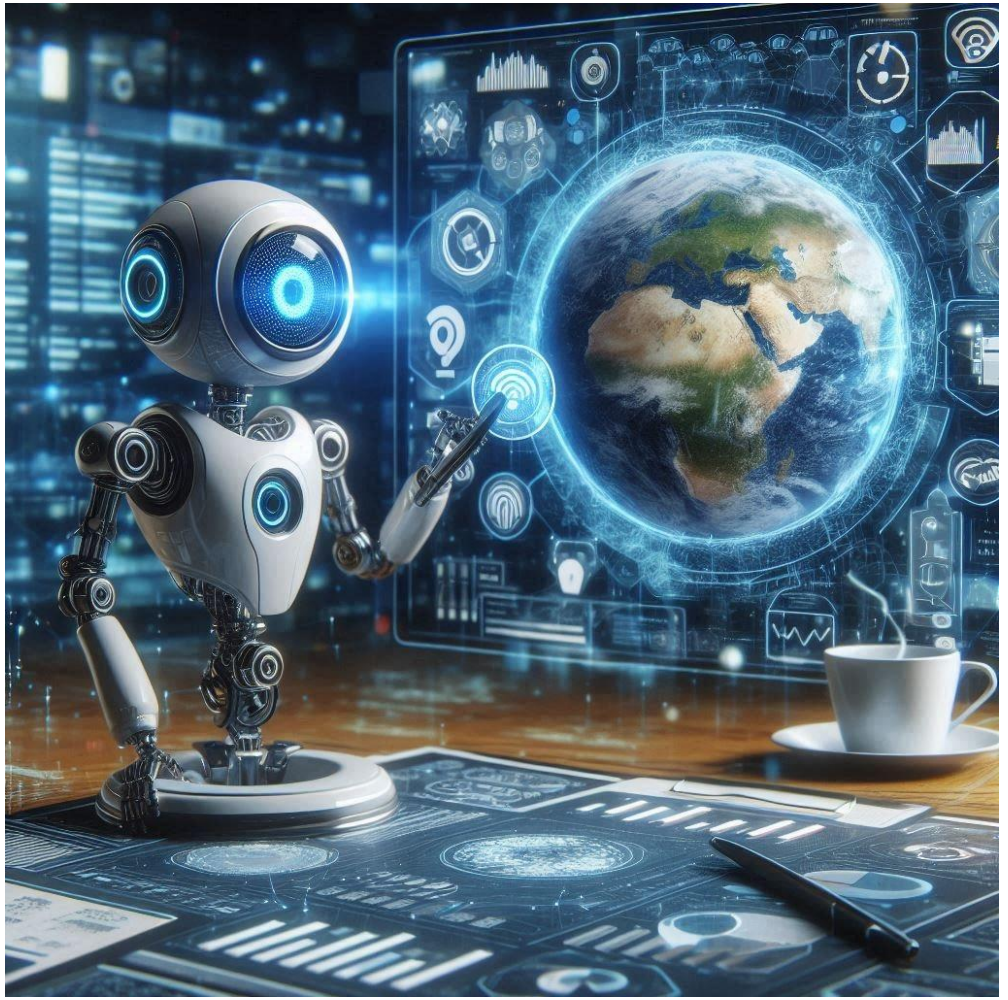


names beginning with "@" are reserved, e.g. @do for actions

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*
- Digital Twin affordances (properties, actions, events) and semantics are described using W3C Thing Descriptions, and searchable in Thing Description Directories
- 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

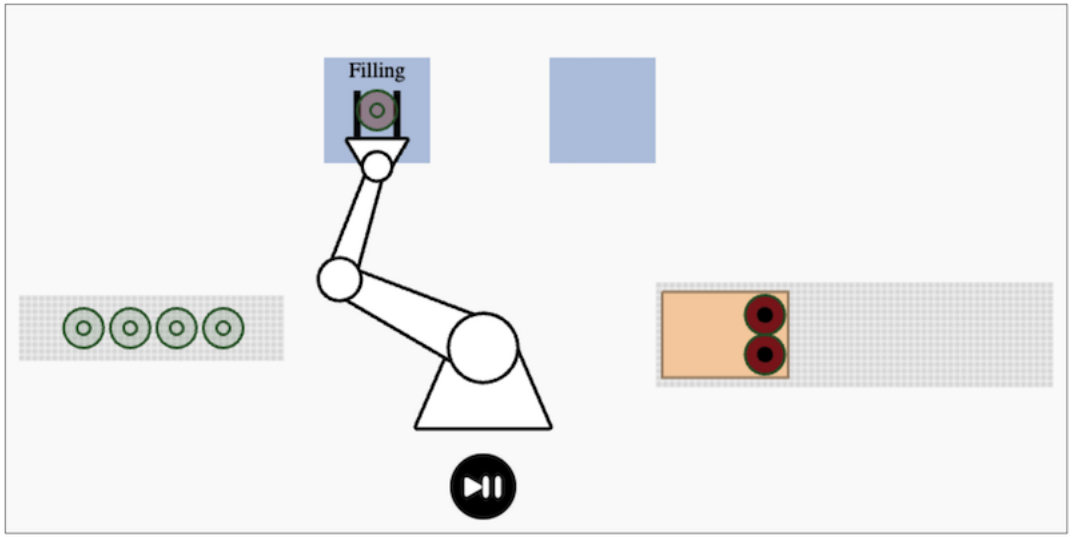


* Digital twins for use in healthcare applications, and for virtual devices as abstractions over multiple physical devices (i.e. composite virtual objects)

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 invoke ROS services
  - Delegation for planning and execution
- Existing JavaScript libraries for integration with ROS

Swarm Intelligence

- A single cognitive agent may be used to control multiple devices
  - Factory demo: a single chunks & rules agent controls two conveyor belts, one robot arm, a filling station and a capping station
- Cognitive agents can fuse information from local and remote sources for situational awareness
  - Not limited to on-device sensors
- Cognitive agents can share information that other agents may find useful*
  - e.g. unexpected obstacles
- You can think of the swarm as a hive mind composed from multiple communicating cognitive agents
  - Asynchronous chunk messages
- Increased resilience and flexibility

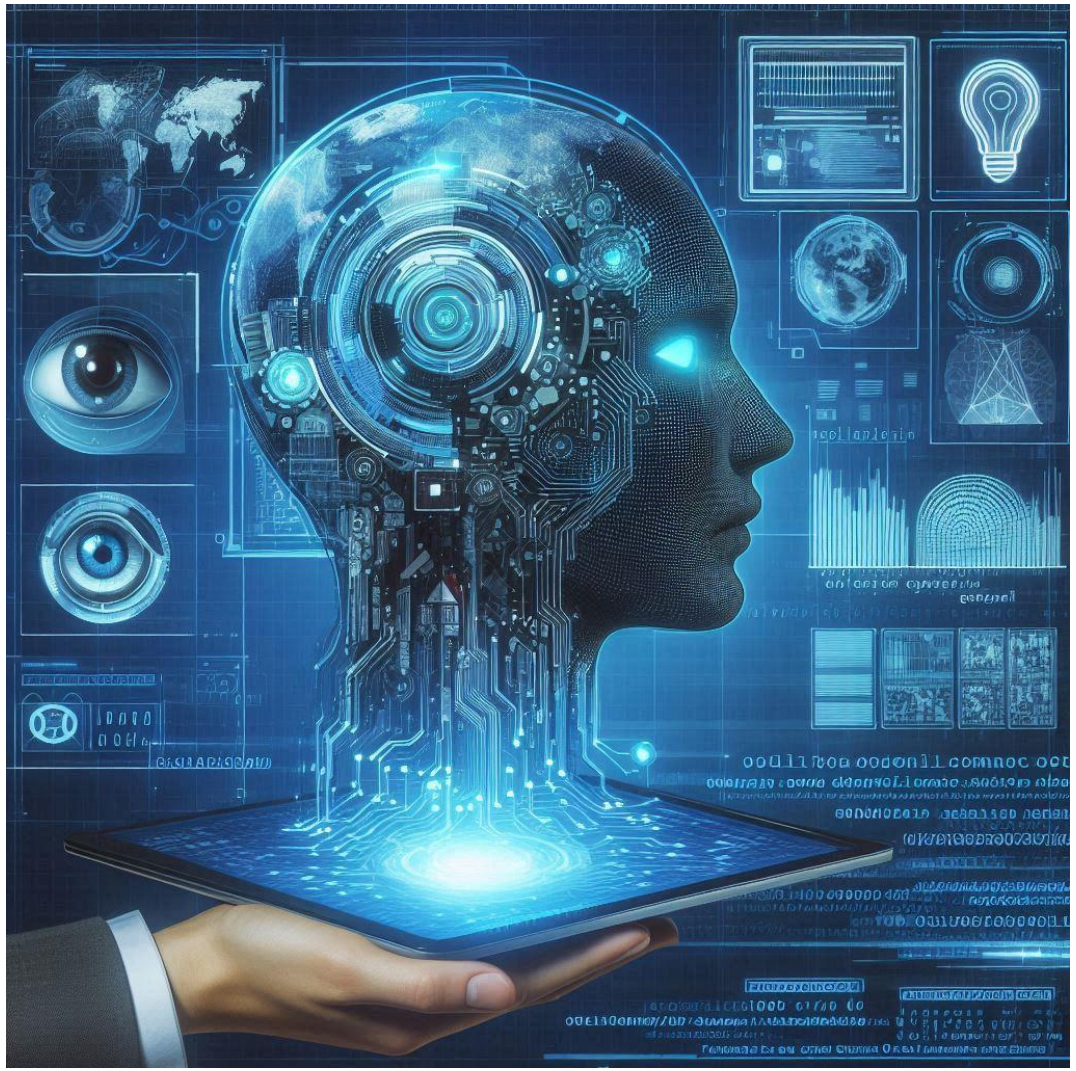
Agents can leave information in the physical or virtual environment as a form of *stigmergy*. This is a mechanism of indirect coordination through the environment, between agents or actions. The principle is that the trace left in the environment by an individual action stimulates the performance of a succeeding action by the same or different agent. Agents can also communicate directly, e.g. when they are in close proximity, or via their names or roles.

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?



Grateful acknowledgement of support from the Nephele and SmartEdge projects under the EU Horizon Europe programme