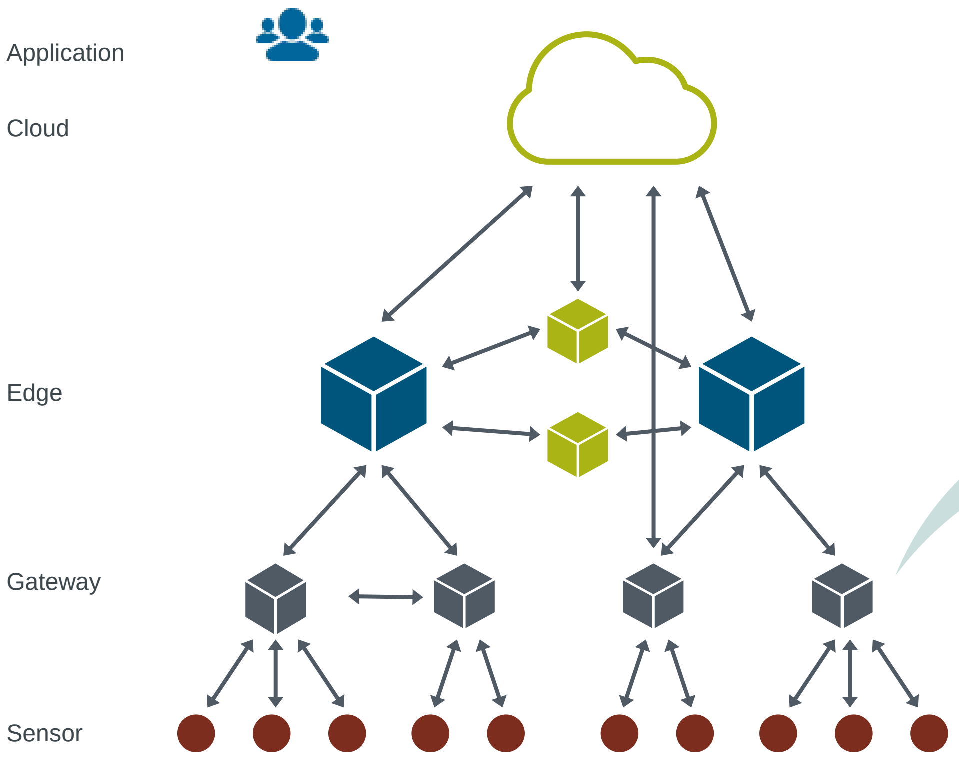
Coaty
A Framework for Collaborative IoT

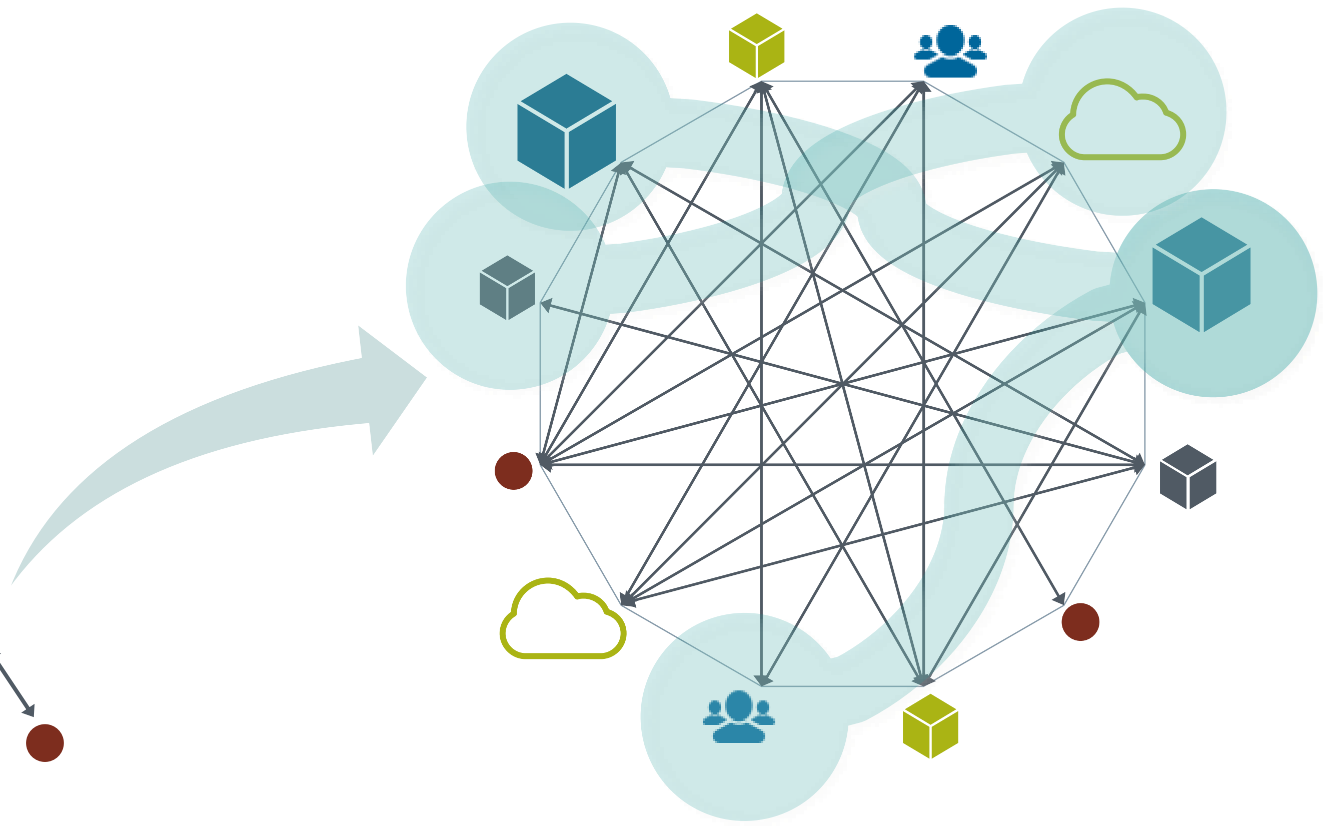
Siemens Technology Research

SIEMENS

Market shows a clear trend towards systems collaborating independently and autonomously as self-organizing system of systems



cloud-centric - hierarchical
IoT network



decentralized - non-hierarchical
IoT network

SIEMENS

Demand of collaborative smart autonomous systems identified across all major industrial domains

Industry

Energy

Mobility

Building

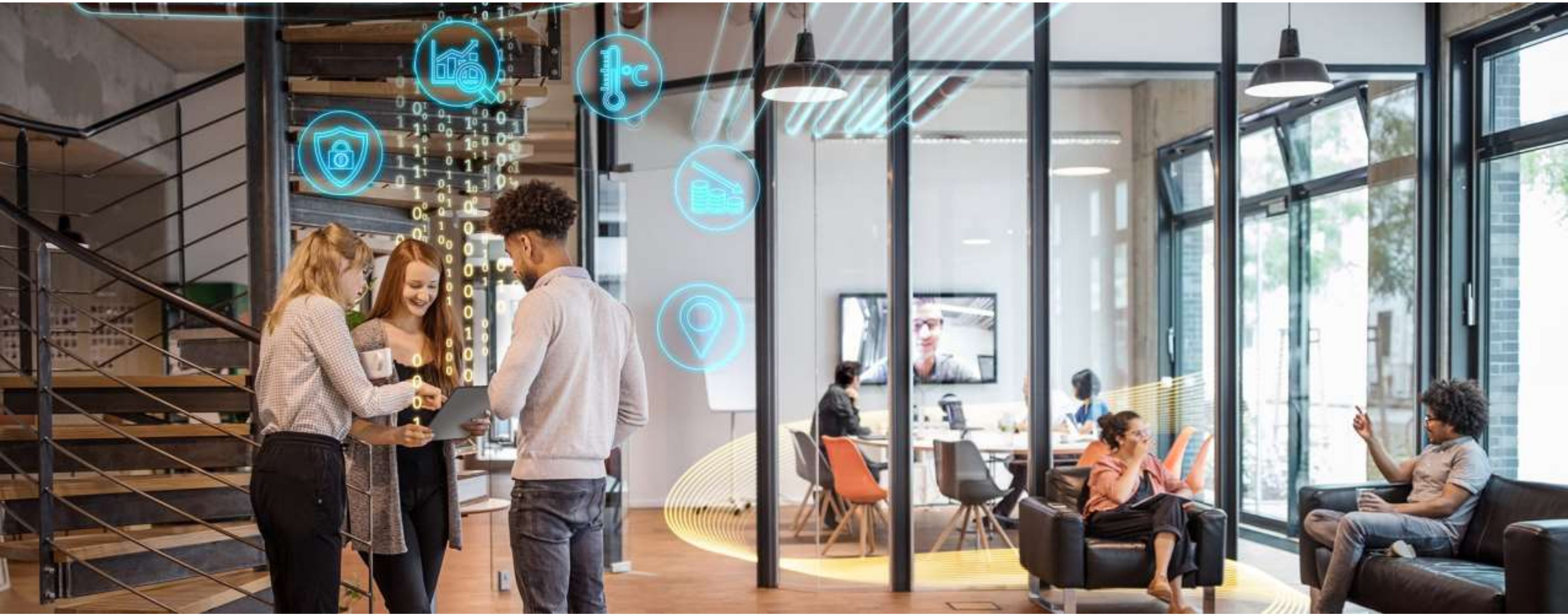
Logistics

“ **Autonomous Agents and Things**
Over the next five years we will evolve to a post-app world, with intelligent agents delivering dynamic and contextual actions and interfaces. ”

Source: Gartner, © 2018 Gartner, Inc. and/or its affiliates. All rights reserved

SIEMENS

Most prominent autonomous ’Systems’ acting as ’System of Systems’
Interacting and collaborating humans



SIEMENS

Designing a collaboration framework for smart autonomous systems

Interaction of Smart Autonomous Systems

Loose Coupling

- Event-based interaction of system components
- Data-centric not device-centric paradigm
- Publish/Subscribe messaging

Any-to-Any Communication

- Communication patterns on top of pub/sub transport layer
- One-way (pub/sub) and two-way (request/response) patterns
- Many-to-many communication for different types of interaction

Programmability

- Modular and extendable software framework
- Asynchronous event handling through reactive programming
- Building blocks

Collaboration Functions

- Context specific routing of information flows
- Negotiation and delegation mechanisms
- Consensus finding mechanisms

Cross-Platform Deployments

- Extensibility by defined communication protocol based on standards
- Availability for multiple types of deployment

SIEMENS

Coaty - The framework for collaborative IoT

1. Implementation of **interaction and communication foundation** for **smart autonomous systems** in **distributed, decentralized applications**

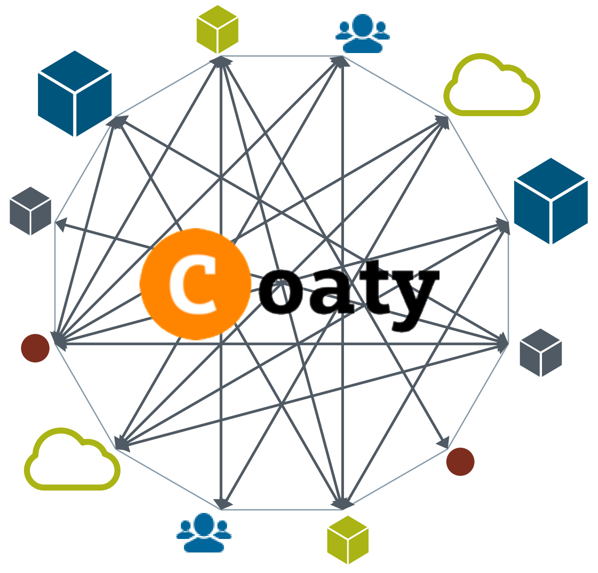
2. Provides **software framework** for **data-centric agent interaction** with loosely coupled systems, any-to-any communication, and smooth **handling** of **asynchronous events**

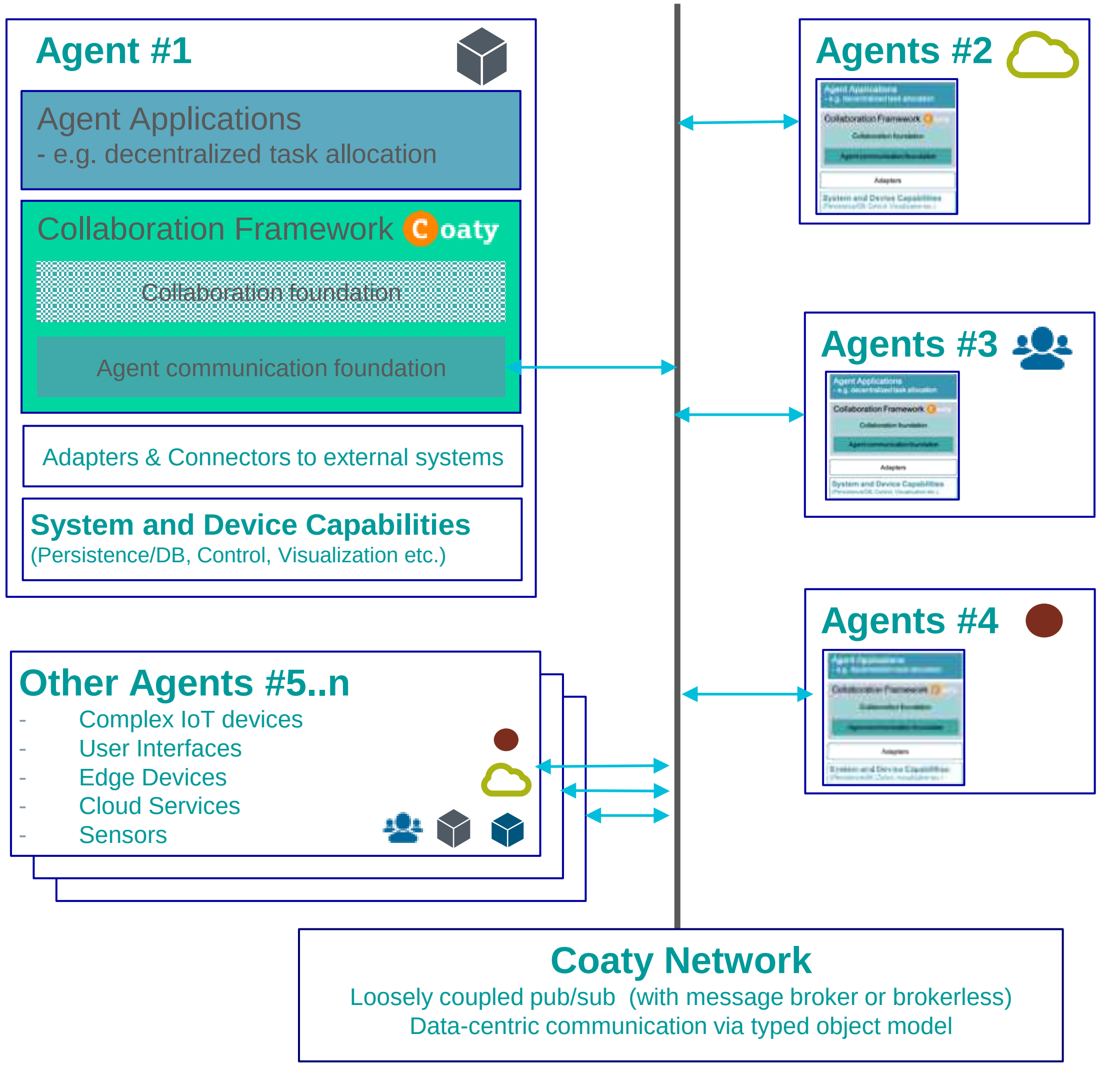
3. Provides **collaboration capabilities** in a **middleware layer** on top of transport protocols and OS layer / stacks

4. **Applicable** to the **full** scale **of potential deployments** of agents from Cloud, Edge, Smartphones, Wearables, Browser, etc.

5. **Open Source** framework powered by Siemens (https://coaty.io)

High-level system design
Collaborative application based on Coaty

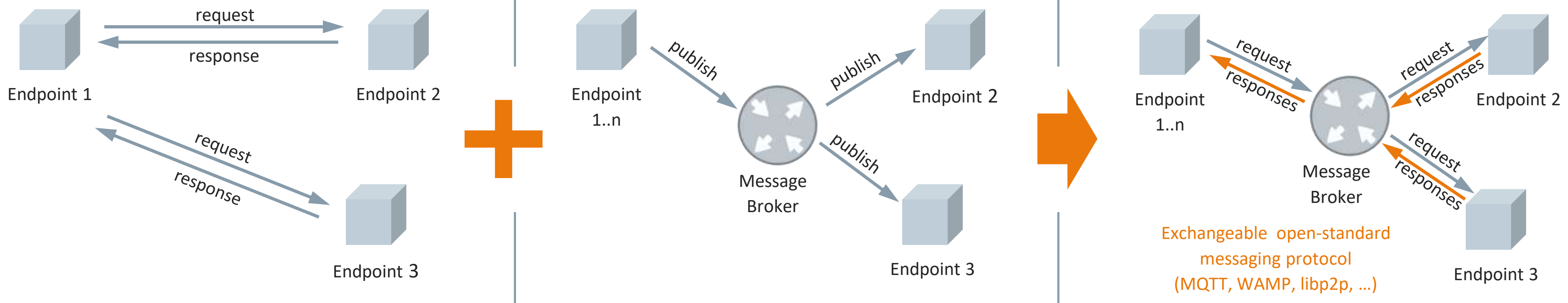



Agent #1
Agent Applications
- e.g. decentralized task allocation
Collaboration Framework Coaty
Collaboration foundation
Agent communication foundation
Adapters & Connectors to external systems
System and Device Capabilities
(Persistence/DB, Control, Visualization etc.)
Other Agents #5..n
- Complex IoT devices
- User Interfaces
- Edge Devices
- Cloud Services
- Sensors
Agents #2
Agents #3
Agents #4
Coaty Network
Loosely coupled pub/sub (with message broker or brokerless)
Data-centric communication via typed object model

SIEMENS

Coaty – Communication foundation

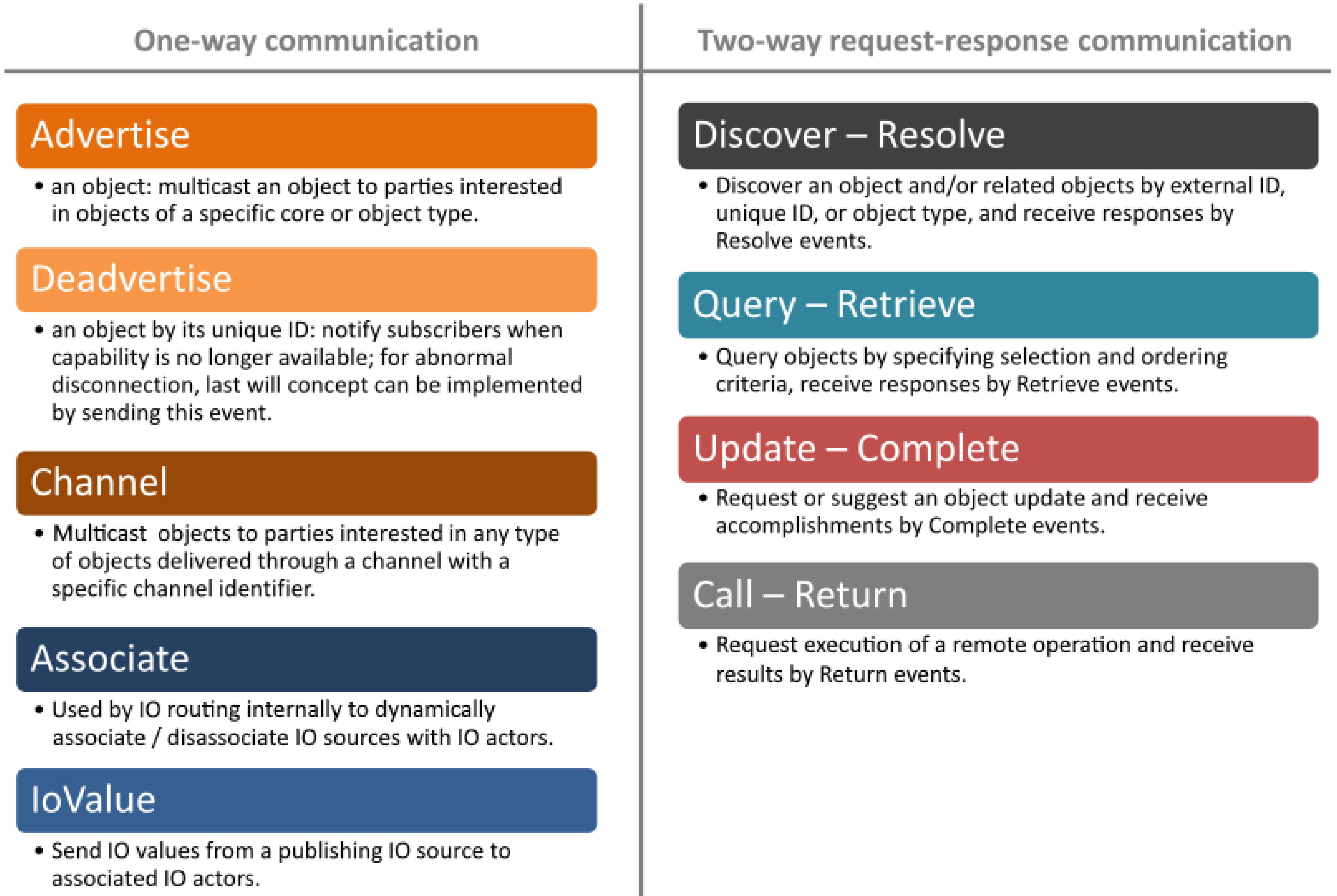
Loose coupling of systems and data centric paradigm



Request-Response (e.g. REST, RPC)	**Publish-Subscribe Messaging (e.g. MQTT, WAMP, libp2p)**	**Coaty**
Two-way communication	No two-way communication	**Two-way communication**
One response per request	No response	**Multiple responses per request over time** (even from the same endpoint)
No many-to-many communication	**Many-to-many communication**	**Many-to-many communication**
Strongly coupled endpoints (endpoints know about each other)	**Loosely coupled endpoints** (endpoints do not know about each other)	**Loosely coupled endpoints** (endpoints do not know about each other)

SIEMENS

Coaty – Communication event patterns for system interaction



One-way communication

Advertise

- an object: multicast an object to parties interested in objects of a specific core or object type.

Deadvertise

- an object by its unique ID: notify subscribers when capability is no longer available; for abnormal disconnection, last will concept can be implemented by sending this event.

Channel

- Multicast objects to parties interested in any type of objects delivered through a channel with a specific channel identifier.

Associate

- Used by IO routing internally to dynamically associate / disassociate IO sources with IO actors.

IoValue

- Send IO values from a publishing IO source to associated IO actors.

Two-way request-response communication

Discover – Resolve

- Discover an object and/or related objects by external ID, unique ID, or object type, and receive responses by Resolve events.

Query – Retrieve

- Query objects by specifying selection and ordering criteria, receive responses by Retrieve events.

Update – Complete

- Request or suggest an object update and receive accomplishments by Complete events.

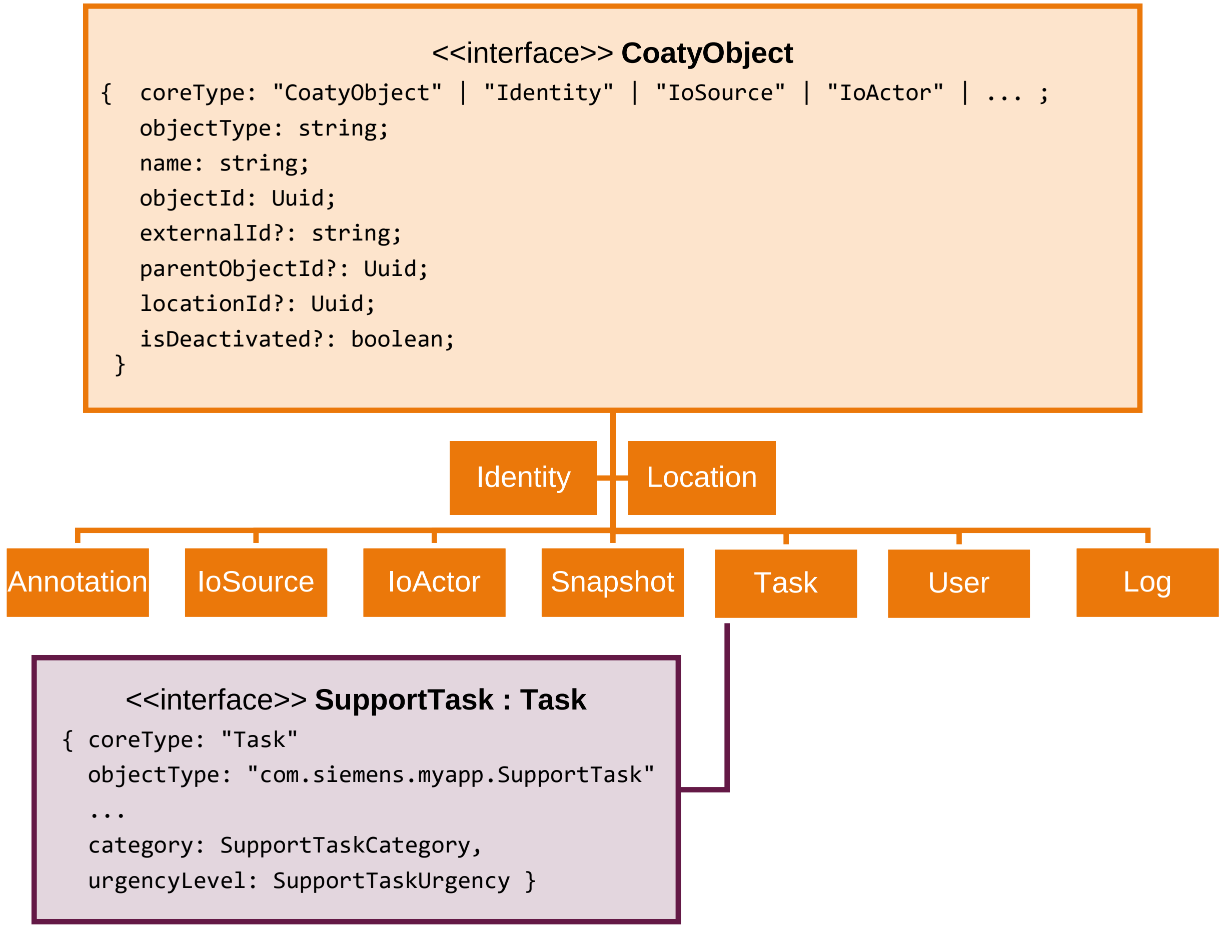
Call – Return

- Request execution of a remote operation and receive results by Return events.

SIEMENS

Coaty object model

An opinionated set of core object types to be used or extended by applications



<<interface>> **CoatyObject**
Identity
Location
Annotation
IoSource
IoActor
Snapshot
Task
User
Log
<<interface>> **SupportTask : Task**

```
{  coreType: "CoatyObject" | "Identity" | "IoSource" | "IoActor" | ... ;
   objectType: string;
   name: string;
   objectId: Uuid;
   externalId?: string;
   parentObjectId?: Uuid;
   locationId?: Uuid;
   isDeactivated?: boolean;
 }
```

```
{ coreType: "Task"
  objectType: "com.siemens.myapp.SupportTask"
  ...
  category: SupportTaskCategory,
  urgencyLevel: SupportTaskUrgency }
```

Supports discovery, distribution, sharing, and persistence

- Objects consist of attribute-value pairs that model state but no behavior
- Objects are uniquely identified without central coordination by a Version 4 UUID
- Cross-component, cross-platform representation in JSON format
- Object types form a hierarchy defined by Interfaces
- Framework-supplied core object types are extensible by applications
- Communicated object shape is schema validated against interface definition
- Schemaless persistence in NoSQL and SQL data stores

SIEMENS

Coaty event pattern example – Discover-Resolve

“Discover information for an external ID encoded in a QR code”

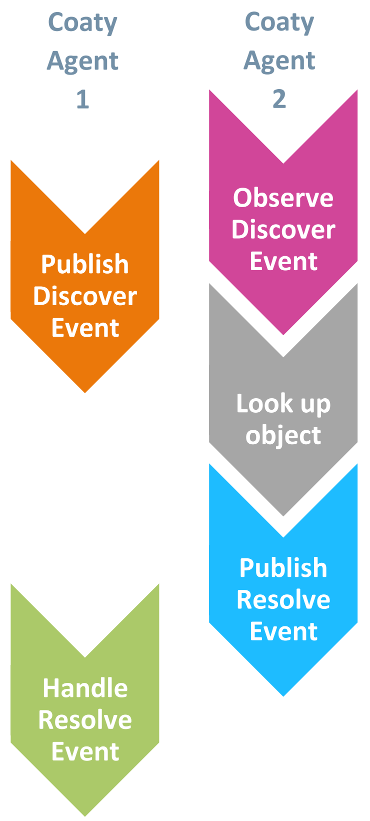
Coaty Agent 1

```
// QR Code of asset
const externalId = "00000042";

// Publish a Discover event and observe first Resolve event response
this.communicationManager
    .publishDiscover(DiscoverEvent.withExternalId(this.identity, externalId))
    .pipe(first(), map(event => event.eventData.object), timeout(5000))
    .subscribe(
            object => {
               // Handle object of Resolve response event
            },
            error => {
               // No response has been received within the given timeout period
               this.logError(error, "Failed to discover external ID");
            });
```

Coaty Agent 2

```
// Observe Discover events and respond with a Resolve event
this.communicationManager
    .observeDiscover(this.identity)
    .pipe(filter(event => event.eventData.isDiscoveringExternalId))
    .subscribe(event => {
            // Agent-specific lookup of an object with given external ID.
            const object = findObjectWithExternalId(event.eventData.externalId);
            // Respond with found object in Resolve event
            event.resolve(ResolveEvent.withObject(this.identity, object));
});
```



Collaborative IoT applications
Two examples

Dynamic Context-Based Information Routing

Dynamically associate information source and actors based on context information

Example Use Cases:

- Multi-device HMIs
- Information routing between collaborating machines in manufacturing
- Smart grid information routing based on physical reconfiguration

Resource Allocations in Resource Networks

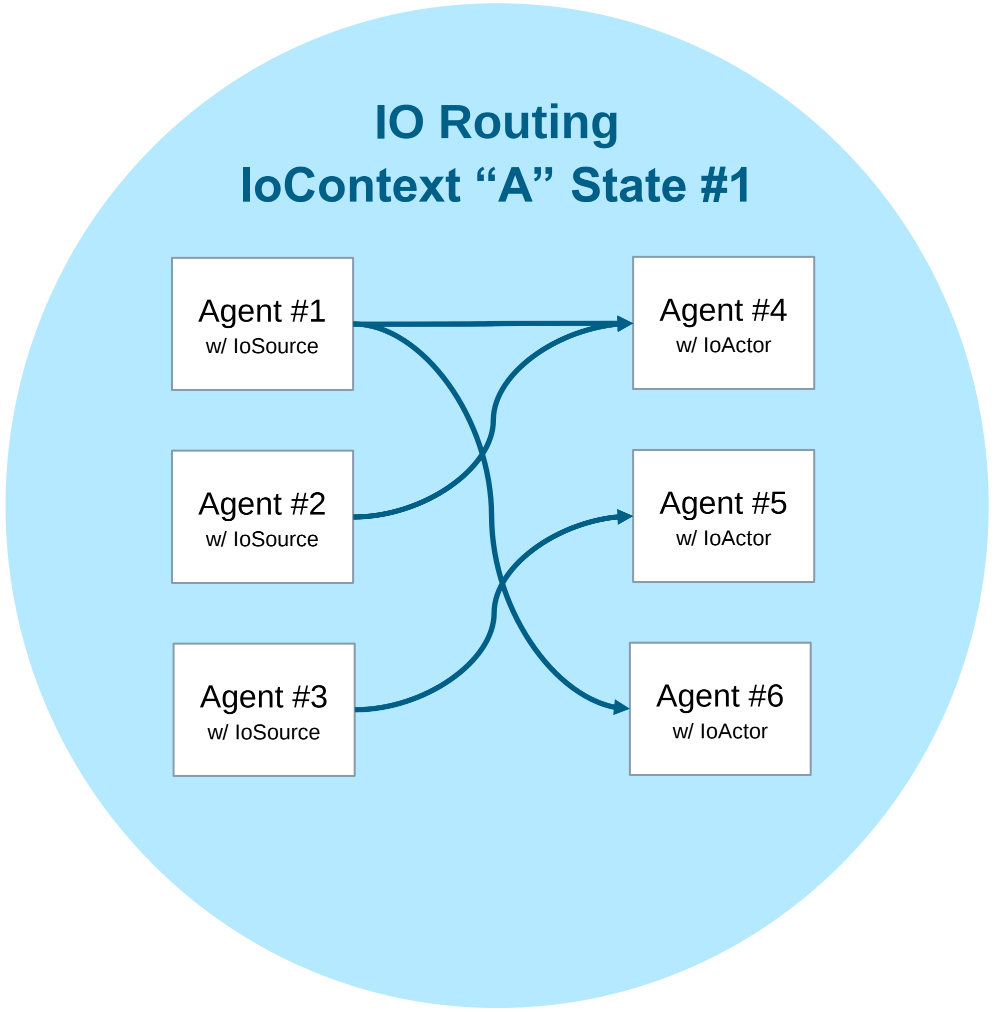
Dynamic decentralized resource negotiation and allocation in distributed systems

Example Use Cases:

- Transport job negotiation of AGVs in self-organizing production and ware-house logistics with heterogeneous fleets
- Self organizing fleet management with a maximum of flexibility and scalability

SIEMENS

Dynamic context-based information routing
Coaty IO Routing

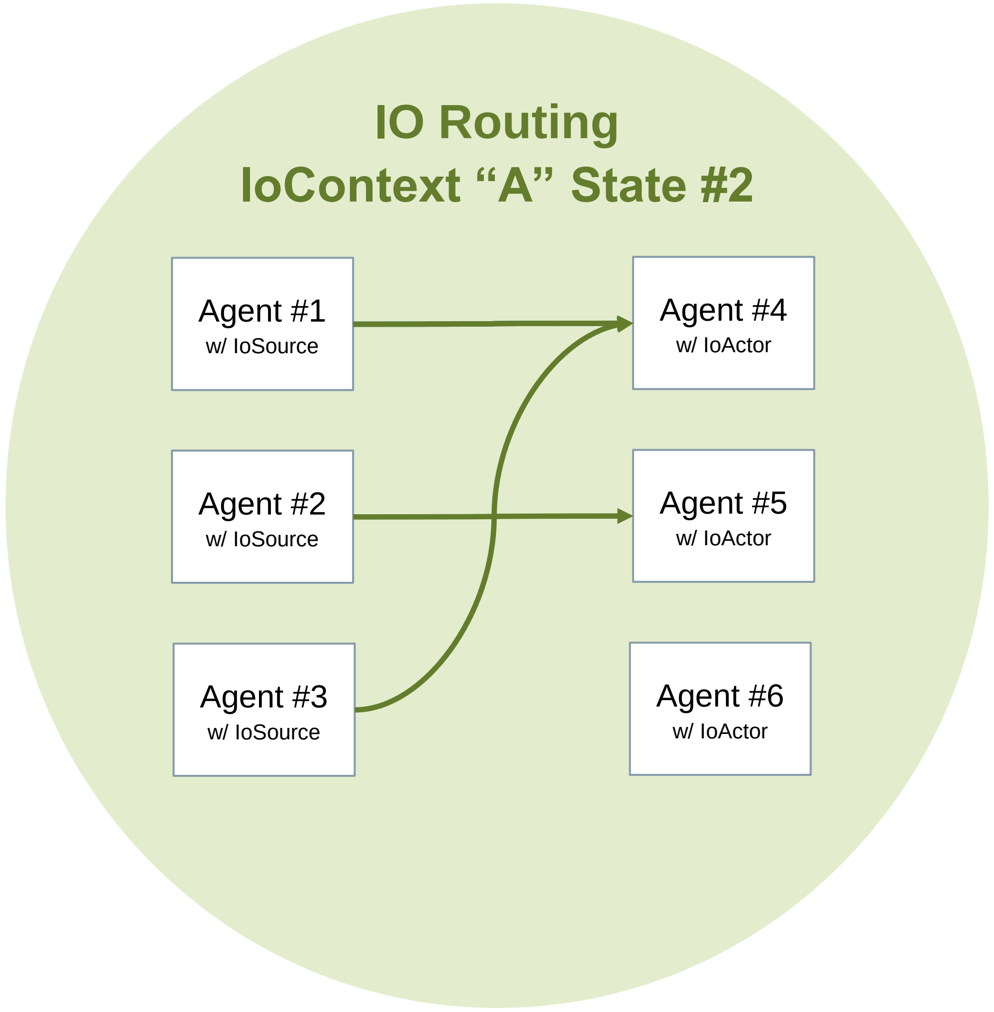


- Coaty agents can have any number of `IoSource` (Information Source) and `IoActor` (Information Consumer).
- `IoSource` and `IoActor` are part of a named shared, distributed `IoContext`.
- An IO Routing component manages information routing for one `IoContext` based on a rule engine; application-defined rules determine the association between `IoSource` and `IoActor` of the different agents.
- On `IoContext` state changes, the IO Routing component uses Coaty ASSOCIATE event pattern to update publication and subscription topics of `IoSources` and `IoActors` of the agents.

Note: Detailed implementation example available in Coaty 2.0 developer guides

SIEMENS

Dynamic context-based information routing
Coaty IO Routing

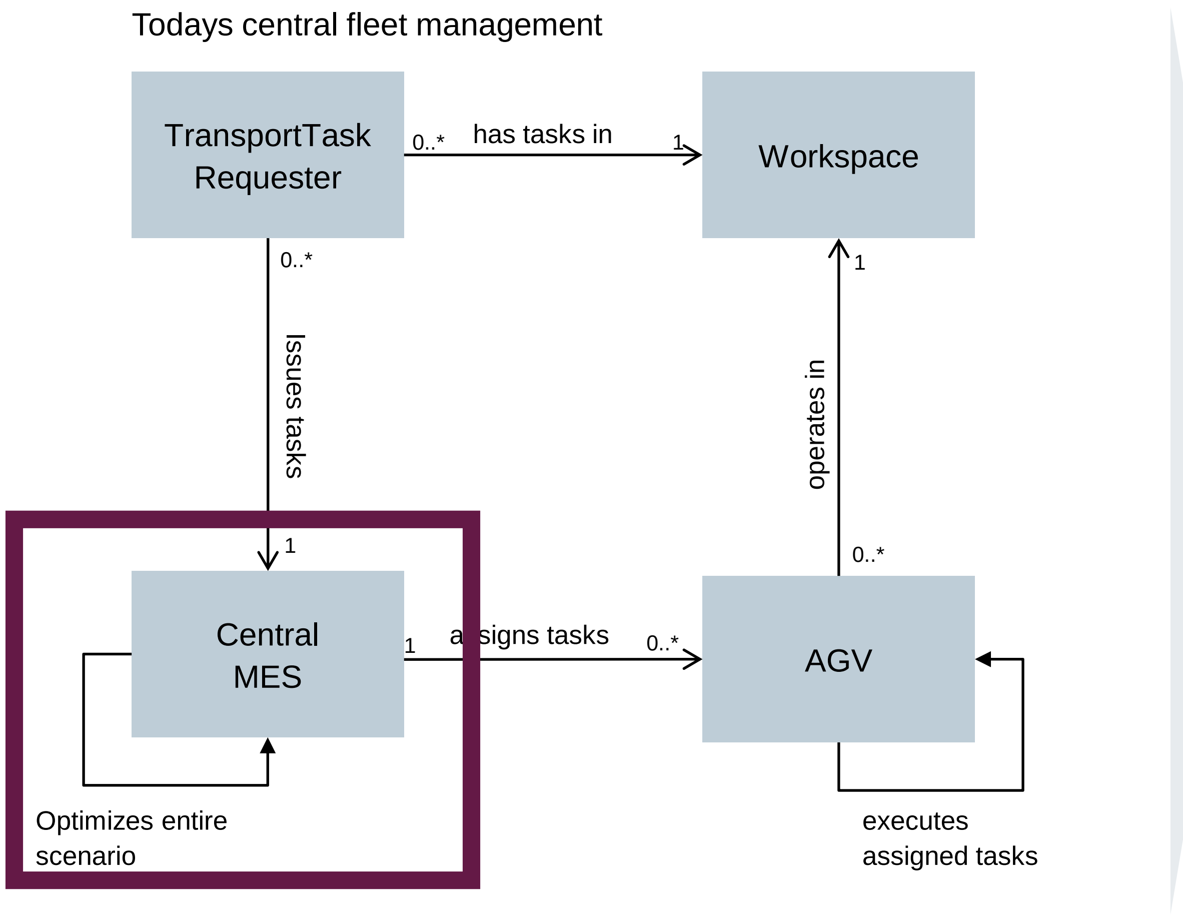


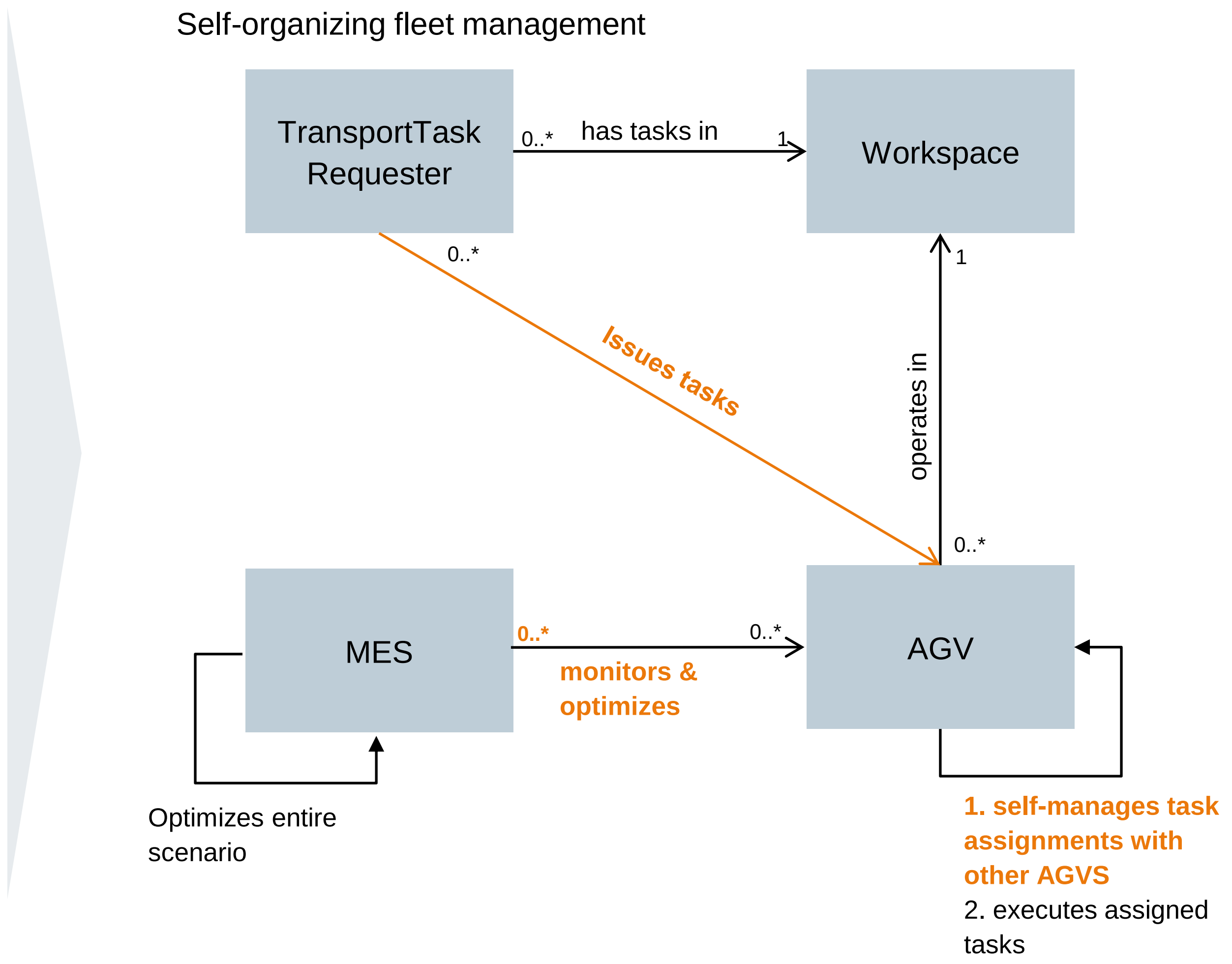
- Coaty agents can have any number of `IoSource` (Information Source) and `IoActor` (Information Consumer).
- `IoSource` and `IoActor` are part of a named shared, distributed `IoContext`.
- An IO Routing component manages information routing for one `IoContext` based on a rule engine; application-defined rules determine the association between `IoSource` and `IoActor` of the different agents.
- On `IoContext` state changes, the IO Routing component uses Coaty ASSOCIATE event pattern to update publication and subscription topics of `IoSources` and `IoActors` of the agents.

Note: Detailed implementation example available in Coaty 2.0 developer guides

SIEMENS

Resource allocations in resource networks
Example: AGV transport task assignments




Self-organizing fleet management
TransportTask Requester
0..*
has tasks in
1
Workspace
0..*
Issues tasks
1
operates in
0..*
MES
0..*
monitors & optimizes
0..*
AGV
Optimizes entire scenario
1. self-manages task assignments with other AGVS
2. executes assigned tasks

SIEMENS

Coaty – How you get started

Learn how to use Coaty JS

- Developer guide
- Best practice code examples and template
- Coding style guide
- Communication protocol specification
- Framework API documentation

Learn how to use CoatySwift

- Tutorial
- Developer guide
- Best practice code examples and template
- Framework API documentation
- Design rationale

Coaty

https://coaty.io

https://github.com/coatyio

Get in contact with us
coaty.team@gmail.com