Create your Digital Twin in days, not months.

IoT sensors with Eclipse Arrowhead framework

Jens Eliasson, ThingWave

jens.eliasson@thingwave.eu

ThingWave™

ARROWHEAD TOOLS · KYKLOS 4.0 · CHANGE2TWIN · DAIS · DRYADS

Supported by advanced EU H2020 projects

Eclipse Arrowhead framework

- Design goals
- System of systems architecture
- Reference implementation
- Implementation platform
- Eclipse open source

ARROWHEAD

Design goals

- Open standards, high interoperability
- Security from start
- Light weight
- Multi-protocol, multi-technology

ARROWHEAD

System of System architecture

- Service-Oriented Architecture (SOA)
  - Flexibility
  - Late binding
  - Loosely coupled
- Multi-protocol, multi-technology
  - *“one size never fits all”*
- Integration platform
  - To integrate OT and IT, with IoT

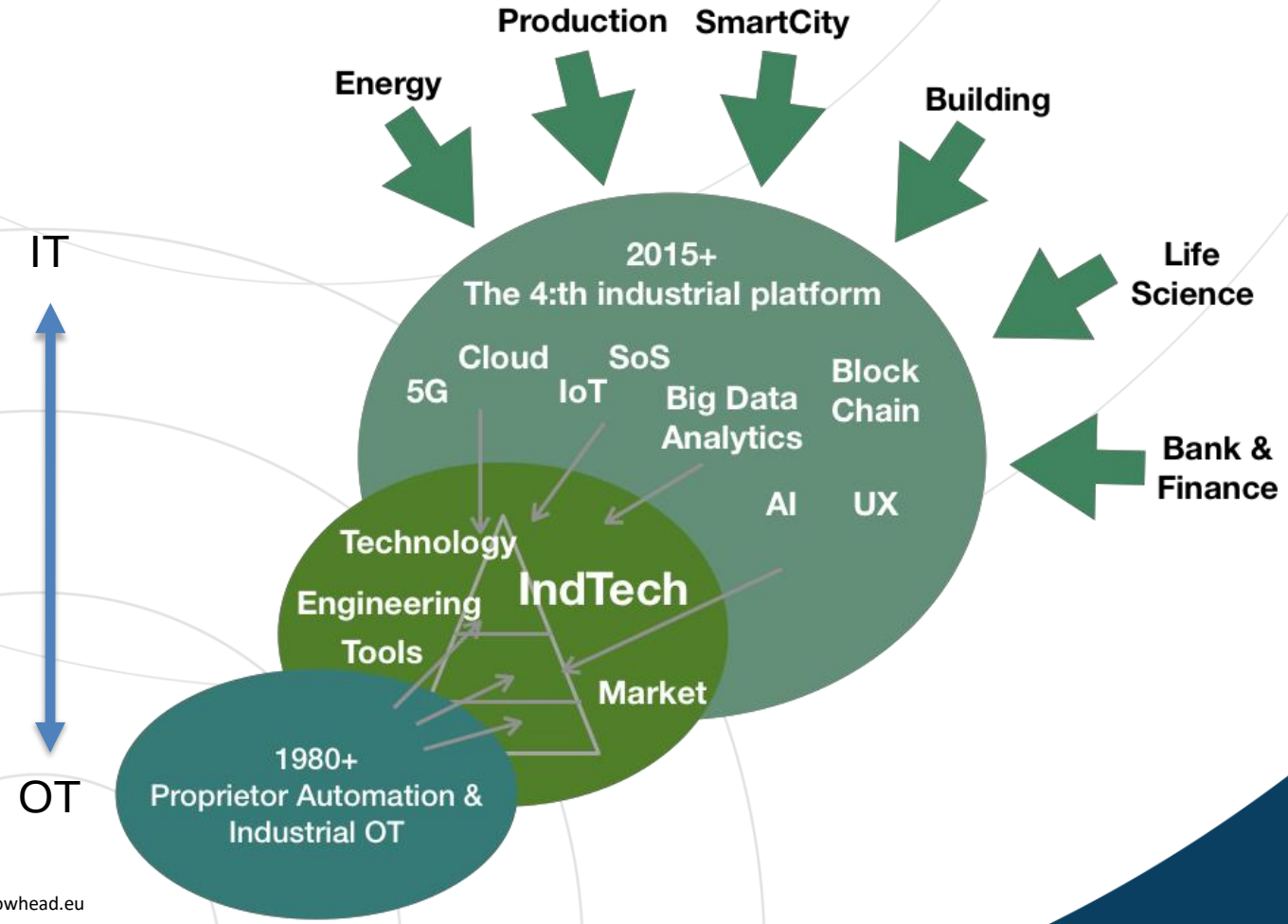
ARROWHEAD

Guiding requirements

Addressing industrial and complex production in multiple domains e.g.

- Manufacturing, e.g. automotive
- Continuous processing, e.g. steel, mining, paper & pulp
- Construction
- Energy - smart grid
- Electric vehicles
- Smart cities
- …

OT meets IT



ARROWHEAD

Capabilities

- Scalable
- Realtime
- Security
- Standards-based
- Engineering efficiency
  - Open technologies
- Integration platform
- ….

Technology basics

Service Oriented Architecture
Self contained local clouds

Service Oriented Architecture
Self contained local clouds

ARROWHEAD

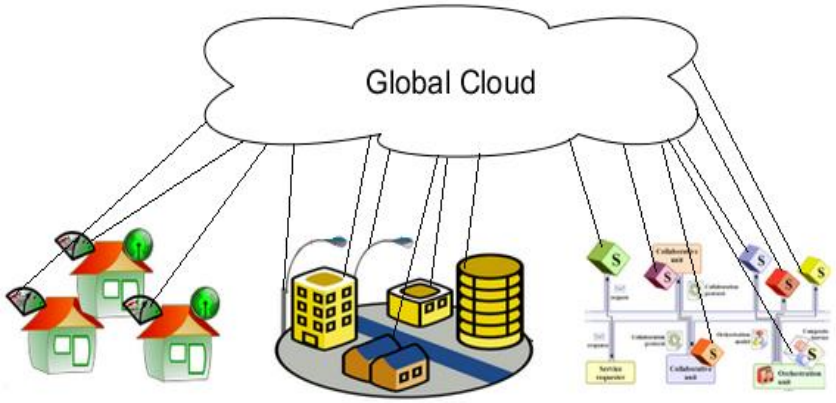
Arrowhead Objectives

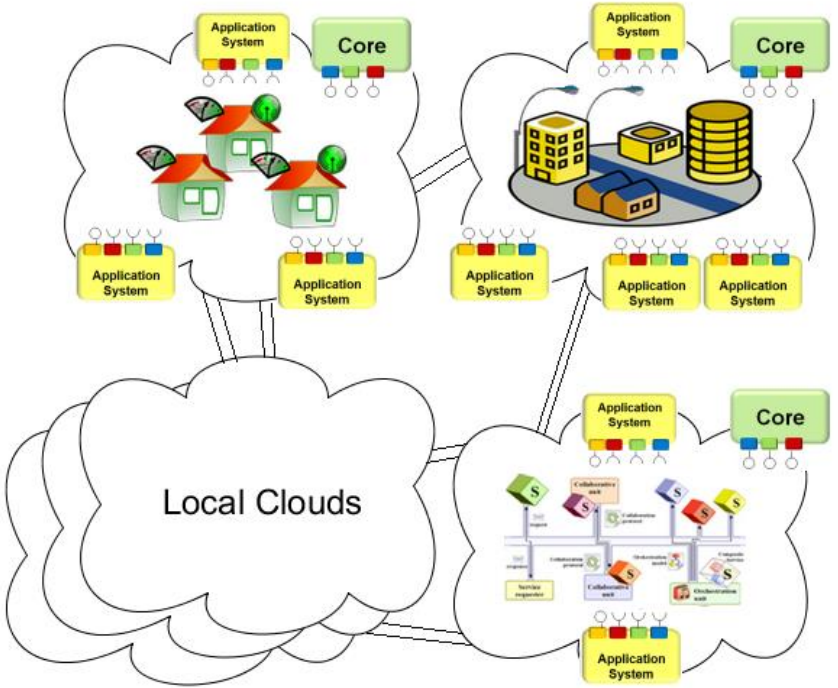
- **System of System Architecture for Industrial IoT and CPS**

- **Interoperability**: Service Oriented Architecture
  - Late Binding - Loose Coupling – Lookup (of Service Consumers+Providers)
- **Integrability**
  - easy interaction between Legacy and New (native Arrowhead) systems
- **Independence**
  - from underlying technologies (services)
  - from application protocols (translation)

ARROWHEAD

Global vs. Local Automation Clouds



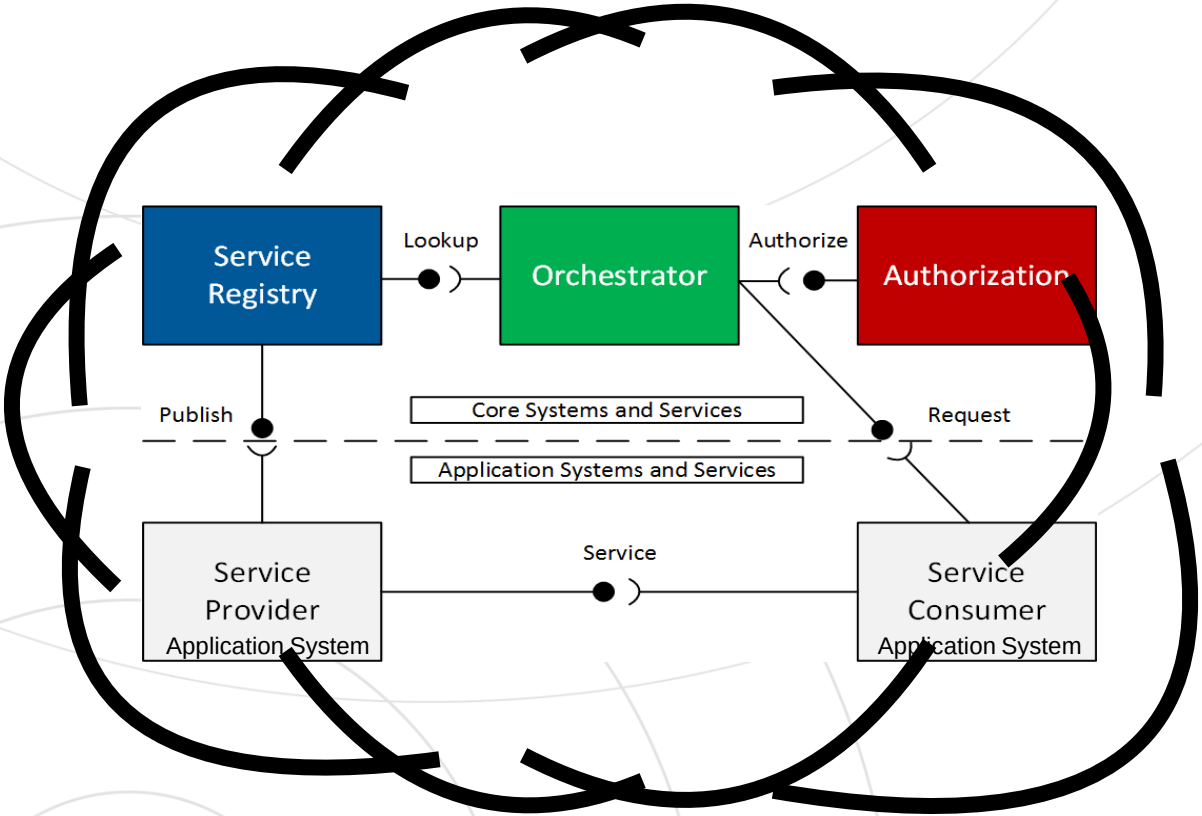


Advantages of Local Clouds:

- Real-time operation
- Security
- Engineering complexity reduction
- Inter-cloud service exchange enables (security)-controlled collaboration

ARROWHEAD

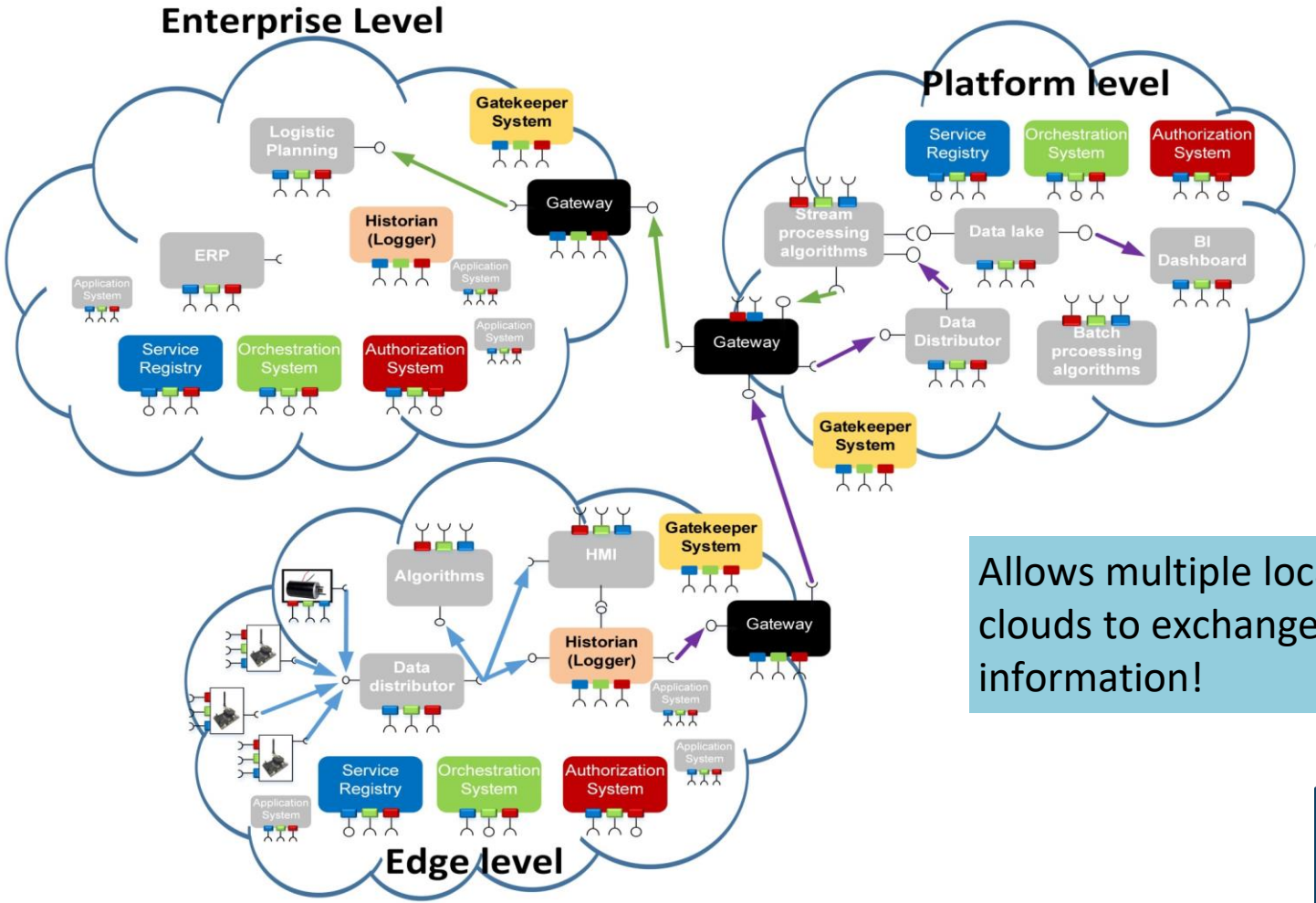
Self-contained local clouds



Mandatory Core systems

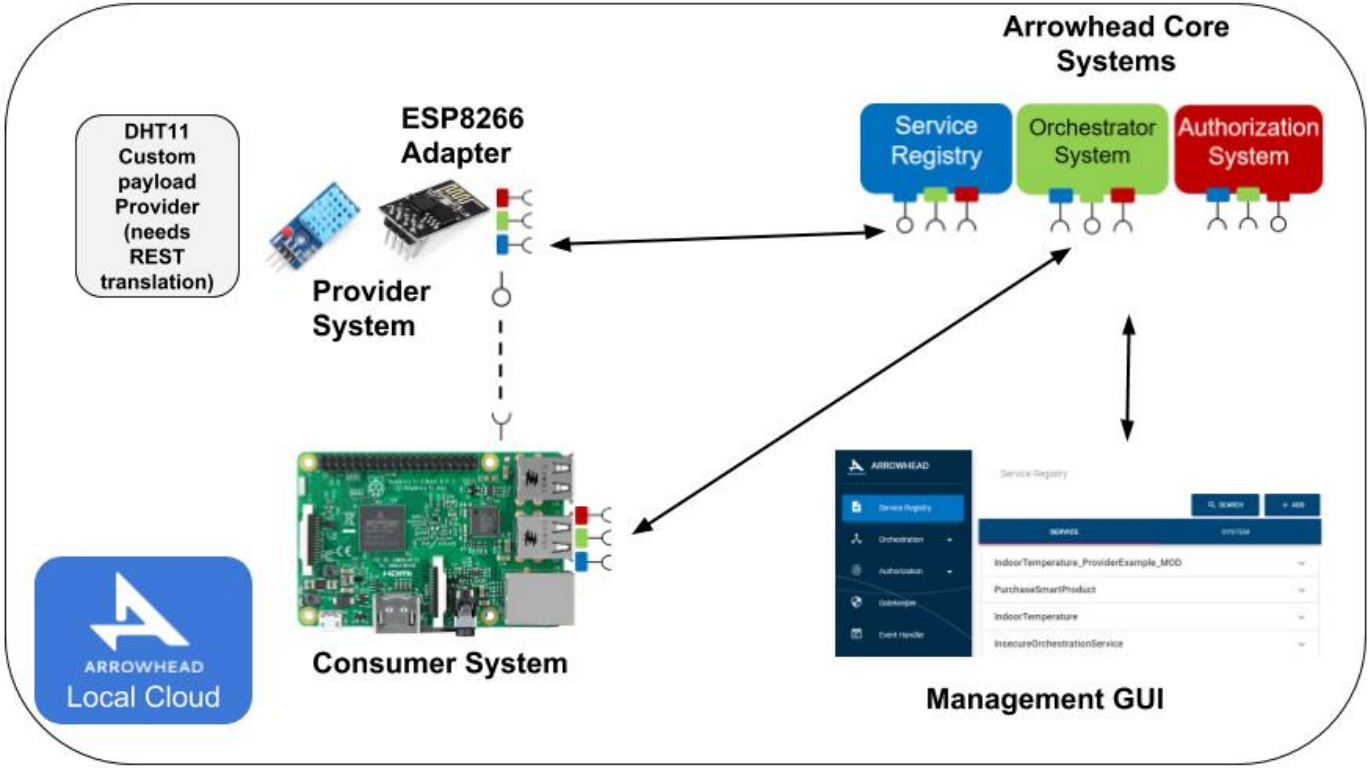
Application systems

Multi-level Service Exchange



Allows multiple local clouds to exchange information!

Simple application example



ARROWHEAD

Examples of supporting core system features

- **Translators and adaptors:**

  to allow protocol translation between various application systems (e.g. RESTful HTTP, CoAP, MQTT, WebSockets, OPC-UA, Modbus-TCP, Z-wave, etc.)

- **OnboardingController:**

  handle device onboarding to the local cloud with the support of SystemRegistry and DeviceRegistry

- **Choreographer:**

  to manage workflows and execute recipes in a dynamic manner: the next step is executed depending on service provider availability

- **PlantDescription:**

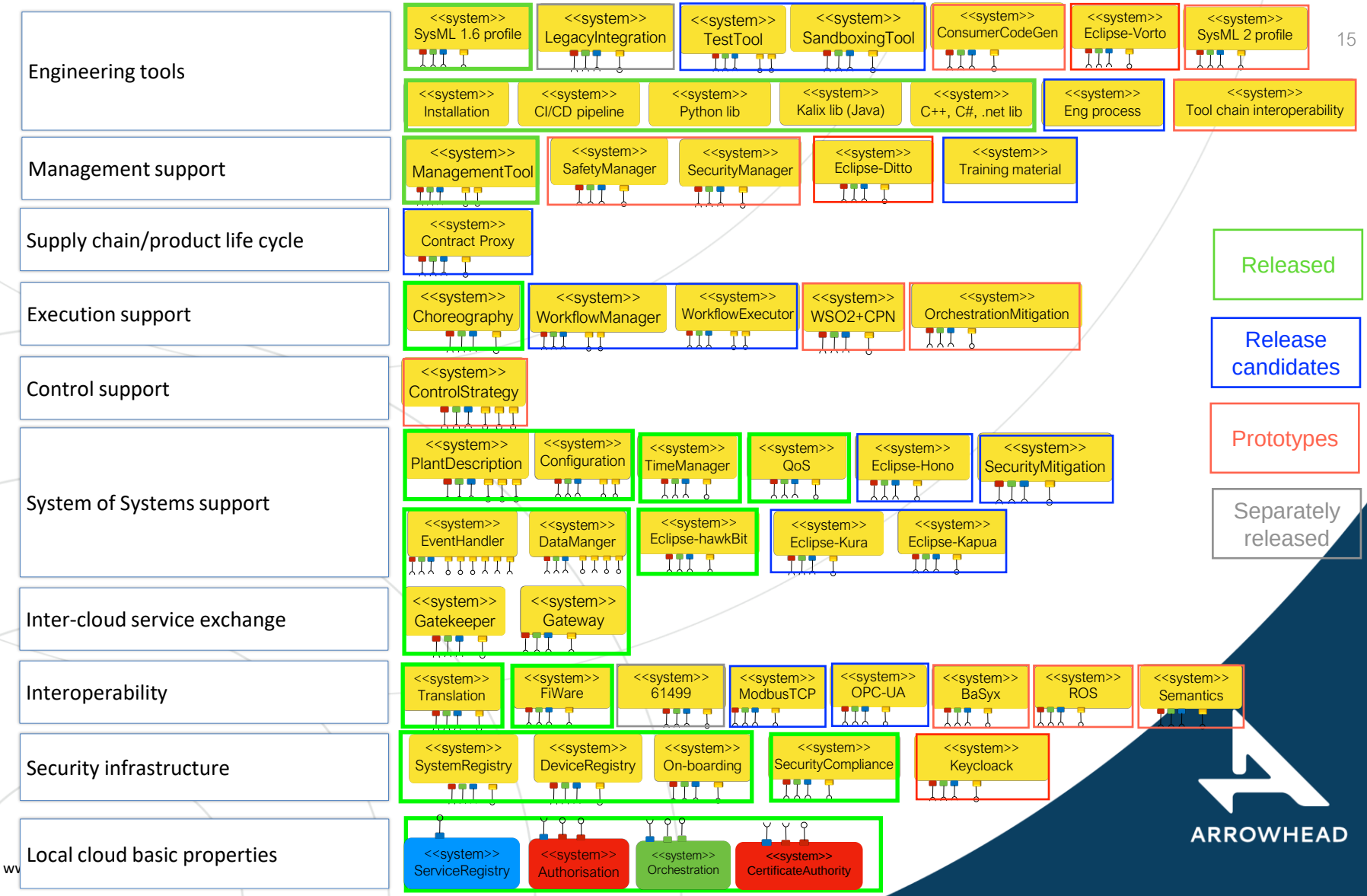
  to provide up-to-date information on the engineering plant

- **Event Handler, QoS Manager, CertificateAuthority, DataManager, TimeManager**

- **…**

ARROWHEAD

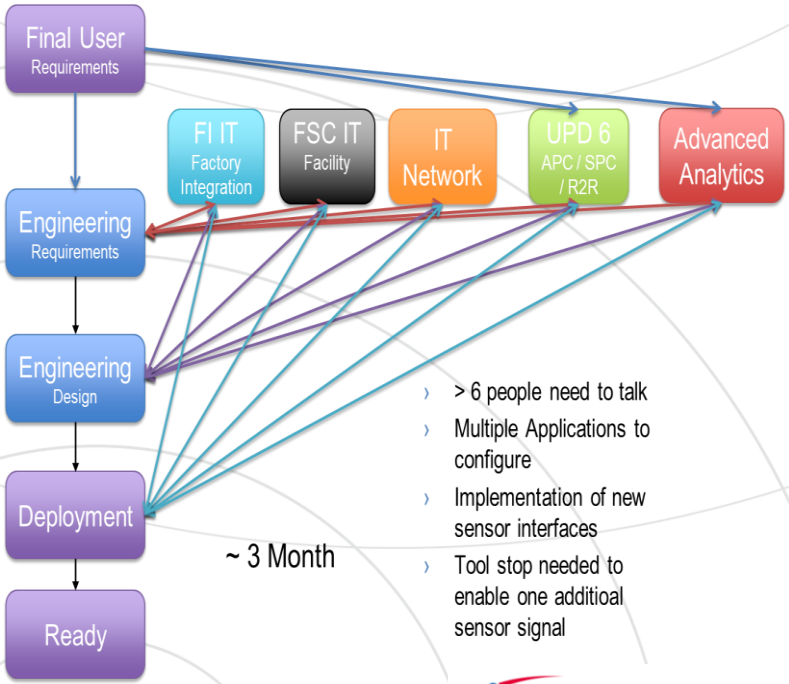
Arrowhead v4.4.0



ARROWHEAD

Engineering efficiency

Before Arrowhead

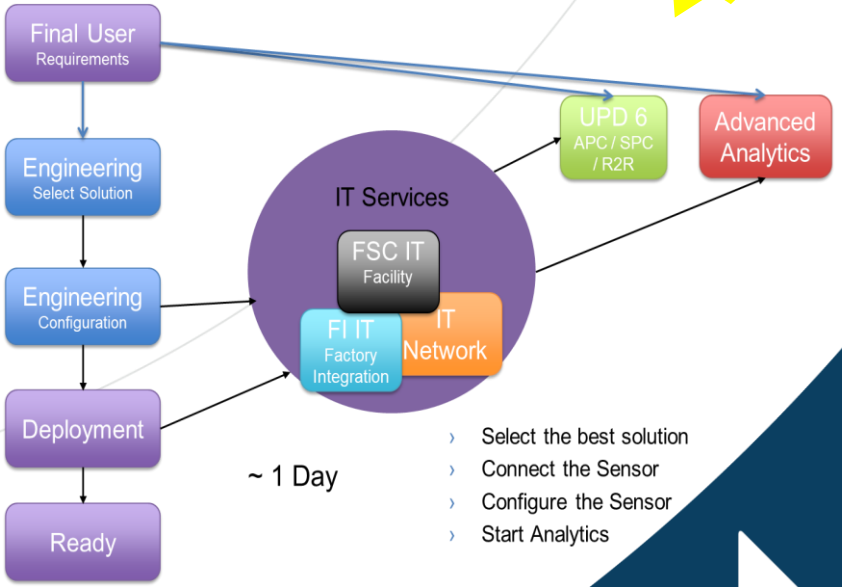


~ 3 Month

- > 6 people need to talk
- Multiple Applications to configure
- Implementation of new sensor interfaces
- Tool stop needed to enable one additioal sensor signal

With Eclipse Arrowhead

Highlights Y3



~ 1 Day

- Select the best solution
- Connect the Sensor
- Configure the Sensor
- Start Analytics

A comparison of IoT-SoS Architectures & Platforms

Features	Arrowhead	AUTOSAR	BaSyx	FIWARE	IoTivity	LWM2M	OCF
Key principles	SOA, Local Automation Clouds	Runtime, Electronic Control Unit (ECU)	Variability of production processes	Context awareness	Device-to-device communication	M2M, Constrained networks	Resource Oriented REST, Certification
Real-time	Yes	Yes	No	No	Yes (IoTivityConstrained)	No	No
Run-time	Dynamic orchestration and authorization, monitoring, and dynamic automation	Runtime Environment layer (RTE)	Runtime environment	Monitoring, dynamic service selection and verification	No	No	No
Distribution	Distributed	Centralize	Centralize	Centralize	Centralize	Centralize	Centralize
Open Source	Yes	No	Yes	Yes	Yes	Yes	No
Resource accessibility	High	Low	Very low	High	Medium	Medium	Low
Supporters	Arrowhead	AUTOSAR	Basys 4.0	FIWARE Foundation	Open Connectivity Foundation	OMA SpecWorks	Open Connectivity Foundation
Message patterns	Req/Repl, Pub/sub	Req/Repl, Pub/sub	Req/Repl,	Req/Repl, Pub/sub	Req/Repl, Pub/sub	Req/Repl	Req/Repl
Transport protocols	TCP, UDP, DTLS/TLS	TCP, UDP, TLS	TCP	TCP, UDP, DTLS/TLS	TCP, UDP, DTLS/TLS	TCP, UDP, DTLS/TLS, SMS	TCP, UDP, DTLS/TLS, BLE
Communication protocols	HTTP, CoAP, MQTT, OPC-UA	HTTP	HTTP, OPC-UA	HTTP, RTPS	HTTP, CoAP	CoAP	HTTP, CoAP
3rd party and Legacy systems adaptability	Yes	Yes	Yes	Yes	No	No	No
Security Manager	Authentication, Authorization and Accounting Core System	Crypto Service Manager, Secure Onboard Communication	--	Identity Manager Enabler	Secure Resource Manager	OSCORE	Secure Resource Manager
Standardization	Use of existing standards	AUTOSAR standards	Use of existing standards	FIWARE NGSI	OCF standards	Use of existing standards	OCF standards

C. Paniagua and J. Delsing, "Industrial Frameworks for Internet of Things: A Survey," in *IEEE Systems Journal*, doi: 10.1109/JSYST.2020.2993323.

ARROWHEAD

Real-world case study

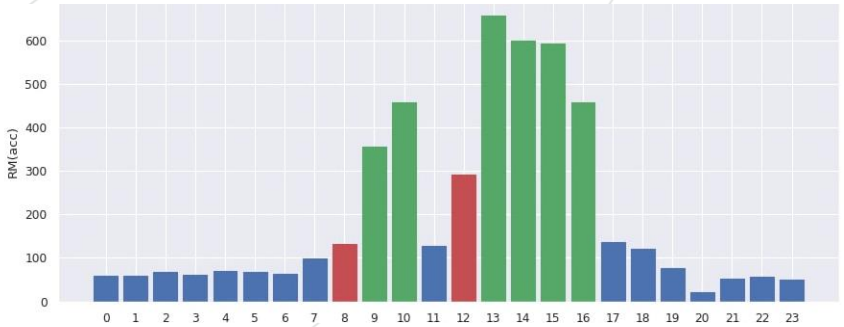
- Advanced IoT-based condition monitoring
  - Vibration, temperature, etc
- Smart device, remote configured and controlled by Arrowhead systems
- Streaming of sensor data over MQTT
- Arrowhead Local Cloud on Gateway
- Arrowhead-compliant control software

Wi-Fi IoT sensor

Targeted application

- Mining

- Overall equipment effectiveness (OEE)

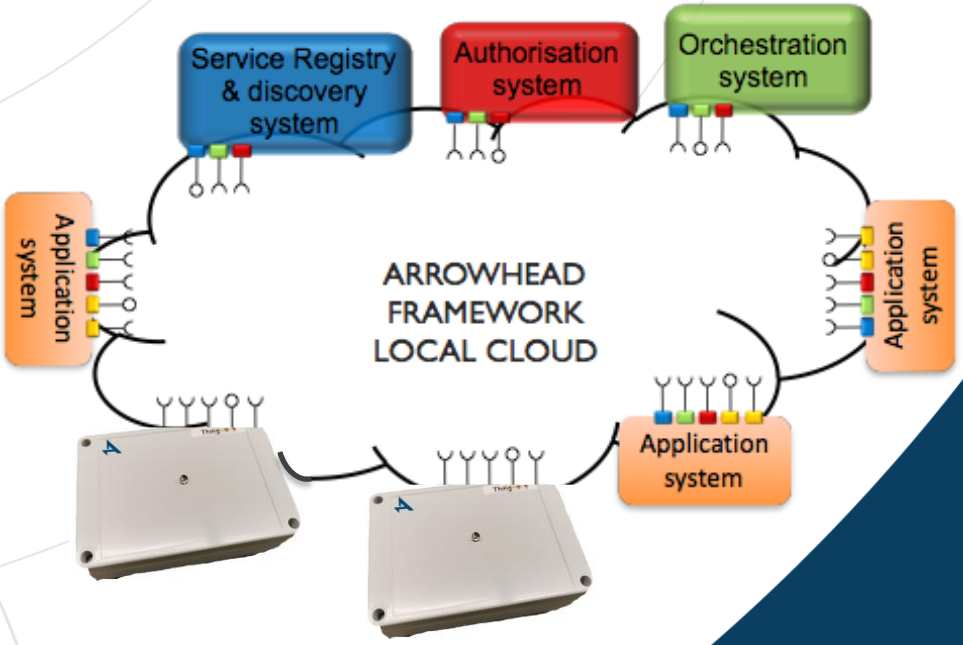


- Manufacturing

- Predictive maintenance

Real-world case study

- The Arrowhead approach enables easy customizations with SoS
- Software “plugins” can be implemented as services and deployed at any time
- Customers can integrate their own processing software



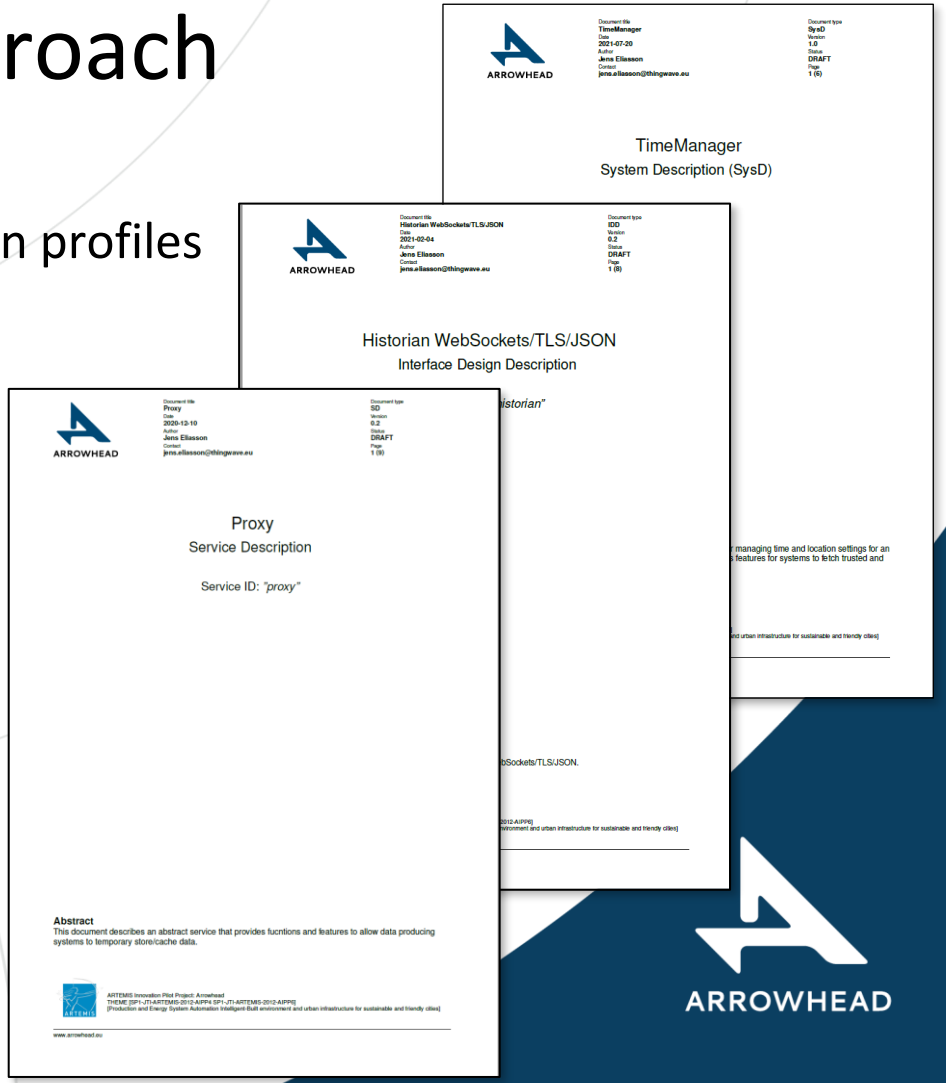
Real-world case study

1. ThingWave Nucleus™ registers with the Service Registry (SR)
2. IoT device registers its services with the Service registry
3. IoT device queries the SR for the *Orchestration service* (OS)
4. The OS returns to address and port of TW Nucleus
5. IoT device knows where to send data
6. This procedure is repeated for all services for:
    1. configuration
    2. time
    3. data storage,
    4. etc.

Documentation approach

1. Data models and communication profiles
2. Services
3. Systems
4. Systems of Systems

- Simplifies development and documentation!



Brief summary - Arrowhead framework

- System of Systems
  - Interoperability, Integrability, Independence
  - Service Oriented Architecture
- Local Automation Clouds
- Various Multi-clouds: Edge, Platform, Enterprise
- Maturity Levels for Integrability – hardware and software adaptors
- Mandatory and Supporting Core Systems
- Collaborations with other Eclipse projects

ARROWHEAD

Resources

Web

www.arrowhead.eu/arrowheadframework

Github

www.github.com/arrowhead-f

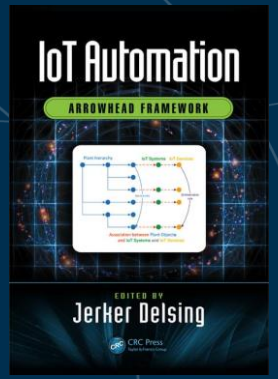
Support (Slack)

arrowhead-dev.slack.com

Youtube

www.youtube.com/channel/UCC-kTqFXh7StNwR7IFCRCjw

Book



ARROWHEAD