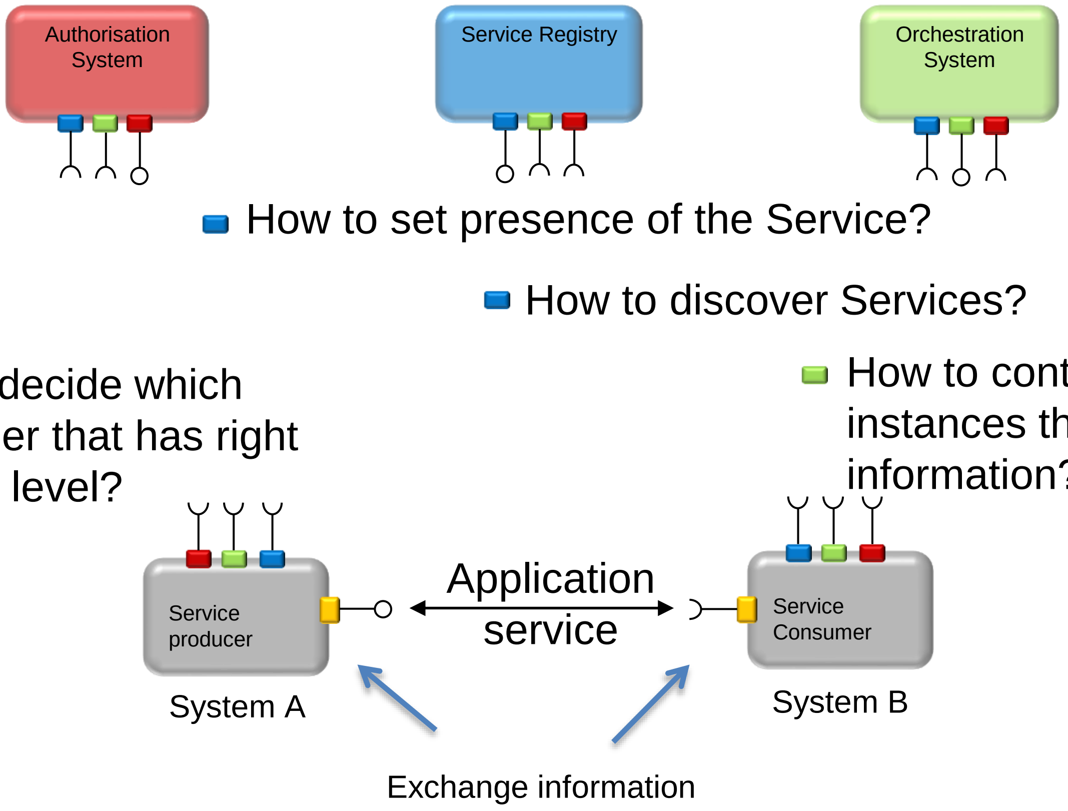
Customer requirements, pains and challenges
- The Eclipse Arrowhead Framework

Fredrik Blomstedt, BnearIT AB in cooporation with Sinetiq AB

Architecture vs Solution

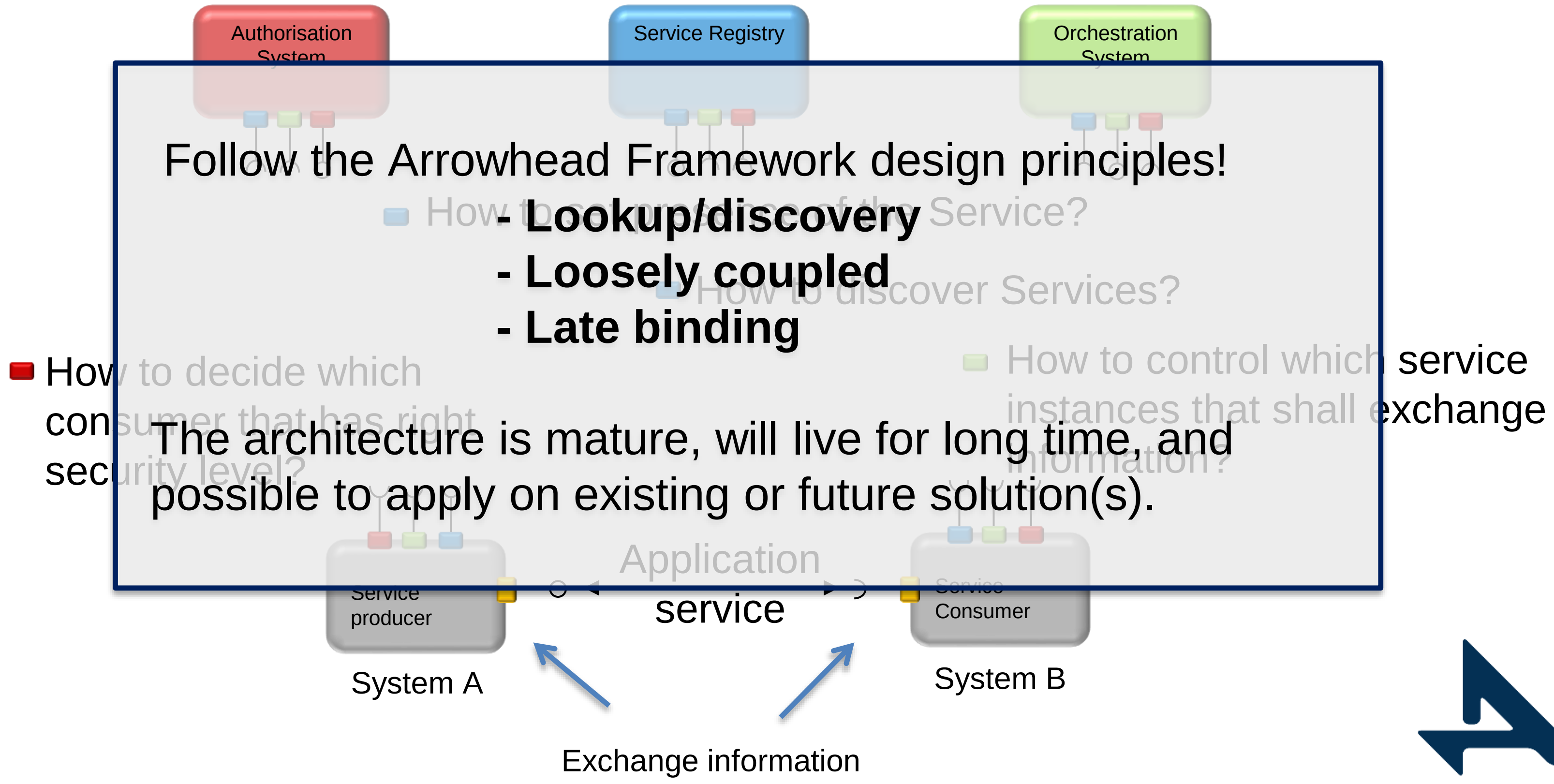
The Architecture – the goal!



- 🟦 How to set presence of the Service?
- 🟦 How to discover Services?
- 🟥 How to decide which consumer that has right security level?
- 🟩 How to control which service instances that shall exchange information?

ARROWHEAD TOOLS

The Architecture – mature and strong!


Authorisation System
Service Registry
Orchestration System
How to ... presence of the Service?
How to discover Services?
How to decide which consumer that has right security level?
How to control which service instances that shall exchange information?
Application service
Service producer
Service Consumer
System A
System B
Exchange information

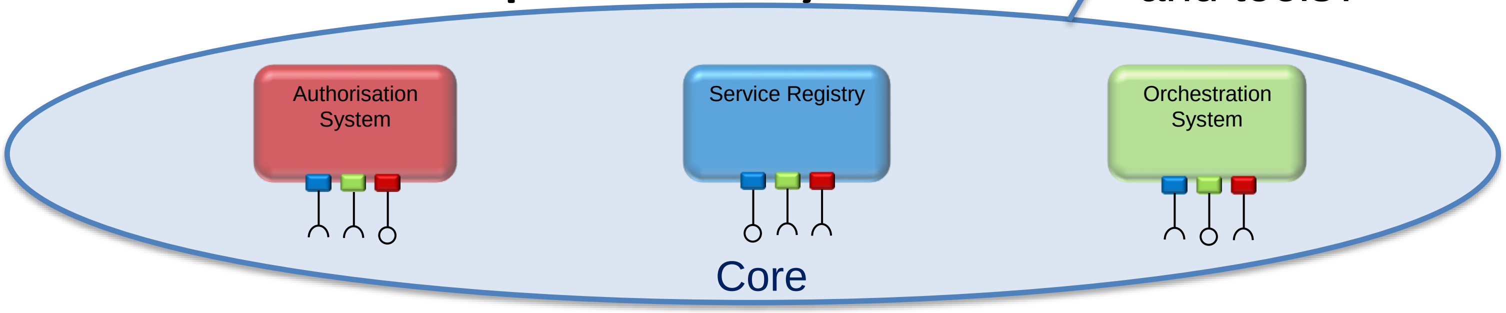
Follow the Arrowhead Framework design principles!

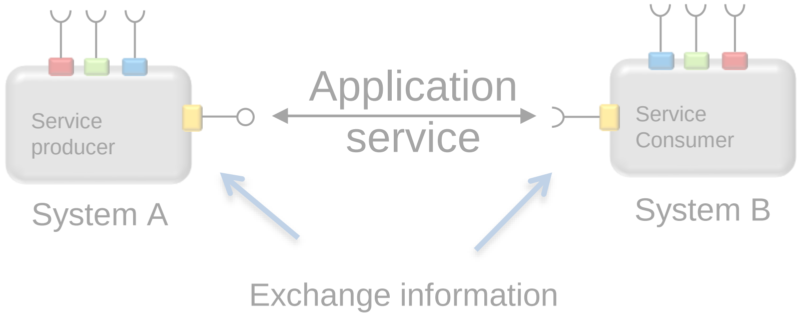
- **Lookup/discovery**
- **Loosely coupled**
- **Late binding**

The architecture is mature, will live for long time, and possible to apply on existing or future solution(s).

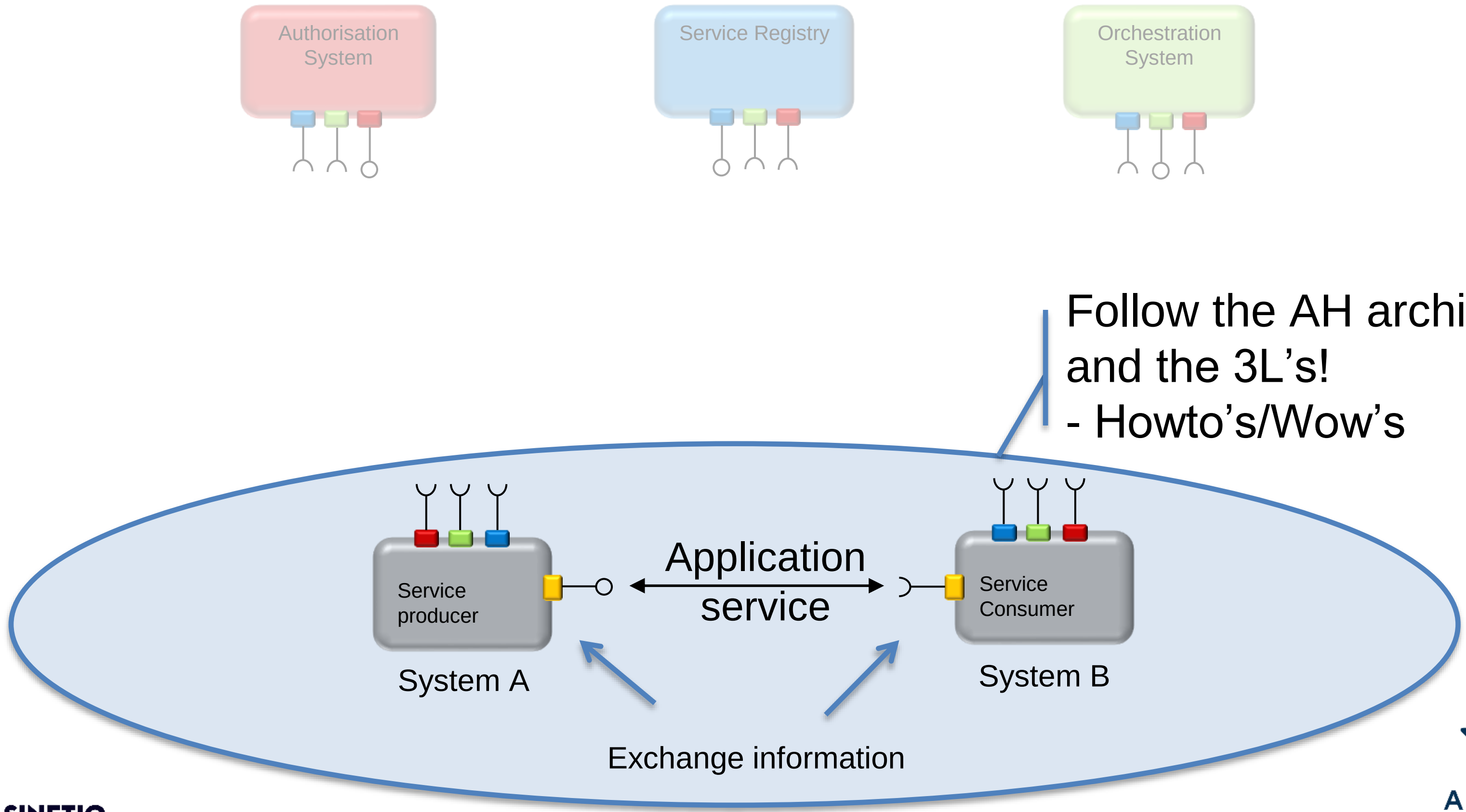
Enable interoperability

How can we use customer applications, middlewares and tools?



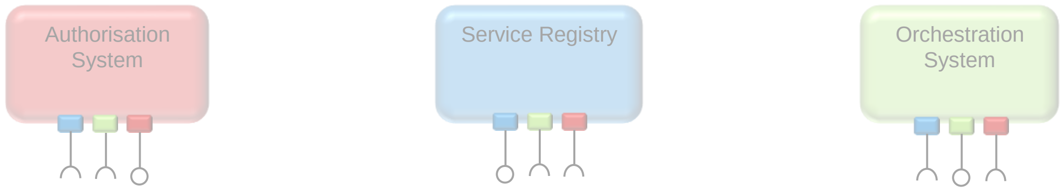


Application – Business value (information as an asset!)



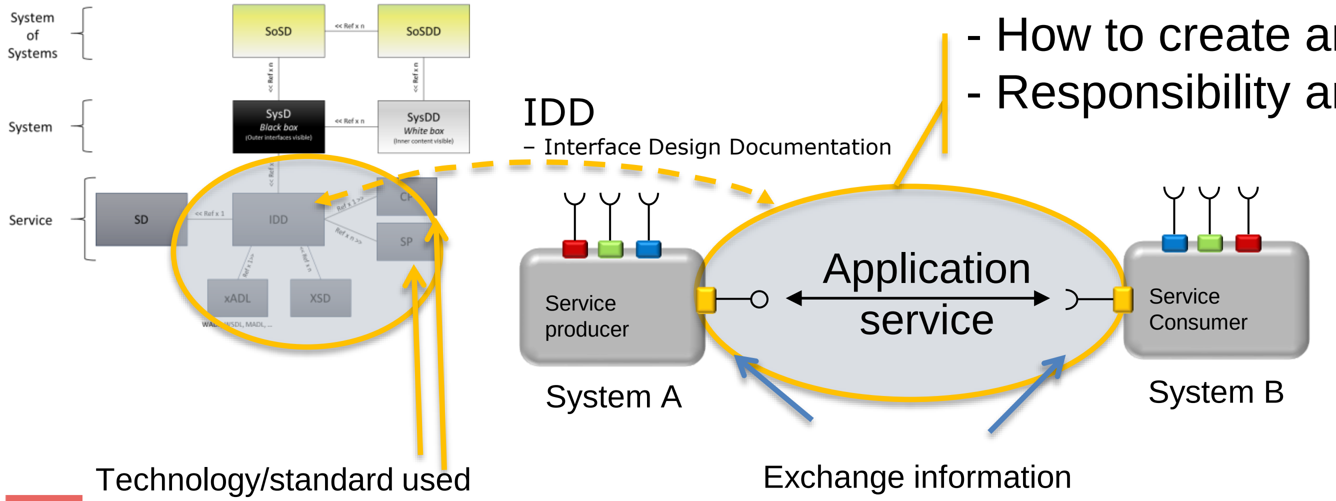
Follow the AH architecture and the 3L's!
- Howto's/Wow's

Govern integrations and achieve interoperability

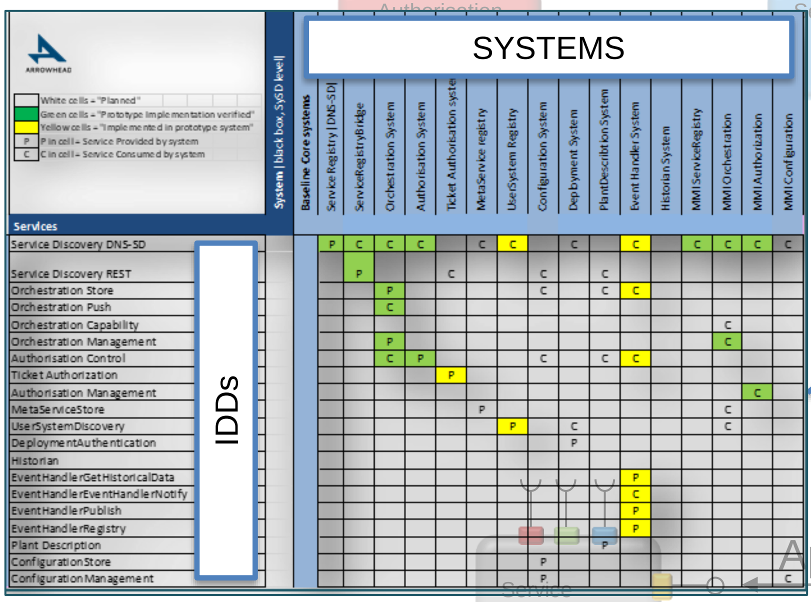


- How to create and govern(LCM) IDD's?
- Responsibility and ownership?

IDD
– Interface Design Documentation



Govern integrations and achieve interoperability



ARROWHEAD

⬜	White cells = "Planned"
🟩	Green cells = "Prototype implementation verified"
🟨	Yellow cells = "Implemented in prototype system"
P	P in cell = Service Provided by system
C	C in cell = Service Consumed by system

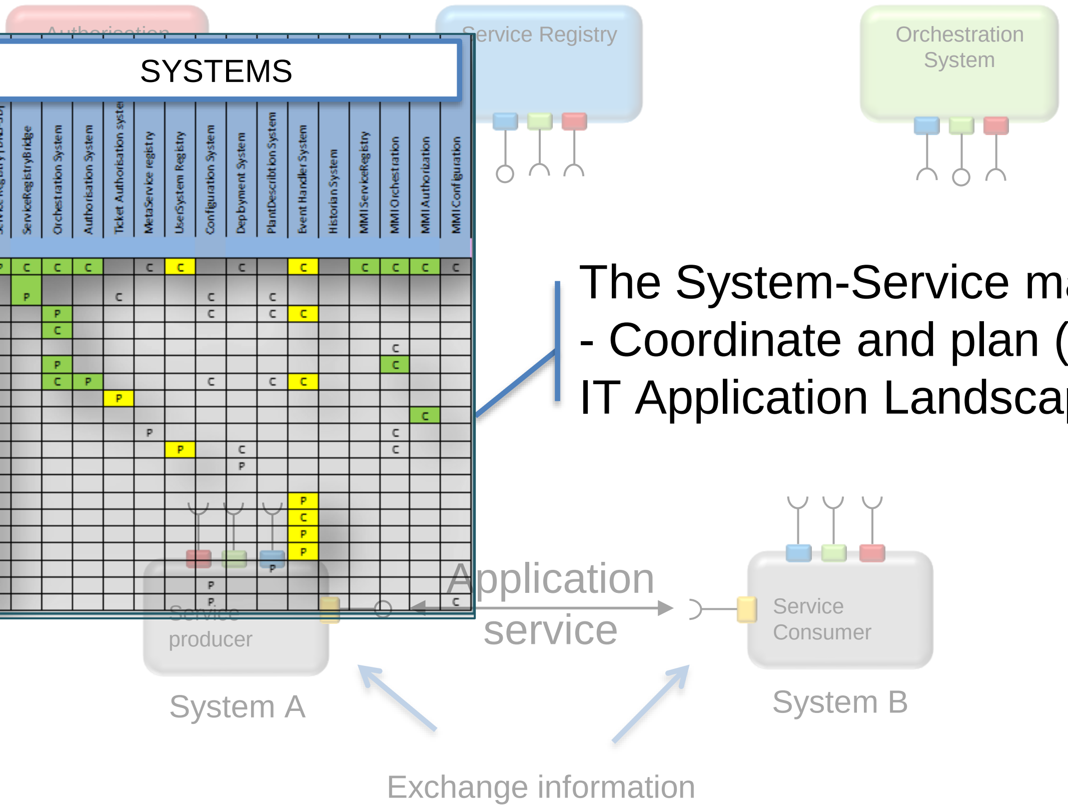
SYSTEMS

System \| black box, SySD level]

Baseline Core systems

IDDs

Services	Service Registry | DNS-SD\]	ServiceRegistryBridge	Orchestration System	Authorisation System	Ticket Authorisation syste	MetaService registry	UserSystem Registry	Configuration System	Deployment System	PlantDescribtion System	Event Handler System	Historian System	MMI ServiceRegistry	MMI Orchestration	MMI Authorization	MMI Configuration
Service Discovery DNS-SD	P	C	C	C		C	C		C		C		C	C	C	C
Service Discovery REST		P			C			C		C						
Orchestration Store			P					C		C	C					
Orchestration Push			C													
Orchestration Capability														C		
Orchestration Management			P											C		
Authorisation Control			C	P				C		C	C					
Ticket Authorization					P											
Authorisation Management															C	
MetaServiceStore						P								C		
UserSystemDiscovery							P		C					C		
DeploymentAuthentication									P							
Historian																
EventHandlerGetHistoricalData											P					
EventHandlerEventHandlerNotify											C					
EventHandlerPublish											P					
EventHandlerRegistry											P					
Plant Description										P						
ConfigurationStore								P								
ConfigurationManagement								P								C



The System-Service matrix
- Coordinate and plan (system of system)
IT Application Landscape capability

SINETIQ

ARROWHEAD TOOLS

Benefits, challenges, learning and findings

Business goals

- **Faster** and more **flexible** support of **new needs**
- **Better** and **clearer requirements** that can be used for example new **acquisition**
- **Re-use of investments** (systems, hardware and software)

__Small__ and __clear components__ have __fewer__ requirements, __easier__ to describe, __easier__ to verify and can __easily__ be deployed, in a __seamless__ way, __possible__ direct into production. A small component require __less__ economic and personnel __resources__. __Small__ changes and improvements can fit within small and __limited budgets__ and used for the __current__ and __actual need__ at the __time__.

Technical goals

- **Easier** to **maintain**
- **Well defined** boundaries which lead to better/higher **dynamics**, **flexibility** and **modularity**
- Faster (**cheaper**) development
- System **support** that is adopted to **current business processes**
- **Reduced personnel**
- **Reduced supplier**-dependent, **technology**-dependent and **product**-dependent

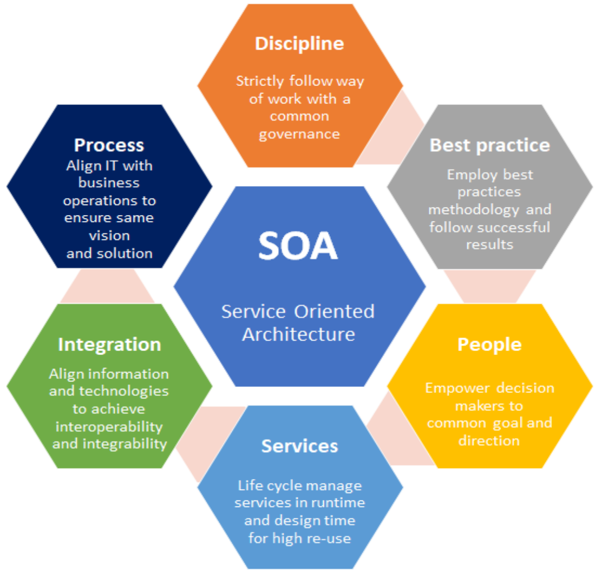
***System design**, based on serviceoriented architecture, **handled properly**, brings above stated **benefits/features**.*

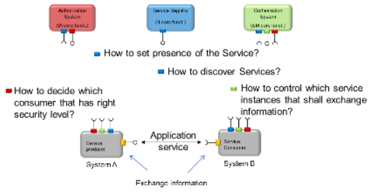
Main objectives

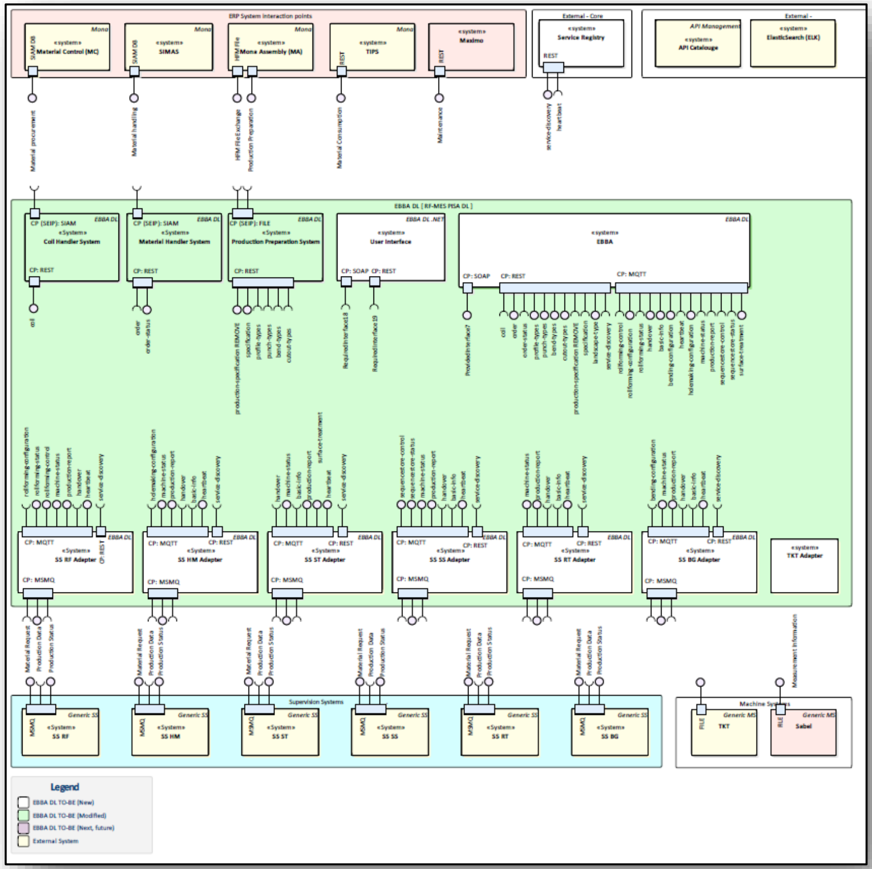
- **Increase compatibility**
  Exchange information with **minimal integration needs.**
- **Increase coordination**
  Coordinate resources and applications. Ultimate leads support of increased coordination (Federation) into a naturally coordinated environment.
- **Increase collaboration**
  Increase interaction between business and technology. The technology can be easily and flexibly adapted to new, changing business requirements.
- **Reduce dependence**
  Increased flexibility in choice of supplier / manufacturer / technology ("Provider independence"). Increased ability to choose ("best-of-breed") business and technology solutions.

The solution and architecture meet above stated main objectives.

Architecture


Discipline
Strictly follow way of work with a common governance
Process
Align IT with business operations to ensure same vision and solution
Best practice
Employ best practices methodology and follow successful results
SOA
Service Oriented Architecture
Integration
Align information and technologies to achieve interoperability and integrability
People
Empower decision makers to common goal and direction
Services
Life cycle manage services in runtime and design time for high re-use


How to set presence of the Service?
How to discover Services?
How to decide which consumer that has right security level?
How to control which service instances that shall exchange information?
Application service
System A
System B
Exchange information



Key Benefits using serviceoriented concept and mindset

- **Increased re-use**
  Increase service life for existing invested system solutions. Lifecycles are governed by the service life.
- **Increased adaptability**
  Increase the ability to efficiently adapt technology after organizational changes
- **Reduced IT load**
  Reduce overall load and limited / limiting system solutions. Increase the ability of strategic goals with fast and flexible adoption of the IT-landscape.

*Service oriented based system solutions have **decoupled life cycles** for each of the **components**. It is **easier** to **govern** and **maintain** a **modern, long-term** and **effective integration**. This leads to **reduced resistance** to developing a system solution. System architecture **reduces** system solution **maintenance costs** and **maintenance needs** can be focused on the **necessary features**. The **flexibility** of service based oriented system solutions **allows adjustments** to **new situations** to be done **without** the **need** for **changing** the **existing** building blocks (**components**).*

*With interoperable system solutions, participants **can choose which rate to adapt their system to changing needs.***

*Because the architecture is naturally federative, you can **decide how much resources and at what rate** you should be interoperable.*

*In addition, participants **can choose which parts** of a federal system solution that **fits the individual needs**.*

ARROWHEAD TOOLS

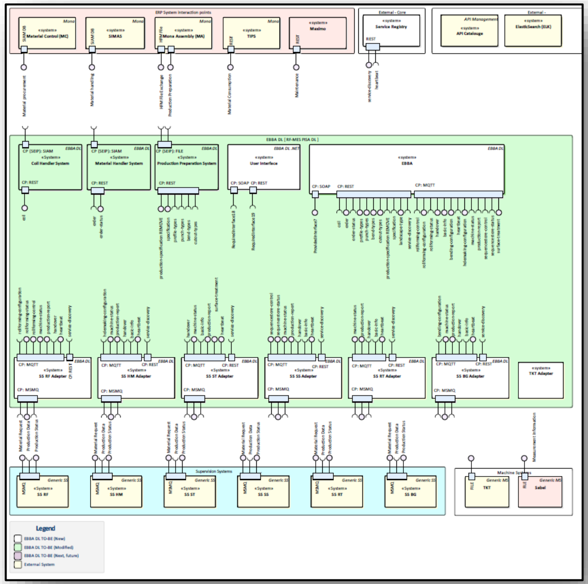
SINETIQ

Customer solution and some figures!

- 7 MES-Level systems (one running in 6 instances)
- 26 Services (+60 producers/consumer instances)
- 5 Technologies (MSMQ, MQTT, REST, SQL, FILE)
- Integration ERP 5 systems
- Integration SCADA 4 systems
- Service Registry, Orchestration, Authorization

Two first operational weeks

- 3 new production days records (600 -> 780)



Findings, lessons learned

Distributed (decentralized) and **centralized** successfully used together!

- Functions small, fast and easy to manage/govern (buy, requirements, develop, test, verify, deploy and maintain)
- IT-operations centralized

Future proof (re-use investments, evolve controlled)

- **Legacy** (OT) with **new**/future (IT) **systems**

Governance (top-down AND bottom-up)

- **Discipline**, coordinate services/api:s/technologies between groups for re-use and global success
- **Definitions** (Service, API, I/F, MicroService, System, Application)
- Service Based Architecture (Loosely coupled, Lookup, Late binding)
- Service registry, API-catalogue, API-gateway

  NOTE: API's for **ALL** technologies, not only REST/HTTP…

SINETIQ

ARROWHEAD TOOLS

Findings, lessons learned

Possible **traps** to handle

- Knowledge and common view in the organization
  - Align all, practical examples at all levels, **ownership**, costs, way of work and more..
- Vendor **lock-in**
- Product **dependent/lock-in**
- Technology **dependent**
- Distributed **mud** (on all levels in the organization)
- **Legacy** becomes the **driver** of the future **solution**
- Solution control(**limit**) our future possibilities
- Dependence of **critical personnel**/resources
- End to end **dependencies**

Aspects and targets

- Verification and validation
- Deploy
- Monitoring
- Logging
- Ownership
- Operation, service and support
- Way of work (governance)
    - Coordination

Customer example: Time-to-delivery

- A System, measurement quality control (deviation for holes)
- Re-use three services (Service registry, one producer and one consumer)
- One employee, new in the serviceoriented area, solved it in **~80 hours** (two weeks period). Estimated around 240+ hours.
- Easy to test and deploy
- Deploy at runtime in operation (seamless in running production!)
- Future proof

Goal fulfillment

- Increased collaboration
- Increased information exchange
- Decreased dependencies
  - Enable options to choose solution, products and technologies
- Easier to maintain
  - Reduced need of personnel
- Faster, easier, cheaper and enables flexible support for new decisions and business needs
- Clear and better requirements, capabilities and functions
- Enable to fulfill return of investments
- Project cost ended up at approx. 13msek for original scope
  - Created possibility to make a couple of the CR´s that were really good but not ”must have”

Thank you!

www.arrowhead.eu