



The eIMS framework

Open Workshop, Brussel 16.11.2016

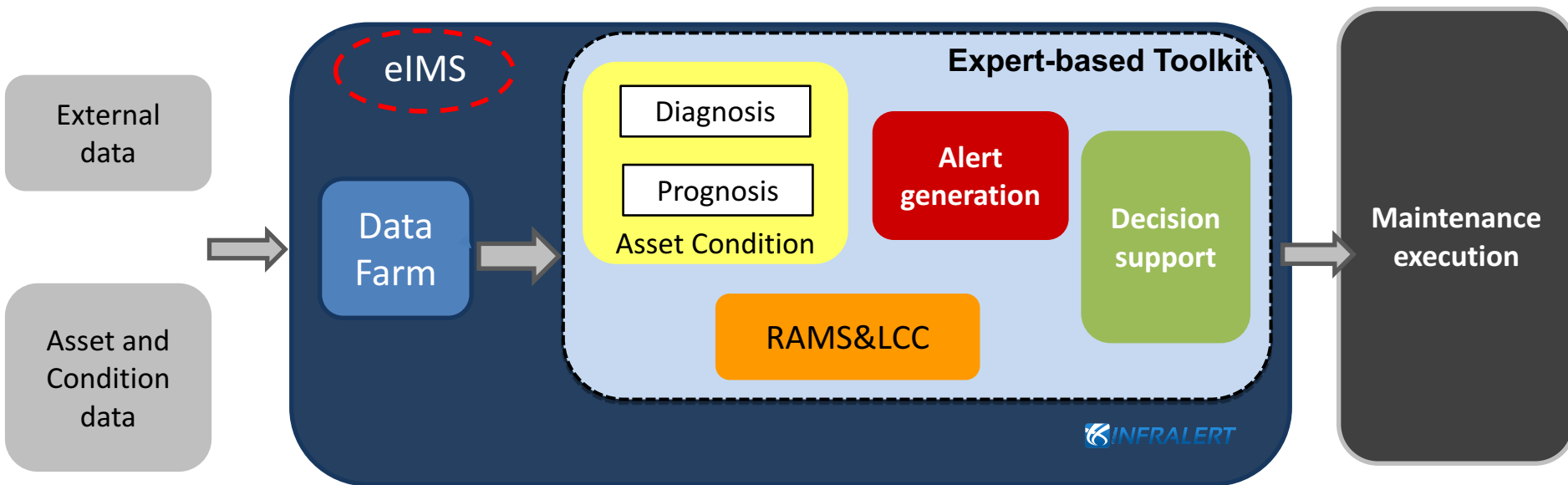
András Juszt



- General overview
- eIMS Architecture
  - eIMS Framework
  - Expert based toolkits
  - Integration gateway
  - Presentation layer
- Functional design of eIMS
  - Use Cases and Actors
  - Software design
  - Middleware classes
- System Architecture
- IT infrastructure
- UI wireframes



General Overview

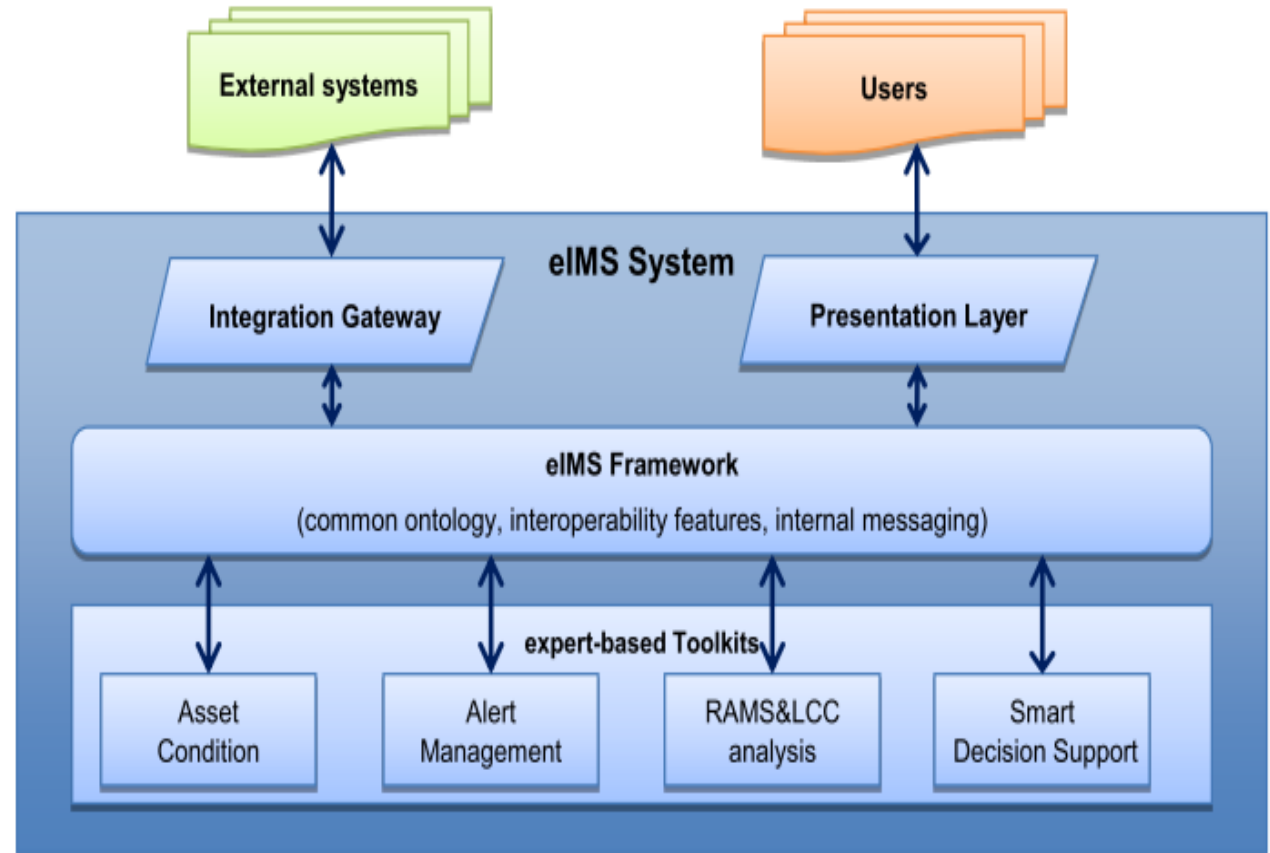


Objectives and Status

- Define eIMS architecture and functionality – **submitted**
  - Software architecture
  - Functional design
  - IT infrastructure
  - Security aspects
- Implement eIMS Framework and Presentation Layer – **ongoing**
  - User interface prototype
  - Middleware for eIMS ontology
- Implement and integrate Expert-based Toolkit features
- Develop the Integration Gateway

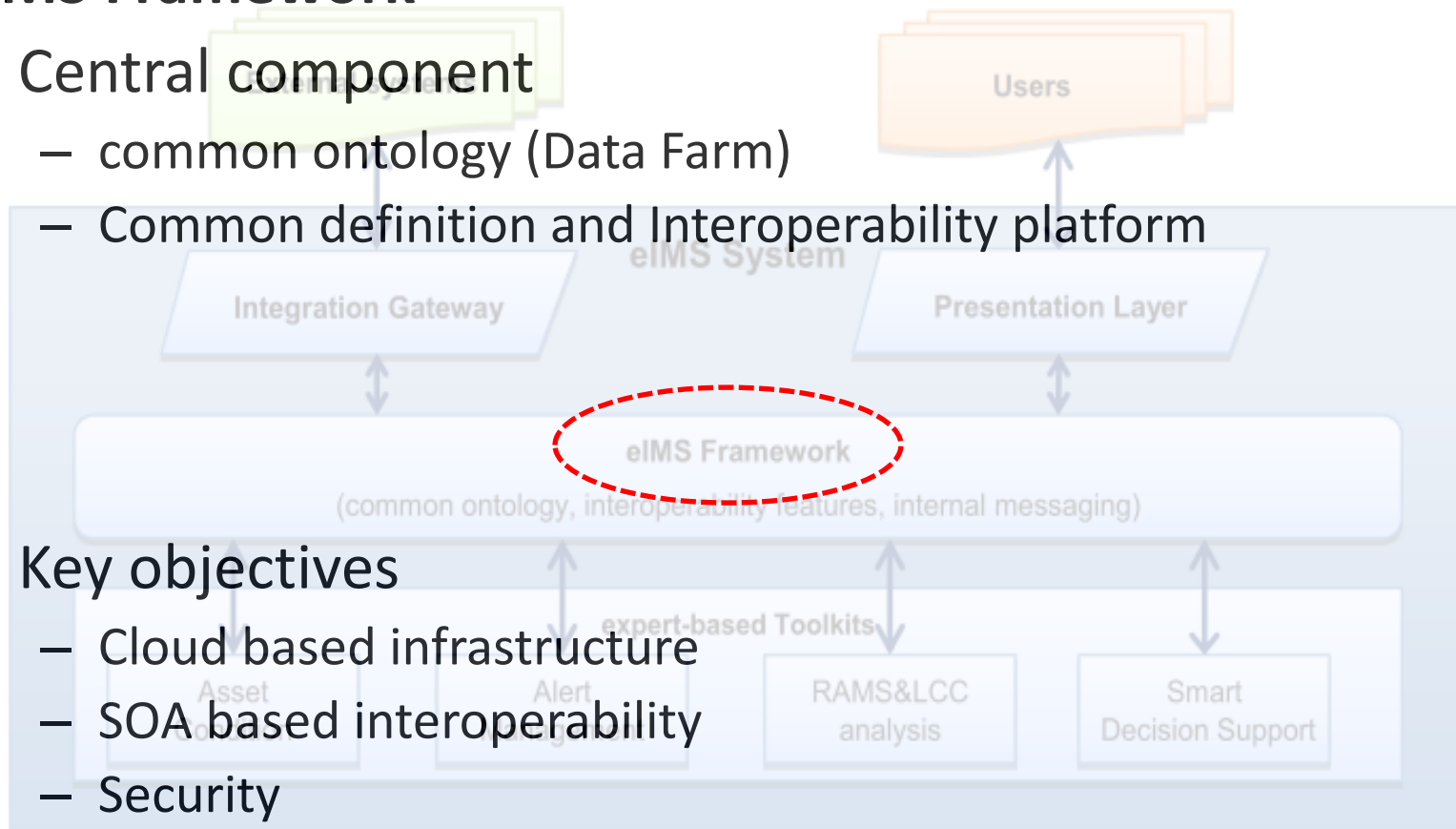


- Expert-based
- Integrated platform
- Cloud-based
- Service Oriented Architecture
- Common ontology
- Single Page Application



eIMS Framework

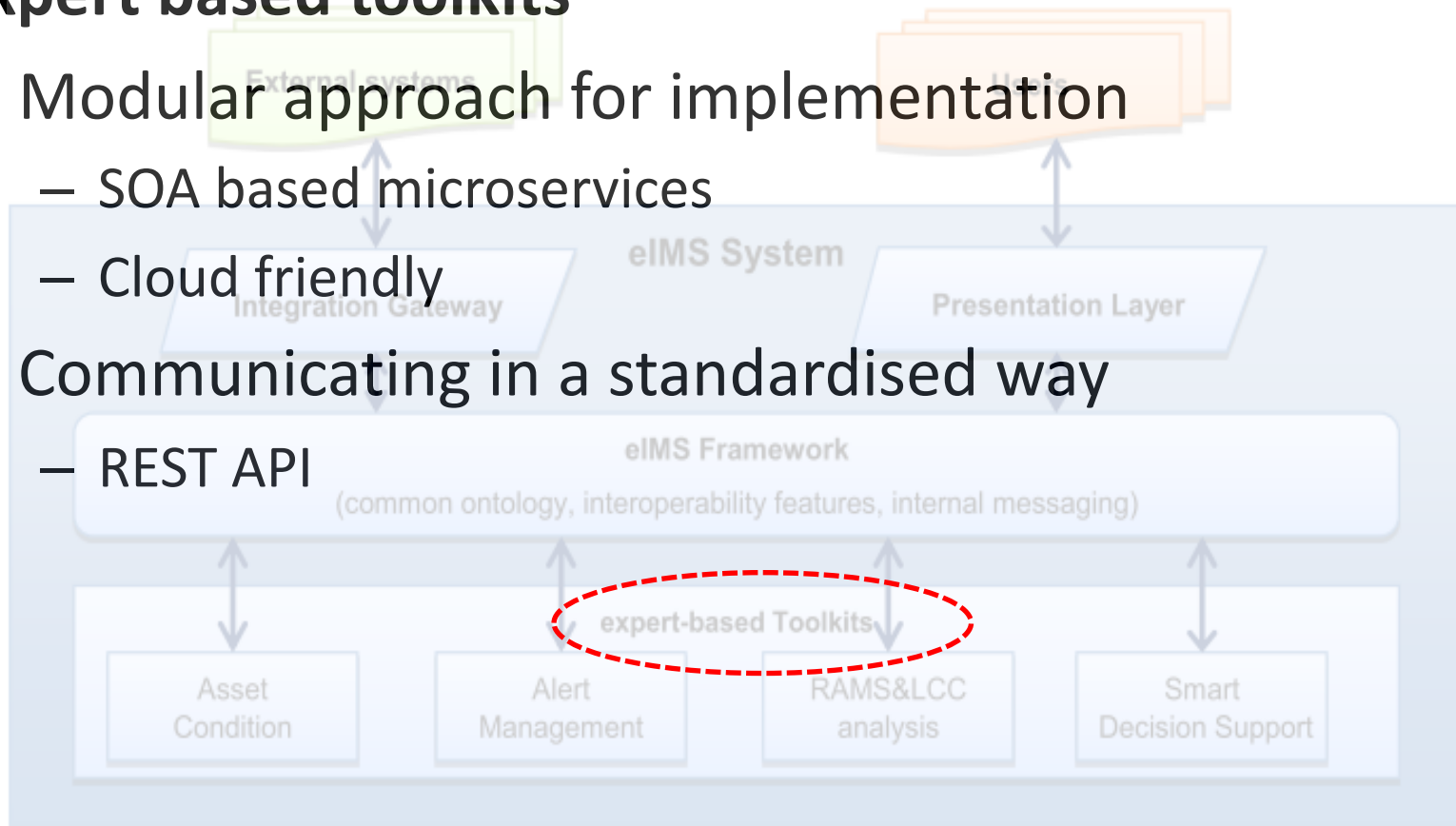
- Central component
  - common ontology (Data Farm)
  - Common definition and Interoperability platform

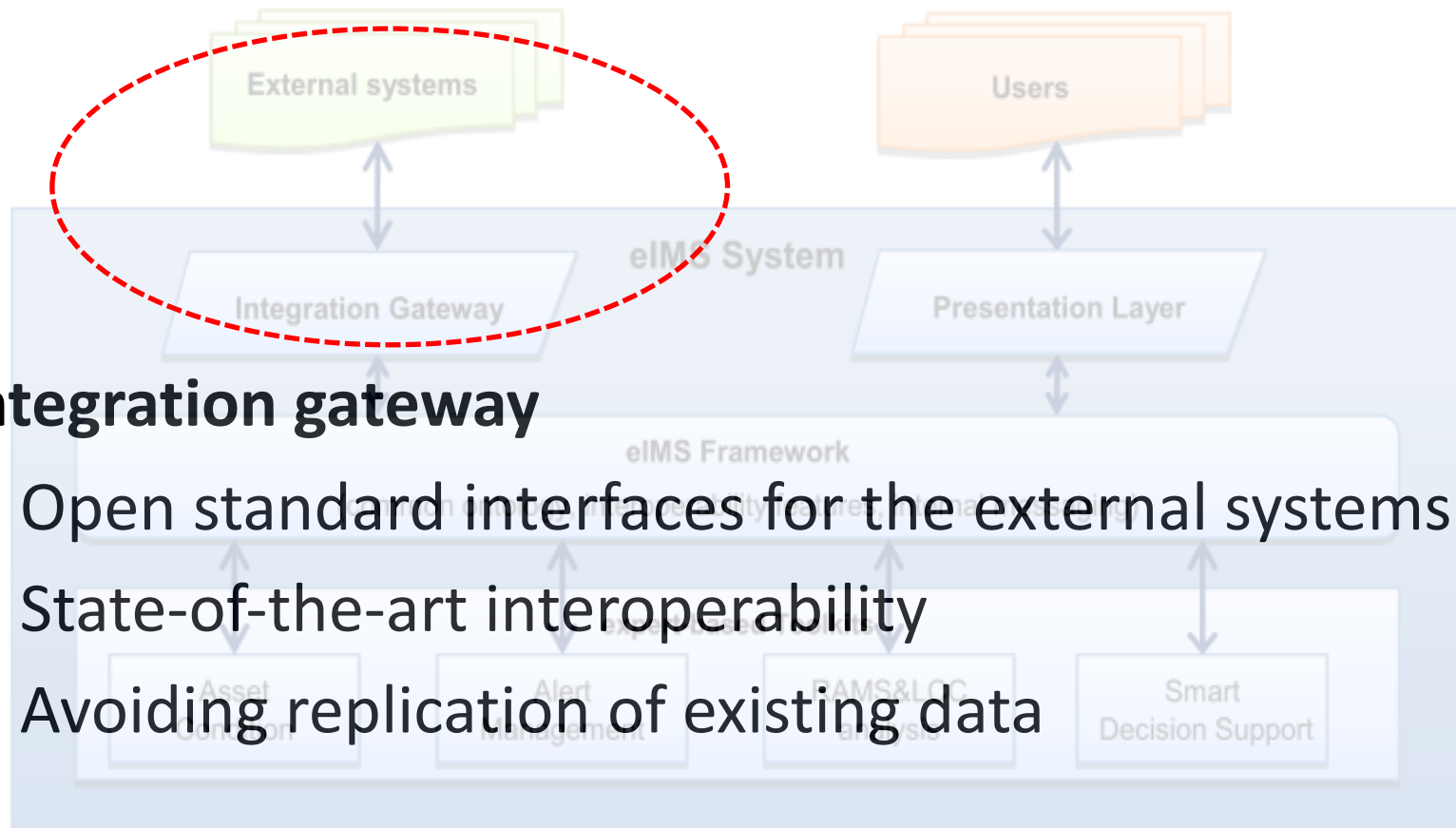


- Key objectives
  - Cloud based infrastructure
  - SOA based interoperability
  - Security

Expert based toolkits

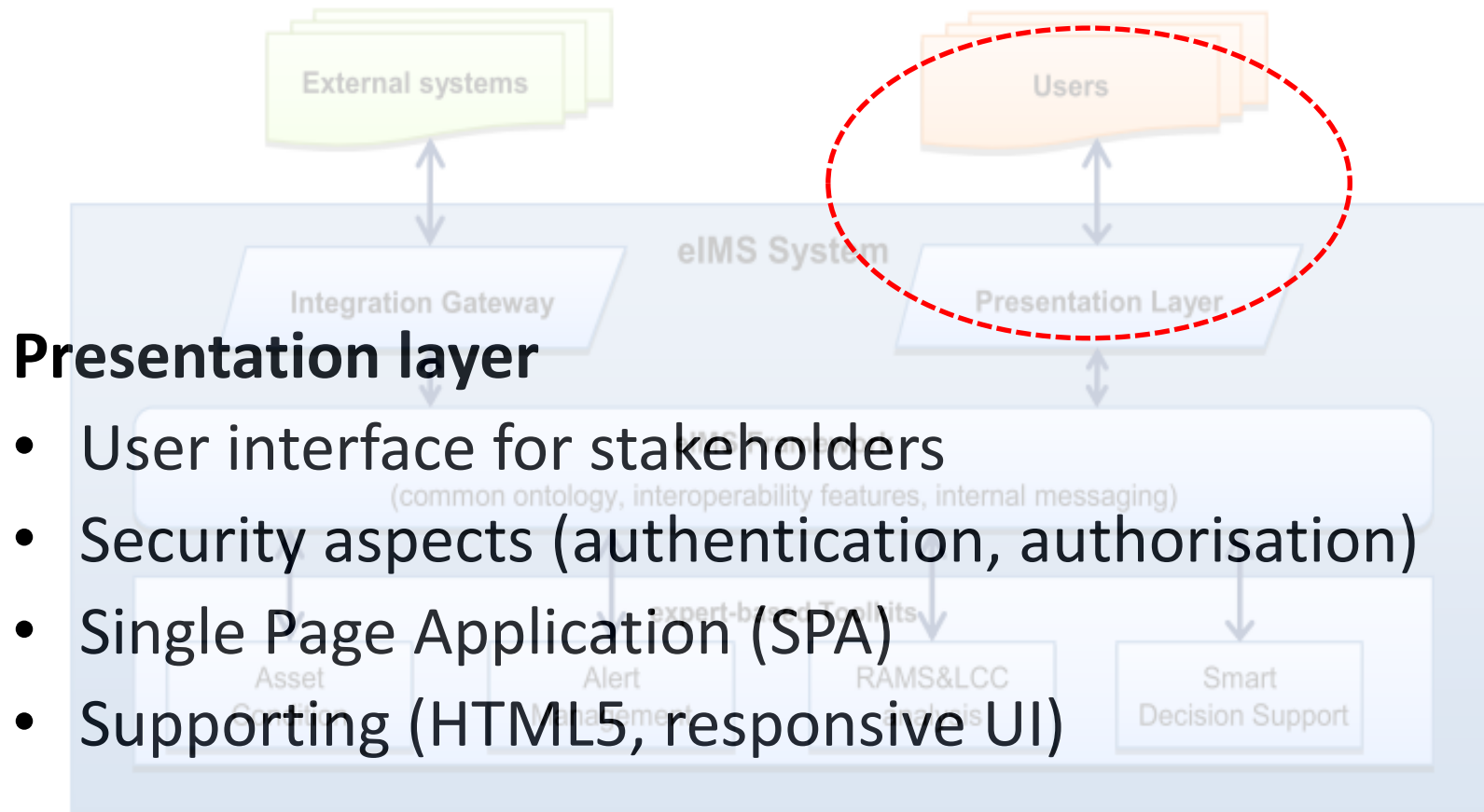
- Modular approach for implementation
  - SOA based microservices
  - Cloud friendly
- Communicating in a standardised way
  - REST API





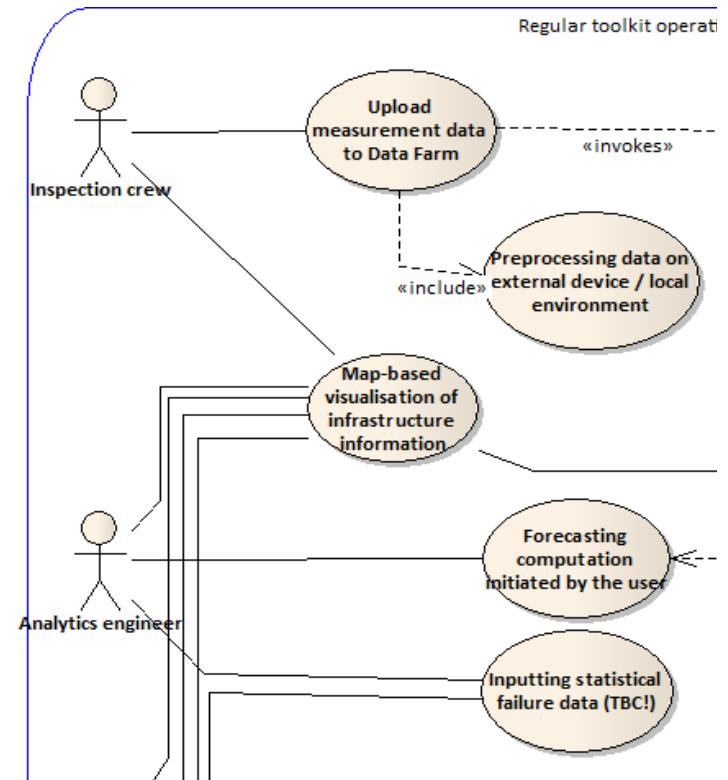
Integration gateway

- Open standard interfaces for the external systems
- State-of-the-art interoperability
- Avoiding replication of existing data



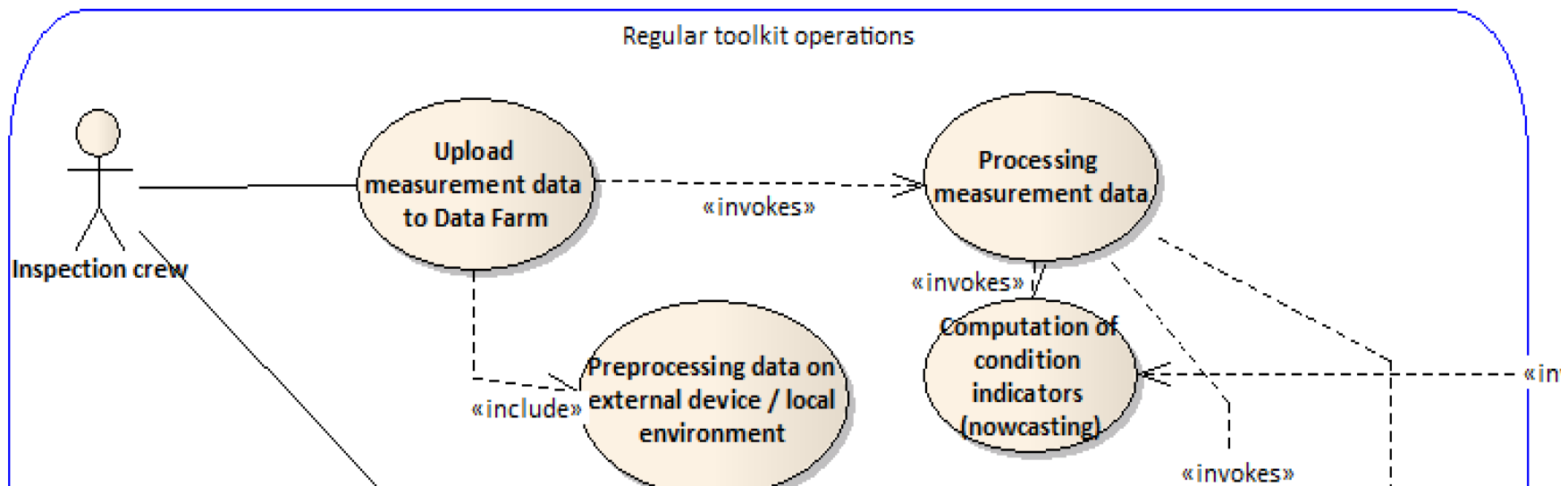
Use Cases and Actors

- Interaction between a role and the system
- Unified Modelling Language (UML)
- The Actors
(Analytics engineer, Maintenance engineer, Inspection crew, Works manager, Inspection crew, Designer, Infrastructure owner, System administrator, eIMS internal scheduler)
- The Use cases
  - Regular toolkit operations
(Upload measurement data to Data Farm, Computation of condition indicators, Alert management, ...)
  - Configuration and system administration
(...)



Use Cases and Actors (Example)

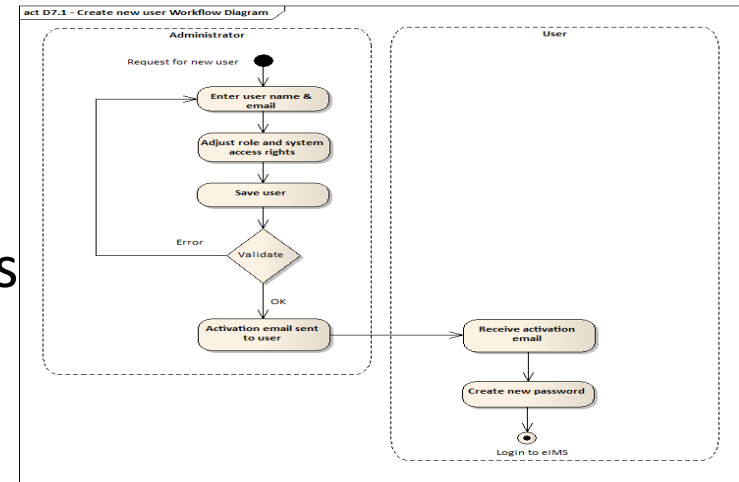
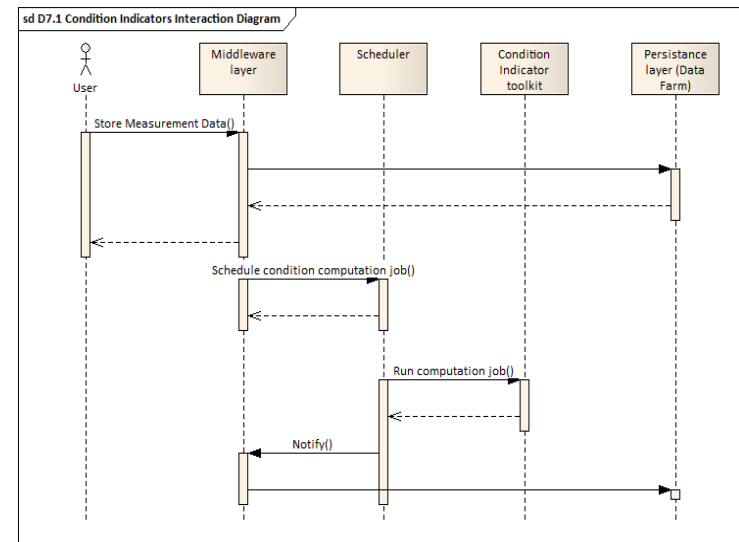
- Actor : Inspection crew



- Use case : Upload measurement data to Data Farm

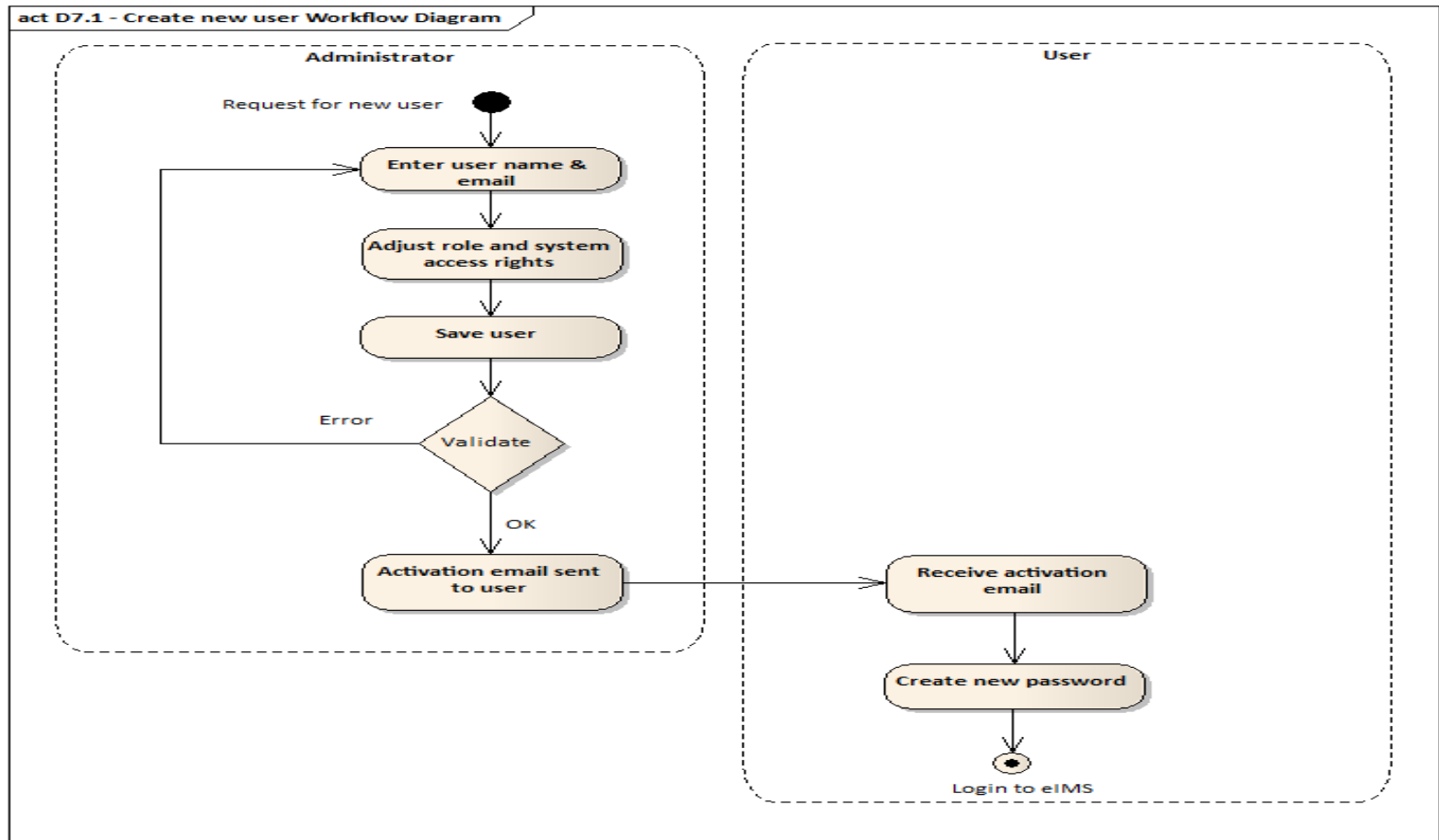
Software design

- Workflow (activity) diagrams
  - Process flow-control, activity orders
  - User oriented view of the system
- Interaction (sequence) diagrams
  - Technical view of the system
  - Interoperating between components



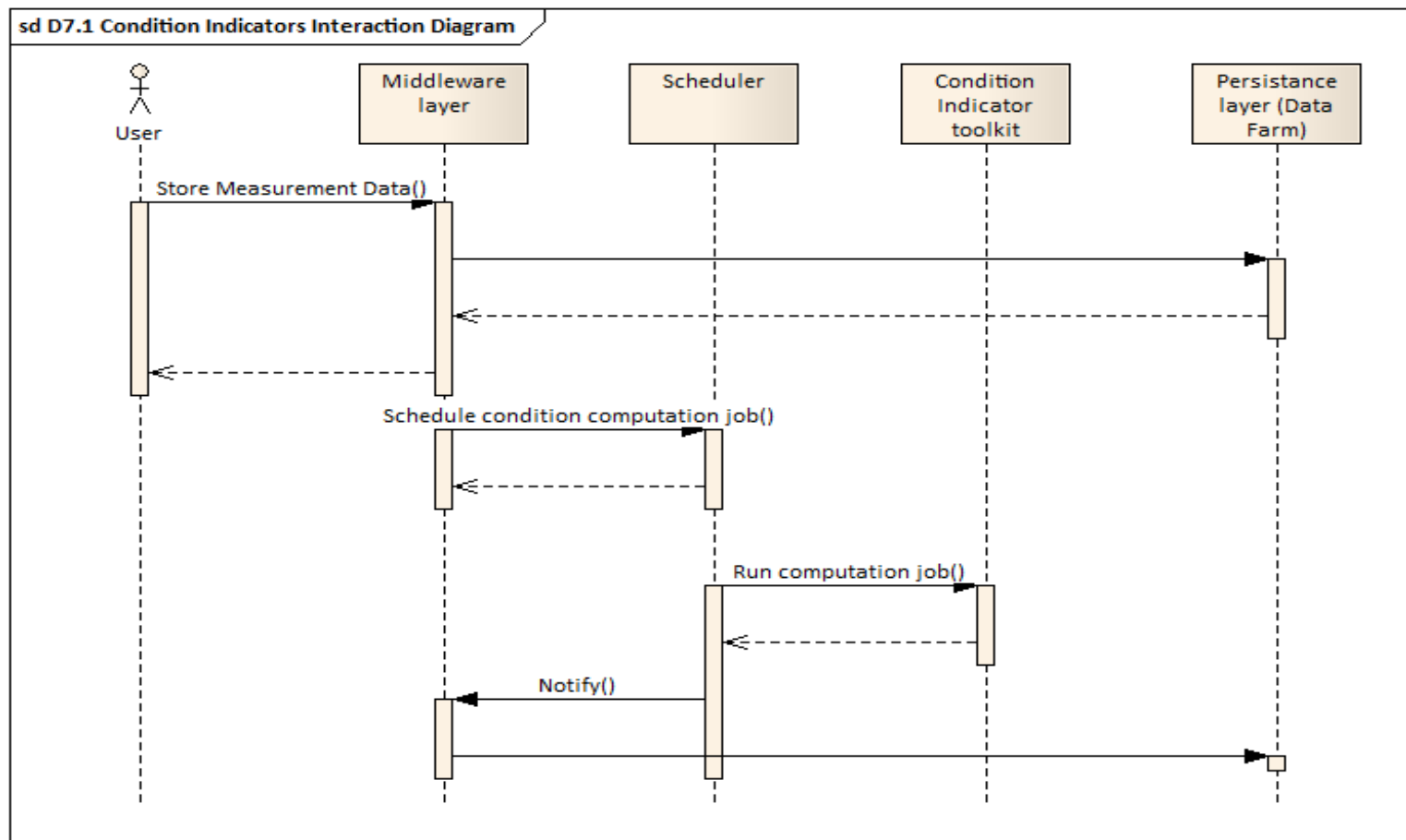
Software design (example)

- Create new user - Workflow



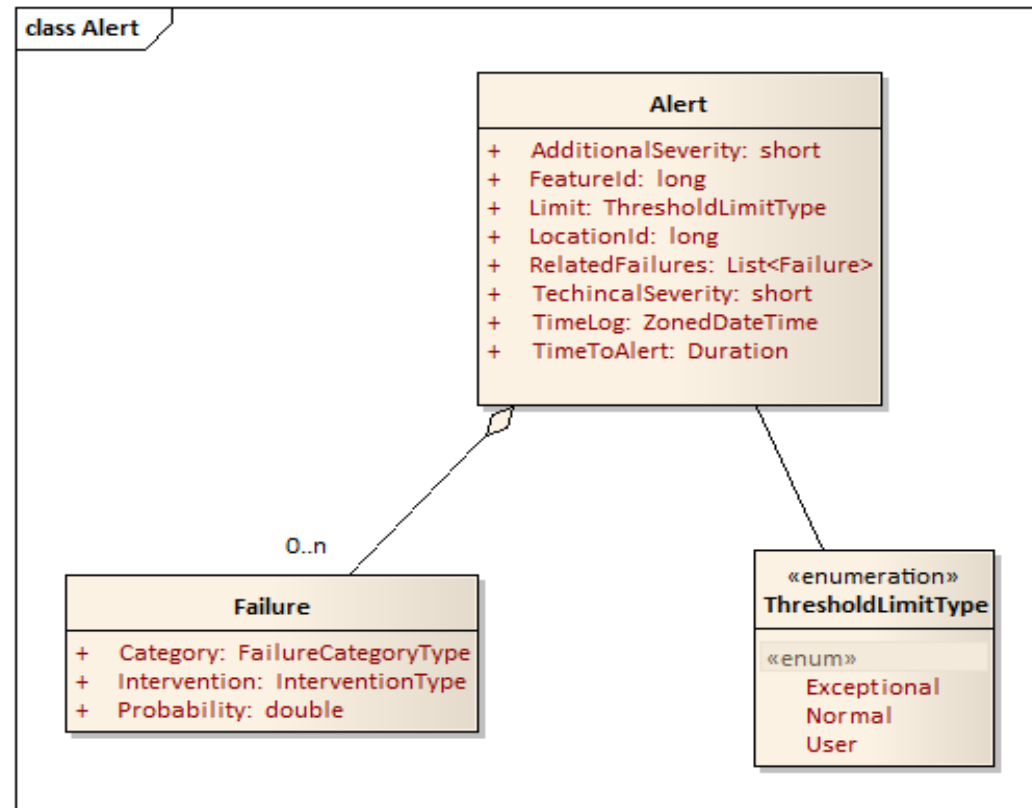
Software design (example)

- Condition indicators - interaction



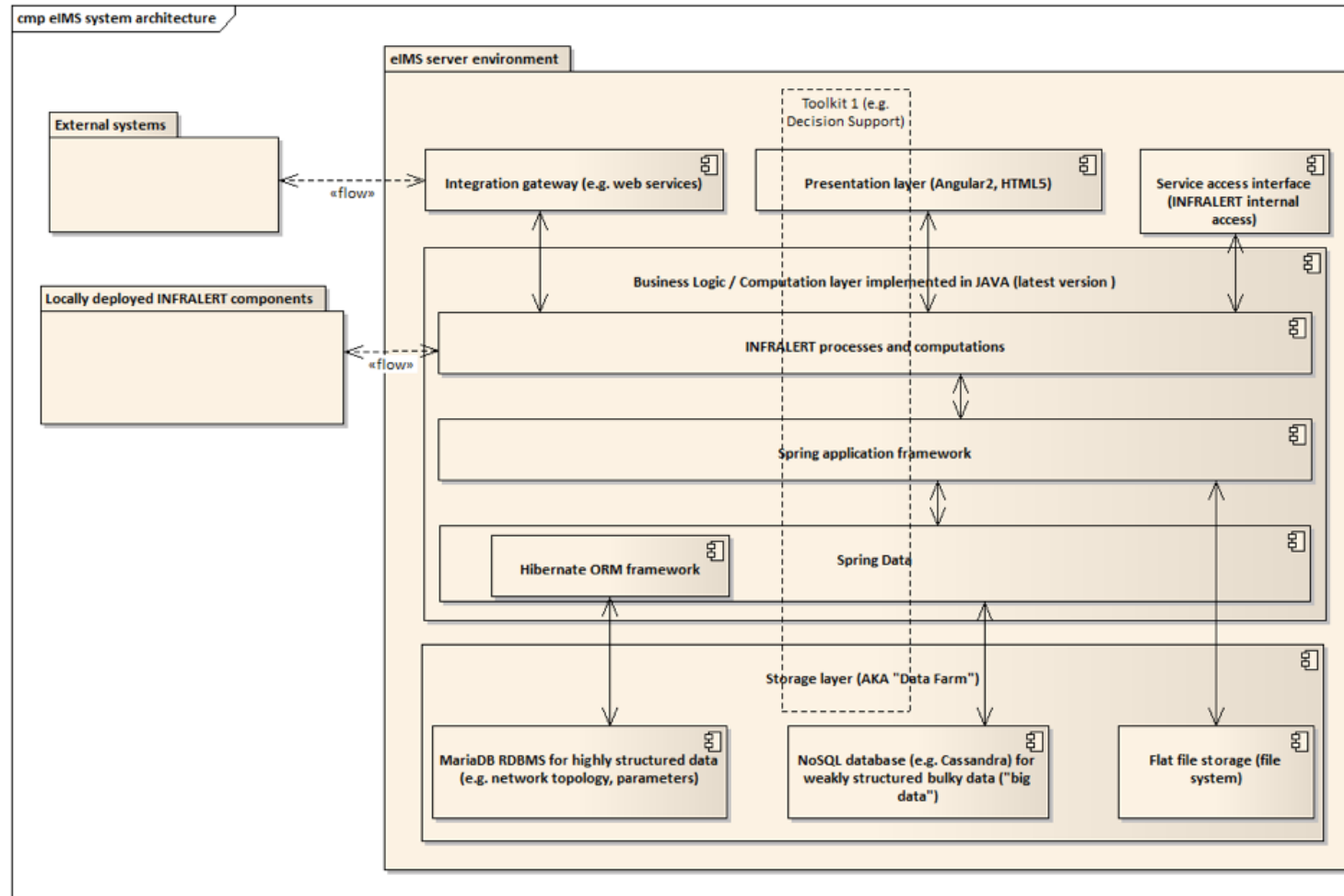
Middleware classes - Java Spring

- Core classes
  - Alerts, Asset, Condition indicator, Degradation model, Failure tree, Maintenance Work, Measurement Data, Probability Distribution, Resource, RAMS/LCC
- Toolkit classes
  - Job interface has to be implemented in every toolkit
- Helper classes
  - Security, Batch, Data



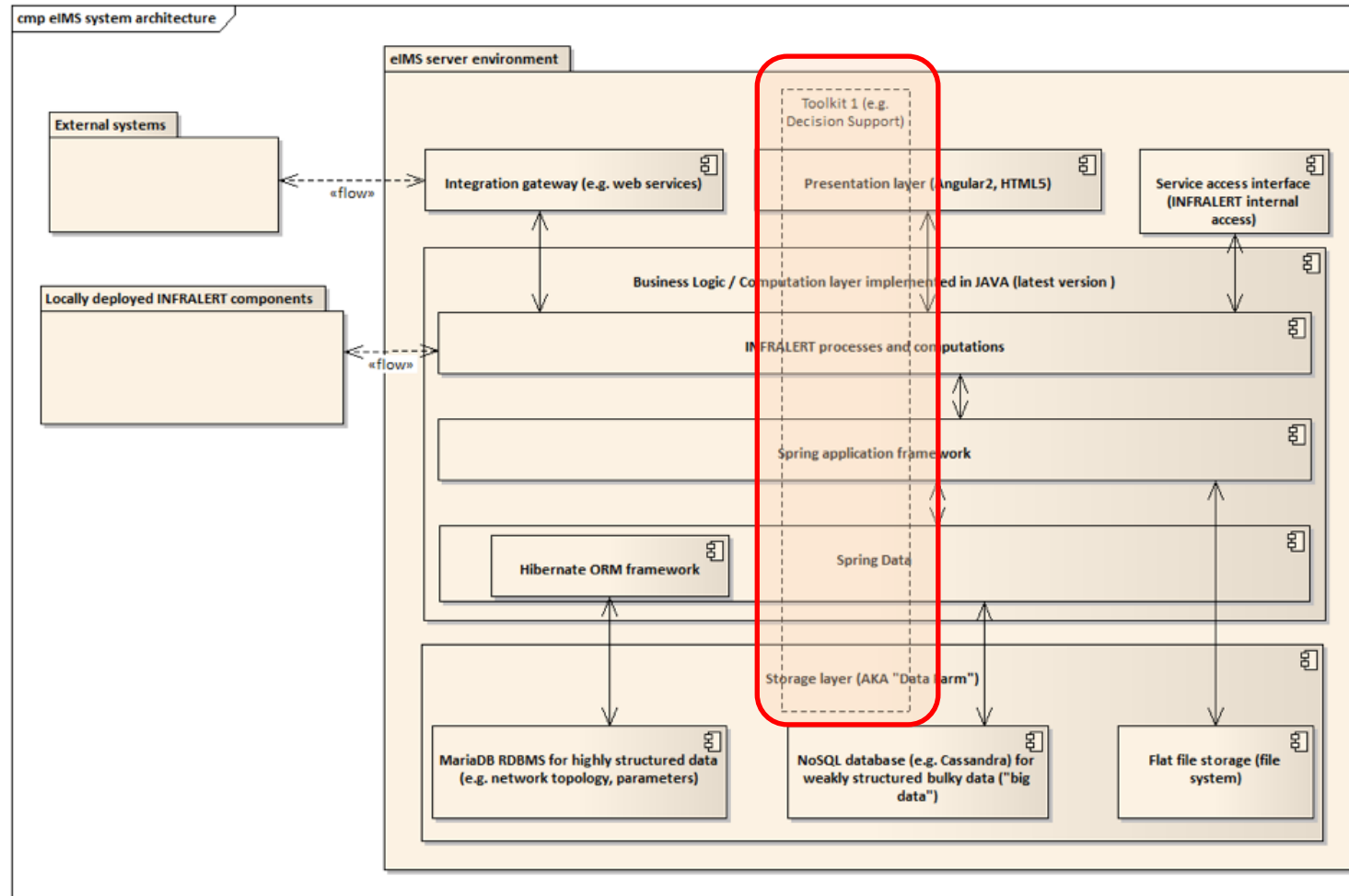
System Architecture

- Presentation layer
- Business logic
- Storage layer



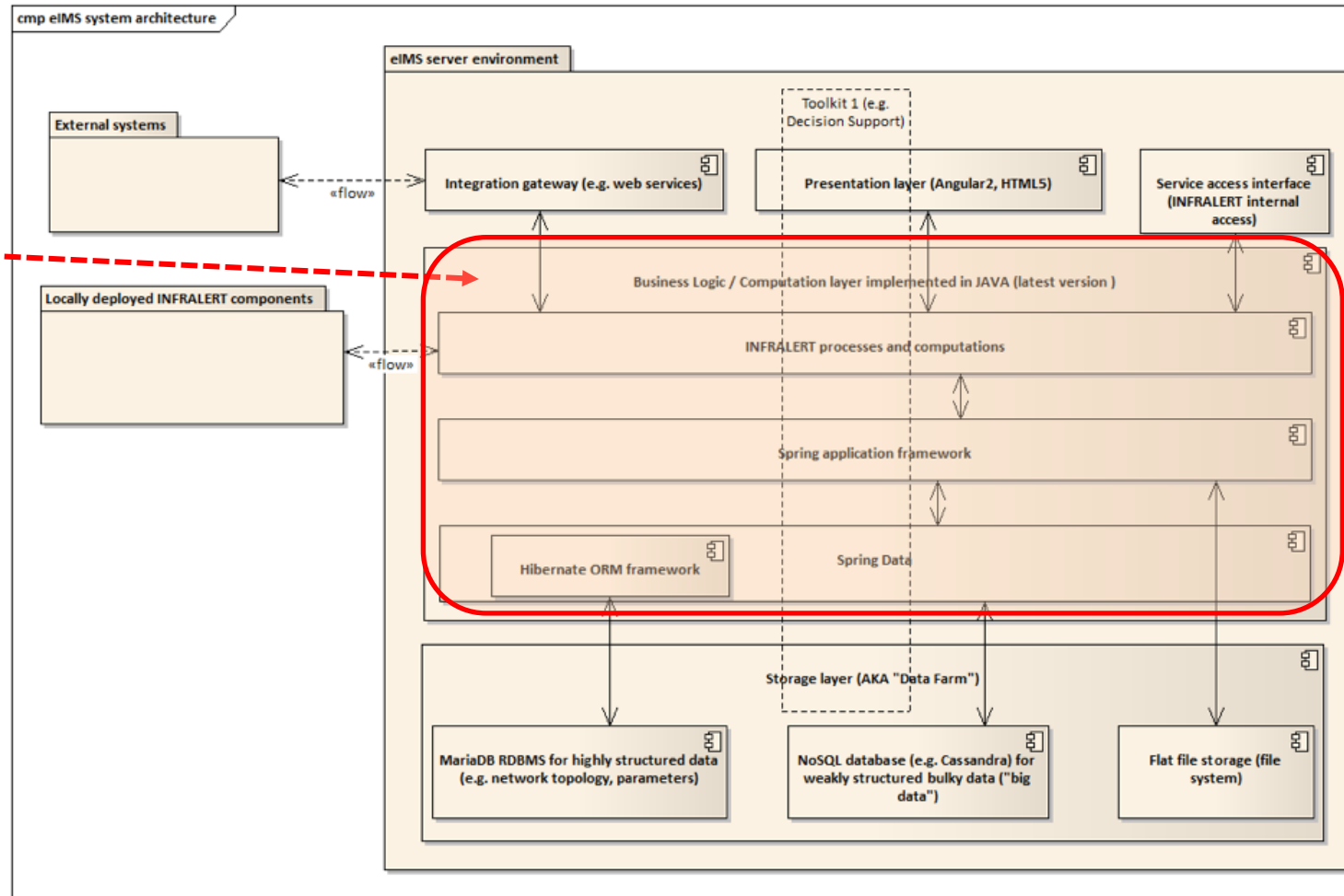
System Architecture

- Presentation layer
- Business logic
- Storage layer



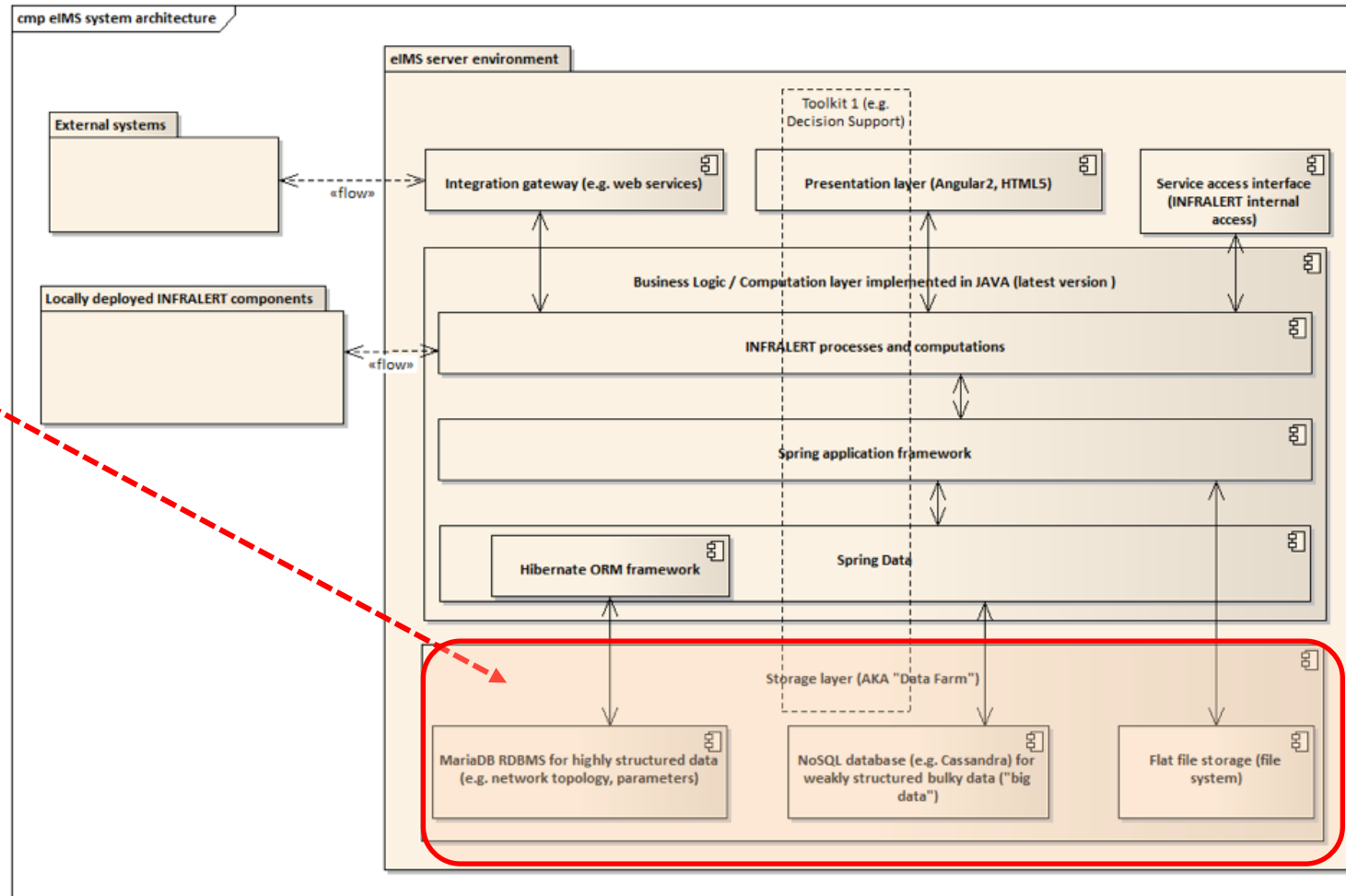
System Architecture

- Presentation layer
- Business logic
- Storage layer



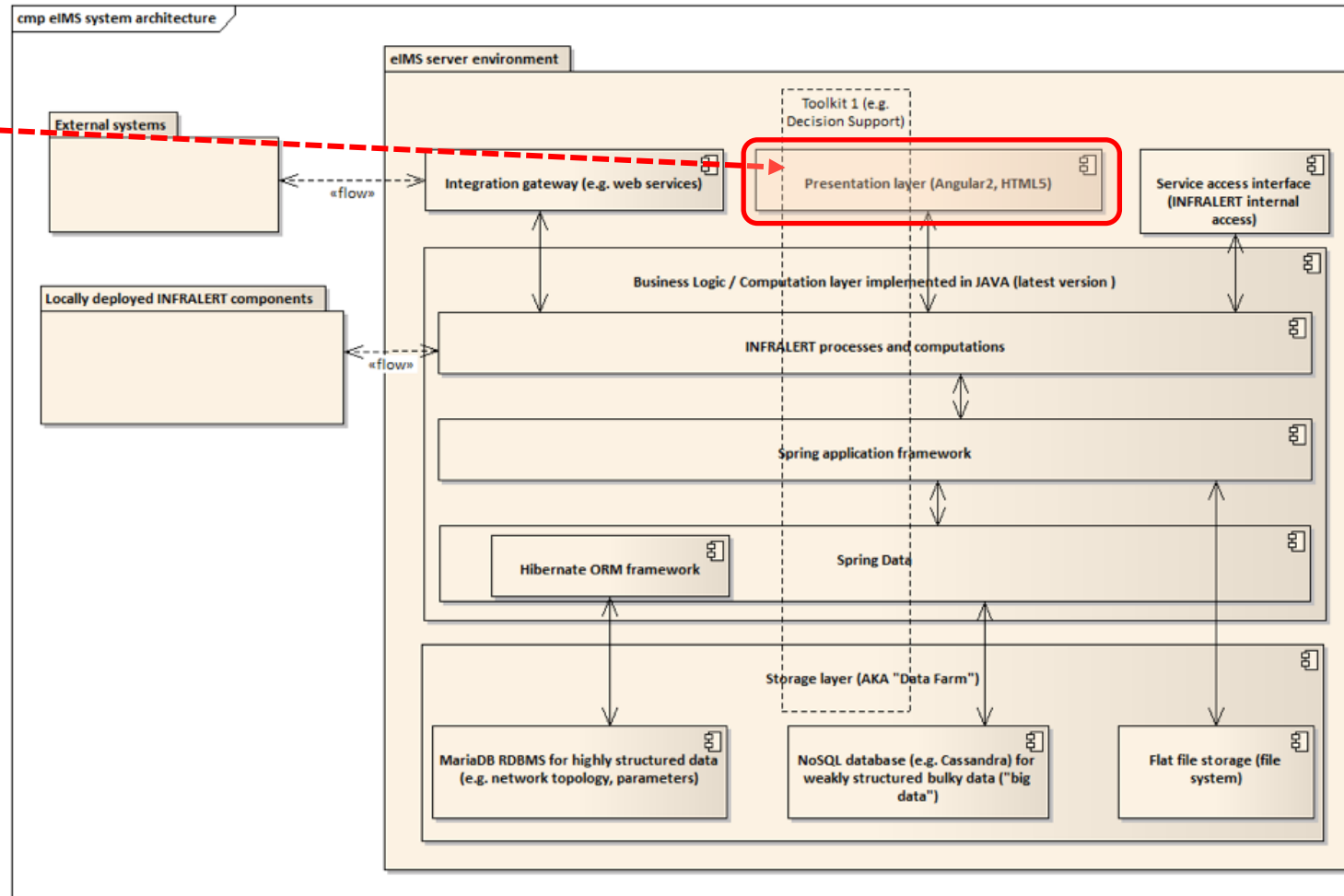
System Architecture

- Presentation layer
- Business logic
- Storage layer

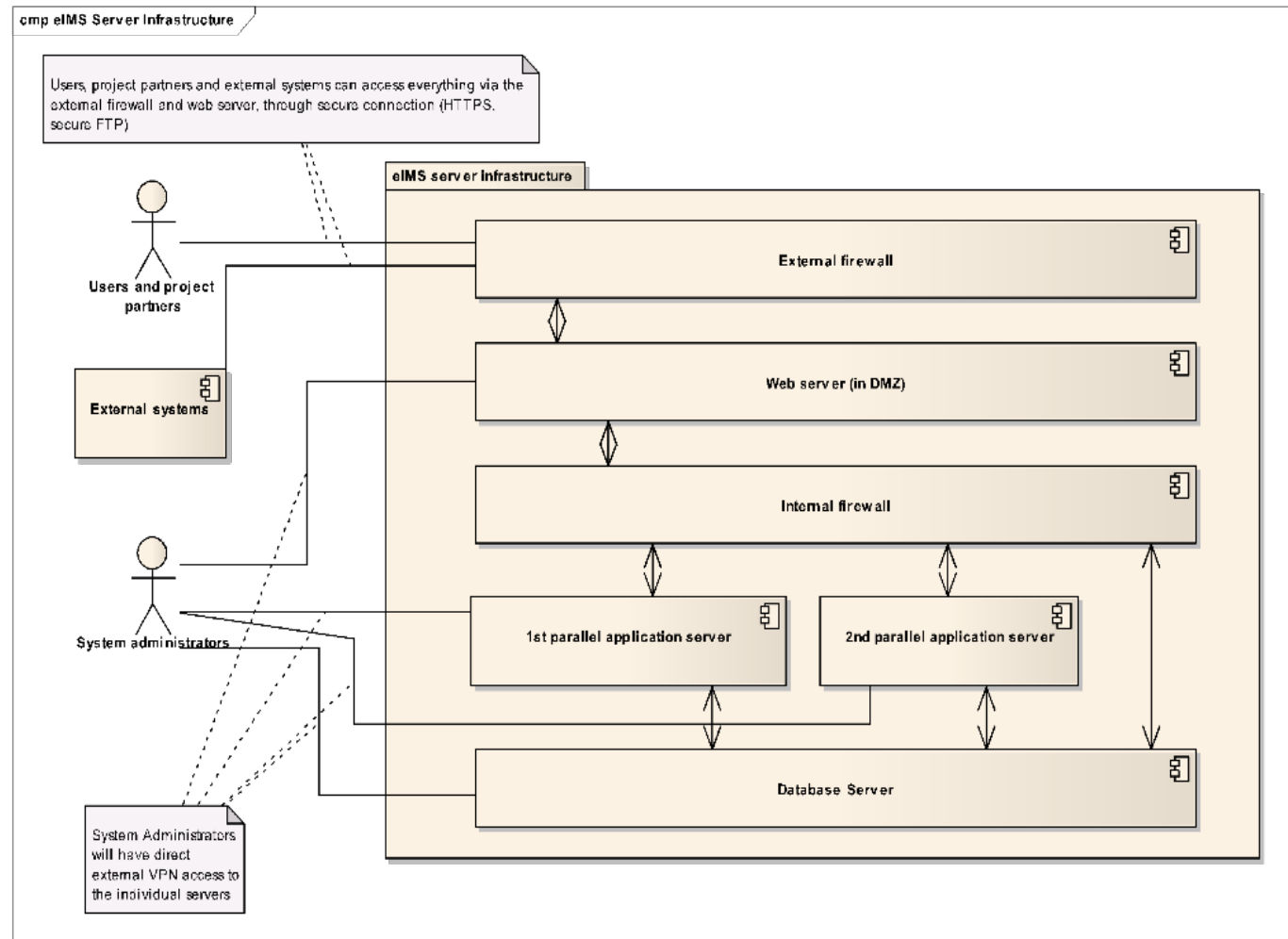


System Architecture

- Presentation layer
- Business logic
- Storage layer

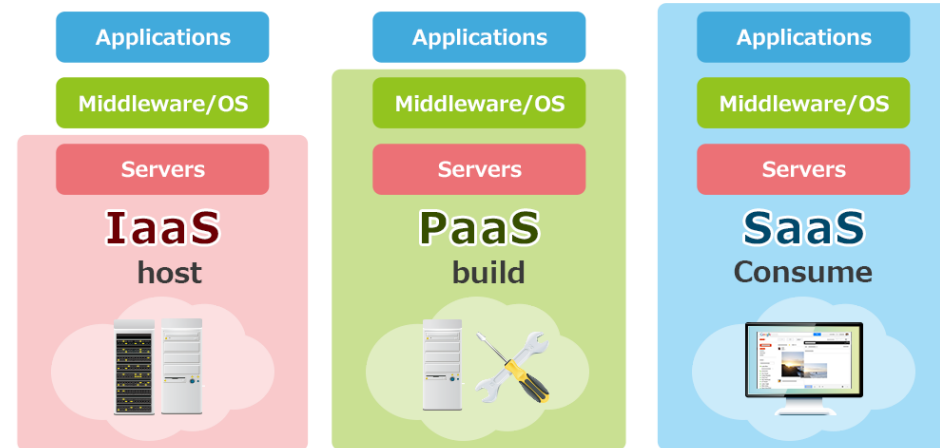


- Scalability
- Accessibility of Servers
- Security and Privacy
- High availability (HA)
- High performance



Horizontal scaling (Cloud based, Microservices)

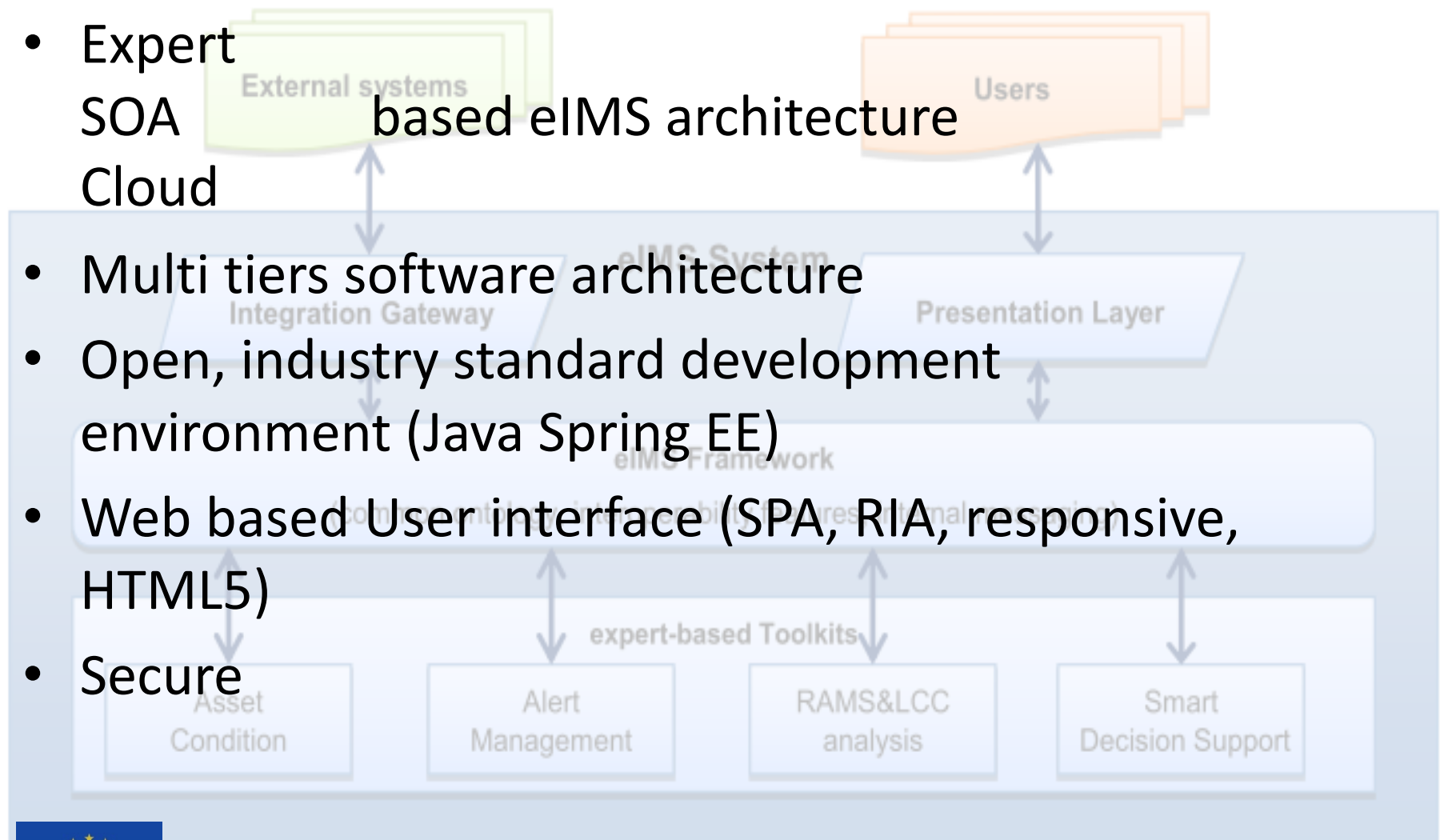
- Infrastructure as a Service (IaaS)
  - Whole application (all microservices) runs on the virtual machines
  - Scaling means firing up more virtual machines
- Platform as a Service (PaaS) – Cloud Foundry
  - Provides an OS and tooling to host the application
  - Handles scaling the application, no need for configs for separate VM instances
  - Microservices can be scaled independently
- Software as a Service (SaaS)
  - The whole infrastructure is hidden, the user can subscribe for the solution





Conclusion

- Expert SOA based eIMS architecture
- Multi tiers software architecture
- Open, industry standard development environment (Java Spring EE)
- Web based User interface (SPA, RIA, responsive, HTML5)
- Secure





www.infralert.eu



András JUSZT



Juszt.andras@regens.hu