UrbanFlood

Common Information Space:

a framework for creating and hosting
Early Warning Systems

Bartosz Baliś, Tomasz Bartyński,
Marian Bubak, Tomasz Gubała,
Marek Kasztelnik, Piotr Nowakowski,
Jeroen Broekhuijsen

ACC Cyfronet & Department of Computer Science
AGH University of Science and Technology
Krakow, Poland

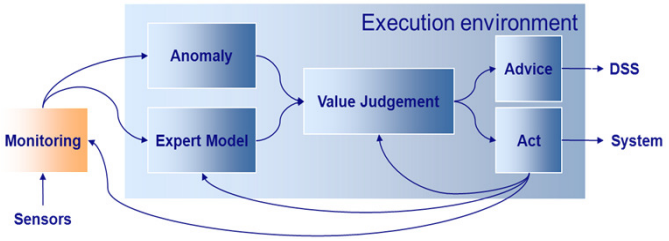
{bubak,balis}@agh.edu.pl, M.Kasztelnik@cyfronet.pl

AGENDA

- Motivation & goals
- Requirements
- Early Warning System – the CIS view
- CIS technology
- Example: Dike Monitoring Early Warning System

Motivation & Goals

- Facilitate creation, execution and interoperability of Early Warning Systems
- **E**arly **W**arning **S**ystem: any system working according to four steps:
  - Monitoring
  - Analysis
  - Judgment
  - Action



- Example: **environmental monitoring**
  - Focus of UrbanFlood: flooding due to dike failures
- But also: cloud infrastructure monitoring, EWS self-monitoring

General properties / requirements

- Existing sources of live and archive sensor data
- Existing legacy applications
- Enable interoperability
  - Between EWS components and different EWSs
- Complex scenarios that involve:
  - Real-time processing of sensor data
  - Decision-making, automatic or by human interaction
  - Compute-intensive simulations
  - What-if scenarios
- High priority (urgent) computing
- Sharing of limited resources

Translated to system requirements…

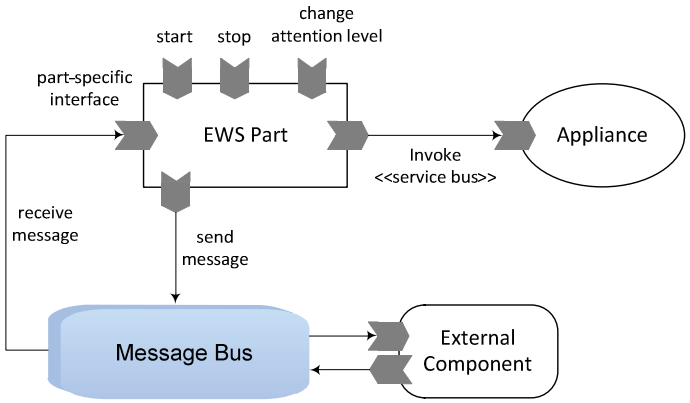
- Application integration
  - Heterogeneous, legacy componets
- Integration of components into ***composites***
  - Scientific / business workflows
  - Integration patterns
- Dynamic resource allocation management
- Metadata
  - Domain (applications, data)
  - System information (resources, data sources, running tasks)
  - Provenance

Early Warning System – the CIS view

- Each EWS is composed of:
  - Parts
  - Appliances
- **Appliance**: application component exposed as a service and wrapped into a virtual image
  - Interface described in WSDL
  - Messages exchanged in XML
  - Integration technology NOT limited to Web Services / SOAP
    - JMS, FTP, REST, …
- **EWS Part**: composite integrating and orchestrating control and data flow between appliances (and possibly other parts)

EWS Part

- Generic operations:
  - Start, stop, change alert level
- Part-specific interface
- Invokes appliances

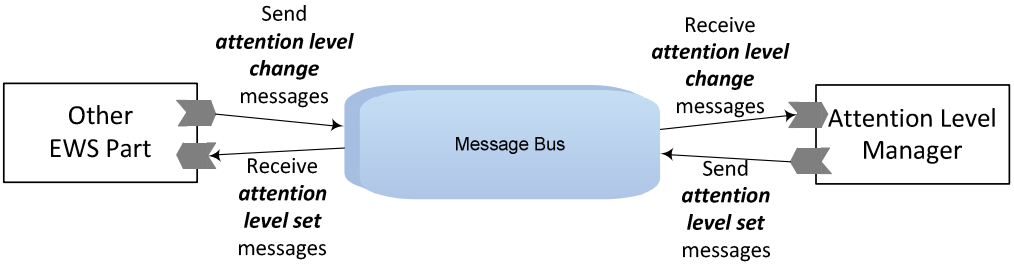


- Sends messages to and receives from a message queue
- External components can also use the message queue to send/receive messages to/from the EWS part

Loose coupling

- Parts are loosely coupled
- Communicate via the message queue
- No direct dependencies between parts
- Facilitates extendibility of the EWS
  - Addition of new parts that further process already published data
  - Connecting new visualization front-ends
  - Can even be done at runtime

Attention level



- General state of every EWS shared by all its parts
- All parts may send **attention level change**' (raise or decrease) message
- **Attention Level Manager** (general-purpose specialized EWS part) receives the messages, makes decision, and may send **'attention level set**' message
- Other EWS parts may adjust their operation depending on the alert level

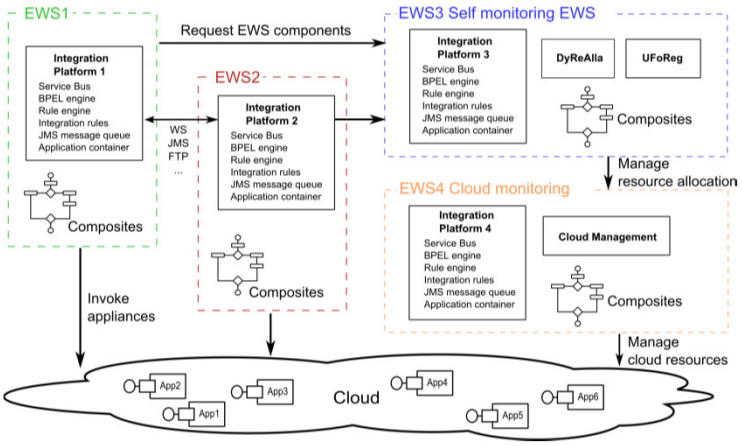
CIS technology

- CIS Core: Integration Platform (**PlatIn**)
- CIS Services:
  - Dynamic resource allocation (**DyReAlla**)
  - Metadata Registry (**UFoReg**)

High-level architecture (multiple EWSs)

Each EWS has

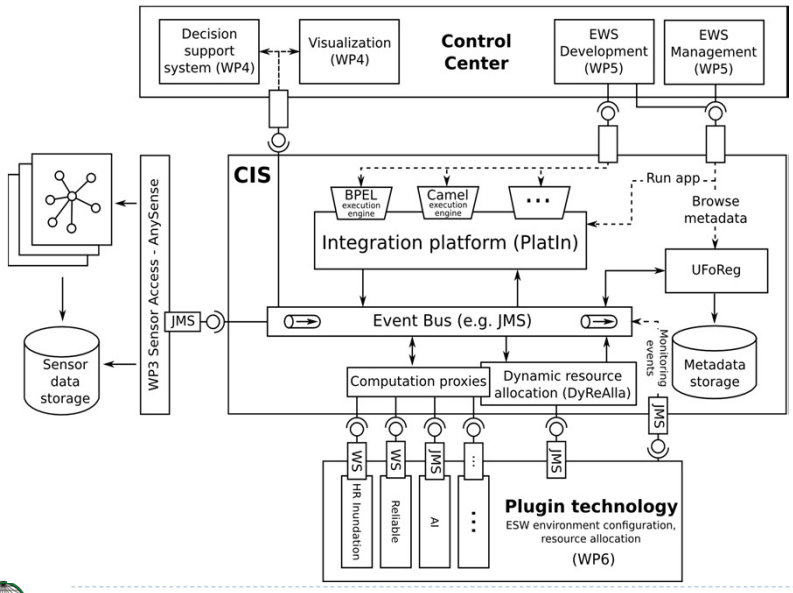
- Its own instance of **PlatIn**, the CIS integration platform,
- Specific appliances in the cloud
- EWS Parts: Composites which orchestrate the execution of appliances



Multiple EWSs:

- EWS1, EWS2: two instances of Dike Monitoring EWS
- EWS3: Monitoring and management of other EWSs (**DyReAlla** and **UFoReg**)
- EWS4: Cloud monitoring and management

Functional architecture



PlatIn – Integration Platform

- Hosting platform for EWSs
  - Enterprise Service Bus (Glassfish ESB)
  - Workflow engine (BPEL)
  - EIP engine (Camel)
- Runtime EWS-level management
  - Starting / stopping / pausing EWSs
- EWS monitoring
  - Including provenance logging

DyReAlla – Dynamic resource allocation

- Optimization of application performance and resource utilization
- Management at the level of virtual appliances
  - Start VMs requested by EWSs
  - Stop/Pause VMs
  - Adjust resource allocation to VMs
- Dynamic adaptation of the execution environment
  - Ocurrence of urgent, high priority computing
  - Ocurrence of new EWSs

UFoReg – Metadata registry

- General purpose metadata storage
- Distinction to two metadata types:
  - Domain metadata
  - Integration metadata
- Several purposes in CIS:
  - Data / resource discovery metadata
  - Provenance repository
  - Domain metadata

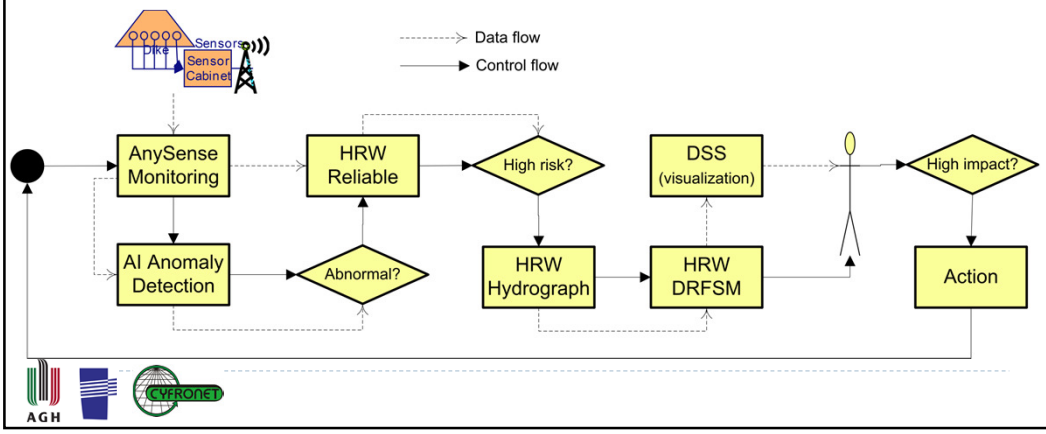
Creating an EWS template using CIS technology

- **Adapt** existing (legacy) **components** to enable communication with the CIS
  - Sensor data sources
  - Legacy applications
- **Define composites** (workflows, integration rules) which organize the components into complex scenarios
- **Wrap componets** into virtualized appliances
- Deploy composites & virtual images of components

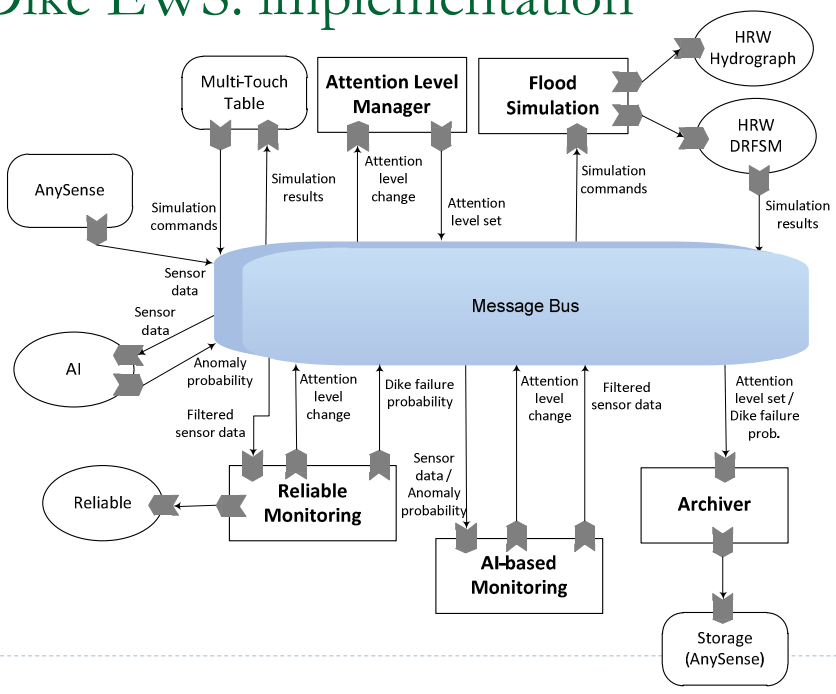
Configure & run → EWS instance

Example: Dike monitoring EWS

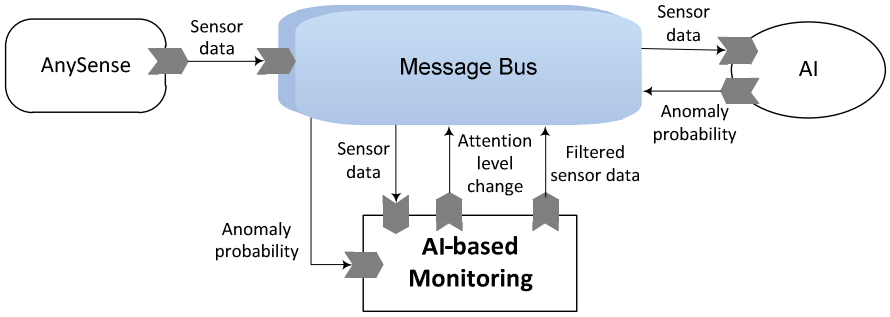
- Monitoring of dikes using wireless sensors
- AI-based detection of sensor signal anomalies
- Dike failure prediction
- Simulation of inundation due to failure
- Visualization and user interactions on Multi-touch Tables



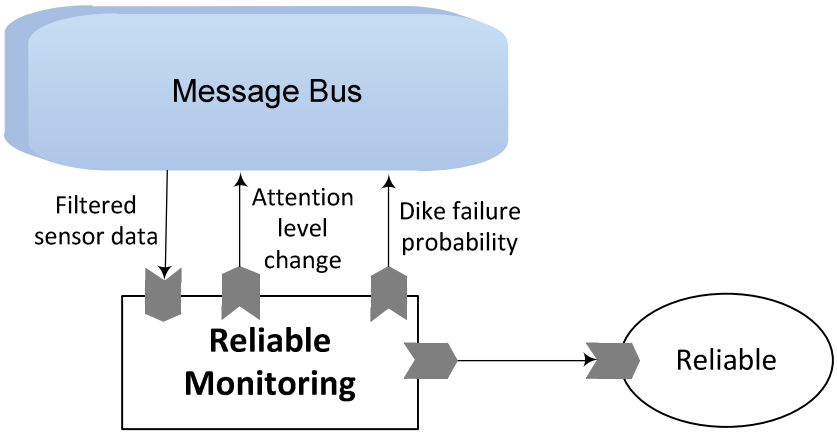
Dike EWS: implementation



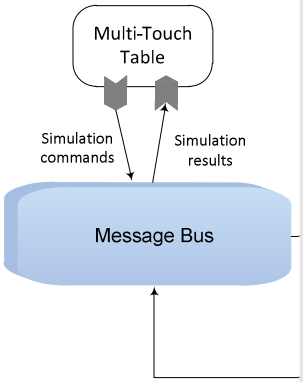
Parts: AI-based Monitoring

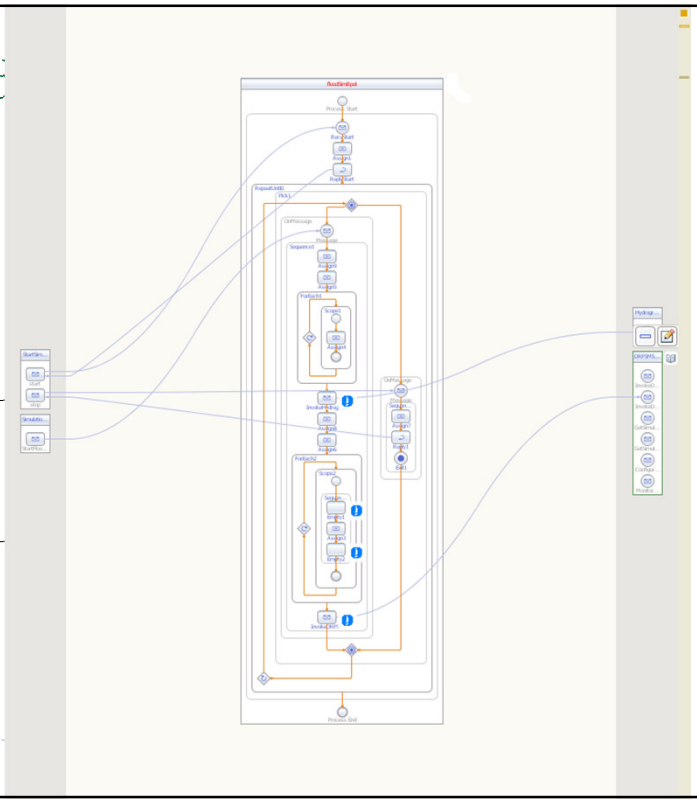


Parts: Reliable Monitoring

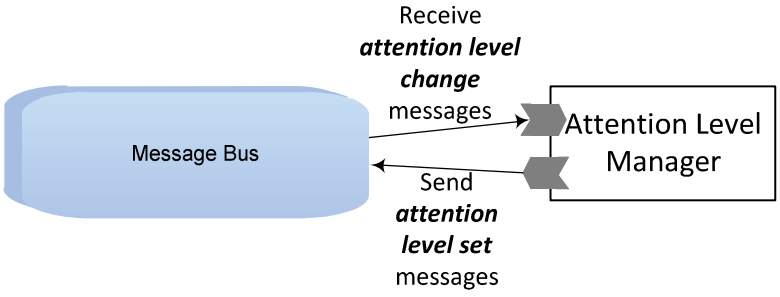


Parts: Flood

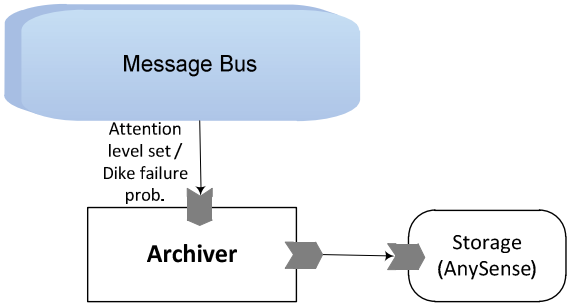




Parts: Alert Level Manager



Parts: Archiver



Summary

- CIS: generic framework for creating and hosting Early Warning Systems
- Main challenges:
  - Integration of existing components
  - Orchestration into workflows
  - Dynamic resource allocation
  - Metadata management
- Same technology applied to different types of EWSs
  - Dike Monitoring EWS
  - Self-Monitoring EWS
  - Cloud Monitoring EWS

Future work

- Dynamic resource allocation
- Self-monitoring
- Provenance tracking
- Security
- Fault tolerance
- Auxiliary tools, templates, etc.
- …