Robust monitoring

- 2009.02.01.2 Architecture and prototype dike monitoring system

Version number	*1*
Date	*December 23, 2009*
Status	***final***
Owners	*Deltares, TNO, IBM*
Confidential	*Not confidential*

Table of contents

Document Management

Contributors

Name	Initials	Company	Role
Erik Langius	IL	TNO	Project manager
Bram van der Waaij	BW	TNO	IT Scientist
Wiltfried Pathuis	WP	TNO	IT Scientist
Bram Havers	BH	IBM	IT Architect
Joost Zeinstra	JZ	IBM	IT Architect

Confidentiality

Not confidential

Acknowledgment

This research was carried out within the Flood Control 2015 program. For more information please visit http://www.floodcontrol2015.com.

1 Introduction

This document defines the solution architecture of a Flood Control System. In the project plan of Flood Control 2015 – tranche 2009 this deliverable is has the Dutch title “Architectuur en prototype dijk monitoring systeem”

Besides this document also several software components were developed and configured to implement a demonstrator according to the architecture is described here.

1.1 Architectural approach

This paragraph describes architectural methods that are used to design the Robust Monitoring Flood Control System.

- IBM’s Enterprise Architecture Method for outlining the structure of the solution and is aligned with The Open Group Architecture Framework (TOGAF)

- IBM’s Unified Method Framework is based on the Open Unified Process and provides the
    - process (the steps) to create the solution
    - work product that describe the solution.

The Open Unified Process (OpenUP) is a part of the Eclipse Process Framework (EPF), an open source process framework developed within the Eclipse Foundation. The base concept is iterative creation of artefacts (work products), which are well defined and described.

The primary reason for developing an enterprise architecture is to structure the process from business goals (the agreed Flood Control 2015 project plan) through requirements gathering, solution design and implementation.
When working on the architecture you can relate every aspect of the system back to the original goal.
The diagram below shows a simplified Enterprise Architecture. The Mission, vision and strategy of Flood Control 2015 is the base of the project plan. The project plan specifies the goals to achieve to implement the vision and strategy. In the solution that will be designed in Robuust Monitoring we have to take all five aspects into account. The organisation (who will use the system), the processes (how will they use it), what is their information need and how is that supported by applications. These application have to run on the hardware that will be specified in this document.

Figure 1-1 Simplified enterprise architecture framework (source: Gemeente Amsterdam)

The model above is a simplified view. IBM extends this model with the IBM Enterprise Architecture method which is described below and on which this document is based.

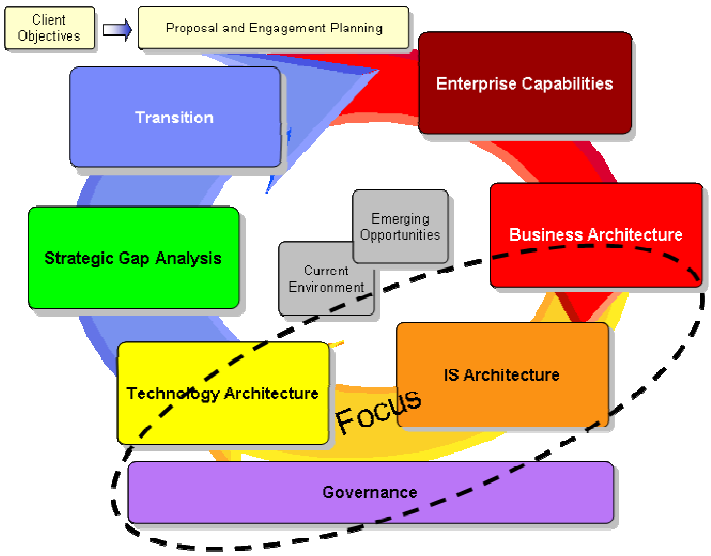

Figure 1-2 IBM Enterprise Architecture

The focus will be on documenting the minimum required information in the circled area.

- **Business Architecture:**
  The business architecture defines the business requirements. It defines which organizations are involved and what their interest is. Use cases will describe how actors (people with a specific role in these organizations) will pursue their professional goals in the proposed furutre scenario. The business architecture ensures that the technical solution will be in line with the overall goals of the Flood Control 2015 program.
  The Business Architecture is described in the Visualisation document.

- **Information Systems Architecture**
  Describes the information that is being used by actors above. Identification of the information and how it flows is key, because that are the items that can be automated and on which innovation will take place. Functional components will be identified that are needed to improve and automate this information flow. Eventually application and databases will be identified to implement the functional components and information stores.

- **Technology Architecture**
  Defines the technical infrastructure that the applications and databases will run on. This is the hardware, network and infrastructure software components (for example systems management or clustering software)

These three architecture views are set by the Flood Control 2015 architecture, to which this document complies.

1.2 Key work products

The IBM Enterprise Architecture approach has a set of predefined work products that are used to describe the solution. Below is a minimal selection of these work products that will be used in this document.

1.2.1 Business Architecture

Captures decisions about which business processes will be supported

- Describes the system as a "black box", focusing on what the system will be required to do;
- Identifies the scope of the system;
- Outlines the requirements;
- Identifies the users of the new system and their characteristics;

The next paragraphs will list the deliverables. The Business Architecture deliverables are described in separate documents:

1.2.1.1 Business Case

In the business case the economical viability of a Flood Control System is investigated. This is described in a separate document: “2009.02.02.2 Opschalingsperspectief voor dijkmonitoring systeem”.

1.2.1.2 Requirements

Requirements are separated in functional requirements (what the system should do) and non-functional requirements (reliability, performance). For functional requirements it has to be clear how the system will be used. This is specified by Use Cases. These paragraphs are described in the Visualisation document: “2009.02.02.1 Visualisatie actuele kwaliteit waterkeringen obv sensordata”

1.2.1.2.1 Use Cases

Use cases describe the context in which the envisioned solution will operate and be used. Use cases are the base for specifying the functional requirements. It also supports user interface design.

The Use Cases have been described in the Visualisation document.

1.2.1.2.2 Functional Requirements

The functional requirements are a list of categorized and prioritized functional requirements.

1.2.1.2.3 Non functional Requirements

The definition of the minimum and maximum acceptable criteria for the system.

1.2.2 Information System Architecture

1.2.2.1 Architecture overview

The architecture overview is created early in the lifecycle of a project. It reflects early decisions and working assumptions. It takes the form of an informal, rich picture. Conceptually, it illustrates the essential nature of the proposed solution, conveying the governing ideas and including the major building blocks.

1.2.2.2 System Context Diagram

The System Context identifies in a graphical diagram all information exchanges between the envisioned system and its environment and sources or destinations of data used by the system. Identify all external entities that need to interface to the system. Determine the inputs and outputs need to support each of these entities. These information exchanges and data sources represent constraints that the system under development is bound by. The development team and the client have some control over objects within the system boundary. Establish System Context identifies required interfaces with other systems (i.e. dike monitoring systems, meteorological sources, etc.) and provides context for describing internal aspects of the system.

1.2.2.3 Architectural Decisions

A list of the architectural decisions and their rational.

1.2.2.4 Available Assets

Identify re-usable assets (e.g. reference architecture, similar projects, software packages) that might be relevant for the project. Assess whether available assets could contribute to solving the key challenges of the current project, and are compatible with its constraints.

1.2.2.5 User Interface prototype

Design user interface prototype, concepts, criteria, and design decisions that will apply to and affect the overall application. It will be used to involve the end user. Prototypes are used for many purposes:

- Designing the user interface: A user interface is impossible to get right the first time. User interface design involves iteration. Prototypes give designers a way to explore designs and test them iteratively with users.

- Clarifying user requirements: Users will give more concrete specifications of what they want the system to do and what they don't want it to do if they are presented with a prototype rather than with analysis diagrams.

- Clarifying technical requirements and feasibility: Analysis of user interface designs in the context of the architecture in which they will be implemented typically reveals numerous technical and accessibility requirements and issues.

- Getting buy-in to the project from the project sponsor: Project sponsors tend to be unimpressed by large quantities of analysis documents and diagrams. On the other hand, showing sponsors prototypes early in the development effort can improve the designers' credibility and convey much more information about the system.

- Getting buy-in from the end users: In general, when users feel they have been involved in the design of a product, they are more likely to view it positively.

- Providing an exit criterion for the design phase.

- Verifying the user model, use cases, and scenarios.

This work product is described in the Visualisation document.

1.2.2.6 Component Model

The component model describes the structure of a system in terms of its software components with their responsibilities, interfaces and relationships to deliver the required functionality. The component model is the main artifact documenting the functional view of the architecture and serves as an abstraction for the design.

Component models can be documented at 2 levels:

- At the logical level, focusing on specifying the components' responsibilities and characteristics required to deliver the requirements. These specifications are typically technology and product neutral.
  For example a Message Bus or a modelling framework.

- At the physical level, focusing on how the components will be implemented to meet the previously established specifications. Specified components can be transformed into physical components via custom development, the purchase of products or the reuse of assets.
  For example IBM WebSphere Message Broker or Deltares FEWS

1.2.2.7 Interfaces

Interfaces define the characteristics of required external interfaces with other systems (e.g. dike monitoring systems). This task adds detailed characteristics to the existing system context diagram. Detailed characteristics include number of instances, available access time, type of communication, hardware brand/model, operating system brand, volume and frequencies.

1.2.3 Technology Architecture

1.2.3.1 Operational Model

This Operational Model describes the operational distribution of a system's components onto nodes, the placement of nodes and users across locations, the connections between nodes necessary to support the required interactions between components, in order to achieve the system's functional and non-functional requirements within the constraints of technology, skills and budget.

An Operational Model can be described at two basic levels:

- Logical: This level describes the characteristics and capabilities of the operational aspect of the system architecture in a technology independent and product neutral manner.
- Physical: This level describes characteristics and capabilities of the operational aspect of the system architecture in a technology and product dependent manner.

2 Information System Architecture

2.1 Architecture overview

The architecture overview illustrates the essential nature of the proposed solution (conceptually), conveying the governing ideas and including the major building blocks.

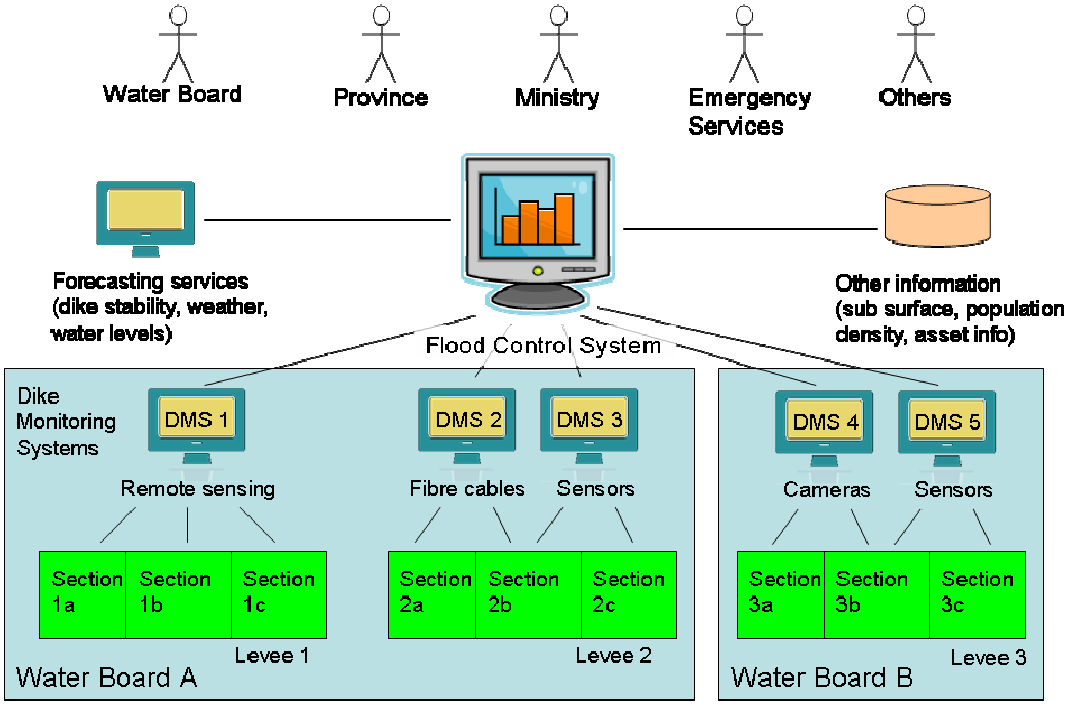

Figure 2-1 Architecture Overview of Flood Control System

Levee maintainers (in the Netherlands a responsibility of the water boards) will increasingly install more systems to automatically monitor the stability of levees, so called Dike Monitoring Systems. As they do so, they will use different technologies from different vendors to use the best technology for the right levee and gain the most optimal insight in the stability and inner working of the levee. This also leads to different informational views on the levees which can not be compared or combined into one consistent for the organisation. Or even over organisational borders, for example to compare performance of levees. This project focuses on the design of a Flood Control System that will bring the information from those Dike Monitoring Systems together. The system can be used by the water board to gain an consistent overview of the levees under their responsibility. They can use this to improve the maintenance process of levees by using sensor readings and stability forecasts as additional information next to the current inspection reports and design level information. This will result in the reduction of maintenance cost and improved quality of the levee and certainty of its stability. It can also be used to fulfil the legal duties of levee maintainers by reporting the 5 yearly inspection status (to the province) more accurate and complete. Actually, other organisation might be able to use the information. The province gains real-time insight of the status of levee system which can be used to set direction for water boards, create policy or spatial planning. Water boards among themselves can use the information for benchmarking and research on best practices for levee maintenance. In case of a levee breach the emergency services will gain an immediate ‘common operational picture’ of the situation.

Consistent and information access is critical for emergency services to be able to make the right decisions based on the latest information.

In short the Flood Control System will integrate the various sources of information and create a common operational picture (consistent information about the situation) for use in two situation:

1. Operational (regular) business processes (like maintenance and 5-yearly reporting of levee status)

2. Crisis situations (for example at a dike breach)

2.2 System Context Diagram

The System Context identifies all information exchanges between the envisioned system and its environment and sources or destinations of data used by the system. Identify all external entities that need to interface to the system. Determine the inputs and outputs need to support each of these entities. These information exchanges and data sources represent constraints that the system under development is bound by. The System Context Diagram identifies required interfaces with other systems (i.e. dike monitoring systems, meteorological sources, etc.) and provides context for describing internal aspects of the system.

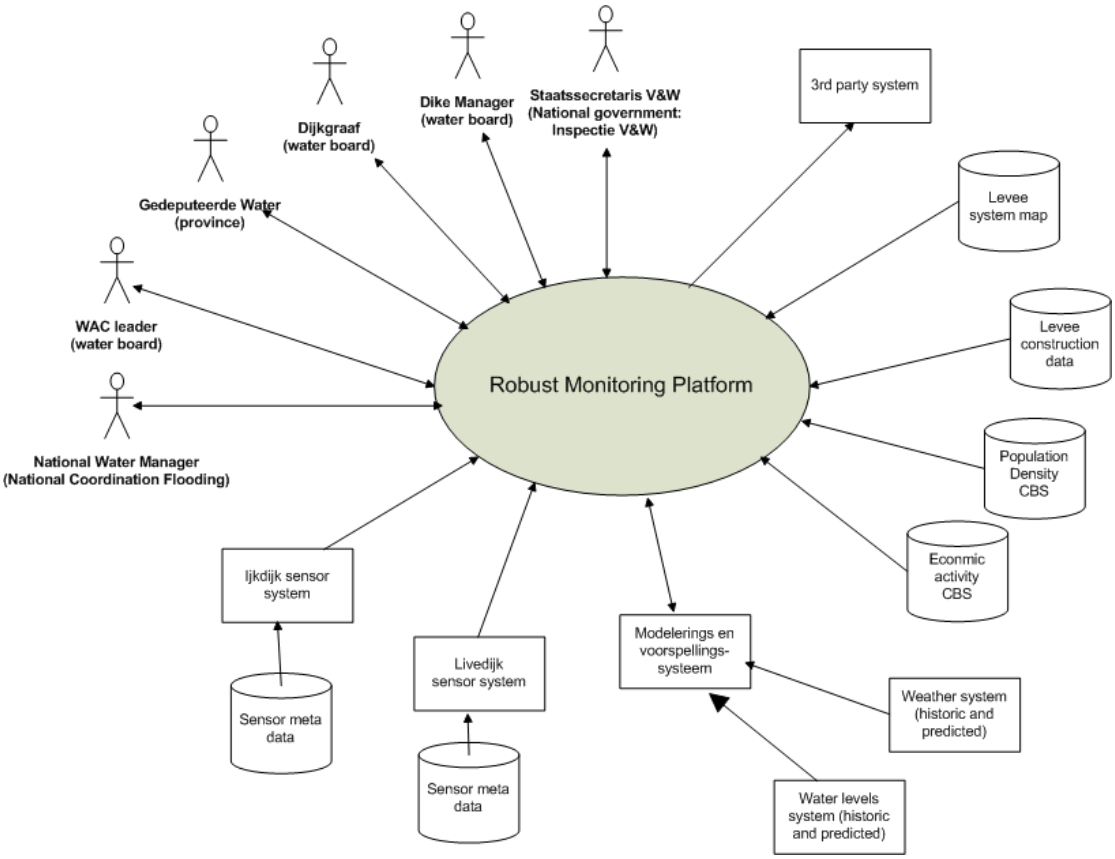

Figure 2-2 System Context Diagram

2.3 Guiding principles

The following principles will guide the design process of the solution. These principles have been taken from the Flood Control 2015 Reference Architecture and are aligned with guiding principles from governmental organisation like Rijkswaterstaat and Het Waterschapshuis.

- The architecture will comply with the NORA principles (Nederlandse Overheids Referentie Architectuur
  As the envisioned solution will be used by government agencies it should follow the governmental guideline for interoperability between governmental organisation specified in the NORA.
  http://www.e-overheid.nl/atlas/referentiearchitectuur/nora/nora.html

- The architecture will use and provide open standards.
  As the envisioned solution will be used by government agencies it should follow the governmental guideline to use open standards according to the plan "Nederland Open in Verbinding" NOiV.
  http://noiv.nl/

- The architecture will be based on Aquo standards for data models
  The Dutch government has installed the IDSW (Informatiedesk Standaarden Water). This will make it easier to share information in the future with other governmental agencies.
  http://www.idsw.nl/

2.4 Architectural Decisions

A list of key decisions for the solution design

1. Solution will be based on Service Oriented Architecture principles
   This provides the flexibility to easily integrate applications and systems based on open and well defined standards.

2. Use of proven technology
   The solution will use proven technologies available on the market or as open source. This speeds up the implementation process and improves the market readiness of the solution (a Flood Control 2015 goal)

2.5 Non-functional requirements

This paragraphs lists the non-functional requirements that the solution is constrained to.

Code	***Requirement***	***Description***
RF 5	Connect different and multiple DMSes	The solution should be designed to be able to integrate Dike Monitoring System from various vendors
RF 6	Use of external data sources	The solution must be able to use external data sources other then sensor information. e.g. weather data
RF 7	Create new information	Create new information based on external data sources and knowledge created by the FCS (e.g. dike stability)
RF 9.1	Support for prediction models	In order to calculate expected situations
RF 10	Report status of the	A FCS must be able to give a value for dike stability

	dike	
RF 12	Share information	A FCS must be able to share information about the dike
RF 13.1	Store raw sensor values	If the FCS stores sensor data in its system (for caching), it must store the at least the raw data in order to proof later what the basis of the information was that higher level applications have based their conclusions.
RF 17	Data sample rate	The FCS does not necessarily follow the data sample of the underlying DMS. An FCS might have a more aggregated view on this.
RF 19	Logging mechanism	All data that the FCS receives and calculates is logged
RF 20	Find alternative information sources with fit the information need	In case of failure of a sensor system, try to find an alternative
RF 21	Enable/disable sensors	If for example a sensor is broken, it should not be included in calculations before it is fixed.
RF 22	Information is shared by a 'virtual sensor interface'	All information that is created in the FCS will be exposed as new (enriched) data to other applications for reuse. The Virtual Sensor interface is equivalent to a Web Service following SOA principles.
RF23	Design for High availability	Design the system for high availability. For cost reasons (because this is a demonstrator) it doesn't have to be implemented like that. Cold standby is good enough. The repair time in case of failure should be not more than 1 working day.
RF24	Restrict permission to data	The data from the Livedijk should not be available outside the scope of project team. There user access control on the system required.

Table 2-1: Non-functional requirements for a Flood Control System

2.6 Component Model

The component model describes the structure of a system in terms of its software components with their responsibilities, interfaces and relationships to deliver the required functionality. The component model is the main artefact documenting the functional view of the architecture and serves as an abstraction for the design.

Component models are documented at 3 levels:

- Conceptual level, focussing on the grouping of components. It identifies the high level type of components to be able to group them logically together.
- At the logical level, focusing on specifying the components' responsibilities and characteristics required to deliver the requirements. These specifications are typically technology and product neutral.
  For example a Message Bus or a modelling framework.
- At the physical level, focusing on how the components will be implemented to meet the previously established specifications. Specified components can be transformed into physical components via custom development, the purchase of products or the reuse of assets.
  For example IBM WebSphere Message Broker or Deltares FEWS

2.6.1 Conceptual Component Model

The purpose of the conceptual component model is to identify the high level component grouping. This helps to understand what type of components are involved in the solution and how they relate to each other.

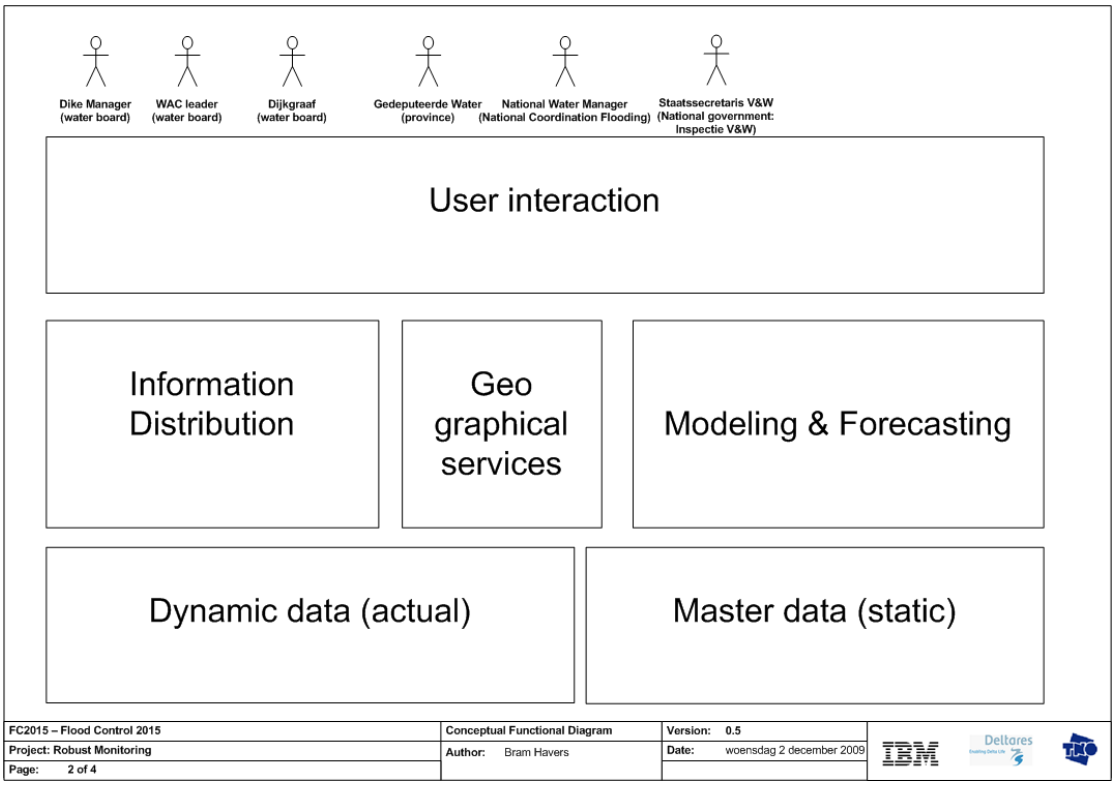

Figure 2-3 Conceptual Component Model (identify major functional areas)

Description of grouping

Grouping	Description
Actors	All actors as specified by the Use Cases
User Interactions	The components that are used by the actors to interact with the system.
Information Distribution	The components that handle the sharing and integration of information and services.
Geo graphical services	The infrastructure components that enable geo graphical services
Modelling & Forecasting	The components that are responsible for creating forecasts.
Dynamic data	All data that is dynamic from nature, for example sensor measurements. The amount of data can vary depending on many parameters which makes it less predictable than master data. The data structures are often much simpler (sensor data with name, value and time stamp).
Master data	All data that is relatively static by nature. The predictability of the amount of data is high. Master data often has a more complicated data structure (GIS layers representing assets and its parameters).

2.6.2 Logical Component Model

The logical component model focuses on specifying the components' responsibilities and characteristics required to deliver the requirements. These specifications are typically technology and product neutral.

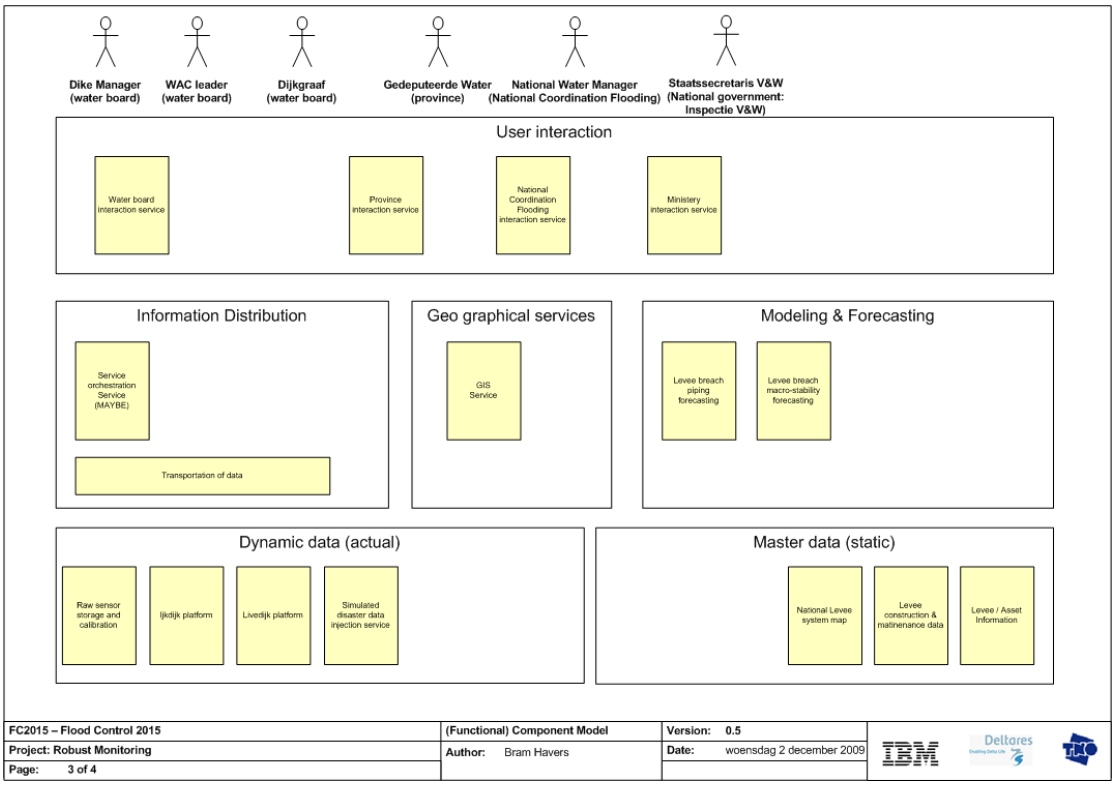

Figure 2-4 Logical Component Model (functional view of the system)

In table below you will find a description of the components in term of their responsibilities and characteristics.

	Component	Responsibility and characteristics
User Interaction		
	Water board interaction service	The user interface that users at a water board use to interact with the system
	Province interaction service	The user interface that users at a province use to interact with the system
	National Coordination Flooding interaction service	The user interface that people at a National Coordination Flooding department (LCO) use to interact with the system
	Ministry interaction service	The user interface that people at the ministry use to interact with the system
Information Distribution		
	Service Orchestration Service	Handles the execution of processes over various (web) services.
	Transportation of data	Facilitates the distributes of data, the decoupling of systems and services via a bus structure and provides functionality to transform data on the fly (mediation)
Geo graphical services		
	GIS service	The Geo Information Service that provides access to map data: satellite imagery and layers of data like the levee system.
Modelling & Forecasting		
	Levee breach piping forecasting	A service that predicts when the levee will breach. It will present a forecast of the pipe length in the IJkdijk

	Levee breach macro-stability forecasting	A service that predicts when the levee will breach. It will present a forecast of the macro stability in the Livedijk
Dynamic Data (actual)		
	Raw sensor storage and calibration	The component in which raw sensor values (measurements) are stored before and after calibration. Needed for audit purposes and to reinitialise sensor data when a new calibration needs to be done (for example because a sensor had degraded in quality without being compensated for)
	IJkdijk platform	The sensor platform in use for the IJkdijk experiments
	Livedijk platform	The sensor platform in use for Livedijk operations
	Simulated disaster data injection services	Synthesized (fake) sensor data that simulates a flooding event. Needed to test the crisis situation use case.
Master data (static)		
	National levee system map	Geographical information about the Dutch levee system. National data and specifically around the IJkdijk and Livedijk.
	Levee construction & maintenance data	Data about the construction of the levees (geometric information, e.g. 20% talud, etc.) and maintenance information (last inspection and changes).
	Levee asset information	Information about the levee itself (grass, clay, road on top, etc.)

Table 2

2.6.3 Physical / Software Component Model

The Physical (Software) Component Model focusing on how the components will be implemented to meet the previously established specifications. Specified components can be transformed into physical components via custom development, the purchase of products or the reuse of assets

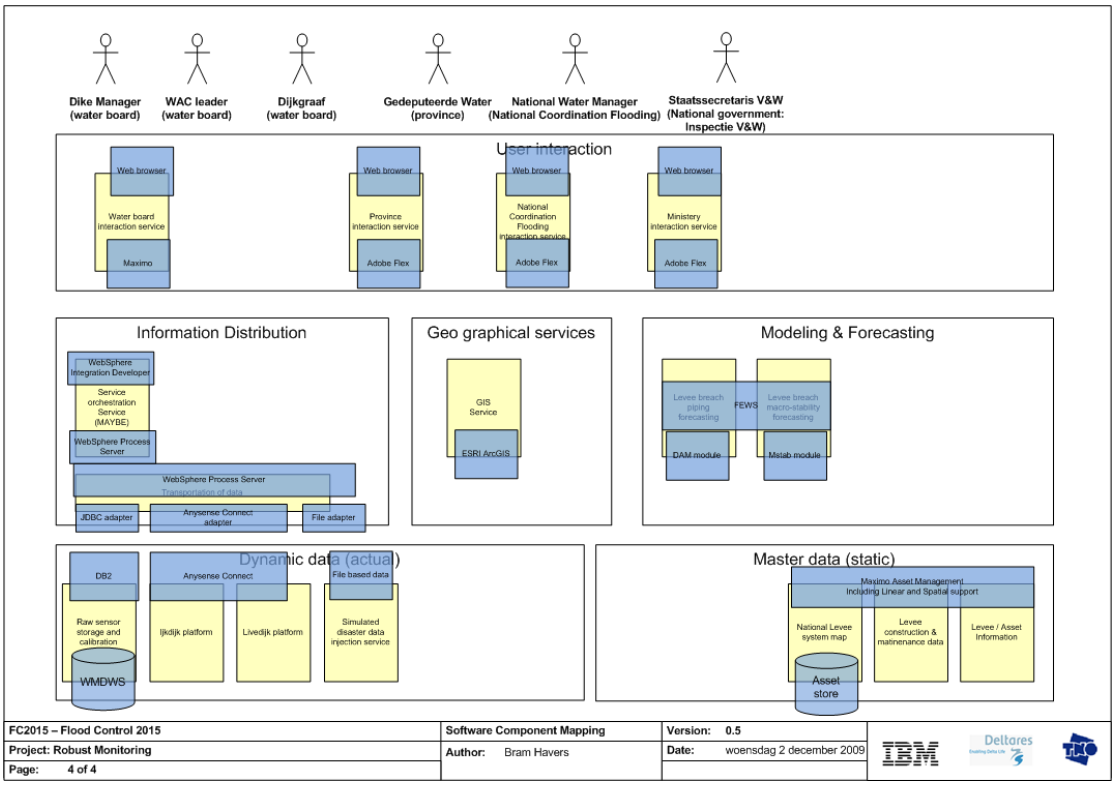

Figure 2-5 Physical Component Model (identify software component to fulfil functionality)

In the table below you will find a description of the software components that will be used in the system.

	Software component	Description
User Interaction		
	Web browser	The functionality of the system will be available via the internet in a general web browser. There is no need for client software
	Maximo	See Chapter “Available assets to be used” for a description.
	Adobe Flex	See Chapter “Available assets to be used” for a description.
Information Distribution		
	WebSphere Process Server	See Chapter “Available assets to be used” for a description.
	WebSphere Integration Developer	
	JDBC Adapter	Standard adapter in WebSphere Process Server to connect JDBC compliant databases
	Anysense Connect Adapter	Custom build adapter based on HTTP adapter from WebSphere Process Server to connect TNO Anysense systems.
	File Adapter	Standard adapter in WebSphere Process to read files (that are send over FTP) and make the information available on the Enterprise Service Bus and in the database.
Geo Graphical Services		
	ESRI ArcGIS	Leading GIS mapping application server from ESRI.

		This will integrate with Maximo Spatial enabling the user to see the position of an asset (levee) on a map.
Modeling & Forecasting		
	FEWS	See Chapter “Available assets to be used” for a description.
	DAM module	See Chapter “Available assets to be used” for a description.
	Mstab module	See Chapter “Available assets to be used” for a description.
Dynamic Data		
	DB2	The database to store all sensor, forecast and configuration data of the applications in.
	WMDWS	Tablespace within DB2 that holds all raw sensor data and forecasts
	Anysense Connect	The TNO platform that extracts the information from the sensors of the IJkdijk and the Livedijk.
	File based data	A file that will hold data for a simulated event. This can be ran through the system as if it where actual data.
Master Data		
	IBM Maximo	See Chapter “Available assets to be used” for a description.
	IBM Maximo Spatial	See Chapter “Available assets to be used” for a description.
	IBM Maximo Linear	See Chapter “Available assets to be used” for a description.
	Asset Store	Tablespace within DB2 that holds the information about the assets (levees and sensors).

2.7 Information model

The information analysis is done in the Visualisation document. The Entity-Relationship Diagrams in that document describes the information that needs to be handled in this solution.

2.8 Interfaces

Now that the software components are identified and the information that needs to be processed is clear (see Information Analysis in visualisation document), the interaction between the systems will be defined. Systems will not be coupled directly to each other but uncoupled by an Enterprise Service Bus as indicated in the component model. This brings the advantage that information submitted to the bus can be reused by all other applications. It also makes the replacement of one system by another easier. As long as the same interface definition is used the other systems will not notice that the system has been changed.

The following interfaces have been identified.

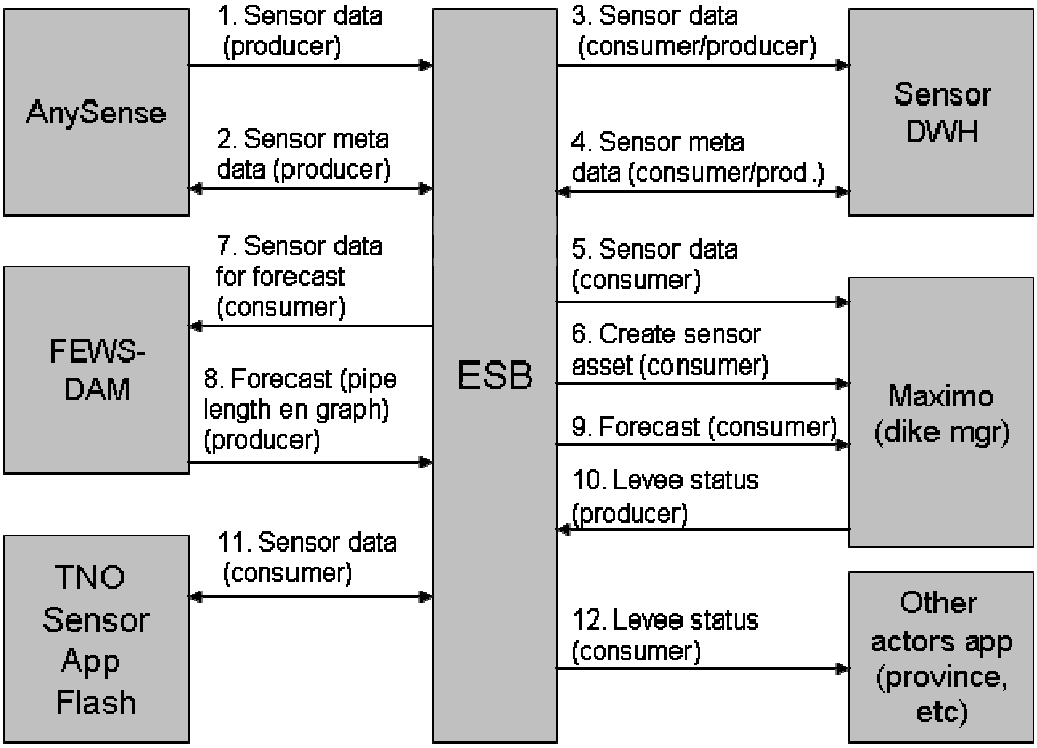

Figure 2-6 Overview of interfaces

The following sub paragraphs will describe the details of these interfaces. It will describe:

- Name: the name that we will use to uniquely identify this interface
- Description: the type information that is exchanged
- Producer/consumer: indicates if this application (not the bus) produces information or consumes (uses) information.
- Request/Response or Push/Pull: Is this a request response interface (client requests information and receives an answer. Or a push or pull interface in which the server sends data (or the client retrieves data) from to a predefined location
- Synchronous: Is this a synchronous interface (an response is directly expected, client wil wait for response), or an a-synchronous interface in which actions (requests, update actions) can be queued and the response can be returned separated of the request.
- Reliable: is this a reliable interface so that it is assured that information will not be lost.
- Secure: Is this a secure interface
- Transport protocol: the protocol to be used for transporting the message data. Example of such protocols are: HTTP, SMTP, FTP, MQ, MQtt, JMS
- Messaging protocol: the encoding protocol for the message. Could be SOAP, XML-RPC, CSV file.
- Amount of data: an indication of amount of data. Needed to define sizing estimates for the hardware and software configuration.
- Interval: the frequency of data interchange between the systems. Also needed for sizing.

2.8.1 Sensor Data interface

Characteristic	Description
Name	1. Sensor Data
Description	The sensor data measurements from the Dike Monitoring System
Producer/Consumer	Producer
Request/Response or Push/Pull	Push interface to a predefined Web Service URI
Synchronous	Synchronous
Reliable	No, it's not of the highest importance that we have assured delivery of every sensor measurement.
Secure	No, for demonstration purposes chosen not to complicate the setup by configuring security.
Transport protocol	HTTP
Messaging protocol	SOAP
Amount of data	1 KB
Interval	10 sec.

The main element in the SOAP message is "DeliverMeasurementsRequest", which can contain multiple elements of type "measurement". A "measurement" is one measurement delivered by a sensor. Two timestamps are coupled to a measurement:

- anysenseTimestamp which represents the data of registration in the AnySense system.
- ownerTimestamp which has been send by the sensor to AnySense.

The ownerId can contain an ID that the owners identify the sensor with.
A measurement always originates from one sensor. The sensor has an ID to identify it and a number of sensor part that always deliver measuments (values). These measurements are of type double.

An outline of the message structure:
DeliverMeasurementsRequest
----measurement *
--------anysenseTimestamp
--------ownerId
--------ownerTimestamp
--------sensor
----------id
----------sensorPart *
----------id
----------value
----------doubleValue

The schema of this interface and an example XML message is provided below.

2.8.1.1 Schema XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://anysense.tno.nl/measurements"
        xmlns:tns="http://anysense.tno.nl/measurements" elementFormDefault="qualified">

        <complexType name="MeasurementType">
                <sequence>
```

```
                <element name="anysenseTimestamp" type="dateTime"></element>
                <element name="ownerId" type="string"></element>
                <element name="ownerTimestamp" type="dateTime"></element>
                <element name="sensor" type="tns:SensorType"></element>
            </sequence>
        </complexType>

        <simpleType name="IdType">
            <restriction base="string"></restriction>
        </simpleType>

        <complexType name="SensorPartType">
            <sequence>
                <element name="id" type="tns:IdType"></element>
                <element name="value" type="tns:ValueType"></element>
            </sequence>
        </complexType>


        <complexType name="ValueType">
            <sequence>
                <choice>
                    <element name="doubleValue" type="double"></element>
                    <element name="longValue" type="long"></element>
                    <element name="stringValue" type="string"></element>
                </choice>
            </sequence>
        </complexType>


        <complexType name="SensorType">
            <sequence>
                <element name="id" type="tns:IdType"></element>
                <element name="sensorPart" type="tns:SensorPartType"
                    maxOccurs="unbounded" minOccurs="1">
                </element>
            </sequence>
        </complexType>

        <complexType name="MeasurementGroupType">
            <sequence>
                <element name="measurement" type="tns:MeasurementType"
                    maxOccurs="unbounded" minOccurs="0"></element>
            </sequence>
        </complexType>

        <element name="measurements" type="tns:MeasurementGroupType"></element>

</schema>
```

2.8.1.2 Example XML

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:measurements xmlns:tns="http://anysense.tno.nl/measurements"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://anysense.tno.nl/measurements measurements.xsd ">
  <tns:measurement>
    <tns:anysenseTimestamp>2001-12-31T12:00:00</tns:anysenseTimestamp>
```

```
  <tns:ownerId>tns:ownerId</tns:ownerId>
  <tns:ownerTimestamp>2001-12-31T12:00:00</tns:ownerTimestamp>
  <tns:sensor>
    <tns:id>tns:id</tns:id>
    <tns:sensorPart>
      <tns:id>tns:id</tns:id>
      <tns:value>
        <tns:doubleValue>0.0</tns:doubleValue>
      </tns:value>
    </tns:sensorPart>
  </tns:sensor>
 </tns:measurement>
</tns:measurements>
```

2.8.2 Sensor Meta Data interface

Characteristic	Description
Name	2. Sensor Meta Data
Description	Information about the sensor in the Dike Monitoring System
Producer/Consumer	Producer
Request/Response or Push/Pull	Request / Response interface
Synchronous	Synchronous
Reliable	No, in case we will not get a response within 1 minute a new request is fired.
Secure	No, for demonstration purposes chosen not to complicate the setup by configuring security.
Transport protocol	HTTP
Messaging protocol	SOAP
Amount of data	1 KB
Interval	On user request.

To understand from which type of sensor the measurements are coming you need sensor meta data that describes the sensor, like description of the sensors, its parts and the unit of measurement.

Outline of structure of message:
----sensor *
--------id
--------name
--------sensorPart *
----------id
----------name
----------quantity
----------name
----------description
----------unit

The schema of this interface and an example XML message is provided below.

2.8.2.1 Schema XSD

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://anysense.tno.nl/sensors"
        xmlns:tns="http://anysense.tno.nl/sensors" elementFormDefault="qualified">

        <simpleType name="IdType">
                <restriction base="string"></restriction>
        </simpleType>



        <complexType name="SensorType">
                <sequence>
                        <element name="id" type="tns:IdType"></element>
                        <element name="name" type="string"></element>
                        <element name="sensorpart" type="tns:SensorPartType"
                                maxOccurs="unbounded" minOccurs="0">
                        </element>
                </sequence>
        </complexType>


        <complexType name="QuantityType">
                <sequence>
                        <element name="name" type="string"></element>
                        <element name="description" type="string"></element>
                        <element name="unit" type="tns:UnitType"></element>
                </sequence>
        </complexType>


        <simpleType name="UnitType">
                <restriction base="string"></restriction>
        </simpleType>

        <complexType name="SensorPartType">
                <sequence>
                        <element name="id" type="tns:IdType"></element>
                        <element name="name" type="string"></element>
                        <element name="quantity" type="tns:QuantityType"></element>
                </sequence>
        </complexType>


        <complexType name="SensorGroupType">
                <sequence>
                        <element name="sensor" type="tns:SensorType" maxOccurs="unbounded"
                                minOccurs="1"></element>
                </sequence>
        </complexType>

        <element name="sensors" type="tns:SensorGroupType"></element>
</schema>
```

2.8.2.2 Example

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:sensors xmlns:tns="http://anysense.tno.nl/sensors"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://anysense.tno.nl/sensors sensors.xsd ">
  <tns:sensor>
```

```
    <tns:id>tns:id</tns:id>
    <tns:name>tns:name</tns:name>
    <tns:sensorpart>
      <tns:id>tns:id</tns:id>
      <tns:name>tns:name</tns:name>
      <tns:quantity>
        <tns:name>tns:name</tns:name>
        <tns:description>tns:description</tns:description>
        <tns:unit>tns:unit</tns:unit>
      </tns:quantity>
    </tns:sensorpart>
  </tns:sensor>
</tns:sensors>
```

2.8.3 Sensor Data interface to Sensor Datawarehouse

Characteristic	Description
Name	3. Sensor Data
Description	Updating the raw sensor database with the latest sensor reading
Producer/Consumer	Consumer
Request/Response or Push/Pull	Push. Interface is triggered by incoming data.
Synchronous	Synchronous, update done immediately.
Reliable	No, updates will be lost if database goes down. But the primary source of sensor (Dike Monitoring System) can be requested to resubmit data.
Secure	No, for demonstration purposes chosen not to complicate the setup by configuring security.
Transport protocol	Database update (JDBC call) from Mediation module in ESB
Messaging protocol	SQL
Amount of data	1 KB
Interval	10 sec.

2.8.4 Sensor Meta Data interface to Sensor Datawarehouse

Characteristic	Description
Name	3. Sensor Data
Description	Updating the raw sensor database with the latest sensor reading
Producer/Consumer	Consumer
Request/Response or Push/Pull	Push. Interface is triggered by incoming data.
Synchronous	Synchronous, update done immediately.
Reliable	No, updates will be lost if database goes down. But this process is manually triggered. Can just be triggered again to receive new sensors from the primary source (Dike Monitoring System)
Secure	No, for demonstration purposes chosen not to complicate the setup by configuring security.
Transport protocol	Database update (JDBC call) from Mediation module in ESB
Messaging protocol	SQL
Amount of data	1 KB

Interval	None, Bulk load process, manually triggered.

2.8.5 Sensor Data interface to Maximo

Characteristic	Description
Name	5. Sensor Data
Description	Updates Maximo with the latest sensor reading
Producer/Consumer	Consumer
Request/Response or Push/Pull	Push. Interface is triggered by incoming data.
Synchronous	Synchronous
Reliable	No, updates will be lost if Maximo is down. But the primary source of sensor (Dike Monitoring System) can be requested to resubmit data.
Secure	No, for demonstration purposes chosen not to complicate the setup by configuring security.
Transport protocol	HTTP
Messaging protocol	SOAP
Amount of data	1 KB
Interval	10 sec.

2.8.5.1 Example XML

Below a xml message is described for updating a meter reading (in Maximo a sensor is called a meter)

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:max="http://www.ibm.com/maximo" baseLanguage="EN">
  <soapenv:Header/>
  <soapenv:Body>
    <max:UpdateMXASSET>
      <max:MXASSETSet>
        <!--Zero or more repetitions:-->
        <max:ASSET action="Change">
          <max:ASSETNUM>sen44</max:ASSETNUM>
              <max:SITEID>NOORD</max:SITEID>
          <max:ASSETMETER>
                    <max:LINEARASSETMETERID>0</max:LINEARASSETMETERID>
            <max:LASTREADING changed="1">1.0</max:LASTREADING>
            <max:METERNAME>TS</max:METERNAME>
          </max:ASSETMETER>
        </max:ASSET>
      </max:MXASSETSet>
    </max:UpdateMXASSET>
  </soapenv:Body>
</soapenv:Envelope>
```

2.8.6 Create Sensor Asset interface to Maximo

Characteristic	Description
Name	6. Create Sensor Asset
Description	Create a sensor asset within Maximo and associate this asset

	with a levee asset.
Producer/Consumer	Consumer
Request/Response or Push/Pull	Push. Interface is triggered by incoming data.
Synchronous	Synchronous, update done immediately.
Reliable	No, updates will be lost if database goes down. But this process is manually triggered. Can just be triggered again to receive new sensors from the primary source (Dike Monitoring System)
Secure	No, for demonstration purposes chosen not to complicate the setup by configuring security.
Transport protocol	HTTP
Messaging protocol	SOAP
Amount of data	1 KB
Interval	None, Bulk load process, manually triggered.

2.8.6.1 Example XML

The XML-message below describes the creation of a new meter (sensor)

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:max="http://www.ibm.com/maximo">
  <soapenv:Header/>
  <soapenv:Body>
    <max:UpdateMXASSET>
      <max:MXASSETSet>
        <max:ASSET action="Change">
          <max:ASSETNUM>sen45</max:ASSETNUM>
          <max:LOCATION>IJKDIJK01</max:LOCATION>
          <max:ORGID>LCO</max:ORGID>
          <max:SITEID>NOORD</max:SITEID>
          <max:ASSETMETER action="Add">
            <max:ACTIVE>1</max:ACTIVE>
                    <max:ASSETMETERID>84</max:ASSETMETERID>
                    <max:METERNAME>STABILITY</max:METERNAME>
            <max:LINEARASSETMETERID>0</max:LINEARASSETMETERID>
          </max:ASSETMETER>
        </max:ASSET>
      </max:MXASSETSet>
    </max:CreateMXASSET>
  </soapenv:Body>
</soapenv:Envelope>
```

2.8.7 Sensor Data for Forecast to FEWS DAM

Characteristic	Description
Name	7. Sensor Data for Forecast
Description	Send sensor data to FEWS forecasting module to receive a forecast on that data.\ Note: this interface is not implemented as the FEWS DAM system already had the actual sensor data. We could use the response straight away.

Producer/Consumer	Consumer
Request/Response or Push/Pull	Push.
Synchronous	Synchronous, request time triggered by ESB
Reliable	No
Secure	No, for demonstration purposes chosen not to complicate the setup by configuring security.
Transport protocol	FTP
Messaging protocol	CSV format
Amount of data	Will grow with sensor data, approx. 50 MB a day when all sensor data is used.
Interval	Every hour.

2.8.8 Forecast to ESB

Characteristic	Description
Name	8. Forecast
Description	Receive levee stability forecast
Producer/Consumer	Producer
Request/Response or Push/Pull	Push.
Synchronous	Synchronous
Reliable	No
Secure	No, for demonstration purposes chosen not to complicate the setup by configuring security.
Transport protocol	FTP
Messaging protocol	CSV format
Amount of data	1 KB
Interval	Every hour.

2.8.8.1 Example CSV file

The message contains in order of sequence:
Datetime stamp, stability raai 1, stability raai 2, stability raai 3, stability raai 4.

Note a Raai is string of sensors over the with of a levee. Four of these strings are placed next to each other in a matrix.

2009-11-24,14:10:00,1.86,1.692,1.692,0.226
2009-11-24,14:15:00,1.86,1.692,1.692,0.223
2009-11-24,14:20:00,1.86,1.692,1.692,0.243
2009-11-24,14:25:00,1.806,1.692,1.692,0.224
2009-11-24,14:30:00,1.806,1.692,1.692,0.224
2009-11-24,14:35:00,1.86,1.692,1.692,0.237
2009-11-24,14:40:00,1.806,1.692,1.692,0.23
2009-11-24,14:45:00,1.86,1.692,1.692,0.228
2009-11-24,14:50:00,1.806,1.692,1.692,0.293
2009-11-24,14:55:00,1.86,1.692,1.692,0.239
2009-11-24,15:00:00,1.806,1.692,1.692,0.222

Value -999 means no stability could be calculated for that particular moment.

Values below 1 should be interpreted as unstable levees (the last value / raai 4 in error due to a failing sensor).

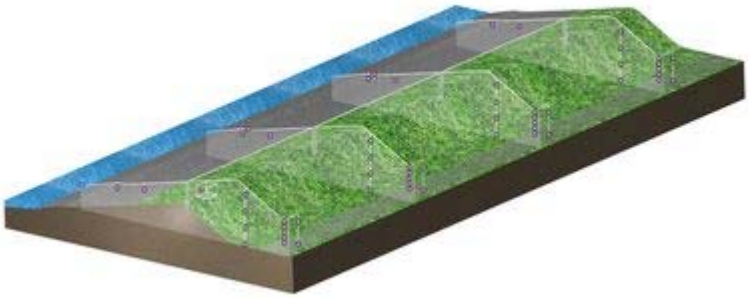

Figure 2-7: 3D image of the livedike location with the 4 'raaien' for which the dike stability is calculated.

2.8.9 Forecast to Maximo

Characteristic	Description
Name	9. Forecast
Description	Receive levee stability forecast
Producer/Consumer	Consumer
Request/Response or Push/Pull	Push.
Synchronous	Synchronous
Reliable	No
Secure	No, for demonstration purposes chosen not to complicate the setup by configuring security.
Transport protocol	HTTP
Messaging protocol	SOAP
Amount of data	1 KB
Interval	Every hour.

2.8.10 Levee Status to bus

Characteristic	Description
Name	10. Levee status
Description	Provides the status of each levees as assessed by the dike manager in terms of bad, moderate, good.
Producer/Consumer	Producer
Request/Response or Push/Pull	Request / Response
Synchronous	Synchronous
Reliable	No
Secure	No, for demonstration purposes chosen not to complicate the setup by configuring security.
Transport protocol	HTTP
Messaging protocol	SOAP
Amount of data	1 KB

Interval	On request.

2.8.11 Levee Status to TNO Sensor Data Visualisation Application

Characteristic	Description
Name	11. Sensor data
Description	Provides sensor data to the sensor data visualisation application Note: not implemented yet, TNO Anysense using data now directly from the Anysense system.
Producer/Consumer	Consumer
Request/Response or Push/Pull	Request / Response
Synchronous	Synchronous
Reliable	No
Secure	No, for demonstration purposes chosen not to complicate the setup by configuring security.
Transport protocol	HTTP
Messaging protocol	GET parameters, custom HTTP payload
Amount of data	1 KB
Interval	On request.

Because the visualisation application is running in a browser (it is an Flash application) the communication uses HTTP GET calls. There are three calls:

- GetDikes to retrieve information on which levee should be visualised and which sensors belong to those levees. Parameters are:
    - boundary : een geo area in which the levee should be. Not implemented at this moment.
- GetAvailableDataBounds to indicate the beginning and end in time for which sensor data is available. Parameters are:
    - dikeid = the id of the dike from which the time boundaries are requested
- GetMeasurements to retrieve sensor data for specific sensors. Parameters are:
    - requestid : An id for the application to see which request this is. Provided with the result.
    - sensorid : A list of sensorids
    - life : wether the data is live or historical. Boolean (live=1 is live)
    - endtime : the last point in time of which a measurement is requested. endtime may be in the unix time format (long) or in yyyyMMdd'T'HHmmss
    - timelinesize : the length of the measurements in ms.
    - resolution : the number of measuringpoint that should be returned.

Schemas of response and example XML are available upon request.

2.8.12 Levee Status to other applications

Characteristic	Description
Name	12. Levee status

Description	Provides levee status to other applications: Note: not implemented.
Producer/Consumer	Consumer
Request/Response or Push/Pull	Request / Response
Synchronous	Synchronous
Reliable	No
Secure	No, for demonstration purposes chosen not to complicate the setup by configuring security.
Transport protocol	HTTP
Messaging protocol	SOAP
Amount of data	1 KB
Interval	On request.

2.8.12.1 Example XML

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:max="http://www.ibm.com/maximo">
  <soap:Header/>
  <soap:Body>
    <max:QueryMXASSETMETER rsStart="0">
      <max:MXASSETMETERQuery operandMode="AND">
        <max:ASSETMETER>
          <max:ASSETNUM operator="=">LIVEDIJK01</max:ASSETNUM>
          <max:METERNAME operator="=">LCON</max:METERNAME>
        </max:ASSETMETER>
      </max:MXASSETMETERQuery>
    </max:QueryMXASSETMETER>
  </soap:Body>
</soap:Envelope>
```

2.9 Processing flows

The sequence of interaction of systems is described and implemented in process flows. These flows define the sequence in which interfaces are called in order to execute a certain function.
These flow will be run in the ESB (implemented by WebSphere Process Server) and developed using the WebSphere Integration Development tool.
This table describes the process flows

Flow#	Interfaces involved	Name
101	1 -> 3	Store sensor data in Sensor DWH
102	2 -> 4	Stora sensor meta data in Sensor DWH
103	5	Store (selected) data in Maximo on sensor asset
104	6	(optional) Automatically create sensor asset in Maximo, based on new sensor meta data.
105	7	Deliver sensor data to FEWS as input for forecast

106	8 -> 9	Poll for new forecast, retrieve and store forecast in Maximo on levee asset
107	10	Publish sensor data
108	11	Retrieve sensor data

All flows are relatively straight forward and don't need extra explanation. Only the flows to request and receive a forecast are a bit more complicated and will be explained using an interaction diagram.

2.9.1 Forecasting Process Flow Interaction Diagram

Flow 105 and 106 are somewhat more complex than the other flows and are therefore detail with the interaction diagram below.

Note that flow 105 has not been implemented, see the comment at interface "7. Sensor Data for Forecast".

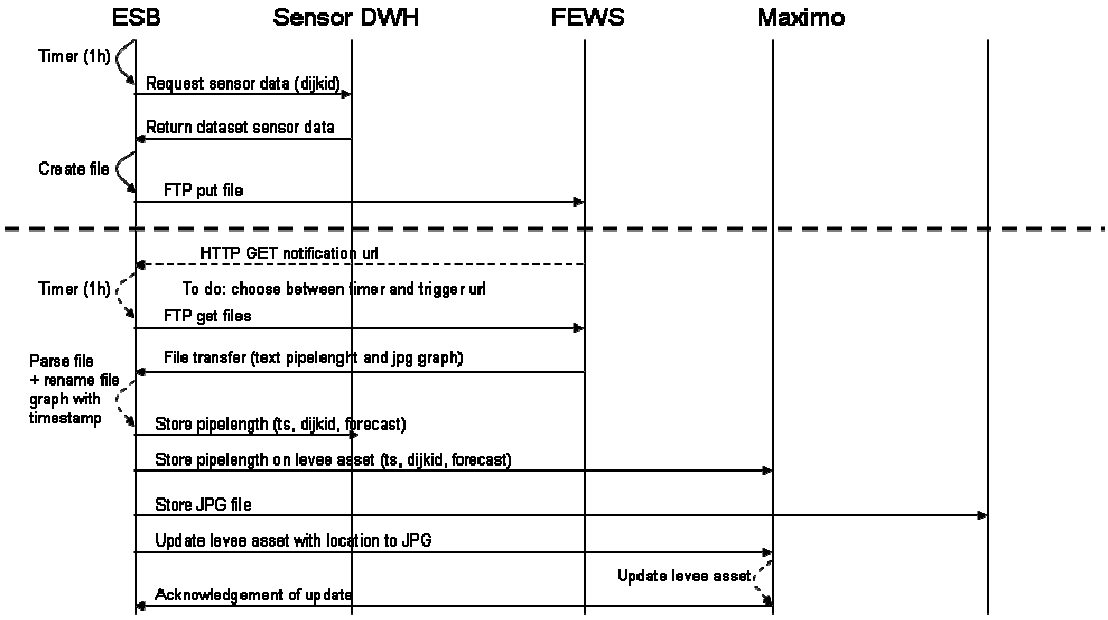

Figure 2-8 Interaction Diagram for forecast requesting

3 Technology Architecture

3.1 Operational Model

This Operational Model describes the operational distribution of a system's components onto nodes, the placement of nodes and users across locations, the connections between nodes necessary to support the required interactions between components, in order to achieve the system's functional and non-functional requirements within the constraints of technology, skills and budget.

An Operational Model can be described at two basic levels:

- Logical: This level describes the characteristics and capabilities of the operational aspect of the system architecture in a technology independent and product neutral manner.
- Physical: This level describes characteristics and capabilities of the operational aspect of the system architecture in a technology and product dependent manner.

3.1.1 Logical Operational Model

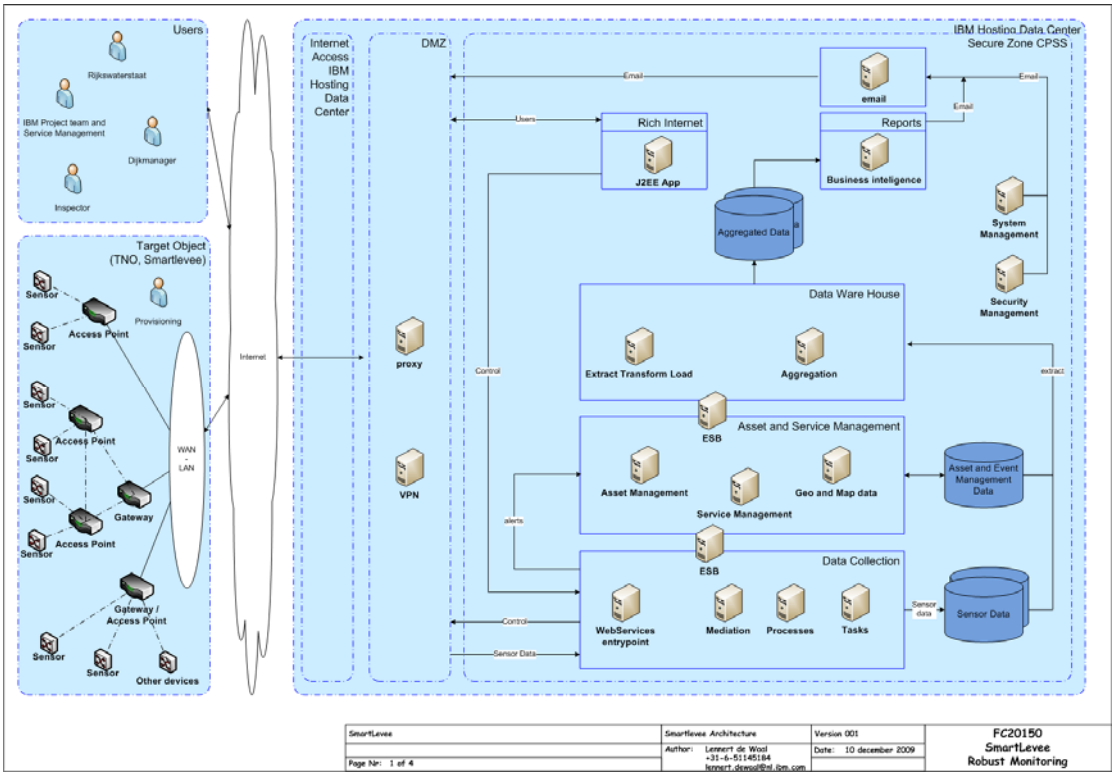

Figure 3-1 Logical Operational Model

Shown is the logical seperation of components.

3.1.2 Physical Operational Model

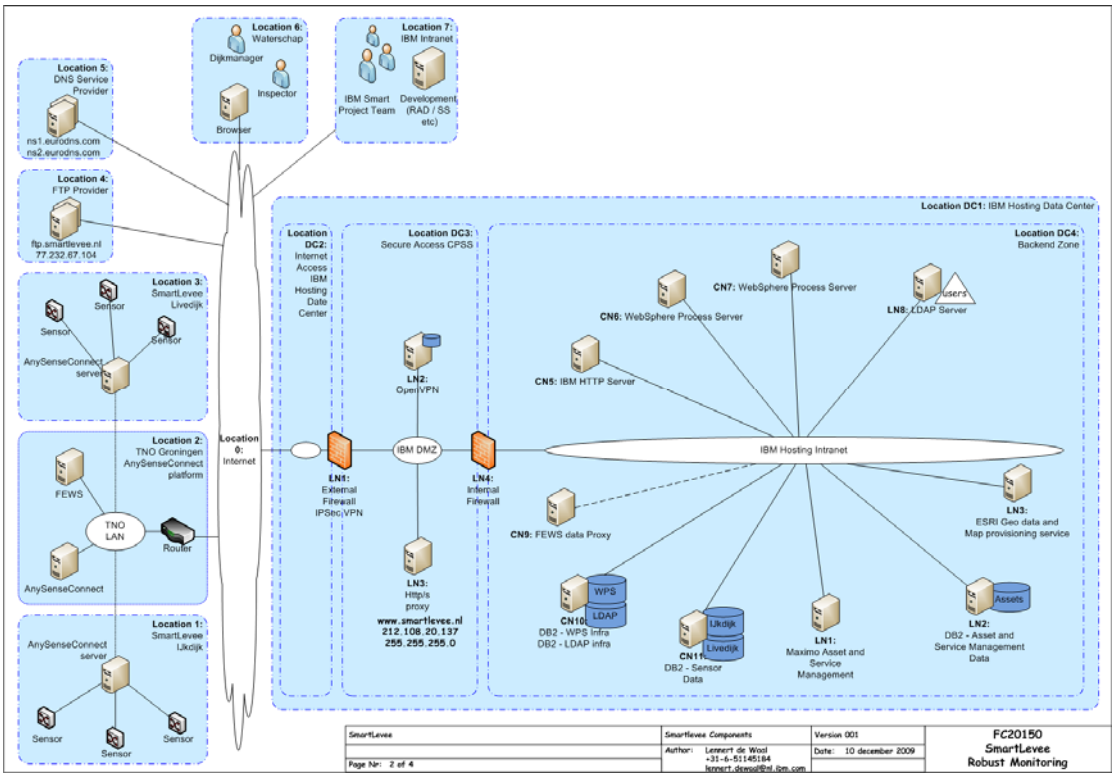

Figure 3-2 Physical Operational Model

The physical model shows components and products used to build a highly available, high performance solution.

Data from the TNO AnySenseConnect platform is sent from this platform to an HTTP-proxy server, placed in a secure-DMZ zone within the IBM datacenter complex. From this proxy server, the data is forwarded to the IBM HTTP Server (IHS). From IHS, the request is forwarded to IBM WebSphere Process Server (WPS). WPS extracts the data from the request, seperating multiple sensors and sensorvalues sent in each request and stores each in the DB2 backend database. From the Datawarehouse, an update is sent to IBM Maximo Asses Management where the value of the sensor-asset is updated.

Routing HTTP requests to application instances running in WAS nodes in a cluster is done by the WebSphere Plugin. This plugin is embedded in the HTTP Server routing requests to WAS. In this architecture the HTTP server is the IBM HTTP Server, bases on the Apache HTTP server. IHS includes the WebSphere Plugin which enables it to forward requests to WAS based on to request-URI. The WAS Plugin contains technology to distibute HTTP requests over n-number of application instances running in a cluster of WAS servers. In the event that one of these application instances or WAS servers fails, the WAS Plugin detects this and routes the request to a different application server hosting the same application. Combined with the WebSphere high availability framework, this effectively eliminates single points of failure in the architecture and provides peer to peer failover for applications and processes running within the Robust Monitoring environment.

WebSphere Application Server
IBM WebSphere Application Server version 6.1 is the industry standard, Java EE 1.4 compliant, application server which includes support for Java Standard Edition 1.5.

WAS is built using open standards and contains IBM technology to support a high performance, high available JAVA runtime environment to run J2EE applications. Seemless failover between running instances is one of the features the WAS platform offers.

WebSphere Process Server
WebSphere Process Server (WPS) v6.1 implements a WS-BPEL compliant process engine and is built on top of WAS. WPS leverages all benefits the WAS platform offers. WebSphere Process Server includes WebSphere Enterprise Service Bus.

DB2
The relational database used in the Robust Monitoring architecture is IBM DB2 Enterprise Server Edition version 9.5. DB2 offers unique features to fascilitate working with very large datasets. It is possible to improve performance and achieve greater scalability by using table partitioning. As tables grow is size, it may be easier to manage the data in chunks or by limited ranges. For the Robust Monitoring architecture the Datawarehouse tables are partitioned used data-ranges. Using this technology very large datasets can be searched very efficiently.

As data availability requirements are high, DB2 offers a High Availability Disaster Recovery replication system based on DB2 logging mechanics. A DB2 HADR scenario is based on a primary and standby implementation. In the event of failure, the standby is promoted to primary by a single command, which can be automated. HADR provides for synchronous replication of all logged DB2 operations. The basic requirement for two systems to participate in a DB2 HADR pair is TCP/IP connectivity. The total overhead of HADR is typically small.

IBM Maximo Asset Management
Maximo provides a comprehensive asset life cycle and maintenance management for all asset types on a single unified platform. Sensors are stored as assets and update via the Enterprise Service Bus.

3.1.3 Virtualisation of components

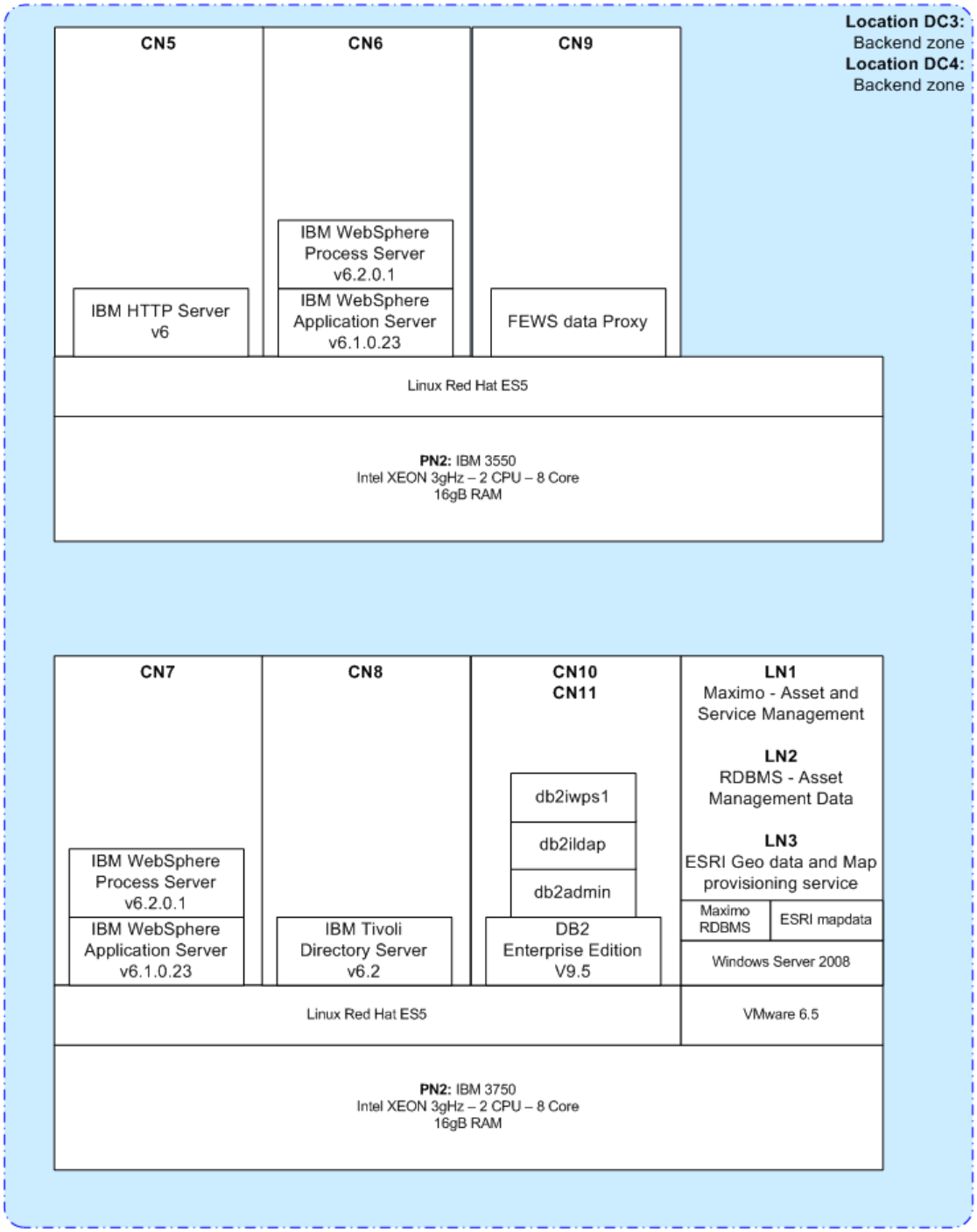

Figure 3-3 Virtualisation of the components

The base platform is a multi-core multi CPU x86 based IBM xSeries Server running RedHat Enterprise Server 5. RAM is 16gB, local storage is a redundant set of 73gB 15kRPM hard disks. To provide sufficient I/O bandwidth, throughput and minimize latency, these systems are conneced to an IBM Storage Area Network using dual GBIC pathways. Database, WebSphere configuration files and VMWare virtual machines are stored on disks in the SAN.

As depicted above, multiple software components are collocated and running in conjunction.

3.1.4 Configuration of WebSphere Cluster

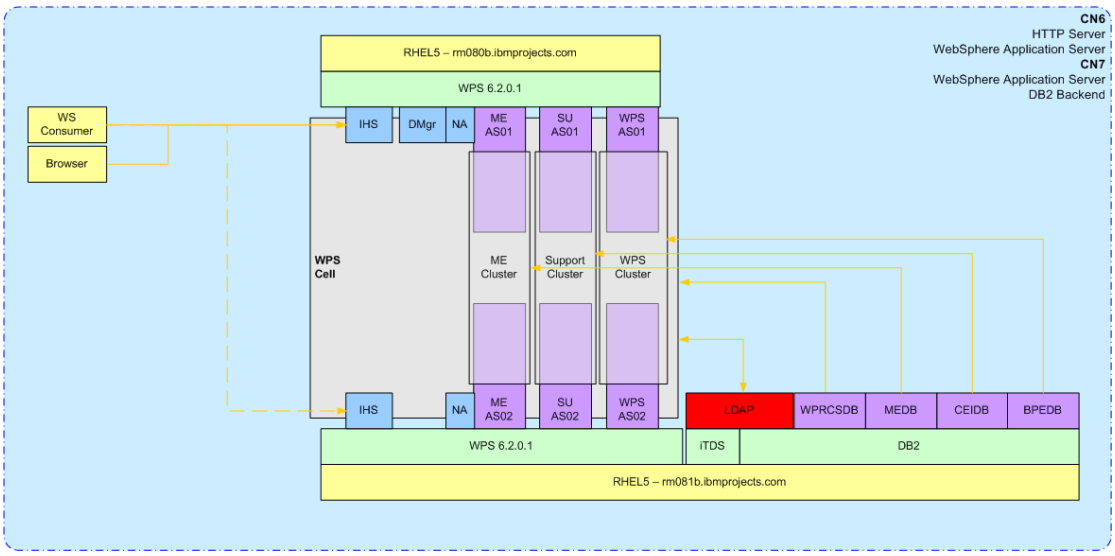

Figure 3-4 Configuration of WebSphere cluster

WebSphere Process Server is configured using the standard IBM Process Server 'Golden Topology' blueprint. This caters for a High Available and scalable WebSphere environment. All BPEL processes and J2EE applications deployed in the Robust Monitoring environment run in an active-active configuration. Because this topology is used, any one component can fail without application downtime.

4 Available assets to be used

This chapter describes re-usable assets (e.g. reference architecture, similar projects, software packages) that might be relevant for the project. And will provide an assessment whether the available assets could contribute to solving the key challenges of the current project. The assessment will be done against the Use Cases, the non-functional requirements and the information requirements.

Re-usable asset	Relevant for Use Case	Description of relevance
TNO AnySense platform	All	The TNO AnySense platform delivers the sensor data to the system and is already in place for this system to re-use with minimal implementation effort
Adobe Flex	Province and Ministery roles	Adobe Flex is a web development environment that enables web developers to rapidly create prototypes that look with a very good look and feel.
IBM Tivoli Maximo	Water Board roles	The Water Boards maintain levees. Maximo is a maintenance / asset management package that is targeted at this usage. It is used at 14 of the 26 Water boards at the moment.
Deltares FEWS	All	Provides forecasting of macro-stability and piping mechanism in levees.

4.1 TNO AnySense platform

TNO's AnySenseConnect platform is capable of retrieving data from sensor systems which deliver data in different ways. AnySenseConnect transforms this different data streams into a normalized data model it can be stored an retrieved.

The delivery (or retrieval) of sensor data can vary in many different ways, these are some examples:

- Some sensor data is time stamped according to the number of seconds passed since 1970
- Some sensor systems timestamp their data relative to the starting time of the system
- Some sensor data are delivered via push mechanism, others need to pulled (requested for data)
- Sensor systems deliver there data in different frequencies, varying from 10 times a second to once in 30 minutes.
- A timestamp, including the date, can be spread over multiple fields,
- Some sensor systems deliver sensor data by writing in a file, this file needs to be polled regularly and a record needs the kept to avoid double reading of sensor values
- Some sensor systems deliver data via sockets over which a specified data format is pushed
- Some systems provide only a direct connection on their database as interfacing possibility
- Some sensor systems provide there data column-based, others line-based and some systems produce a new file of every new measurement.

AnySenseConnect synchronizes timestamps and has a generic mechanism to cope with different interfaces and ways of providing data. AnySenseConnect makes alle necessary translations in order to store is according to normalized datamodel.

The mechanism to cope with the different ways of data delivery by sensor systems is implemented in a workflow, which is illustrated in the figure below. The raw sensor data is parsed according to the specified format as supplied by the manufacture. If the raw data can be parses the sensor data in converted to a in memory data model (map converter) for further processing. A filter is applied to filter out invalid sensor data. The raw, original, supplied data is always stored but not processed in the database. If an invalid reading is supplied the system will notify the administrator, so the error can be analysed. If everything is validated a unique sensor identification number is attached to the measurement and the measurement is posted on the a JMS (Java Message Service) for processing in the database.
For every sensor system a new workflow is configured which works independent from other workflows.

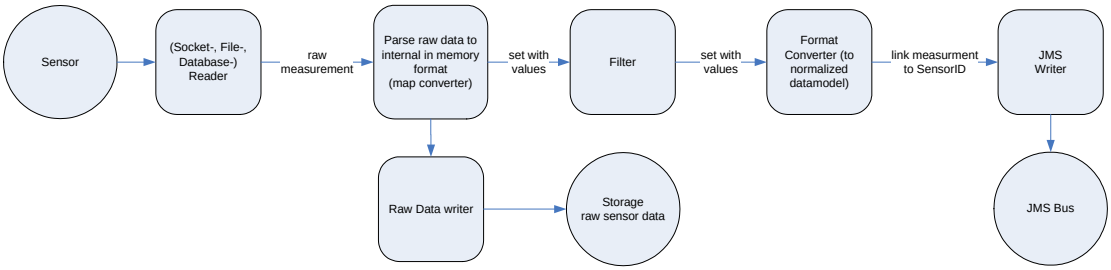

Figure 4-1 workflow for processing raw sensor data

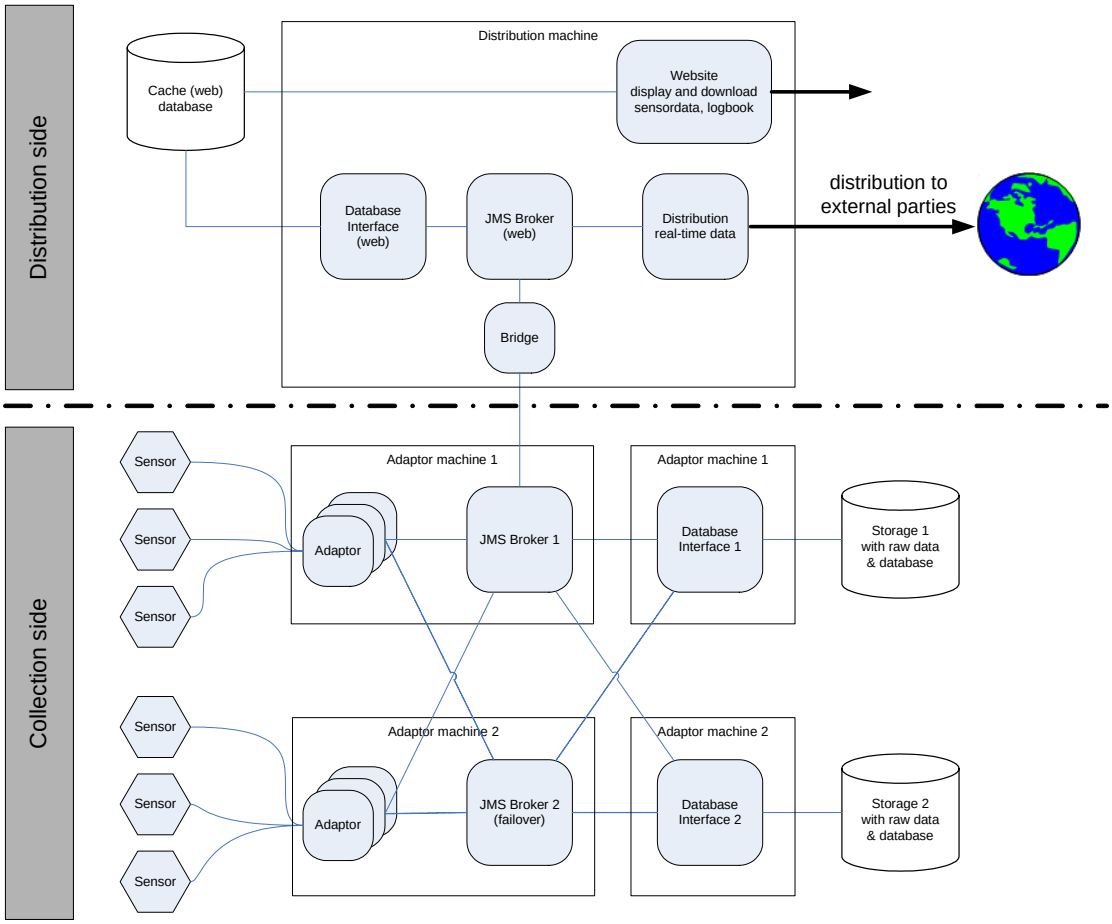

Figure 4-2 AnySenseConnect architecture with collection and distribution side

Figure 4-2 show the architecture of AnySenseConnect. Besides sensor data collection functionality AnySenseConnect has also functionality to deliver sensordata to interested parties. This distribution side of AnySenseConnect was used to deliver normalized sensordata from the ijkdijk experiments and livedike to this Flood Control project.
The collection of sensor data is build up in such a way that most components are redundant and can take over each others tasks if some functionality might fail

4.2 Adobe Flex

See: http://www.adobe.com/products/flex/

Flex is a open source framework for building interactive, expressive web applications it provides a standards-based language and programming model that supports common design patterns. MXML is used to describe User Interface layout and behaviors

Flex can be used to create so called “Rich Internet Applications” which will run in webbrowers using Adobe Flash Player software of on the desktop using Adobe AIR.

Abobe Flex was used to implement a configurable interface for various roles. The result can be found in the visualization document of this project (deliverable 2009.02.02.1)

4.3 IBM Tivoli Maximo

See: http://www.ibm.com/software/tivoli/maximo

IBM Tivoli Maximo is an Asset Management solution. Asset Management delivers a comprehensive view of all asset types — production, facilities, transportation and IT — across your enterprise. This holistic perspective allows you to see all of your assets (levees and the sensors), as well as identify all of the untapped potential within them. You gain the knowledge and control you need to closely align your organization's goals with the overall goals of your business.

Get everything you need to optimize your assets and your business Consisting of six key management modules — asset, work, service, contract, materials and procurement management — The IBM Maximo Enterprise Asset Management solution is what you need to optimize the performance of every asset. To help maximize return on assets, Maximo Asset Management enables you to develop comprehensive programs for preventive, predictive, routine and unplanned maintenance. Together, these programs contribute to your goals of reducing costs and increasing asset uptime.

When you use Maximo Asset Management to help maximize the performance and lifetime value of complex assets and closely align them with your overall business strategy. The end results:

- Improve return on assets.
- Decrease costs and risk.
- Increase productivity.
- Improve asset-related decision making.
- Increase asset service delivery responsiveness and revenue.
- Facilitate regulatory compliance efforts.
- Lower total cost of ownership.

Turn knowledge into decision-making power and asset performance with IBM Maximo Asset Management.

4.3.1 IBM Maximo Spatial

See: http://www-01.ibm.com/software/tivoli/products/maximo-spatial-asset-mgmt/

IBM Maximo Spatial Asset Management enables users to capture, analyze, and display assets, locations, and work orders. IBM Maximo Spatial Asset Management allows asset managers to visualize the spatial relationships among managed assets and other mapped features.

IBM Maximo Linear
See: ftp://ftp.software.ibm.com/software/tivoli/solutionsheets/TIS14023-USEN-00.pdf
IBM Maximo® Linear Asset Manager extends the capabilities of IBM Maximo Asset Management by enabling the management of linear asset types, like:

- Levees or dikes
- Railways
- Roads
- Pipelines
- Power lines

Usually assets are set to a specific point location. But linear assets have stretched form which is important for the management. Linear assets can be segmented in smaller parts. A levee for example contains multiple levee parts.

4.4 IBM WebSphere Infrastructure Software

See: http://www.ibm.com/websphere
WebSphere is infrastructure software for Service Oriented Architecture (SOA) environments that enables interconnected business processes and delivers high performing and robust application infrastructures.
WebSphere is a software family brand of which the following components can be used for robust (levee) monitoring systems. These software products are availability on a variety of platform (Linux, Windows, Sun, POWER, zOS and others).

4.4.1 WebSphere Application Server

See: http://www-01.ibm.com/software/webservers/appserv/was/
WebSphere Application Server is a platform on which Java (J2EE) based SOA application can be run and managed, from business critical and key enterprise-wide applications to the smallest departmental level applications.
WebSphere Application Server offers a central means to manage the highest levels of reliability, availability, security and scalability for all deployed applications.

4.4.2 WebSphere Enterprise Service Bus

See: http://www-01.ibm.com/software/integration/wsesb/
IBM WebSphere Enterprise Service Bus helps enable fast and flexible application integration with standards (SOA) compliant interconnectivity. It's main purpose is to decouple complex integration logic from each application with a central, integration solution eliminating point-to-point connectivity programming.

- Expose existing applications and data as new business services opportunities without impact to the current IT environment.
- Leverage existing ESB infrastructure for universal service delivery and extend easily to federated ESB and service federation model.
- Integrate seamlessly with existing SOA and BPM platform while optimized for WebSphere Application Sever as well as products within the IBM SOA Foundation.
- Reduce ongoing maintenance costs by decoupling connectivity and integration logic from each and every application.

4.4.3 WebSphere Process Server

See: http://www-01.ibm.com/software/integration/wps/
IBM WebSphere Process Server is a high-performance business process automation engine to help form processes that meet your business goals
Built on open standards, it deploys and executes processes that orchestrate services (people, information, systems, and trading partners) within your service-oriented architecture (SOA) or non-SOA infrastructure.
Main features:

- Automates complicated processes that span people, partners, and systems.
- Enable flexible business processes with reusable assets, reducing the need to hard-code changes across multiple applications.
- Centralizing business processes and sharing them across the enterprise to increase ROI.

- Support for human workflow and rapid process changes

WebSphere Process Server is a total solution that includes WebSphere Application Server and WebSphere Enterprise Service Bus.

4.4.4 WebSphere MQ

See: http://www-01.ibm.com/software/integration/wmq/
Makes SOA standard data transport reliable. As the leading ESB Messaging backbone, WebSphere MQ improves the flow of information across organizations.
Main features:

- Leverage existing ESB infrastructure
- Reduce costs and process disruptions from data loss with a unifying solution for guaranteed message/transaction delivery.
- Decoupling connectivity and integration logic.
- Support for (among others) HTTP, REST and JMS.
- Can prevent costly security breach exposures and help achieve compliance through message & transaction integrity.

WebSphere MQ can be plugged into the other mentioned WebSphere infrastructure products to make data transport fail save. This product was also used in the Robust Sensor Networks project of 2008.

4.4.5 WebSphere Integration Developer

See: http://www.ibm.com/software/integration/wid/
WebSphere Integration Developer is the Eclipse-based development tool for building SOA-based BPM and integration solutions across WebSphere Process Server, WebSphere ESB, and WebSphere Adapters.
Main features:

- Constructs integration solutions using intuitive drag-and-drop technology to visually define the sequence and flow of business processes.
- Simplifies integration by encouraging reuse and efficiency.
- Enhanced mediation tooling support.
- Integrates testing, debugging, and deployment for solution development.
- Enables Business-Driven Development, fully integrating with WebSphere Business Modeler to import models for rapid implementation.
- Supports generation of human interaction user interfaces that can be easily customized.

4.5 Deltares FEWS

See: http://www.wldelft.nl/soft/fews/int/index.html

Deltares' Flood Early Warning System (Delft-FEWS) provides a state of the art flood forecasting and warning system. The system is a sophisticated collection of modules designed for building a flood forecasting system customised to the specific requirements of an individual flood forecasting agency.
The philosophy of the system is to provide an open shell system for managing the forecasting process. This shell incorporates a wide range of general data handling utilities, while providing an open interface to a wide range of forecasting models.

The modular and highly configurable nature of the system allows it to be used effectively both in rudimentary forecasting systems and in highly complex systems utilising the full range of hydrological and hydraulic modelling. Delft-FEWS can either be deployed in a stand-alone, manually driven environment, or as a fully automated distributed client-server application.

4.5.1 DAM module

The module of DAM is coupled to FEWS to FEWS-DAM. FEWS is a Flood Early Warning System, which is developed by Deltares. In FEWS, sensor data can automatically imported, model calculations and predictions can be made and visualized and output can be generated. In FEWS-DAM, FEWS is the platform for data import and output and visualization. DAM is the module which performs the model calculations on dike stability.

The software DAM which was developed in 2008 had a long computation time. For real time data, fast answers are needed. Therefore, the kernel of the DAM software was improved in project another Flood Control 2015 project. In effect, it is now fast enough to consider it (almost) real-time. A computation on the stability of a dike of considerable length now takes 5 minutes, instead of one day.

The underlying models are Mstab and Mpiping.

4.5.2 MStab and Mpipings models

The models used in FEWS DAM are Mstab and Mpiping. These models use a geometry and several geotechnical parameters of the levee and a freatic water line or the difference in water pressure at both sides of the levee as input parameters for dike stability calculations. In Mstab and Mpiping, the output of the model is a safety factor or a piping factor respectively. In DAM the result of the calculation is coupled to alert levels with a traffic-light code. An example of the traffic-light for the stability factor (macro-stability) is:

- Green, value > 1.17: dike stability sufficient.
- Yellow, value between 1 and 1.17: an indication of decreased levee stability.
- Orange, value between 0.85 and 1: dike stability severely decreased.
- Red, value below 0.85: dike stability insufficient.

An example of the traffic-light for the piping factor (piping) is:

- Green, piping factor > 3: dike stability sufficient.
- Yellow: piping factor between 1.5 and 3:
- Orange: piping factor between 1-1.5:
- Red: piping factor between 0 and 1: dike stability insufficient.