SINTEF

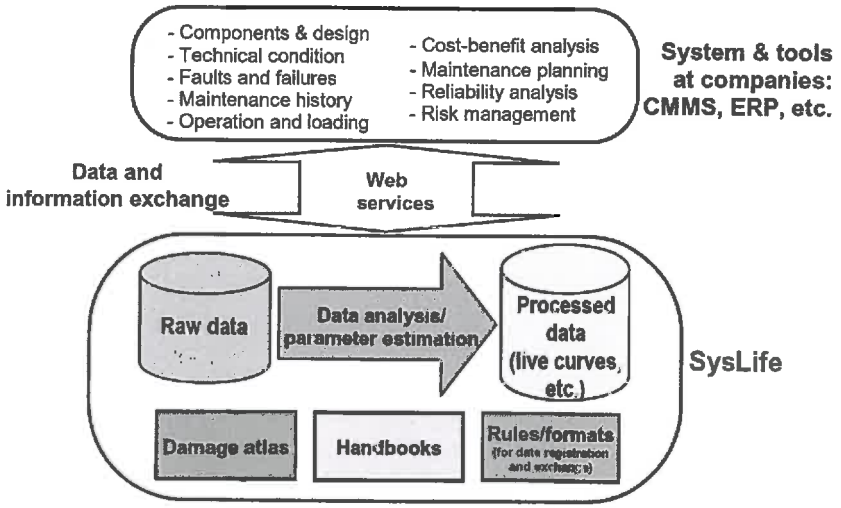
A25900- Unrestricted

Report

SysLife System Architecture Specification

Subtitle

Author(s)
Franck Chauvel
Arnor Solberg



SINTEF ICT
Networked Systems and Services
2014-01-28

SINTEF

SINTEF ICT
Networked Systems and Services
2014-01-28

SINTEF IKT
SINTEF ICT

Address:
Postboks 124 Blindern
NO-0314 Oslo
NORWAY

Telephone:+47 73593000
Telefax:+47 22067350

postmottak.IKT@sintef.no
www.sintef.no
Enterprise /VAT No:
NO 948 007 029 MVA

Report

SysLife System Architecture Specification

KEYWORDS:
SysLife, software architecture, specifications

VERSION
1.0

DATE
2014-01-28

AUTHOR(S)
Franck Chauvel
Arnor Solberg

CLIENT(S)
SysLife project

CLIENT'S REF.
Bjarne Børresen

PROJECT NO.
102002263

NUMBER OF PAGES/APPENDICES:
39 + Appendices

ABSTRACT

Abstract heading

This document describes the software architecture of the SysLife system and contains descriptions of the system components, the main components' services and how the components interact. The document provides the overall system understanding, and it can be used by software engineers as a reference document and to prepare for software maintenance or evolution.

PREPARED BY
Franck Chauvel

SIGNATURE 

CHECKED BY
Thomas Welte

SIGNATURE 

APPROVED BY
Bjørn Skjellaug

SIGNATURE 

REPORT NO.
A25900

ISBN
978-82-14-05344-9

CLASSIFICATION
Unrestricted

CLASSIFICATION THIS PAGE
Unrestricted

Document history

VERSION	DATE	VERSION DESCRIPTION
0.1	2012-01-13	Version presented at the working group meeting February 8th, 2012
0.2	2013-03-20	Addition of the architecture evaluation and corrections of minor typos
0.3	2013-10-23	Addition of the description of relational data structure used to store raw data and correction of minor typos
0.4	2013-12-17	Correction of minor typos and final alignment with respect to the current implementation by Catenda
0.5	2014-01-15	Final proof-reading and minor typos correction by SINTEF ICT
1.0	2014-01-28	Final version

Table of contents

1 Introduction

The main objective of the SysLife project is to develop an information system (also called SysLife system) for collecting, storing, processing and reporting data/ information related to lifetime analysis and estimation of failure probability for power system components.

This report describes the overall software architecture of the SysLife system and the design decisions that were made throughout the SysLife project. This final version concludes the initial analysis that were conducted during a SysLife pre project in 2010 [1] and the subsequent architecture specification.

This report describes the software architecture of the SysLife system and contains descriptions of the system components, their main features and how they interact. The document provides an overall system understanding, and it can be used by software engineers as a reference document or to prepare for software maintenance or evolution. However, this document does not document the source code of the SysLife system.

This report is structured as follows: Section 2 briefly recalls the methodology that has been used to design and conceive the SysLife system. Section 3 presents the business architecture, including the main actors and their roles and responsibilities. This model is then further refined into system requirements, which are described in Section 4. In Section 5 we specify the various underlying services of the SysLife system. Section 6 presents the current data model, while Section 7 presents the technical architecture of the SysLife system. Section 8 presents a qualitative assessment of the proposed architecture before Section 9 concludes.

2 Specification Methodology

For elaboration and specification of SysLife, the SiSaS[1] methodology is applied. The SiSaS methodology is developed by SINTEF, and is based on state-of-the-art principles for iterative and incremental development using a model-driven process. The objective is to develop the system with a service-oriented and distributed architecture. In this chapter, a brief overview of the method is provided. More detailed information about the methodology can be found at http://epf.thingml.org/wikis/sisas/index.htm.

The development process is model driven. Four models involved in the process (see also Figure 2.1):

- Business Architecture Model (BAM)
- Requirements Model (RM)
- Service Architecture Model (SAM)
- Platform Specific Model (PSM)

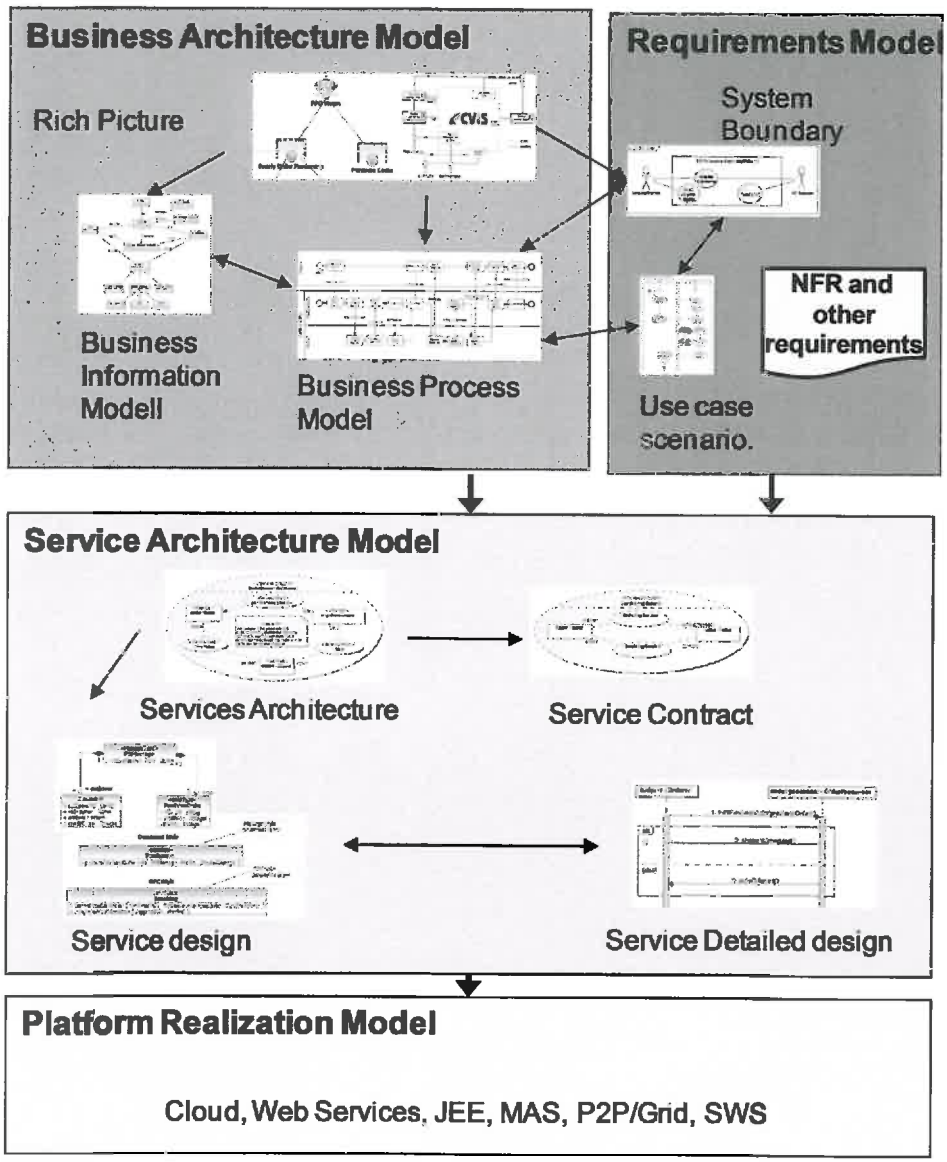


Figure 2.1 Models and work products

[1] SiSaS is currently developed as part of the SiSaS project (Scientific Software as a Service), this project is a strategic research project funded internally by SINTEF.

3 Business Architecture Model (BAM) for SysLife

The set of work products of the Business Architecture Model is depicted in Figure 3.1. The business architecture model contains the following work products:

- The rich picture, which is a domain[2] picture aiming to give an overall understanding of the domain.
- The business information model, which is used to identifying and defining the main concepts of the domain. This model typically embraces actors and information objects that flow within the business processes of the domain.
- The business processes model, showing the main processes and value chains of the domain.

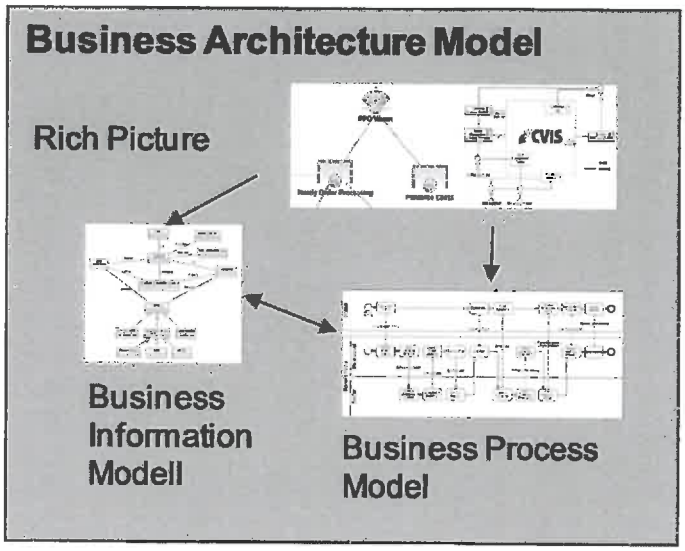


Figure 3.1 Business Architecture Model Works Products

3.1 Rich Picture

Rich pictures are used to provide a model for thinking about the system and to help the analyst to gain an understanding of the problem situation. Rich pictures are artistic and individualistic expressions, and therefore neither "right" or "wrong". However, rich pictures should represent structure, processes and issues of the organization which could be relevant to the problem definition.

The initial SysLife rich picture is shown in Figure 3.2. The light blue arrows in the figure represent transfer of information/data between different systems, or between systems and system users. The main flow of information goes from the maintenance operators (*maintenance data collectors*), who collects raw data and makes observation of the condition of the components data, to the *maintenance planner*, who carries out analyses and makes maintenance decisions. In addition, *SysLife experts* may add additional knowledge data to the system. The *SysLife expert* may gain the knowledge from *Expert groups* on different components. Furthermore, the information in SysLife may be supervised by the *quality assurer*, and *equipment vendors* may add additional information.

Raw data is received by the system from external systems, such as the Computerized Maintenance Management System (*CMMS*) or the Enterprise Resource Planner (*ERP*), and alternatively, from systems that collect and analyse *sensor data*. SysLife provides information to the maintenance planner in terms of *processed data* (e.g., *life curves*).

The reliable and safe operation of SysLife is ensured by the *SysLife system operator*, who also maintains the system (illustrated by the red arrow).

There are several stakeholders that may have interest in the system. These are:

[2] Domain = Area of concern or Area of interest

- SysLife owner
- CMMS/ERP system vendor
- Facility owner (owner of power plants, electricity grid, ...)
- Risk manager
- REN (Rasjonell Elektrisk Nettforvaltning)
- Government/NVE/DSB
- Providers of other data sources (e.g. Statnett, NVE, ...)
- Strategist/network evolution
- Benchmarking group

Some of the stakeholders' functions and potential interests in SysLife are shown in the rich picture.

Note that the aim of the SysLife rich picture is to set up the context in which the design was conducted. It includes all relevant aspects and he most important stakeholders, processes and issues. The final SysLife system developed in the SysLIfe project does not include all the aspects shown in this rich picture, but only some of them. This means that the rich picture can be a basis for further system developments to extend the system to cover aspects that are not included in the current version. The support for high frequency sensor data coming from continuous monitoring, for example, was not included in the final system developed in the project.

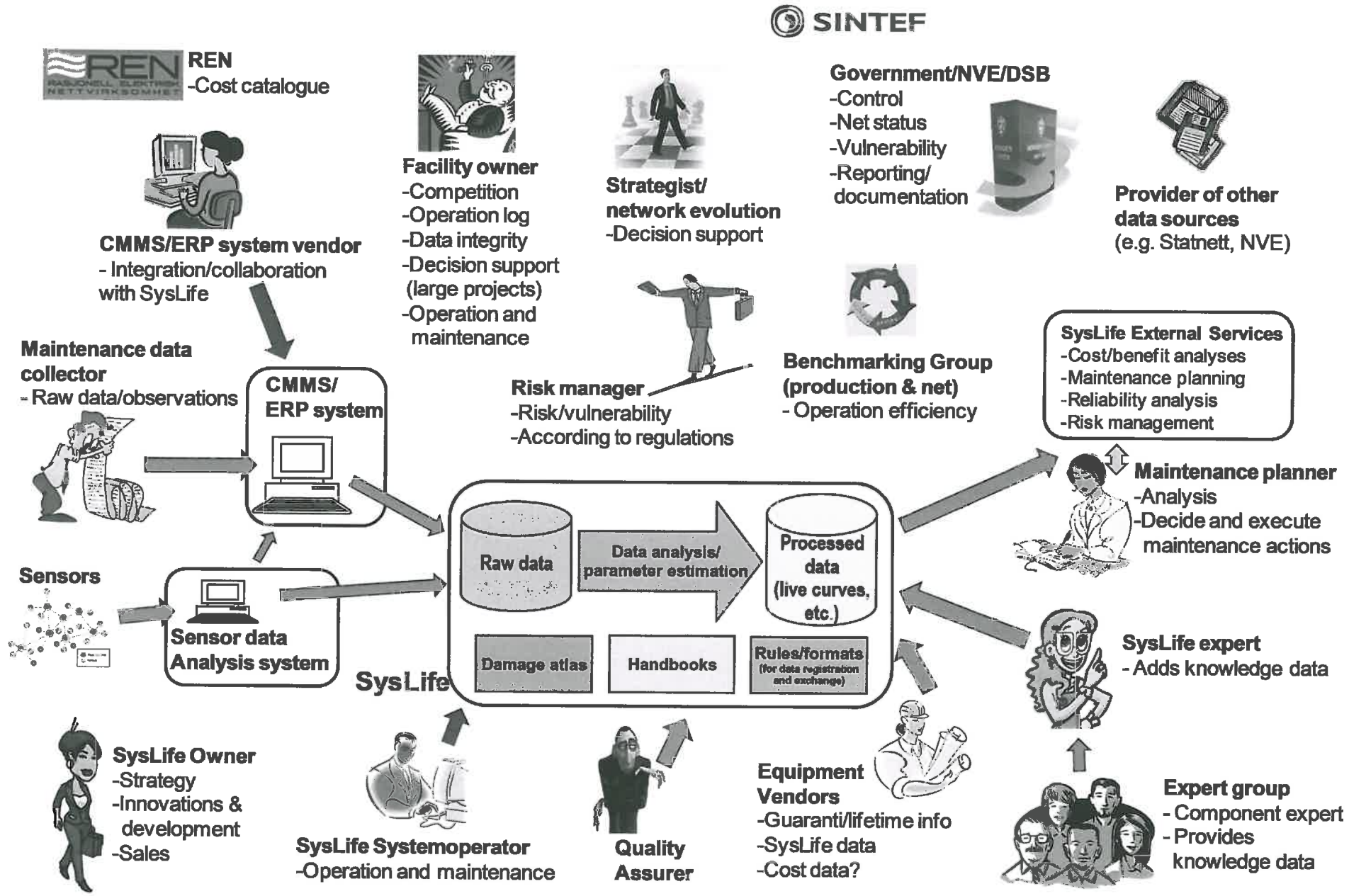


Figure 3.2 SysLife Rich Picture

3.2 Business Information Model

A business Information Model (BIM) can be thought of as a conceptual model of a domain of interest (often referred to as a problem domain) which describes the various entities[3], their attributes and relationships, plus the constraints that govern the integrity of the model elements comprising that problem domain. The BIM is created in order to represent the vocabulary and key concepts of the problem domain. The BIM also identifies the relationships among all the entities within the scope of the problem domain, and commonly identifies their main attributes.

An important benefit of a BIM is that it describes and constrains the scope of the problem domain. The BIM can be effectively used to verify and validate the understanding of the problem domain among various stakeholders. It is especially helpful as a communication tool and a focusing point both amongst the different members of the business team as well as between the technical and business teams. A first version of the SysLife BIM, which has to be further developed, is illustrated in Figure 3.3.

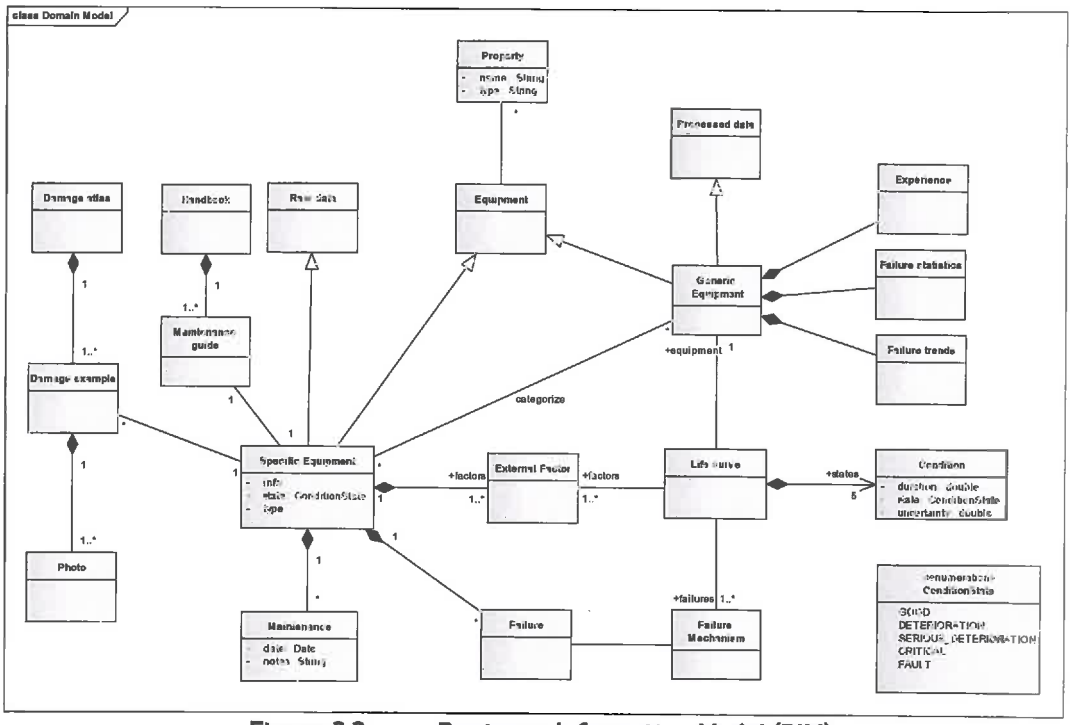


Figure 3.3 Business Information Model (BIM)

[3] An entity typically denotes articles or things; it can be both physical and non physical/abstract things such as handbook, damage example, lifetime curve and experience.

The main parts of the SysLife system are the raw data and the processed data. The raw data consists of the condition, as well as a history of condition development, maintenance work and failure events for a specific (real) equipment. The processed data consists of life curves for generic equipment, as well as different types of statistics and analysed data that the maintenance operators can use to define strategies and plan their work. Basis for both the raw and processed data is a common equipment structure (template) including various equipment properties.

The system contains also handbooks that serve as guides for the maintenance operators, as well as a damage atlas with photos of different types of damages.

3.3 Business Process Model

Figure 3.4 shows the overall business process view of SysLife. The figure does not show all system functionalities, but focuses on the processes that give business value. The figure shows also the flow of information between these processes. More specifically:

- The raw data is produced mainly by the *maintenance data collector*, which here includes data coming from inspections, maintenance and sensors, while equipment data is provided by an *Equipment Data Registrator* which could be a *SysLife expert* and based on information coming from *expert groups*, or it could be the *equipment vendors* themselves playing this role. The raw data is stored in the *SysLife DataStore*. The SysLife DataStore stores both raw and processed data
- Next, the raw data is processed by the *Experts*. Figure 3.5 details this business process. The experts take existing raw data and can both manage the data or add data, before performing data analysis. The analysis can produce a variety of processed data, such as experiences, warnings, statistics and life curves. The analysis results are also stored in the *SysLife DataStore*.
- Then, the *quality assurer* can audit the processed data.
- Finally, the *Maintenance planner* can use the processed data to perform risk-based maintenance planning.

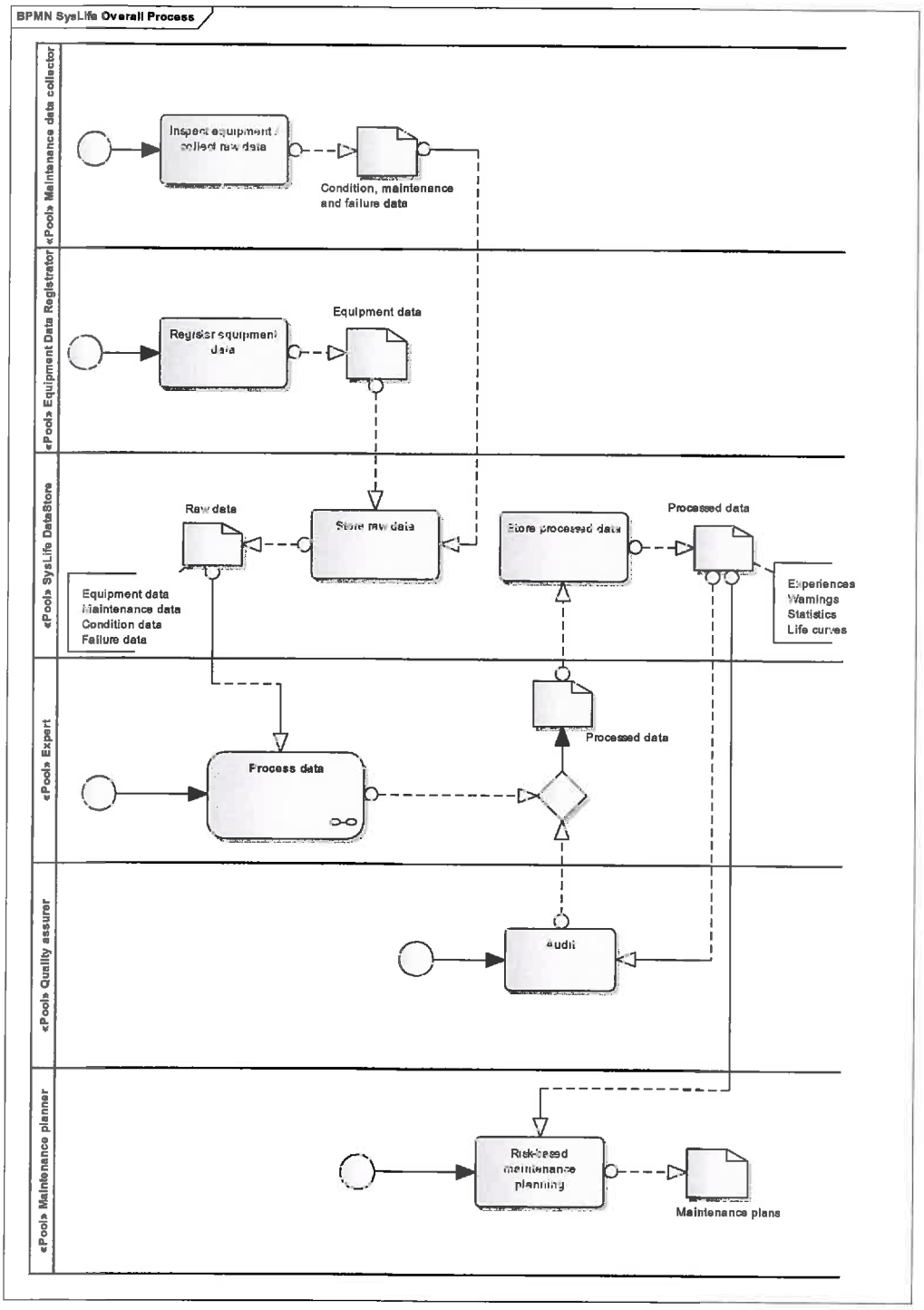


Figure 3.4 **Overall value chain of SysLife**

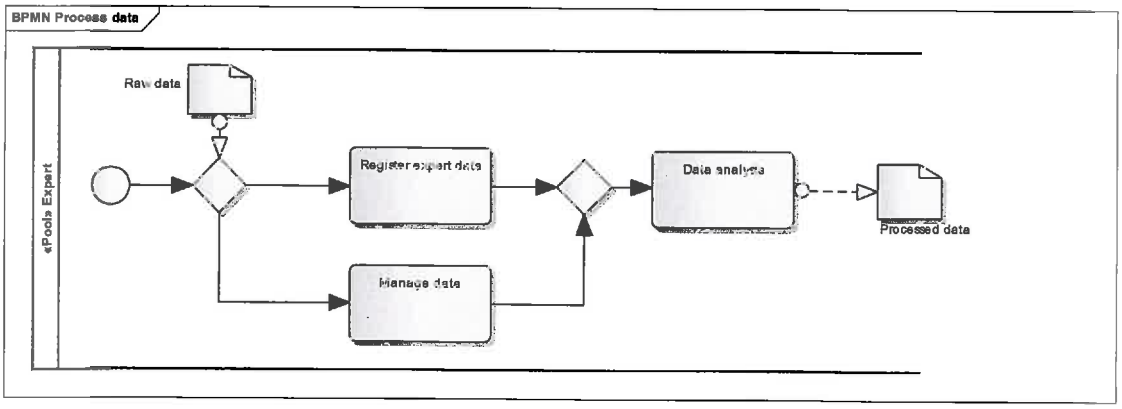


Figure 3.5 The SysLife expert tasks

4 Requirements Model for SysLife

The Requirements Model (RM) focuses on identifying the system requirements, and consists of the *system boundary model, use case scenarios* and *non-functional and other requirements* (Figure 4.1). In the early stages of the project, we have developed a system boundary model that represented our current understanding of the system. The subsequent phases of the SysLife project explored the use case scenarios and through these evolved the system boundary model, as well as looking at Non-Functional Requirements (NFRs).

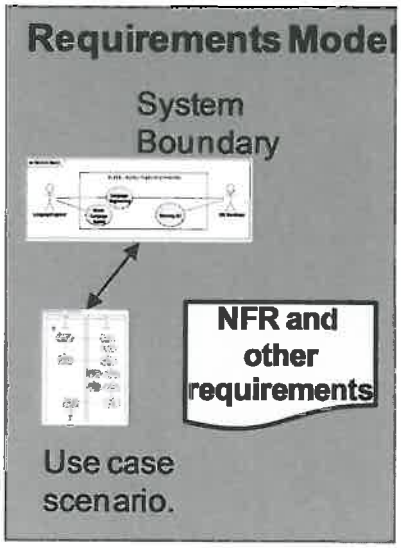


Figure 4.1 Requirements Model work products

4.1 System boundary

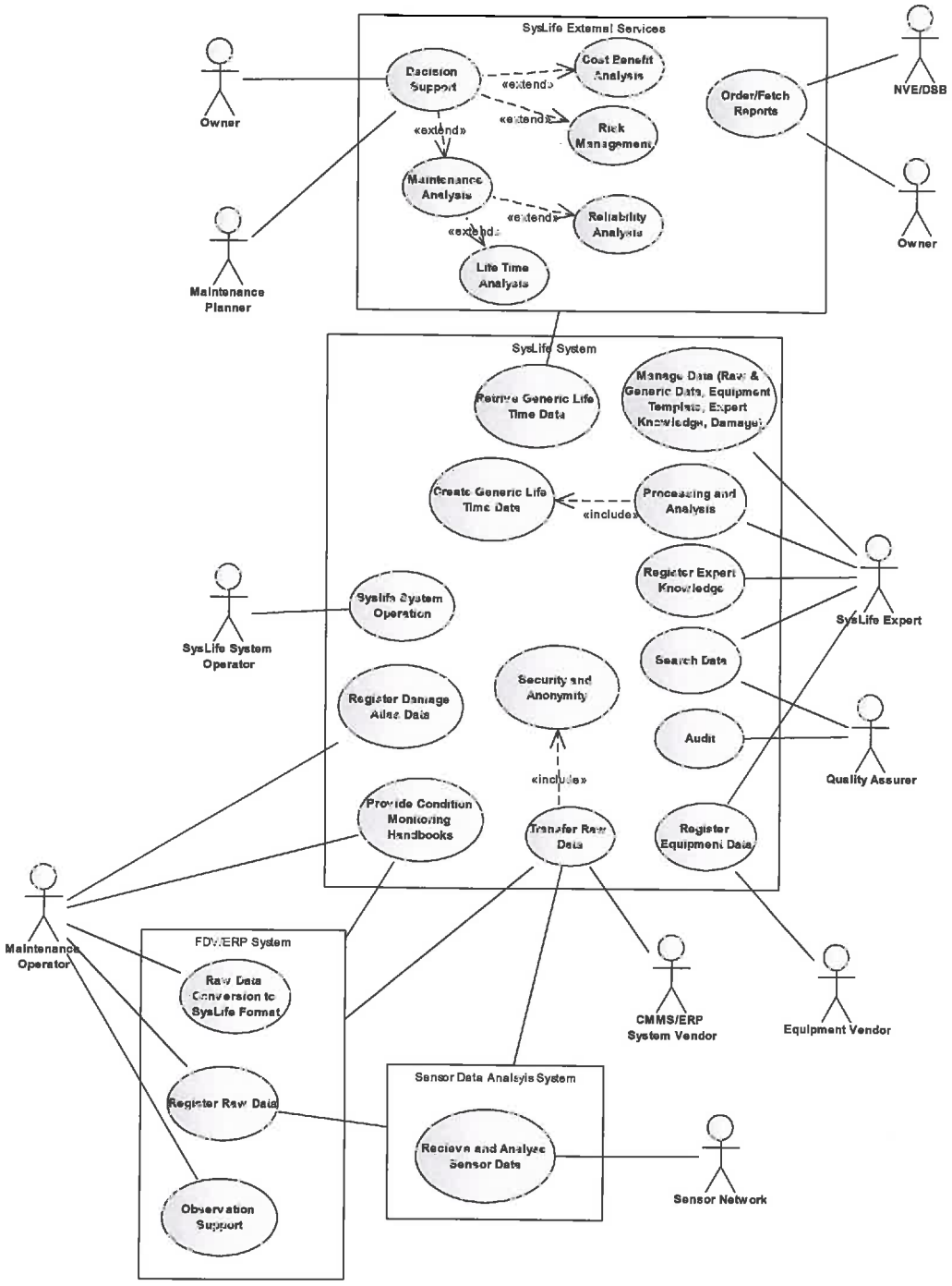


Figure 4.2 depicts the system boundary model for the SysLife system. This model shows the boundaries between the interacting systems and stakeholders. As it can be seen from the figure, there are several groups of stakeholders in the system:

- Those who enter raw data into the system concerning equipment and observations:
  - Maintenance Data Collector, Sensor Network, CMMS/ERP System Vendor and Equipment vendors.
- Those who validate and process the raw data:
  - SysLife System Operator, Quality Assurer and SysLife expert.
- Those who use the processed data:
  - Maintenance planners, owners and other bodies of interest (NVE/DSB)

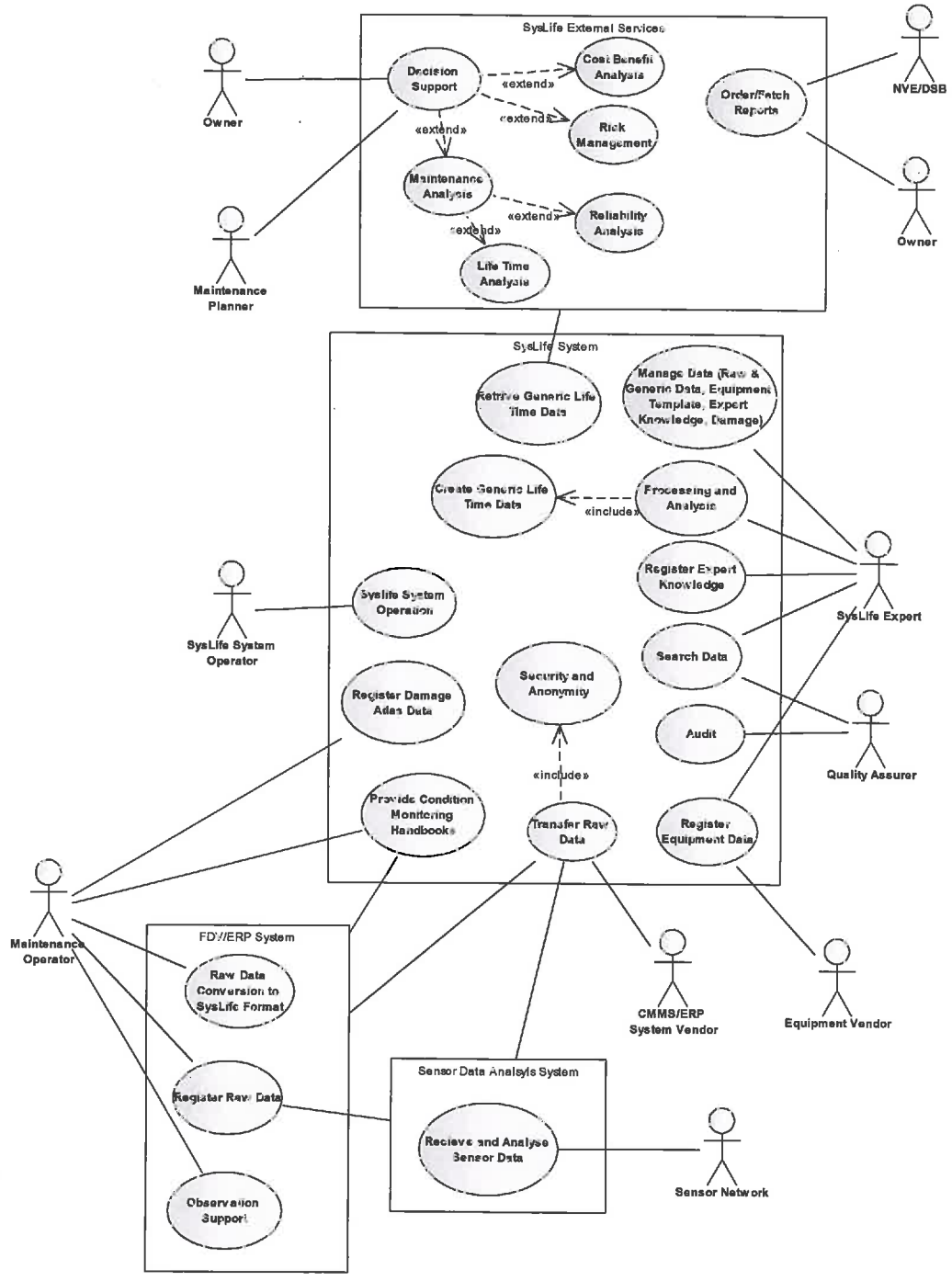


Figure 4.2 System Boundary Model

The figure shows several actors/stakeholders and other systems that interact with the SysLife system. The functionality of the SysLife system itself is described by the 13 following use cases:

1. Transfer raw data.
    a. *Actors*: Maintenance data collectors, CMMS/ERP systems, Sensor data analysis system

   b. *Description*: The system must allow the actors to transfer raw data (equipment condition, maintenance reports, failures, etc.) into the system. An important aspect of realising the SysLife system is to make it interoperable with the data formats of the other systems that provide input, such as the FDV/ERP systems or sensor systems.
   c. *Other*; Includes the 'Security and anonymity' use case.
2. Security and anonymity
   a. *Actors:* Same as for 'Transfer raw data'
   b. *Description:* The system must ensure that necessary security and privacy measures are upheld during the transfer of raw data.
3. Provide condition monitoring handbooks
   a. *Actors:* Maintenance data collectors and the FDV/ERP systems.
   b. *Description:* The SysLife system contains handbooks that guide the maintenance workers on how to determine and analyse the condition of the equipment. These handbooks need to be available in an appropriate format for the maintenance workers during their work.
4. Register damage atlas data
   a. *Actors:* Maintenance data collector
   b. *Description:* When the maintenance workers are inspecting or performing work on equipment, they should take pictures to document the condition. These pictures are basis for the damage atlas and should be registered in SysLife if the system does not contain equivalent pictures from before.
5. Register equipment data
   a. *Actors:* SysLife expert or Equipment vendor
   b. *Description:* The SysLife expert and the equipment vendors should be able to register new equipment and equipment related data (properties) in the system.
6. Audit
   a. *Actors:* Quality assurer
   b. *Description:* To ensure that the processed data is reliable, experts need functionality that allows them to audit the data that others have entered.
7. Search data
   a. *Actors:* Quality assurer and SysLife expert
   b. *Description:* The SysLife system needs robust mechanisms for finding and filtering data.
8. Register expert knowledge
   a. *Actors:* SysLife expert
   b. *Description:* Experts need to be able to enter expert knowledge about the equipment properties related to life curves and expected lifetime.
9. Processing and analysis
   a. *Actors:* SysLife experts.
   b. *Description:* An important function of the system is to provide processed lifetime data for generic equipment either by processing raw data or adding expert knowledge. This use case will provide the SysLife experts with the means to generate processed data, which are the main output of the SysLife system. The processed data is e.g., used in the *SysLife external services* by the maintenance planer for maintenance scheduling.
10. Create generic lifetime data
    a. *Actors:* SysLife expert
    b. *Description:* Generic lifetime data is the main result of the SysLife system. Processed data (e.g., life curves or expected values for a specified set of parameters) are specified for generic equipment.
11. Manage data
    a. *Actors:* SysLife expert
    b. *Description:* The SysLife expert should generally be given the opportunity to manage and change both raw and processed data.
12. Retrieve generic lifetime data
    a. *Actors:* SysLife external services
    b. *Description:* There needs to be a specific interface to the SysLife system to get the processed data and reports. Considerations need to be taken regarding the data format of the output so that it is easily digestible by the external services. This use case enables the end users of

SysLife (e.g., the maintenance planner) to retrieve processed data that finally is used in the SysLife external services for further analyses.

13. SysLife system operation
    a. *Actors:*SysLife System Operator
    b. *Description:* The system operator operates and maintains the system and is responsible for further development of SysLife.

4.2 Non-functional and Other Requirements

This subsection identifies non-functional and other requirements of the SysLife system.

4.2.1 Extra-functional Requirements

Some initial non-functional requirements have been identified and are listed in the following:

- *Data security and integrity*: It is required to ensure anonymity of data that is fed into the system by different data producers.
- *Data quality*: Data fed into the system need to be qualified and processed in order to:
  - ensure correctness of data
  - check the relevance of the data (e.g., include production year, history/logs, etc.)
  - evaluate the completeness of the data
  - ensure unambiguous data
- *IT security in general*: The system needs to adhere to certain security standards. Threats and risk analysis will be performed as part of the system design to investigate and decide on security issues
- *Sufficient data base*: It is critical for the SysLife system to ensure input of sufficient raw data. The minimum amount of data which is necessary for providing significance of the analyses must be identified.

Note that more detailed elaboration of the data security and integrity is provided in a separate project document.

4.2.2 Other Requirements

- SysLife should be based on modern and best practice architecture and technology principles for distributed networked systems. SysLife will be based on Service Oriented Architecture (SOA) principles and technologies [2]
- Processed data must be adapted to the needs of the SysLife external services such as the existing Excel based tool for estimation of failure probability or the tool for calculation of profitability.

5 Overall Service Architecture Model for SysLife

The Service Architecture Model (SAM) describes the overall architecture of the system and its partitioning into services in terms of collaborations of services and sub-services, services structures and service contracts including service interfaces and protocols. Figure 5.1 shows the four diagrams that are a part of the service architecture model. In this phase of the project, we are focussing on the upper two diagrams: *Services architecture* and *Service contracts*. The *Service design* and *Service detailed design* will be elaborated in the SysLife project.

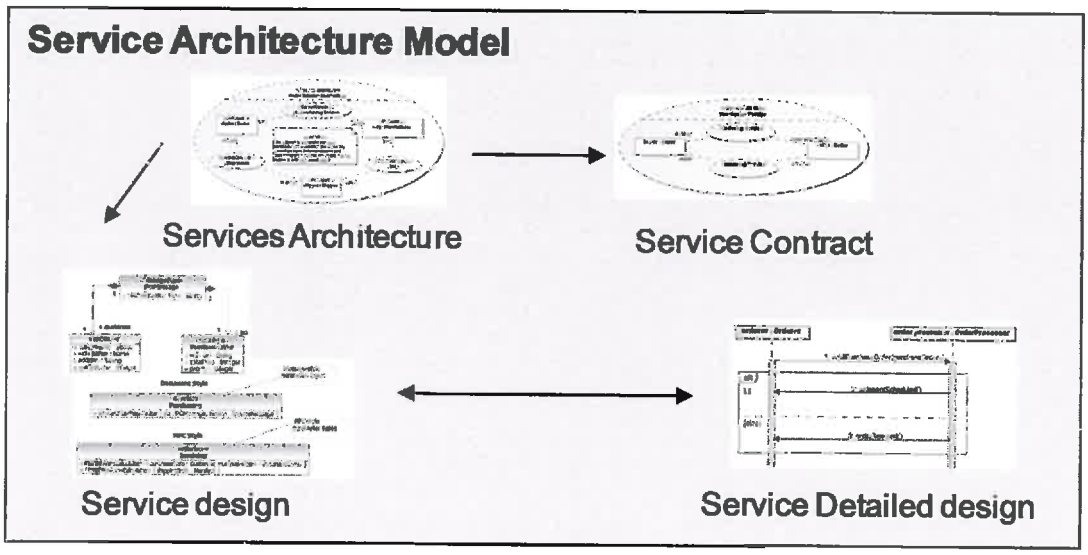


Figure 5.1 Service Architecture Work Products

5.1 Services Architecture

Figure 5.2 depicts the overall service architecture for the SysLife system. It is currently divided into seven main services:

- *Registering Raw Data*: The system should be able to receive information from the systems used by the maintenance operator (FDV and ERP) and from the sensor networks (condition monitoring).
- *Manage SysLife Data*: The system shall allow SysLife operators to perform create, read, update, delete operations (CRUD operations) on all SysLife data, including the raw and processed data.
- *Manage Equipment*: The system shall allow experts to define new equipment (data, templates, etc.) and to adjust or delete existing ones.
- *Manage Live Curve*: The system shall allow experts to insert and update expert knowledge such as Life Curves for instance.
- *Analyse Raw Data*: The System shall allow SysLife experts to process the raw data into data useful to the maintenance planners (such as life curves and expected lifetime).
- *Audit data:* The system must allow quality assessors to control and validate the existing data.
- *Browse processed data*: The system shall allow users to retrieve data and reports that support the decision making process concerning maintenance planning.

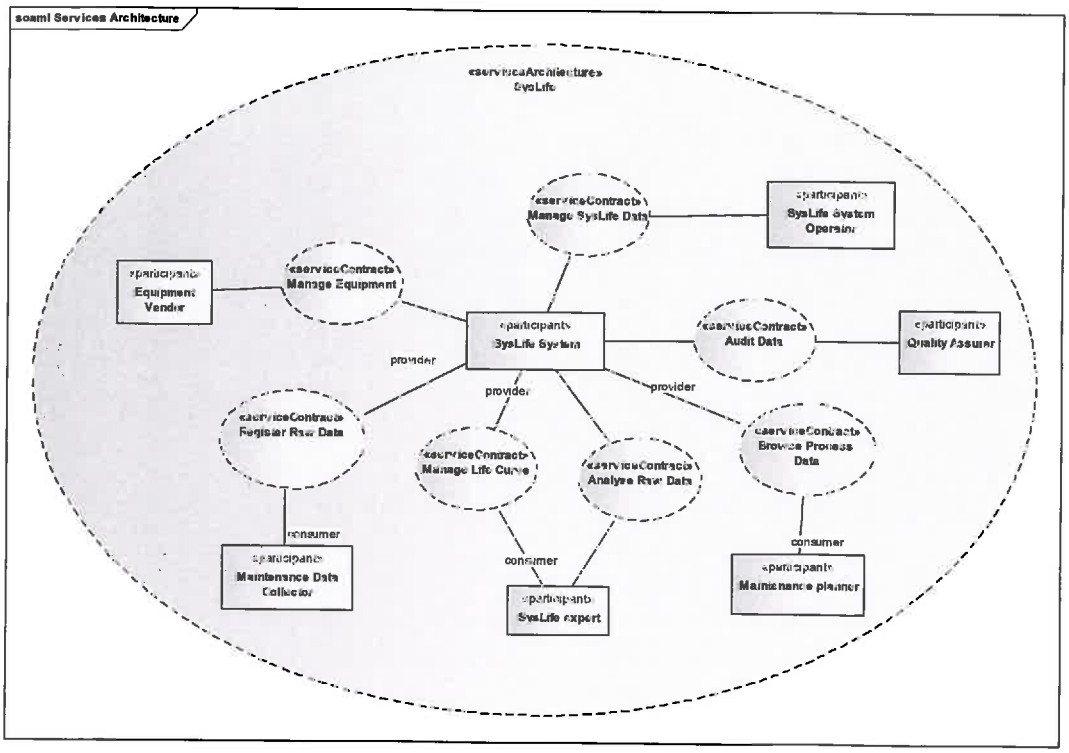


Figure 5.2 Overall Service Architecture for SysLife

5.2 Service Contracts

In the following, we describe the services depicted in the services architecture diagram (Figure 5.2). Each of these services can be further decomposed into more detailed services, but at this stage, we provide preliminary service contract descriptions that are meant to capture the high-level intent of the service. As the project progressed, the development of the *service design*, and in particular the *service detailed design*, provided more detailed interface specifications in these models as well.

5.2.1 Browse Processed Data

Figure 5.3 depicts the contract attached to the service "BrowseProcessedData". This service provides the SysLife expert with the ability to explore existing processed data, especially life curves. This service is a "single interface" and requires no specific additional capability by the user of this service, namely the SysLife expert.

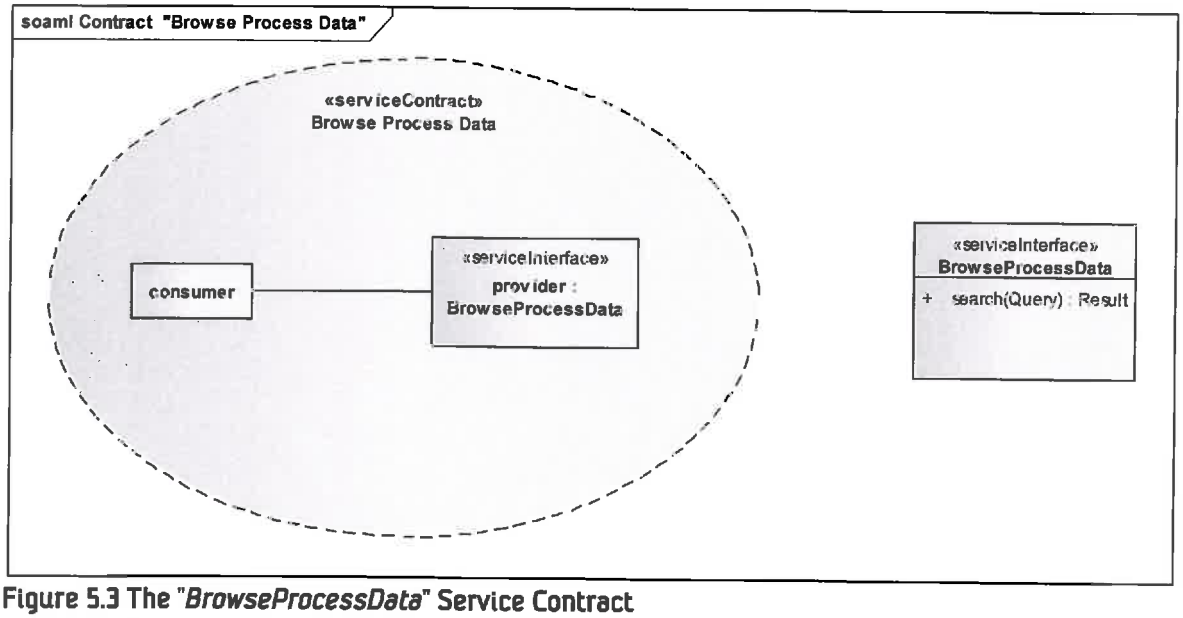


Figure 5.3 The "*BrowseProcessData*" Service Contract

Analyse Raw Data and Create Processed Data Figure 5.9 below portrays the capabilities of the "AnalyseRawData" service. This service basically provides a means to analyse raw data, and produced the so called "processed data", especially life curves. This service is provided to the SysLife experts, who can add life curves directly according to their judgment or trigger statistical regression that will infer the life curve for existing equipment. This service is also a single interface service, as there is no specific additional capability required from its user.

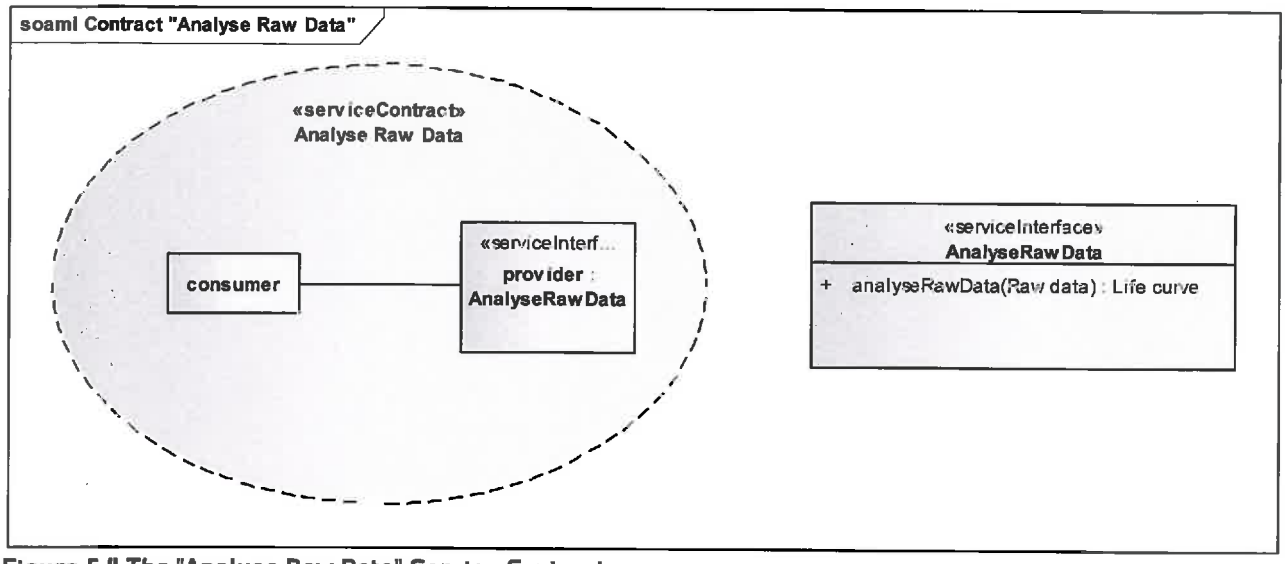


Figure 5.4 The "Analyse Raw Data" Service Contract

5.2.2 Manage Life Curve

Figure 5.5 below details the "Manage Life Curve" service contract. It is the companion service of the previous one depicted on Figure 5.4. This service allows the SysLife expert to add new life curve based on its judgment (by contrast with the previous, which triggers statistical regression to estimate life curves). It also allows the SysLife expert to manage existing life curve, basically adding, modifying, or deleting them as needed.

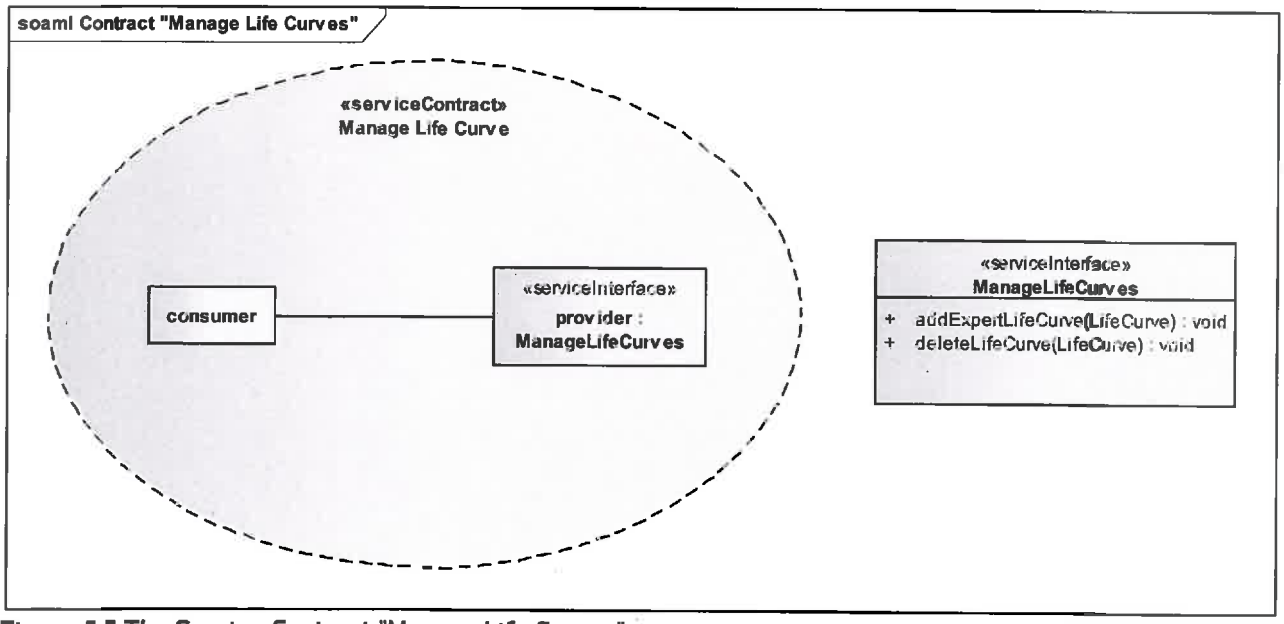


Figure 5.5 The Service Contract "Manage Life Curves"

5.2.3 Audit Data

Figure 5.6 details the "Audit Data" service, which realizes the quality control associated with the SysLife system. This service allows the Quality Checker to search for issues in the data stored in the SysLife system, such as inconsistency, redundant entries, missing information, etc. He may as well search manually through the data to detect and correct more peculiar issues.

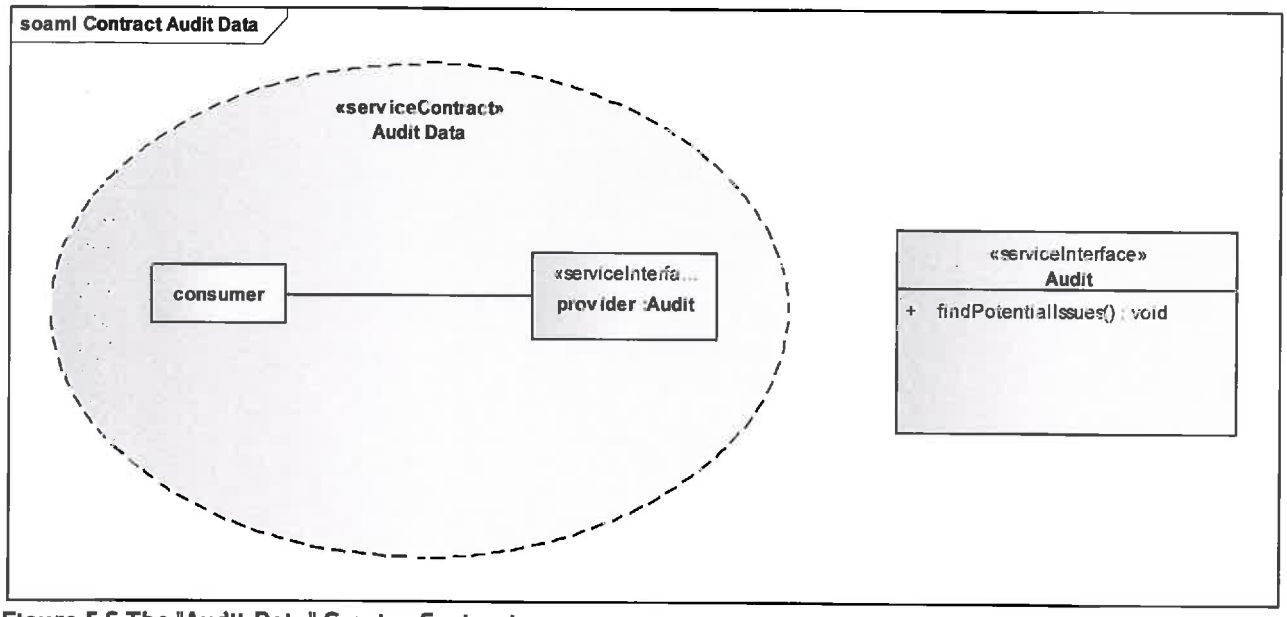


Figure 5.6 The "Audit Data" Service Contract

5.2.4 Manage Equipment

Figure 5.7 describes the "Manage Equipment". It allows the equipment vendor to describe new equipments, modify existing ones, or to remove old ones. This service is as well a "Single interface service" as no peculiar capacities are required from the equipment vendor.

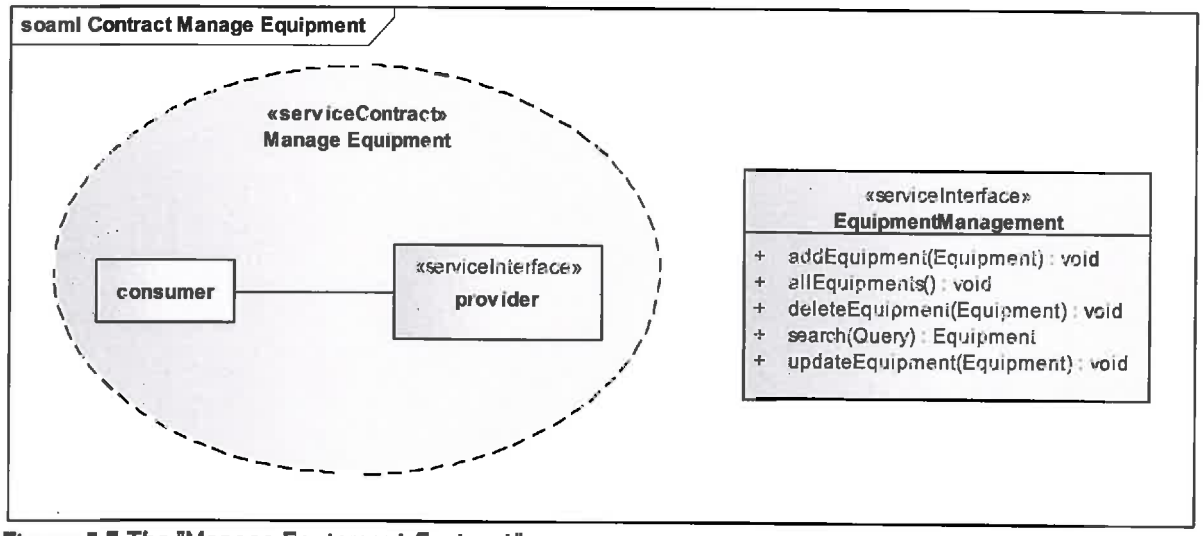


Figure 5.7 The "Manage Equipment Contract"

5.2.5 Manage SysLife Data

Figure 5.8 below illustrates the ManageSysLifeData service, which offers to the SysLife system's administrator a means to fix potential issues related to the data. This is a single "facade" service that provides the administrator (*i.e.*, The SysLife System operator) to access any other capabilities offered to others users.

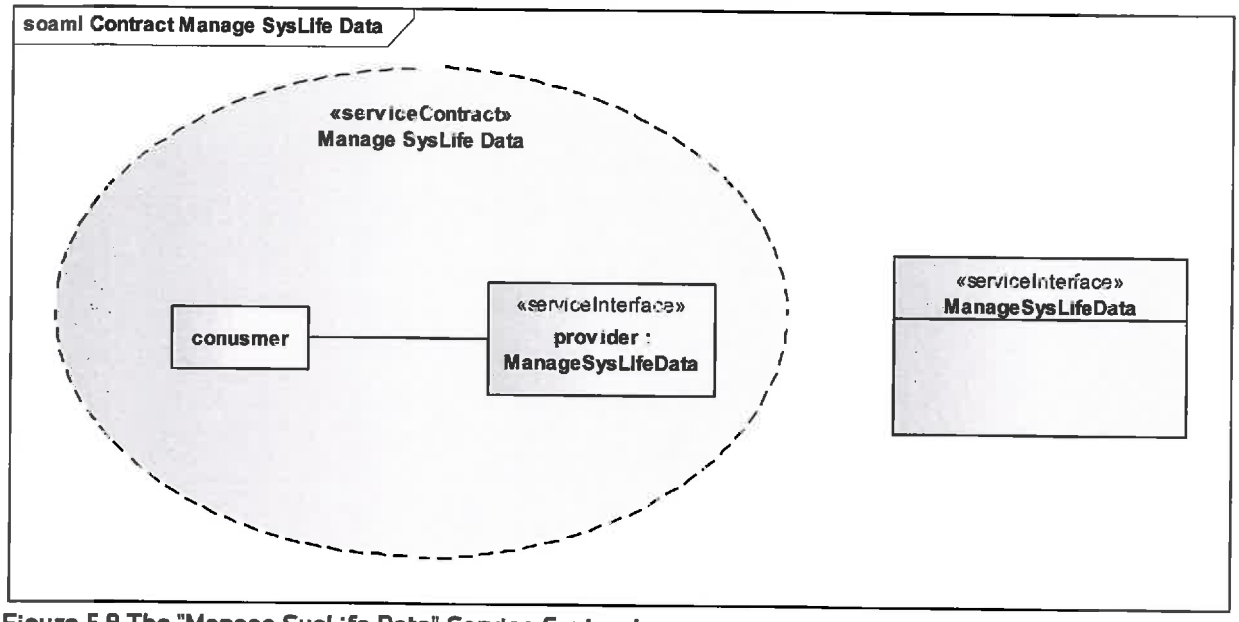


Figure 5.8 The "Manage SysLife Data" Service Contract

5.2.6 Register Raw Data

Figure 5.9 shows the *RegisterRawData* service that was a part of the service architecture. As one can see, the service consists of several operations for registering raw data into the SysLife system. This service is a collaboration between two roles: the provider of the service and the consumer. The service architecture diagram shows which participants fulfil these roles; the "SysLife system" acts as the provider while the "Maintenance data collector" (which can be either the "Sensor network" or "Maintenance operator").

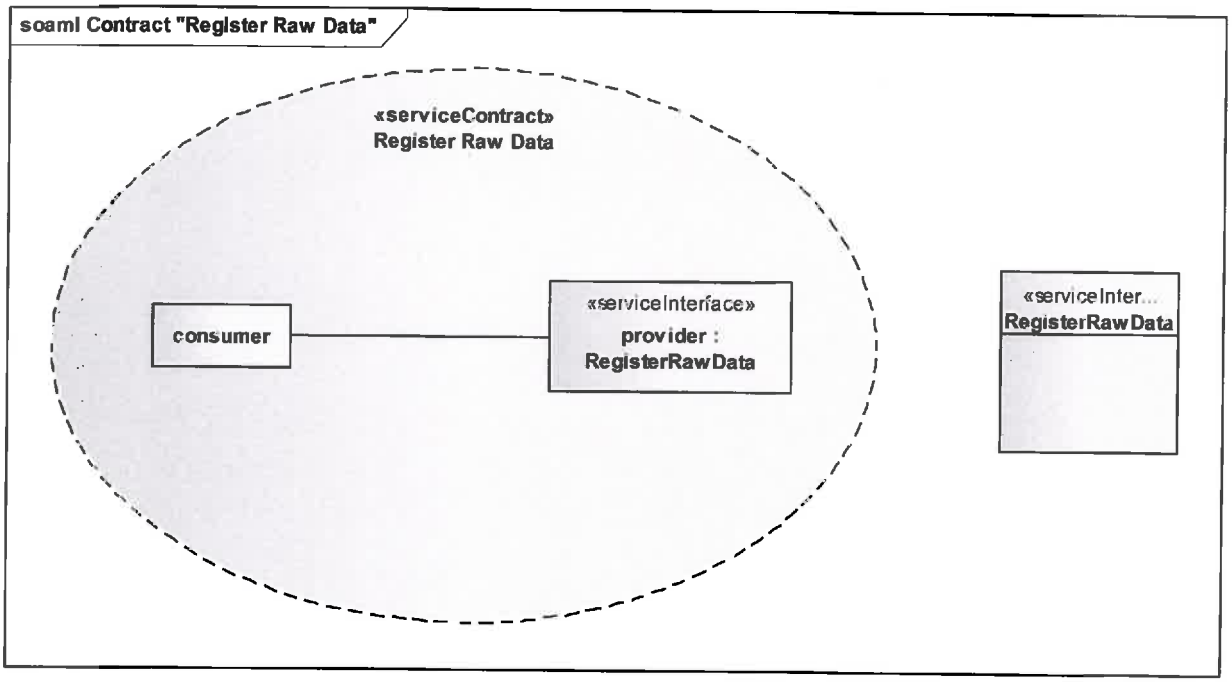


Figure 5.9 registerRawData service contract

6 SysLife System Data

6.1 IFD: International Framework of Dictionaries

IFD is a framework that enables the definition of ontologies, that is, a set of concepts and their relationships used to capture domain specific knowledge. IFD, and more generally ontologies, provide an effective means to deal with data interoperability issues, or interoperability issues that arise at the semantic level. In the *context of energy* for instance, an interoperability issue at the semantic level could be that two stakeholders are discussing about "turbines", but one thinks about a "Pelton turbine" whereas the other about a "Francis Turbine".

To avoid such interoperability issues, IFD advocates the creation of a common domain specific ontology that could serve as a basis to solve semantic issues. When two concepts or notions are to be compared, they must be matched with respect to their definition in the dictionary (i.e., the common ontology) rather than regarding their "local" names or meanings.

The IFD is based on a concept developed by the standards organisation ISO, notably in ISO 12006-3: 2007 (Building construction: Organization of information about construction works, Part 3: Framework for object-oriented information). Thanks to the dictionary, any data model can be linked to data from many sources, improving interoperability; furthermore it enables semantic analysis of the system at an early stage of the project. In its simplest form, IFD is a mechanism that allows for creation of multilingual dictionaries or ontologies. There is nothing in the ISO 12006-3 standard that limits it to building and construction, and the model as such can be used to describe most things.

As shown in the rich picture (see Figure 3.2), the SysLife system is the meeting point of various stakeholders. Avoiding semantic-issue is thus a key-point towards high-quality data in the SysLife system and, in turn, to reliable lifetime predictions.

At the technical level, IFD provides general unique identifiers (GUID) as keys towards the concepts. Therefore, it becomes possible to tag or label the SysLife concepts/data with the related semantics information within the IFD.

Although SysLife does not primarily aim at the definition of a standard ontology/dictionary for the energy domain, the SysLife dictionary will be a very interesting proposal for any standardization committee.

6.2 Business Data Model

Figure 6.1 shows the main data that are manipulated by the SysLife system. The three main parts that appeared on the rich picture (damage atlas, maintenance handbooks, and equipment) emerge as well here. The *damage atlas* is shown on the left, aside of the *maintenance handbooks*. In the middle the *Specific Equipment* class gather raw data, whereas the *Generic Equipment* and their associated *life curves* represent the so-called "*processed data*".

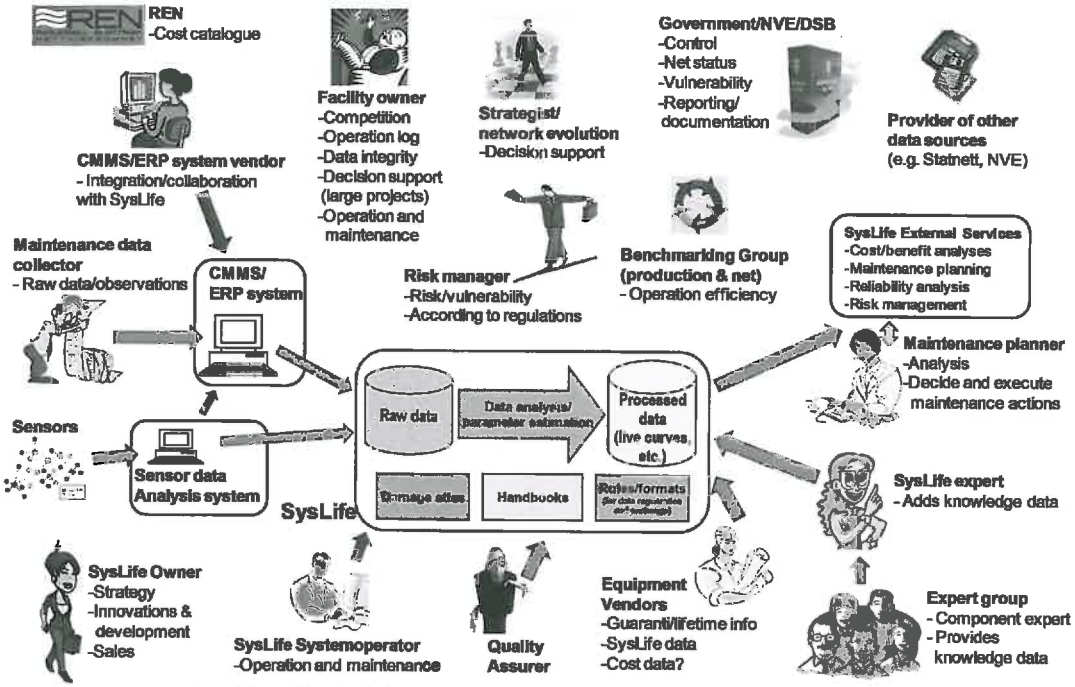


Figure 3.2 SysLife Rich Picture

It is worth to note that equipment is linked to an existing concept within the IFD, using a GUID (see the *ifd_guid* attribute in the *Equipment* Class: all subclasses inherit this attribute and can thus be linked to IFD definition consequently)

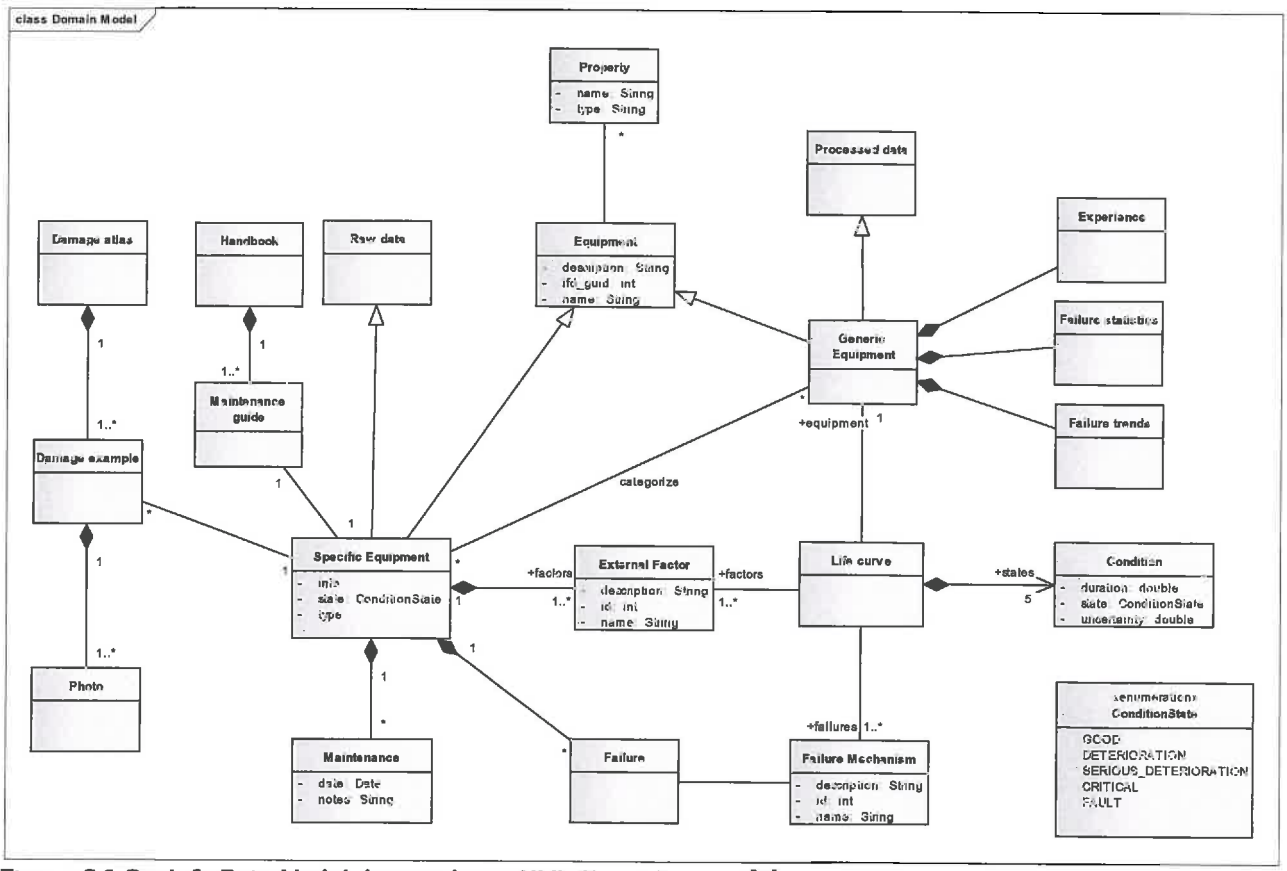


Figure 6.1 SysLife Data Model depicted as a UML Class diagram [3]

This initial version of the SysLife data model is used for the calculation of failure probabilities from the life curves that are specified for one or several failure mechanisms of a particular equipment. These probabilities are, so far, computed by the SVS service (a Matlab application) integrated into the tool for estimation of failure probability (available as an Excel sheet), which is part of the Optimal Maintenance Toolbox. Life curves are actually specified for a given component and failure mechanism as a set of parameters describing the duration and the associated uncertainty of the residence times in given quality states. For instance, a turbine life curve might specify that the turbine will be in a good shape for 5 years (+/- 1 year), and then have some minor deterioration for 2 years (+/- 6 months), before to suffer more serious issues during the following year (more or less 2 months), and eventually be in a critical state for the last 6 months before to be faulty. For each of the quality states ("1: As good as new or very good condition", "2: Minor deterioration", "3: Major deterioration", "4: Critical", and "5: Fault") the *Life Curve* class is associated with the duration and the uncertainty in the *Condition* class.

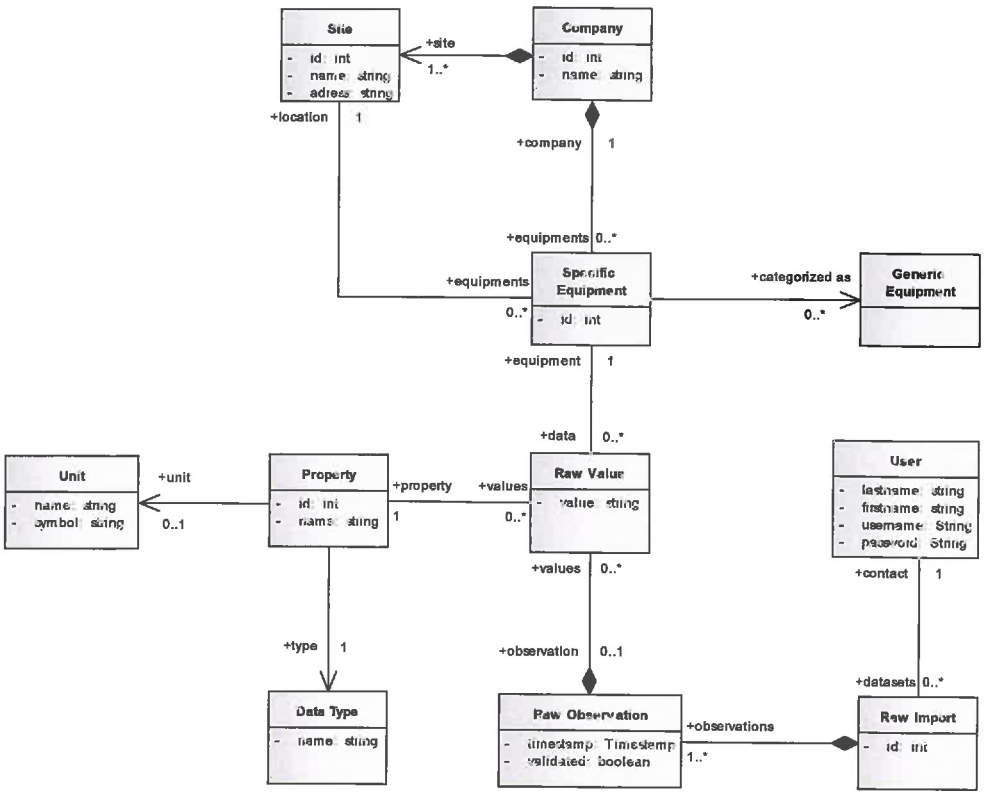


Figure 6.2 The "raw data" model, depicted as a UML class diagram [3]

Figure 6.2 above portrays the relational data structure used to store the information regarding concrete equipment (also called "raw data"), in contrast with the generic equipment represented by Figure 6.1. Raw data is organised around the concept of *Raw Import*, which represents a collection of field observations (see class *Raw Observation*). Each *Raw Observation* characterises a *Raw Equipment* by valuing a predefined set of properties. A prototypical observation would contain general equipment properties such as name, EBL code as well as time dependant properties such as corrosion level, observation methods, etc. Each *Raw Import* is placed under the responsibility of the particular user, which performed the initial data upload. The relationship between the "raw data" and the "generic data", possibly resulting from statistical analysis is shown by the association between *Raw Equipment* and *Equipment* (*cf.* Figure 6.1).

7 Detailed System Architecture

This section details the component-based architecture of the SysLife system. The idea beyond this component view is to describe how the different "parts" of the system will fit together. The partitioning of features between separate components will later lighten the maintenance and the evolution of the system.

The SysLife architecture, summarized by Figure 7.1 below, combines the services identified using the SoaML methodology [2] with a standard N-tiers architecture. N-tiers architecture is the *de facto* standard practice for large IT systems, and advocates the separation between the main IT concerns: namely information presentation, business logic, and data persistence. At the component-level, each of these concerns is encapsulated into a separated component, which can be independently developed or later evolved. In Figure 7.1, these three concerns emerged as follows:

- **Presentation:**
  The presentation layer is encapsulated into a single component called *web user interface* (on the right side of the figure). This component realizes the boundaries between the system and its users, regardless of the role they take. It ensures the proper presentation of the data to the user, and performs syntactic/logic validation of the data that the user provides. It's important to note that this component does not contain any business logic, but solely invokes services provided by the *business layer* (i.e., the component called *core services*). It will be basically implemented as a standard web portal, but can potentially be replaced by other technology, as long as the interfaces are preserved.
- **Business logic:**
  The business tier is represented by the components whose name ends with *logic*. These components expose various services (resulting from the SoaML [2]) methodology. In the current version of the architecture, these services are mainly connected to the presentation layer, but also to external systems such as sensors, CMMS systems, or additional optimal maintenance facilities. This business-tier relies on two main services: the persistence service in charge of data storage and the existing legacy SVS service. The later one is connected to the architecture through a wrapper component, which ensures the proper conversion and data exchange between the SysLife system and the legacy system.
- **Persistence:**
  The persistence layer, which is in charge of storing the data manipulated by the business services, is realized by the component named *persistence*. This component leverages two kinds of databases: a standard dictionary framework named IFD, which standardizes concepts used at the business level and a "regular" database, which stores the real data (referring concepts from the dictionary).

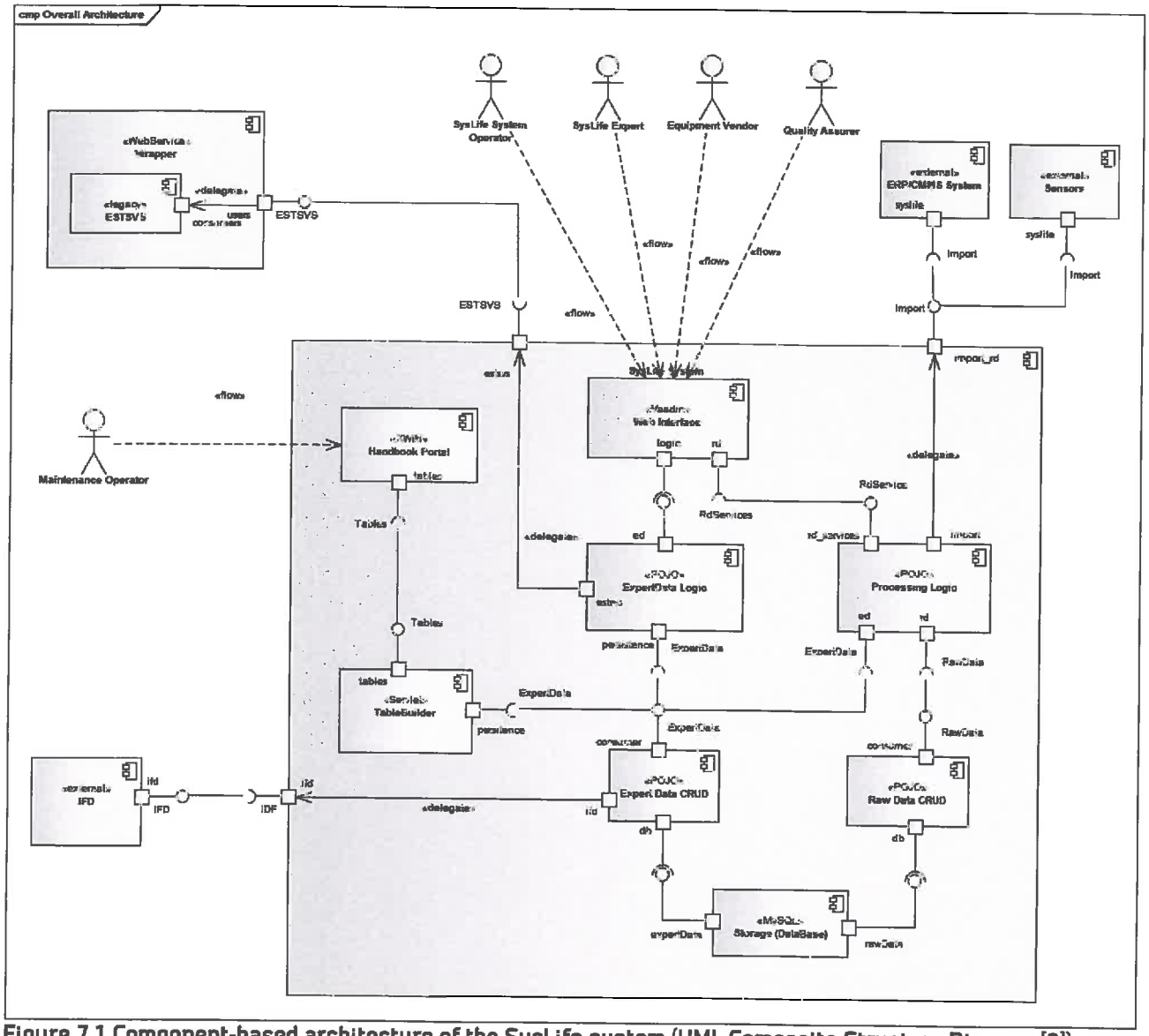


Figure 7.1 Component-based architecture of the SysLife system (UML Composite Structure Diagram [3])

By preserving a clean separation between concerns through services, this architecture eases future development and maintenance of the SysLife system. During the development, it allows different developers to work simultaneously on separate components that will be later easily assembled as they exhibit compliant interfaces. In the long run, such components can be replaced or updated independently as long as interfaces are preserved.

7.1 Presentation Layer

The SysLife presentation layer is supported by two keys components: an XWiki installation and the Syslife Vaadin [4] web interface. The following paragraphs detail the implementation and status of these two components.

XWiki is an open source wiki system implemented in Java whose main focus is to be easily extensible. In SysLife, XWiki mainly supports the retrieval of condition monitoring handbooks, as shown in Figure 7.2 and Figure 7.3. XWiki can be seen as an application framework, as pages behaviour may be control and extended using various scripting languages such as Ruby, Groovy or Python. This permitted to easily integrate the SysLife XWiki component with the Vaadin part.

Vaadin is a framework that supports the development of Rich Internet Application. In contrast with other solutions, Vaadin developers write plain Java code, which runs on the server side (i.e., not in the browser of the user) reusing as many as possible the existing customisable UI components (e.g., buttons, tables, menus, checkboxes). Interested readers may find a detailed description of the architecture of Vaadin applications in the Book of Vaadin [4, Chapter 4]. **Error! Reference source not found.** below summarises the status of the thirteen use cases identified in Section 4.1 (*cf.* page 15).

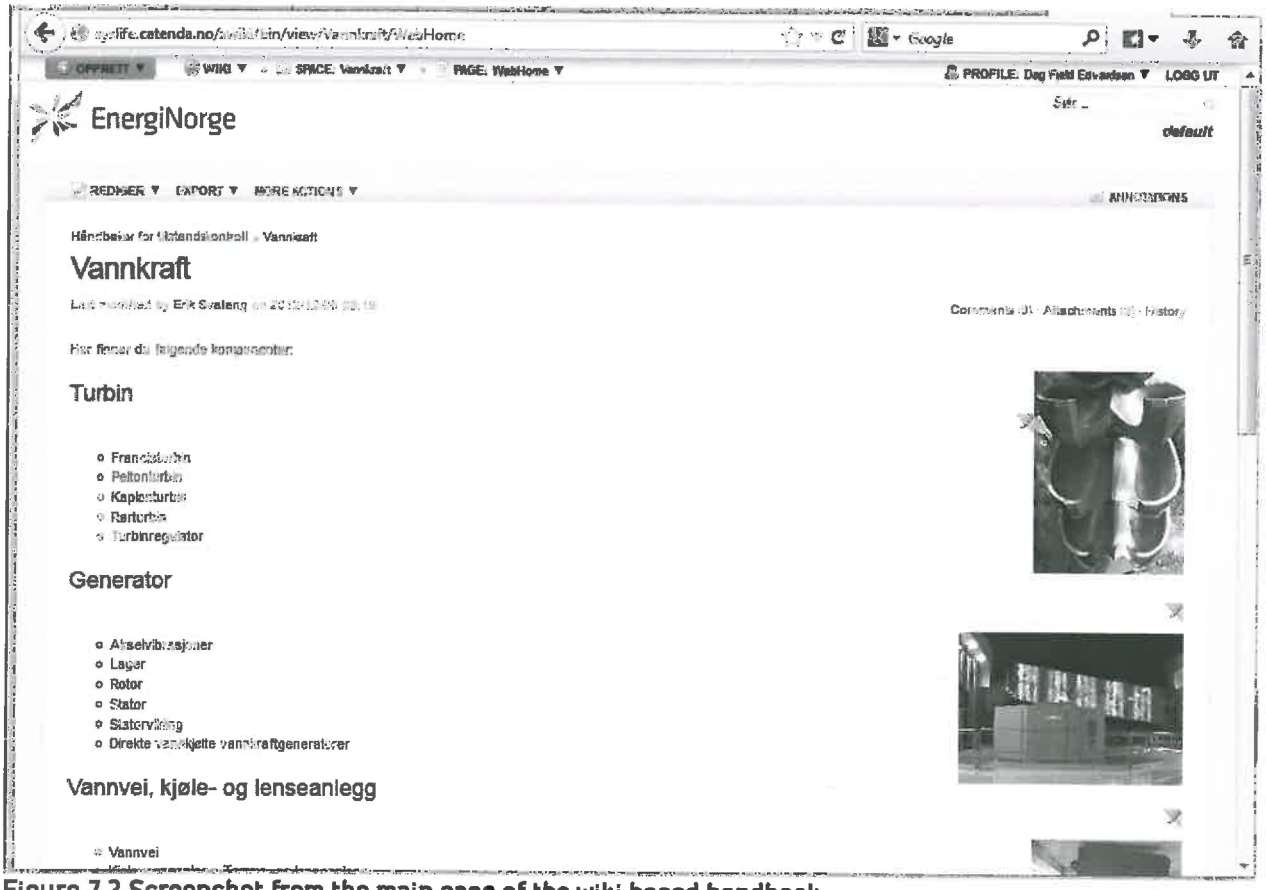


Figure 7.2 Screenshot from the main page of the wiki based handbook

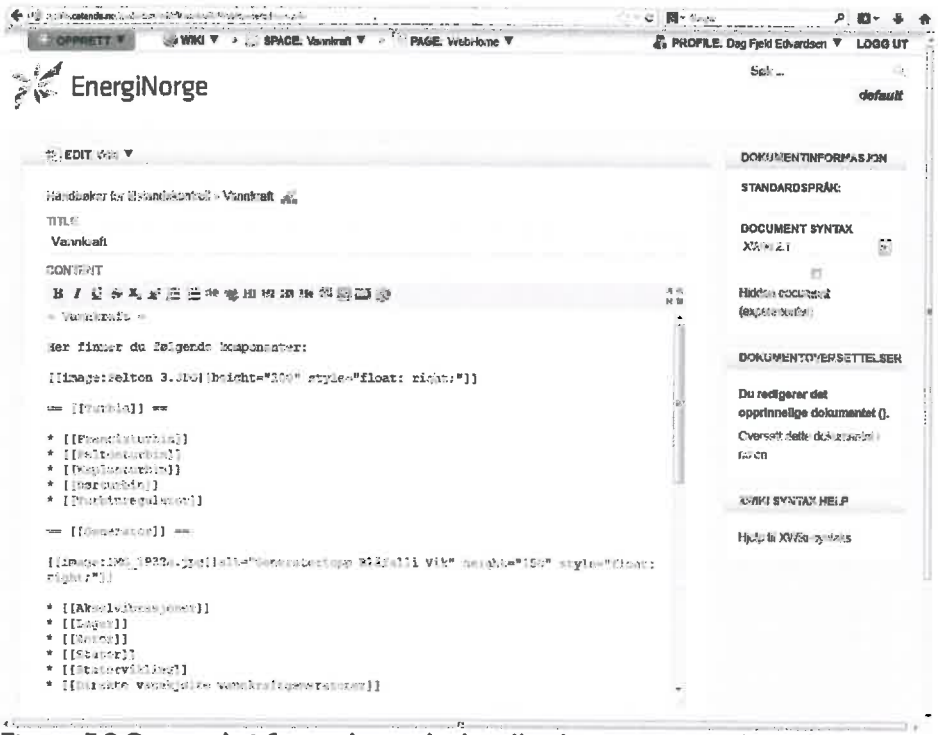


Figure 7.3 Screenshot from editing the handbooks using a wiki based syntax

As the figure above shows, the syntax for creating the content is wiki markup (xwiki syntax 2.1). This makes it easy to add to and maintain the content.

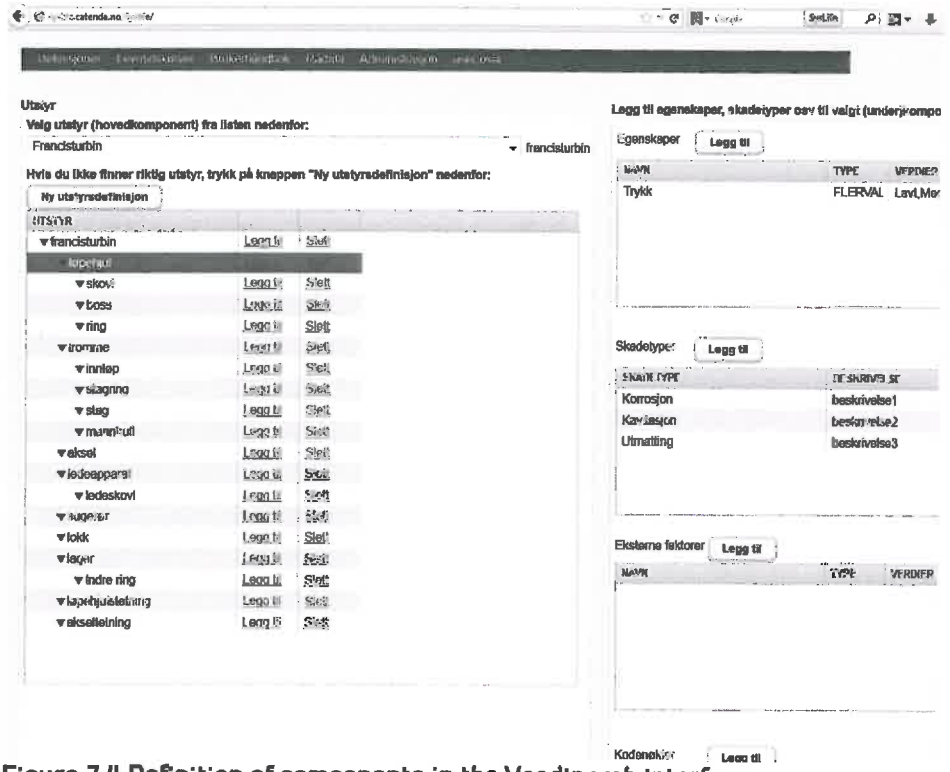


Figure 7.4 Definition of components in the Vaadin web interface

Figure 7.4 shows a screenshot from the Vaadin based application where the user is defining a component (in this case a Francis turbine).

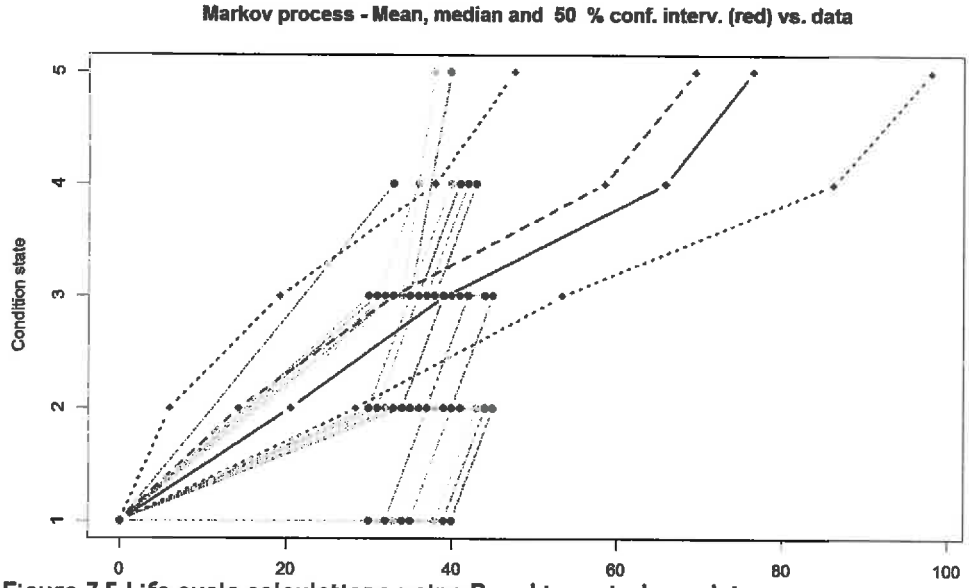


Figure 7.5 Life cycle calculations using R and imported raw data

Using the Vaadin based web application, the user can use "raw data" imported into the system to run R-based statistical estimation of life cycle parameters.

Table 7.1 The SysLife use-cases status in the presentation layer

Use Case	Description	Access Point	Status
UC 01	Transfer Raw Data	Vaadin Web Interface	Complete
UC 02	Security And Anonymity	N.A.	Complete
UC 03	Provide Condition Monitoring Handbooks	XWiki	Complete
UC 04	Register Damage Atlas Data	XWiki	Complete
UC 05	Register Equipment Data	Vaadin Web Interface	Complete
UC 06	Audit	N.A	**To be developed**
UC 07	Search Data	Vaadin Web Interface	Complete
UC 08	Register Expert Knowledge	Vaadin Web Interface	Complete
UC 09	Processing and Analysis	Vaadin Web Interface	Complete

UC 10	Create Generic LifeTime data	Vaadin Web Interface	Complete
UC 11	Manage data	Vaadin Web Interface	Complete
UC 12	Retrieve Generic LifeTime data	Vaadin Web Interface	Complete
UC 13	SysLife System Operation		**To be developed**

7.2 Business Logic

The business logic of the SysLife system is implemented as plain Java code running on the server side, hence directly integrated with the Vaadin Web Interface.

The Use-case 02, namely Security and Anonymity has been the object of a separate investigation within the SysLife project [5] and concludes that the solution that is implemented is satisfactory. There is no tight integration with the power companies' computer systems, and the "raw data" is imported into the Syslife system by uploading CSV files. The anonymity control is based on the facts that (1) only people given user credentials have access, and (2) a user only has access to data uploaded by users from his/her own company or flagged as "public" (meaning available to other Syslife users).

The key business logic of the SysLife system is the extraction of life-time data from the available "raw" data (*cf.* UC 10 in **Error! Reference source not found.**). This is made through R scripts [6], implementing statistical analysis. Today, there are 4 different statistical routines implemented in R. These are referred to as Markov, Gamma, MTFT, and descriptive statistics. They are described in detail in the user manual. It is relatively simple to add new R-based calculations (in addition to creating a new R-script this requires adding the required parameters to the database and adapting a simple java class).

7.3 Persistence Layer

The storage solution selected for the SysLife system is a MySQL data-base. The persistence layer, which manages interactions between the business logic and the storage, is automatically generated from the data-based itself, by the DBManager: a tool developed in-house at Catenda. Given a database schema, the DBManager automatically generates a set of Java classes, which let the programmer manipulate the data as Java objects, and hence seamlessly integrates the business logic with the persistence layer.

The database layer is created using a database wrapper ("DB-wrapper" created by Catenda) that creates java classes from MySQL schemas. This makes it easier to create and maintain the persistence layer.

In order to make it easier in a possible future to integrate with other systems the definition of hardware components include the possibility to make references to IFD [7] (Industry Foundation Classes also known as buildingSMART data dictionary). IFD is a semantic system where one can define terms and define how they relate to other terms (one thing can be composed of other things or be a subtype of other things), and there mechanisms for synonyms and translations into other languages.

8 Evaluation and Assessment

This section summarizes the initial qualitative evaluation of the SysLife system architecture that was jointly carried out by SINTEF ICT and Catenda, during the fall 2012. The objectives, which are elaborated in the following sections, were threefold:

- Formalizing the key requirements underlying the development of the SysLife system;
- Conducting a preliminary evaluation of the architecture, and of the main design or development decisions, so as to identify possible deviations or flaws;
- Suggesting recommendations to guide the further development and/or mitigate possible deviations.

8.1 Evaluation Criteria

We identified three major dimensions, along which the SysLife architecture and the related design decisions were evaluated, namely usability, flexibility, and security. *Usability* captures the extent to which the architecture fits the initial SysLife purpose, and also includes performance. *Flexibility* describes to which extent the SysLife system can be adjusted to unforeseen requirements, such as new features or significant increase of its number of users. *Security* covers several of the security concerns expressed by stakeholders, especially regarding privacy and resilience. The table below further refines these key concepts and classify them according to their importance[4], ranging from negligible to critical.

Table 8.1 Classification of the key evaluation criteria according to their importance

	Name	Description	Importance Factor
Usability	Functionality	Reflect to which extent the system works as it is intended to do	Major
	Configurability	Reflect to which extent the user can configure the system so as she can complete her tasks in an easy way	Moderate
	Performance	Reflect to which extent the system does complete its tasks, including the readiness, continuity, and accuracy of the service	Minor
Flexibility	Coupling & Cohesion	Reflect to which extend the system is properly divided into independent components, and to which extent these components are logical, respectively	Major
	Evolutivity	Reflect to which extent the system can changed so as to integrate new features or components	Moderate
	Scalability	Evaluate to which extent the system can be exposed to alternative usage profiles, especially regarding the number of user and throughput of data.	Moderate
Security	Anonymity	Evaluate to which extent a malicious user can identify SysLife user from the the SysLife data	Critical
	Resilience	Evaluate to which extent the SysLife system is able to recover from failures, both hardware or software, or malicious attacks.	Major

[4] To give an order of magnitude of an importance factor, they can be characterise integers ranging over [0, 100], such as critical (~ 100); major (> 65), moderate (~ 50), minor (> 35), negligible (~ 0)

8.2 Architecture Evaluation

This qualitative evaluation encompasses two faces of the SysLife systems, namely its architecture, and the associated work processes. The architecture encompasses the way the various parts of the SysLife system are organized and composed together, whereas processes covers the various workflows which drive the user activities.

This qualitative evaluation associates with each dimension a qualitative score, reflected by labels as "unsuited", "poor", "fair", "good" or "excellent", which can then be converted into quantitative scores ranging from over [0, 100]. The overall evaluation is then obtained as a weighted average of these scores, where the weights correspond to the importance factor associated with the aforementioned evaluation criteria.

Usability

- Functionality / Data Structure
  - The system addresses all its key requirements, through specific developments or through the integration/customization of third party systems, such as the XWiki platform used as a user interface for the handbook visualization.
- Configurability
  - By being broken down into independent pieces, the system gain in configurability as each component can be separately configured. For instance, the XWiki system, which provides access to handbooks, exhibits various configuration parameters, whose values do not impact the rest of the system.
- Performance
  - In the initial stage, performance of the system have not been assessed aside of some regular functional tests. However, as such SysLife is expected to face high workloads.

Flexibility

- Coupling & Cohesion
  - The coupling and cohesion of the system is very good. The architecture follows as classical 3-tiers architecture, which enforces a clear separation between concerns. The presentation layer is implemented in a Vaadin framework, whereas the core functionalities are implemented as pure Java objects, which access an abstract storage layer.
- Evolutivity
  - Evolutivity is high, as a result of the low coupling and high cohesion. However, the monolithic implementation in Java used in the current version does not permit to redeploy part of the application dynamically. It is therefore needed to stop the system to update any of its components. Alternative platform could be used to further lighten maintenance and evolution
- Scalability
  - The scalability of the application is fair, but hindered by its monolithic implementation. Although the SysLife system is not expected to face massive flows of users, a distributed implementation could help parallelising it.
  - The use of separated data layer, permit to easily scale up the storage capacities of SysLife system, although the current version uses a single storage.

Security

- Anonymity & Authorization
  - Good, risk analysis where conducted to reduce the risk [5]. It revealed that, from the user standpoint, any leakage of confidential data to competitors and other SysLife-users, would damage their trust in the system. In the final system, the user is therefore given a finer control over the visibility of the data she pushes in the SysLife system. In addition, it was made clear from the companies that a tight integration between the SysLife system and their CMMS system

would not be acceptable. Thus, only a loose coupling, by data file exchange, has been implemented.
    - Quid of the encryption of communication and authentication.
- Resilience
    - The proposed architecture does not include mechanisms to replicate the data base, or alternatively to back-up the underlying storage. This reduces the resilience of the data in case or hardware failure especially, and could lead to loss of data in the worst cases.
    - The centralized storage that is currently in use represents also a single point of failure, which directly threatens the overall resilience of the system. An alternative could be to distinguish between raw and expert data, and to use separate storages facilities.

Table 8.2 Qualitative evaluation of the developed solutions of the key criterions

	Name	Importance Factor	Qualitative Evaluation
Usability	Functionality	Major	Good
	Configurability	Moderate	Good
	Performance	Minor	N-A
Flexibility	Coupling & Cohesion	Major	Good
	Evolutivity	Moderate	Fair
	Scalability	Moderate	Fair
Security	Anonymity	Critical	Good
	Resilience	Major	Fair

8.3 Evaluation Summary and Recommendations

As shown in Table 8.2 , which summarizes the evaluation of the overall architecture, the Syslife System fits the initial system requirements. This preliminary evaluation resulted in three major recommendations, which could enhance the SysLife System.

- **Modern Development Setup.** There has been significant progress regarding software engineering tools to package, deploy and manage dependencies in the past years. Tools such as Maven, Ivy, SBT enables to properly built, package, deploy and distribute complex applications. Adopting such tools in the SysLife development could significantly foster the development and the adoption of the SysLife system.
- **Isolating vendor-specific technologies.** Refactoring the application to isolate as much as possible technical choices. This partially done with the clear separation between, the business logic and the persistence layer. From a technical perspective, the architecture would benefit from a similar separation between the Vaadin technology used to build the graphical user interface, and core business services, which capture the SysLife work processes.
- **Moving towards Distribution.** Gradually move to a distributed architecture, where different features are isolated into separated services, which realization can be distributed over the network. This would foster cohesion and, in turn, avoid a simple point of failure of having a single execution node for the whole system.

9 Conclusion

This document provides an overview of the architecture of the SysLife system and can be used either by software engineers or by business decision makers.

The main design decisions and the subsequent refinements that were made during the SysLife project are surveyed in the document. Both the main actors and their responsibilities are described in the business architecture model. Furthermore, the requirements formalized as UML use cases are presented. The document shows the key services identified using the SoaML methodology and how they are realized by software components. Finally, a qualitative evaluation of the design decisions discusses candidate improvements of the SysLife system.

References

1. Thomas Welte, Arnor Solberg, Jørn Heggset, Eivind Solvang, Arnt Ove Eggen, Vegard Dehlen. *System for Lifetime Related Data (SysLife).* report no. 327-2011, Energy Norway, Oslo.
2. Object Management Group (OMG). *Service oriented architecture Modeling Language (SoaML) - Specification for the UML Profile and Metamodel for Services (UPMS) SoaML.* ptc/2009-12-09. 2009. (see http://www.omg.org/spec/SoaML/)
3. Object Management Group (OMG). *Unified Modeling Language – Superstructure* (v2.4.1). formal/2011-08-06. 2011. (see http://www.uml.org/)
4. Vaadin Ldt. *The Book of Vaadin (Vaadin Framework 6.7.0).* Marko Grönroos editor, 4th edition, 1st revision. Vaadin Ltd. August 2012 (available at https://vaadin.com/download/book-of-vaadin/vaadin-6/pdf/book-of-vaadin.pdf)
5. Arnor Solberg, Dag Fjeld Edvardson, Thomas Welte and Franck Chauvel. *SysLife: Security and Risks.* SINTEF technical report. 2013-12-22.
6. Rob Kabacoff. *R in Action.* Manning, 2010.
7. Industry Foundation Classes (IFD) / buildingSMART Data Dictionary (bsDD). See: http:// http://www.ifd-library.org

Technology for a better society

www.sintef.no