Antti Tuomi

Application integration for condition based maintenance

Faculty of Electronics, Communications and Automation

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 12.5.2010

Thesis supervisor:

Prof. Jukka Manner

Thesis instructor:

Dr.Sc. (Tech.) Ilkka Seilonen

Aalto University
School of Science
and Technology

AALTO UNIVERSITY
SCHOOL OF SCIENCE AND TECHNOLOGY

ABSTRACT OF THE
MASTER'S THESIS

Author:	Antti Tuomi
Title:	Application integration for condition monitoring
Date:	May 12th, 2010 **Number of pages:** 8 + 51
Faculty:	Faculty of Electronics, Communications and Automation
Department:	Department of Communications and Networking
Professorship:	S-38 Networking Technology
Supervisor:	Prof. Jukka Manner
Instructor:	Ilkka Seilonen, Dr.Sc.(Tech.)

Contemporary maintenance processes particulary in condition based maintenance are very information-intensive. However, the information is usually divided to several information systems that are not interconnected. OPC Unified Architecture is a new protocol expected to bridge the gap between the service-oriented enterprise IT systems and the automation systems.

This work studies the feasibility of OPC Unified Architecture to enable connectivity of intelligent devices and condition monitoring systems. A test platform, consisting of a condition monitoring system (Metso FieldCare) and an enterprise asset management system (CalemEAM), is implemented and a design to connect the systems is presented. A part of the design is implemented on the test platform with the .NET framework and the implementation is validated with processes of condition based maintenance. MIMOSA OSA-EAI and OpenO&M Common Interoperability Registry specifications were applied in the implementation.

The results suggest that OPC Unified Architecture and in particular its information modelling capabilities offer benefits over the older technologies. However, the existing information models are insufficient for presenting condition data. If the information modelling capabilities are not utilised, they remain a source of unnecessary complexity.

Keywords: application integration, condition based maintenance, OPC UA, SOA, MIMOSA OSA-EAI

AALTO-YLIOPISTO
TEKNILLINEN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä:	Antti Tuomi		
Työn nimi:	Sovellusintegraatio kuntoon perustuvassa kunnossapidossa		
Päivämäärä:	12.5.2010	**Sivuja:**	8 + 51
Tiedekunta:	Elektroniikan, tietoliikenteen ja automaation tiedekunta		
Laitos:	Tietoliikenne- ja tietoverkkotekniikan laitos		
Professuuri:	S-38 Tietoverkkotekniikka		
Työn valvoja:	Prof. Jukka Manner		
Työn ohjaaja:	TkT Ilkka Seilonen		

Nykyaikaiset kunnossapitomenetelmät erityisesti kuntoon perustuvassa kunnossapidossa ovat hyvin tietointensiivisiä. Tieto on kuitenkin usein jaettu tietojärjestelmiin, jotka eivät ole yhteydessä toisiinsa. OPC Unified Architecture on uusi yhteyskäytäntö jonka odotetaan kaventavan palvelukeskeisten tietojärjestelmien ja automaatiojärjestelmien välistä kuilua.

Tämä työ tutkii OPC Unified Architecturen soveltuvuutta älykkäiden toimilaitteiden ja kunnonvalvontajärjestelmien yhdistämiseen. Työssä rakennetaan koeympäristö, joka koostuu kunnonvalvontajärjestelmästä (Metso FieldCare) ja kunnossapitojärjestelmästä (CalemEAM), ja suunnitelma näiden yhdistämiseen esitellään. Osa suunnitelmasta toteutetaan .NET sovelluskehyksellä ja toteutus testataan kuntoon perustuvan kunnossapidon prosesseilla. Toteutuksessa sovelletaan MIMOSA OSA-EAI ja OpenO&M Common Interoperability Registry-spesifikaatioita.

Tulosten perusteella voidaan sanoa, että OPC Unified Architecture ja erityisesti sen tiedonmallintamisominaisuudet ovat hyödyllisiä verrattuna tämän hetkisiin tekniikoihin. Olemassaolevat tietomallit eivät kuitenkaan riitä kuntotiedon mallintamiseen. Jos mallintamisominaisuuksia ei käytetä, niiden vaikutus on vain spesifikaatiota monimutkaistava.

Avainsanat: sovellusintegraatio, kuntoon perustuva kunnossapito, OPC UA, SOA, MIMOSA OSA-EAI

Preface

This work was done in the Information and Computer systems in Automation research group. I would like to thank my colleagues, my supervisor prof. Manner and my instructor Dr. Seilonen for their guidance.

Otaniemi, 12.5.2010

Antti Tuomi

Contents

Abbreviations

ADI	Analyser Device Information model (of OPC UA)
API	application programming interface
CBM	condition based maintenance
CIR	Common Interoperability Registry
CM	condition monitoring
CRIS	Common Relational Information Schema
DCOM	Distributed Component Object Model
DI	Device Information model (of OPC UA)
DPWS	Devices Profile for Web Services
DTM	Device Type Manager (in FDT)
EAM	enterprise asset management
EDDL	Electronic Device Description Language
ERP	enterprise resource planning
FDI	Field Device Integration
FDT	Field Device Tool
GUID	Globally Unique Identifier
MES	manufacturing execution system
O&M	operations and maintenance
OPC UA	OPC Unified Architecture
OSA-CBM	Open Systems Architecture for Condition-Based Maintenance
OSA-EAI	Open Systems Architecture for Enterprise Application Integration
PCS	process control system
PLM	product lifecycle management
SDK	software development kit
SOA	service-oriented architecture
SQL	Structured Query Language
UML	Unified Modeling Language
WCF	Windows Communication Foundation
WSDL	Web Service Definition Language
XSL	Extensible Stylesheet Language
XSLT	XSL Transformations

List of Figures

1 Introduction

The highly integrated production processes of contemporary manufacturing depend on the reliability of the involved assets. Advances in maintenance have enabled the use of sophisticated data-intensive methods for assessing the health and maintenance need of production equipment. However, the data for this is usually stored in separate IT systems that do not communicate with each other. Integrating these data sources could better support the data-intensive maintenance processes.

The service-oriented architecture (SOA) paradigm is commonly used in the domain of enterprise application integration. However, applying it in the integration of devices is rare. OPC Unified Architecture (OPC UA) is a new protocol that is expected to bridge this gap between the service-oriented enterprise IT systems and the automation and control systems, including intelligent devices. These two technologies are expected to play a major role in application integration in manufacturing operations management.

This thesis was written as a part of the POJo-project, which is part of the second work package of the EffTech research program of Forestcluster Ltd. Integration of IT systems involved in operations and maintenance was a part of the research project.

1.1 Objectives, research questions, research methods

SOA has already established itself as the dominating enterprise integration paradigm. OPC UA was designed with this in mind and is thus meant to be applicable in SOA-based IT architectures. OPC UA and especially its capability to present domain-specific information models is expected to facilitate the integration of domain-specific data in a standard manner. However, there has been little research in applying both OPC UA and SOA in condition based maintenance.

The main research problem of this thesis is to find out a suitable design for the integration of IT systems involved in condition based maintenance with SOA and OPC UA. In particular, how does OPC UA facilitate the integration of devices with SOA-oriented IT systems, what are the properties of the design and what other standards, if any, are required to implement the design.

These problems are approached by developing an experimental design for the application integration. The requirements for the design are drawn by examining the processes of condition based maintenance. The design is validated with an experimental implementation which is tested with use cases identified in requirements analysis.

The scope of this thesis is limited to vertical integration in condition based maintenance. Particularly the work focuses on the integration of enterprise asset management and condition monitoring systems, even though the test platform contains also other systems. The test platform is constructed so that it can be used to evaluate the feasibility of the design, but not its scalability and performance properties.

1.2 Results

In this work, a description of a feasible design for integrating OPC UA-based condition monitoring systems to SOA-oriented IT systems is presented. To evaluate the design, a test platform is constructed using commercial off-the-shelf IT systems. Because the systems used do not support the interfaces that are to be evaluated, such interfaces are implemented separately. The test platform that can be used to evaluate the selected test platforms is described. The design is validated with an experimental implementation. The implementation is demonstrated to be able to supply the data necessary to support use cases in the condition based maintenance process.

1.3 Outline of this thesis

The first two chapters present the application domain by describing the relevant standards and related research. The second chapter presents the domain of maintenance and its relation to other activities in the enterprise. In addition the IT systems used to support the maintenance processes, especially condition based maintenance, are described. The third chapter discusses application integration in maintenance operations management. Application integration in general is presented following with an overview of application integration standards (including OPC UA) used in maintenance.

Chapter 4 describes the requirements and a design that covers them. The chapter follows the methodology presented in the IEEE 1471 standard. The stakeholders, their concerns and needs are identified to draw the requirements of the design. Finally, a design that covers the requirements and its rationale is presented.

The fifth chapter presents the test platform and the experimental implementation for evaluating the viability of the design. First, the actual IT systems, their capabilities and implemented extensions of the test platform are described. The description of the implemented parts of the design follows along with the test cases used to evaluate the implementation. Finally, the sixth chapter presents the conclusions, discussion and proposals for future work.

2 Maintenance

This chapter presents the domain of maintenance. First, the importance of maintenance and the different types of it are presented with a focus on condition based maintenance. Then, the maintenance activities are positioned in relation to other activities in a manufacturing enterprise. Finally, the most important IT systems that support the maintenance processes are presented with two examples.

2.1 Overview

With the highly integrated production processes and tight schedules used in manufacturing today, it is essential that equipment is reliable and efficient. Maintenance keeps the equipment operating at optimal capacity. A more formal, all-encompassing definition can be found in the European standard EN 13306, where maintenance is defined as “combination of all technical, administrative and managerial actions during the life cycle of an item intended to retain it in, or restore it to, a state in which it can perform the required function” [1].

Originally maintenance mostly involved lubrication, calibration and cleaning of the equipment. Equipment was repaired after a failure. Because equipment was usually designed with larger margins of error, breakdowns were rare. Production equipment was simple and repairs were easy. Because the production processes were not highly integrated, a single equipment failure had limited effect on production. [9]

The second world war had a large impact on production processes. To meet the larger production requirements of war industry, the level of automation was increased and production processes were integrated more tightly. Equipment used in production were connected to each other to form longer production lines. Because of these changes, breakdowns started to have more serious consequences. A single equipment failure could halt the entire production line. Preventive maintenance processes, mostly scheduled equipment overhauls, started emerging to reduce the number of breakdowns. [9]

Nowadays production processes are highly integrated, production equipment is expensive and competition is global. On the other hand, we have better tools for detecting impending equipment failures. Maintenance processes used in the industry today reflect this change. In addition to the older processes of corrective and scheduled maintenance, there are new processes in use today. [9]

In EN 13306, the maintenance processes are divided to different categories that are illustrated in Figure 1. The two main categories are preventive maintenance, which is done before a fault is detected, and corrective maintenance that is done afterwards. Preventive maintenance is further divided to condition based maintenance and predetermined maintenance. Unlike condition based maintenance, predetermined maintenance is done without a previous condition investigation [1]. ISA-95 adds an additional category that includes maintenance that is related to resource performance and efficiency optimisation [2].

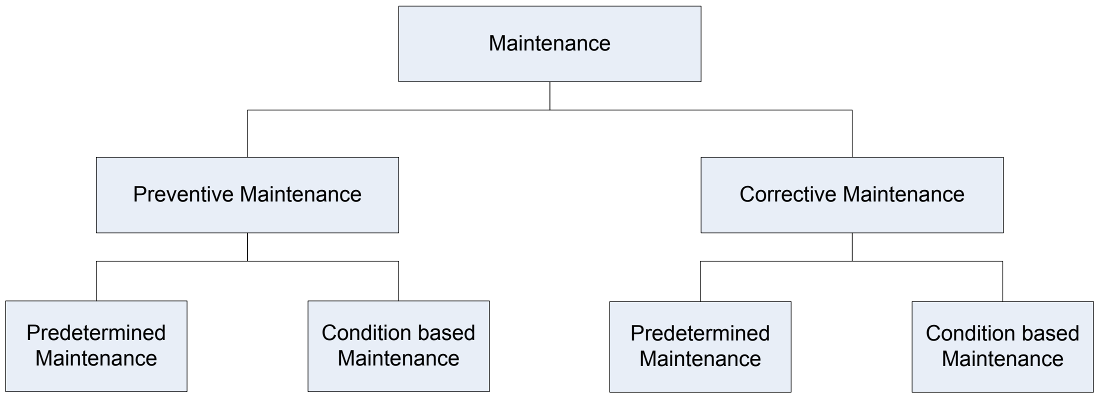

Maintenance
Preventive Maintenance
Corrective Maintenance
Predetermined Maintenance
Condition based Maintenance
Predetermined Maintenance
Condition based Maintenance

Figure 1: Types of maintenance adapted from [1].

Preventive maintenance makes it possible to raise the reliability level of the production equipment considerably. Also maintenance planning that is enabled by effective preventive maintenance can have a major economic impact. If the maintenance work can be planned in advance, the production schedule, orders for required spare parts and so on can all be organised properly. However, the costs of preventive maintenance rise dramatically if the attempted level of reliability is set too high. The maintenance organisation should decide what is the optimal level of preventive maintenance taking into account the effects of equipment failure and the efficiency of preventive maintenance. [10]

2.1.1 Condition based maintenance

The intent of condition based maintenance is to reduce the probability of failure and prevent irreversible damage to the equipment [1]. The effects that cause the equipment failure are called the failure modes and include changes such as material fractures and leaks. The conditions of the machine that contribute to a failure mode are called the failure causes. Failure causes include normal wear, improper operation and defects in the equipment. [11]

In simple equipment, time dependant failure modes dominate. However, more complex equipment have more complex failure modes. It is insufficient to rely on regularly scheduled equipment overhauls to keep such equipment reliable. Knowing the failure modes of the equipment and their causes makes it possible to detect impending failures with methods such as vibration analysis and oil analysis. [11]

The deterioration of the equipment can be visualised with the Potential Failure (P-F) curve, illustrated in Figure 2. The curve has two important points, the potential for failure point (P) and the failure point (F). After the point P, the impending failure can be detected and at the point F, the equipment fails. The time difference between F and P defines the feasibility of condition based maintenance strategy. If there is no time to react to the failure, then condition based maintenance can not be used. With contemporary analysis methods, the impending failure can be detected

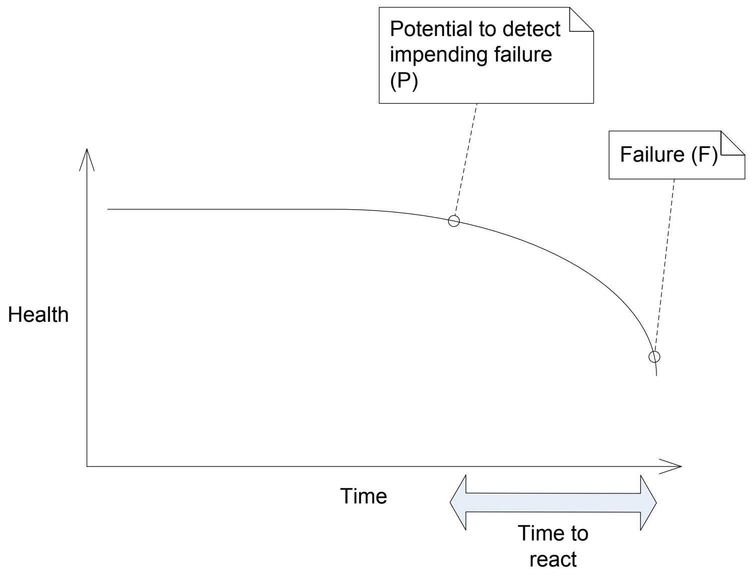


Figure 2: P-F curve

earlier thus increasing the time interval between the detection and failure. Better analysis methods essentially increase the time to react to the equipment failure. [12]

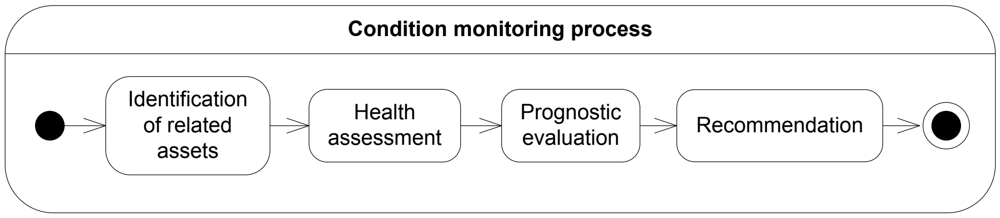

Condition monitoring process
Identification of related assets
Health assessment
Prognostic evaluation
Recommendation

Figure 3: Condition monitoring process

The process that starts when an abnormal condition is detected is illustrated in Figure 3. The process is based on the workflow described by the ISO 13374 standard, which will be examined in more detail in the section 2.3.2. ISO 13374 is a standard that defines the data transfers and interfaces of an automated condition monitoring system. First the maintenance engineer needs to identify what assets are involved with the abnormal condition. When the assets are known, the engineer assesses the health of the involved equipment. This process can involve methods such as lubrication analysis, vibration analysis and thermographic analysis. [3]

When the current health is known, the maintenance engineer creates a prognosis for the equipment. Prior knowledge about the equipment and assumptions on the future use are useful for this process. Based on the prognosis, the maintenance engineer can propose what should be done next, for example should the equipment be repaired or replaced immediately or is it possible to defer the maintenance work. [3]

2.2 Maintenance operations management

The relation of condition based maintenance to other functions performed by an enterprise involved in manufacturing can be understood with the framework provided by the ISA-95 standard. ISA-95 identifies two domains in a manufacturing enterprise and the boundary between them: the enterprise domain and the manufacturing operations and control domain. These domains are further divided to a hierarchy of 5 levels, where the lowest level 0 is the actual physical process and the highest level 4 contains the business-related activities needed to manage a manufacturing organisation. These levels are illustrated in Figure 4. [13, 2]

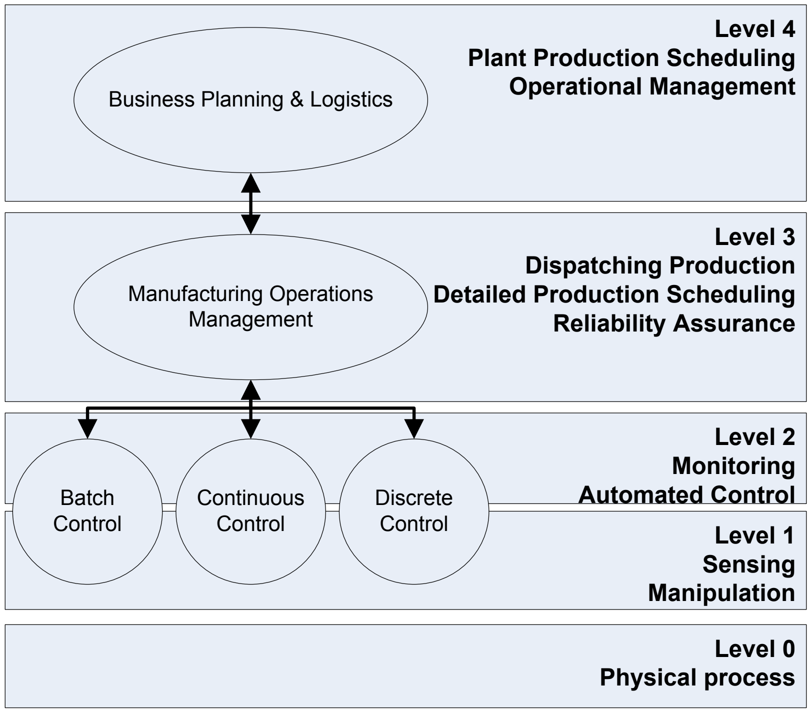


Figure 4: ISA-95 functional hierarchy model, adapted from [2]

The third level, manufacturing operations management, is divided into 4 main categories: production operations management, quality operations management, inventory operations management and maintenance operations management. ISA-95 defines maintenance operations management to be "activities within Level 3 of a manufacturing facility that coordinate, direct, and track the functions that maintain the equipment, tools, and related assets to ensure their availability for manufacturing and ensure scheduling for reactive, periodic, preventive or proactive maintenance" [2].

The definition in ISA-95 can be compared to the definition of maintenance management in EN 13306, which defines it to be "all activities of the management that determine the maintenance objectives, strategies, and responsibilities and implement them by means such as maintenance planning, maintenance control and supervision, improvement of methods in the organization including economical aspects"[1]. This definition has a broader scope, but is not contradictory to the definition used by ISA-95 which is used in this work.

The activities in maintenance operations management are illustrated in Figure 5. Equipment state of health data refers to the data from levels 1 and 2 that indicates the health of the equipment. This data is produced by condition monitoring processes. ISA-95 refers to ISO 13374 for defining these processes and the involved data. [2]

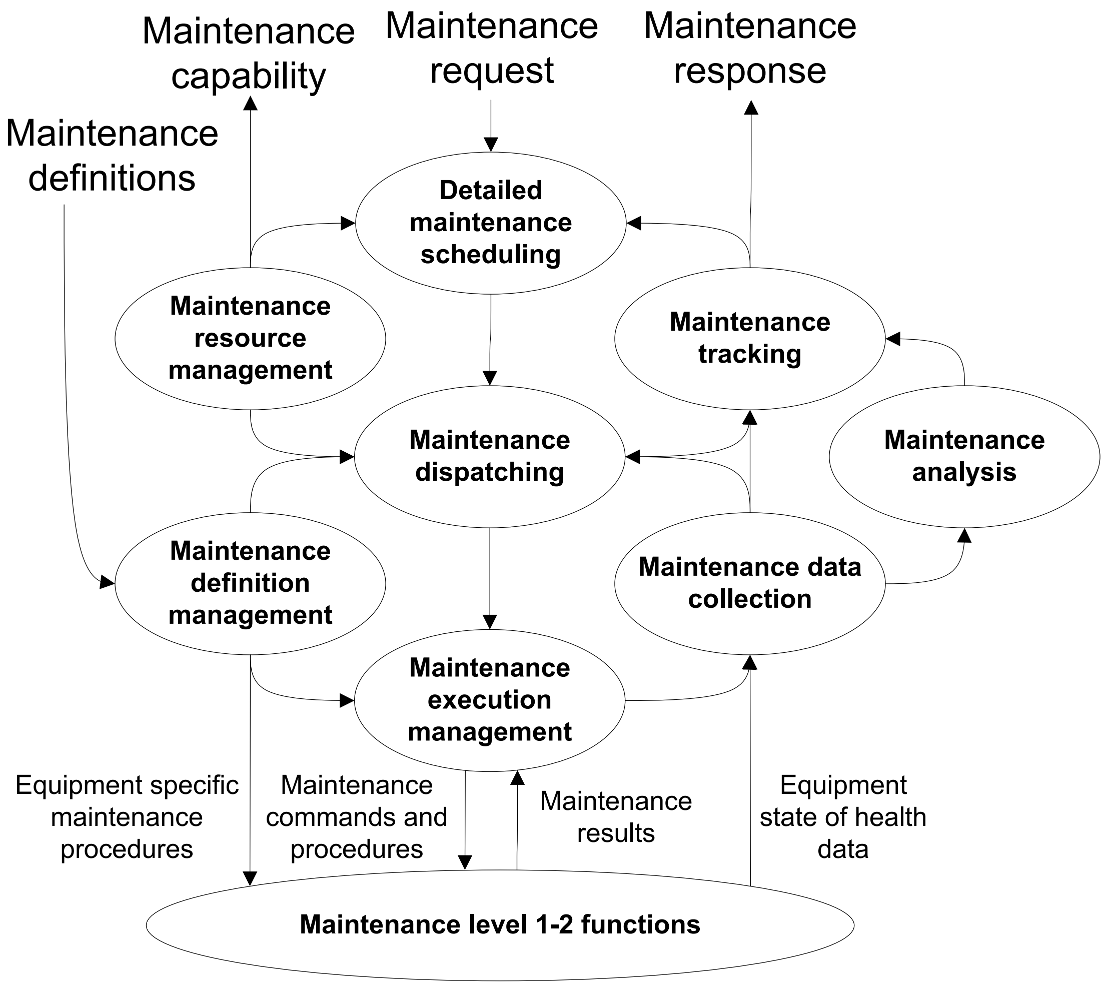

Maintenance capability
Maintenance request
Maintenance response
Maintenance definitions
Detailed maintenance scheduling
Maintenance resource management
Maintenance tracking
Maintenance dispatching
Maintenance analysis
Maintenance definition management
Maintenance data collection
Maintenance execution management
Equipment specific maintenance procedures
Maintenance commands and procedures
Maintenance results
Equipment state of health data
Maintenance level 1-2 functions

Figure 5: Activity model of maintenance operations management, adapted from [2]

2.3 Automation and information systems in maintenance operations management

The main information system in maintenance operations management is the Enterprise Asset Management (EAM) system, also called the Computerised Maintenance Management System (CMMS). In condition based maintenance in particular, Condition Monitoring (CM) systems are also important. This section describes the functionality by applying the relevant standards and presents examples of both systems.

2.3.1 Enterprise asset management systems

Enterprise asset management systems support the activities in maintenance operations management. These activities are described by the ISA-95 activity model [2]. These descriptions, adapted from [2], are presented next.

Maintenance definition management activity manages the information necessary to complete maintenance tasks, such as documentation about the equipment and maintenance procedures.

Maintenance resource management controls maintenance personnel and their training and skills, maintenance supplies and spare part inventory.

Detailed maintenance scheduling includes work such as processing the work requests and generating the work orders if necessary. Planning the maintenance work so that its impact on the production is minimised is a part of this activity.

Maintenance dispatching assigns the work orders to the maintenance resources identified by the maintenance resource management activity.

Maintenance execution management directs the maintenance work. This activity monitors the maintenance work and reports any unexpected events to the other activities in maintenance operations management.

Maintenance data collection collects and reports the measured data about the maintenance work, for example the duration of the maintenance work.

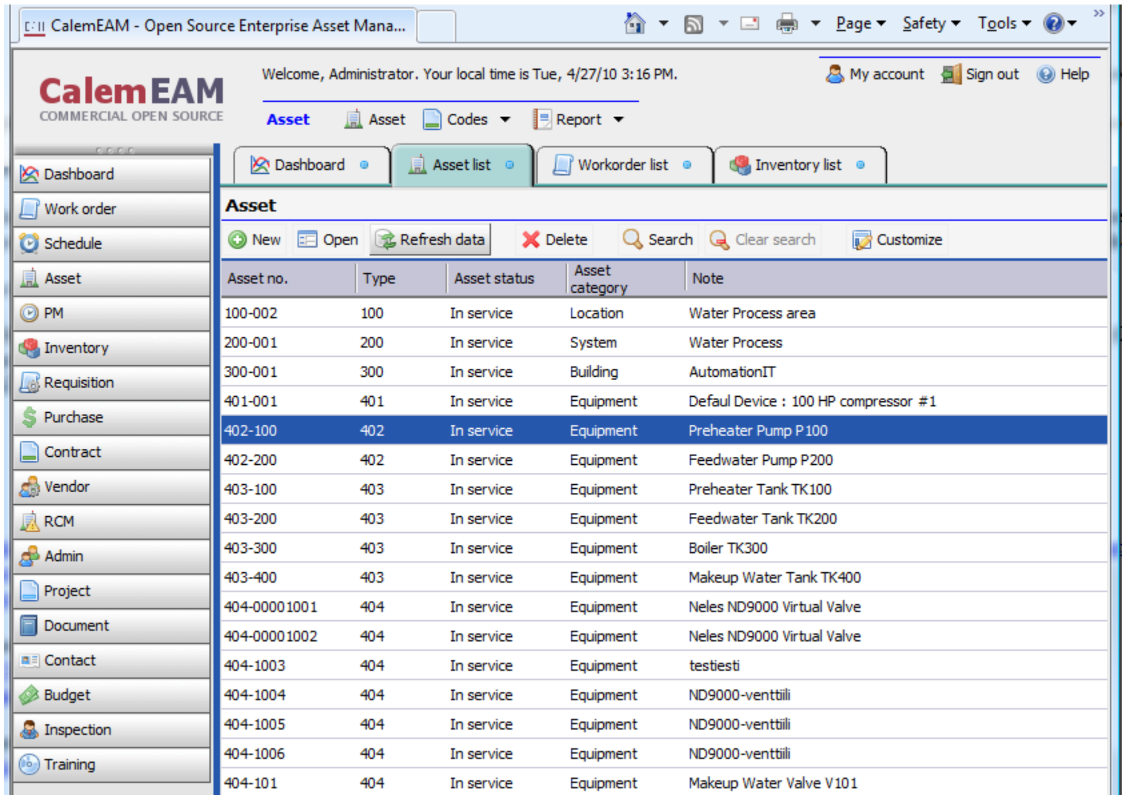
Maintenance tracking analyses the maintenance data and produces reports about the utilisation of resources and effectiveness of the maintenance work.

Maintenance analysis identifies problem areas, such as impending equipment failures, and areas of improvement. This activity may produce information that suggests maintenance process improvements, such as where the maintenance work should be focused to improve the reliability of critical equipment.

The functionality between different EAM systems varies. Some of these functionalities can be found in other systems, or they can be unsupported by IT. However, all of these activities are performed with varying levels of sophistication in a manufacturing enterprise [2].

An EAM system can be a part of an Enterprise Resource Planning (ERP) system. For example, the EAM system of SAP, SAP Plant Maintenance (PM), is a module of their ERP system [14]. Separate EAM systems include IBM Maximo, Solteq Artturi, CalemEAM and Infor EAM.

The functionality of CalemEAM is presented as an example of a typical EAM system. Figure 6 illustrates the menu structure and the different functionalities of the system. CalemEAM is a web-based EAM system, backed by the MySQL relational database management system. The system has been implemented with the PHP programming language. There are two editions of the system: community edition that is available for free, and the enterprise edition that requires a subscription. This description refers to the community edition. [15]



Welcome, Administrator. Your local time is Tue, 4/27/10 3:16 PM.

Asset

Asset no.	Type	Asset status	Asset category	Note
100-002	100	In service	Location	Water Process area
200-001	200	In service	System	Water Process
300-001	300	In service	Building	AutomationIT
401-001	401	In service	Equipment	Defaul Device : 100 HP compressor #1
402-100	402	In service	Equipment	Preheater Pump P100
402-200	402	In service	Equipment	Feedwater Pump P200
403-100	403	In service	Equipment	Preheater Tank TK100
403-200	403	In service	Equipment	Feedwater Tank TK200
403-300	403	In service	Equipment	Boiler TK300
403-400	403	In service	Equipment	Makeup Water Tank TK400
404-00001001	404	In service	Equipment	Neles ND9000 Virtual Valve
404-00001002	404	In service	Equipment	Neles ND9000 Virtual Valve
404-1003	404	In service	Equipment	testiesti
404-1004	404	In service	Equipment	ND9000-venttiili
404-1005	404	In service	Equipment	ND9000-venttiili
404-1006	404	In service	Equipment	ND9000-venttiili
404-101	404	In service	Equipment	Makeup Water Valve V101

Figure 6: Screenshot of CalemEAM Community Edition

The information CalemEAM can present about an asset include its identification, location, current status, purchasing information (warranty, deprecation), the service history and meter readings. CalemEAM organises the assets into a hierarchy, so an asset can be a part of another asset. For example, a production plant can be modelled as an asset, and can include multiple production line assets which consist of the actual equipment. [15]

CalemEAM organises maintenance information as work orders. The life of a work order begins when a user of the system creates a work request. Work orders can also be generated automatically according to the selected preventive maintenance strategy. The maintenance personnel can check the work request, and if necessary accept it and schedule it to a maintenance engineer. The user can attach maintenance instructions, information about the required spare parts and information about planned downtime to the work order. When the maintenance work is done, the information about used spare parts, labor (including possible overtime) and other notes can be added to the work order. CalemEAM can also manage the spare part inventory. [15]

CalemEAM also includes other functionality such as maintenance personnel information management. Personnel work shifts, training and certifications can be tracked by the system. However, CalemEAM lacks support for IT system integration. It does not support any specific integration standards. If CalemEAM must be connected to other IT systems, a custom solution must be developed.[15]

2.3.2 Condition monitoring systems

Condition monitoring systems are used to monitor the condition of equipment. Functionality of a condition monitoring system is presented by the ISO 13374 standard, which presents guidelines for the requirements. The standard divides the equipment condition assessment to 6 blocks of functionality. The blocks are illustrated in Figure 7. The data acquisition block collects the data in a digital form from the

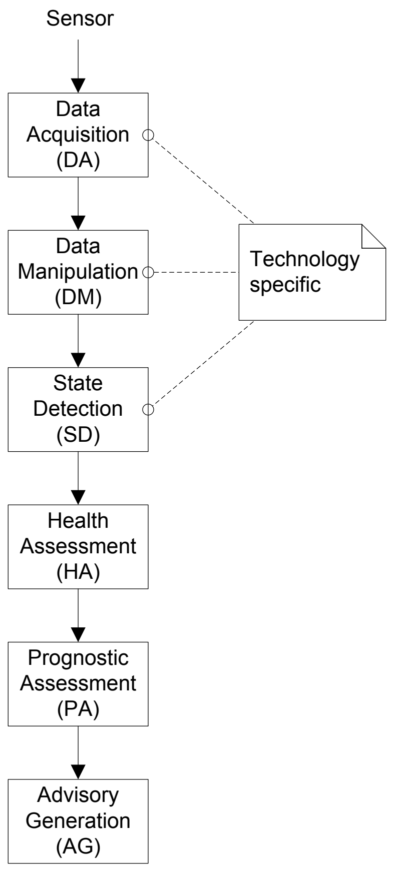
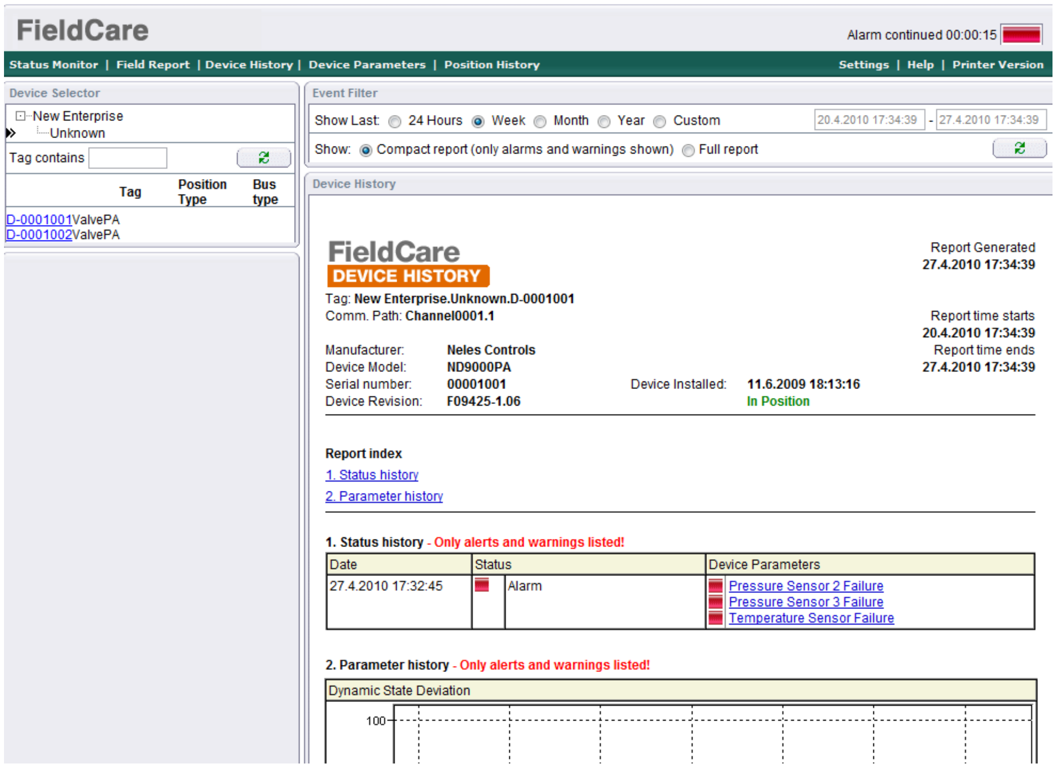


Figure 7: ISO 13374 data processing blocks, adapted from [3]

transducers. Data manipulation block analyses the data and produces virtual sensor readings from the measurements produced by the data acquisition block. State detection block detects abnormal readings and produce warnings and alerts. Health assessment block can rate the current health of the equipment. Prognostic assessment block can predict remaining life of the equipment according the the projected usage and the data produced by the other blocks. Finally, the advisory generation block generates actionable information from all this data. [3]

Condition monitoring systems do not always implement all blocks of functionality. Health assessment, prognosis assessment and advisory generation are often left to maintenance personnel or to more sophisticated analysis tools. [3]



Tag	Position Type	Bus type
D-0001001	ValvePA	
D-0001002	ValvePA	

Show Last: 24 Hours [✓ Week] Month Year Custom 20.4.2010 17:34:39 - 27.4.2010 17:34:39

Show: [✓ Compact report (only alarms and warnings shown)] Full report

FieldCare DEVICE HISTORY

Report Generated 27.4.2010 17:34:39

Tag: New Enterprise.Unknown.D-0001001
Comm. Path: Channel0001.1

Report time starts 20.4.2010 17:34:39
Report time ends 27.4.2010 17:34:39

Manufacturer:	Neles Controls		
Device Model:	ND9000PA		
Serial number:	00001001	Device Installed:	11.6.2009 18:13:16
Device Revision:	F09425-1.06		In Position

Report index
1. Status history
2. Parameter history

1. Status history - Only alerts and warnings listed!

Date	Status		Device Parameters
27.4.2010 17:32:45	🟥	Alarm	🟥 Pressure Sensor 2 Failure 🟥 Pressure Sensor 3 Failure 🟥 Temperature Sensor Failure

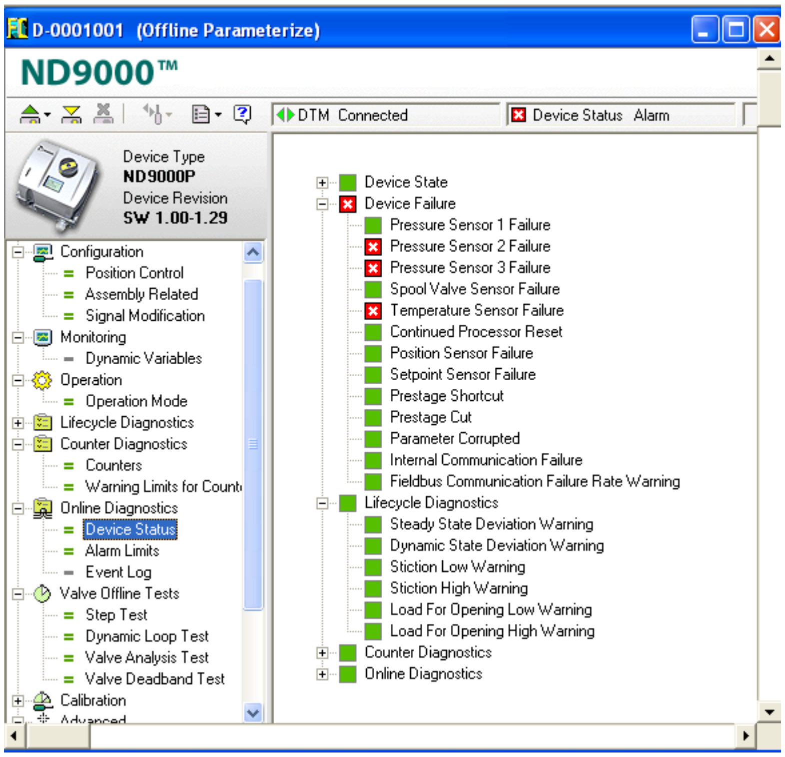
2. Parameter history - Only alerts and warnings listed!

Dynamic State Deviation
100

Figure 8: Screenshot of the condition monitoring web interface

Metso FieldCare is a device management software with an optional condition monitoring module. FieldCare is based on the Field Device Tool (FDT) technology, which is described in more detail in section 3.3. FieldCare can communicate with intelligent devices over the FDT interface and present the interface of the device to the user of the application. The condition monitoring module can additionally monitor the diagnostics of the device. Figure 8 illustrates the web interface of the FieldCare condition monitoring module.

Metso FieldCare does not include the advanced prognosis, health assessment and advisory generation functionalities of the ISO 13374 standard. However, it can be used as a data collector for a more advanced system that can produce this information from the data provided by FieldCare. FieldCare does not produce the data by itself, but instead can access the data provided by intelligent devices that include self-diagnosis features. Neles ND9000PA is a valve controller that contains multiple sensors that can be used to monitor the state and health of the equipment. The



Device Type
ND9000P
Device Revision
SW 1.00-1.29

- Configuration
  - Position Control
  - Assembly Related
  - Signal Modification
- Monitoring
  - Dynamic Variables
- Operation
  - Operation Mode
- Lifecycle Diagnostics
- Counter Diagnostics
  - Counters
  - Warning Limits for Counte
- Online Diagnostics
  - Device Status
  - Alarm Limits
  - Event Log
- Valve Offline Tests
  - Step Test
  - Dynamic Loop Test
  - Valve Analysis Test
  - Valve Deadband Test
- Calibration
- Advanced

- 🟩 Device State
- 🟥 Device Failure
  - 🟩 Pressure Sensor 1 Failure
  - 🟥 Pressure Sensor 2 Failure
  - 🟥 Pressure Sensor 3 Failure
  - 🟩 Spool Valve Sensor Failure
  - 🟥 Temperature Sensor Failure
  - 🟩 Continued Processor Reset
  - 🟩 Position Sensor Failure
  - 🟩 Setpoint Sensor Failure
  - 🟩 Prestage Shortcut
  - 🟩 Prestage Cut
  - 🟩 Parameter Corrupted
  - 🟩 Internal Communication Failure
  - 🟩 Fieldbus Communication Failure Rate Warning
- 🟩 Lifecycle Diagnostics
  - 🟩 Steady State Deviation Warning
  - 🟩 Dynamic State Deviation Warning
  - 🟩 Stiction Low Warning
  - 🟩 Stiction High Warning
  - 🟩 Load For Opening Low Warning
  - 🟩 Load For Opening High Warning
- 🟩 Counter Diagnostics
- 🟩 Online Diagnostics

Figure 9: Screenshot of the ND9000PA DTM interface

DTM interface of the controller is pictured in Figure 9. The valve controller can automatically generate warnings and alarms that can be presented by FieldCare or any other FDT-based application. The condition monitoring module of FieldCare also supports presenting the valve condition data over different interfaces.

2.4 Summary

In this chapter the condition based maintenance was presented as a type of maintenance process. The need for condition based maintenance was motivated by presenting the evolution of the production equipment and processes. The relation of condition based maintenance to other processes in a manufacturing enterprise were presented by applying the ISA-95 framework. The ISA-95 framework also provides description of the activities in the maintenance operations management and these were presented. The descriptions were used to define the functionality of the most important IT system in maintenance operations management, the enterprise asset management (EAM) system. Condition monitoring (CM) systems are important in condition based maintenance. The functionality of this system and its connection to the EAM systems was described by applying the ISO 13374 condition monitoring standard along with the ISA-95 framework.

3 Application integration in maintenance

The new maintenance paradigms, such as condition based maintenance and predictive maintenance, are much more information intensive than the old approaches. Information must be readily available to make these new maintenance processes viable. To ease the flow of information, IT systems must be connected at both information and service levels. The process of binding these information sources is called application integration [16]. In this chapter, the selected technologies, standards and specifications that are involved in application integration in maintenance are presented, with a focus on OPC Unified Architecture.

3.1 Concepts

Application integration is a clear trend in the realm of IT, with the Service-oriented architecture (SOA) paradigm dominating. There are many definitions to the term, but it is generally agreed that an architecture based on SOA consists of discrete services that provide functionality described by service contracts and processes that compose these services to provide new services at higher level of abstraction [17]. It is also commonly agreed that the services should provide functionality that has some business value, such as invoicing and order placement [18, 19].

SOA is commonly implemented using a technology stack based on SOAP and Web Services Description Language (WSDL) extended with a set of WS-* specifications [18]. SOAP, which used to abbreviate Simple Object Access Protocol, is a messaging format for XML documents. This technology stack is commonly called the Web Services stack. SOAP messages can be sent over different transport channels, but most are sent over HTTP. WSDL is used to describe services, including the data types used and the addressing information of the service. A web service technology stack can use the WSDL description to discover how it should format its SOAP messages to communicate with a given service. WS-* specifications can be used to implement additional features such as encryption or transactions. [18]

The most popular platforms used in enterprise IT are the Java platform and the .NET platform. In the Java platform, Web Services can be implemented with the Java API for XML Web Services (JAX-WS). The Microsoft counterpart, Windows Communication Foundation (WCF), has a broader scope. It can be used to communicate using not only Web Services, but also other communication protocols. [4]

Windows Communication Foundation is a framework for building connected applications using services. An application can use WCF to expose its functionality as services or it can use WCF to access services provided by others. The applications communicate with each other using messages. WCF framework encodes these messages and sends them to the recipient using the correct protocol. [4]

The WCF architecture is illustrated in Figure 10. If a client wants to use a remote service, it calls the service using methods provided by the WCF service framework.

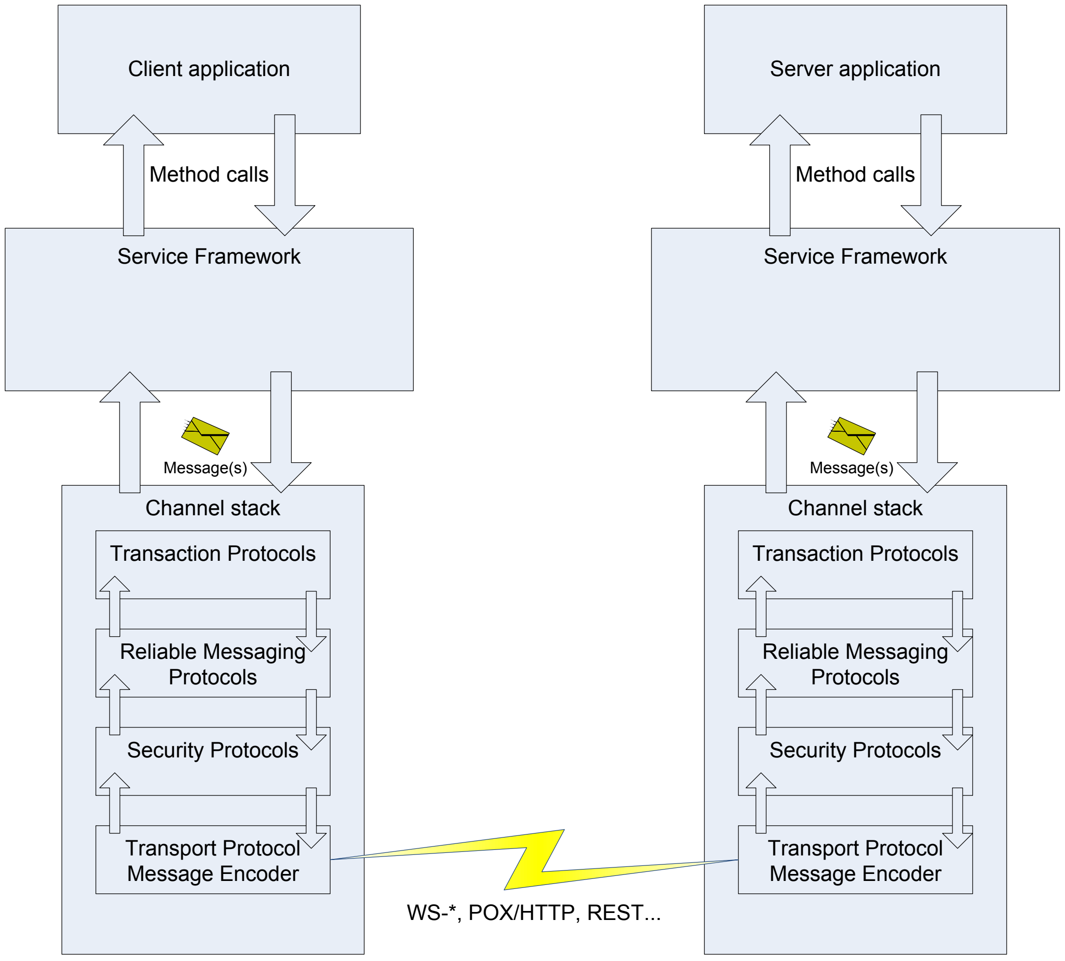


Figure 10: WCF architecture. [4, 5]

The framework then converts this method call to a SOAP message and passes the message to the WCF channel stack. The stack adds necessary metadata to the SOAP message header to implement features such as encryption, transactions and so on. Finally the message is received by the final element in the channel stack: the transport protocol message encoder which transforms the SOAP message to the protocol-specific wire format. This wire format can be based on SOAP but can also be something completely different. [4, 5]

3.2 Application integration in condition based maintenance

The most important applications to be integrated in condition monitoring are the Condition Monitoring (CM) and Enterprise Asset Management (EAM) systems [20]. These IT systems handle device data differently. Field Device Tool (FDT) and Electronic Device Description Language (EDDL) are two important technologies that facilitate device integration [7]. OPC UA is a new technology that has been proposed for device integration. OPC UA can also be applied to other integration scenarios. Condition monitoring data interfaces have been defined by the MIMOSA OSA-CBM standard, but it has not been adopted as successfully as FDT and EDDL have.

Enterprise asset management systems usually use protocols based on the Web Services technology stack. These specifications are usually vendor-specific; proposed standardisation efforts such as MIMOSA OSA-EAI have not been widely adopted. Solutions based on MIMOSA OSA-EAI still need custom vendor-specific adapters to connect to EAM systems [21].

The interface standards used in automation IT differ greatly from the standards used in higher level IT systems. Devices communicate their condition data over fieldbus protocols which are usually not based on the IP stack. Unlike protocols based on the IP stack, fieldbus protocols must guarantee a bounded response time [22]. In this work the simulated valves of the research platform use the PROFIBUS PA fieldbus protocol. Features of PROFIBUS PA include its electrical properties that allow it to be used in hazardous areas, the ability to support power to field devices and its capability to transfer diagnostic data from devices along with the control data [23].

3.3 FDT

FDT is an interface specification that allows communication with devices over a generic interface. Developers can use the FDT interface to communicate with field devices without implementing fieldbus-specific functionality. In the FDT concept the device vendors deliver a device-specific software component called the Device Type Manager (DTM) that is used by a FDT frame application to provide a user interface for the device. DTM components can also provide access to different fieldbus networks. The DTM component implements the ActiveX interfaces specified by the FDT specification. ActiveX is a feature of Microsoft Windows which makes FDT dependant on the Windows platform. An example of an FDT frame application was presented in section 2.3.2. [24]

The FDT concept allows device vendors to implement complex DTM components that provide rich functionality. The inner workings of DTM component are left unspecified by the FDT specification, which concentrates on the interfaces. This means that the interpretation of device data is left to the closed DTM components. Using solely the FDT interface it is not possible to implement vendor-independent condition monitoring software, because the FDT interface cannot identify the necessary data. The scope of the FDT concept is limited to the engineering and commissioning of field devices. [24]

3.4 OPC UA

OPC Unified Architecture (OPC UA) is the successor of OPC, which is an older set of communication standards used in industrial automation. The most important specification of OPC is the Data Access (OPC DA) specification, which is used to move real-time data from control devices to other systems. OPC standards also include specifications for historical data access (OPC HDA) and alarms and events (OPC A&E). OPC standards were implemented using Distributed Component Object Model (DCOM) which is a proprietary technology by Microsoft. This effectively made the OPC standards unportable. Traversing firewalls with classic OPC was also difficult [25]. DCOM is considered to be obsolete and is being replaced by Web Services [26].

In addition to the problems mentioned earlier, OPC vendors and collaborating organisations needed a way to move data at a higher level of abstraction. The original OPC also had different standards for different data models, which was seen as a problem. The new specification is called OPC Unified Architecture (OPC UA). OPC UA unifies the functionality of the classic OPC standards under a single technology framework. Unlike the classic OPC, the framework does not depend on proprietary technologies such as DCOM and can thus be implemented on different platforms. OPC UA provides a rich set of tools to model data on a higher level of abstraction compared to classic OPC. UA also has more robust set of security features, a fundamental requirement of a protocol used over insecure networks. Because of these features, OPC UA can be used in situations where the classic OPC could not. [6]

3.4.1 Concepts

The set of information that an OPC UA server presents to its clients is called the address space of the server. The address space consists of nodes, references and attributes of the nodes. Node is the fundamental component of an address space. Attributes define the characteristics of the node. All nodes are identified by their id attribute. References are pointers from one node to another. All references are defined by reference types. An example of a reference type would be “Contains”. Such a reference from node A to node B denotes a relation “A contains B”. [27]

Some but not all references are hierarchical. Hierarchical references can be used to construct a node hierarchy in the address space in the form of a tree. In general, the address space of an OPC UA server is a mesh network. The entry point to this mesh network is called the root node, and is defined by the OPC UA specification. [27]

On top of the mesh network, the specification defines a object model called OPC UA Object Model. The model defines how OPC UA objects should be defined in terms of nodes, attributes and references. OPC UA objects support member variables, methods, events and references to other objects. The primary objective of the OPC UA address space is to represent such objects with the basic concepts of nodes, references and attributes. The relation of OPC UA Object Model and address space is illustrated in Figure 11. [27]

An important attribute of an OPC UA object node is BrowseName. The intent is that object types in the address space define the browse names of the members of

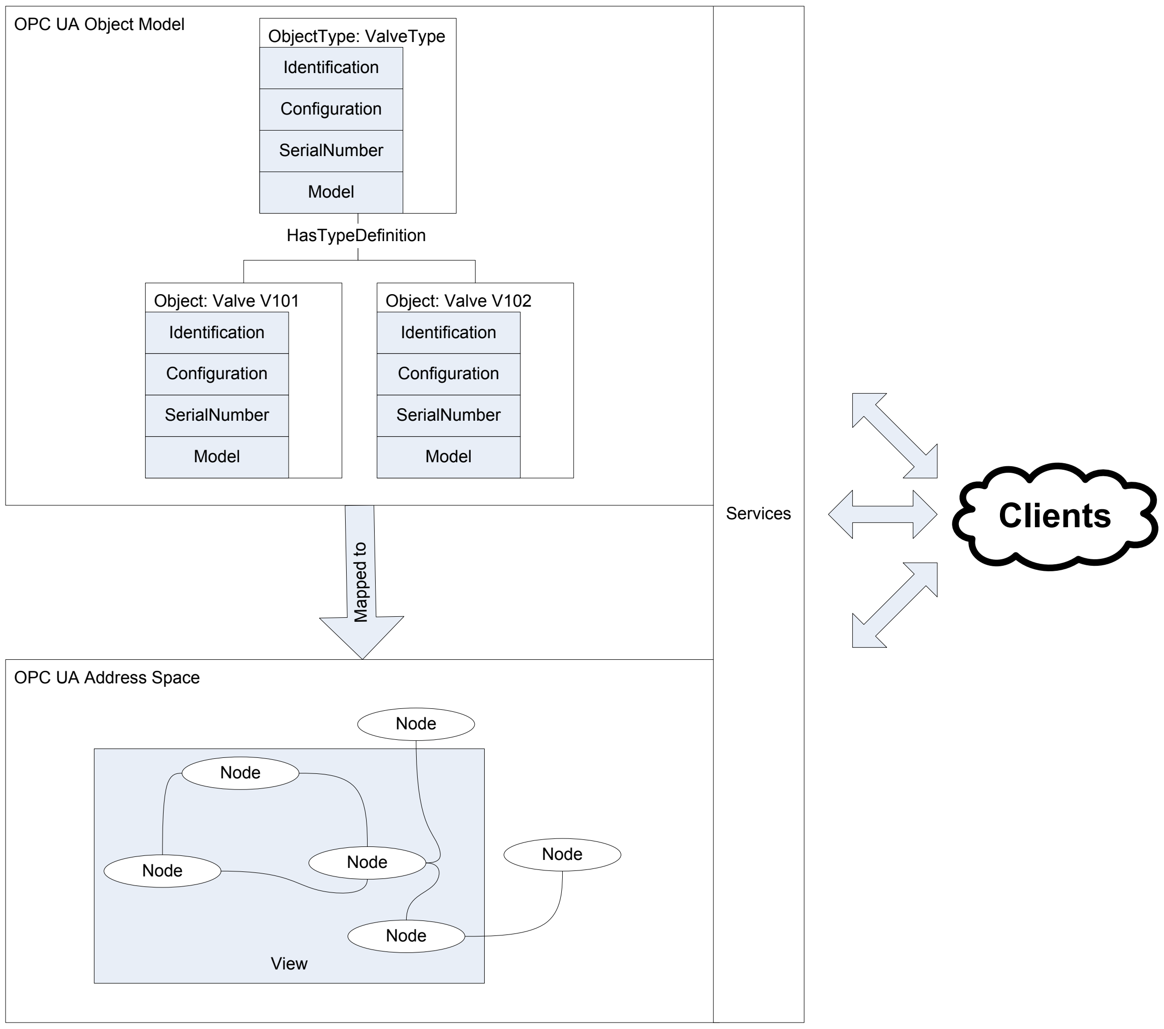
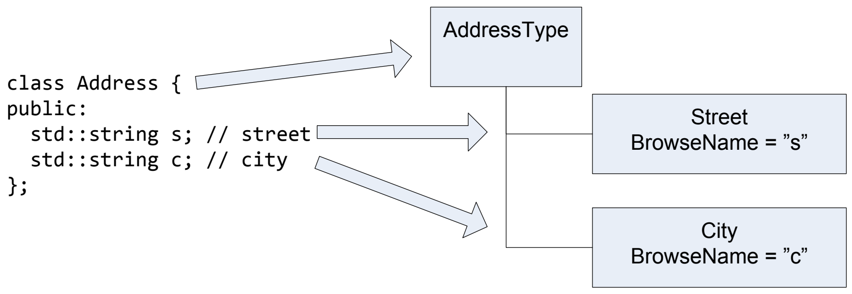


Figure 11: OPC UA address space, object model and services [6].



```
class Address {
public:
  std::string s; // street
  std::string c; // city
};
```

Figure 12: Mapping a C++ class to UA ObjectType, adapted from [6].

objects of that type. These browse names can then be used to access the members of any object with that same object type, or any of its subtypes. The browse name of the object in an object type definition can be compared to the variable name of a class member in object oriented programming languges, as illustrated in Figure 12. [6]

OPC UA Services are an abstract set of interfaces which define how clients can access the address space of the server. Important services to access the data in OPC UA address space include the Read service, which is used to read attribute values of nodes, and TranslateBrowsePathsToNodeIds, which is used to find out members of an object if its object type is known. If a client wants to read the contents of an object, then it first uses the TranslateBrowsePathsToNodeIds service to find out the node id values of all nodes related to the object, and then the Read service to read the attribute values of the nodes. [28]

The abstract services are mapped to concrete implementations by an OPC UA stack. This mapping process is defined by the Service Mappings part of the OPC UA specification [29]. The OPC Foundation provides stack implementations and their intent is that typical UA application designers do not need to concern themselves with the message representations on the wire. Currently, the OPC UA services can be mapped to a Web Service or a custom binary TCP protocol binding. The OPC Foundation provides a C# SDK implementation that supports both bindings and stack implementations in Java and C that only support the binary TCP protocol [30].

OPC UA information models define how data should be represented in the address space as OPC UA objects, nodes, attributes and references. OPC Foundation defines a few basic information models for common use cases. The most important information model is the data access (DA) information model, which defines how to represent and use automation data. The base UA specification also has information models for alarms and conditions (AC), historical data access (HA), aggregate values and long-running programs. On top of these information models more specialised information models can be built. [6]

The base OPC UA specifications are considered to only provide the basic infrastructure for information modelling [6]. At the time of writing, there exists two information models that have been defined outside the base specification. The OPC

UA device information model (DI) defines how field devices should be represented in the address space [31]. Built on top of this model is the analyser device information model (ADI) which has more detailed rules on how to model analyser devices [32].

Using these information models, vendors can present their data in a standard data model which makes it easier for clients to consume their data. The layering of information models is illustrated in Figure 13. The OPC Foundation expects other domain-specific models from other industry organisations[6]. The PLCopen information model has already reached the release candidate status [33].

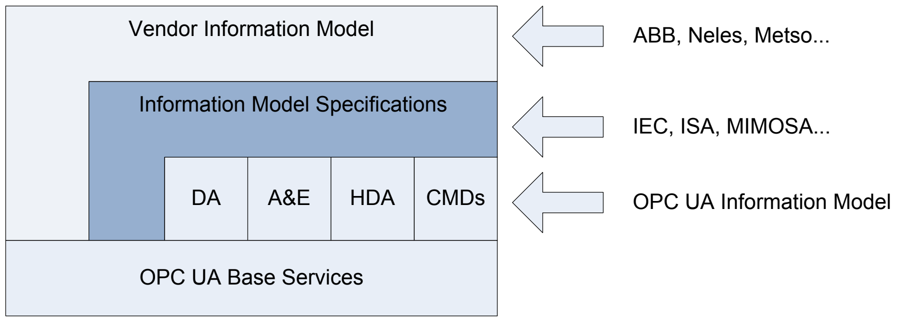


Figure 13: Layering of OPC UA information models, adapted from [6].

3.4.2 OPC UA device information model

OPC Foundation has released a companion specification that describes how devices should be presented in the OPC UA address space. The specification mostly describes general rules on how to organise parameters and methods of devices in the address space and has very light requirements on what parameters should be present. It is assumed that more domain-specific information models, such as the analyzer device information model (ADI), have more detailed descriptions on the semantics. [31]

The fundamental object type of this information model is the TopologyElementType. All configurable elements in the device topology are instances of this type or its subtypes. Instances of TopologyElementType may contain parameters and methods. They are respectively contained in objects ParameterSet and MethodSet as a flat list.[31]

Parameters and methods can optionally be organised under different functional groups to mirror the structure of the TopologyElement. Functional groups are presented as instances of FunctionalGroupType, which is a subtype of FolderType. There is one mandatory functional group called Identification, which is used to organise parameters used for identification of the topology element. TopologyElementType is abstract, so it is not possible to create instances of it. [31]

The most important subtype of TopologyElementType is DeviceType, which is used to represent devices. In addition to the Identification functional group defined in its supertype, DeviceType also defines properties that are used to further describe the device. These include properties that define the model, manufacturer, version and serial number of the device. DeviceType is also abstract, which means that vendors must inherit this type if they want to present their devices in the address space. [31]

3.4.3 Field Device Integration

There is currently a new standard in development related to OPC UA device integration called Field Device Integration (FDI). The aim of the standard is to combine the advantages of EDDL and FDT under a single technology framework. The standard is not yet finished nor there are any draft versions available, but one presented concept is to separate the device model into two parts. [7]

In the concept, the two parts are called the Device Information Model (DIM) and Device Operation Model (DOM). The Device Information Model (DIM), described in EDDL, includes basic functionality and the device data and state. A device always has one Device Information Model, but can also have Device Operation Models (DOM) that can define more advanced business logic functionality, graphical user interfaces and so on. The DOM concept is comparable to the DTM concept in FDT. [7]

The Device Information Model is mapped to OPC UA server address space so that the device data can be accessed using ordinary OPC UA clients. The concept also stores the Device Operation Models in the address space, but it is only handled by the server in terms of server side storage so that customised clients can load the DOM when more complex functionality is required. This FDI concept is illustrated in Figure 14. [7]

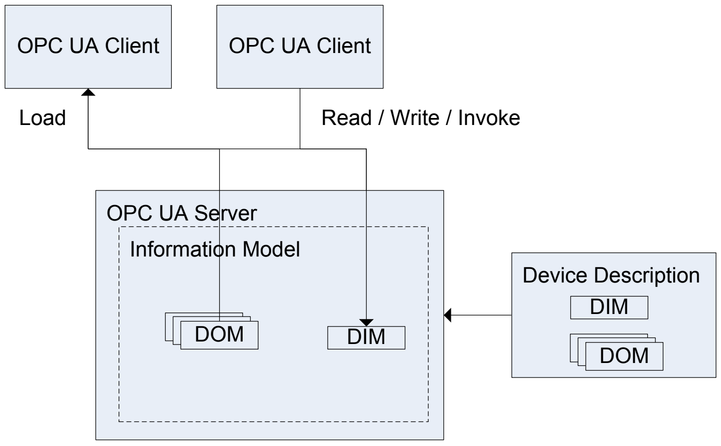


Figure 14: FDI concept adapted from [7].

The status of FDI is currently unclear. Information regarding the state of the FDI specification, which is scheduled for summer of 2010 [34], is scarce. However, it is expected that the end result will somehow involve OPC UA [6].

3.4.4 Current status and future

At the time of writing the OPC UA base specifications are almost complete - part 9, describing alarms and events, has not yet reached 1.00 but a release candidate is already available. The utility specifications, Aggregates and Discovery, are still missing [30]. The specifications are in the IEC standardisation process, and will be known as IEC 62541 standards [6].

At the time of writing, numerous SDKs and stack implementations are available from third-party developers. OPC Foundation also provides SDKs and stack implementations to its corporate members. However, end-user products using OPC UA are still relatively rare. OPC Foundation is still working on a certification process for OPC UA products and it is expected that it becomes available in the early 2010. OPC Foundation also arranges interoperability workshops where vendors can test their implementations with other products. [35]

OPC UA has a strong emphasis on information modelling, but domain specific information models are still very rare. The device information model is very generic and adds little semantic information, mostly related to the identification of the element in device topology [31]. The analyser device information model builds on the device information model, and is much more specific in its domain [32]. Similar models for other device types are needed to fully reap the benefits of OPC UA. The FDI specification can have a large impact on the device modelling domain. [7]

In other domains, PLCopen is working on a information model that defines how function blocks and data structures of IEC 61131-3 should be presented in an OPC UA address space [33]. OPC Foundation has also identified candidates such as ISA S95, S88, OAGiS and MIMOSA for companion specifications [30].

3.5 MIMOSA and OpenO&M standards

MIMOSA is a trade association that is involved with the development of information integration standards for operations and maintenance. MIMOSA consists of operations and maintenance (O&M) solution providers and end-user companies. The most important specifications published by MIMOSA are the Open Systems Architecture for Enterprise Application Integration (OSA-EAI) and Open Systems Architecture for Condition-Based Maintenance (OSA-CBM). OSA-EAI is a standard that describes how to integrate asset management information such as reliability, diagnosis and identification. OSA-CBM specifies how to transfer information in a condition-based maintenance system.

OpenO&M is an organisation with a similar mission. OpenO&M tries to harmonise the standards used for application integration in operations and maintenance. The OpenO&M initiative is organised into industry-specific Joint Working Groups (JWG). There exists a manufacturing JWG, a military JWG and a facilities JWG. Manufacturing JWG members are MIMOSA, the OPC Foundation, ISA88, ISA95, The Organization for Production Technology (WBF) and the Open Applications Group (OAGi). The manufacturing JWG has presented two technologies that can be used to connect information systems in the manufacturing domain: The OpenO&M Information Service Bus, which provides the infrastructure for information transfers and Common Interoperability Registry (CIR), which is used to link local names of entities in separate systems together. [36]

3.5.1 OSA-EAI

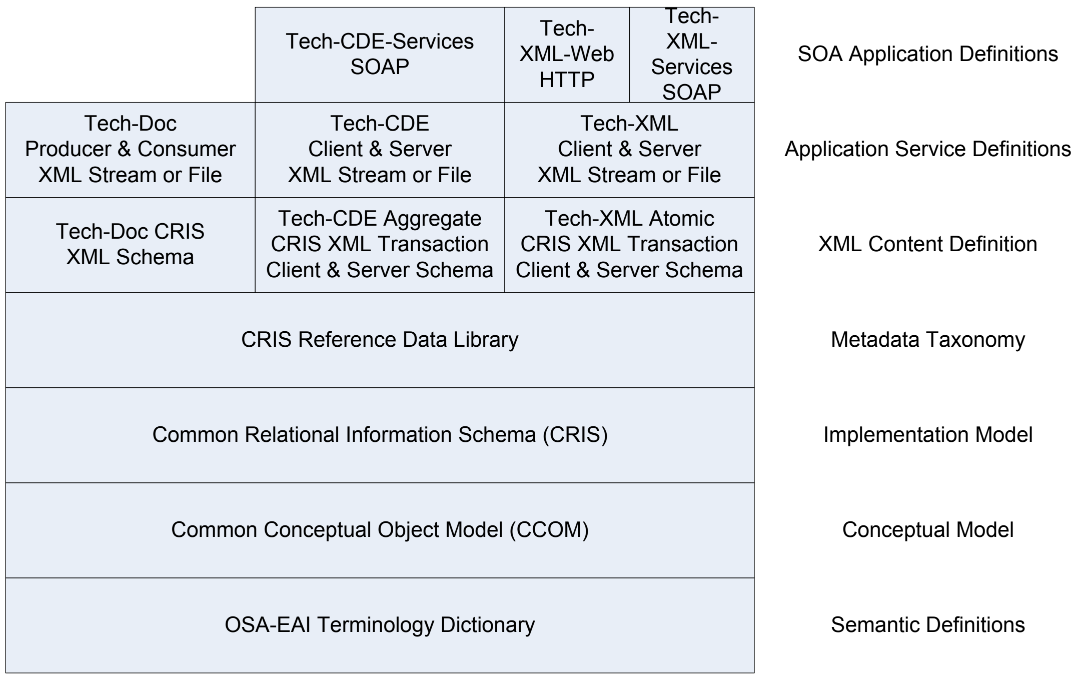

Tech-CDE-Services SOAP
Tech-XML-Web HTTP
Tech-XML-Services SOAP
SOA Application Definitions
Tech-Doc Producer & Consumer XML Stream or File
Tech-CDE Client & Server XML Stream or File
Tech-XML Client & Server XML Stream or File
Application Service Definitions
Tech-Doc CRIS XML Schema
Tech-CDE Aggregate CRIS XML Transaction Client & Server Schema
Tech-XML Atomic CRIS XML Transaction Client & Server Schema
XML Content Definition
CRIS Reference Data Library
Metadata Taxonomy
Common Relational Information Schema (CRIS)
Implementation Model
Common Conceptual Object Model (CCOM)
Conceptual Model
OSA-EAI Terminology Dictionary
Semantic Definitions

Figure 15: OSA-EAI V3.2 architecture, adapted from [8].

OSA-EAI is a specification that defines how to model data used in operations and maintenance and how to transfer it between systems. The architecture is illustrated in Figure 15. OSA-EAI defines its conceptual model with the technology-neutral Common Conceptual Object Model (CCOM), specified using UML notation. The implementation of this object model in a relational database management system is defined by the Common Relational Information Schema (CRIS). CRIS is defined to support installations where the data is distributed to many separate databases. OSA-EAI specification includes a reference data library which is used to classify the most common entities. [8]

Data stored in a CRIS-compliant database can be exported from the database in the XML format using Tech-* schemas defined by OSA-EAI. These schemas are Tech-Doc (Document), Tech-CDE (Compound Document Exchange) and Tech-XML. A Tech-Doc document can contain unlimited amount of CRIS-compliant data. Tech-CDE is used for aggregate sets of CRIS data and can be used to implement query services in the server. Tech-XML is used to transfer small packets of atomic CRIS data. Because Tech-XML is more limited compared to the other technologies, it can be used more easily in systems that do not store their data in a CRIS-compliant database. Tech-CDE and Tech-XML have a SOAP binding available, so they can be used with the Web Service technology stack. [8]

These specifications (Tech-Doc, Tech-CDE and Tech-XML) are further divided into nine technology types. These technology types are listed in Table 1. It is not necessary to implement all these technology types to be OSA-EAI conformant. Vertical applications can only choose a subset that makes sense in their domain and thus do not need to implement the whole CRIS schema. When transferring simple condition monitoring data, the REG and TREND technology types are most important. [8]

Table 1: OSA-EAI Technology Types [8]

Technology Type	**Functionality**
REG	Registry Management
REL	Reliability
WORK	Work Management
DIAG	Diagnosis / Prognostics / Health Assessment
DYN	Dynamic Vibration / Sound Condition Monitoring
TREND	Static "Trendable" Condition Monitoring
SAMPLE	Oil / Fluid / Gas / Solid Tests Condition Monitoring
BLOB	Binary Data / Thermography Condition Monitoring
TRACK	Physical Geospatial Tracking

Tech-XML includes a a specification called the Tech-XML-Services Client & Server Specification, which is a SOAP binding for Tech-XML data. The interfaces are defined using WSDL files with the message contents used in the operations defined in separate XSD schema files. The operations accept these messages as input data, add their response to the message and then finally return the message containing both the request and the response to the client. Tech-XML-Services defines what operations must be supported to be compliant with a given OSA-EAI Technology Type. For example, to be compliant with REG Technology Type, over 50 operations must be implemented. [37]

3.5.2 OSA-CBM

OSA-CBM is a specification that defines how to move condition-based maintenance data between systems. It is an implementation of the ISO 13374 functional specification, which divides the functionality of a CBM system into six blocks of functionality. OSA-CBM builds on that standard and specifies interfaces to these blocks. The specification is defined using UML and is technology-neutral - it does not define any specific binding, so it can be implemented using any communication standard such as WS-* and DCOM. [38]

The interfaces to the function blocks can be divided into three different groups: data, configuration and explanation. Data interfaces provide information and events from the block, configuration interfaces provide information on the inputs, outputs and algorithms used by the block and explanation interfaces provide the data that was used by the block to produce an output. The interfaces are the same for every function block. [38]

3.5.3 Common Interoperability Registry (CIR)

Common Interoperability Registry is used to connect different tags used in separate systems together so that it is possible to find out what is the tag in system B if the tag in system A is known. CIR provides a model that assigns every entity a globally unique ID (GUID). This GUID can then be associated with system-specific tags. CIR also allows users to define properties for the entities that can be useful for identifying the entity in other systems even if the tag is now known.[39]

This model is accessed by using transactions defined in the specification. These transactions exchange XML messages conforming to the schemata defined by XSD files. There are no specific bindings defined, but it is relatively simple to implement these XML message exchanges using SOAP. These transactions include functionality for registry management (creating, combining and deleting registries), accessing registry data, and defining equivalent entries in different systems (entry E_1 in system S_1 in the same as entry E_2 in system S_2).

3.6 Related research

Existing research has different approaches to the problem of device integration in general, with little research done on device integration for purposes of condition monitoring. Use of OPC UA for field device integration has been widely discussed.

Hadlich describes the data available using existing FDT and EDDL interfaces that can be presented in OPC UA address space and concludes that an OPC UA server can provide data from both FDT and EDDL regardless of them having fundamentally different approaches [40]. Huang and Liu concentrate on mapping EDDL elements to OPC UA concepts [41]. Both of these works focus on presenting the devices in OPC UA address space, but lack description on how to utilise this data.

In other works, the SOA-based IT architecture of the enterprise is connected directly to the devices using Web Service protocols. An integration architecture has been presented by the SOCRADES project that uses the Device Protocol for Web Services (DPWS) for connecting to devices [42]. The SOCRADES middleware uses SAP MII to connect to enterprise systems. The work includes a Figure which proposes that OPC UA could be used over DPWS, but this combination is not further explained.

OSA-EAI has been applied in condition monitoring system in the work by Mathew et al. [43]. In the work, the OSA-EAI database schema (CRIS) was implemented for a condition monitoring system based on OSA-CBM. The work focuses on the database implementation issues and the implementation of OSA-EAI interfaces was not considered.

The POSC Caesar Association (PCA) is a nonprofit organisation that promotes the development of open specifications. The MIMOSA organisation has presented use cases of the integrated operations and maintenance solution to the Integrated Operations in High North (IOHN) project of the POSC Caesar Association [44]. These use cases can be divided to two groups. The first group includes the use cases that involve information regarding the engineering of the plant, for example, what equipment is used and why. These use cases are outside the scope of this work. The second group includes preventive maintenance use cases. Part of the requirements for the design done in this work are derived from these use cases.

3.7 Summary

The contemporary enterprise IT systems claim to follow the SOA paradigm. On the other hand, the IT systems used in automation have different approaches to the integration. It is claimed that the gap between these two worlds can be bridged with the new standard, OPC Unified Architecture. In this chapter the relevant standards in both domains were described. The chapter focused on the features of OPC Unified Architecture. Even though there is a lot of research regarding device integration and OPC Unified Architecture, there is relatively little literature available on using OPC UA to present device data to SOA-oriented IT systems.

4 Design of application integration

The main objective of this work is to create a feasible design that is based on using OPC Unified Architecture to bridge the gap between automation systems and IT systems in condition based maintenance. The integrated system should be able to cover the concerns and needs of the stakeholders in condition based maintenance. In this chapter, first the main stakeholders of the system are identified and the needs of them are described. The chapter adapts IEEE 1471 as its conceptual framework. The architectural description is not meant to conform to it, but it uses the concepts of the standard to communicate the architecture.

Viewpoints and views that conform to them are a central concept in IEEE 1471 [45]. Viewpoints are specifications and conventions for constructing representations (views) of the system. The structure and behaviour of the system are described with UML 2.0 viewpoints. Component diagrams are used to describe the static structure of the system, and sequence diagrams illustrate dynamic behaviour such as conversations and data flows. The design applies the ISA-95 standard to identify the functions and the boundaries of the IT systems.

4.1 Requirements

In this section, the requirements of the application integration are identified. The integrated system is meant to provide information on assets and their condition to maintenance engineers performing condition based maintenance. The integrated system operates as a part of the complete IT architecture of the enterprise. The system deals mostly with enterprise asset management and condition monitoring systems, but should be extensible enough so that if necessary, it can be connected to other information sources such as manufacturing execution systems (MES) and product lifecycle management (PLM) systems. Some of these systems, like condition monitoring systems, are on levels 0-2 in the ISA-95 functional hierarchy model, and some, such as PLM or EAM systems, operate on level 3. The integration environment on these levels are different, so the integrated system should be designed so that it can be connected to both kinds of systems. To derive the requirements of the application integration, the stakeholders of the system and their needs are identified. This approach is further described in the IEEE 1471 standard [45].

Stakeholders are users or classes of users that have interests or concerns regarding the system. The main users of the integrated system are the maintenance engineers that need information on the health of the assets and their maintenance history to perform condition based maintenance successfully. The type of the health data can vary depending on the sophistication of the condition monitoring system, from raw sensor data to prognosis and diagnosis of the state of the equipment as defined by ISO 13374. The use cases of maintenance engineers are described later in this section and further needs are identified.

The integrated system is implemented and maintained by the developers. They want that the system should be easy and fast to implement and maintain. The integration architecture should be applicable to wide variety of existing IT architectures deployed by the client. The integration architecture should be flexible enough to accomodate the evolution of the connected systems, with the assumption that the lower level systems will use OPC UA for communication.

The owners of the system are the plant operators. They want the equipment to operate at optimum capacity and they want that the implementation of the architecture goes smoothly so that costs stay low and the integration work does not take too much time. These needs are satisfied if the needs of maintenance engineers and developers are met.

4.1.1 Use cases

The use cases for maintenance engineers are derived from the business processes in condition based maintenance, which were described in section 2.1.1. The integrated systems should provide services that directly support these use cases. These services can then be used directly, or they can be included and orchestrated in other manufacturing operations management processes.

The condition based maintenance process starts when a notification is received, and is roughly divided to two parts: situation assessment and execution management. Situation assessment requires the maintenance engineer to identify related assets, assess their performance and finally estimate the maintenance need. The execution management is mostly taken care of by existing EAM systems. To make the transition between the parts seamless, the asset management functions in EAM system should be easily accessable from the integrated system. Because the maintenance processes can be outsourced, the integrated system should be designed so that the organisation boundaries are taken into account. This process can be decomposed to discrete use cases.

The integration system can also enable new use cases. Some of these use cases have been identified by the POSC Caesar Association. The use cases that involve product lifecycle management and as-designed information are outside the scope of this work. The use cases addressed by the integrated system are use case #5, automated O&M configuration, and use case #7, open automatic or semi-automatic CBM triggering [44]. These use cases and the use cases from the condition based maintenance process enhanced by the integrated system are illustrated in Figure 16.

The situation assessment process is started when a notification arrives. The maintenance engineer then proceeds to identify the assets related to the notification. The integrated system should be able to communicate the topology of the area where the situation was detected. This functionality is provided by existing EAM systems and is not enhanced by the integration.

When the related assets are known, the maintenance engineer assesses their performance. The maintenance engineers needs to be able to see information from various sources related to the asset. The CM system is one source for self diagnosis data,

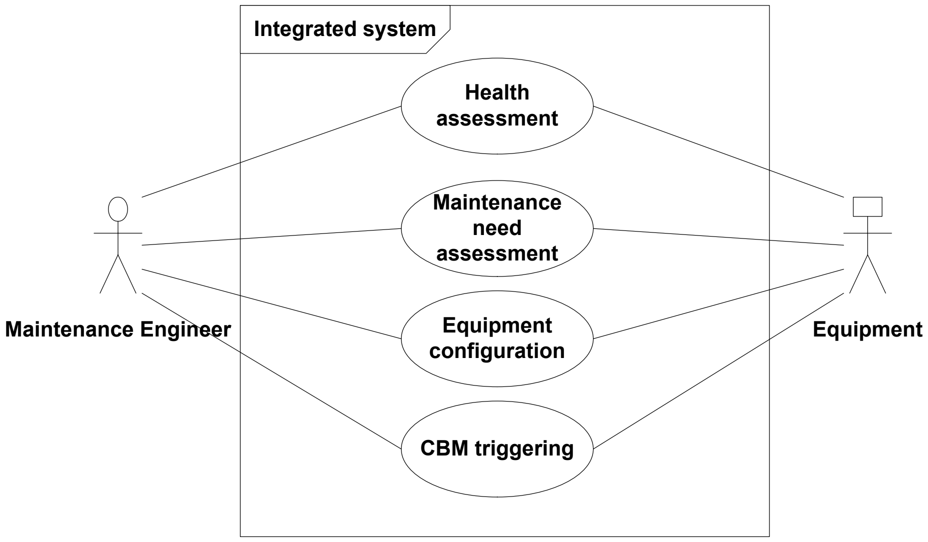


Figure 16: Use cases enhanced by the integration

but data from the process control system (PCS) can also be useful. The integrated system should be able to support this by presenting all information that can be gathered in a single screen to the maintenance engineer. The contents of the view have been defined by ISO 13374 to include at least 5 components: state detection, health assessment, prognosis, recommended actions and identification. Presenting the prognosis, recommended actions and health assessment requires a fairly sophisticated condition monitoring system. If the condition monitoring system cannot produce the data, the task of data analysis is left to the maintenance engineer.

After the performance of related assets is known, the maintenance engineer estimates the need for maintenance. To be able to make this decision, the maintenance engineer needs to know the design parameters and the maintenance history of the asset so the current performance can be compared to the baseline performance. Information on the process measurements from the PCS system can also be valuable here. The integrated system should be able to make this information easily available.

If there is need for maintenance, the maintenance engineer needs to create a work request. This task is often done in the EAM system and thus integrated system does not need to implement this use case. However, the integrated system can support the use case by making the EAM system easily accessible from the integrated system. For example, if the EAM system is web-based, the integrated system can produce a hyperlink to the work order management page of the system.

4.2 Structure of the integrated system

In SOA terms, the integrated system provides composite services. The system provides services that support the condition based maintenance process by processing the data from multiple related sources including the EAM and CM systems. Furthermore, because the services are aligned to the business processes in condition based maintenance, this composite service lies at the business process service layer of the SOA model defined by [17]. The integrated system as part of a SOA architecture is illustrated in Figure 17.

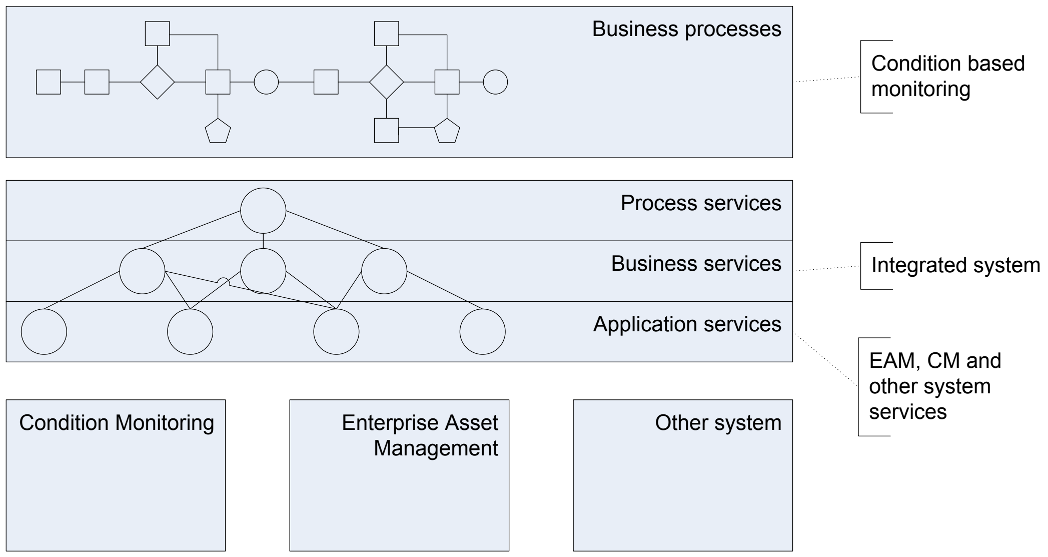


Figure 17: The integrated system as part of SOA architecture

The middleware abstracts the connected systems as a set of services to the rest of the IT architecture. The system can be used directly by implementing a user interface to the services, or the services can be orchestrated by workflows defined at the process service layer. The services provided by the middleware should be designed so that they provide business value, but do not embody any specific business rules. Keeping the business rules separate from the business services increases the organisational agility, because changing the business process will not require re-engineering the services of the integrated system. For example, if the enterprise wants to outsource its maintenance, it does not need to rebuild the integrated system.

The system should provide its services with the Web Services technology stack, because it is the most commonly used technology used in SOA-oriented architectures. Using Web Services allows the integrated system to be connected to other systems using the infrastructure in place in the organisation, such as the enterprise service bus (ESB). The services that the middleware offers to other systems should use a standard message format so that other systems that want to use the services do not need to implement custom transformation logic to connect to the middleware. Unfortunately in maintenance operations management there are no message formats that are supported by a broad range of vendors. If there are no pre-existing message formats already used in the enterprise, then a standard format should be selected such as Tech-XML defined by OSA-EAI.

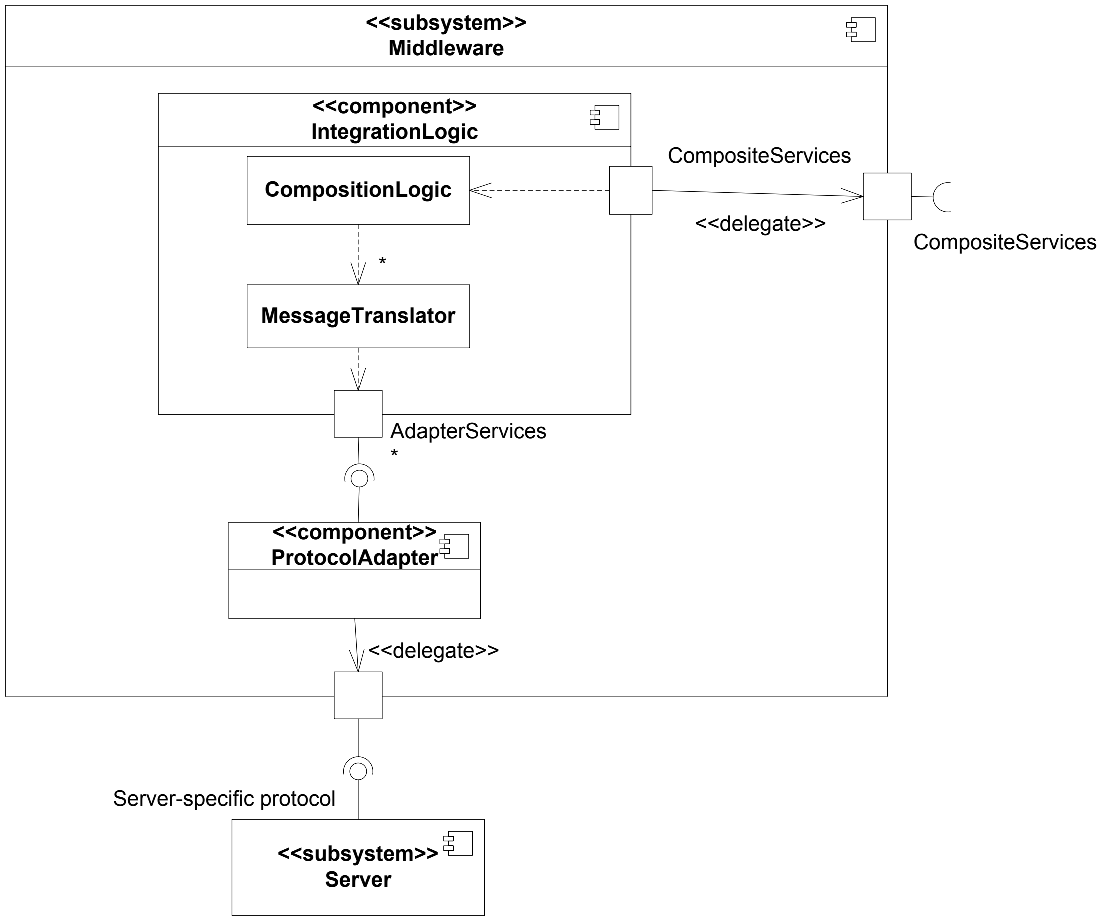


Figure 18: Components, connectors and interfaces

The system, its elements and the connectors are further illustrated in Figure 18. The middleware has two types of external interfaces. It communicates with the connected servers with server-specific protocols and offers their data as a set of composite services to other systems. The middleware communicates with other systems using adapters. Adapters offer the functionality of the connected servers over the interface defined by the adapter framework. This allows the middleware to communicate with the servers without implementing protocol-specific logic for each connection. This design follows the general architecture of WCF, described in section 3.1. Adapters are reusable and can be used in other integration scenarios as well.

It is expected that at least some of the connected servers are OPC UA servers. In that case the protocol-specific adapter must include a OPC UA Client to be able to communicate. The adapter should at least be able to offer the OPC UA Services over the interface defined by the adapter framework. However, the data provided by OPC UA Services is not structured. Read services only return arrays of values. Higher level systems deal with higher level abstractions; for example, XML documents and objects. The adapter should be able to provide the OPC UA objects in a format that is understood by the adapter framework. Following the architecture of WCF, this format should use XML.

A generic WS-* adapter can also be used if the OPC UA Server supports the Web Service mapping. However, such adapters do not understand the semantics of OPC UA and thus their data is not as well refined as the data that is produced by a OPC UA-specific adapter which can benefit from the rich information model of OPC UA.

The middleware must include at least two components: the message translation component and the message composition component. The message translation component translates the messages understood by the adapters to a common message format. Because the adapters have already translated the protocol-specific data to use a well-defined format, the message translation component can focus on the schema translation. If the used format in the adapter framework is XML, then the message translation can be implemented with XSLT.

The documents from multiple sources are combined by the composition logic. The combined document offers the information in a well-defined message format over a WS-* interface. This information can be sent to a program that presents the data to a maintenance engineer or it can be sent to other systems. These other systems can be programmed against the WS-* interface and they do not need to know what systems are involved to produce the required information.

4.3 Dynamic behaviour and interactions in the integrated system

The dynamic behaviour and the interactions of the elements is presented using UML 2.0 sequence diagrams. This viewpoint addresses the concerns of the developer such as how the systems communicate with each other, what kind of messages do they send and what kind of transformations are applied to the messages. The actual communication patterns depend on the capabilities of the connected system. Figure 19 illustrates the interaction between a client accessing a composite service, the integration system and the connected server.

The client sends a service request as a SOAP message to the integration server. The integration logic in the integration server identifies which servers it needs to access to be able to construct the response. When these servers are known, the integration server sends the request messages to the adapters that are connected to the servers that need to be accessed. The adapters access the connected servers using a server-specific protocol.

Data transformations are a essential part of the integrated architecture, because the middleware lies at the boundary between the plant floor and the manufacturing operations management level. The systems at the lower level are assumed to use OPC Unified architecture which uses a meta-model that is essentially a mesh net-

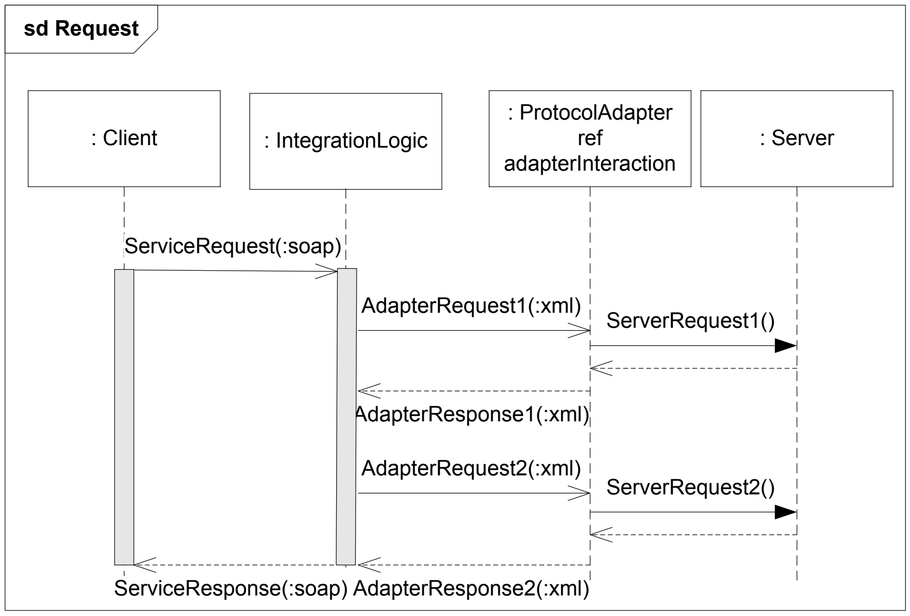


Figure 19: Interaction between the systems

work. The higher level systems, however, tend to use XML-based messaging. The middleware must address this difference in communication styles.

The integration system uses adapters to convert all incoming data to a meta-model that is suitable for further processing. The used meta-model is the XML Infoset, because that is the most used meta-model in data processing in SOA architectures. Because XML Infoset is a hierarchical data model, more complex data models must be transformed so that the data can be represented as a XML document.

4.4 Data transformations

The most important functionality of the integration system is its capability to transform the data so that the connected systems can understand it. The view in Figure 20 illustrates how the data moves through the system from one system to another.

It is essential that the integration system knows how the resources are identified in all connected systems. If the integration system does not know the name of a resource in a certain system, it cannot query for information about that resource. In an ideal situation all systems have the same name for the same resource. When this is not possible, the integration system must be able to resolve the name for any given resource in all connected systems. This can be implemented by having a separate service that can do the name translation.

The adapters in the integration system take care of the meta-model transformation between the meta-model used by the external system to the XML Infoset. The message translation component in the integration system itself transfers the data

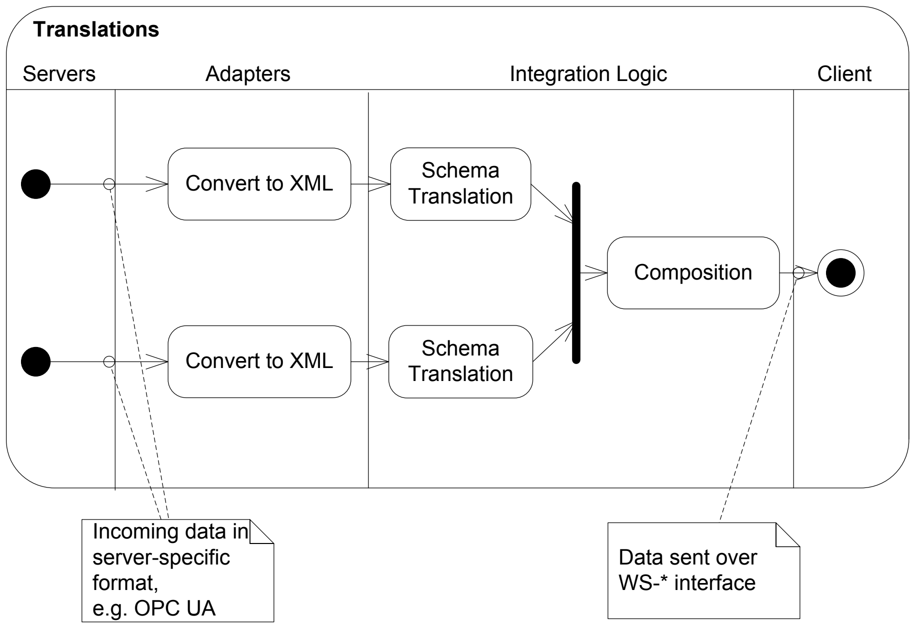


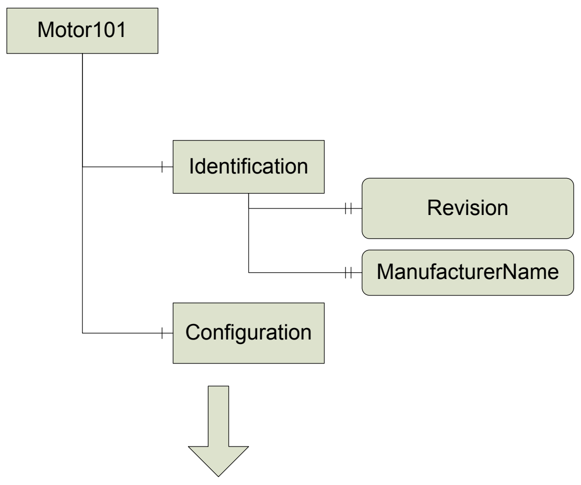
Figure 20: Flow of data between the systems

models using some transformation method, such as XSL Transformations (XSLT). Because the meta-model transformation is done at the adapter level, the integration system can concentrate on transformation of XML documents. This functionality is found in most commercial off-the-shelf integration servers which usually provide a visual tool for mapping XML documents between models. If the XML documents conform to a standard schema, it is possible that the transformation is already implemented and the transformation rules can be reused.

Because Web Services use XML as the message format, there is no need to do any meta-model transformation in the adapter. Thus any Web Service library or framework can be used as an adapter in the integration system. Most integration servers provide functionality to connect to other Web Services, so existing integration servers already provide the required Web Service adapters.

OPC Unified architecture uses a meta-model that is more complex than the meta-model of XML. OPC Unified Architecture defines a set of Services, but they provide data on the node level. There are no services in OPC UA that can be used to read complete objects. Additionally, the Web Service interface of OPC UA is not well supported. At the time of writing, only the C# SDK has implemented it. Thus, even if the OPC UA server supported the Web Service API, the user of these services would have to compose a lot of OPC UA service calls to read an OPC UA object, such as a Device in the OPC UA Device Information Model. The Web Service API of OPC UA is not optimised for high performance so it is unadvisable to send a torrent of Web Service calls every time an object in the address space is accessed.

To facilitate reading data at a higher level of abstraction, the adapter should provide functionality to read complete objects and return them as XML documents. Be-



```
<Motor name="Motor101">
 <Identification>
  <Revision value="1" />
  <ManufacturerName>
   ACME
  </ManufacturerName>
 </Identification>
 <Configuration />
</Motor>
```

Figure 21: Mapping an OPC UA Object to a XML document

cause of the restrictions for references and BrowseNames in OPC UA ObjectTypes, all OPC UA Objects are relatively but not completely hierarchical. The mapping of OPC UA Objects to XML documents is illustrated in Figure 21. In the figure, the OPC UA Object is described using the notation defined in [46]. The user of the adapter should be able to explicitly define what parts of the object type are mapped to XML. Because the object access service is implemented at the adapter, the integration server can communicate using the optimised binary protocol of OPC UA. The service can aggregate the requests so instead of reading every node separately, the read request can include all nodes that belong to the requested object. Mapping simple nodes to XML documents can be done according to the rules defined by OPC UA mapping specification.

4.5 Rationale

The proposed design can be connected to the rest of the IT infrastructure of the organisation relatively easily, because it presents its data using a standard XML messaging format over a Web Service interface. Web Service interface was selected, because it is assumed that the services of the integrated system are mostly used by higher level systems. OPC Unified Architecture is mostly suited to the systems operating at a lower level, so it was not used in this case.

The middleware aggregates data from variety of sources. The adapter architecture and the meta-model transformations done in adapters allow commercial off-the-shelf integration servers to be used. Such servers already provide the required functionality such as message routing and transformations. It is also possible to implement a custom integration system in which case same adapters can be used. The adapters can be reused elsewhere even for point-to-point integration purposes. To use adapters successfully, the used platform must specify a standard adapter interface. Such specifications already exist - for example in .NET platform the adapters can be implemented as WCF channel stacks by using the WCF adapter SDK. The proposed design is meant to be closely aligned with the communication model of WCF.

Because the adapters provide all data in XML, the data model transformations can be done using existing technology such as XSLT. Integration servers provide visual tools to create XSLT transformation rules. If an integration server is not used, there exists multiple high-grade libraries to do XML transformations.

Because the meta-model transformation is delegated to the adapter, the transformation of the data model and the meta-model are kept separate. It is enough to do the meta-model transformation only in the adapter instead of doing it every time for every data model transformation.

The OPC UA adapter supports configuration of the meta-model transformation per OPC UA ObjectType. These configurations can be reused. If the transformation from OPC UA Device Information Model to XML is configured once, then the same configuration can be used in all other places where the same information model is used. These configurations can be reused in other projects where the same information model is encountered.

4.6 Summary

In this chapter, a design that applied OPC Unified Architecture to integrate IT systems involved in condition based maintenance was presented. The chapter focused on providing the information from an OPC UA address space to IT systems that had interfaces based on the Web Service technology stack. The integration was facilitated by utilising the OPC UA object model to map the data from the OPC UA server to the meta-model used by XML.

5 Implementation and testing

The proposed design was evaluated by implementing it on the integration research platform that was developed in the laboratory during this thesis. The chapter begins with a description of the integration research platform concentrating on the maintenance operations in the platform and then proceeds to describe the implementation of the design described in the earlier chapter.

The implementation applies the CIR and MIMOSA OSA-EAI standards to support the integration, but the intent is not evaluate these standards. The purpose of the implementation is to demonstrate the feasibility of the design and discover properties of using OPC UA with the selected technologies. Due to the nature of the test platform, performance and scalability of the implementation were not tested. The chapter concludes with the description of the test scenarios used to validate the implementation. Finally, the communication between the integrated systems is illustrated with an example use case.

5.1 Integration research platform

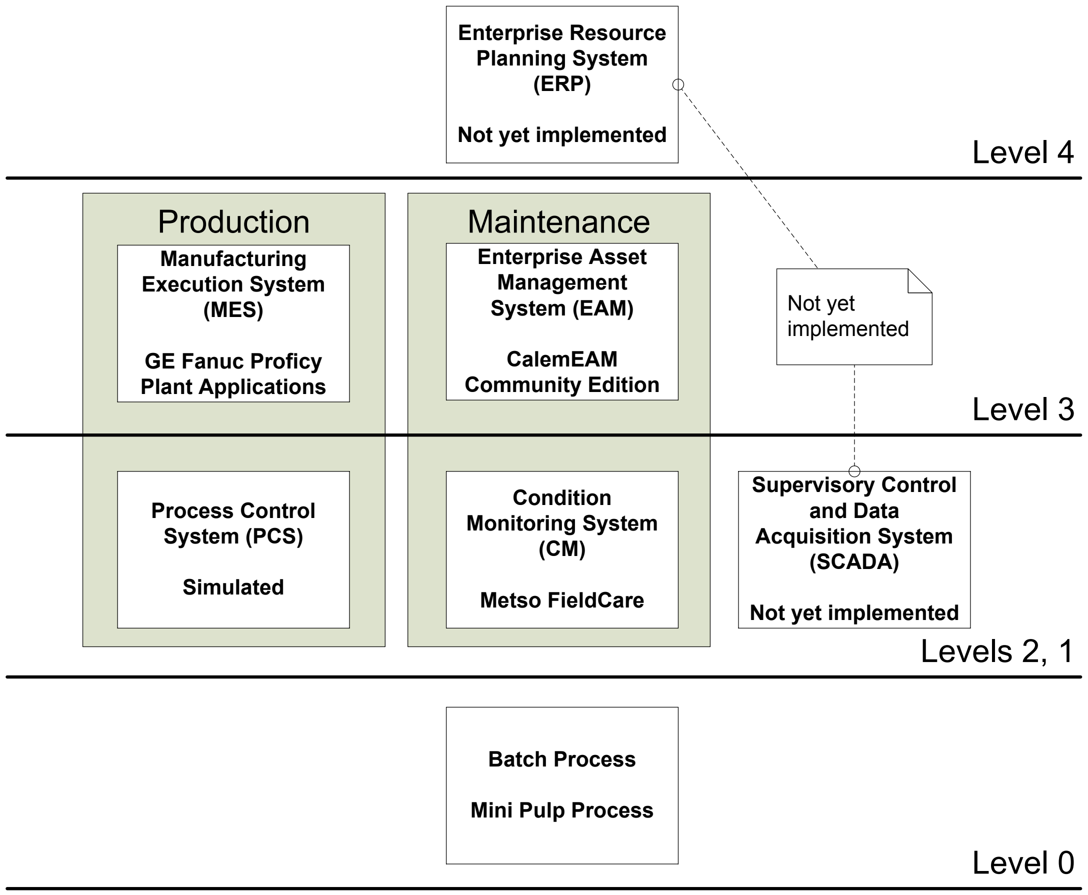

Enterprise Resource Planning System (ERP)
Not yet implemented
Level 4
Production
Manufacturing Execution System (MES)
GE Fanuc Proficy Plant Applications
Maintenance
Enterprise Asset Management System (EAM)
CalemEAM Community Edition
Not yet implemented
Level 3
Process Control System (PCS)
Simulated
Condition Monitoring System (CM)
Metso FieldCare
Supervisory Control and Data Acquisition System (SCADA)
Not yet implemented
Levels 2, 1
Batch Process
Mini Pulp Process
Level 0

Figure 22: Integration research platform

During the course of this work, a part of an integration research platform was implemented in the laboratory for research and teaching purposes. The platform is meant to represent a subset of an IT architecture in a manufacturing plant, focusing on operations and maintenance. The platform divided to the levels of ISA-95 functional hierarchy model is illustrated in Figure 22. The maintenance parts of the platform were implemented for this work.

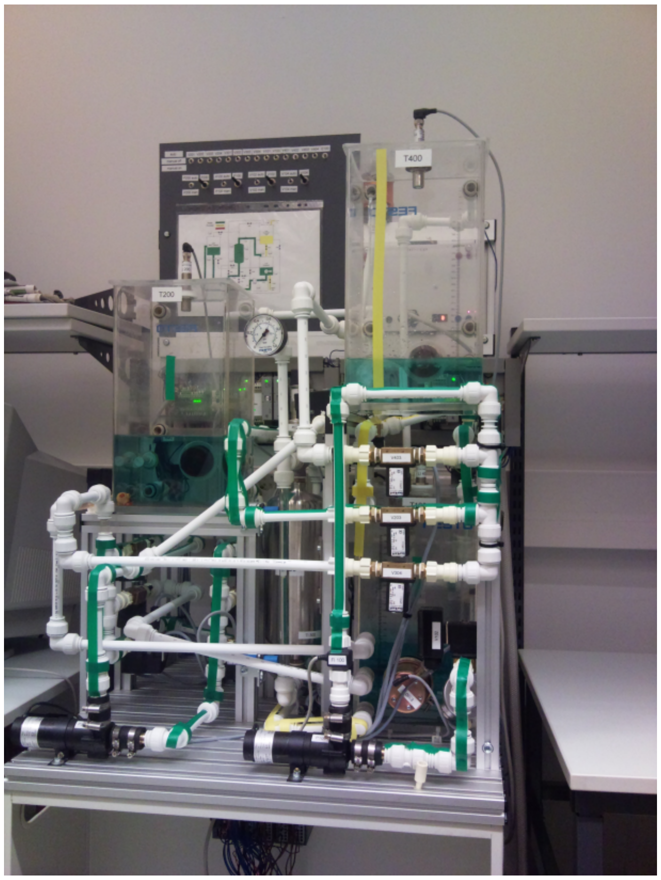


Figure 23: Mini Pulp Process (MPP)

The research platform is based on system "Mini Pulp Process (MPP)" which models a pulp batch cooking process. MPP is pictured in Figure 23. Pulp batch cooking process converts wood chips to a batch of dissolved wood fibres, but due to practical issues the implemented equipment just moves water. MPP was implemented as a part of an earlier master's thesis done in the laboratory [47]. The IT systems in the research platform model the equipment and processes of this system.

The platform can be divided to two parts: the production part and the maintenance part. The maintenance part was implemented during this work and is explained in more detail later in this section, while the production part was implemented simultaneously as a part of an another master's thesis. The process is controlled by a process control system (PCS). At this stage of the project, the PCS system is not connected to the actual process but rather simulates the process execution. The production management (scheduling, dispatching, quality control and other activities in manufacturing operations management) is done with a manufacturing execution system (MES). The interaction between the PCS and the MES systems are based on the ISA-95 and the ISA-88 batch production standards. [48].

The integration research platform is not yet complete. The MES system is to be connected to a enterprise resource planning (ERP) system which is responsible for the enterprise level activities. The production process is to be monitored and supervised with a supervisory control and data acquisition (SCADA) system. The SCADA system connects to the processes control system and provides a interface for plant operators to see the state of the process and control it if necessary.

5.1.1 Maintenance operations integration in the platform

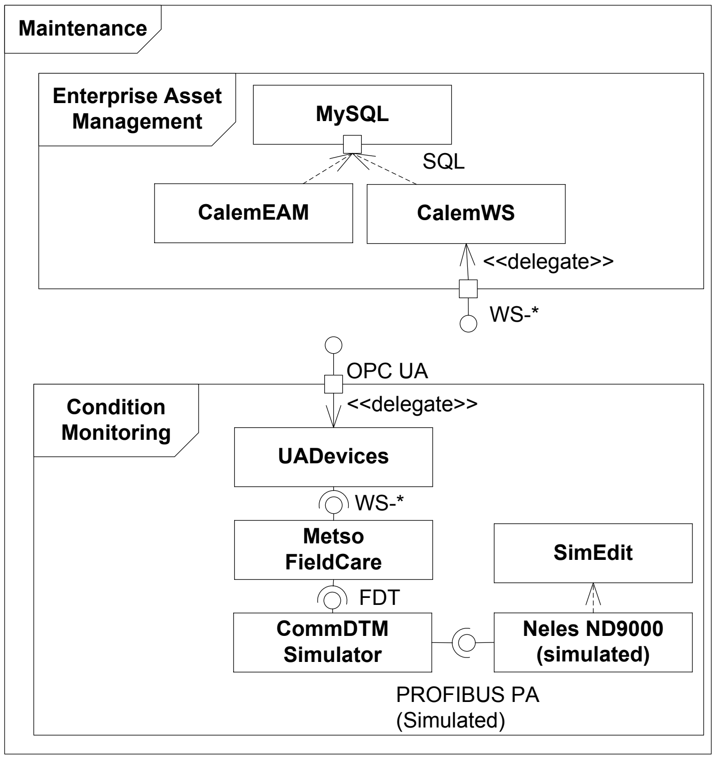


Figure 24: Maintenance part of the research platform

The maintenance systems in the platform, illustrated in Figure 24, consists of the simulated valves, the condition monitoring system and the enterprise asset management system. The test platform provides an OPC UA interface to simulated condition data of valve controllers. The communication between involved systems is illustrated in Figure 25.

The condition monitoring system, Metso FieldCare, is configured to use a communication DTM simulator for PROFIBUS PA field bus. The communication DTM simulator is configured to use data dumps (ND9000_1.txt and ND9000_2.txt) of a

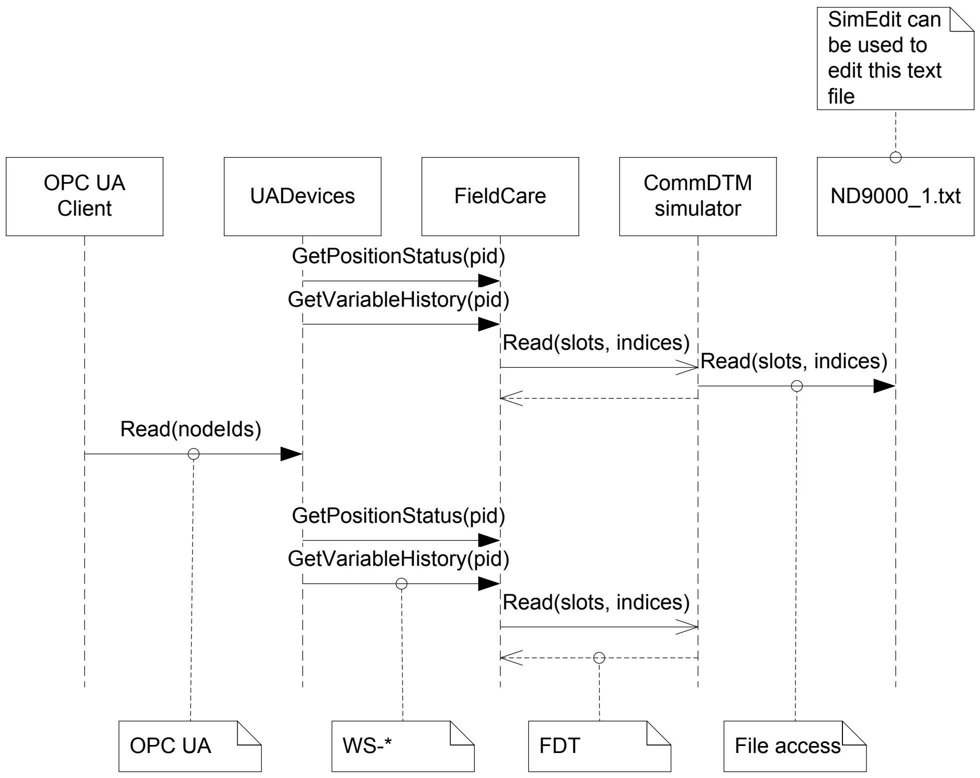


Figure 25: Accessing valve condition data over OPC UA

Neles ND9000P valve controller to simulate a connection to two of such valve controllers. Neles ND9000P is an intelligent valve controller equipped with self diagnosis features. The valve controller can trigger warnings and alarms when it is operating outside the specification. Maintenance engineers can use Metso FieldCare or a generic FDT frame application to see the self-diagnosis of the valve controller and predict when maintenance is required. A separate program "SimEdit" was developed that can be used to modify the simulated data of the valve controller so that events, such as alarms and warnings, can be generated.

A SOAP server module was installed to Metso FieldCare, which makes the condition monitoring data available over a Web Services interface. One of the objectives of this work was to evaluate how OPC UA can be used to deliver condition data, so a separate OPC UA interface had to be developed because FieldCare does not have support for OPC UA. A separate OPC UA server "UADevices" was developed that accesses the condition monitoring data stored in FieldCare over the SOAP server module. UADevices polls FieldCare to update its address space regarding the monitored devices every 10 seconds.

The data was modelled in the OPC UA address space using the OPC UA Device Information Model. The information model does not define how condition data should be represented, so a subset of the data was made available in the ParameterSet folder of the device as an array. Because the Alarms & Events information model of OPC UA was not complete during this work, it was not used to model the alarms and warnings of the device.

The enterprise asset management software used is CalemEAM community edition. CalemEAM community edition was selected, because its source code is available and there are no licensing fees. CalemEAM community edition had no documented integration API, so a separate application “CalemWS” was developed that provides a set of Web Services that can be used to access the database of the EAM system. CalemEAM uses MySQL to store its data, so CalemWS can directly connect to the database and provide its data over a Web Service interface.

5.2 Implementation of the design

An user interface was developed to the maintenance engineer that provides functionality for a subset of the use cases in condition based maintenance. The implemented functionality was selected so that it covers the requirements as well as possible.

The implemented system and its connection to the integration research platform is illustrated in Figure 26. The user interface is connected to a middleware that provides services to support the use cases. The implemented services conform to the WSDL definitions in the OSA-EAI standard. OSA-EAI was selected, because the pre-existing IT systems in the integration platform did not already use any specific messaging format. OSA-EAI has well-defined message formats and interfaces for asset management.

Only the services that are required to support the use cases were implemented on the middleware. The middleware was implemented on the .NET 3.5 platform, because at the time of writing, the OPC Foundation’s OPC UA C# SDK had the best support for OPC Unified Architecture specifications. Furthermore the adapter architecture was designed to be compatible with the WCF architecture.

Because the assets in the research platform use different tags in the condition monitoring system and the asset management system, a separate server was developed that can do the tag translation. The separate server, implemented with C#, nHibernate and Microsoft SQL Server Express, follows the CIR specification. In the interactions where the integration system needs to know the names of a given resource, the CIR server is contacted and the list of names is requested. Every asset in the integration platform has at least 3 tags: one tag in the condition monitoring system, one tag in the EAM system and one tag (GUID) in the CIR server. The CIR server database was configured so it can link all the other tags with the GUID tag in the CIR server. If other systems are added to the integration platform that use yet another naming scheme for the assets, the CIR server database can be updated.

The OPC UA adapter was implemented with the WCF Line-of-Business Adapter SDK and OPC Foundation OPC UA SDK. WCF adapters can be used directly by .NET applications that use the WCF framework, or they can be used in Microsoft Biztalk integration server solutions. A set of OPC UA services were implemented in the adapter, and an additional service “UA2XML”. UA2XML is used to convert

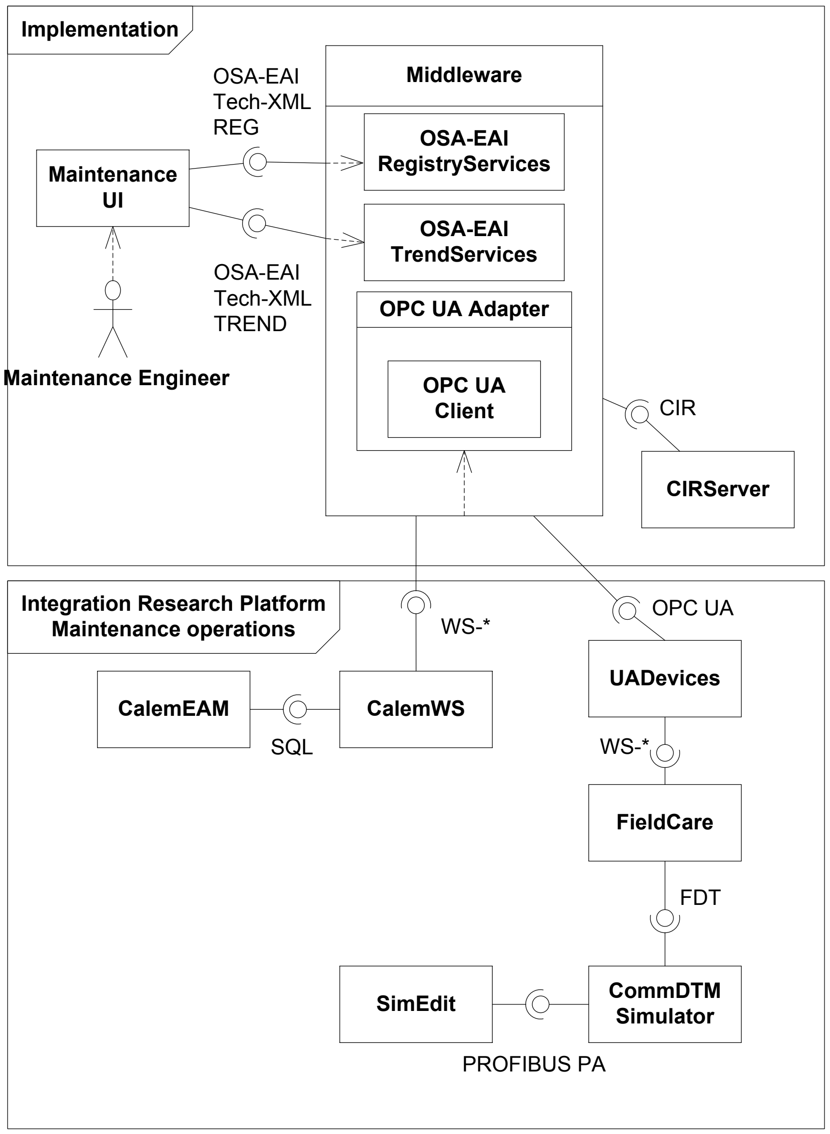


Figure 26: Implemented system and the maintenance operations in the integration research platform

OPC UA objects to XML documents. The service can be configured with XML files that contain declarative rules on how to convert an OPC UA object with given ObjectType to an XML document fragment. The intent is that the developer can look at the object type in the OPC UA server and then configure the adapter to provide the needed data as an XML document.

The configuration file format is inspired by XSLT. The format is essentially a set of XML fragment templates that are applied recursively to obtain a complete XML document. A simplified configuration file that defines rules to convert a valve object to XML is presented in Listing 1.

Listing 1: Configuration for converting valve objects to XML

```
<Match type="ValveType">
  <Valve>
    <Name value-of="@DisplayName" />
    <ModelName value-of="3:FactorySettings/3:Model" />
    <Parameters>
      <Stiction value-of="3:ParameterSet/3:Stiction"  />
    </Parameters>
    <Alarms value-of="3:Status/3:Alarms" />
  </Valve>
<Match>
```

The templates can also be written against a BrowseName so they can be used with generic object types. The implemented directives essentially mirror XSLT elements apply-templates, template (Match in the implemented format) and value-of (implemented as an attribute). These were enough to implement the transformation of OPC UA Device Information Model to XML. XSLT was chosen as the basis because it is the most used XML transformation language. Because system integrators are usually familiar with XSLT, the barrier of entry is low for writing OPC UA metamodel transformations.

A configuration file was created for the adapter that describes how to transform OPC UA Devices to XML documents. With this configuration, it is possible to read the state of a valve controller in the condition monitoring system with a single service call. When the user interface calls the UA2XML service of this adapter, the adapter knows what nodes it should read from the OPC UA Server and how the result should be presented. The result of this call is an XML document which the user interface presents to the user of the application. Because the OPC UA Device information model does not specify how condition data should be presented, the user interface just prints all the data to the screen without further analysis - the user interface application does not know the meaning of the data.

5.3 Testing

The integrated system was tested with a scenario that was identified during the requirements identification process. The communication sequences in other scenarios are similar, and the transferred data from the OPC UA server is identical which is why the testing concentrates on just one scenario.

In the scenario, an impending valve failure causes anomalies in the production process. The anomaly is detected by the operator who notifies the maintenance engineer. The parts of this scenario that are relevant to the integrated system were simulated. The parts where the maintenance engineer interacts only with a single system were omitted.

The maintenance engineer after receiving the notification must discover what equipment are involved in the part of the production process that had anomalies. It is expected that the EAM and the PCS system can provide this information. After the equipment is known, the maintenance engineer needs to assess the health of the involved equipment. When the health of the equipment is known and a prognosis can be made, a work request can be generated to the EAM system. The health assessment part of the process is illustrated next.

5.3.1 Health assessment use case

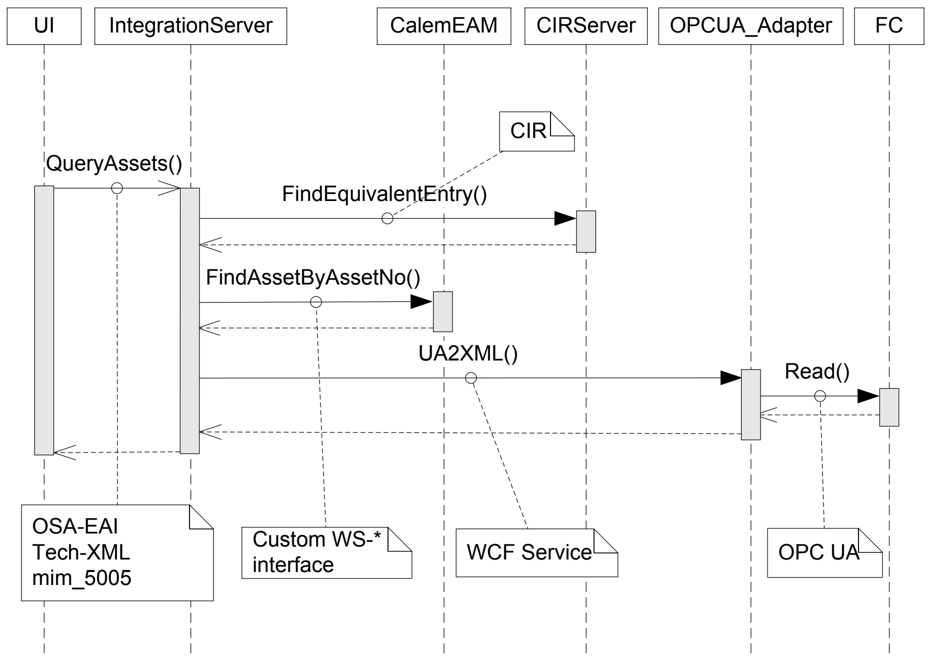


Figure 27: Interaction between the servers

Figure 27 illustrates the interaction between the servers involved. First the maintenance engineer accesses the user interface and enters the tags of the equipment involved. In the test these tags are V101 and V102. The user interface sends an OSA-EAI Tech-XML QueryAssets message to the integration server. The integration server queries the CIR server to find out what systems store information regarding the assets that were in the query. The reply tells the integration server that some of the equipment are monitored by the condition monitoring system. The reply also includes the tag names used by the condition monitoring system for the equipment. These tag names also function as the Node ID:s for the equipment in the OPC UA address space. Because the query used the asset names in the EAM system, the integration system can directly query the EAM system for additional information regarding the equipment.

Next the integration system queries the OPC UA Server of the condition monitoring system. The integration system calls the UA2XML service with the Node ID:s of the equipment. The adapter reads these Node ID:s from the server, notices that the nodes are Objects that have the Valve ObjectType and finally converts the objects to

XML according to the rules of its mapping file. The integration system converts the XML document to a format that is suitable for the QueryAssets response document.

Finally the document in the new format is combined with the response received earlier from the EAM system. Part of the response is shown in Listing 2. The response is sent back to the user interface and it can display the basic information about the equipment. The maintenance UI still needs to issue a few more OSA-EAI messages from the TREND technology type to discover if there are alarms or warnings active and to read the readings of the diagnostic sensors (such as temperature and stiction). The communication sequences for these queries are identical to the one demonstrated earlier.

Listing 2: Part of the mim_5005 QueryAssets response

```
<s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <QueryAssetsResponse
     xmlns="http://www.mimosa.org/TechXMLV3-2/
         RegistryManagementServices">
    <QueryAssetsResult
       xmlns="http://www.mimosa.org/TechXMLV3-2">
      <ack>
        <row>
          <asset serial_number="00001001"
                 as_type_code="0"
                 user_tag_ident="404-00001001"
                 name="404-00001001"
                 as_db_id="0"
                 asset_id="1"
                 as_db_site="BEEFF00D00000000"
                 cs_type_db_site="0000000000000000">
          </asset>
          <asset_chr_data
             data_value="Simulated valve number 1."
             eu_type_code="241"
             eu_db_id="0"
             ac_db_site="0000000000000000"
             asset_id="1"
             eu_db_site="0000000000000000"
             ac_type_code="0"
             ac_db_id="0">
          </asset_chr_data>
        </row>
      </ack>
    </QueryAssetsResult>
  </QueryAssetsResponse>
</s:Body>
```

5.4 Summary

In this chapter, the test platform that was used to facilitate the implementation of the design and the actual implementation were described. The communication patterns between the integrated systems were illustrated with an example scenario. The implemented interfaces could provide the data necessary to provide the necessary data to the maintenance engineer.

6 Conclusions and discussion

In this thesis, a design for integrating IT systems involved in condition based maintenance based on OPC Unified Architecture and Service Oriented Architecture was presented. An experimental implementation was constructed to integrate enterprise asset management and condition monitoring systems. The implementation was evaluated with a test scenario identified by analysing the processes of condition based maintenance. The experimental implementation applied the MIMOSA OSA-EAI and OpenO&M CIR specifications to support the integration.

6.1 Conclusions

A design for integrating IT systems and devices for condition based maintenance was presented in this work. In the design, adapters are used to communicate with condition monitoring systems, intelligent devices and other systems involved for producing the data regarding the health of the equipment. Separating the adapters from the rest of the integration logic allows the adapters to be reusable in other integration solutions as well. In the presented design, the protocol-specific data is transformed to XML format before passing it to the integration logic. This allows the integration logic to concentrate on transformations of XML data, regardless of the data models used by the connected systems. The experimental implementation demonstrated the feasibility of this approach in the integration of relatively small amount of IT systems.

The work demonstrated that the abstractions of OPC Unified Architecture simplify the design and implementation of the integration. Information models such as the OPC UA Device Information Model allow developers to access device identification data in a standardised manner. The object model of OPC UA allows integrators to implement their solution according to the type definitions in the address space. However, the OPC UA Device Information Model was found to be insufficient to model condition data. However, it can serve as a basis for an extended data model for presenting such data.

The separation of the services and mappings of OPC UA allow it to be used in service based frameworks. The implementation demonstrated this with the Windows Communication Foundation. This allows OPC UA to be used with integration servers based on WCF, such as Microsoft Biztalk, if an adapter for the protocol is implemented. In this work, a proof of concept version of the adapter was developed. This allows integration servers and solutions based on WCF to communicate directly with OPC UA servers that do not support the Web Service mapping.

The web service interfaces of MIMOSA OSA-EAI Tech-XML SOAP were used to present the device data to other IT systems. Because OSA-EAI defines its own identification numbering for assets, a separate service was required to map the entity names in the integration solutions. The OpenO&M CIR specification was sufficient to implement this mapping support for the purposes of the experimental implementation.

Presenting the asset data using the OSA-EAI Tech-XML elements is difficult if an OSA-EAI CRIS database is not used. The message schemas are based on the CRIS data model, which is relational. The keys and foreign key references are present even in the XML documents and constructing them is not straightforward if the data is not backed by an actual relational database. A CIR server can facilitate this integration by mapping OPC UA object types to keys that refer to the OSA-EAI reference data.

6.2 Discussion and proposals for future work

OPC UA is still a relatively new specification and there exists little research on how to utilise it for data integration. The security, the performance and platform independence of OPC UA already offer benefits over the older OPC standards. However, the information modelling tools of OPC UA can offer even larger benefits. OPC UA gives powerful tools to model complex domain-specific data and present it in a standard way. If more domain-specific information models are developed and are adopted by the vendors, the integrators can integrate the systems in a vendor-independently. This will reduce the cost of integration. On the other hand, if the modelling tools are ignored, they remain a source of unnecessary complexity and can hinder the adoption of the standard.

OPC Unified Architecture can be used to present condition data from intelligent devices, but the currently available information models do not define how this data should be made available. Extending the Device Information Model to present condition data in a standard manner could be a reasonable topic for further research. It is expected that the work on FDI can provide input to this research. Also, bridging the gap between MIMOSA OSA-EAI asset model and OPC UA device model could ease the use of OSA-EAI web service interfaces together with OPC UA data sources.

This work tested the proposed design in a rather small scale. The design should be evaluated in a more realistic IT environment to evaluate its scalability and performance with larger number of systems. The proposed approach of converting OPC UA objects to XML using a declarative template language appears promising for converting simple data models such as OPC UA DI. The feasibility of the approach should be tested with more complex data models such as the PLCOpen and Analyser Device information models.

References

[1] Suomen Standardisoimisliitto SFS, Helsinki, Finland, *SFS-EN 13306:2001 – Maintenance terminology*, 2001.

[2] The Instrumentation, Systems and Automation Society, Research Triangle Park, NC, USA, *ANSI/ISA-95.00.03-2005 – Enterprise Control System Integration – Part 3: Activity Models of Manufacturing Operations Management*, 2005.

[3] International Organization for Standardization, Geneva, Switzerland, *ISO 13374-1:2003: Condition monitoring and diagnostics of machines – Data processing, communication and presentation – Part 1: General guidelines*, 2003.

[4] M. Bustamante, *Learning WCF.* Sebastopol, CA, USA: O'Reilly Media, Inc., 1st ed., 2007.

[5] Microsoft, "Data transfer architectural overview." Available at http://msdn.microsoft.com/en-us/library/aa347789.aspx, accessed on 2010-03-04.

[6] W. Mahnke, S.-H. Leitner, and M. Damm, *OPC Unified Architecture.* Berlin, Germany: Springer, 2009.

[7] D. Grossmann, K. Bender, and B. Danzer, "OPC UA based Field Device Integration," in *SICE Annual Conference*, pp. 933–938, 20-22. Aug 2008.

[8] Machinery Information Management Open Standards Alliance, "MIMOSA's open systems architecture for enterprise application integration (OSA-EAI) technical architecture summary." Available at http://www.mimosa.org/sites/default/files/TechDocs/OSA-EAI_Technical_Architecture_Summary_Dec_2006.pdf, accessed on 2010-05-06, 2007.

[9] J. Järviö, "Mitä on kunnossapito," in *Kunnossapito* (J. Järviö, ed.), pp. 11–25, Hamina, Finland: KP-Media Oy, 2006.

[10] J. Järviö, "Ehkäisevä kunnossapito," in *Kunnossapito* (J. Järviö, ed.), pp. 66–77, Hamina, Finland: KP-Media Oy, 2006.

[11] G. M. Knapp and B. Wang, "Fundamentals of maintenance," in *Computer-Aided Maintenance Methodologies and practices* (J. Lee and B. Wang, eds.), Manufacturing Systems Engineering Series, pp. 3–19, Dordrect, The Netherlands: Kluwer Academic Publishers, 1999.

[12] J. Järviö, "Vikaantuminen," in *Kunnossapito* (J. Järviö, ed.), pp. 48–66, Hamina, Finland: KP-Media Oy, 2006.

[13] The Instrument Society of America, Research Triangle Park, NC, USA, *ISA-95.00.01-2000 – Enterprise-Control System Integration – Part 1: Models and Terminology*, 2000.

[14] B. Stengl and R. Ematinger, *SAP R/3 plant maintenance: making it work for your business*. Harlow, Great Britain: Pearson Education, 2001.

[15] CalemEAM Inc., "CalemEAM website." http://www.calemeam.com, accessed on 05-05-2010.

[16] D. S. Linthicum, *Next Generation Application Integration: From Simple Information to Web Services*. Boston, MA, USA: Addison-Wesley Professional, 2003.

[17] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2005.

[18] N. Josuttis, *SOA in Practice*. Sebastopol, CA, USA: O'Reilly Media, Inc., 1st ed., 2007.

[19] D. Krafzig, K. Banke, and D. Slama, *Enterprise SOA: Service-Oriented Architecture Best Practices*. Upper Saddle River, NJ, USA: Prentice Hall, 2004.

[20] I. Seilonen, J. Olli, M. Rintala, and K. Koskinen, "Application integration for condition monitoring: A case study with installed base information," in *Proceedings of the 21st International Congress and Exhibition Condition Monitoring and Diagnostic Engineering Management (COMADEM) 2008*, 11-13. Jun 2008.

[21] Mtelligence Corporation, "Mtelligence MIMOSA interop server (MIS) datasheet." Available at http://www.mtelligence.net/downloads/default.aspx. Accessed on 2010-04-29.

[22] G. Cena, L. Durante, and A. Valenzano, "Standard field bus networks for industrial applications," *Computer Standards & Interfaces*, vol. 17, no. 2, pp. 155–167, 1995.

[23] PI, "PROFIBUS PA system description." Available at http://www.profibus.com/nc/downloads/downloads/profibus-pa-technology-and-application-system-description/download/191/, accessed on 2010-05-06.

[24] FDT Joint Interest Group, Diegem, Belgium, *FDT Interface Specification Version 1.2*, 2001.

[25] M. Nelson, "Using Distributed COM with firewalls." Available at http://msdn.microsoft.com/en-us/library/ms809327.aspx?ppud=4, accessed on 2010-05-06, 1998.

[26] M. Wasznicky, "Using Web Services instead of DCOM." Available at http://msdn.microsoft.com/fi-fi/library/aa302336(en-us).aspx, accessed on 2010-05-06, Feb. 2002.

[27] OPC Foundation, Scottsdale, AZ, USA, *OPC Unified Architecture Specification – Part 1: Overview and Concepts 1.01*, 2009.

[28] OPC Foundation, Scottsdale, AZ, USA, *OPC Unified Architecture Specification – Part 4: Services 1.01*, 2009.

[29] OPC Foundation, Scottsdale, AZ, USA, *OPC Unified Architecture Specification – Part 6: Mappings 1.00*, 2009.

[30] OPC Foundation, "Opc unified architecture website." Available at http://www.opcfoundation.org/UA, accessed on 2010-05-06.

[31] OPC Foundation, Scottsdale, AZ, USA, *OPC Unified Architecture for Devices (DI) 1.00 Companion Specification*, 2009.

[32] OPC Foundation, Scottsdale, AZ, USA, *OPC UA Companion Specification for Analyzer Devices 1.00*, 2009.

[33] PLCopen, "PLCopen and OPC foundation combine their technologies." Available at http://www.plcopen.org/pages/tc4_communication/, accessed on 2010-05-06.

[34] ABB, "ABB supports new industry-wide collaboration for accelerated Field Device Integration." Available at http://www.abb.fi/cawp/seitp202/8749114d6534b9688525766b00544560.aspx, accessed on 2010-02-17.

[35] OPC Foundation, "OPC UA certification road map." Available at http://www.opcfoundation.org/Default.aspx/Compliance-Certification/roadmap-ua.asp?MID=Compliance, accessed on 2010-05-06.

[36] K. Bever, "OpenO&M Information Service Bus and CIR." Available at http://www.openoandm.org/files/OpenO&M%20Information%20Service%20Bus%20and%20CIR%2010-16-2009.pdf, accessed 2010-03-09.

[37] Machinery Information Management Open Systems Alliance, Tuscaloosa, AL, USA, *Tech-XML-Services Client & Server Specification Version 3.2.1*, Dec. 2008.

[38] Penn State University / Applied Research Laboratory, The Boeing Company, and Machinery Information Management Open Standards Alliance, "Open systems architecture for condition-based maintenance (OSA-CBM) primer," 2006.

[39] OpenO&M, Research Triangle Park, NC, USA, *OpenO&M Common Interoperability Registry Specification draft V0.6*, 2009.

[40] T. Hadlich, "Providing device integration with OPC UA," in *IEEE International Conference on Industrial Informatics (INDIN) 2006*, pp. 263–268, 16-18. Aug 2006.

[41] R. Huang and F. Liu, "Research on opc ua based on electronic device description," in *3rd IEEE Conference on Industrial Electronics and Applications (ICIEA) 2008*, pp. 2162–2166, 3-5. Jun 2008.

[42] L. Moreira, S. Souza, P. Spiess, D. Guinard, M. Köhler, S. Karnouskos, and D. Savio, "SOCRADES: A web service based shop floor integration infrastructure," in *Internet of Things 2008 Conference*, pp. 50–67, 26–28 Mar. 2008.

[43] J. Mathew, J. Kennedy, L. Ma, A. Tan, and D. Anderson, "A Review of the MIMOSA OSA-EAI Database for Condition Monitoring Systems," in *1st World Congress on Engineering Asset Management (WCEAM) 2006*, pp. 837–846, 11-17. Jul 2006.

[44] K. Bever, "Oil & Gas/PetroChem Industry OpenO&M Interoperability Use Cases & Scenarios." Available at `https://trac.posccaesar.org/attachment/wiki/IOHN/InformationDissemination/Oil%26Gas-PetroChem%20Industry%20OpenO%26M%20Interoperability%20Use%20Cases%20--%20Complete%20(updated%202008-08-22).ppt`, accessed on 2010-05-06, Sept. 2008.

[45] The Institute of Electrical and Electronics Engineers, Inc., New York, NY, USA, *IEEE Std 1471-2000: IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*, 2000.

[46] OPC Foundation, Scottsdale, AZ, USA, *OPC Unified Architecture Specification – Part 3: Address Space Model 1.01*, 2009.

[47] P. Arrenius, "Evaluation environment for new technologies in automation design," Master's thesis, Helsinki University of Technology, Department of Automation and Systems Technology, Espoo, Finland, 2006.

[48] J. Virta, "Application integration for production operations management using OPC Unified Architecture," Master's thesis, Aalto University School of Science and Technology, Department of Automation and Systems Technology, Espoo, Finland, 2010.