Engineering Virtual Domain-Specific Service Platforms

Specific Targeted Research Project: FP7-ICT-2009-5 / 257483

Description of Feasible Case Studies

Abstract

This document gives a detailed overview of the case studies planned as part of INDENICA.
The case studies will consist of an Integration Scenario of a Warehouse Management platform, a Remote Maintenance platform and a Yard Management platform as main scenario plus a number of small scenarios to cover the whole scope of the INDENICA tools.

Document ID:	INDENICA – D5.1
Deliverable Number:	D5.1
Work Package:	5
Type:	Deliverable
Dissemination Level:	PU
Status:	final
Version:	1.0
Date:	2011-07-31
Author(s):	SIE, SAP, TEL

Project Start Date: October 1st 2010, Duration: 36 months

Version History

1.0	31. Jul 2011	Final version

Document Properties

The spell checking language for this document is set to UK English.

Table of Contents

1 Objectives of the Case Studies within INDENICA

The INDENICA project is structured into several work packages. Four of them (WP 1 - 4) are related to the actual technical work of the project. In order to provide realistic requirements and a working back drop for these work packages WP5 provides a set of case studies. This document describes the case studies of the INDENICA project in more detail.

The objectives of work package 5 can be outlined as follows:

- Define realistic requirements and challenges for the technical work
- Collect data about the applicability of the solutions that are developed in the technical work packages
- Analyze and evaluate the collected data
- Draw conclusions from the case study setup and the collected data

Realistic requirements are key to a successful development of research prototypes. The academic partners responsible for those prototypes can rely on assumptions that are very close to reality, which prevents the development to move in a direction where later results may become useless.

The industrial partners will also collect data from the ongoing development and perform monitoring tasks to guide the actual development process and to prevent the people from losing sight of the original research purpose. Main tasks in this context are tests for the feasibility of the tools that are developed in the INDENICA project (as INDENICA is a project that is heavily focused on tool development). This will be carried out as an iterative process which involves regular feedback loops among the industrial and the academic partners. Based on the collected data the industrial partners will finally draw conclusions from the work performed in the technical work packages.

Beside the supervision and the technical steering the case studies will also form the basis for later exploitation of results of the project, as those results can be presented in a much more illustrative way. In addition this will also give deeper insights for people outside the project, such as reviewers, interested researchers or other parties.

2 Overview of the Main Scenario

2.1 Introduction

2.1.1 High-level Overview

The aim of INDENICA is to provide means for building virtual platforms for large systems spanning different application areas as well as different levels of the automation pyramid. The main scenario that is used as a case study to demonstrate the feasibility of the approach is based on a well-known example from the industry automation area – a Warehouse Management System (WMS).

This section introduces the case study starting from a high-level view diving into some details needed to understand the approaches chosen in the corresponding subsections.

Figure 1 illustrates a large warehouse from an external point of view exposing the visually recognizable parts. It shows the trucks on the yard to be loaded and unloaded, the staplers carrying the goods, and the racks with the storage units, which are picked and placed by automated stapler cranes. Furthermore, there are conveyers used to carry the storage units from the stapler cranes at the loading / unloading zone to the automated stapler cranes to store goods back in the high rack and other conveyers to retrieve storage units from the high rack to compile the load of a truck at a specific packing place.

Some parts of the main scenario are not covered by Figure 1 or are too small to be recognizable in the picture. Nevertheless, they are essential parts of an overall warehouse management system and are therefore mentioned here. Regarding the yard only the loading platform is shown, which is just a small part of it. In addition, a yard consists of a gateway where the trucks register when they enter and deregister when they leave the yard, respectively. Often, a parking area is used for the trucks to stay after unloading while they are waiting for their new load. A yard management system also consists of cameras to observe and control the activities of the trucks on the yard.

Inside a large warehouse there are cameras and other means to observe and monitor the automated retrieval and storage of the storage units (bins). These monitoring capabilities are needed to get an overview of the current warehouse status to be able to adequately and quickly react on different failure situations during runtime to avoid down times of the system. A quick reaction to unusual situations is crucial to the system's availability because of the lack of "buffer space" in the goods processing chain of a warehouse.

Figure 1: External perspective on a large Warehouse

To catch all the three major parts of the overall warehouse – yard management, warehouse management system, and monitoring capabilities – Figure 2 presents a schematic overview concentrating on the important parts needed to further describe the main scenario.

2.1.2 Yard Management

The yard consists of a gateway where the trucks can register when they enter and deregister when they leave the yard to be scheduled and coordinated during the load and unload operations. The scheduling can be planned in advance with the help of shipping notifications of transport providers. Especially for large warehouses a parking place is needed to be able to handle a large amount of trucks especially during peak hours where a lot of trucks arrive and leave within a short period of time. Cameras are used to observe the trucks and the activities on the yard and the yard personnel are coordinated efficiently to run all processes smoothly.

2.1.3 Warehouse Management

The warehouse management system starts at the loading and unloading platform, typically located at different parts of a warehouse, where the stapler cranes pick up

the goods to carry them to the goods reception station or to take them from the goods issuing station to load a truck.

If goods arrive at the goods reception, left side of the warehouse in Figure 2, they are probably repacked into storage bins and then carried to the high racks via conveyers. Automated stapler cranes carry the storage bins to the place at the high rack pre-allocated by the warehouse management system. The successful storage of the goods is reported to an Enterprise Resource Planning system (ERP) that handles the overall commissioning of goods on an abstract level beyond real storage places inside a high rack.

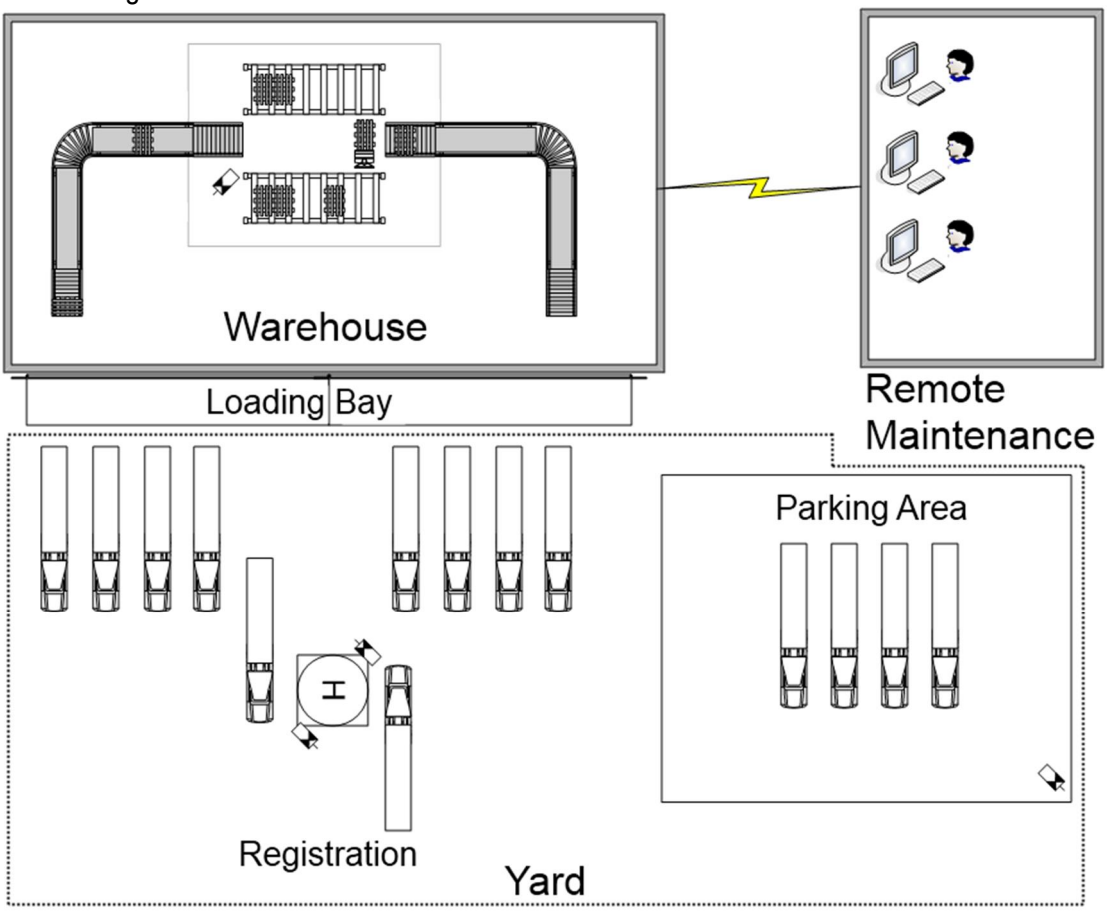

Figure 2: Schematic overview of a large warehouse

If goods are commissioned by the ERP system for delivery an automated stapler crane takes the bins from the high rack to the conveyor, which carries them to the goods issue station.

2.1.4 Remote Maintenance

A remote maintenance system is connected to the warehouse to monitor the procedures inside the warehouse and the yard. The operators get instant status information about the automated warehouse system as well as visual information from cameras. With this information the operators are able to react in case of unusual situations and to recover from failure situations to keep the overall warehouse up and running.

This main scenario gives a very high level and rough overview of a large warehouse system and how they are dependent on each other, to become familiar with the warehouse's major subsystems, which are described in more detail in the following sections.

2.2 Business Case

The bare complexity of a warehouse as described above is hardly manageable by a few engineers. Therefore the design and planning of such a system is a challenging endeavour and not easy to complete within time and budget.

On the other hand the customer's value add lies exactly in the availability of the warehouse and he is willing to pay for a solution that not only fits his needs but also solves related problems like integrating different systems, providing a certain service level, probably guaranteed by a SLA (service level agreement).

Companies that can offer such a system as a custom tailored solution out of one hand have an excellent unique selling point beyond their competitors as the integration of multiple systems is a crucial point for all projects that have a huge integration part.

Managing variability on a system wide level is obviously the right way to address these issues and opens up new opportunities like a unified remote maintenance that can be offered to multiple customers. The cost savings can be used to further reduce the price of succeeding projects, which leads to an even stronger market position.

Furthermore, to model the commonalities and variability of a warehouse on an abstract level, spanning all three major platforms underneath, leads to internal resource savings and the availability of experts. It reduces the effort for the implementation and deployment of such a system to the real variable or custom tailored parts, which must be implemented for every new solution.

2.3 Main Challenges

There is a multitude of challenges that have to be faced when designing a warehouse including a yard management and a remote maintenance. Although, the realization of the case study is possible with technology available today there are a lot of drawbacks of today's approaches.

First of all, it is very difficult to describe the requirements of the complete system. Industry praxis is to describe the requirements of each subsystem and the additional integration requirements.

The Variability is usually described only subsystem by subsystem. A way to describe the overall variability of the system and then derive the subsystems' variability is currently missing. Usually, for the different subsystems, different architectural descriptions are used and connectors between different platforms are usually handcrafted leading to a huge integration effort for every solution.

For deployment and monitoring, platform specific tools are used. Therefore, no tool for the whole platform is available, which prohibits the maintenance of multiple warehouses from a few remote maintenance offices.

Finally, the runtime adaption of the system in case of failures, especially the adaptation of a subsystem due to state changes in another subsystem, requires detailed knowledge about the system's implementation and often needs manual interaction and implementation making this task difficult and error-prone.

2.4 How INDENICA helps to cope with these challenges

The goal of INDENICA is to provide means for managing variability on a system wide level covering the aspects of vertical as well as horizontal integration of different platforms. The case study will show the feasibility of the investigated methodologies and tools.

INDENICA addresses the challenges by describing requirements of the whole system family using a goal-oriented approach. Based on these requirements the variability of the system as a whole is examined employing new variability approaches. Such a holistic view on the system covers not only the variability aspects of the involved platforms but also the issues related to the integration of multiple platforms and the derivation of variability from the virtual platform to the single platforms.

Furthermore, INDENICA will provide unified views to describe the architecture of the complete system. These views can be used to generate connectors between the different platforms covered by the architectural views.

A single tool suite will be investigated and implemented by a prototype for the deployment and monitoring the single platforms and the system as a whole. This finally allows for runtime adaption that spans different parts of the system.

2.5 Variability

The following chapters will cover the subsystems of the case study in more detail and provide insight of some implementation details specific to the single platform.

As the management of variability of an overall system is key issue to be addressed by INDENICA some variability are introduced in this section and referenced by the different platforms to describe how the variability is reflected in the corresponding subsystem.

Variability is something that is captured and described in the problem domain. Therefore, it is usually a decision point visible to the customer or user. The following examples focus on variability that is easy to understand obvious to the reader.

2.5.1 Warehouse size and topology

The size and topology of a warehouse have a huge impact on the warehouse management system as well as on the remote maintenance. To keep this example of variability simple, assume that the customer can choose the amount of high racks, their height and length, given in storage units.

The amount of high racks has a direct impact on the warehouse management system. For every high rack, or at least for every two high racks, an automated stapler crane is needed to transport the bins from and to the conveyors.

It also impacts the structure of the remote maintenance. Every stapler crane must be monitored and should be observable by a camera. If there are multiple cameras to monitor multiple automated stapler cranes a priority based monitoring is needed to have the most important information available to the maintenance personnel.

Dependent on the height and length of the high racks more than one camera is needed to observe the activities in a certain area of the warehouse. In this case a kind of information fusion might be needed to gather all information related to a single failure at a certain location but to also leave out unrelated and unimportant data.

Typically, the size of the warehouse is derived from the warehouse topology. If there are many racks with a large number of transport systems, the warehouse is considered to be a large warehouse.

Another variation point in combination with the topology of the warehouse is e.g. the stacking of goods and bins. Stacking means that the high racks have double the depth of a normal high rack so that two bins can be stored at the same bin location, one behind the other.

Usually you try to store always the same kind of piece or material in one bin. If the goods are so small that only a part of the bin would be used a bin can have a so called panel – a box with up to 16 compartments – for small pieces like screws). In such a setup the remote monitoring system should be equipped with high-resolution cameras to be able to detect the pieces in the panels.

2.5.2 Quality of Service (QoS)

Very typical variations of a warehouse are described in terms of quality requirements like throughput and latency for the transport and delivery of time-critical goods (e.g. chilled goods). If there are different types of goods that need special storage locations like chilled goods, food, or maybe chemical substances where a defined distances must be kept between two bin locations, a special storage handling and strategy might be used. Furthermore, the warehouse management needs some means to track the right handling of such goods.

For warehouses storing small goods that need a very quick handling a special conveyer system might be needed to guarantee the throughput from the high racks to the packing stations and back into the warehouse. Although such products, like contact lenses for example, have a small package size and a negligible volume, it comes with an enormous communication overhead because millions of these items have to be carried around within minutes. As a result, not only the conveyer systems must be prepared to run at higher speed than normal but also the communication system and the warehouse management system must be ready to handle the huge amount of messages.

2.5.3 Upgradability and extensibility of existing systems

A typical business case is to replace an old WMS by a new one or to extend an existing system to server new business needs. A variation point in the first scenario is the adaption of the new transport subsystem to an existing infrastructure. This

means that the platforms must be able to be easily adapted to or configured for different hardware and PLC configurations.

The second use case, the extension of an existing system, needs other variation points. The major non-functional requirement to support this targets the scaling of the platforms to support small to large warehouses.

In both cases the time frame for the introduction of the new system is very tight and often restricted to a few hours to switch from the old to the new system. To guarantee availability of the new system intensive test must be conducted. This requires a bunch of different strategies, which have an impact on the platforms. A step-by-step upgrade (process controls remain active, software is upgraded) might be needed if the warehouse must keep its 24/7 availability. Simulated components might be replaced successively by its real or implemented counterparts to continuously switch to the new system step-by-step.

2.5.4 Adaptation of storage and retrieval strategies

As mentioned before, operators of warehouses might decide to stack two or more storage bins one after another in the same storage location in order to reach a higher packing density. The increased packing density would of course be bought at the cost of a lower access rate. The seasonal variation of the market demand, for instance, could be balanced by an adaptation of the operational strategy: work with a higher packing density around holidays (e.g. Christmas), where an increased demand is anticipated while preferring a faster access rate during the rest of the year.

If the topology of a warehouse is designed in such a way, that alternate routes can be used to process transport jobs, the route occupancy indicator could be used to adapt the strategy accordingly. Let’s assume that there are lot of checkout operations in an automated cold storage house because the expiration date of a product charge has been reached. In order to clear the respective storage locations as fast as possible, all available transport routes can be used to move to expired goods from their storage location to the loading bay.

2.5.5 Prerequisites for storage

There are two types of boxes that are delivered to the warehouse: the ones that are ready for storage and the ones that need to be re-packed. While the former may already be equipped with machine-readable labels that allow for an automatic handling of the storage process, the latter need to be processed by a human worker in order to be able to store them in the warehouse. This activity is often as simple as taking the goods from original box and putting them into another type of box special to the respective warehouse. The operator could even decide to organize the good receipt in such a way that it can handle both types of boxes. In that case the yard management system could guide the trucks to an appropriate loading bay.

Inventory management is an essential part of a well-performing warehouse management system because only the knowledge of the availability of goods allows for an efficient order process. This is especially important for companies that follow an on-demand delivery approach like most of the car manufacturers do. Basically

there are two possible inventory management approaches at the good receipt: count by scan or manual count. Boxes with machine-readable labels can often contain information on the contents of the box, allowing the system to register the box and update the inventory automatically. If the goods have to be re-packed, the packing worker often has to count and enter the number of products for the current box manually.

The nature of the goods to be stored in a warehouse might also affect the storage and retrieval process. Let's assume a warehouse that has been designed to store perishable foods. After the goods have been delivered and transported to their respective storage locations, they are locked. This means that they cannot be retrieved or checked out until the food inspector has taken samples and attested the quality and conformity to current food laws. The remote monitoring system could be involved to capture the quality assessment in order to support the creation of an audit trail later on.

There might also be warehouses that are supposed to store a multitude of products that differ in size, weight and shape. In order to make sure that a specific product can be in a warehouse, a set of checks can be performed prior to the actual storage process. The racks of each warehouse have been designed to carry a maximum weight and there might also be a maximum weight per storage unit. The product which is about to be stored can be weighed in order to verify that its weight matches the limitations of the racks. The same holds true for the shape of the product to be stored. Even if the total volume of the product is smaller than the volume of the dedicated bin location, there is no guarantee that it will actually fit in. Therefore a contour check can be performed in order to detect deviations from the tolerable dimensions. If the product does not match the warehouse's weight or size specifications, it is rejected. In that case the operator has to decide how to proceed. Large or heavy products are often transported on pallets whose specifications may differ from company to company or from country to country. In order to guarantee that the product can be stored using the palette on which it is currently placed, the palette itself has to be measured. In case the palette is not suitable to be used in the current warehouse, the product has to be transported to a repacking place. The product will then be placed on another palette that conforms to the warehouses specifications.

2.5.6 Commissioning

There are various strategies when it comes to commissioning. Consider a set of orders that consist of only one product of the same type. Here the typical strategy is as follows: the storage unit that contains the desired product is located and transported to the checkout workplace. There the products are picked from the storage unit and directly packed. The storage unit will be returned to its location after all orders have been processed.

Another strategy makes use of additional boxes for a temporary storage of goods. Consider a set of orders for a composite product (e.g. a PC or a bicycle). Instead of retrieving a storage unit for each order (order-based processing), the storage unit for

each component is retrieved (component-based processing). As soon as a storage unit arrives at the checkout workplace, the products representing the component are picked and placed into temporary boxes (one box per component per order). After all components have been retrieved, the boxes for each order are grouped and then packed. This strategy, however, is only applicable if the volume of the temporary storage area is big enough to handle the amount of articles to be checked out.

2.6 Governance

In order to ensure that the Virtual Domain-specific Service Platform (VDSP) meets the business requirements SOA Governance is required. SOA Governance addresses not only the challenges of design and implementation of VDSP, but also of deployment and operation.

SOA Governance extends IT Governance and the Enterprise Architecture Governance (1). The SOA Governance Reference Model of the Open Group, which will be the base of our considerations, defines a set of constituent parts, e.g. Guiding Principles, Processes (governing and governed processes) and Roles & Responsibilities. As some aspects of SOA Governance are mentioned in the following case studies, a short definition of these parts is given here.

Guiding Principles

Which guiding principles are selected and how strictly they are applied depends on the governance maturity of an organization. The SOA Governance Reference Model of the Open Group defines a set of guiding principles, e.g.

- An SOA Reference Architecture is required (addressed by the INDENICA View-Based Architecture)
- Service reuse (addressed by the VDSP)
- Service monitoring (addressed by the INDENICA governance model)

Processes

The SOA Governance Reference Model differentiates between governing and governed processes.

The *Governing Processes* include compliance, dispensation and communication processes. The objective of the compliance processes is to ensure adherence to policies, guidelines and standards defined by the INDENICA governance model. There are two categories of policies. Policies used to govern services prior to deployment are called *design-time* policies. Policies for describing the correct behaviour of service operation are called *run-time* policies.

In the context of INDENICA there are different levels of platforms to be considered, the existing platforms and the VDSP. First, policies on the VDSP level have to be defined. In a second step these VDSP policies are refined and mapped to the existing platforms. For these platforms sets of policies are already available as a result of the governance activities on this level. The refined policies have to be consolidated with these policies resulting in updated sets of policies on this level.

High-level policies on the VDSP level that influence the existing platform are e.g.

- The request for compliance with business rules and regulations like Sarbanes-Oxley Act or Basel III. From these a derived and measurable policy for e.g. the warehouse management system is to have at any time an overview on the bound capital in the warehouse.
- Base platforms must be chosen in a technology selection process.
- All services are subject to the Service Portfolio Management process and the request for additional or new services has to be forwarded to the EA Governance Board for approval and decision to implement or purchase a new service.
- All available services have to be published in a centrally accessible service repository.
- Services exposed by the existing platforms should be used whenever appropriate. For approval of exceptions the SOA Governance Board should be asked.
- All services should provide variability to be tailored to the customer's needs.

Examples of additional policies on both levels are shown in the following chapters.

The *Governed Processes* include planning, design and operational aspects of Solution / Service Portfolio Management and Lifecycle Management. The Service Portfolio Management has to address the variability aspects in the scoping phase and has to decide which services should be available on the VDSP. The Solution or Application Portfolio Management ensures that the organization has a set of applications appropriate to satisfy its needs.

The Service Portfolio Management and Lifecycle Management on the VDSP level should be consolidated with the corresponding processes of the existing platforms.

Roles and Responsibilities

The SOA Governance Reference Model of the Open Group defines a set of boards (e.g. SOA Steering Board, EA Governance Board or SOA Center of Excellence) and teams (Solution Development Team and Service Development Team) based on several key roles within the organization. As there are two levels of platforms within INDENICA, the boards and teams of the higher level should include roles of the existing platform level and vice versa.

In the context of the INDENICA case studies the main focus is on roles involved in the integration of the existing platforms. These are key roles of the Solution and Service Development Teams, e.g.:

- Existing Platform Provider
  is a technology expert and describes the current variability and the variability binding process of the existing platform he owns.

- VDSP Architect
  is responsible for VDSP requirements, variability within VDSP, baseline architecture and adaptation behaviour of VDSP
- VDSP Integrator
  generates the VDSP instance.
- Application Developer
  develops applications based on the VDSP instance
- VDSP Administrator
  Monitors the current state of the existing platforms and is responsible for making adaptation decisions for the VDSP instance

3 Warehouse Management Subsystem

3.1 Description of the Domain and existing platform

3.1.1 The Warehouse Management Domain

In general a warehouse is a type of a building used for storage of finished products or materials. Warehouses can be as small as a parts room at a car repair shop or as large as a central storage building of an online-sales company. Larger warehouses often feature a yard with loading docks to load and unload goods from trucks, trains and ships and a transportation system with conveyors and forklifts for moving them. In this project we will focus on larger warehouses with a storage space of 25.000 feet or more. During recent years the number of fully automated warehouses has grown due to the increased freight transport volume and the subsequent demand for more efficiency in logistics. The transport system's feeding rate within modern warehouses may well reach 3 meters per second, thus exceeding our human reaction rate and our ability to take in information. Therefore computer-controlled systems manage the transport of products on material handling equipment and storage and retrieval systems (2).

The ability to efficiently manage the storage, pick-up and flow of goods is referred to as warehouse management and a system that deals with this process is called warehouse management system (WMS). WMS focus on the movement and storage of goods within a warehouse as well as the associated transactions like receiving put away, picking and shipping. It offers means to manage the inventory and control the actual transport subsystem. Modern systems may utilize tracking techniques such as Barcode scanners or RFID tags to monitor the actual flow of goods within the warehouse. This information can be used to optimize the process of storage and retrieval in order to increase the throughput rate. Vendors of these kinds of logistic systems often provide service platforms as interface to their automated warehouses. These service platforms can be integrated with enterprise resource planning (ERP) or supply chain management (SCM) systems in order to connect the inventory management systems with the actual storage and retrieval system.

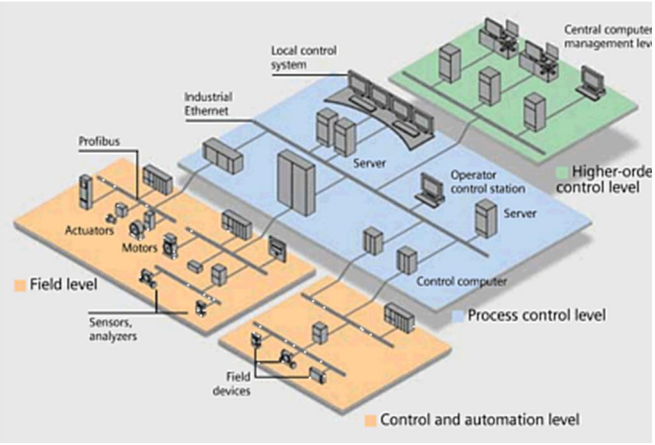

Figure 3: System Architecture for control levels

Automation in a warehouse takes places on several levels of the automation pyramid, as shown in Figure 3. Field devices like motors, actuators and sensors are logically integrated on the control and automation level which in turn is monitored and controlled by the process control level. Systems on a higher-order control level integrate several subsystems from lower levels into a single

monitoring and operator control system. In this project we will focus on the higher-order control level in the context of the overall WMS and the process control level in form of the transportation subsystem.

Of course a WMS is more than just software: the major part of a warehouse consists of metal sheets in form of racks, conveyors, vertical lifts and rail systems that are responsible for much of the total cost of the overall system. Nevertheless we want to focus on the software service platforms that may be used in a WMS.

3.1.2 Sample Application Use Cases

Since the domain of warehouse management is very complex, there are more use cases than can possibly be considered in the context of the INDENICA project. That is why we focused on two use cases that are common to all warehouses management systems. Although the order of some steps may vary from system to system, the semantics are comparable.

Storage of new products

If a new product is to be stored, it has to be registered first. That usually happens when the underlying transport system notifies the WMS about a newly arrived storage unit. The inventory management system will then register all articles that reside in the storage unit with their respective amounts. Subsequently the WMS will search for a suitable bin location. If there is currently no location available, the storage request will be rejected.

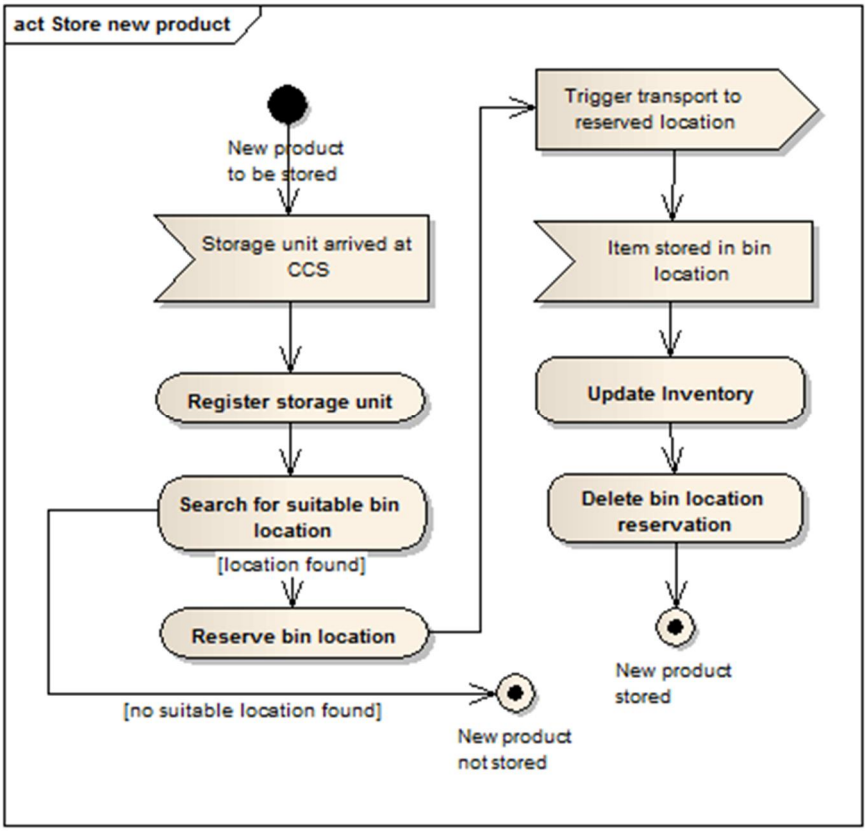

Figure 4: use case depicting the storage of items in the WMS

If a suitable location can be found, it will be reserved and the transport to the reserved location using conveyors and lift modules is triggered. The underlying transport system notifies the WMS about the successful storage which in turn allows the WMS to update its inventory and to delete the existing reservation for the bin location.

Retrieve some products from a storage unit

In case some products are needed from the warehouse, they can be requested using the WMS. First of all, the name of the article and the desired amount has to be specified. The inventory management system then searches for a storage unit that contains enough items to fulfil the request. Should more than one storage unit contain the products that were requested, a list of storage units to be checked out is retrieved. Subsequently a checkout order based on the returned storage unit ids will be created and processed.

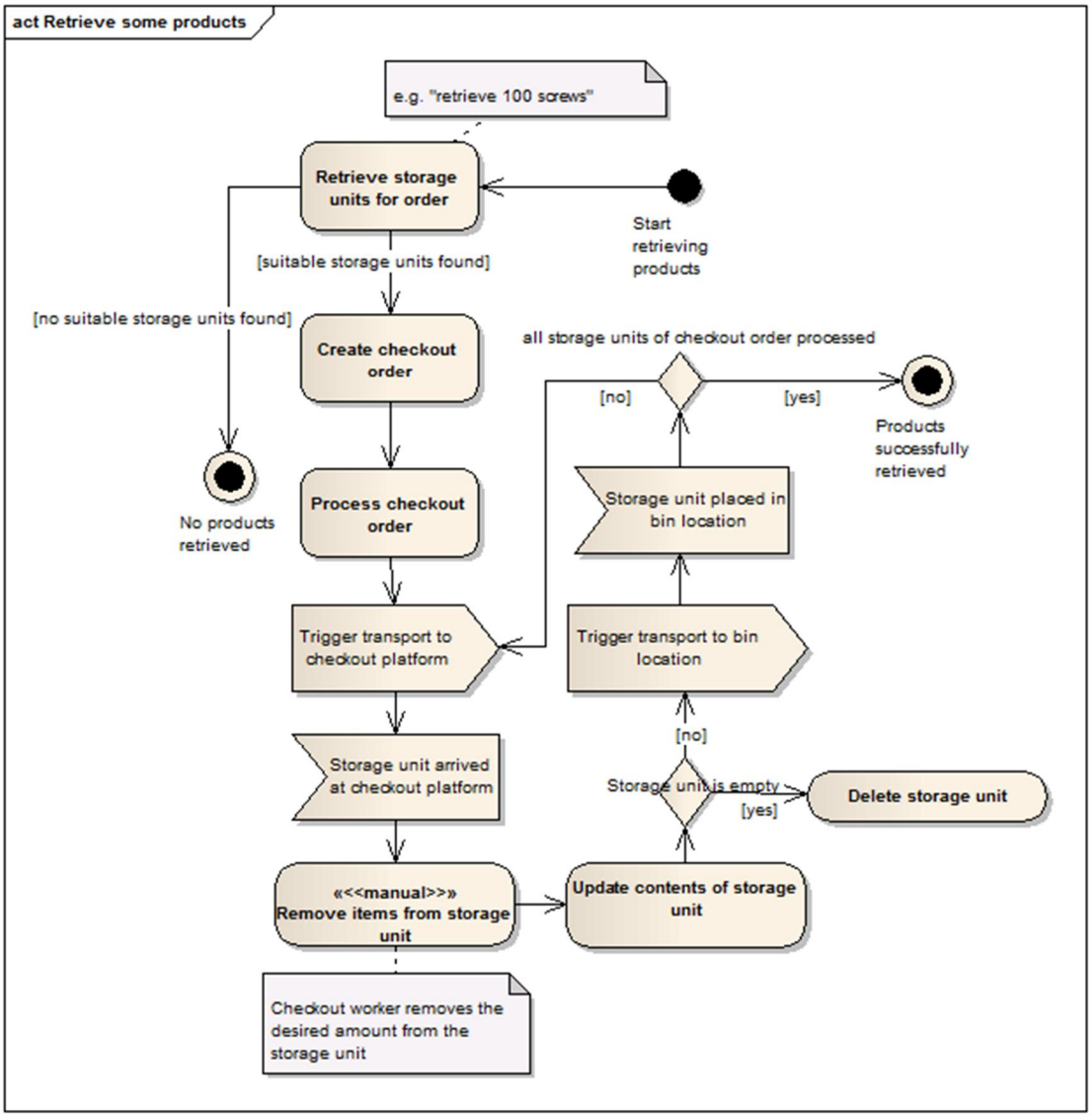

Figure 5: use case depicting the retrieval of items from the WMS

The WMS triggers the underlying transport system in order to transport the specified storage units to the checkout platform. As soon as they arrive, a checkout worker can retrieve the desired amount of products from the bin. After the number of removed items has been specified, the storage unit is sent back to its storage location. The checkout order is considered to be complete as soon as the desired number of items has been retrieved from the bins.

3.1.3 Software Base Platforms

This section describes the software platforms used to host a variant of a warehouse management system that will be implemented in the course of the INDENICA project. As the service platform for the warehouse management system consists of two subsystems with quite different requirements, two separate platforms and runtime environments were chosen.

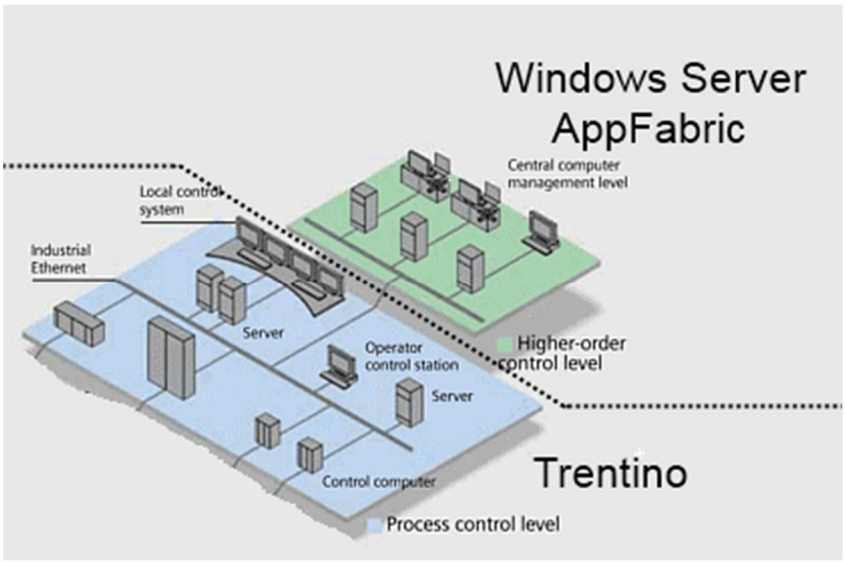

Figure 6: Location of used base platforms on the respective control levels

Windows Server AppFabric

At the high-order control level there is the inventory management subsystem which takes care of tracking the goods and offers an interface for ERP systems. Since Microsoft's .NET framework is used widely within Siemens, the Windows Communication Foundation should be used as API for providing access to the top-level WMS service. In order to host and monitor the WCF services, an application server with enhanced hosting and monitoring capabilities was needed. The Windows Server AppFabric has been introduced along with other new features as part of the version 4.0 of the .NET framework. Windows Server AppFabric extends Windows Server to provide enhanced hosting, management, and caching capabilities for Web applications and middle-tier services. The AppFabric hosting features add service management extensions to Internet Information Services (IIS), Windows Process Activation Service (WAS), and the .NET Framework 4. This includes Hosting Services and Hosting Administration tools that make it easier to deploy, configure, and manage Windows Communication Foundation (WCF) and Windows Workflow

Foundation (WF) based services. The AppFabric caching features add a distributed, in-memory object cache to Windows Server that makes it easier to scale out high-performance .NET applications, especially ASP.NET applications. (3)

Trentino

The transportation control subsystem operates on the process control level and serves as abstraction layer to the actual motors, conveyors and vertical lift modules, offering services to trigger transport jobs within the warehouse. Although an extension of service oriented paradigms to the lower levels of the automation pyramid are desirable, there are currently only few frameworks that support these activities. Trentino is a C++ based lightweight and non-invasive SCA (Service Component Architecture) runtime that has been optimized for the use in the embedded domain. As Trentino addresses the service oriented integration of low level components it has been chosen as base platform for the transport control service.

3.1.4 Domain Services

In order to create a domain specific variant of the base platforms, two services will be offered: a WMS Service which is to be used by an external controlling application and a Transport Control Service that is used internally by the WMS Service in order to move goods within the warehouse.

WMS Service

The service provides access to the warehouse from a stock controller point of view. New Storage units can be registered for storage or single items can be retrieved from existing storage units, keeping an overview of the actual bin occupancy. It keeps track of the storage units and their respective location within the warehouse. Additionally it provides an abstraction to transportation activities like transport to a reserved bin location or to the checkout desk: the various calls to the underlying transport service are handled by the WMS service while the client receives a call-back after the completion of the transport job.

Transport Control Service

The transport control system service, on the other hand, controls the actual system of conveyors and the automated storage and retrieval system and is consumed by the top-level WMS service. It is responsible for actuating the programmable logic controllers (PLC) of the actual hardware. Since some transport activities may need a while to complete, callbacks are offered to notify the service client about the completion. The service itself offers a more or less direct access to the actuators in the warehouse and provides no additional logic for handling transport jobs. That implies that the calling client has to keep track of the storage units and their respective location within the warehouse's transportation system.

3.2 Challenges for Integration into virtual Platform

3.2.1 Operational Requirements

The system is designed to be used in non distributed warehouses. Its intended use is the management of the flow of goods within a warehouse in one location, from the time of delivery to the time of distribution.

Critical system parameters are the time for transport, storage and retrieval. The system is only able to operate efficiently if goods can be transported to their respective destination as fast as possible.

Since a WMS can be considered as backbone of a functioning sales or trading company, its availability is crucial for the success of the business. A tight integration of WMS and remote maintenance system is desired in order to detect problems within the workflow as early as possible and to reduce the risk of unexpected failures.

The life cycle of a storage unit starts with the delivery and handover at the loading bay. After the unit has been registered, the inventory management system of the WMS is notified and the storage workflow is triggered. The life cycle of a storage unit ends when it does not contain any items and is therefore withdrawn from circulation.

3.2.2 Variability of the platform

A WMS is often connected to an ERP application which in turn is used to process customer requests and orders. A minimal latency and almost immediate access to the WMS data is therefore essential. Application performance may be increased by speeding up the access to the application data. There are many ways to speed up the data access, the simplest being a larger application server. However, also employing bigger hardware has its limits and may become rather expensive at a certain point in time. A much more effective way of achieving this is using the extended caching capabilities of AppFabric. It provides functionality to employ an interconnected cache cluster which consists of a set of interconnected caching servers. The caching services client library allows transparent access to the machine's local cache as well as to the cache cluster, thus allowing the creation of scalable applications without the need to handle caching within the application scope.

If high-availability is demanded by the operator of a warehouse, AppFabric's built-in high-availability caching option can be activated. That option mirrors the cache data on another machine in the cache cluster, making the data accessible even if the primary cache server fails. In cases of server failures, client applications are able to proceed as if nothing had happened.

AppFabric provides a pre-configured persistence storage that may be used in order to persist the current workflow state of the WMS. This feature is part of the AppFabric Hosting Services that allow for the creation of scalable business logic without the need to handle the workflow state manually.

Especially in small-parts warehouses panels or multi-compartment storage bins can be transported at very high velocities. The WMS has to address this fact with the

ability to process lots of items in a short time frame. AppFabric provides load balancing functionalities that allow for an automated distribution of user requests to a set of servers.

3.2.3 Governance

As already mentioned in chapter 2.6 the adherence to policies is ensured by compliance processes. There are two sources for these policies, the governing process of the warehouse management subsystem and the set of refined policies derived from the VDSP policies (examples see 2.6). Based on this a consolidated set of policies is defined during the integration process.

Examples for additional Warehouse Management Subsystem policies regarding the usage of the sample services are e.g.:

- The WMS service utilisation must be registered at the warehouse platform owner.
- The usage of the transport control service is restricted to the warehouse internal applications and must be protected by appropriate authentication.

Further policies in the context of the above scenarios are among others:

- The rejection rate of storage requests should be less than 1%. This leads to the following monitoring and dispensation activities:
    - Keep the capacity utilisation of the warehouse below 95 %. If utilisation exceeds 95% notify warehouse management, who shall take remediate action.
    - If the storage request for a new item fails, retry after 5 minutes. If it fails again, notify the warehouse operator and stop all incoming items until the operator releases operation.
- Retrieval jobs have to be processed en-bloc.
- The rate of non-retrieval of requested items shall not exceed 0.1%.

4 Remote Maintenance Subsystem

4.1 Description of the Domain and existing platform

4.1.1 The Remote Maintenance Domain

Remote maintenance can be characterized as a maintenance activities performed or controlled using external channel. Maintenance activities itself can be performed by single person, group of people in cooperation or automatically based on policies, rules, SLAs, etc. with its separation between subjects by physical distance.

Working with high-throughput environment it's crucial to detect anomalies (sometimes even predict) and act to them accordingly. Remote Maintenance tools enable to detect such situations based on valued of some pre-defined parameters as well as on events that are provided by underlying systems. Supporting tools are also able to improve the usage efficiency of underlying infrastructure by prioritizing network flows (audio/voice streams) based on current situation in the environment.

Real-time and Near-Real-time systems can sometimes have QoS-related requirements that need to be fulfilled by reacting to the system states in general. Such requirements can be specified as a set of policies and the available actions can be defined per request-response basis or can require some personal decisions. The corrective actions can include changing settings to improve performance or prevent problems in the future as well as replacing manpower at the location by experts in the central location.

As a part of larger systems, the remote maintenance subsystems are able to take directives from different subsystems and react based on their events. The same applies for the different direction – remote maintenance subsystems are able to provide information about ongoing activities or performed system changes to other subsystems. These types of messages can be also passed using direct Machine-to-Machine communication since remote maintenance logic can be distributed across multiple devices.

4.1.2 Sample Application Use Cases

In the context of INDENICA the more interesting maintenance supporting activities are the ones that support the integration between various subsystems. That is why we concentrate on the monitoring of various system parameters that are a base for intelligence retrieval and prediction on future states. Moreover we want to support the system in various emergency situations in order to keep up availability requirements and agreed QoS parameters.

Video Monitoring Live Streaming

The security staff is able to control the incoming and upcoming personnel and vehicles using live video streaming. They are controlling several monitors at the same time and in order to help them, we are able to focus on one of the streams by increasing its resolution with the cost of all other streams. This might happen especially when the truck arrives at the gate or a new shift of staff enters/leaves the

location. At the same time the quality assurance team members are able to monitor the flow of the goods for efficiency/safety reasons. They can also use the same mechanism for focusing on selected streams. Using IP cameras staff is able to perform video calls with workers across different location to monitor/update/exchange of information.

Video and audio streaming prioritization

In case of emergency, accident or malfunction proper staff will be automatically notified. In case of some for serious accidents or fire alarms emergency services will be notified and the live audio and video stream from the site will be provided.

At the same time these live streams will be prioritized based on the event, and location to deliver the most useful view at the site.

Monitoring and prediction of the network

The overall state of the network needs to be monitored in near-real time in order to ensure high availability and to reduce the downtime of the running services.

Events gathered from every subsystem are collected and processed by Complex Event Processing (CEP) Engine in order to detect anomalies and violations. Proper derivation of the near-real-time complex events will be the base for prediction of future state of the network and will enable network administrators to take some action in advance.

4.1.3 Base Platform

For the purpose of ensuring some functionality of the remote maintenance system we will provide a monitoring and adaptation framework together with Mobicents JAIN SLEE[1]. Monitoring and adaptation framework will be build to ensure proper monitoring of system events and its non-functional parameters in near-real time. The framework will be able to communicate with different subsystems to exchange information about these events as well as to pass some directives for adaptation purposes.

Mobicents JAIN SLEE will act as a communication platform for audio/video streaming, call initiation and management and short message gateway.

Mobicents JAIN SLEE

Mobicents is the first Open Source VoIP Java Platform. It provides scalable environment to create, deploy and manage services and applications integrating voice, video and data across a range of IP and communications networks. The platform acts as an event-driven application server with a robust component model and fault tolerant execution environment. It complements J2EE to enable convergence of voice, video and data in next generation intelligent applications.

[1] http://www.mobicents.org/slee/

4.1.4 Domain Services

To create domain-specific services on top the previously described platform there will be built several JAIN SLEE components for session initiation and handling, message queuing/passing and machine-to-machine communication. In order to make the cooperative with other subsystem as well as external applications and devices, a common integration interface will be provided.

Staff will also be provided with remote access stations in order to communicate with different crew members from various locations and using various types of equipment. Staff will be able to use application for cooperation together with audio/video/message exchange. Some part of these streams will be stored and divided into sections accordingly to the corresponding event/situation.

Call Session management

Every user will be provided with his own private credentials needed to perform a call or report to other staff member. The caller will be able to connect to other staff member based on his current location, case, and place in the corporate structure or his need. He will be also able to make a call to security or emergency services.

In case of need, every party in the call will be able to add additional parties to the call. The call is provisioned until the last party leaves the call.

Callers will also be able to exchange some pictures/video stream during the call to remotely consult about the ongoing activities at the site.

Remote monitoring

The monitoring engine will allow responsible staff to check the performance of the whole system in near-real time. Information from various sources will be aggregated and pre-analyzed and on-demand reports will be able to be generated in just few seconds. Stored data will also give a chance to perform some data mining in order to produce charts and deeper analysis of trends.

Complex Event Processing and supporting prediction components will allow to observe critical parameters of the subsystems and prepare/perform actions before the critical events occur.

4.2 Challenges for Integration into virtual Platform

4.2.1 Operational Requirements

In order to cooperate outside the single subsystem, the integration between will need to be done on platform level. Different platforms will be provided with common monitoring (event-based) interface and adaptation interface (policy-based).

Different platforms will need to cooperate to reduce the overhead activity such as logging in or locating a staff member. Platforms will need to exchange some information regarding current activities and states of the system.

To build a heterogeneous environment based on different service platforms there is a need to combine the operational (functional) capabilities of these platforms. In

order to fulfil this concept, the semi-automatic integration of base platforms (and its variants) will be supported by tools delivered in WP3.

4.2.2 Variability

Instance of the remote maintenance subsystem will be able to be preconfigured in order to fulfil requirements. In order to ease the configuration process, the different values of configuration parameters of the platform will be mapped to the platform variability. For example if the system integrator wants to deploy the subsystem for single warehouse, the more emphasis will be made to ensure the computing power rather than the efficiency of communication channels. This way we will be able to deliver several variables of the platform that matches specific high-level requirements.

Different subsystem variant will also differ with regards to its internal structure and dependencies. Different variants may use different databases, communication channels or any other underlying technologies to better fit the requirements.

In order to automatically develop/deploy different platform variants or in some cases, change the running variant into another, variability description model will be used to deliver variability configuration model. These models together will be able to generate deployment descriptors and configuration parameters.

Different variability information and generated descriptors will be stored in the repository to enable caching and reusing once created deployment process. This information will also provide some instructions for monitoring and adaptation framework.

4.2.3 Governance

To ensure adherence to run-time policies on the VDSP and the existing platforms level, monitoring activities are required. These activities are in the duty of the Remote Maintenance Subsystem.

Monitoring will be based on the supporting monitoring and adaptation framework. CEP engine will be used to detect and aggregate event and later to produce benchmark reports which will be stored in the repository. Based on these reports, data mining techniques will be used to gather intelligence that will also take part in near-real time analysis. The whole monitoring of the platforms can be seen as a closed loop which self-adopts to current state in time.

Standard message formats will be provided to provide scalable environment which can be easily extended with additional platforms.

From the management/supervision point of view, Java Management Extensions (JMX) technology will be used to pass adaptation directives and create policy-based management solution.

To govern the development and implementation of the Remote Maintenance Subsystem a set of design-time policies is defined, e.g.:

- Monitoring should be done in near-real time.

- In case of emergency staff should be notified automatically
- Corrective actions should be taken to improve performance or prevent problems
- Information should be provided to the other subsystems
- The Remote Maintenance Subsystem should be able to get directives from other subsystems.

5 Yard Management Subsystem

5.1 Description of the Domain and existing platform

5.1.1 The Yard Management Domain

Large distribution centres have to handle ten thousands truckloads each year with up to 900 trucks a day. As many as 800 employees have to collaborate to provide fast and reliable shipment completion of those truckloads. With so many trucks, the need arises to organize yard traffic in a proper fashion to avoid bottlenecks and to raise the overall throughput of goods in the warehouse.

Several persons have to work hand in hand to assure the continuous flow of goods at the yard. Incoming trucks have to be registered at the gate guard and assigned to free docks for unloading the trailer. Some goods can only be unloaded at special docks, while other docks are preferred for shorter storage distance. Every loading process has to be handled by warehouse staff, so they need to be notified on new delivery tasks. The yard jockeys are in charge of fetching trailers from the parking lot and bring them to a specific dock. All processes are administered and monitored by the yard manager.

Missing proper management of these processes can result in miscommunication, relocating of trucks including a relocation-fee due to occupied docks or unnecessary waiting time of trucks. Missing information can lead to overly long fetching time of trailers because the jockey has to search for the trailer.

To solve these problems, Yard management systems (YMS) can be used for regulating trucks and trailer movements on the yard. Current YMS support several features. Dock door scheduling allows automatic scheduling of inbound deliveries by assigning arriving trucks to free docks based on business rules. With the support of location-conscious mobile devices, YMS are able to monitor location and state of various entities on the yard. Some YMS support identification of trailers via RFID or similar techniques.

Because of the position of yard management in the overall supply chain, YMS are often interfacing with transport management systems (TMS) as well as warehouse management systems (WMS). This allows exchange of information like advanced shipping notice or optimized material flow in the warehouse.

All these features allow maximizing throughput of goods with a decrease of the error rate during scheduling. Finally, YMS allow optimized flow of information for better transparency and analyzability of processes on the yard.

5.1.2 Sample Application Use Cases

This section describes the sample application use cases which should be implemented for the Yard Management subsystem. The two main features of Yard Management are Dock Door Scheduling Dock Door Scheduling (DDS) triggered by Advance Shipping Notices (ASN) and Truck Status Information (TSI), as well as Yard

Jockey Support. In the following the scenarios which should be supported will be described more detailed.

Dock Door Scheduling

The supplier or transport service provider sends an ASN to the warehouse. With the included information about the approximately expected arrival time and content of the loading, the Warehouse Manager is able to plan further actions to prepare a smooth loading or unloading process. He informs also the Yard Manager who needs the information to prearrange the occupation of the dock doors.

Truck drivers should be able to update the Warehouse Manager about a more concrete arrival time or delays. If he receives such a message he would be able to react and can plan more concretely, respectively reschedule assignments and tasks.

If the truck arrives at the yard, the driver has to check in via a terminal which provides information about its assignment. The Yard Manager now assigns either a dock door or a parking lot if no appropriate dock door will be available. For this task the Yard Manager uses a graphical planning board.

If further information or service is required he should be able to inform the Yard Manager, who can react in an appropriate way. An example for such a reaction can be a notification of a Yard Jockey.

Yard Jockey Support

The Yard Jockeys must be able to get informed about new tasks. Such tasks can be:

- Inform the truck driver who wait about loading start
- Contact a truck driver after a service request
- Pick up a trailer for loading.

The assignment of Yard Jockeys to tasks should be based on their location and further schedule in an intelligent and efficient way.

5.1.3 Base Platform

The Base Platform provides common services which are necessary for the development of most of the features of the Yard Management platform variant. Decomposing all features shows that most of the services need access to a bunch of domain objects at least indirectly. There is also a need to access the services remotely in synchronous and asynchronous ways, whereas the usage must be protected against unauthorized access. For these requirements it is advisable to use an application server which provides these services by definition. Typical provided services which are needed for the base platform are persistence, messaging, authentication and web development support in general.

Another requirement which is resulted by the demand for variability and a Java-based runtime is the usage of an OSGi[2] platform. The modular platform, specified by the OSGi alliance, is based on bundles which contain services managed by a “Service

[2] OSGi http://www.osgi.org/

Registry". It is a dynamic platform, because bundles can be deployed, updated and removed at runtime. In addition to we need a framework which supports us in the creation of distributed applications. Therefore we want to use the Spring Framework[3] which provides with the Spring Dynamic Modules[4] a bridge to OSGi.

All these requirements are fulfilled by the Virgo Web Server[5] of the Eclipse Foundation. This server is OSGi based and powered by Spring which helps to easily integrate technologies for development and running distributed applications.

5.1.4 Domain Services

The base platform will be combined with several domain services to produce a domain-specific platform variant. These services include general support for yard management as well as additional services for data interchange, mobile communication and collaboration.

Yard Management Service (YM)

This service provides basic logic to handle common yard management processes.

New shipping tasks including their advanced shipping notice can be registered in the system. Arriving truckloads will be scheduled and assigned to loading docks or to a waiting area (dock door scheduling, DDS). Thereby business rules have to be taken into account like docks for oversized goods or docks for rapidly spoiled food, which needs cooling. Also logic for rescheduling because of delays is included. Additionally, it provides a basic framework for building a Spring-based yard management web interface, including a graphical interactive representation of the actual yard.

The yard management service will mainly be used by the yard manager for administration and monitoring as well as by the gate guard to register new truckloads and communicate scheduled docks.

Yard Jockey Service (YJ)

This service allows scheduling of tasks for yard jockeys. These tasks include fetching or relocating trailers on the yard. Additionally, locations of trailers are maintained which allows intelligent scheduling of tasks and optimizes the path of the yard jockey. The trailer position will also be used to select the jockey to which to assign the specific task.

By providing generic user interface components, tasks can be created by the yard manager. After creation, the yard jockey will be notified of these new tasks. The jockey can update his status for monitoring purposes.

Mobile Communication Service (MC)

This service provides functionality for communicating with mobile devices. To allow fast and effective communication, several persons can be equipped with such

[3] http://www.springsource.org

[4] http://www.springsource.org/osgi

[5] http://www.eclipse.org/virgo/

devices. This service can be used to distribute notifications and to monitor the state of several yard entities in near real-time. The yard jockey receives notification on new tasks and updates his state whether he is searching for a trailer, carrying a trailer or idling. In case of a delay, the truck driver can send a notification that triggers a rescheduling of the dock occupations. Truck drivers on the yard can receive information about their assigned docks. They will be notified whenever a change occurs. The warehouse staff can update the loading or unloading status of the current trailer easily and receive notifications about new loading tasks.

This service also facilitates development of mobile user interface by supplying an application framework for mobile YMS UIs. It thereby provides a development environment for building native apps for mobile devices based on Spring Android[6].

Location Service (LS)

Mobile devices can also be used to communicate their position via GPS or similar. This information helps to determine the time till arrival of truck drivers (GPS-supported DDS). Additionally, it provides yard jockeys with precise positions about the trailers to be fetched. Because the YMS knows the position of all yard jockeys, it can better assign fetching tasks to the best suited yard jockey.

EDI Service (ES)

Electronic data interchange allows for standardized information exchange between organizations. This service interfaces the YMS to other organizations by allowing information exchange via EDI. This service will be used by external organizations to transfer advanced shipping notices to the YMS in an electronic way.

5.2 Challenges for Integration into virtual Platform

5.2.1 Operational Requirements

The yard management system is tightly integrated with the warehouse management above. Naturally both cannot be used independently from each other as ongoing changes in the yard will affect the warehouse and vice versa. For this reason the operational requirements for both systems will also be linked to each other.

One non-functional requirement for both management platforms which entails functional requirements is high availability. In such a case certain components must be developed in a fail-safe manner. One way to achieve an improved availability is the dual design of certain components or whole systems. These systems and/or components need to be synchronized to contain the same data at any time. For this reason functionality for scheduled synchronization tasks need to be implemented in the platform.

Another non-functional requirement that manifests itself in functional properties is performance. A high performance solution can make use of caching mechanisms which need to be implemented in the platform itself. These caching mechanisms need also be synchronized with the warehouse management platform in order to

[6] Spring Android: http://www.springsource.org/spring-android

ensure that no invalid set of data is processed. Another aspect is a dedicated deployment to parallel systems which balance the accumulated network traffic. In addition parallelization (and therefore data synchronization) needs also be implemented in the platform.

Another scenario which is quite common nowadays is the deployment of the platform or parts of the platform into some cloud environment. In this case the integrated solution is actually a distributed system spread over different system providers (both in-house and external). This entails certain requirements for the integrated platform, i.e. secure communication between providers or making certain components “cloud-aware”, which means they need to conform to guidelines provided by the appropriate cloud provider.

5.2.2 Variability

5.2.2.1 Variability of the base platform

An OSGi based platform is variable by definition, because it allows an easy exchange as well as the extension of bundles. In theory this gives the user a lot of freedom to adapt the platform regarding his requirements and favours. But in practice not all variation points are relevant, because the decisions often motivated from business side or on a higher functional levels. That’s why our Base Platform provides only tree variation points: persistence, connectivity and authentication.

Persistence

We use a relational database for the persistence of the domain objects. For the access two variants are available. One variant is the access via JDBC[7] which allows the usage of SQL queries directly in the Java Code. The other variant is the usage of an implementation of the Java Persistence API (JPA)[8] which provides comfortable O/R mapping functionality to the user.

Connectivity

Besides web server functionality to consume HTTP-Requests, there is also a need to consume and provide web services. For this we provide three possibilities. The first is the classical usage of Remote Function Calls (RFC); the second is the usage of SOAP-based services and the third possibility is the usage of REST.

Authentication

For Authentication we use Java Authentication and Authorization Service (JAAS) an API which allows connecting Java-based applications with services for authentication and access rights. Because there are a variety of such services (Kerberos, LDAP, SAML-based Services, PKI, etc.) we only provide the interface as an Extension Point, which must be implemented by the user.

[7] http://www.oracle.com/technetwork/java/overview-141217.html

[8] http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html

5.2.2.2 Service variability

All mentioned services provide different variability to be tailored to the customer's needs. YM as well as YJ are open for extension by other services. Both alone supply a simple web-based user interface. With extensions by MC they can also be used via mobile devices. MC itself can be extended with GPS-support via LS to allow more intelligent scheduling. To communicate EDI-information via ES, several protocols can be used, for example EDIFACT or ebXML, both of which are used in practice.

Because of the use case domain, every variant of the base platform will contain at least the basic YM- and YJ-services. To support business data interchange, V1 will contain EDI tailored to the usage of the EDIFACT protocol.

As an extended variant, V2 supports mobile devices with GPS incorporating MC and LS. V2 supports ebXML instead of EDIFACT as the EDI protocol to demonstrate exchanging variability.

5.2.3 Governance

Chapter Error! Reference source not found.2.6 shows some examples for high-level policies on the VDSP level that influence the yard management subsystem.

Further policies on the yard management subsystem level are e.g.:

- The Yard Management Subsystem should allow maximizing throughput of goods with a decrease of the error rate during scheduling.
- The Yard Management Subsystem should allow optimized flow of information for better transparency and analyzability of processes on the yard.
- The distribution of notifications and the monitoring of the state of yard entities should be done nearly in real-time.
- A smooth loading or unloading process should be guaranteed, e.g. by advanced shipping notices.
- The Yard Management Subsystem should know the position of all Yard Jockeys.
- The assignment of Yard Jockeys to tasks should be done in an intelligent and efficient way, e.g. based on their location and further schedule.
- Information exchange between the Yard Management Subsystem and external organizations should be possible via Electronic Data Interchange.

6 Integration Scenarios

6.1 General Scenarios

The sections above outline the scenarios for each of the platforms the industrial partners of the INDENICA project provide. However, the case studies work package does not only cover each of the platforms individually, but also in an integrated manner. Therefore we also outline scenarios for an integrated usage of the platforms.

A first scenario is the quite common scenario of errors in the stored data. In this case the set of data on which the management systems work do not match the reality. Slots in the warehouse may be marked as empty or full, but in reality they are not. In this case a tracing through the whole delivery chain may be triggered. Goods may be tracked for their way through the yard and through the warehouse, which might reveal their true whereabouts. In addition an operator may check via video or pictures taken by cameras about the real state of slot in the warehouse. In such a scenario all three platforms must be tightly integrated to pass around the necessary data and to support the tracking process.

Another scenario which is quite related is the handling of bottlenecks. It may happen that certain shelves (which are intended to hold special goods) are full and cannot store additional goods of the same type. This will cause consequences for the warehouse management and also for the yard management. Trucks delivering the same types of goods may be directed to special parking places on which they cannot interfere with the regular processes in the yard or can even be sent to remote yards at different locations. Such a scenario also requires a tight integration of warehouse and yard management platforms and also of monitoring facilities.

Situations in which a rescheduling is necessary are not only the error conditions outlined above, but may also occur when the warehouse is run in special modes. One of these modes is the so-called just-in-sequence mode. This mode is used mainly in the automotive industry when parts of cars need to installed in a certain order in a construction factory. This order must be maintained at all costs otherwise the assembly process will fail. This results in the requirement that not only the construction must follow a special sequence, but also the warehouse needs to store the parts in exactly that sequence which in turn affects the scheduling of deliveries on the yard. The platforms for management must interact accordingly to guarantee the aforementioned just-in-sequence mode.

In addition as a side effect to support the scenarios outlined above the tracking of trucks throughout the yard in order to ensure they take the route they were assigned to is desirable. This requires an integration of the yard management system with the monitoring platform.

6.2 Challenges Targeted by INDENICA

6.2.1 Governance

A goal in the context of INDENICA is organization-wide reuse of services to reduce development costs. In order to reach this goal a Virtual Service Platform is introduced.

Appropriate policies have to be defined to track these goals. Examples for such policies on the VDSP level also influencing the level of the existing platforms are already shown in chapter 2.6.

Additionally there are policies only relevant for the VDSP level, e.g.:

- Services of the Virtual Service Platform shall be used preferably. Direct use of services of the underlying platforms is only allowed when services are not available through the Virtual Service Platform.
- For the user of VDSP applications VDSP is a black box. The usage of services of the existing platforms is not visible for him.
- Another high-level policy addresses the smooth operation of the integrated system. It is refined by a number of policies which include:
    - The remote maintenance video stream is prioritized when the warehouse system or the yard management system is in error state
    - The video stream of the yard reception is prioritized while a truck is doing reception process (Yard management <-> Remote maintenance)
    - The warehouse management gives goods storage process a higher priority, when there are too many delivering trucks on the yard.
    - The warehouse management gives the goods retrieval process a higher priority, when there are too many empty trucks on the yard.

7 Planned Usage of INDENICA Tools

In this chapter, we will outline the usage of the tools developed during the course of the INDENICA project, in order to build and use a Virtual Domain-specific Service Platform (VDSP).

The implementation of a VDSP involves several roles, as described in Section 2.6, i.e.:

- Platform Provider
- System Integrator
- Platform Integrator
- Application Developer
- System Administrator

To assist the various stakeholders in realizing applications based on VDSPs, the INDENICA project will provide several tools.

Goal-based Requirements Engineering Tool (GbRE): Enables the goal-based derivation of service platform requirements (c.f. D1.2.1, M12).

Variability Modelling Tool (VM): Centrally manages the variability information in the project (c.f. D2.2.1, M12).

Variability Engineering Tool (VE): Works in conjunction with the VM to resolve variability (c.f. D2.2.1, M12)

Decision Support Framework (DSF): Responsible for modelling architectural decision information (c.f. D1.3.1, M18).

View-based Modelling Tool (VbM): Relevant for describing the architecture of the service platforms. It will also be used to represent the effects of architectural decisions (c.f. D3.1, M12).

Variability Resolution Tool (VR): Used for instantiating concrete variants from the variability engineering (c.f. D2.2.1, M12).

Monitoring Rule Editor (MRE): Allows for the definition of rules that control and configure the monitoring components of the VDSP (c.f. D4.1, M12).

Adaptation Policy Editor (APE): Allows for the definition of policies that control and configure the adaptation capabilities of the VDSP (c.f. D4.1, M12).

Generation Tools (GT): The Generation Tools comprise different generators provided by various INDENICA tools such as the Decision Support Framework, Variability Modelling Tool, View-based Modelling Tool, Adaptation and Monitoring Rule Editor, etc., will produce code, configurations, and runtime directives of the VDSP.

Deployment Manager (DM): Instantiates a configured VDSP, as well as necessary artefacts of existing platforms (c.f. D4. 1, M12).

In the following paragraphs we will describe the responsibilities of the identified roles in the process of implementing a VDSP instance, supported by the provided tools, in greater detail.

The Platform Provider (PP) uses the *Variability Modelling Tool* (VM) to describe the current variability properties, as well as the variability bindings of the existing platforms. Furthermore, he is responsible for implementing a component identifying INDENICA-compliant monitoring events, if no event service exists.

The System Integrator (SI) utilizes the provided ROI-Models to scope the virtual platform. Then, the SI uses the *Goal-based Requirements Engineering Tool* (GbRE) to describe the requirements of the VDSP to be created. After elaborating the requirements, he uses the VM Tool to describe the required variability of the VDSP, and, using the *Variability Engineering Tool* (VE), he describes the dependencies between VDSP variability and variability of the existing platforms. The SI can furthermore extend the variability of existing Platforms using identified variability implementation techniques. After describing the variability properties, the SI uses the *Decision Support Framework* (DSF) Tool to express architectural decisions based on the requirements, generating architectural guidance (e.g., initial component model) and constraints, which are used as starting points for modelling the basic architecture of the VDSP using the *View-based Modelling Tool* (VbM). The VbM tool constantly checks the created model for errors based on the previously defined architectural constraints, and is furthermore annotated for variability. Moreover, the VbM tool incorporates monitoring and adaptation views, which allow the description and specification of desired monitoring and adaptation behaviour, based on the requirement model. Furthermore, the *Monitoring Rule Editor* (MRE) and the *Adaptation Policy Editor* (APE) are used to specify desired monitoring and adaptation behaviour. After modelling the VDSP architecture, the SI invokes the *Generation Tools* (GT) to generate common application code.

The Platform Integrator (PI) binds the variability of the virtual platform by applying the modelled decisions using the *Variability Resolution Tool* (VR). The VR is then used to generate artefacts for existing platforms. Moreover, the PI generates the VDSP instance based on the architecture model and bound variability. Finally, the PI deploys the VDSP instance, as well as existing platform artefacts, using the *Deployment Manager* (DM).

The Application Developer (AD) develops an application based on the VDSP instance and can specify additional adaptation policies using the APE.

The System Administrator (SA) monitors the current state of the service platforms using the Monitoring Dashboard. Furthermore, the SA is responsible for making adaptation decisions for the VDSP instance when prompted.

8 Evaluation Criteria

This section aims to define the methodology for measuring the progress of the project and the degree to which the project results contribute to the Use Case realization (i.e. the exploitation of the INDENICA tools in order to develop business scenarios derived from the experience and expertise of the INDENICA end-users). The evaluation methodology outlined here does not focus on establishing if project results are valid contributions to science. Contributions to science are measured using traditional impact metrics such as papers published, invited papers and talks.

The Use Case evaluation is divided into the following types of evaluation:

- Fit-for-purpose Evaluation: the techniques, models and tools developed in every WP will be tested against the requirements of the use cases (i.e. requirements validation);
- User Oriented Technical Evaluation: the qualities[9] of the software modules and algorithms/techniques developed by each WP will be estimated based on the measurable characteristics of the software produced by the project.
- Usability Evaluation: the user interfaces produced by the project will be evaluated by representative samples of the target end users.

The Fit-to-purpose Evaluation and the Usability Evaluation will be performed following the GQM methodology[10]. The GQM paradigm is a specific technology for goal-oriented measurement in software projects.

A GQM plan or GQM model document is the refinement of a precisely specified measurement goal via a set of questions into a set of metrics. Thus, a GQM plan documents which metrics are used to achieve a measurement goal and why these are used. The questions provide the rationale underlying the selection of the metrics. The relevant steps in a GQM plan are:

- Definition of the goals to be evaluated. They are usually closely linked to the business objectives of the organization for which the GQM plan is developed.
- Definition of the quality focuses. Quality focuses define all the aspects that should be evaluated in order to assess the accomplishment of the goal.
- Definition of variation factors and their impact on quality focuses. Variation factors are all that elements that can influence the degree of accomplishment of quality focuses[11].

[9] such as parameters present in quality models such as ISO/IEC 9126

[10] A more detailed presentation of the GQM paradigm can be found in (Basili et al., 1994) and (Fuggetta et al., 1998)

[11] For instance, the capability of system to support electronic payment can be influenced by the fact that users trust electronic payment or not.

- Definition of baseline hypotheses. These hypotheses act as the reference point against which the data gathered as a result of the evaluation phase are compared.
- Definition of questions and metrics. Each quality focus and variation factor is detailed in a number of questions that have to be answered in order to evaluate the accomplishment of the quality focus and the corresponding goal. The way questions are answered is defined by metrics.
- Collection and analysis of data.

For the Technical Evaluation we will use the MOSST methodology described in the QualiPSo project (4) and summarized below.

In the context of the IST project QualiPSo some work has been done in order to find a quantitative relationship between the perceived quality of OSS, from the end-user point of view, and a few simple objective measures. QualiPSo collected the users' and developers' evaluations of trustworthiness, reliability and other qualities of OSS products and correlated them to static code measures. As a result, QualiPSo produced a set of quantitative models that link static measures of the source code to perceivable qualities of OSS. These models can be used by end-users and developers of software products, who can set code quality targets based on the level of trustworthiness, reliability, etc. they want to achieve.

In the context of WP5 some of the perceivable quality of tools that would be hard to evaluate directly can be evaluated, using the QualiPSo MOSST methodology, on the basis of simple static code measures.

8.1 Set of Measurable Indicators for INDENICA

The evaluation of the outcome of INDENICA started from initial objective described in the Section 1.1.2 on page 10 of the DoW (25.10.1009):

- to tame the complexity of service platform development caused by fragmentation;
- to support platform convergence and interoperability to avoid the increased dependency on external service and platform vendors;
- to support automatic deployment, monitoring, governance and adaptation of services in a Virtual Service Platform.

Given this three main objectives of INDENICA and the three types of evaluation described above, in the following table define a list of measurable indicators for INDENICA. These measurable indicators can be further refined using the GQM and/or to the QualiPSo methodology in order to be better aligned with the Use Case requirements and with the end user expectations.

Table - High level measurable indicators for INDENICA and the related evaluation methodology

High-Level Measurable	Type of Evaluation	Evaluation Methodology

Indicators		
Support to platform convergence	Fit-for-purpose Evaluation	GQM
Support to openness, modularity and interoperability	Fit-for-purpose Evaluation	GQM
Avoid the increased dependency on external service and platform vendors	Fit-for-purpose Evaluation	GQM
Enable automatic deployment of services in a Virtual Service Platform	Fit-for-purpose Evaluation	GQM
Allow for monitoring of QoS-aspects of services in a Virtual Service Platform	Fit-for-purpose Evaluation	GQM
Support services governance and role-based governance in a Virtual Service Platform	Fit-for-purpose Evaluation	GQM
Support to tailoring and adaptation of Virtual Service Platforms	Fit-for-purpose Evaluation	GQM
Enable runtime-dynamism and runtime variability decisions for services in a Virtual Service Platform	Fit-for-purpose Evaluation	GQM
Expressiveness of the goal-based requirements engineering language	Fit-for-purpose Evaluation	GQM
Completeness of	Fit-for-purpose Evaluation	GQM

the decision model for platforms as a service		
Usability of the requirements engineering framework	Usability Evaluation	GQM
Usability of the decision support framework	Usability Evaluation	GQM
Usability of the Variability engineering tool	Usability Evaluation	GQM
Usability of tool suite for service platform engineering	Usability Evaluation	GQM
Trustworthiness of INDENICA tools[12]	User Oriented Technical Evaluation	MOSST QualiPSo Methodology

[12] According to (5) by "Trustworthiness" we mean the ensemble of qualities that can convince a potential user to adopt a software product, maybe preferring it to closed source competitors.

9 Further Case Studies

Some of the aspects of the INDENICA project cannot be evaluated using the main scenario. This chapter gives a brief overview of smaller case studies that will cover special cases, therefore exploring the frontier of the INDENICA scope.

9.1 Integration of existing PLC software

Today, much existing software automating / controlling production processes is realized on top of so called Programmable Logical Controllers (PLCs), that is, specialized embedded devices that contain a specialized runtime for factory automation. Software for such devices is usually written in programming languages specialized for PLC programming.

To integrate these controllers with higher-level systems, today either proprietary protocols or a standardized access model called OPC is used. Both kinds of access mechanisms use cyclic polling, therefore are not well suited to be integrated with service oriented systems.

A small case study will test how to bridge the gap between existing PLC software controlling a demo plant in Nuremberg called SmA) and the INDENICA platform.

9.2 Concepts for field devices

Field devices are deeply embedded systems, which have only restricted CPU power and memory. They are not powerful enough to run runtime configurable service platforms according to the INDENICA definition. Still, we want to be able to use the INDENICA virtual service platform concept to explore how to utilize development time service configuration mechanisms in such a context, and thus build field device applications more efficiently. Some particular challenges in this space are:

- Service components may require functional tailoring for deployment in a particular device, e.g. a ‘parameter service’ component may or may not apply security checks to validate access rights of external users trying to read or write a parameter value. Different versions of the component may thus expose different versions of its interface; expressed as additional methods or parameters to methods.
- Service components may require provisioning with device- or application-specific metadata definitions, e.g. a ‘parameter service’ needs parameter definitions to know which parameters are defined, or an error component needs to know which status, warning and alarm conditions the device can exhibit.
- Most components require other components to provide their service. Thus, they must be capable to correctly invoke the services of these other components, depending on the concrete features of that component, e.g. providing the proper parameters for a service invocation as described in the first bullet. Consistency between all components is difficult to achieve if done manually, but essential for the functioning of the entire system.

- Service components may require specific (partial) metadata content in other components, e.g. a ‘limit supervision’ component requires a particular subset of parameters, containing the limits to check against.
- The resulting system must adhere to non-functional requirements, such as real-time constraints or code base size constraints. We see a potential to validate such constraints early based a formal component model.
- The resulting system may need to be certified with respect to its functional safety. Safety critical code may have to adhere to standards such as MISRA, must not contain unused statements, must not depend on unsafe parts, and must be documented thoroughly etc. Since certification takes place at the code level, all these constraints must be followed when generating code. We believe that a formal device component model can be of help in ensuring all this to be in place.

The approach towards these challenges is – true to the idea of INDENICA – to use product line engineering techniques for the engineering of the device-specific service platform. The resource constraints of deeply embedded devices, and potentially functional safety requirements, pose limits to the variability mechanisms that can be applied at the code level alone; e.g. code size constraints or MISRA rules may conflict with unused platform code that is present in the device only because it is part of the service platform. Thus, we intend to focus on modelling variability in domain specific languages that describe the service platform, and generate the tailored specific service platform from such a description.

Table of Figures

References

1. The Open Group. The Open Group SOA Governance Framework Technical Standard. [Online] www.opengroup.org/projects/soa-governance.

2. Michael ten Hompel, Thorsten Schmidt. *Warehouse Management: Automation and Organisation of Warehouse and Order Picking Systems.* s.l. : Springer, 2006.

3. Microsoft. Introducing Windows Server AppFabric. *MSDN.* [Online] 2011. [Cited: 07 11, 2011.] http://msdn.microsoft.com/en-us/library/ee677312.

4. QualiPSo project portal. [Online] http://www.qualipso.org.

5. Luigi Lavazza, Sandro Morasca, Davide Taibi, Davide Tosi. *Predicting OSS Trustworthiness on the Basis of Elementary Code Assessment.* ESEM : s.n., 2010.

6. Heather Kreger, Jeff Estefan. Navigating the SOA Open Standards Landscape Around Architecture. [Online] www.opengroup.org.