DA3.6: Implementation of the PLM Process model for the Demonstrator on information management for tracking & tracing of products for recycling

Written by:
David Potter, INDYON GmbH,
Karl Hribernik, Robertino Solanas, BIBA

DELIVERABLE NO	DA3.6: Implementation of the PLM Process model for the Demonstrator on information management for tracking & tracing of products for recycling
DISSEMINATION LEVEL	**CONFIDENTIAL**
DATE	15.04.2008
WORK PACKAGE NO	WP A3: PROMISE EOL information management for tracking & tracing of products for recycling
VERSION NO.	V0.2
ELECTRONIC FILE CODE	DA3.6 v.0.2.doc
CONTRACT NO	507100 PROMISE A Project of the 6th Framework Programme Information Society Technologies (IST)
ABSTRACT	This deliverable (DA3.6) summarises the implementation of the PLM process model for the A3 demonstrator, in terms of scenes, use of PROMISE components and technology, as proposed in DA3.3 and DA3.4. The detailed results are given of the activities performed for the implementation including any deviations from the original plan.

STATUS OF DELIVERABLE		
ACTION	**BY**	**DATE** (dd.mm.yyyy)
SUBMITTED (author(s))	David Potter	15.04.2008
VU (WP Leader)	David Potter	15.04.2008
APPROVED (QIM)	Dimitris Kiritsis	16.04.2008

Revision History

Date (dd.mm.yyyy)	Version	Author	Comments
15.04.2008	V0.2	David Potter	Initial Draft

Author(s)' contact information

Name	Organisation	E-mail	Tel	Fax
David Potter	INDYON	david.potter@indyon.de	+44 23 9234 5152	+44 23 9259 2327
Karl Hribernik	BIBA	hri@biba.uni-bremen.de		
Robertino Solanas	BIBA	sol@biba.uni-bremen.de		

Table of Contents

List of figures

Abbreviations

Abbreviations used in this document:

BIBA	Bremer Institut für Betriebstechnik und angewandte Arbeitswissenschaft
BOL	Beginning of Life
CCP	CorePAC Control Point
DC	Device Controller component of the PROMISE Middleware
DSS	Decision Support System
ECP	Embedded Core PEID
EOL	End of Life
ERP	Enterprise Resource Planning
GUI	Graphical User Interface
IDE	Integrated Development Environment
JAXB	Java Architecture for XML Binding
J2EE	Java 2 Platform, Enterprise Edition
MDA	Model Driven Architecture
OBC	On-Board Computer
PAC	PEID Access Container
PDKM	Product Data and Knowledge Management
PEID	Product Embedded Information Device
PLM	Product Lifecycle Management
PMI	PROMISE Messaging Interface
PPS	Production Planning and Scheduling
RFID	Radio Frequency Identification
SOM	Semantic Object Model
SSDP	Simple Service Discovery Protocol
UDP	User Datagram Protocol
UML	Unified Modelling Language
UPI	Unique Product Identification
UPnP	Universal Plug and Play (UPnP™)
WMS	Warehouse Management System
WP	PROMISE Project Work Package
XMI	XML Metadata Interchange
XML	Extensible Markup Language

1 Introduction

1.1 Purpose of this deliverable

This deliverable describes the implementation of the PROMISE A3 demonstrator (PROMISE EOL information management for tracking & tracing of products for recycling) according to the design described in the preceding deliverables *DA3.3: Design of the EOL Demonstrator on Tracking and Tracing V2.0* and *DA3.4: Process model workflow description for the demonstrator Tracking and Tracing*.

This deliverable fulfils milestones *MA3.3, MA3.4 and MA3.6: Implementation of the Process Model for the demonstrator (M30, M34 and M40)* and is the summary result of task *TA3.6: Implementation of the PLM Process model for the demonstrator on information management for tracking and tracing of products for recycling.*

1.2 Objectives of the A3 Demonstrator

The objectives of the A3 Demonstrator application, which were first described in the deliverable DA3.2, are as follows:

- To use PROMISE technologies in combination with indoor and outdoor navigation systems in order to enhance the processing of plastics products identified for recycling.
- To increase the probability of recycling in response to legislative directives which demand reduction in waste and increased reuse of materials, at the same time reducing cost and increasing profitability.
- To track and trace materials and manage the availability, security, accuracy and integrity of all relevant product data at every stage of the recycling process.
- To use all available information during the EOL phase of the chosen product (e.g. car bumpers) to optimise decision making at input to the recycling process.
- To use all available information during the BOL phase of the resulting recycled material (e.g. granular plastic) to enable optimal use of the material.

The following paragraphs summarise the refinements which have been made in the detail of the scenario definition since the publication of the PROMISE deliverable DA3.2 “Documentation of applicability of technologies and process solutions”.

We decided to take care to avoid reproducing in the A3 requirements for the PDKM/DSS any product decision making functionality which typically exists in the Materials Management modules found in many ERP or WMS solutions.

Within the scope of the A3 scenario, a used plastic object for recycling arrives in the plant at what is seen to be its EOL. However the actual EOL event does not really occur until it is physically destroyed in either a mechanical or chemical process. Therefore we should not record an EOL event for this object until that action really occurs. In the case we act as a reseller and any material is passed physically unchanged to another enterprise, it is a PROMISE responsibility to record that transfer of ownership in the PDKM in order to assure that the object can be traced to its next owner.

In the case any used, new or partly processed material is received to be used in plastics re-processing, whenever that material has been consumed (destroyed) in the process, an EOL event should be recorded for each product that has been used.

Whenever a new plastic product is created at the end of any production process, whether it is an intermediate or final process, a BOL event should be recorded. This ensures the integrity of the product BOL data even when a partly finished material is shipped to another company.

We proposed that the combination of sensors and on-board computer (OBC) of a forklift equipped with the INDYON Track+Race®[1] system (http://www.indyon.com/trackrace.php) can be seen as an automated identification system that uniquely identifies a product based on its XYZ position coordinates. We have therefore positioned and enabled the INDYON Track+Race® vehicle equipment as a high function PROMISE PEID (PEID Type 4).

We previously proposed to optimise the use of recycled materials in new plastic compounds. We recognised that the focus of this demonstrator is on tracking and tracing. In addition, the Production systems, to which the new compound decision making applies, will be simulated in our demonstrator. Therefore we decided that this objective was no longer applicable for our demonstration scenario.

One further objective of the A3 PDKM/DSS functionality was to demonstrate the automation of the human decision making which is currently necessary to compensate for the lack of integration between the ERP, PPS and WMS systems in a typical plastics recycling company.

1.3 Acknowledgements

Although this document is credited to a small number of authors, it is only fair to emphasise that the results that it documents would not have been possible without the enthusiasm and high level of both competence and commitment from the excellent team of people who have been heavily involved in the design, development, integration and realisation of the A3 demonstrator:

BIBA: Karl Hribernik, Robertino Solanas, and Martin Schnatmeyer.

CIMRU: Hui Cao and Paul Folan.

INDYON: Rainer Vu-Dinh, Berthold Klose-Paril, Jonas Christ, Jörg Lehmann, Hermann Feigl and Andreas Plettner

[1] Track+Race is the registered trade mark of INDYON GmbH.

Figure 1: A partial view of the BIBA LogDynamics Lab

2 Description of the A3 Demonstrator

The PROMISE A3 Demonstrator has been established in two separate locations:

1. The demonstration hall of BIBA (Bremer Institut für Produktion und Logistik GmbH), in Bremen, Germany, in co-operation with the LogDynamics Lab (http://www.logdynamics.com/lab.html), which is a research and technology transfer centre for technologies such as RFID, sensors and telematics within logistics.
2. The demonstration hall of INDYON GmbH in Pöcking, near Munich, Germany.

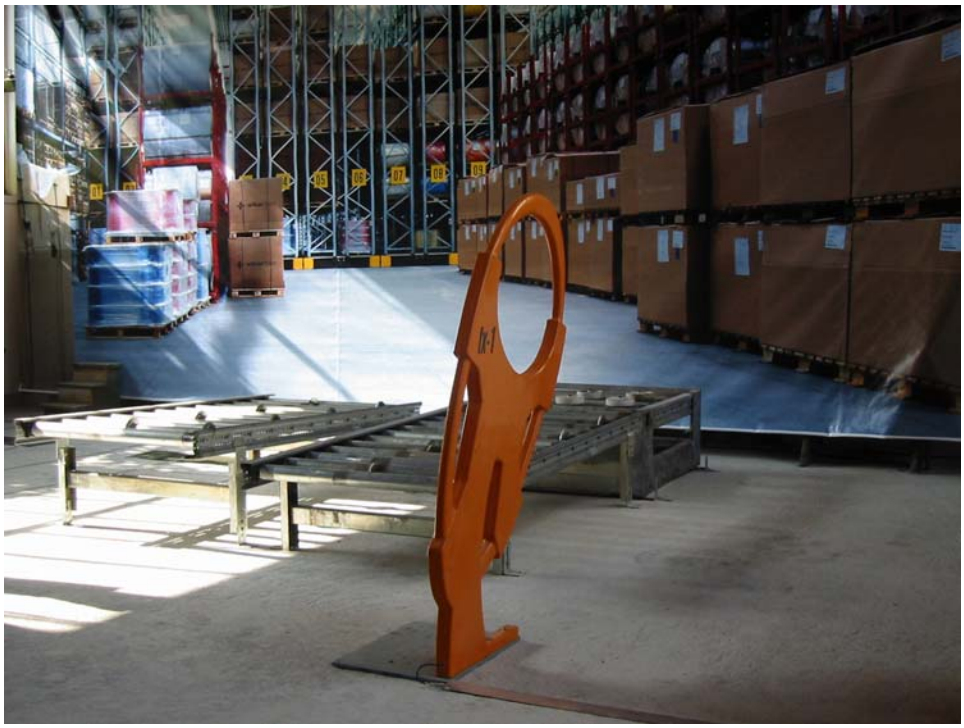

Figure 2: A partial view of the INDYON Demonstration Hall

2.1 Scenes implemented

The A3 Demonstrator design identified the following scenes or locations where the activities take place:

1. Incoming Goods**:** 1st stage identification of materials for plastics recycling
2. Sorting**:** 2nd stage identification to separate mixed materials
3. Clearings**:** 3rd stage identification for "unidentifiable" materials
4. Place in Storage**:** storage and transport (Warehouse Management)
5. Monitored Storage**:** storage and transport (Warehouse Management)
6. Production: materials re-processing
7. Outgoing Goods: despatch of finished products

The seven scenes anticipated in the original design concept have all been logically implemented in both the A3 Decision Support System (DSS) and the A3 Production Planning and Scheduling

(PPS) simulator. However, the physical implementation of the scenes has been adapted to the different physical layout of each of the two demonstrator halls in BIBA and INDYON.

2.2 BIBA

Incoming and Outgoing goods are co-located in a block storage area on the BIBA premises in front of the main loading bay behind the building, as shown in the picture alongside. The block storage area comprises 4x4 positions, logically split in Track+Race® to serve the demonstrator's purposes.

Sorting and Clearings are co-located in a separate block storage area, while normal and Monitored Storage are located in a shelving system in the BIBA hall/LogDynamics Lab. For production it is intended to make use of LogDynamics facilities.

Each of the co-located process areas is large enough so that the areas may be split into separate logical areas for the individual processes, thus avoiding overlap.

2.3 INDYON

Incoming Goods and Outgoing Goods locations have been initially co-located, and the differentiation between Incoming and Outgoing Goods depends on the direction of travel of the loaded forklift truck through the RFID reader gate shown in the adjacent picture.

As the forklift drives through the gate, the ID of the container of plastic materials loaded on the forklift is read. If necessary this is also filtered from any other RFID identifications which are detected within the container.

The Sorting and Clearings areas may also be co-located in the demonstrator as the A3 DSS user interface allows the user to choose either the Sorting or Clearings process at any one time. Therefore the implementation has the flexibility to support either separate or co-located areas.

The original design concept also anticipated that the physical area for the normal Place in Storage location needed to be separate from that of the Monitored Storage locations. This resulted from a lack of understanding at design time of the functionality of the UPnP enabled Embedded Core PEID (ECP). However we now know that the range of the ECP and its related "reader", the so-called CorePAC Control Point, means that we have the flexibility to store containers which are being monitored for potentially hazardous temperature conditions together with containers being stored under "normal" conditions.

2.4 FOCUS Components from DA3.3 Demonstrator Design

Section 5 of DA3.3 highlighted those key components in the A3 demonstrator scenario which have either a high PROMISE profile, and therefore strongly highlight PROMISE technologies, or which require a special focus because of the need for additional partner support, or both.

2.4.1 PDKM/DSS

The PROMISE PDKM/DSS emerged as the most important functional components in the A3 demonstrator scenario. It has the supervisory role of managing product data and events and coordinating product events and operational actions throughout all scenes of the scenario.

Because of the high focus on real-time, event-driven operations, the PDKM/DSS has a different role to play in the A3 scenario compared to the other PROMISE demonstrators, although several others also have a requirement for event handling from time to time. The A3 scenario also has a significant requirement for real-time feedback to 3rd party systems. This may be unique in the context of the PROMISE project demonstrators, but it is likely to be commonplace as a general requirement.

We also thought it would be especially important for our scenario that, for demonstration purposes, the PDKM/DSS can display key elements of their input, decision making and output in a clear and non-technical way so that an onlooker can easily see what the PDKM/DSS decided to do as a result of any data input or event.

We planned to work very closely with the involved technology partners to ensure that the functional characteristics of the A3 scenario could be met, and until May 2007 excellent progress was made. However by the end of October 2007 the situation had changed, forcing the A3 partners to re-evaluate their implementation plan and seek an alternative technical solution. This is described in more detail in the topic *Deviations from design proposed in DA3.3 and DA3.4* on page 8.

2.4.2 INDYON Track+Race® enabled for PROMISE operation

INDYON has enabled its existing Track+Race® system for PROMISE interfaces. This allows Track+Race® to communicate in a standard way with reader devices, the PDKM/DSS and 3rd party systems using common PROMISE middleware interfaces.

In addition, because once a product has been initially identified in a facility, existing Track+Race® functionality can uniquely identify that product repeatedly by its position within a warehouse or production facility.

Figure 3: INDYON Track+Race® On Board Computer (PEID:4)

Therefore the on board computer of a forklift equipped with Track+Race® is now enabled as a PROMISE PEID:4 which can identify or verify a product just by its position.

2.4.3 Sindrion® based PEID and Monitored Storage

The advantage of using of Sindrion®[2] based PEIDs for monitoring potentially hazardous materials while they are in storage is another key advantage of using PROMISE-enabled technologies. The possibility to pre-empt dangerous situations could lead to the possibility to negotiate reductions in industrial insurance premiums.

The packaging of the Sindrion® PEID used in the A3 demonstrator can be seen in the picture to the right.

In order to realise this capability, it is necessary to have the necessary UPnP PEID hardware together with suitable connected sensors, and a UPnP CorePAC Control Point for monitoring areas of rack storage. The latter will be described in more detail later in this document.

2.4.4 PPS-simulated EOL and BOL events

Because some key PROMISE EOL and BOL events take place in Production, it is important that, for demonstration purposes, our simulated PPS gives a clear visual indication of any significant event that is being simulated.

These are the key Product Life Cycle management events in the A3 demonstrator scenario and so it is vital that their visibility is highlighted as strongly as possible. It is also a clear indicator of the importance of identifying exactly where in any process, the real EOL and BOL events occur.

2.4.5 PROMISE Middleware

Although it is the least visible component in the PROMISE infrastructure, the middleware plays a crucial role in the A3 demonstrator, since it provides the communications layer between all other components and systems.

Because elements of the PROMISE middleware architecture were still to be defined at the design and the A3 scenario contains the greatest complexity in terms of component and system interaction, it was essential that INDYON, as a participant in WP R6 (Middleware), was able to both influence the work and cooperate with the other R6 partners.

In reality, this lead to the required architectural flexibility which will suit not only the needs of every PROMISE demonstrator, but also ensure a sound architecture foundation for each implementation beyond the life of the PROMISE project.

2.5 Deviations from design proposed in DA3.3 and DA3.4

Initially the A3 Demonstrator partners had been completely committed to using all of the standard PROMISE technology components, and in particular the PDKM/DSS based on technologies being supplied and adapted by SAP, InMediasP and Cognidata.

By May 2007, all required A3 DSS algorithms had been developed, unit-tested by CIMRU and submitted for integration, and the necessary PDKM initial test data had been entered by BIBA into the PDKM database running on an SAP provided server in Karlsruhe.

By the end of October 2007, however, very little progress had been made owing to a number of different causes:

[2] Sindrion® is a registered trade mark of Infineon Technologies AG.

1. External access to the Karlsruhe server was severely limited owing to recurring problems with first establishing and then maintaining access permissions through the firewall.
2. Data that had been entered into the server at considerable cost of time and effort was lost and had to be re-entered.
3. Expected progress by the technology providers was not made because the data was seen to be missing (ref. 2 above).

At the end of October 2007, the A3 demonstrator partners found that it was not possible to get any firm commitment from some technology providers for either the effort or timescales required to integrate the A3 DSS algorithms and GUI interface, or achieve stability of access to or persistence of data in the Karlsruhe PDKM.

Therefore we concluded that there was a very high risk of failing to complete our demonstrator if we continued to try to use the core PDKM and DSS technologies.

Instead, the A3 demonstrator partners were unanimous in proposing to develop an alternative implementation platform for the A3 PDKM using open source technologies and to re-use the stand-alone test shell that had been used to unit-test the A3 DSS algorithms as the basis for our own DSS.

This proposal was accepted and approved after some discussion by the Project Officer and the team of Reviewers during the 3rd Interim Technical Assessment meeting in Milan, Italy, November 1-2, 2007.

Consequently, the implementations of the PDKM and DSS components in the A3 demonstrator described in the following section are based on open source technologies.

3 Implementation of the A3 Demonstrator (TA3.6)

3.1 A3 Demonstrator Architecture

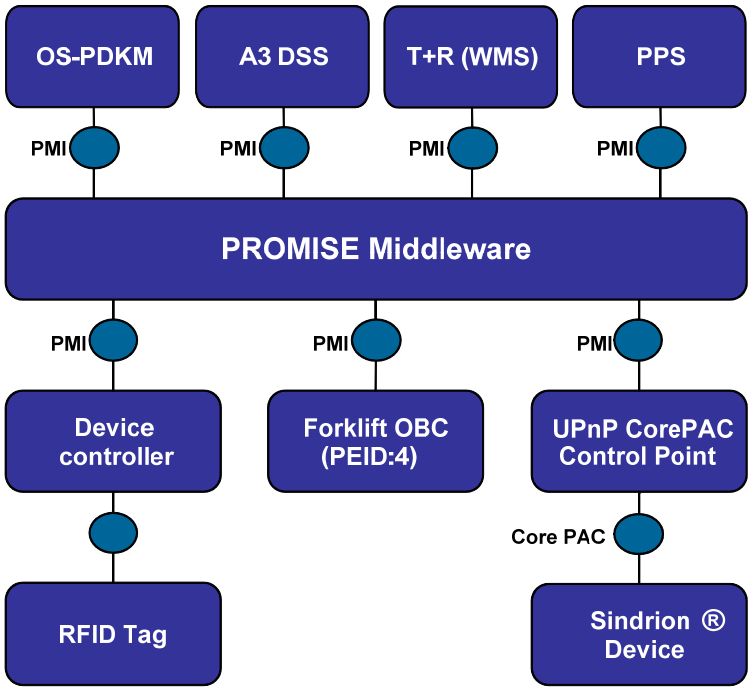

Figure 4: Final A3 Component Architecture

The final component architecture, as actually implemented in the A3 demonstrator, is shown in Figure 4 above.

The most fundamental significance is that of the Promise Messaging Interface (PMI), which has been implemented in the INDYON middleware, and in every other component that connects to it. This interface has allowed us to integrate all the required components using a common interface.

All the so-called “back-end” systems in the A3 demonstrator, the Open Source PDKM, the DSS, the Track+Race® system and the PPS Simulator are all able to make independent and parallel subscriptions.

Thus whenever data is available from a PEID, whether it is from an RFID source, a Sindrion® PEID or from the Track+Race® on board computer, that data is immediately available either in its entirety, or in subset, to multiple subscribers depending on their own individual subscriptions. This demonstrates the power and flexibility of the PMI and its future potential in a large scale implementation.

In the A3 demonstrator, the PMI has also been used as the common interface for integrating the different kinds of PEID found in our scenario. It permits the common integration of Device Controllers (DC) for RFID and UPnP technologies, the latter in the form of a very specialised DC, the CorePAC Control Point, and of course direct communication with a PMI-enabled PEID:4.

Each of the components represented in Figure 4 has been developed or adapted by different A3 partners, and time needed for the final proof of integration was three days, which again emphasises the potential for the PMI.

3.2 A3 Open Source PDKM

The PDKM system used in the A3 demonstrator has been developed by BIBA using open source components. The A3 PDKM supports both PMI Version 2 and PMI Version 3, although the PROMISE demonstrators were not required to implement Version 3.

This later PMI version has continued and will continue to be developed after the end of the PROMISE project. The decision to stabilise demonstrator development on Version 2 was to avoid the need for unnecessary re-work by multiple technology providers late in the project which could have led to problems that could impact the schedule for implementing the demonstrators.

Before the decision by the A3 partners to switch from the SAP-based PDKM/DSS foundation, BIBA had already begun to investigate an open source alternative, for two main reasons:

1. To create an open foundation that BIBA could use for continued PDKM research beyond the duration of the PROMISE project.
2. To support the PROMISE Standardisation efforts (WP I1) in being able to demonstrate that the PROMISE PDKM could be instantiated on a platform other than SAP.

Consequently, the decision and then approval to implement the A3 PDKM on open source components merely sharpened the focus and deadline for work that was already in progress, and therefore work was intensified on an already existing roadmap.

BIBA had already identified a new focus: to investigate the static data replacement of MySAP PLM as part of the overall PDKM concept in order to support continued research. This had also already led to the following:

- the PD(K)M represents the set of all distributed data and their transparent access and management
- there is a separation between dynamic and static data
  - dynamic (current) data are stored initially on the PEIDs (MOL)
  - static (long-living) data from BOL, MOL (already past) and EOL should be stored in Relational DBs
- focus on static data being equivalent to long-living data
- the present solution based on MySAP PLM is used to model and store long-living product life cycle data, providing a meta model and storing field data (MOL)
- investigate the replacement of the SAP foundation with open source components
- evaluate a standardized way for the generation of a persistence model

Therefore BIBA has followed a path to deliver a concrete realization of the PDKM using open source components and Java:

- Using relational DB (open source)
  - MySQL
  - Using the PROMISE semantic object model as basis to model concrete scenarios
  - as UML 2 Model → XMI
- Using Java / J2EE
- Using Hibernate
- Using WebServices (Axis) / Java Servlet / Apache Tomcat
- Automatic generation of (MDA)
  - a needed Java object model
  - Hibernate mapping artefacts
  - Communication
  - PMI (Promise Message Interface)
    - XML → Castor / Xerces / Xalan-XPath

Thus it has been possible to create an open source PDKM foundation using a combination of modelling tools and code generators as represented in the following diagram:

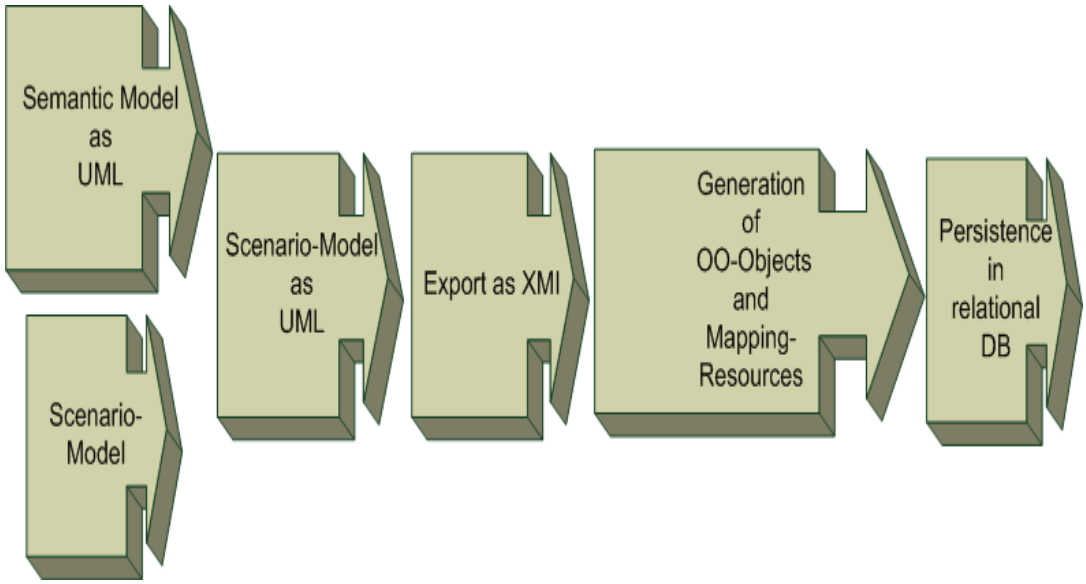

Figure 5: PDKM – Generation of a concrete product model

In order to be able to use an application specific model programmatically we also need an object orientated layer which represents the model's inherent defined concepts. To avoid redundant developing and modelling tasks an automated way is required to generate all necessary artefacts like Java classes and Hibernate mappings based on the SOM model one used here.

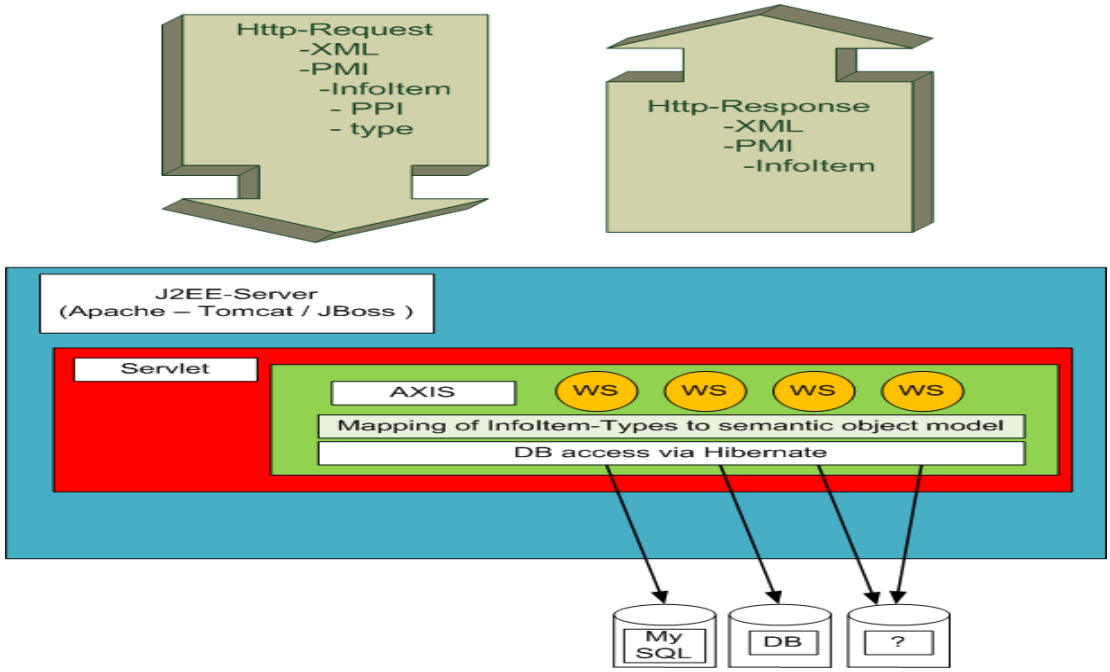

Figure 6: Concrete J2EE implementation of a PD(K)M

The goal was to use the SOM as the basis to be able to model application specific models, in this case the A3 demonstrator application, which has its specific needs in respect to the used field-data types and complex product compositions offered as templates. A standardized way in modelling

application specific models based on SOM and their automatically generation was one of the main goals of this approach.

This approach shows a highly abstract and standardized way to develop any application specific model which is based on a given SOM.

The open source PDKM has been developed using the following:

- System platforms :
  - OS independent but was developed on MS Windows XP
- Programming tools
  - IDE : Eclipse Version 3.3 - J2EE-Version
  - JAXB to handle PMI
  - MagicDraw 14 used as UML-Tool (SOM)
  - Hibernate-cartridge by Fornax (generates necessary hibernate mapping)
  - XML Spy by Altova
- DB components
  - MySQL 5.5. via Hibernate
  - but also DB-independent requiring only a different driver according to the actual database platform
- Run-time environment
  - Java Version 6
  - Apache Tomcat 6

The implementation of the A3 PDKM using open source components has proved the portability of the PDKM concept and clearly demonstrated that it need not be tied to a single vendor implementation. The resulting implementation fulfils all the needs of the A3 demonstrator, and at the same time has exercised a methodology which can be adapted to implement other data models and using further alternative technology platforms.

3.3 A3 Decision Support System (DSS)

The DSS system used in the A3 demonstrator has been developed by CIMRU using open source components. The A3 DSS supports PMI Version 2 for communication with the OS-PDKM, reading and writing the necessary information updates related to the containers and plastic objects.

In the initial stages of the project, CIMRU had developed each of the A3 DSS algorithms as Java beans to be integrated into the DSS framework supplied for the PROMISE project by Cognidata. This entailed an algorithm for each of the scenes of the A3 demonstrator plus an algorithm for handling full and empty container events during Sorting and Clearings.

All of the A3 DSS algorithms had been developed and unit-tested using a stand-alone test driver prior to being submitted for integration by Cognidata into the core PROMISE DSS framework. The first of the A3 algorithms was already integrated and was demonstrated live using the SAP Karlsruhe server during the Turin Review Meeting. All the remaining A3 algorithms were submitted for integration before the end of March 2007.

At the end of October 2007, no further progress had been made with the integration of the A3 algorithms or with the PDKM GUI integration owing to persistent problems with access via the firewall to the Karlsruhe server, and the fact that application data entered into the PDKM was not retained and needed to be re-entered. These problems led to the A3 partners taking an alternate open source direction as already explained earlier in this document.

In order to adapt the A3 DSS algorithms to run without the services of the DSS framework, CIMRU was able to extend the stand-alone shell that had been used for unit-testing, including the

enabling of its own GUI. Together with support from BIBA and INDYON, CIMRU was able also to enable the A3 DSS to use the PMI for communication with the OS-PDKM.

The figure below shows one of the A3 DSS GUI screens. This example shows a step from the Sorting process where type of plastic of the item being sorted has been positively identified and the DSS is controlling the process to ensure that the item is placed in the correct container for that type of material:

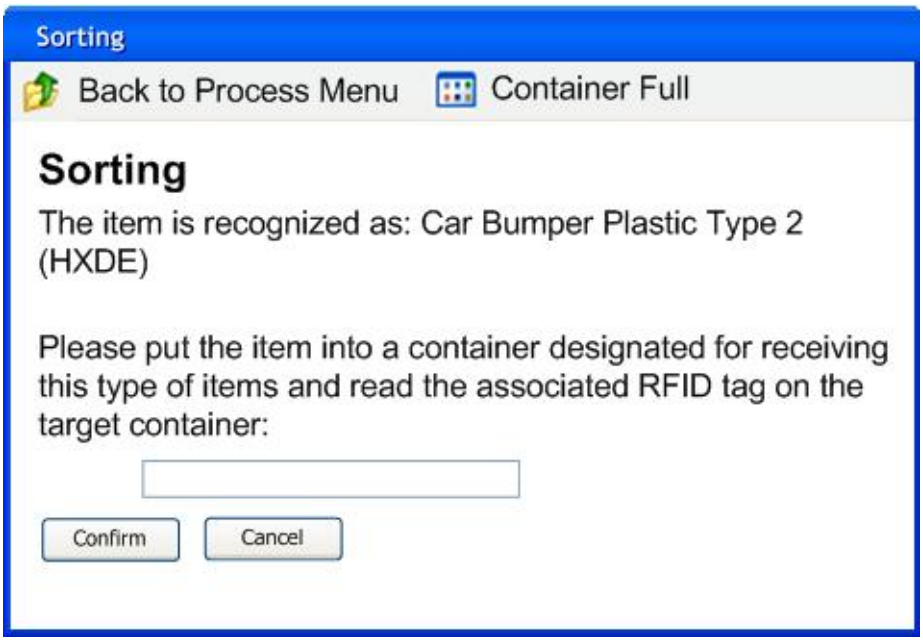

Figure 7: A3 DSS Sorting Process GUI Screenshot

A further example of one of the A3 DSS GUI screens is shown below. This screen controls the process to be followed in Outgoing Goods where the outgoing container is still equipped with the relatively expensive Sindrion® PEID. This need to be removed and replaced with a cheaper RFID tag before the container is shipped.

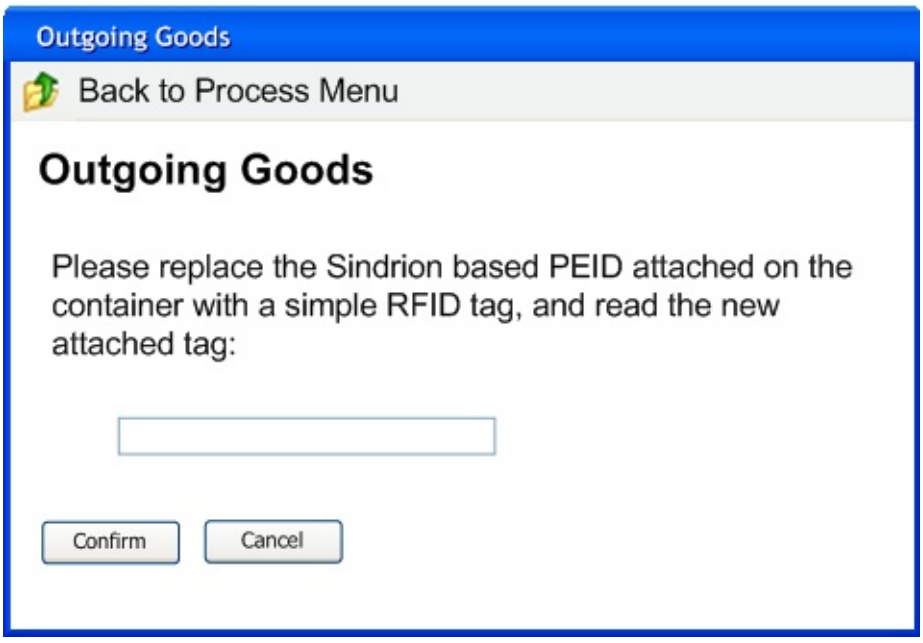

Figure 8: A3 DSS Outgoing Goods GUI Screenshot

The open source DSS has been developed using the following:

- System platforms:
  - Java/platform independent
- Programming tools:
  - Eclipse 3.3
- Run-time environment:
  - JDK 5.0 or above
- Web service server:
  - Apache Tomcat 5.5

3.4 INDYON Track+Race® System

The INDYON Track+Race® system has been enabled to use the PROMISE Messaging Interface (PMI) in order to allow it to exchange data with other PROMISE compliant systems. It uses PMI Version 2 to communicate and exchange data with the OS-PDKM.

The Track+Race® system subscribes to the OS-PDKM to query changes in process-state, and it writes new container position data to the OS-PDKM when a container movement is completed.

Driving orders related to movement requests decided by the A3 DSS are communicated to the forklift driver via the Track+Race® forklift terminal, as shown in the adjacent picture.

3.5 A3 PPS Simulator

The PPS Simulator used in the A3 demonstrator has been developed by BIBA using open source components. The A3 PPS uses PMI Version 3 for communication with the OS-PDKM.

The A3 PPS (Production Planning and Scheduling) Simulator functions as a distributed DSS component in the A3 implementation of the PROMISE architecture. It communicates with the other components by means of the exchange of PMI messages. Its primary function is to provide the operator with an overview of the types of plastic which can be produced at any given time. These production options are displayed in the top third of the main screen, as can be seen in Figure 9 below.

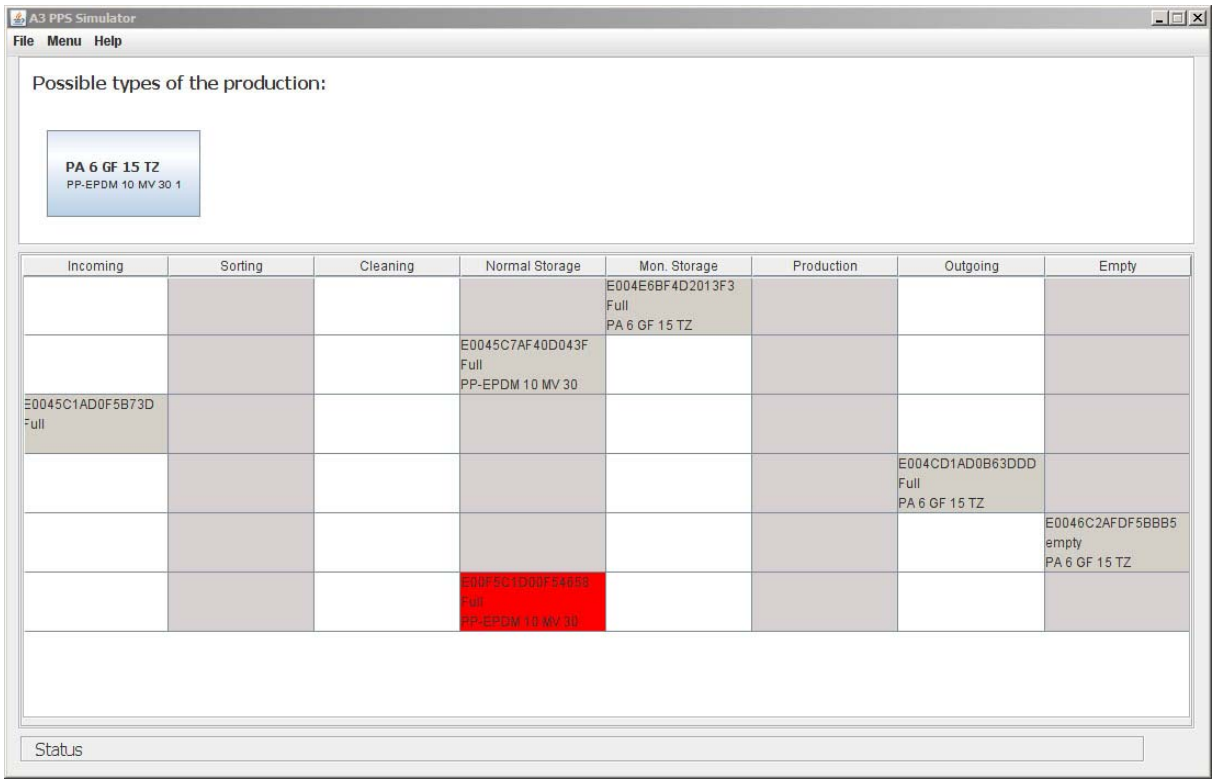

Figure 9: Screenshot of the A3 PPS Simulator

The bottom two thirds of the screen display the containers distributed throughout the scenes EOL processes at any given time. Each column displays a specific scene, from Incoming to Outgoing. Furthermore, a column for "empty" containers is displayed. Hazardous goods are highlighted in red. The data displayed in this part of the screen is kept up-to-date by means of a PMI subscription to the relevant InfoItems (PlasticType, IsHazardous, Space, Process) of each container in the EOL phase in the PDKM and DSS components.

PPS production rules are described in an XML format which can be loaded into the PPS to ensure flexibility.

Based on these rules, the PPS simulator analyses the current distribution of materials in their relation to the processes and the possible types of production. Feasible production options are displayed in the top third of the screen as described. The operator may initiate production of the materials these options represent by pressing the respective button in the application. This initiates a series of actions, which are communicated to the rest of the system by means of corresponding PMI messages:

1. Movement of the relevant containers to the production resources is triggered

2. Production of the specified plastic granulate is initiated
    a. The “input” plastic types are consumed – data in the PDKM is modified accordingly
    b. The “output” plastic granulate is produced – data in the PDKM is modified accordingly
3. Depending on the plastic granulate produced, the corresponding “IsHazardous” field is set in PDKM and the container designated for movement either to
    a. Monitored Storage if the produced plastics are volatile
    b. Normal Storage if not

The displayed distribution of plastic types is then updated in the GUI, and a new production process may be initiated.

The PPS Simulator has been developed using the following:

- System platforms:
    - OS independent but was developed on MS Windows XP
- Programming tools
    - IDE : Eclipse Version 3.3 - J2EE-Version
    - JAXB to handle PMI
    - Java-Swing
- Run-time environment
    - Java Version 6
    - Apache Tomcat 6 used to retrieve subscriptions

3.6 A3 Middleware

For the purposes of the A3 demonstrator, INDYON developed a PROMISE-compliant middleware to be the replacement for the existing communications layer of its Track+Race® system.

This is a restricted implementation which is limited to intra-enterprise communications; it does not have the full Inter-systems Communication (ISC) functionality that is necessary for inter-enterprise communication or communication with other middleware instances, nor does it have any functionality to support discovery of middleware entities outside of the enterprise.

INDYON has the option in future to extend the functionality of this restricted middleware implementation. Alternatively, since this restricted middleware exploits the PMI, the INDYON middleware could connect to another vendor middleware implementation to realise inter-enterprise communications.

The A3 Middleware has been developed using the following:

- IIS (Internet Information Services) and WSE 3.0 (Web service extensions) running on Windows XP Pro or Windows Server
- MS .NET Framework Version 2
- Tools:
    - MS Visual Studio 2005 Pro
    - JAVA Eclipse
    - XML Spy
- Database:
    - SQLite, MS-SQL, Kailua (MS .NET SQL-xxx Interfaces).

3.7 RFID Device Controllers

The A3 Demonstrator uses a number of different physical RFID reader devices:

1. An RFID gate reader at Incoming/Outgoing Goods
2. An RFID reader attached to the mast of the forklift truck
3. A handheld or table-top RFID reader at Sorting/Clearings

Each of these reader types must be integrated with the PROMISE infrastructure, for which purpose INDYON has developed an RFID Device Controller which implements PMI Version 2 on one side, and the reader-specific protocol on the other.

Each RFID device controller comprises two software components written in C# (.NET):

1. a server implemented as .NET web service, that handles peer communication, accepts PMI client requests and handles all synchronous requests. It can support HTTP, SOAP1 and SOAP2 messages. It can support secure communication through HTTPS and expose public interfaces in form of WSDL.
2. a daemon implemented as command line program (in future a Windows service) that polls the RFID reader and forwards requested subscription data to the corresponding peer (as a web service client).

Both components use a common set of library functions (DLLs) for processing of PMI messages. Persistent data and interprocess communication between the two components is realized with a SQLite database.

The present implementation limits PMI requests to the following restrictions:

- Only readmetadata-request and readdata-request (synchronous or as subscription) are allowed.
- No recursion within a request is allowed
- Only 1 “<targetdevices>” and 1 “<nodes>” is allowed.

3.8 CorePAC Control Point

The CorePAC Control Point (CCP) used in the A3 demonstrator has been developed by BIBA using open source components. The A3 PPS uses PMI Version 2 for communication with the OS-PDKM.

The CorePAC Control Point (CCP) is implemented as a Java library that supports reading from and writing to UPnP devices that feature the CorePAC interface. The library can be used in different types of applications. In the PROMISE project it is deployed as a web service and thus any invocation of its library functions will be relayed by the web service.

The library was created using the CyberLink UPnP library which is available at http://www.cybergarage.org/net/upnp/java/.

3.8.1 Connection between devices and the CCP

When the CCP initialises, it will automatically scan the network using the current network interface. It scans for UPnP devices using the SSDP (Simple Service Discovery Protocol). If a device is found, it will be queried for its services.

At present, a UPnP-enabled CorePAC device requires the following services:

- *"urn:schemas-upnporg:service:Content:1"*
- *"urn:schemas-upnp-org:service:Info:1"* and
- *"urn:schemas-upnporg:service:PTInfo:1"*

If one of these services is missing, the device cannot be recognized as a device supporting the CorePAC interface, and thus will be ignored. If everything goes fine, the device will be added to the CCP domain.

It is important that the CorePAC device and the CCP are using the same local network; otherwise they will not be able to contact each other since the UPnP discovery mechanism works via UDP multicast.

3.8.2 Disconnect between devices and the CCP

If for some reason the devices disconnect from the network, the CCP will be notified. A disconnected device will still remain added to the CCP, but any attempt to read or write data will fail. Once the same device reconnects to the network, the functionality will be restored.

3.8.3 Device identification

The current implementation requires the devices to have an unique identifier, and at present the device's url is used, although this might be changed in a future version.

If for some reason no such identifier exists, a device can still be identified using the UPI of the Core PEID that it contains. This however, will only work correctly if the connected devices do not include Core PEIDs with identical UPIs.

3.8.4 Device analysis

When the device has finally been added to the CCP, it will be scanned for its Core PEIDs and infoItems, using the methods 'GetUPIs()' and 'GetKeys(UPI:String)'. The latter method will be invoked for each UPI retrieved.

The values of the retrieved keys will not be read automatically. This will only happen if they are specifically requested. Thse methods will be invoked once per minute to update UPIs and keys since Core PEIDs might be removed from the devices or new ones added. The same is true for infoItems. This process is handled in a separate thread and does not interfere with any data retrieval of infoItems.

3.8.5 Data retrieval

As mentioned before, the infoItem data will be only retrieved when at least one specific request for an item has been made. There are two kinds of read actions:

1. The first retrieval request features a single invocation of the Read(UPI:String,Key:String)'
2. The second retrieval request will consequently invoke the Read(UPI:String,Key:String)' and compare its result with the previous result. If the value changes, the observers of the request will be notified.

To achieve this, the value of the information item has to be retrieved for as long as the request is outstanding. The default interval for data polling on an infoItem is at present 5000ms. However this can be easily altered to any other desired interval necessary. This is also handed in a separate thread.

3.8.6 Data manipulation

The data of infoItems can also be altered. This however depends on the infoItem itself, because information items may be read-only according to the specification of the CorePAC interface. If a write-request on such an item is attempted, it will be refused. If the value of an item is permitted to be altered, the 'Write(UPI:String,Key:String,Value:String)' will be invoked.

4 Conclusion

In principle all of the technical objectives of the A3 demonstrator have been met. There have been no technology barriers which have prevented the proposed implementation. The deviation from plan with regard to the Open Source PDKM and A3 DSS were not as a result of technology problems rather than the result of project management challenges.

The emergence of the PMI enabled a much simpler implementation of the A3 demonstrator by allowing the multiple components to communicate using a single interface. This is a clear and positive indication of the future potential of the PMI in complex integration projects to support closed loop information exchange.

Implementation of Open Source PDKM is also a clear indication that the PDKM is portable and not tied to a single vendor implementation.

The variation in the A3 demonstrator from the core PDKM and DSS components has also made it possible to demonstrate additional alternatives possible through the much enhanced PROMISE architecture.

5 References

1. Hermann Feigl, INDYON, Andreas Plettner, INDYON, David Potter, INDYON, Daniel Barisic, Infineon, Martin Schnatmeyer, BIBA, Paul Folan, CIMRU: DA3.3: Design of the EOL Demonstrator on Tracking and Tracing V2.0, PROMISE Project, February 2007.
2. Celal Dikici, BIBA, Belgin Cirkin, BIBA: DA3.4: Process model workflow description for the demonstrator Tracking and Tracing, PROMISE Project, October 2006.

