DA5.6: Implementation of the PLM Process model for the Demonstrator

Written by:
Cécile Corcelle, Jaime Krull (Caterpillar)
Daniel Barisic (INFINEON)
Youngseok Kim (EPFL)
Falk Brauer, Anja Klein (SAP)

DELIVERABLE NO	DA5.6: Implementation of the PLM Process model for the Demonstrator
DISSEMINATION LEVEL	**CONFIDENTIAL**
DATE	02.06.2008
WORK PACKAGE NO	WP A5: PROMISE MOL information management for heavy vehicle lifespan estimation
VERSION NO.	V9.4
ELECTRONIC FILE CODE	DA5 6 v9_4.doc
CONTRACT NO	507100 PROMISE A Project of the 6th Framework Programme Information Society Technologies (IST)
ABSTRACT	This deliverable (DA5.6) summarises the implementation of the PLM process model for the demonstrator, in terms of scenes, PROMISE components and technology implemented, as described in DA5.3 and DA5.4. The motivation for eventual discrepancies is given, together with the detailed results of the activities performed for the implementation.

STATUS OF DELIVERABLE		
ACTION	**BY**	**DATE (dd.mm.yyyy)**
SUBMITTED (author(s))	Cécile Corcelle	02.06.2008
VU (WP Leader)	Cécile Corcelle	30.05.2008
APPROVED (QIM)	Dimitris Kiritsis	03.06.2008

Revision History

Date (dd.mm.yyyy)	Version	Author	Comments
10.04.2008	V4	Cécile Corcelle, Daniel Barisic	With Infineon inputs & scenes' descriptions by CAT
14.04.2008	V5	Cécile Corcelle, Youngseok Kim	With EPFL inputs
15.04.2008	V6	Falk Brauer	Complement of MW-related sections (SAP inputs)
09.05.2008	V8	Jaime Krull	Editing and Modifications
16.05.2008	V9	Cécile Corcelle	Reorganising the technical content of data flow mechanisms and actual demonstration steps
19.05.2008	V9.2	Falk Brauer	Update on data flow mechanisms
27.05.2008	V9.3	Cécile Corcelle	Refinement and adding of DSS GUI sections from SAP
30.05.2008	V9.4	Falk Brauer, Anja Klein, Daniel Barisic, Cécile Corcelle, Jaime Krull	Final version

Author(s)' contact information

Name	Organisation	E-mail	Tel	Fax
Cecile Corcelle	Caterpillar	Corcelle_Cecile@cat.com	+33 47 62 38 213	
Jaime Krull	Caterpillar	Krull_Jaime_A@cat.com	1-309-578-5774	
Falk Brauer	SAP	falk.brauer@sap.com	+49 351 4811 6138	
Anja Klein	SAP	anja.klein@sap.com		
Daniel Barisic	Infineon Technologies	Daniel.Barisic@infineon.com	+49 89 234 2069	+49 89 234 955 5390
Youngseok Kim	EPFL	youngseok.kim@epfl.ch		

Table of Contents

List of figures

Abbreviations

BC	Board Computer
BOL	Beginning Of Life
BOM	Bill Of Material (parts identification list)
CAN	Control Area Network
CorePAC	Core PEID Access Container
DSS	Decision Support System
ECP	Embedded Core PEID
ECU	Electronic Control Unit
EOL	End Of Life
ET	Electronic Technician (CAT tool to extract ECU data)
GUI	Graphical User Interface
ID	Identifier
IT	Information Technology
MOL	Middle Of Life
NWA	?
PC	Portable Computer
PCB	Printed Circuitry Board
PDKM	Product Data Knowledge Management
PEID	Product Embedded Information Device
PLM	Product Lifecycle Management
PMI	Promise Middleware Interface
P/N	Part Number
RFID	Radio Frequency Identification
R&W	Read & Write
S/N	Serial Number
UPI	Unique Product Identifier
UPnP	Universal Plug and Play

1 Introduction

1.1 Purpose of this deliverable

Aim of this deliverable is to describe the PROMISE customized solution for the A5 application in term of developed technologies and functionalities demonstrated.

Based on previous scenes described in DA5.3, demonstration activities have been conducted and performed to demonstrate information flow between PROMISE components and therefore, the ability of the IT customized system to support the A5 PLM objectives, which is the implementation of predictive maintenance on structures of heavy machinery.

From what has been demonstrated in Sections 2 & 3, we will then be able to specify in DA5.7 remaining technical and business challenges for further implementation steps, from demonstrator scale to large-scale products.

1.2 Objective of demonstrator

Main objective of our demonstrator is the automation of the fatigue monitoring of machine's structural parts. The customized PROMISE architecture allowing the estimation of fatigue status and possible maintenance plan of a structure is composed of physical machine with PEID, Core PAC and PMI interfaces, back-end database called PDKM and Decision support system.

The predictive maintenance plan calculated by the DSS uses both field data sent by the PEID and manual inputs entered in the PDKM to propose maintenance plan for one machine.

2 Technical description of the demonstrator

First the technical solution will be depicted in term of IT system architecture and data flow mechanisms between PROMISE components to demonstrate the ability to perform the scenes of the demonstrator as explained in deliverables DA5.3 and DA5.4.

Then the implementation steps will be described that were conducted with A5 and A0 partners to effectively validate portions of the PLM model built for the A5 application case.

2.1 System architecture of the demonstrator

All PROMISE components are used in the A5 demonstrator, and full PROMISE compliance is created by satisfying all PROMISE interfaces (figure 1).

Main components of the A5 architecture are the Product Embedded Information Device (PEID), the Device Controller (DC) consisting of SAP's Device Handling Layer (DHL), the Request Handling Layer (RHL) also referred to as Data Services, the Product Data and Knowledge Management (PDKM) System and the Decision Support System (DSS).

Interfaces utilised in the configuration are the Core PEID Access Container (Core PAC), the PROMISE Messaging Interface (PMI) and Standard Query Language (SQL).

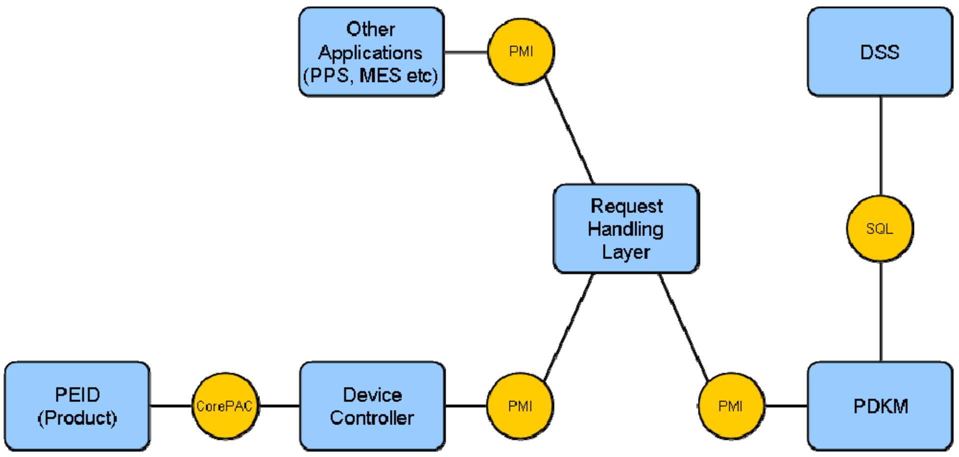


FIGURE 1: CONFIGURATION OF PROMISE COMPONENTS FOR THE A5 APPLICATION SCENARIO

In addition to the PROMISE standard components (RHL, DHL, PMI & Core PAC applications, PDKM back end, PDKM GUI), other pieces of soft ware were specifically developed for the A5 demonstrator (PEID hard and soft ware, Core PAC, DSS, DSS GUI, DSS mapping table, PDKM data model)

Finally, the system solution customized for the A5 demonstrator includes customized PEID (see §2.2) and PDKM data model (see §2.3) with customized soft ware to enable necessary dataflow mechanisms (see §2.5 and §2.6).

As shown in figure 2, items of the A5 system architecture is composed of:

Physical components / hard wares
- CAT machine including structural parts equipped with fatigue devices
- Embedded Core PEID (ECP) chips wirily connected to each fatigue device
- Proxy application to read fatigue data (executed on a PC/Laptop)
- Electronic Control Units of the CAT machine linked to the board of the machine

Soft ware
- SAP Middleware
    - Device Handling Layer (on local PC/Notebook)
    - Request Handling Layer (deployed on server in Karlsruhe)
- PDKM (database server in Karlsruhe)
- DSS (java program integrated to PDKM using SQL statement)
- Interfaces
    - CorePAC
    - PMI
- Graphical User Interfaces:
    - PDKM GUI
    - DSS GUI
    - DHL GUI
- Electronic Technician tool (executed on a PC/Laptop)

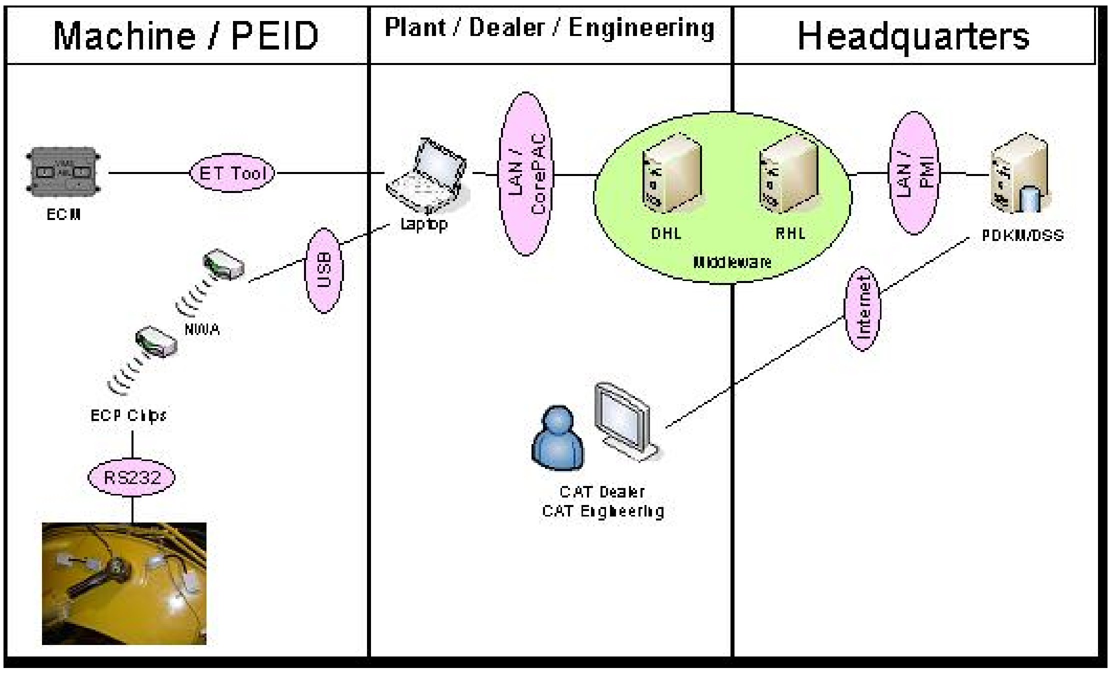


FIGURE 2: PROMISE ARCHITECHTURE AS IMPLEMENTED FOR A5 DEMONSTRATOR

2.2 Integration of customized PEID

2.2.1 "Crack First" PEID integration to the A5 architecture

For the demonstrator we employ two types of PEIDs. First, we use a "Crack first" sensor that is able to physically measure the fatigue status and a corresponding printed circuitry board (PCB), which allows interrogation of the sensor values in a wired fashion (using RS232). Both sensor and PCB have been developed outside of the PROMISE consortium. Second, a customized version of the Embedded Core PEID (ECP) that has been developed in WP R4 is employed. The ECP is able to acquire the fatigue data from the "crack first" assembly via RS 232 and to store part specific information. Read and write access to this information is given via a wireless link.

PROMISE compliant access to the ECP and "crack first" information is created by a proxy application that is executed on a PC/Laptop. The task of the proxy application is to implement the Core PAC interface, which will allow interaction with the PROMISE middleware. Further, it uses the NWA (developed in R4) to communicate to the ECP over the wireless link in order to read/write information.

The interface between ECP and PROMISE middleware is the CorePAC interface, which has been specified at month 18 in deliverable DR 4.3: "Specification of the Embedded Core PEID". An update version of the CorePAC interface was delivered at month 36 (DR4.5: Assessment and refinement of ECP specification), but has not been considered for the tests documented in this deliverable. This is not critical, since the CorePAC specification will only undergo minor changes. The amended specification will specifically remain backwards compatible, so that no integration problems need to be expected.

The interface between Middleware and PDKM is the Promise Middleware Interface (PMI). The version 1.0 of PMI has been specified in DR 6.5: "Interface definition and design of enterprise communication infrastructure" at M24. All updates of this specification will be documented in the architectural series. The PMI version 2.0 is now available (M33) and substantially extends v1.0

with improvements and enhancements obtained from the first implementation and integration experiences with the PROMISE demonstrators. All PROMISE demonstrators are currently being realized with PMI v2.0. In PMI v3.0 additionally elements for subscribing to events like PLM-events, device management events or system management events will be added.

2.2.2 Other PEID integration to the A5 architecture

For data collection about the machine use, a record of ECU information shall be transferred to the PDKM, for further use of the DSS to calculate estimated remaining lifetime of the machine structure. Alternate solutions have been investigated. First is to reuse the A2 PEID with use of the CAT Electronic Technician software and a specific PMI script to translate the ECU data to the "xml" PDKM readable format.

The second alternative is to directly read the MOL data on board the machine (number of running hours of the machine, total fuel consumption of the machine) to subsequently manually enter this information in the PDKM.

Not to duplicate customisation effort, this last solution was implemented for the A5.

2.3 PDKM data model

The following figure 3 shows the PDKM data structure built for the A5 DSS. Foreign key for the relationship among tables are omitted.

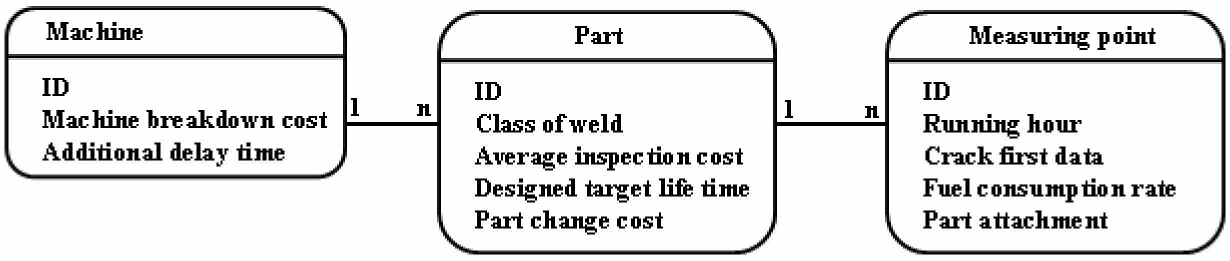


FIGURE 3: A5 PDKM / DSS DATA STRUCTURE

A5 data in PDKM for the DSS has three types of information: machine/part list, static information of part, and historical information of part.

In the machine list and the part list, ID distinguishes each machine and part, and each has static information:

- Machine: machine breakdown cost and additional delay time
- Part: class of weld, average inspection cost, designed target lifetime, and part change cost.

Each measuring point of a part has time series data. For each time stamp, "crack first" data, fuel consumption rate, and part attachment are stored in the PDKM.

Measuring points are field data that are either captured throughout the PROMISE A5 architecture ("Crack first" data, possibly fuel consumption rate and number of running hours) or manually entered by the DSS end-user (part attachment) during use of the CAT machine (MOL scenes).

Whereas static information is entered in PDKM back end during the set up of the machine/ part information, corresponding to scenes where information is first initialised (BOL scene) or modified following a maintenance event (MOL scene after machine repair with BOL information reset in PDKM backend)

A more complete PDKM data structure with additional static data was defined (DA5.4), considering supplementary information about the physical components than the ones used by the predictive maintenance DSS (example is "built date of the machine", "supplier code" of each structure).
In addition to the predictive maintenance DSS use, this BOL information recorded for the PROMISE compliant heavy machines would enable the capture of design efficiency information of segmented machine fleet, and lastly the development of a "DSS for design efficiency" module.

Due to the PDKM / DSS framework progress, the A5 implementation focused on our main objective, with a PDKM data structure built that matches DSS needs only.

2.4 Scenes included in the PLM model

Considering the 2 objectives of the A5 application described in DA5.4, the predictive maintenance on a structure and the design efficiency analysis on a structure, the following demonstration activities were required:

1. At BOL or during MOL in case of part replacement, initialisation of the PEID information on the PDKM and on the PEID itself;
2. During MOL, MOL data up-dated on the PDKM with data retrieved from the PEID, such as fatigue data, and with manual inputs entered in PDKM
3. During MOL, in case of repair or BOL information change (change owner, change lifetime design target), up-date ECP chips data and PDKM
4. During MOL, at any time, use the DSS to schedule maintenance operation on a PROMISE compliant machine and with the possibility to consider various application type (by entering either fuel consumption rate or attachment type) using DSS GUI
5. During MOL, historical information on a PROMISE compliant machine should be recorded onto the PDKM via the DSS
6. At any time, using the PDKM GUI, review the status of PROMISE compliant machines fleet

From these scenes, we can extract functionalities that need to be implemented and validated using the Promise architecture:

1. Initialise PEID data (DHL GUI)
2. Data transfer from PEID to PDKM (via CorePAC, DHL, RHL)
3. Data transfer from PDKM to PEID (via RHL, DHL, ECP)
4. Data capture in PDKM
5. Data capture from Fatigue Sensor (ECP, CorePAC)
6. Capture machine information in PDKM (PDKM GUI)
7. Update data into PDKM (PDKM GUI, PMI)
8. Transfer updated PDKM data to PEID (via RHL, DHL, CorePAC)
9. Read ECU Data (ET tool & PMI script, or directly on the machine board)
10. Read data from PDKM (SQL)
11. Enter DSS parameters (DSS GUI)
12. Run the DSS (PDKM)
13. View DSS results (DSS GUI)
14. View MOL historical information recorded onto PDKM (PDKM GUI)
15. View machines status for fleet management (PDKM GUI)

2.5 BOL data flow mechanisms

Two options can be used to initialize both the PDKM back-end and the ECP chips of the PEID with BOL data.
Prerequisite for both options are:

- PDKM data-structure for a new equipment is available
- Initial data that is not acquired from PEID is entered (or imported) using the SAP-Backend-System
- The PEID is write able

Steps by step processes with screenshots captured are detailed in annex 6.1.

Both options have as result that the PDKM sends a subscription-request to the middleware (RHL). The middleware schedules a subscription with the given subscription-interval. The middleware creates, from this time, periodical requests via the DHL to retrieve the adequate information from the proxy application (PEID). The Middleware retrieves the PEID's data and finally sends the data-result to the PDKM via a call back-address of the PDKM.

2.5.1 Initialization using DHL (Bottom-Up-Initialization)

First option is to use the DHL interface to initialize the PEID first, and then transfer the PEID data to the PDKM later on (Figure 4). This initialization procedure is preferred in the case that no Internet connection is available. Therefore the data is written to the PEID. Meanwhile a mapping from the PEID's info items is created at the first time the PEID connects to the PROMISE-infrastructure.

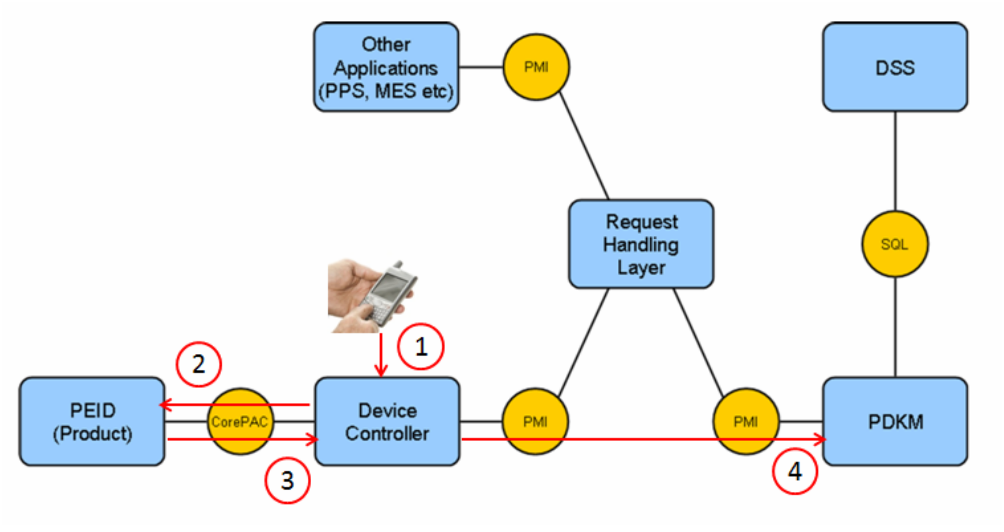


FIGURE 4: BOTTOM-UP-INITIALIZATION

In detail following initialization steps have to be applied:

a) PEID initialization:

- Navigate in DHLs service-structure to the content-service (urn:upnp-org:serviceId:content:1)
- Select the write-method of the PEID
- Enter the UPI's and PEID's ID and the desired data to be written

b) PDKM initialization:

- Log in to PDKM
- Navigate to 'MOL'-functions
- Check option: 'Enter Mapping Data'
- Search for the desired product instance
- Navigate to your equipment's measuring point
- Create a mapping between measuring point and info item
- Create a subscription by selecting the desired MW-node

2.5.2 Initialization using PDKM (Top-Down-Initialization)

Second option is to initialize the PDKM first with BOL data entered using PDKM GUI, and transfer this data to the ECP chips. This is the preferred technique if an internet-connection is available. Therefore, first, the mapping to the desired equipment's measuring-point to a PEID's info-item is created and next the write-method of the PDKM is used for writing initial data to the PEID and in parallel to the PDKM.

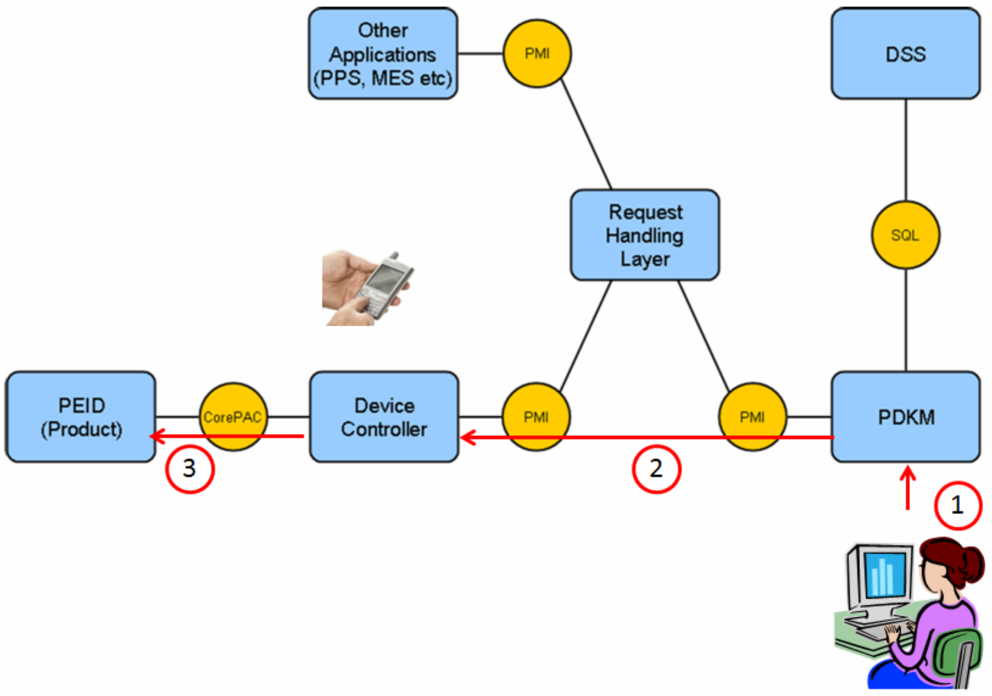


FIGURE 5: TOP-DOWN-INITIALIZATION

In detail following initialization steps have to be applied:

- Log in to PDKM
- Navigate to 'MOL'-functions
- Check option: 'Enter Mapping Data'
- Search for the desired product instance
- Navigate to your equipment's measuring point
- Create a mapping between measuring point and info item
- Create a subscription by selecting the desired MW-node
- Enter initial data to be written to the info item

2.6 MOL data flow mechanisms

2.6.1 MOL data update in the PDKM

The information about the structural part shall be read and written from the PDKM to the local storage of the part. For this, the PDKM will issue requests to the PROMISE middleware, which will finally create requests of its own, which will be directed towards the adequate PEID. This process is similar to that one described in previous section.

2.6.2 MOL data transfer from PEID to PDKM

The demonstrator requires the acquisition of the fatigue status of structural parts as well as storage of part specific information, such as a serial number, locally on the part. During MOL, the sensor information shall be recorded and stored in the PDKM. As result of the initialization process the middleware schedules a subscription with the given subscription-interval. As the proxy application is executed on a laptop it does not know the current fatigue status of the structural part and thus has to query the ECP over the wireless link. The ECP will then interrogate the "crack first" assembly in order to fetch the information and send it back to the proxy. The proxy will forward the information to the Middleware, which will finally give the requested information back to the PDKM via a call back-address of the PDKM, which was defined in the subscription.

2.6.3 Run the DSS with PDKM data

There are two types of input data for the DSS. First, the user can modify administrative parameters guiding the DSS algorithm. This data is valid for one DSS session. The permanent storage was not implemented in the prototype.
Second, the data stored permanently in the PDKM (e.g. PEID data) can be used as basis for the A5 DSS algorithm. During the DSS development there was a strict distinction between the GUI design (data representation, navigation, etc.) and the business logic implementation (DSS algorithms). The DSS modules *GUI* and *Logic* are connected via a web service interface, provided by the *Logic*-side. The start of the demonstrator calls the first web service of the Logic module, which extracts all available machines (see Figure 16) via JDBC from the PDKM database. After a machine is selected in the Maintenance part, a second web service provides the machine component details again via JDBC from the PDKM database (see Figure 18).
The modified admin parameters as well as the extracted PDKM data are shown to the user and build the base for the last web service, the DSS algorithm execution. This web service returns the DSS results, which describe recommended maintenance alternatives for the selected machine component (see Figure 19).

3 Implementation of the demonstrator

3.1 Implementation step 1: CAT machine instrumented with complete PEID at BOL

3.1.1 General description

Structural parts were first manufactured and the excavator boom equipped with four fatigue devices on critical areas. The excavator was then assembled and configured with attachments as requested by the final customer (standard bucket size). After assembly, as the machine was built, BOL data was recorded in CAT systems (machine serial number, attachment part number, built date, dealer code…). Other BOL data about the boom structure (critical locations, class of weld, design lifetime of each location) were identified for later entry in the DSS as parameters for determining remaining lifetime of the boom during its service life.

EPFL partner as recommended during PDKM training by the R9 team, built the PDKM data structure required by the DSS and entered a complete dataset, including these BOL data and estimated historical MOL data, in the PDKM backend.

3.1.2 PROMISE components used

Complete PEID composed of all physical components required to capture information necessary for DSS calculation of estimated remaining lifetime of the boom were used: boom structure equipped with fatigue mechanical devices with electronic units and ECP chips, on-board engine ECU recording engine totalised values and machine board displaying ECU totalised values.

Each fatigue device is made of 6 components (see figure 6)

“CrackFirst” device:
(1) “CrackFirst” mechanical device
(2) “CrackFirst” electronic unit
(3) Metallic protective box of the “CrackFirst”

“ECP” device:
(4) Electronic cable between the “CrackFirst” device to the “ECP” Radio Frequency device
(5) “ECP” Radio Frequency device
(6) Plastic protective box of the “ECP”

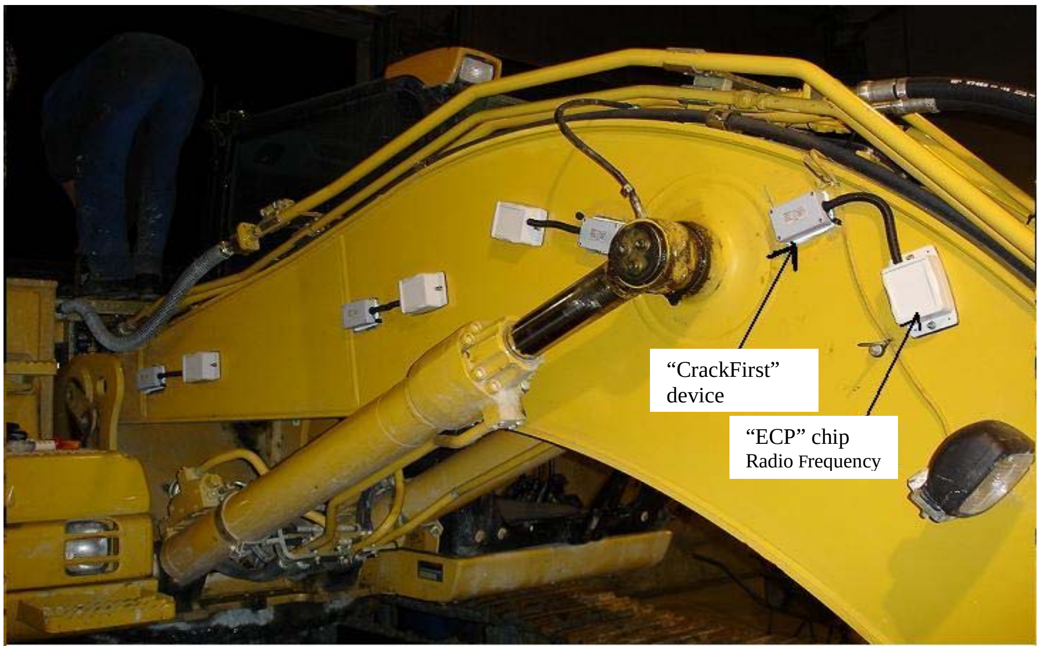


FIGURE 6: BOOM EXCAVATOR EQUIPPED WITH FOUR COMPLETE FATIGUE DEVICES (FEB-07)

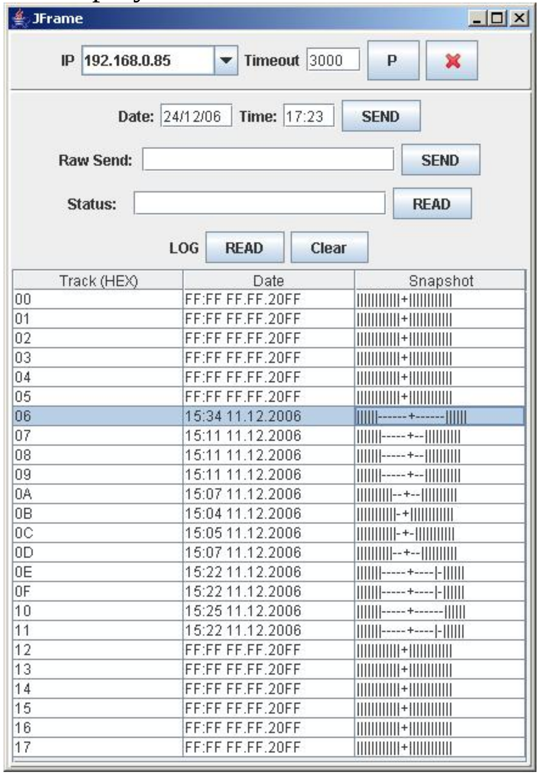
In the first phase of implementation the focus was on the physical realization of the crack-first/ECP combination on a machine. Figure 6 shows the actual field deployment of the set-up which has proven itself to be technically stable enough to be deployed on a machine under real working conditions (10 month deployment). Besides the physical set-up also the stability of wireless communication was tested. At this phase an off board laptop was used to initialise the ECP chips data with ID and to validate the read out of the fatigue data from each mechanical sensor over Radio Frequency. The laptop was situated in a central place in the service garage and was able to access the ECP thus proving to be suitable for real-life adoption. At this point in time the ECP was accessed via proprietary means over a simple GUI as functional tests rather than integration tests have been performed.

In a second implementation step, software has been implemented that allowed the "crack first " information to be available via the CorePAC interface. Further tests showed full integration of the "crack first" PEID with the PROMISE middleware (SAP implementation) and finally the PDKM, thus showing full PROMISE integration and compatibility.

3.1.3 Functionalities demonstrated

BOL initialisation processes have been demonstrated with the use of all PROMISE components (PEID, CorePAC, DHL, RHL, PDKM) to initialise the ECP chips with BOL data as well as to record BOL data in the PDKM back end.

For the first step implementation of the "crack first" and ECP hardware, a graphical user interface was developed (see figure 7). It is a simple Java application that can be executed on a PC. One can choose the IP address of the ECP that is to be queried. Afterwards static information like the time can be set via the GUI. Further the current fatigue sensor status or the complete status log (shown in figure 7) can be downloaded from the crack first/ECP combination. The broken cracks of the sensor are visualized as '-' and non-broken cracks by '+'. The GUI is simple yet powerful for field deployments with no connection to the PROMISE systems.



IP: 192.168.0.85   Timeout: 3000

Date: 24/12/06   Time: 17:23

Raw Send:

Status:

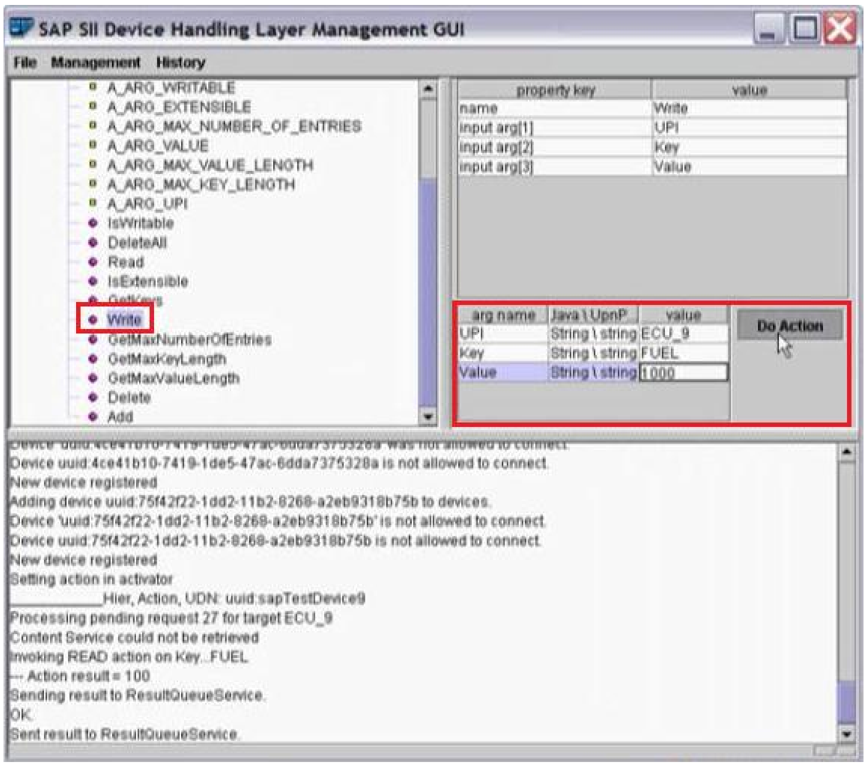
```
Track (HEX)  Date              Snapshot
00           FF:FF FF.FF.20FF  |||||||||||+|||||||||||
01           FF:FF FF.FF.20FF  |||||||||||+|||||||||||
02           FF:FF FF.FF.20FF  |||||||||||+|||||||||||
03           FF:FF FF.FF.20FF  |||||||||||+|||||||||||
04           FF:FF FF.FF.20FF  |||||||||||+|||||||||||
05           FF:FF FF.FF.20FF  |||||||||||+|||||||||||
06           15:34 11.12.2006  ||||||------+------||||||
07           15:11 11.12.2006  |||||||-----+--||||||||||
08           15:11 11.12.2006  |||||||-----+--||||||||||
09           15:11 11.12.2006  |||||||-----+--||||||||||
0A           15:07 11.12.2006  ||||||||||--+--||||||||||
0B           15:04 11.12.2006  |||||||||||-+||||||||||||
0C           15:05 11.12.2006  |||||||||||-+-||||||||||
0D           15:07 11.12.2006  ||||||||||--+--||||||||||
0E           15:22 11.12.2006  |||||||-----+----|-||||||
0F           15:22 11.12.2006  |||||||-----+----|-||||||
10           15:25 11.12.2006  |||||||-----+------||||||
11           15:22 11.12.2006  |||||||-----+----|-||||||
12           FF:FF FF.FF.20FF  |||||||||||+|||||||||||
13           FF:FF FF.FF.20FF  |||||||||||+|||||||||||
14           FF:FF FF.FF.20FF  |||||||||||+|||||||||||
15           FF:FF FF.FF.20FF  |||||||||||+|||||||||||
16           FF:FF FF.FF.20FF  |||||||||||+|||||||||||
17           FF:FF FF.FF.20FF  |||||||||||+|||||||||||
```

FIGURE 7: DEDICATED GUI TO INITIALIZE AND READ "CRACKFIRST" IN FIRST PHASE IMPLEMENTATION (FEB-07)

In addition to the unique identification of the PEID's (crack first/ECP chips), additional structural part information need to be written on both the ECP chips (PEID) and the PDKM back end, following these two steps sequentially:

1. Initialize PEID's data (Bottom-Up-Initialization using DHL GUI)
2. Data transfer from PEID to PDKM (via CorePAC, DHL, RHL, PDKM)

During initialisation, the information is acquired by the DHL-user interface (See Figure 8) in order to initialise a PEID without network-connection. As described in §2.5.1 the acquisition of information and storage in the PDKM is then facilitated.



property key	value
name	Write
input arg[1]	UPI
input arg[2]	Key
input arg[3]	Value

arg name	Java \ UpnP	value
UPI	String \ string	ECU_9
Key	String \ string	FUEL
Value	String \ string	1000

```
Device uuid:4ce41b10-7419-1de5-47ac-6dda7375328a was not allowed to connect.
Device uuid:4ce41b10-7419-1de5-47ac-6dda7375328a is not allowed to connect.
New device registered
Adding device uuid:75f42f22-1dd2-11b2-8268-a2eb9318b75b to devices.
Device 'uuid:75f42f22-1dd2-11b2-8268-a2eb9318b75b' is not allowed to connect.
Device uuid:75f42f22-1dd2-11b2-8268-a2eb9318b75b is not allowed to connect.
New device registered
Setting action in activator
____________Hier, Action, UDN: uuid:sapTestDevice9
Processing pending request 27 for target ECU_9
Content Service could not be retrieved
Invoking READ action on Key...FUEL
--- Action result = 100
Sending result to ResultQueueService.
OK.
Sent result to ResultQueueService.
```

FIGURE 8: DHL-USER INTERFACE – WRITE INITIAL DATA

3.1.4 Current and future refinement

Considering previous descriptions of the A5 demonstrator (DA5.3 and DA5.4) compared to the actual implementation performed as of today, main comments are the following.

1) PEID / Hardware components:

During development steps, investigation was conducted for having a common IT system for the two CAT demonstrators, A2 and A5, with a complete PEID interfacing with on-board CAT existing ECU and including Radio Frequency Read & Write system.
This first solution required the addition of a configured on-board computer with specific connection to both current on-board ECU and new RF system.
For demonstration purposes, and, as the on-board RF system is currently not cost effective, the off-board solution was chosen.
For further application, specifications of the off-board computer (hard and software's) could be integrated in existing on-board ECU, depending on the required level of automation of data transfer, in case access to the machine is difficult or fatigue damage is critical compared to frequency of data acquisition.

2) Interfaces / Data sources:

Although we investigated on developing specific interfaces between CAT existing systems to directly transfer BOL data to PDKM database, we decided to extract manually the required BOL data to use the developed alternate processes to enter information in PDKM that is using direct PMI format or entering manual inputs using DHL GUI or PDKM GUI (see §2.5).
Furthermore, as a customized CorePEID interface has been developed for the EOL demonstrator A2 (see in DA2.6) to enter information in PEID and PDKM, we could replicate the solution for the MOL demonstrator A5. We will then focus in this deliverable, on the description of the Fatigue information flow and Predictive maintenance diagnosis.
In future implementation, we should develop dedicated interfaces to extract data from existing CAT systems and then, use the PROMISE guidelines/processes to transfer automatically the extract files to the PDKM database.

3) RFID / part tracking:

Use of IDs and Bill Of Material up-date have been demonstrated in the CAT EOL demonstrator using RFID passive tags and CorePEID interface. This requirement in the MOL demonstrator will then be facilitated using DSS GUI manual entry to consider machine configuration (ie attachment type: “standard”, “light” or “heavy”) in the predictive maintenance plan.

4) ECU / engine data capture:

Use of ECU data has been demonstrated in the CAT EOL demonstrator using ET tool and script to translate ET outputs into “xml” PDKM compliant format. This requirement in the MOL demonstrator will then be facilitated using DSS GUI manual entry to consider machine payload (ie number of running hours, fuel consumption rate corresponding to “light”, “standard” or “heavy” application) in predictive maintenance plan.

3.2 Implementation step 2: BOL and MOL data up-date between PDKM and PEID

3.2.1 General description

BOL information has to be stored on PEID’s and in the PDKM (as described in §2.1.4.4). Additionally, MOL information has to be recorded at maintenance service interval, with fatigue information automatically transferred from PEID to PDKM and other MOL data manually entered or up-dated in PDKM using dedicated interface (as described in §2.6).

There are two ways for the MOL update of PDKM, depending on the web access around the machine. If the machine is serviced in a garage equipped with web access, fatigue data will be automatically transferred to the PDKM back end; otherwise, fatigue data will be buffered in service laptop for later retrieval and transfer to PDKM when service laptop will be connected to the web afterwards.

The service operator will manually enter all other information in a PDKM interface after required checks on the machine Payload information will be read from the machine board (total number of running hours, total fuel consumption) as machine configuration update will be determined considering previous information entered in PDKM back end (ie, BOL data, Attachment type)

During MOL, in case of repair or BOL information change (change owner), we will need to up-date ECP chips data with information entered into PDKM.

Historical information about the machine and the structural parts that are monitored should be also updated in PDKM by the service Dealer to record major information about the machine that is monitored. In that case, some BOL data and MOL data should be modified using either the BOL initialisation process (as described in §2.5) or the MOL up-date process (as described in §2.6).

3.2.2 PROMISE components used

Complete system architecture is used with ECP chips (PEID), CorePAC, DHL and RHL applications (MW), PDKM backend and PDKM front end with PDKM GUI.

3.2.3 Functionalities demonstrated

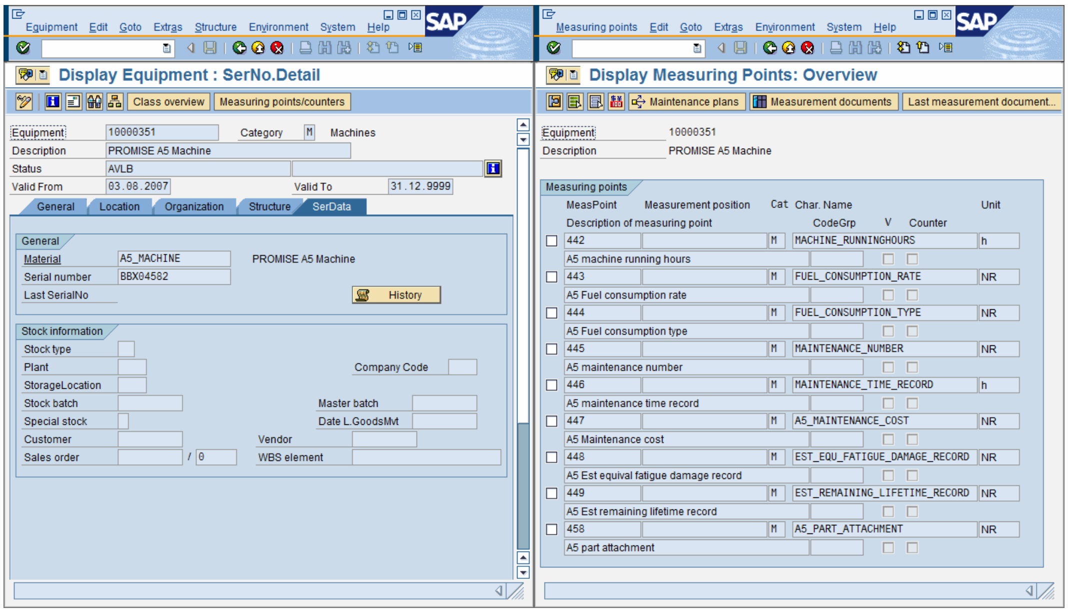
For demonstration simplification, we have demonstrated the ability of the ECP chip tag to handle limited BOL data, such as “A5_PART_NUMBER” and “A5_PART_attachment”, this BOL data being able to be up-dated, such as “A5_PART_attachment” in case attachment is changed by the Dealer or Customer of the machine (see scene 3 of DA5.3).

In this step, further functionalities of the system architecture and data flow mechanisms were demonstrated corresponding to the scenes when a maintenance operation is performed leading to modified part information with historical information entered in the PDKM (in case of repair or part replacement) and ECP chips information possibly updated as well (in case of part replacement).

To **modify structural part information** on both the ECP chips (PEID) and the PDKM back end, following two steps have to be performed sequentially:

1. Modify PDKM data (PDKM front end GUI)
2. Data transfer from PDKM to PEID (via RHL, DHL using CorePAC-interface)

The PDKM allows the manual entry of information to a specific “info item” which is located on a PEID. Initial information like e.g., the part’s serial number can be entered into the PDKM front-end GUI (see figure 9). When pushing the “write” button, the write request is forwarded to the middleware and then to the PEID (see figure 27 in annex).


Equipment Edit Goto Extras Structure Environment System Help
SAP
Class overview
Measuring points/counters
General
Location
Organization
Structure
SerData
History
Measuring points Edit Goto Extras Environment System Help
Maintenance plans
Measurement documents
Last measurement document...

Display Equipment : SerNo.Detail

Equipment	10000351	Category	M Machines
Description	PROMISE A5 Machine		
Status	AVLB		
Valid From	03.08.2007	Valid To	31.12.9999

General

Material	A5_MACHINE	PROMISE A5 Machine
Serial number	BBX04582	
Last SerialNo		

Stock information

Stock type			
Plant		Company Code	
StorageLocation			
Stock batch		Master batch	
Special stock		Date L.GoodsMvt	
Customer		Vendor	
Sales order	/ 0	WBS element	

Display Measuring Points: Overview

Equipment	10000351
Description	PROMISE A5 Machine

Measuring points

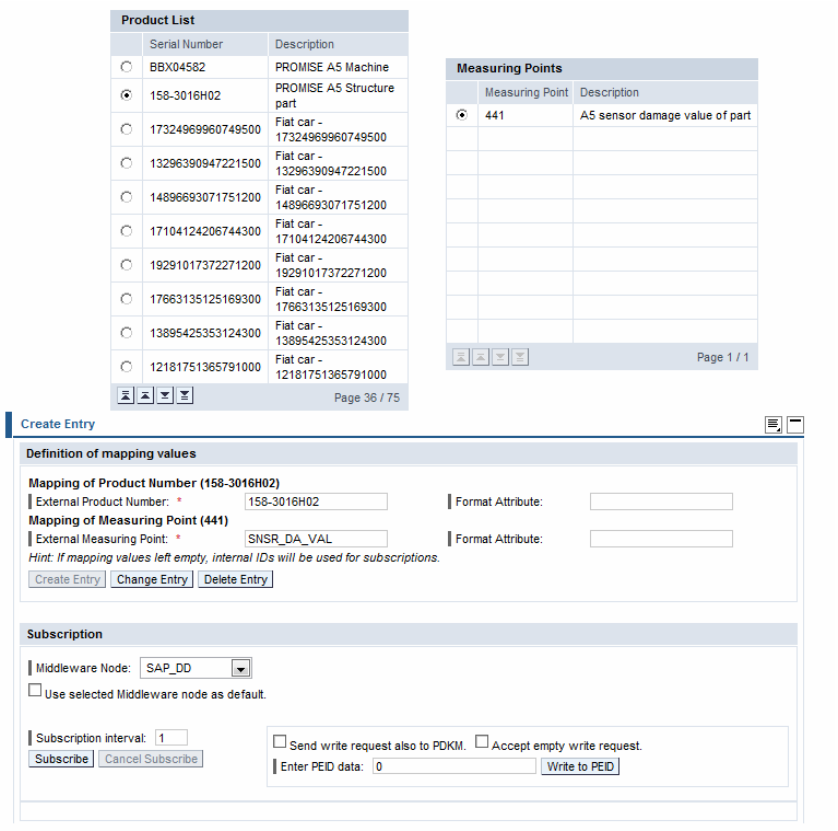
MeasPoint / Description of measuring point	Measurement position	Cat	Char. Name	Unit
442 / A5 machine running hours		M	MACHINE_RUNNINGHOURS	h
443 / A5 Fuel consumption rate		M	FUEL_CONSUMPTION_RATE	NR
444 / A5 Fuel consumption type		M	FUEL_CONSUMPTION_TYPE	NR
445 / A5 maintenance number		M	MAINTENANCE_NUMBER	NR
446 / A5 maintenance time record		M	MAINTENANCE_TIME_RECORD	h
447 / A5 Maintenance cost		M	A5_MAINTENANCE_COST	NR
448 / A5 Est equival fatigue damage record		M	EST_EQU_FATIGUE_DAMAGE_RECORD	NR
449 / A5 Est remaining lifetime record		M	EST_REMAINING_LIFETIME_RECORD	NR
458 / A5 part attachment		M	A5_PART_ATTACHMENT	NR

FIGURE 9: PDKM DATA STRUCTURE (SCREENSHOT)

Our evaluation has shown that the correct information is stored on the PEID. Thus, all relevant information that needs to be stored on the PEID can be entered into the PDKM (if necessary) and written to the PEID using the complete PROMISE architecture.

Data capture from Fatigue Sensor (ECP, CorePAC) was demonstrated using the complete and the following sequential steps:

1. In the PDKM data model for A5, measuring points for field data have been created (see Figure 9) containing initial BOL-data
2. In the PDKM front-end, subscriptions to these measuring points can be created (see figure 10). A subscription will cause the field data to be read either in a defined interval or every time the PEID comes into range.



Product List

	Serial Number	Description
○	BBX04582	PROMISE A5 Machine
●	158-3016H02	PROMISE A5 Structure part
○	17324969960749500	Fiat car - 17324969960749500
○	13296390947221500	Fiat car - 13296390947221500
○	14896693071751200	Fiat car - 14896693071751200
○	17104124206744300	Fiat car - 17104124206744300
○	19291017372271200	Fiat car - 19291017372271200
○	17663135125169300	Fiat car - 17663135125169300
○	13895425353124300	Fiat car - 13895425353124300
○	12181751365791000	Fiat car - 12181751365791000

Page 36 / 75

Measuring Points

	Measuring Point	Description
●	441	A5 sensor damage value of part

Page 1 / 1

Create Entry

Definition of mapping values

Mapping of Product Number (158-3016H02)

External Product Number: * 158-3016H02 Format Attribute:

Mapping of Measuring Point (441)

External Measuring Point: * SNSR_DA_VAL Format Attribute:

Hint: If mapping values left empty, internal IDs will be used for subscriptions.

Subscription

Middleware Node: SAP_DD

☐ Use selected Middleware node as default.

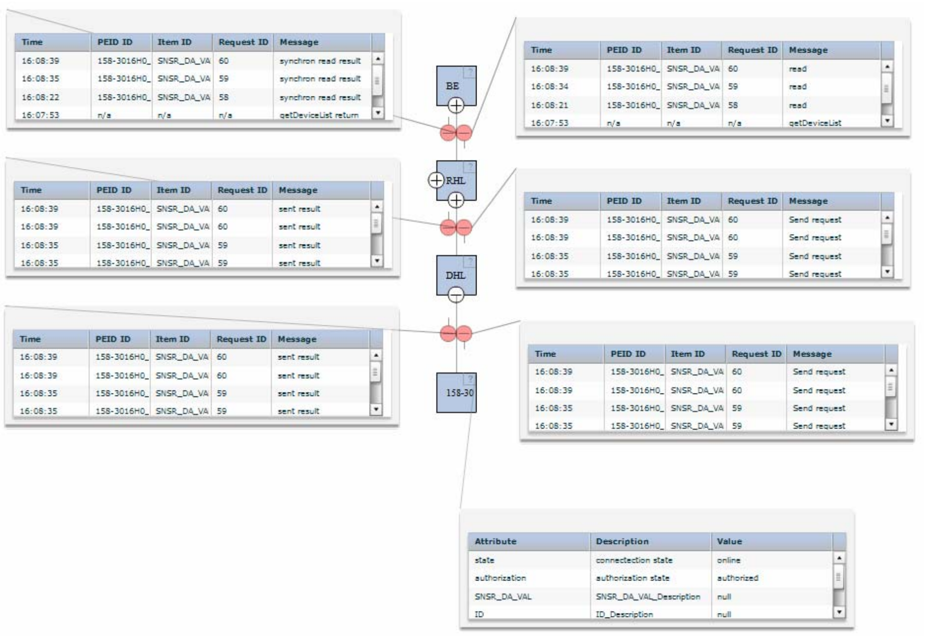
Subscription interval: 1

☐ Send write request also to PDKM. ☐ Accept empty write request.

Enter PEID data: 0

FIGURE 10: SUBSCRIPTION FOR FATIGUE DATA (SCREENSHOT)

Our evaluation has shown that the correct field data is read from the PEID in the given intervals and stored in the PDKM database. This validates that data, including the fatigue status of structural parts, can be acquired and proves full vertical integration of the PROMISE system from PEID to PDKM.



Time	PEID ID	Item ID	Request ID	Message
16:08:39	158-3016H0_	SNSR_DA_VA	60	synchron read result
16:08:35	158-3016H0_	SNSR_DA_VA	59	synchron read result
16:08:22	158-3016H0_	SNSR_DA_VA	58	synchron read result
16:07:53	n/a	n/a	n/a	getDeviceList return

Time	PEID ID	Item ID	Request ID	Message
16:08:39	158-3016H0_	SNSR_DA_VA	60	read
16:08:34	158-3016H0_	SNSR_DA_VA	59	read
16:08:21	158-3016H0_	SNSR_DA_VA	58	read
16:07:53	n/a	n/a	n/a	getDeviceList

Time	PEID ID	Item ID	Request ID	Message
16:08:39	158-3016H0_	SNSR_DA_VA	60	sent result
16:08:39	158-3016H0_	SNSR_DA_VA	60	sent result
16:08:35	158-3016H0_	SNSR_DA_VA	59	sent result
16:08:35	158-3016H0_	SNSR_DA_VA	59	sent result

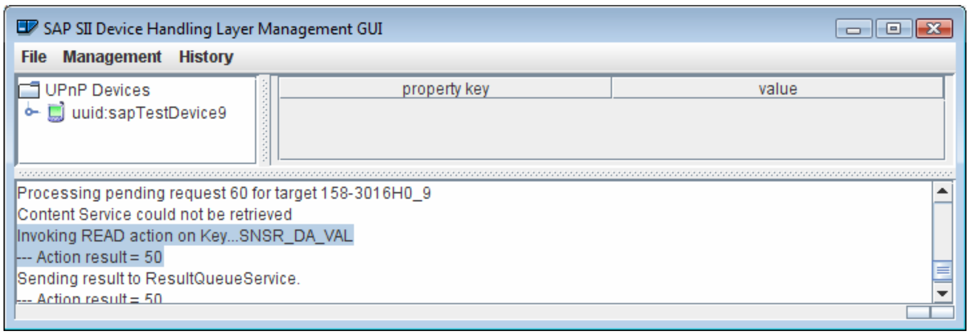
Time	PEID ID	Item ID	Request ID	Message
16:08:39	158-3016H0_	SNSR_DA_VA	60	Send request
16:08:39	158-3016H0_	SNSR_DA_VA	60	Send request
16:08:35	158-3016H0_	SNSR_DA_VA	59	Send request
16:08:35	158-3016H0_	SNSR_DA_VA	59	Send request

Time	PEID ID	Item ID	Request ID	Message
16:08:39	158-3016H0_	SNSR_DA_VA	60	sent result
16:08:39	158-3016H0_	SNSR_DA_VA	60	sent result
16:08:35	158-3016H0_	SNSR_DA_VA	59	sent result
16:08:35	158-3016H0_	SNSR_DA_VA	59	sent result

Time	PEID ID	Item ID	Request ID	Message
16:08:39	158-3016H0_	SNSR_DA_VA	60	Send request
16:08:39	158-3016H0_	SNSR_DA_VA	60	Send request
16:08:35	158-3016H0_	SNSR_DA_VA	59	Send request
16:08:35	158-3016H0_	SNSR_DA_VA	59	Send request

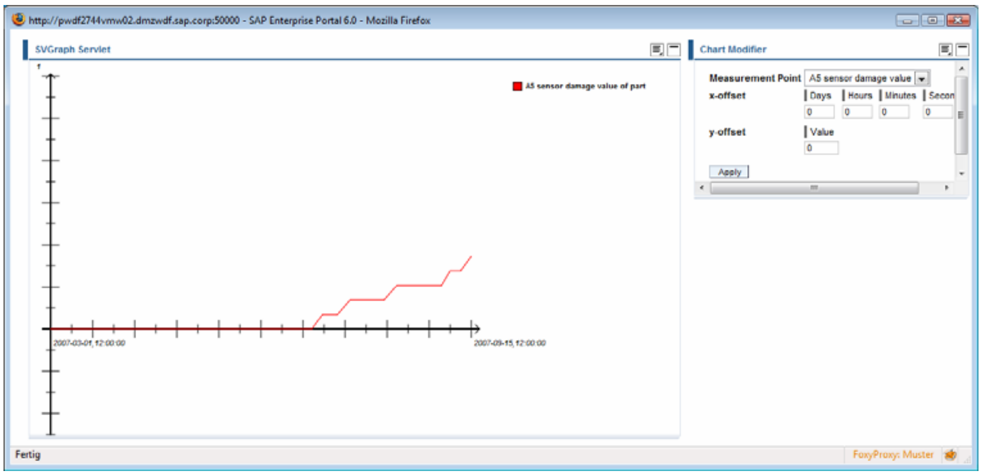
Attribute	Description	Value
state	connectection state	online
authorization	authorization state	authorized
SNSR_DA_VAL	SNSR_DA_VAL_Description	null
ID	ID_Description	null

FIGURE 11: DATAFLOW MONITORED IN SAP'S MOMA-CONSOLE



```
Processing pending request 60 for target 158-3016H0_9
Content Service could not be retrieved
Invoking READ action on Key...SNSR_DA_VAL
--- Action result = 50
Sending result to ResultQueueService.
--- Action result = 50
```

FIGURE 12: READ REQUEST ANSWERED BY DHL



http://pwdf2744vmw02.dmzwdf.sap.corp:50000 - SAP Enterprise Portal 6.0 - Mozilla Firefox
SVGraph Servlet
A5 sensor damage value of part
2007-03-01, 12:00:00
2007-09-15, 12:00:00
Chart Modifier
Apply
Fertig
FoxyProxy: Muster

Measurement Point: A5 sensor damage value
x-offset: Days 0, Hours 0, Minutes 0, Secon 0
y-offset: Value 0

FIGURE 13: DATA RECEIVED BY THE PDKM

While Figure 11 shows the overall data-flow from the PEID to the DHL over the RHL to the PDKM, Figure 12 shows that data was read by the DHL and Figure 13 shows the corresponding result-graph in the PDKM.

3.2.4 Current and future refinement

A common IT solution for the two CAT demonstrators has been discussed and should be easily developed given the synergies between the two demonstrators.
This means technically to combine the two PEID's with a laptop configured for accessing both ECP chips and RFID tags, in addition to the possibility to connect to the CAT on-board ECU's.

The PDKM data structure for both demonstrators should be redesigned with a merged PDKM data structure built for each demonstrator, allowing all scenes of each demonstrator to be performed under the same IT architecture with the use of the CorePEID interface developed by the BIBA, in particular.

3.3 Implementation step 3: DSS Use for Maintenance Decision

3.3.1 General description

During MOL, the DSS is used to plan maintenance services on a PROMISE compliant machine. Combination of application type and machine configuration can be evaluated thanks to DSS parameters modified in DSS administration module of the DSS GUI.

DSS should allow as well the operator to view historical information on a PROMISE compliant machine.

During regular service maintenance on a PROMISE compliant machine at given frequency, say at every specified unit time, i.e., every 960 hours, the information in PEID is uploaded into PDKM system. The DSS manager changes required parameters in DSS GUI if needed, for example, if part change cost has decreased or inspection cost has increased.

Based on the data retrieved from PEID and updated parameters manually entered using DSS GUI, DSS generate alternative maintenance scenarios. With the result, the customer and machine inspector discuss what the optimal solution is at this time: change part now, change part in-between to next regular inspection, do nothing until next regular inspection.

3.3.2 PROMISE components used

Assuming the fatigue information has been recorded in the PDKM backend, the following PROMISE components are used to run the DSS at the service Dealer location:

- PDKM backend
- DSS java program
- Mapping SQL statements for DSS program integration to the PDKM database
- DSS GUI

The DSS module is developed with java platform. It is composed of 3 parts: fatigue data calculation module, maintenance alternative generation module, and input/output module.
The first two modules inherit 'Action' class, which are PROMISE standard module specification.
The input/output modules are for the information passing from/to the first two modules.
User inputs and data from PDKM are passed through input module and the result is received by output module. The returned output modules are utilized to display the DSS result.

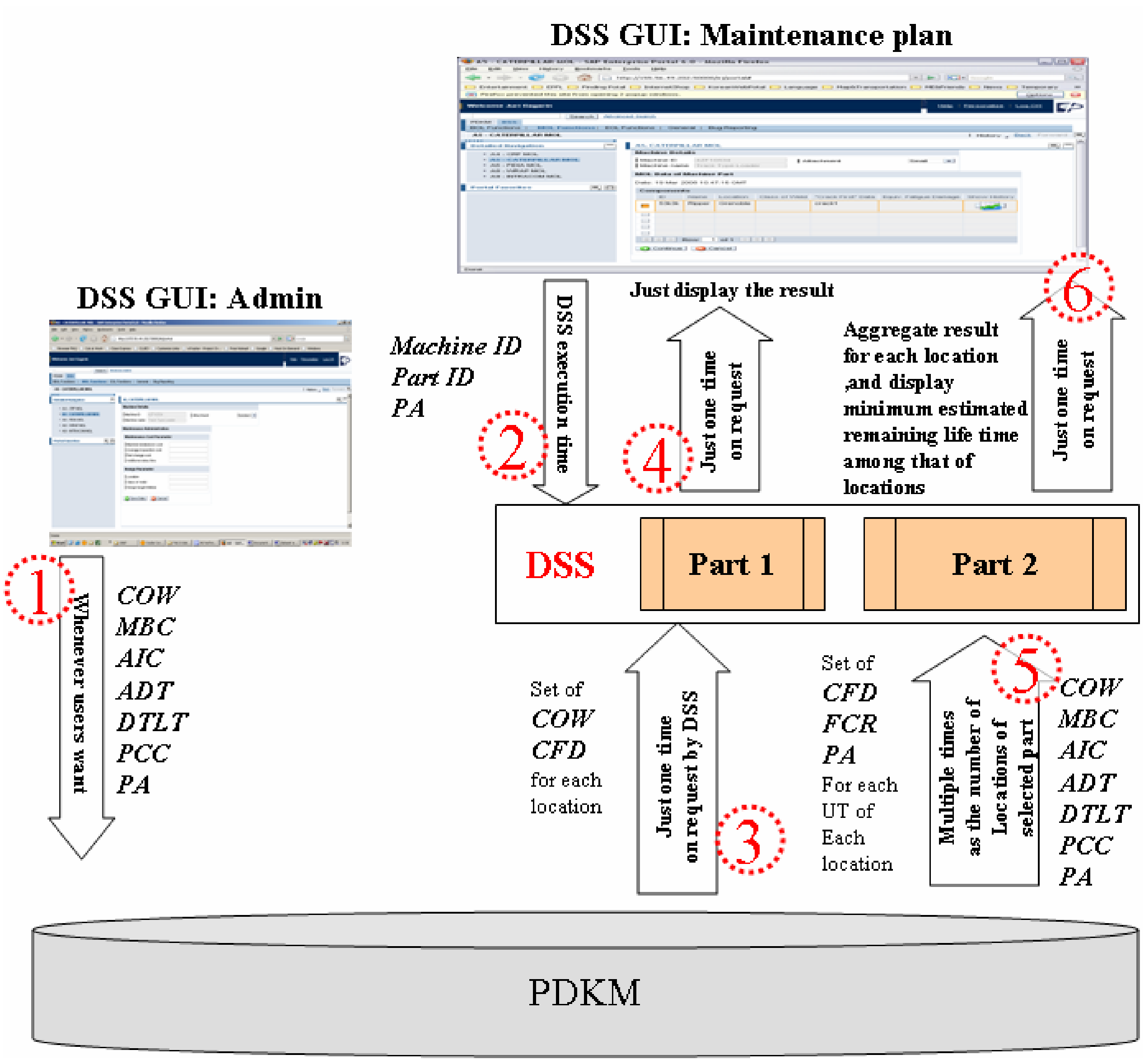

DSS GUI: Maintenance plan
DSS GUI: Admin
Machine ID
Part ID
PA
DSS execution time
2
Just display the result
4
Just one time on request
Aggregate result for each location ,and display minimum estimated remaining life time among that of locations
6
Just one time on request
DSS
Part 1
Part 2
1
Whenever users want
COW
MBC
AIC
ADT
DTLT
PCC
PA
Set of COW CFD for each location
Just one time on request by DSS
3
Set of CFD FCR PA For each UT of Each location
Multiple times as the number of Locations of selected part
5
COW
MBC
AIC
ADT
DTLT
PCC
PA
PDKM

FIGURE 14: DATA FLOW AMONG DSS MODULE, PDKM, AND WEB GUI.

Figure 14 shows data flow during the DSS use. In the figure, the part 1 means the fatigue data calculation module and the part 2 is for the maintenance alternative generation module. During the step 3 to 5, the data are passed with the input/output module. DSS module does not intervene in the admin work, because it is just a parameter setting work for the DSS.

Laptop to run the DSS should be configured with:

- Web access, with access to the PDKM server (located in Kalsrue)
- Java V5 installed

Prerequisite to be able to run the DSS are:

- DSS programmed in java
- DSS GUI programmed
- Interface between DSS and PDKM (mapping task for DSS business logic implementation)
- Interface between DSS GUI and PDKM

3.3.3 Functionalities demonstrated

Full implementation of the DSS has not been achieved due to technical issues for the communication between DSS and PDKM database.
We nevertheless developed and refined the DSS algorithms for predictive maintenance on structures with associated DSS GUI's defined and programmed.

Implementation status of the DSS within the PDKM is as follows:

- A5 DSS Version 2 is connected to the PDKM. But only the input data is read from the PDKM.
- A5 DSS run on a standalone basis
- No write operations are implemented (for entering information using DSS GUI)
- Neither the manual entries nor the DSS results are stored in the PDKM backend (for recording historical information on machine or structural part)

3.3.3.1 DSS program

Two DSS modules were developed on the Java platform:

- Fatigue damage calculation module for each selected part.
- Regression model for the estimation of remaining lifetime of each censoring spot.

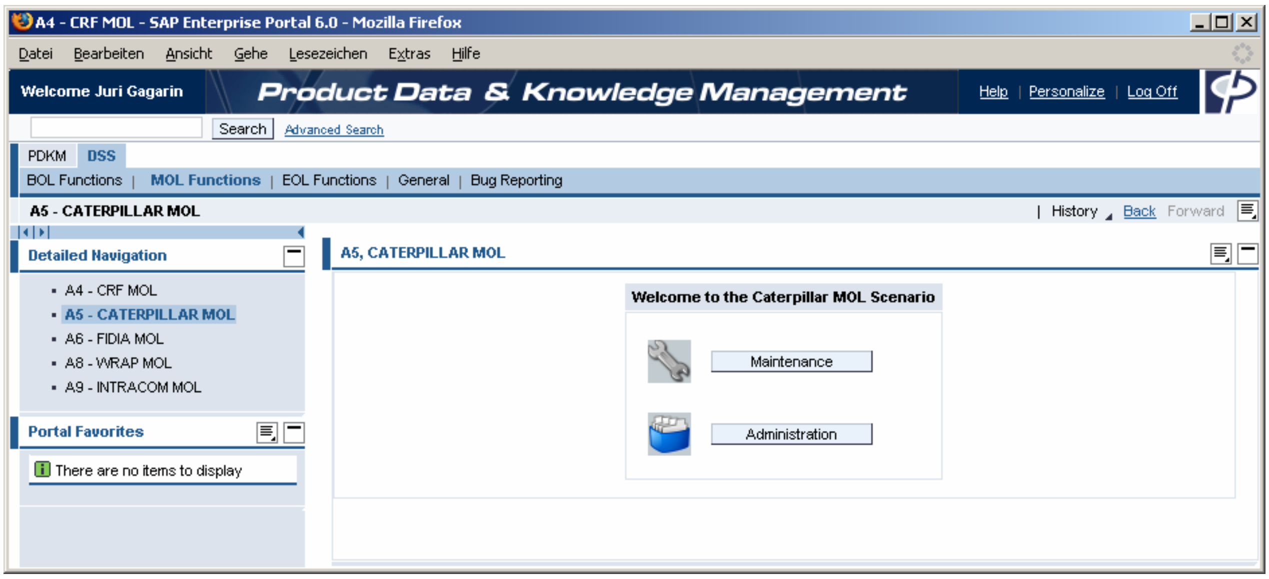
To follow the PROMISE interface standard among core modules and PDKM GUI, it supports 'exec' public function, which takes a defined input class. Of course, the input class should contain all required information for the calculation before passing to the module. The execution of the module returns an output class, which contains calculation result.
Hence, the DSS module gets all required information from input class, which is filled with information from DSS GUI and PKDM.

The developed DSS module was tested to ensure the correctness with an arbitrarily generated sample data before integration. Tests were done under the assumption of the input class contains correct information.
The modules' outputs were compared with the result from manual calculation with Microsoft Excel, and results were exactly matched.

3.3.3.2 DSS GUI

In this section the functionalities of the A5 DSS Scenario will be illustrated. The Caterpillar MOL demonstrator has two main functions: the Maintenance Planning and the Administration of maintenance and design parameters. In the following, the A5 DSS GUI and its navigation is described in detail.

1. The A5 DSS demonstrator starts with the selection of the main function (figure 15).


A4 - CRF MOL - SAP Enterprise Portal 6.0 - Mozilla Firefox

Datei Bearbeiten Ansicht Gehe Lesezeichen Extras Hilfe

Welcome Juri Gagarin — Product Data & Knowledge Management — Help | Personalize | Log Off

Search Advanced Search

PDKM DSS

BOL Functions | MOL Functions | EOL Functions | General | Bug Reporting

A5 - CATERPILLAR MOL — History Back Forward

Detailed Navigation
- A4 - CRF MOL
- A5 - CATERPILLAR MOL
- A6 - FIDIA MOL
- A8 - WRAP MOL
- A9 - INTRACOM MOL

Portal Favorites

There are no items to display

A5, CATERPILLAR MOL

Welcome to the Caterpillar MOL Scenario

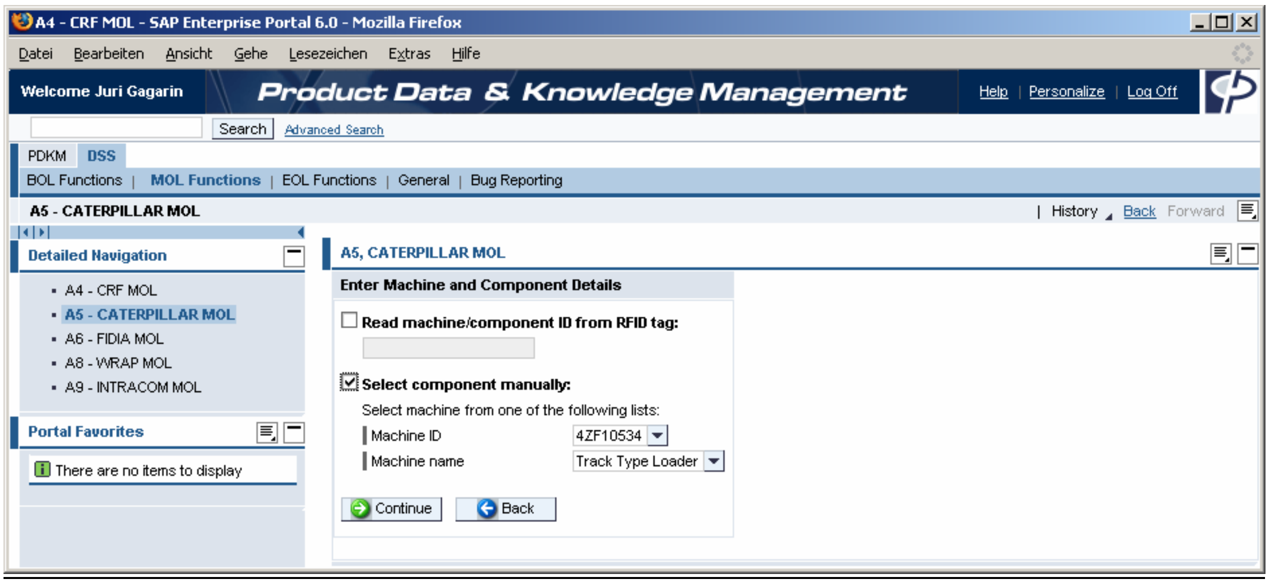
Maintenance

Administration

FIGURE 15: DSS GUI, FUNCTION SELECTION

2. The listed machines are extracted from the PDKM backend with the help of a web service call.

For both functionalities, the user has to determine the respective machine, either by reading the RFID tag or by selecting the machine ID and/or name manually from a drop down list (figure 16).


A4 - CRF MOL - SAP Enterprise Portal 6.0 - Mozilla Firefox

Datei Bearbeiten Ansicht Gehe Lesezeichen Extras Hilfe

Welcome Juri Gagarin — Product Data & Knowledge Management — Help | Personalize | Log Off

Search Advanced Search

PDKM DSS

BOL Functions | MOL Functions | EOL Functions | General | Bug Reporting

A5 - CATERPILLAR MOL — History Back Forward

Detailed Navigation
- A4 - CRF MOL
- A5 - CATERPILLAR MOL
- A6 - FIDIA MOL
- A8 - WRAP MOL
- A9 - INTRACOM MOL

Portal Favorites

There are no items to display

A5, CATERPILLAR MOL

Enter Machine and Component Details

☐ Read machine/component ID from RFID tag:

☑ Select component manually:

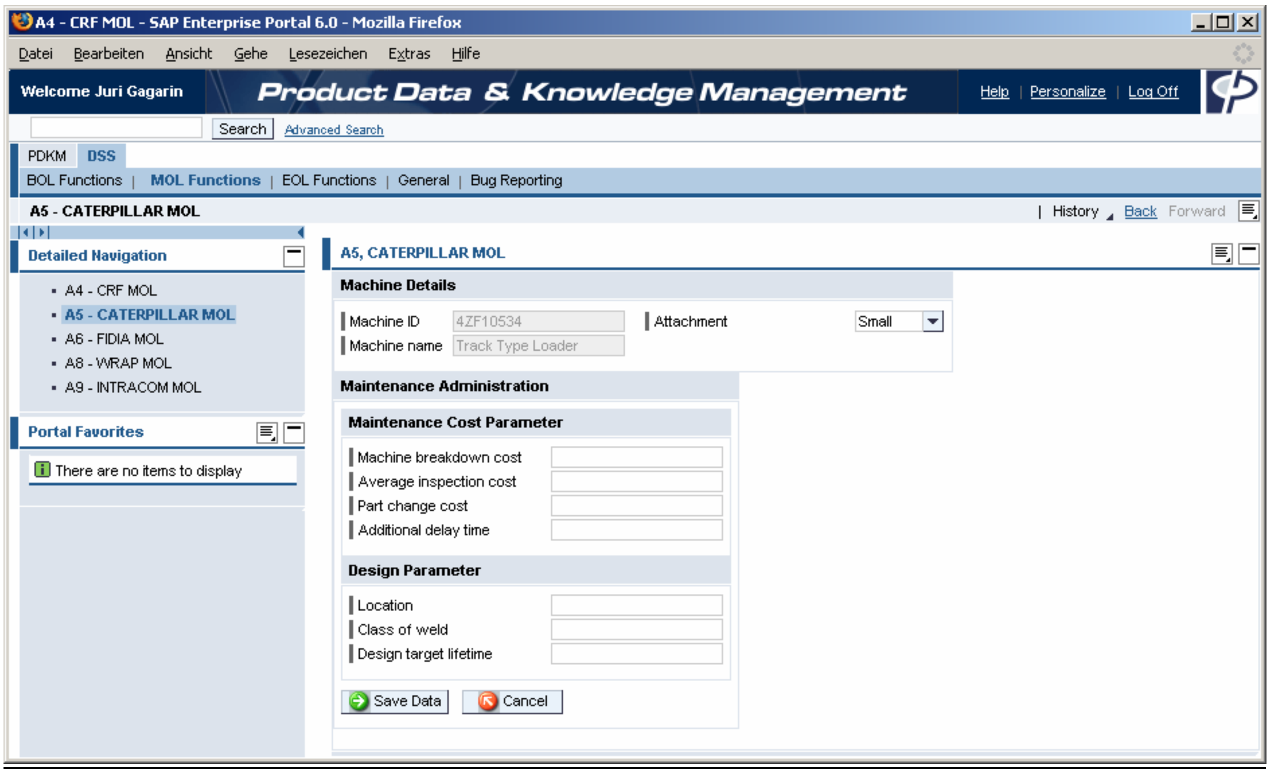
Select machine from one of the following lists:

Machine ID	4ZF10534
Machine name	Track Type Loader

Continue Back

FIGURE 16: DSS GUI, MACHINE SELECTION

3. During the Administration, the maintenance cost and design parameter can be modified for the selected machine (figure 17). The “Save” function is not implemented. This means, the modified parameters are not stored in the PDKM backend, but are valid only for the specific DSS session.

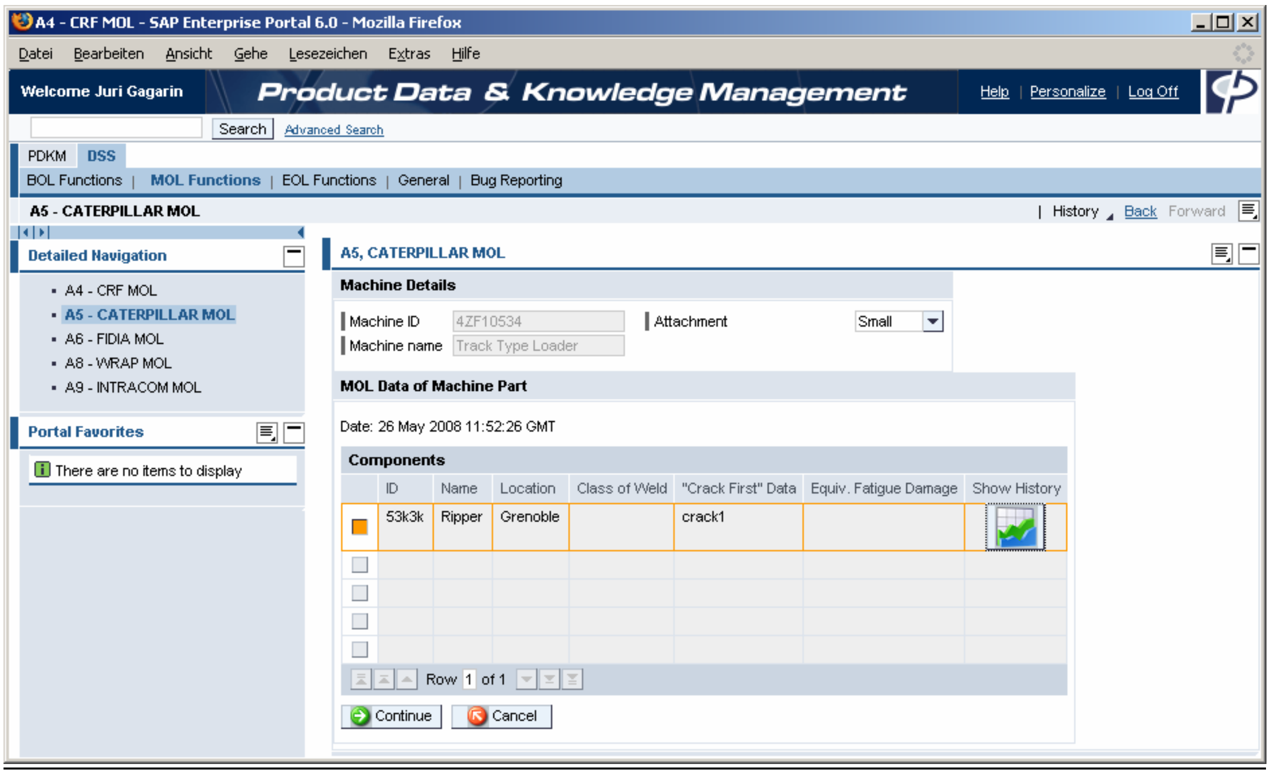

A4 - CRF MOL - SAP Enterprise Portal 6.0 - Mozilla Firefox
Datei Bearbeiten Ansicht Gehe Lesezeichen Extras Hilfe
Welcome Juri Gagarin
Product Data & Knowledge Management
Help | Personalize | Log Off
Search Advanced Search
PDKM DSS
BOL Functions | MOL Functions | EOL Functions | General | Bug Reporting
A5 - CATERPILLAR MOL
History Back Forward
Detailed Navigation
A4 - CRF MOL
A5 - CATERPILLAR MOL
A6 - FIDIA MOL
A8 - WRAP MOL
A9 - INTRACOM MOL
Portal Favorites
There are no items to display
A5, CATERPILLAR MOL
Machine Details
Maintenance Administration
Maintenance Cost Parameter
Design Parameter
Save Data
Cancel

Machine ID	4ZF10534	Attachment	Small
Machine name	Track Type Loader		

Machine breakdown cost	
Average inspection cost	
Part change cost	
Additional delay time	
Location	
Class of weld	
Design target lifetime	

FIGURE 17: DSS GUI, PARAMETERS ENTRIES

4. During the Maintenance Planning, the list of all components of the selected machine is given in a table. Further, a data history is planned. Currently, not all recommended data is inserted in the PDKM backend database. Thus, some of the shown columns are empty and only one component is visible (figure 18).



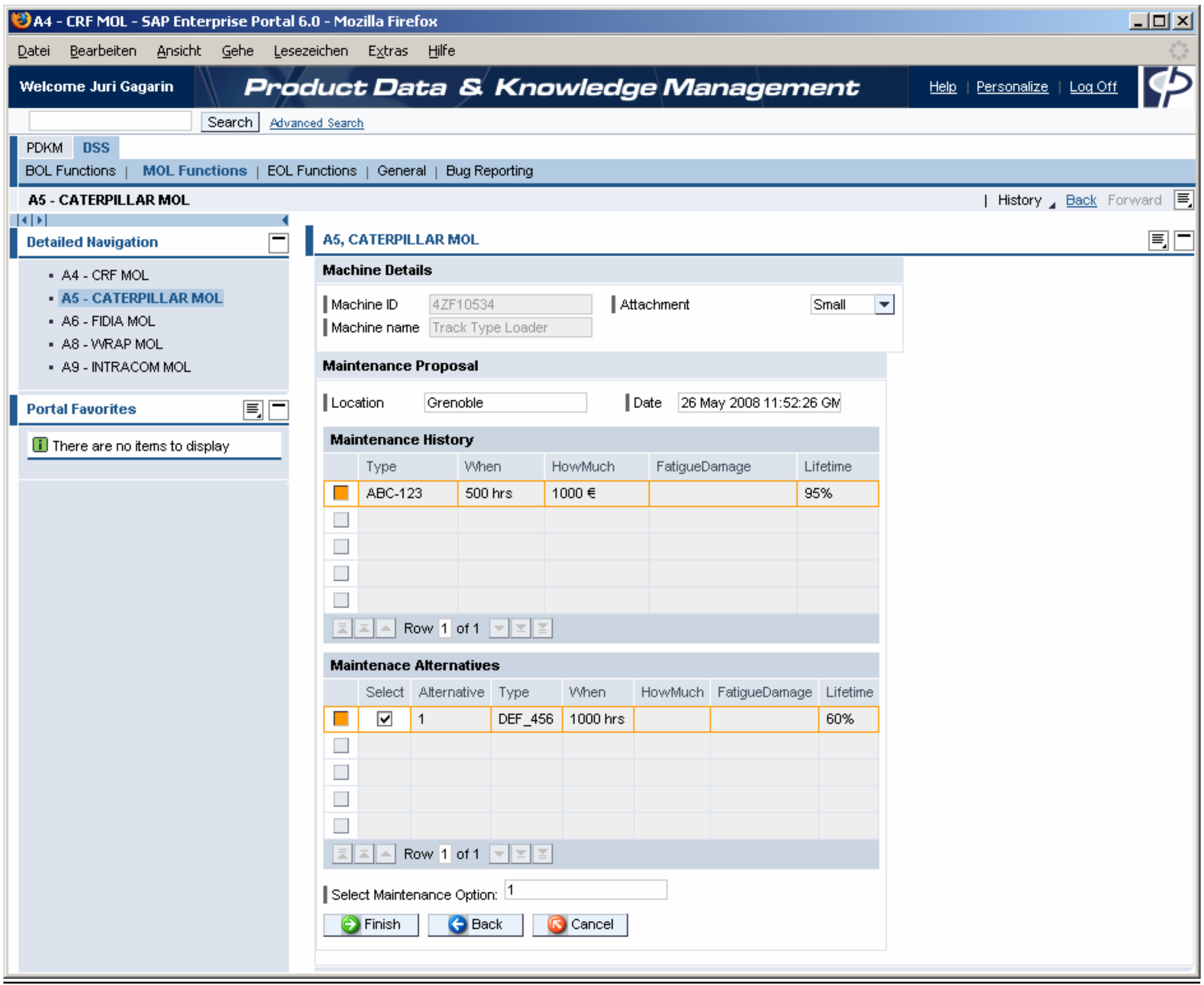
Machine ID	4ZF10534	Attachment	Small
Machine name	Track Type Loader		

Date: 26 May 2008 11:52:26 GMT

	ID	Name	Location	Class of Weld	"Crack First" Data	Equiv. Fatigue Damage	Show History
	53k3k	Ripper	Grenoble		crack1		

FIGURE 18: DSS GUI, MACHINE VIEW

5. After selecting one component in the table and clicking “Continue”, the maintenance history and proposed maintenance actions are shown (figure 19). The DSS algorithm provides multiple alternatives to choose from. In the last step, the user selects the alternative, best fitting to the respective problem and ends the demo by clicking “Finish”.


A4 - CRF MOL - SAP Enterprise Portal 6.0 - Mozilla Firefox
Datei Bearbeiten Ansicht Gehe Lesezeichen Extras Hilfe
Welcome Juri Gagarin
Product Data & Knowledge Management
Help | Personalize | Log Off
Search Advanced Search
PDKM DSS
BOL Functions | MOL Functions | EOL Functions | General | Bug Reporting
History Back Forward
Detailed Navigation
Portal Favorites
Finish Back Cancel

A5 - CATERPILLAR MOL

- A4 - CRF MOL
- A5 - CATERPILLAR MOL
- A6 - FIDIA MOL
- A8 - WRAP MOL
- A9 - INTRACOM MOL

There are no items to display

A5, CATERPILLAR MOL

Machine Details

Machine ID	4ZF10534	Attachment	Small
Machine name	Track Type Loader		

Maintenance Proposal

Location	Grenoble	Date	26 May 2008 11:52:26 GM

Maintenance History

Type	When	HowMuch	FatigueDamage	Lifetime
ABC-123	500 hrs	1000 €		95%

Row 1 of 1

Maintenace Alternatives

Select	Alternative	Type	When	HowMuch	FatigueDamage	Lifetime
✓	1	DEF_456	1000 hrs			60%

Row 1 of 1

Select Maintenance Option: 1

FIGURE 19: DSS GUI, MAINTENANCE ALTERNATIVES

The “Back” button returns to the previous screen by saving the performed selections and modifications temporarily. The “Cancel” button discards the modifications done so far and returns the previous demo step.

The current version of A5 is connected to the DSS business logic and thus to the PDKM backend via a web service interface provided by Cognidata. The data available in the PDKM backend database is read and shown in the GUI. However, no write operations are implemented. Neither the manual entries nor the DSS results are stored in the PDKM backend. To support a comprehensive demo, the missing PDKM data has to be inserted to provide multiple components and various alternative maintenance strategies to choose from.

3.3.3.3 PDKM GUI

The PDKM GUI supports the review of all stored machine parameters. If the status is stored in the PDKM backend, this parameter is visible.

3.3.4 Current and future refinement

Currently, the web services do not return the recommended data items, which have to be shown in the GUI. Either this problem results from missing PDKM data or from an incomplete DSS business logic implementation.

Therefore, for a complete PDKM / DSS implementation, the mapping task between DSS and PDKM needs to be resolved and a large quantity of datasets need to be entered in PDKM backend to finally validate the full DSS integration and its use.

More over, further DSS refinement could be conducted, as well as new DSS modules could be developed for machine fleet management, for BOL perspectives.

4 Conclusion

Steps conducted during the implementation phase of the A5 MOL demonstrator demonstrated the successful use of all PROMISE components and interfaces, except the PDKM / DSS integration.

The customized PEID, the customized Middle ware, the customized PDKM data structure and the DSS program were developed and evaluated with basic tests performed to validate their main functionalities.
Remaining activities would be to further extend the solution in term of workflow and information flow fitting the A5 scenario and lastly, to refine the customized IT system before starting a real world implementation on large-scale CAT products.
For this additional step, critical issues and further technical and business challenges described in deliverable DA5.7 should be considered.

As well as customisation effort on the data structure and customized interfaces, the physical PEID's may be optimised.

From the technical point of view the developed solution of the "crack first" and ECP PEIDs provides suitable means to evaluate the A5 demonstrator. The information relevant for the demonstrator is acquired and is fed into the PROMISE system appropriately. Field tests on excavators (in Austria and UK) have further validated the suitability of the solution. But there is still room for improvement.

The current solution consists of two separated boxes (1 "crack first" box, 1 ECP box) that are connected via a cable. Although the field tests have shown that this set-up can withstand harsh environmental conditions (PEID combination is IP67 protected) it might be favourable to integrate "crack first" and ECP into one box. This will create an even more robust solution and further minimize the size of the box that has to be mounted on the structural parts. As a result, the deployment effort will decrease and mounting of the box in even more spatially constrained locations will be possible. The integration of the two components may be considered in future exploitation activities after the PROMISE project.

5 References

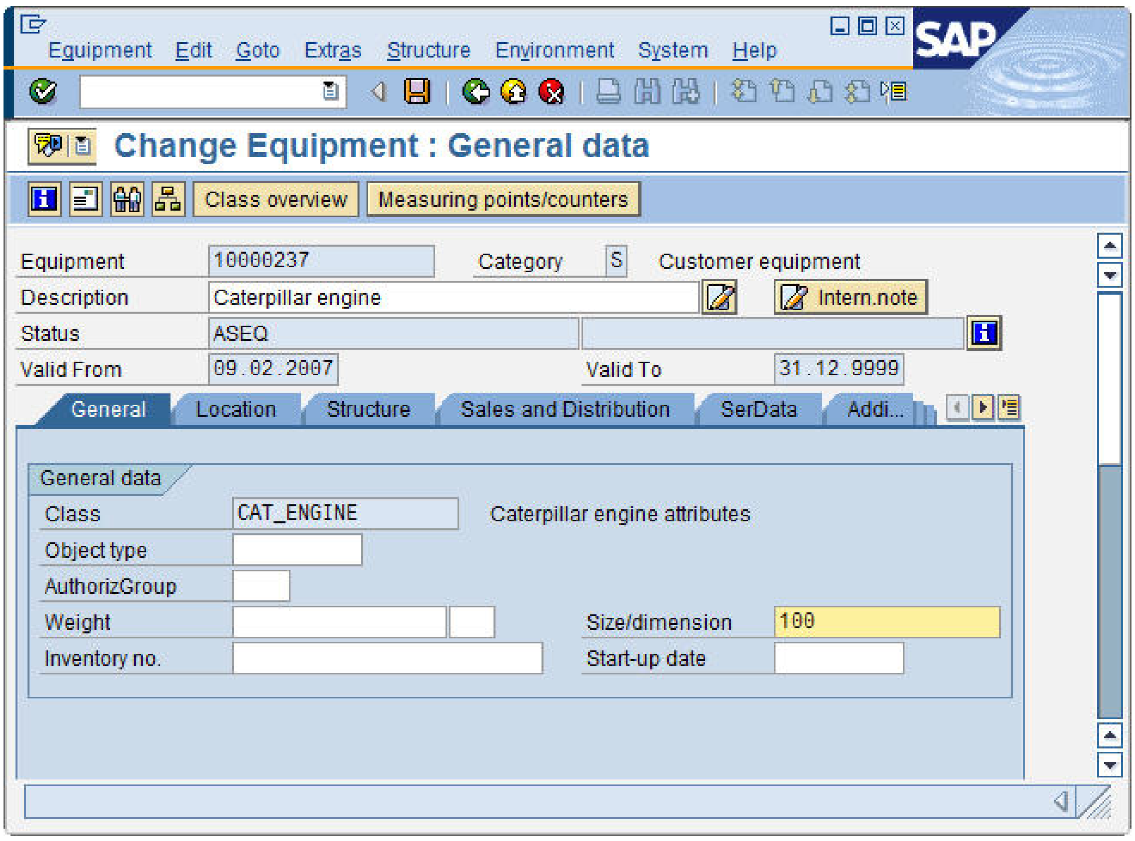
DA5.3 “Design of the A5 Demonstrator on information management for heavy vehicle lifespan estimation”
DA5.4 “Process model workflow description for the demonstrator”
DA5.7 “Critical issues and barriers”
DR8.11 “Refinement and Improvement of the decision support demonstrator (Final version)”

6 Annex

6.1 Alternate initialisation processes of PEID and PDKM

6.1.1 Prerequisite

- PDKM data-structure for new equipment is available
- Initial data that is not acquired from PEID is entered (or imported) using the Backend-System (see Figure 20)
- PEID is started and write-able (see Figure 21)
- DHL is started and PEID recognized (see Figure 22)



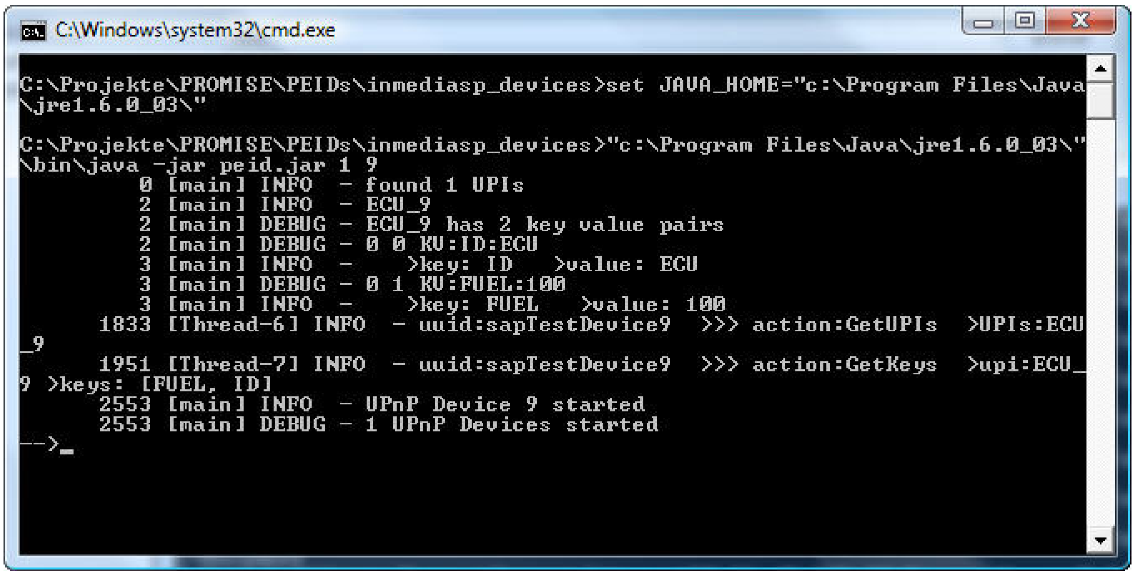
Change Equipment : General data

Equipment	10000237	Category	S Customer equipment
Description	Caterpillar engine		
Status	ASEQ		
Valid From	09.02.2007	Valid To	31.12.9999

General data

Class	CAT_ENGINE	Caterpillar engine attributes	
Object type			
AuthorizGroup			
Weight		Size/dimension	100
Inventory no.		Start-up date	

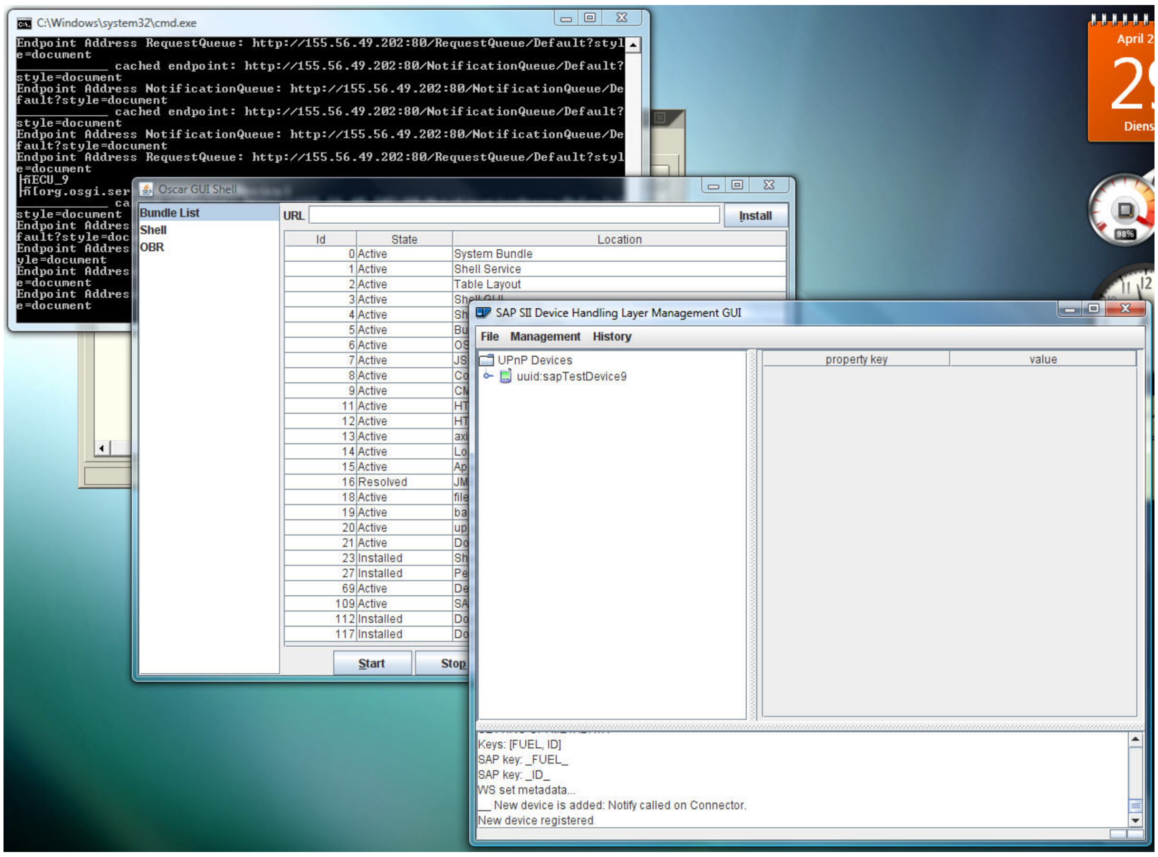
FIGURE 20: EXAMPLE DATA STRUCTURE IN SAP-BACKEND



```
C:\Projekte\PROMISE\PEIDs\inmediasp_devices>set JAVA_HOME="c:\Program Files\Java
\jre1.6.0_03\"

C:\Projekte\PROMISE\PEIDs\inmediasp_devices>"c:\Program Files\Java\jre1.6.0_03\"
\bin\java -jar peid.jar 1 9
         0 [main] INFO  - found 1 UPIs
         2 [main] INFO  - ECU_9
         2 [main] DEBUG - ECU_9 has 2 key value pairs
         2 [main] DEBUG - 0 0 KV:ID:ECU
         3 [main] INFO  -     >key: ID    >value: ECU
         3 [main] DEBUG - 0 1 KV:FUEL:100
         3 [main] INFO  -     >key: FUEL    >value: 100
      1833 [Thread-6] INFO  - uuid:sapTestDevice9  >>> action:GetUPIs   >UPIs:ECU
_9
      1951 [Thread-7] INFO  - uuid:sapTestDevice9  >>> action:GetKeys   >upi:ECU_
9 >keys: [FUEL, ID]
      2553 [main] INFO  - UPnP Device 9 started
      2553 [main] DEBUG - 1 UPnP Devices started
-->_
```

FIGURE 21: SHELL OF EXAMPLE PEID



```
Endpoint Address RequestQueue: http://155.56.49.202:80/RequestQueue/Default?styl
e=document
____________ cached endpoint: http://155.56.49.202:80/NotificationQueue/Default?
style=document
Endpoint Address NotificationQueue: http://155.56.49.202:80/NotificationQueue/De
fault?style=document
____________ cached endpoint: http://155.56.49.202:80/NotificationQueue/Default?
style=document
Endpoint Address NotificationQueue: http://155.56.49.202:80/NotificationQueue/De
fault?style=document
Endpoint Address RequestQueue: http://155.56.49.202:80/RequestQueue/Default?styl
e=document
├ñECU_9
├ñ[org.osgi.ser
____________ ca
style=document
Endpoint Addres
fault?style=doc
Endpoint Addres
yle=document
Endpoint Addres
e=document
Endpoint Addres
e=document
```

Id	State	Location
0	Active	System Bundle
1	Active	Shell Service
2	Active	Table Layout
3	Active	Shell GUI
4	Active	Sh
5	Active	Bu
6	Active	OS
7	Active	JS
8	Active	Co
9	Active	CM
11	Active	HT
12	Active	HT
13	Active	axi
14	Active	Lo
15	Active	Ap
16	Resolved	JM
18	Active	file
19	Active	ba
20	Active	up
21	Active	Do
23	Installed	Sh
27	Installed	Pe
69	Active	De
109	Active	SA
112	Installed	Do
117	Installed	Do

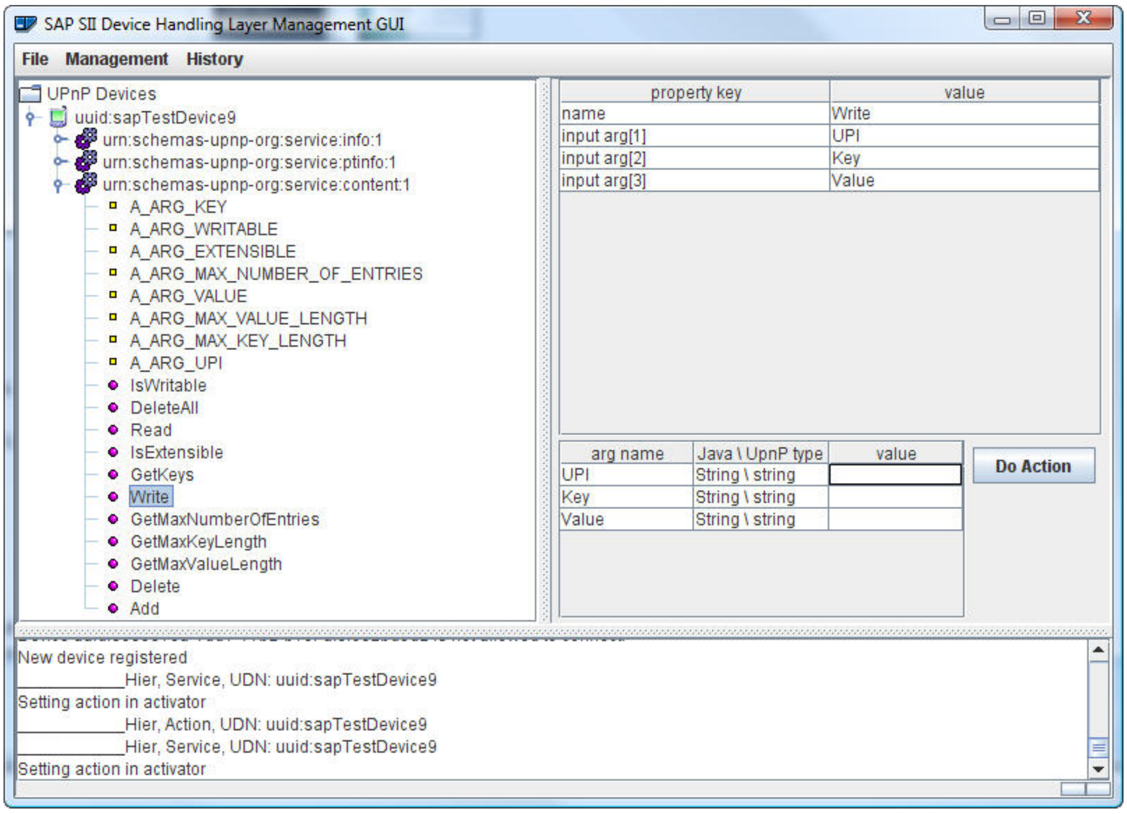
```
Keys: [FUEL, ID]
SAP key: _FUEL_
SAP key: _ID_
WS set metadata...
__ New device is added: Notify called on Connector.
New device registered
```

FIGURE 22: DHL WITH RECOGNIZED PEID

6.1.2 Bottom-Up-Initialization

- Navigate in DHLs service-structure to urn:upnp-org:serviceId:content:1 -> Write (Figure 23)
- Enter UPIs' and PEIDs' ID and the data to be written (e.g., ECU_9, FUEL,100 -> initial fuel-value) (Figure 24)
- Log in to PDKM (Figure 25)

- Navigate to MOL-functions -> Enter Mapping Data -> Search Product Instances -> Navigate to your Equipment and to the info items corresponding measuring point (Figure 26)
- Define a mapping between measuring point and info item by entering UPIs' and InfoItems' ID-> 'create entry' Figure 27)
- Create a subscription by selecting the corresponding MW-node and the desired subscription-interval -> 'subscribe' (Figure 28)
- Result: Data from PEID is transferred from PEID to PDKM automatically (Figure 30)

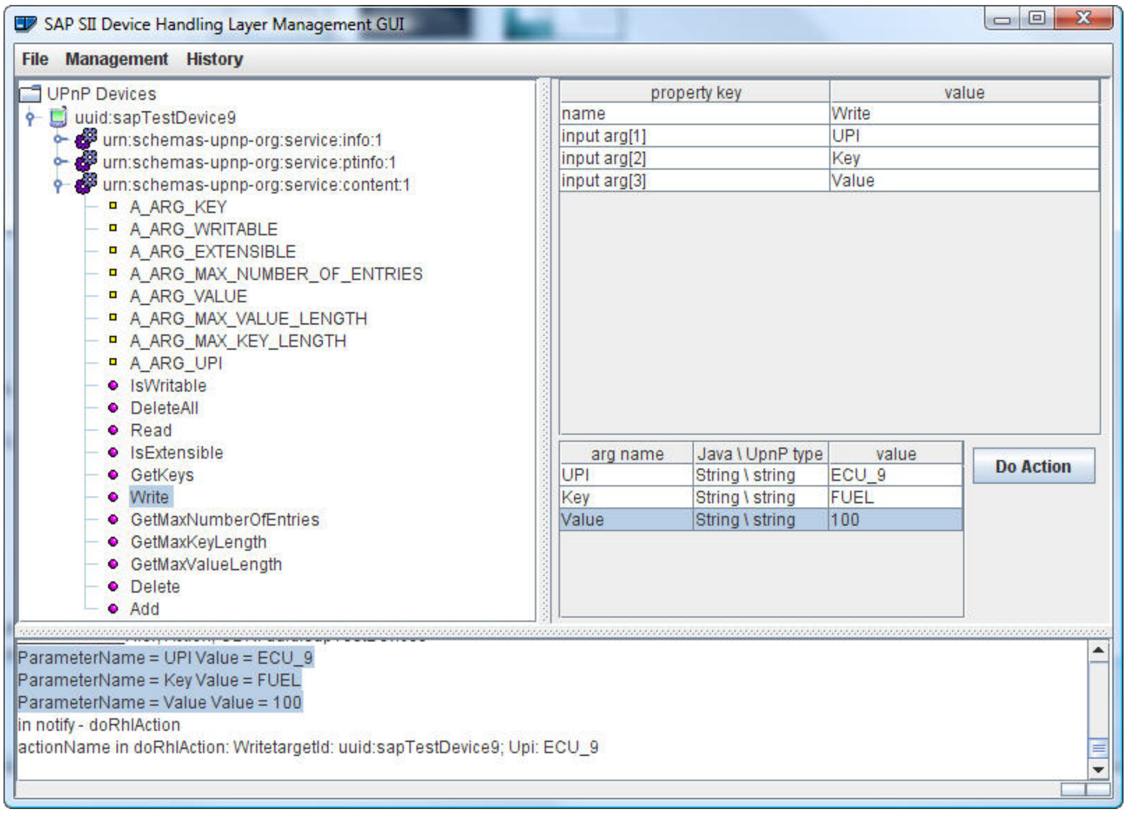


property key	value
name	Write
input arg[1]	UPI
input arg[2]	Key
input arg[3]	Value

arg name	Java \ UpnP type	value
UPI	String \ string	
Key	String \ string	
Value	String \ string	

```
New device registered
___________Hier, Service, UDN: uuid:sapTestDevice9
Setting action in activator
___________Hier, Action, UDN: uuid:sapTestDevice9
___________Hier, Service, UDN: uuid:sapTestDevice9
Setting action in activator
```

FIGURE 23: PEID'S CONTENT-SERVICE (FOR WRITING INITIAL DATA)


SAP SII Device Handling Layer Management GUI

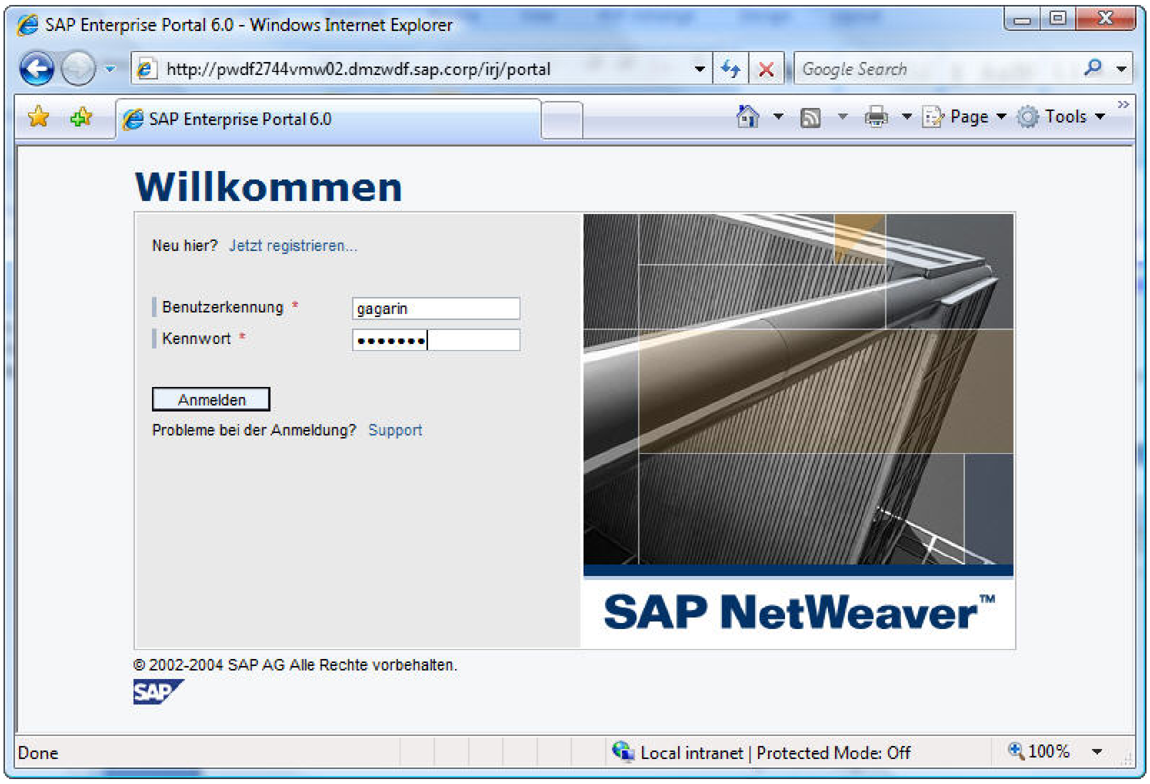
```
UPnP Devices
  uuid:sapTestDevice9
    urn:schemas-upnp-org:service:info:1
    urn:schemas-upnp-org:service:ptinfo:1
    urn:schemas-upnp-org:service:content:1
      A_ARG_KEY
      A_ARG_WRITABLE
      A_ARG_EXTENSIBLE
      A_ARG_MAX_NUMBER_OF_ENTRIES
      A_ARG_VALUE
      A_ARG_MAX_VALUE_LENGTH
      A_ARG_MAX_KEY_LENGTH
      A_ARG_UPI
      IsWritable
      DeleteAll
      Read
      IsExtensible
      GetKeys
      Write
      GetMaxNumberOfEntries
      GetMaxKeyLength
      GetMaxValueLength
      Delete
      Add
```

property key	value
name	Write
input arg[1]	UPI
input arg[2]	Key
input arg[3]	Value

arg name	Java \ UpnP type	value
UPI	String \ string	ECU_9
Key	String \ string	FUEL
Value	String \ string	100

```
ParameterName = UPI Value = ECU_9
ParameterName = Key Value = FUEL
ParameterName = Value Value = 100
in notify - doRhlAction
actionName in doRhlAction: WritetargetId: uuid:sapTestDevice9; Upi: ECU_9
```

FIGURE 24: WRITING DATA TO PEID USING THE DHL


SAP Enterprise Portal 6.0 - Windows Internet Explorer

http://pwdf2744vmw02.dmzwdf.sap.corp/irj/portal

Willkommen

Neu hier? Jetzt registrieren...

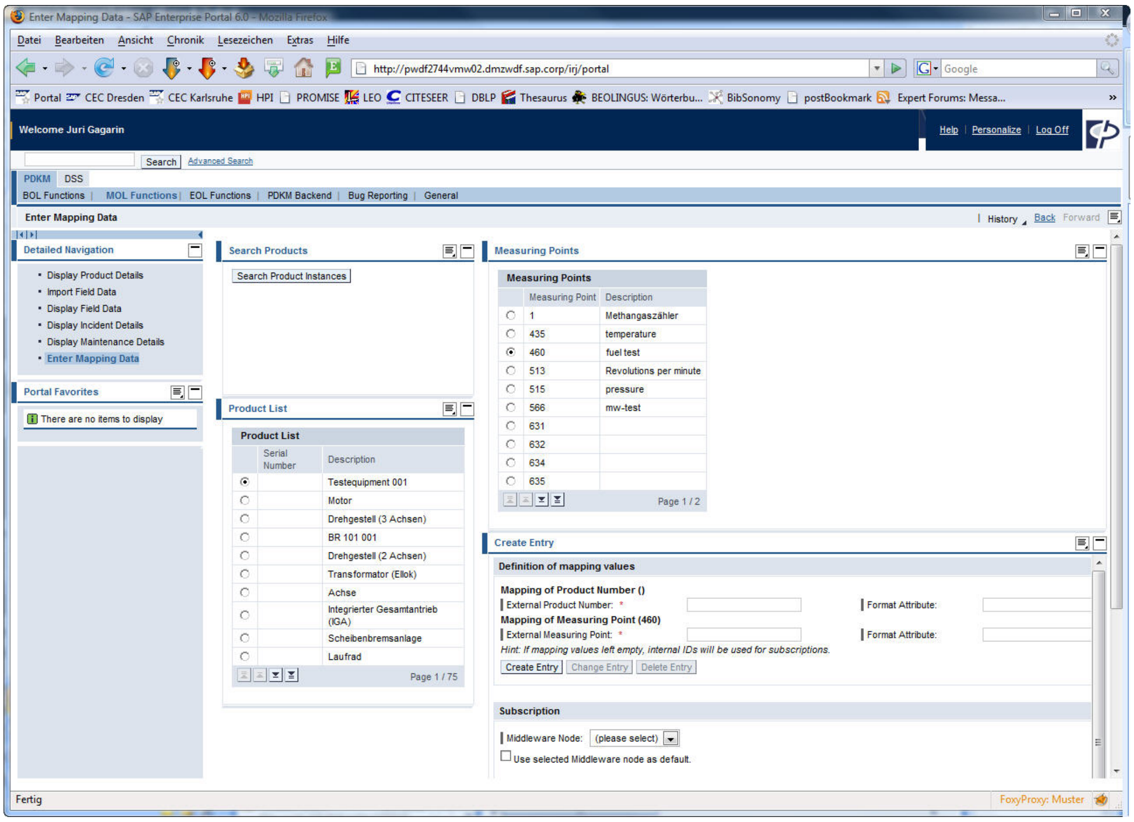
Benutzerkennung *	gagarin
Kennwort *	•••••••

Probleme bei der Anmeldung? Support

© 2002-2004 SAP AG Alle Rechte vorbehalten.

Done | Local intranet | Protected Mode: Off | 100%

FIGURE 25: PDKM LOG-IN


Enter Mapping Data - SAP Enterprise Portal 6.0 - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
http://pwdf2744vmw02.dmzwdf.sap.corp/irj/portal
Google
Portal CEC Dresden CEC Karlsruhe HPI PROMISE LEO CITESEER DBLP Thesaurus BEOLINGUS: Wörterbu... BibSonomy postBookmark Expert Forums: Messa...
Welcome Juri Gagarin
Help | Personalize | Log Off
Search
Advanced Search
PDKM DSS
BOL Functions | MOL Functions | EOL Functions | PDKM Backend | Bug Reporting | General
Enter Mapping Data
History Back Forward
Detailed Navigation
Display Product Details
Import Field Data
Display Field Data
Display Incident Details
Display Maintenance Details
Enter Mapping Data
Portal Favorites
There are no items to display
Search Products
Search Product Instances
Product List
Measuring Points
Create Entry
Fertig
FoxyProxy: Muster

Product List

	Serial Number	Description
●		Testequipment 001
○		Motor
○		Drehgestell (3 Achsen)
○		BR 101 001
○		Drehgestell (2 Achsen)
○		Transformator (Ellok)
○		Achse
○		Integrierter Gesamtantrieb (IGA)
○		Scheibenbremsanlage
○		Laufrad

Page 1 / 75

Measuring Points

	Measuring Point	Description
○	1	Methangaszähler
○	435	temperature
●	460	fuel test
○	513	Revolutions per minute
○	515	pressure
○	566	mw-test
○	631	
○	632	
○	634	
○	635	

Page 1 / 2

Definition of mapping values

Mapping of Product Number ()

External Product Number: * Format Attribute:

Mapping of Measuring Point (460)

External Measuring Point: * Format Attribute:

Hint: If mapping values left empty, internal IDs will be used for subscriptions.

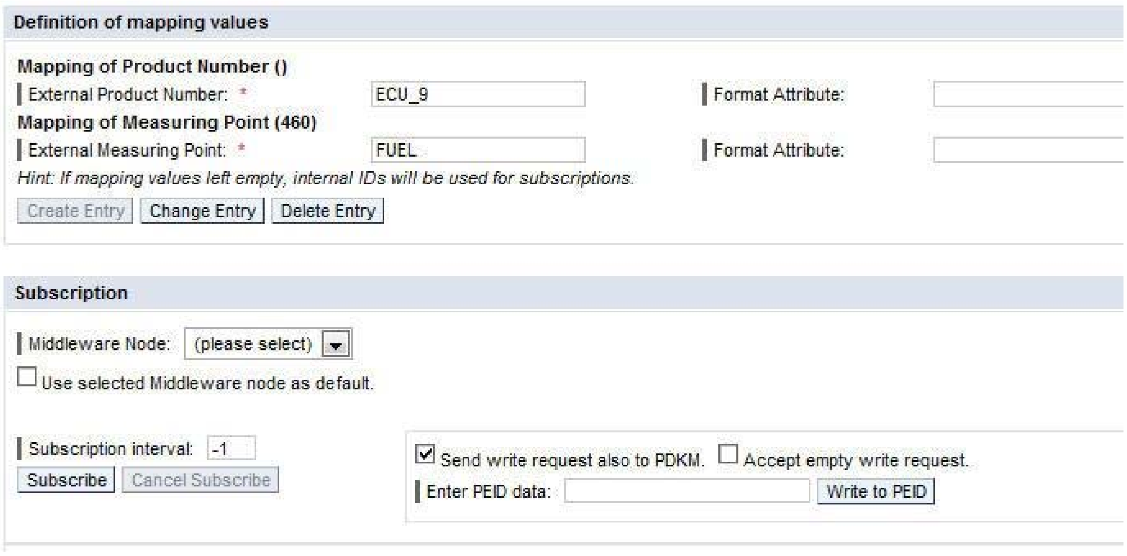
Create Entry | Change Entry | Delete Entry

Subscription

Middleware Node: (please select)

☐ Use selected Middleware node as default.

FIGURE 26: BOL-FUNCTIONS OF PDKM



Definition of mapping values

Mapping of Product Number ()

External Product Number: * ECU_9 Format Attribute:

Mapping of Measuring Point (460)

External Measuring Point: * FUEL Format Attribute:

Hint: If mapping values left empty, internal IDs will be used for subscriptions.

Create Entry | Change Entry | Delete Entry

Subscription

Middleware Node: (please select)

☐ Use selected Middleware node as default.

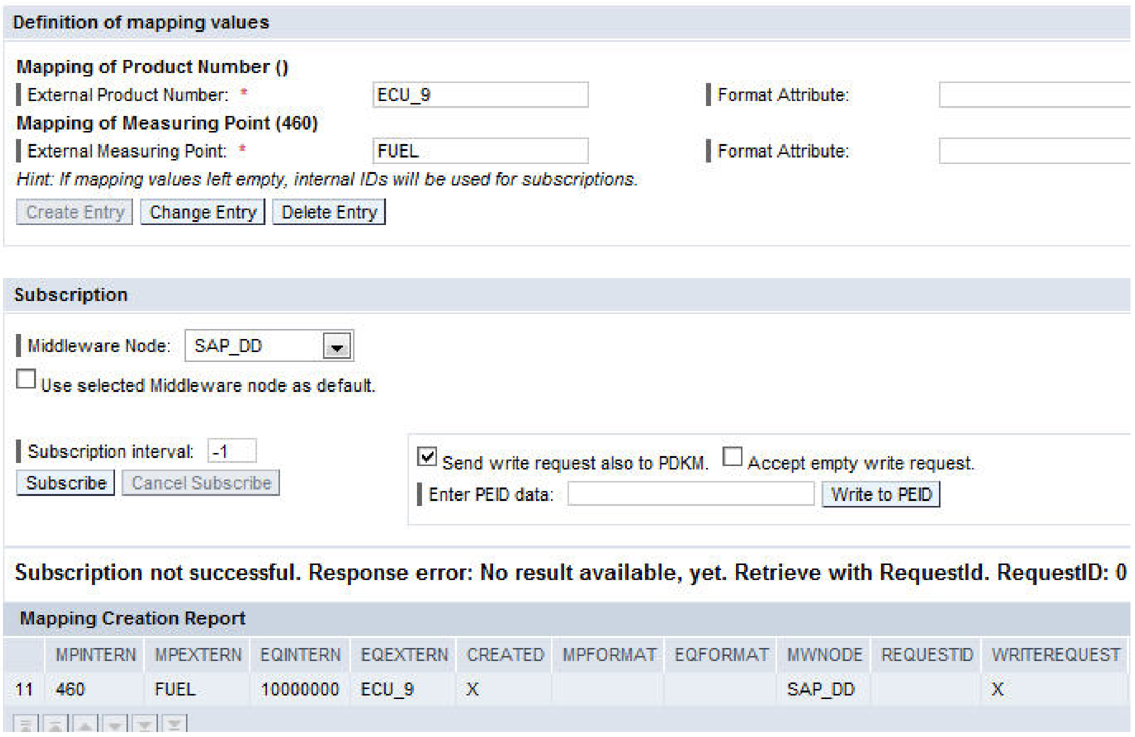
Subscription interval: -1

Subscribe | Cancel Subscribe

☑ Send write request also to PDKM. ☐ Accept empty write request.

Enter PEID data: Write to PEID

FIGURE 27: DEFINITION OF MAPPING PARAMETERS



Definition of mapping values

Mapping of Product Number ()

External Product Number: *	ECU_9	Format Attribute:	

Mapping of Measuring Point (460)

External Measuring Point: *	FUEL	Format Attribute:	

Hint: If mapping values left empty, internal IDs will be used for subscriptions.

Create Entry | Change Entry | Delete Entry

Subscription

Middleware Node: SAP_DD

☐ Use selected Middleware node as default.

Subscription interval: -1

Subscribe | Cancel Subscribe

☑ Send write request also to PDKM. ☐ Accept empty write request.

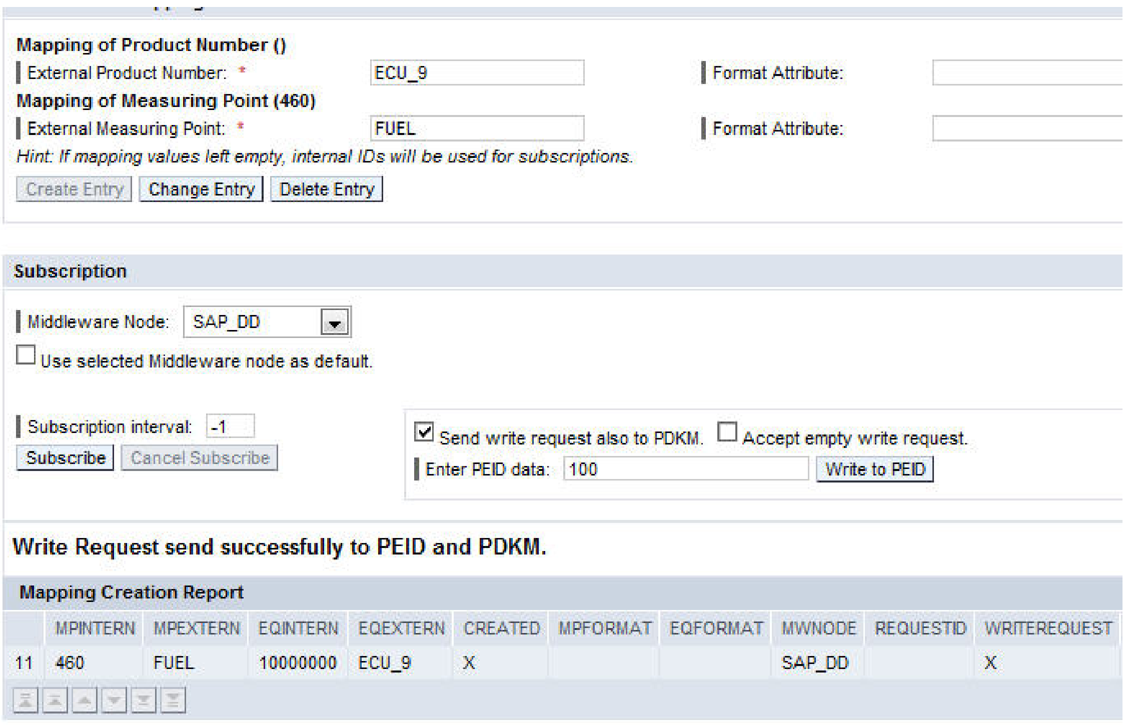
Enter PEID data: | Write to PEID

Subscription not successful. Response error: No result available, yet. Retrieve with RequestId. RequestID: 0

Mapping Creation Report

	MPINTERN	MPEXTERN	EQINTERN	EQEXTERN	CREATED	MPFORMAT	EQFORMAT	MWNODE	REQUESTID	WRITEREQUEST
11	460	FUEL	10000000	ECU_9	X			SAP_DD		X

FIGURE 28: CREATION OF A SUBSCRIPTION



Mapping of Product Number ()

External Product Number: *	ECU_9	Format Attribute:	

Mapping of Measuring Point (460)

External Measuring Point: *	FUEL	Format Attribute:	

Hint: If mapping values left empty, internal IDs will be used for subscriptions.

Create Entry | Change Entry | Delete Entry

Subscription

Middleware Node: SAP_DD

☐ Use selected Middleware node as default.

Subscription interval: -1

Subscribe | Cancel Subscribe

☑ Send write request also to PDKM. ☐ Accept empty write request.

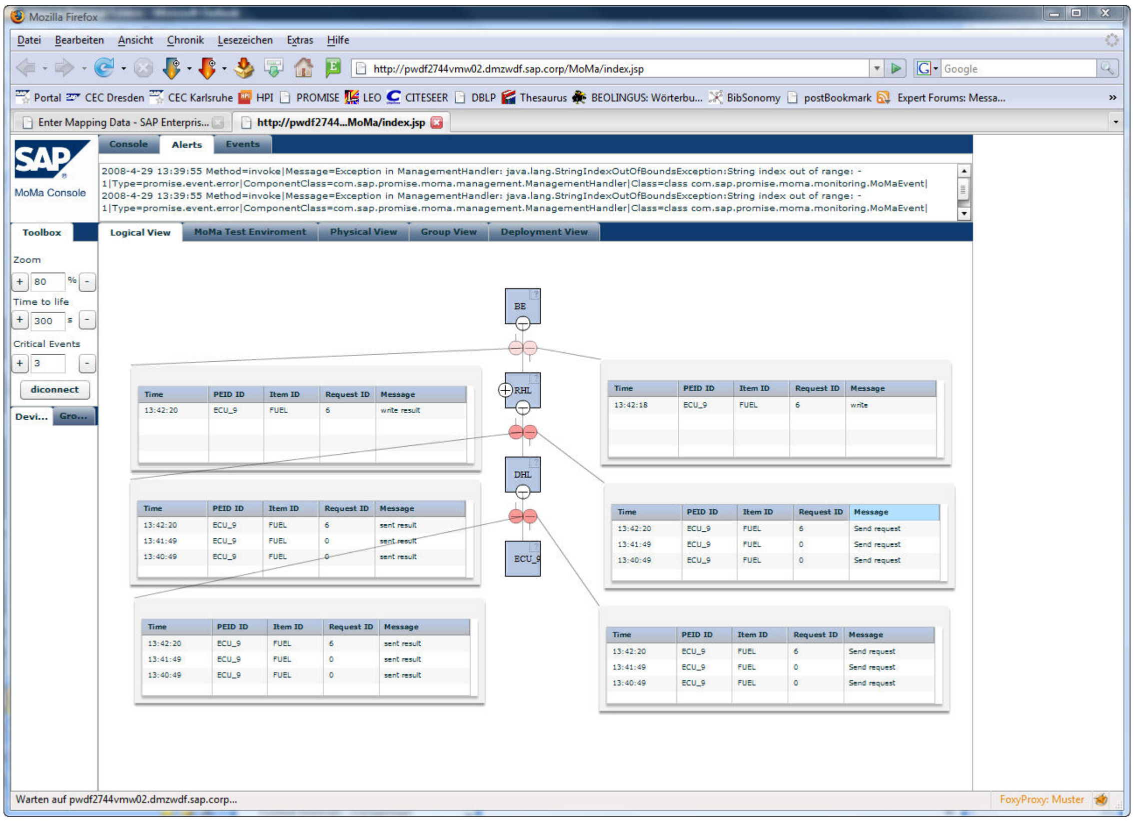
Enter PEID data: 100 | Write to PEID

Write Request send successfully to PEID and PDKM.

Mapping Creation Report

	MPINTERN	MPEXTERN	EQINTERN	EQEXTERN	CREATED	MPFORMAT	EQFORMAT	MWNODE	REQUESTID	WRITEREQUEST
11	460	FUEL	10000000	ECU_9	X			SAP_DD		X

FIGURE 29: WRITE REQUEST FROM PDKM TO BACKEND AND PEID


Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
http://pwdf2744vmw02.dmzwdf.sap.corp/MoMa/index.jsp
Portal CEC Dresden CEC Karlsruhe HPI PROMISE LEO CITESEER DBLP Thesaurus BEOLINGUS: Wörterbu... BibSonomy postBookmark Expert Forums: Messa...
Enter Mapping Data - SAP Enterpris...
http://pwdf2744...MoMa/index.jsp
MoMa Console
Console
Alerts
Events
Toolbox
Zoom
80
Time to life
300
Critical Events
3
diconnect
Devi...
Gro...
Logical View
MoMa Test Enviroment
Physical View
Group View
Deployment View
BE
RHL
DHL
ECU_9
Warten auf pwdf2744vmw02.dmzwdf.sap.corp...
FoxyProxy: Muster

```
2008-4-29 13:39:55 Method=invoke|Message=Exception in ManagementHandler: java.lang.StringIndexOutOfBoundsException:String index out of range: -
1|Type=promise.event.error|ComponentClass=com.sap.promise.moma.management.ManagementHandler|Class=class com.sap.promise.moma.monitoring.MoMaEvent|
2008-4-29 13:39:55 Method=invoke|Message=Exception in ManagementHandler: java.lang.StringIndexOutOfBoundsException:String index out of range: -
1|Type=promise.event.error|ComponentClass=com.sap.promise.moma.management.ManagementHandler|Class=class com.sap.promise.moma.monitoring.MoMaEvent|
```

Time	PEID ID	Item ID	Request ID	Message
13:42:20	ECU_9	FUEL	6	write result

Time	PEID ID	Item ID	Request ID	Message
13:42:18	ECU_9	FUEL	6	write

Time	PEID ID	Item ID	Request ID	Message
13:42:20	ECU_9	FUEL	6	sent result
13:41:49	ECU_9	FUEL	0	sent result
13:40:49	ECU_9	FUEL	0	sent result

Time	PEID ID	Item ID	Request ID	Message
13:42:20	ECU_9	FUEL	6	Send request
13:41:49	ECU_9	FUEL	0	Send request
13:40:49	ECU_9	FUEL	0	Send request

Time	PEID ID	Item ID	Request ID	Message
13:42:20	ECU_9	FUEL	6	sent result
13:41:49	ECU_9	FUEL	0	sent result
13:40:49	ECU_9	FUEL	0	sent result

Time	PEID ID	Item ID	Request ID	Message
13:42:20	ECU_9	FUEL	6	Send request
13:41:49	ECU_9	FUEL	0	Send request
13:40:49	ECU_9	FUEL	0	Send request

FIGURE 30: DATA-FLOW BETWEEN PEID AND PDKM

6.1.3 Top-Down-Initialization

- Log in to PDKM (Figure 25)
- Navigate to MOL-functions -> Enter Mapping Data -> Search Product Instances -> Navigate to your Equipment and to the info items corresponding measuring point (Figure 26)
- Define a mapping between measuring point and info item by entering UPIs' and InfoItems' ID-> 'create entry' (Figure 27)
- Create a subscription by selecting the corresponding MW-node and the desired subscription-interval -> 'subscribe'(Figure 28)
- Enter initial data to be written to info item -> 'Write to PEID' (Figure 29)
- Result: Data from PEID is transferred from PEID to PDKM automatically (Figure 30)

6.2 Manual update of MOL data using the SAP-Backend

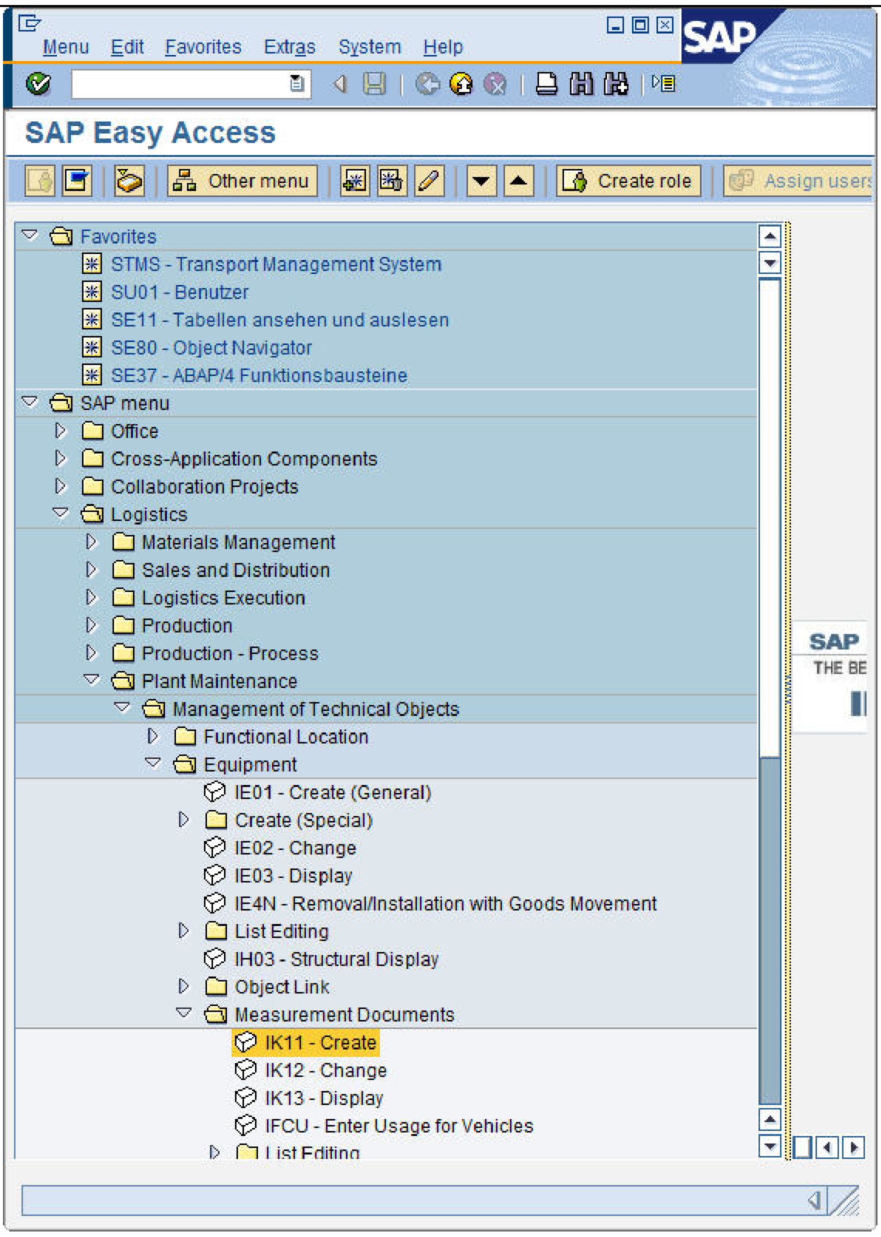
Prerequisite:

- Data structure is initialized
- User is logged in to SAP Backend and navigated to the desired equipment (Figure 31)

Update:

- Navigate to … -> Measuring Documents -> IK11 – Create

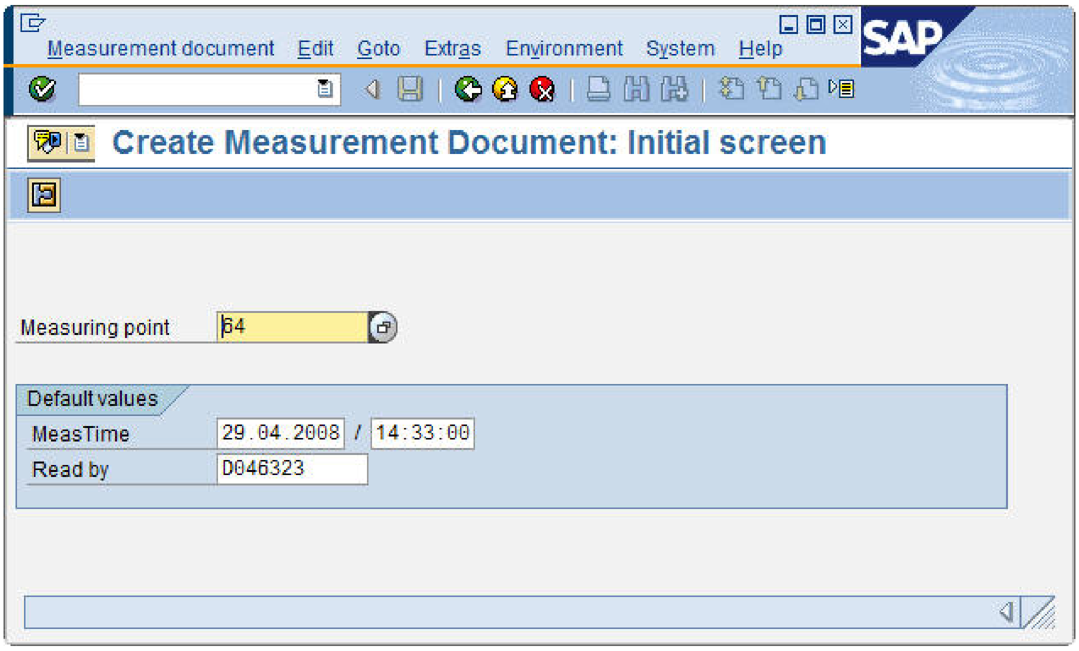
- Enter the ID of the Measuring Point (e.g, retrieved within user-interface shown in Figure 20) an enter it there (Figure 32)
- Enter the current value and save (Figure 33)



SAP Easy Access

- Favorites
  - STMS - Transport Management System
  - SU01 - Benutzer
  - SE11 - Tabellen ansehen und auslesen
  - SE80 - Object Navigator
  - SE37 - ABAP/4 Funktionsbausteine
- SAP menu
  - Office
  - Cross-Application Components
  - Collaboration Projects
  - Logistics
    - Materials Management
    - Sales and Distribution
    - Logistics Execution
    - Production
    - Production - Process
    - Plant Maintenance
      - Management of Technical Objects
        - Functional Location
        - Equipment
          - IE01 - Create (General)
          - Create (Special)
          - IE02 - Change
          - IE03 - Display
          - IE4N - Removal/Installation with Goods Movement
          - List Editing
          - IH03 - Structural Display
          - Object Link
          - Measurement Documents
            - IK11 - Create
            - IK12 - Change
            - IK13 - Display
            - IFCU - Enter Usage for Vehicles
            - List Editing

FIGURE 31: SAP-BACKEND



Create Measurement Document: Initial screen

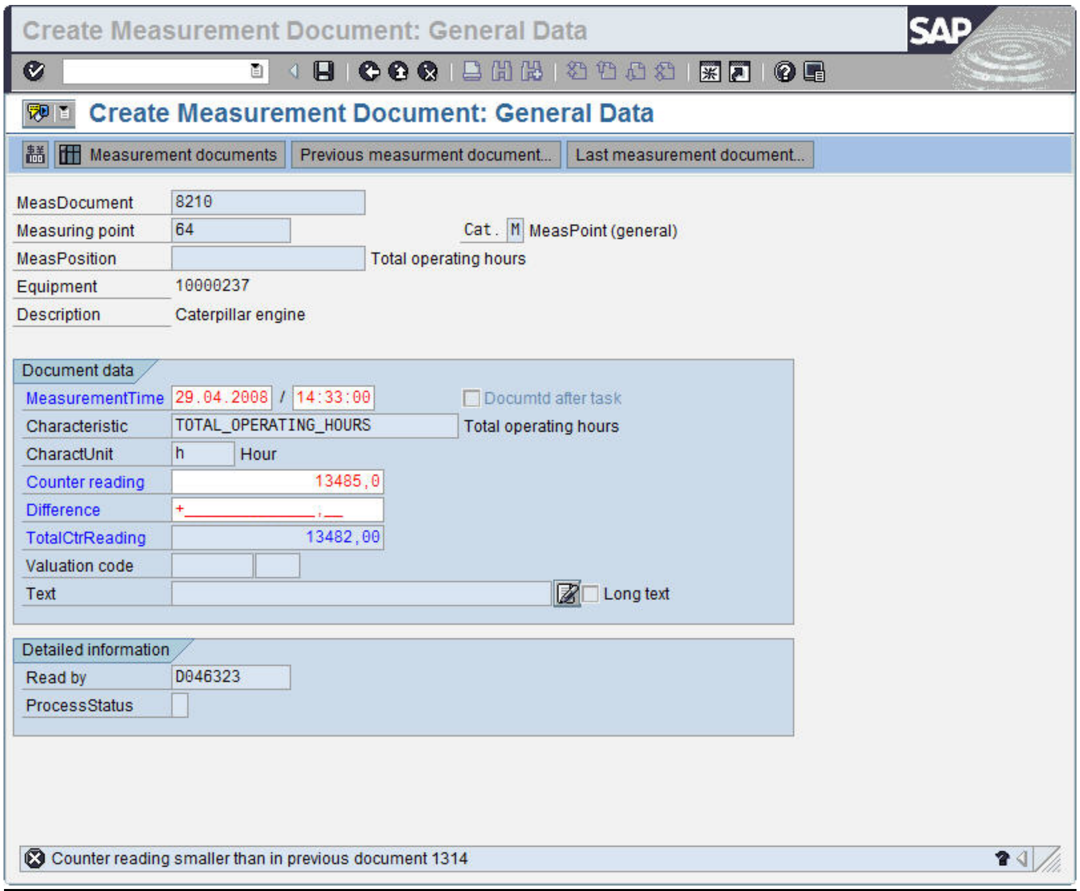
Measuring point 64

Default values

MeasTime 29.04.2008 / 14:33:00

Read by D046323

FIGURE 32: SELECTION OF MEASURING POINT IN SAP-BACKEND


Create Measurement Document: General Data

MeasDocument	8210		
Measuring point	64	Cat. M	MeasPoint (general)
MeasPosition		Total operating hours	
Equipment	10000237		
Description	Caterpillar engine		

Document data

MeasurementTime	29.04.2008 / 14:33:00	Documtd after task
Characteristic	TOTAL_OPERATING_HOURS	Total operating hours
CharactUnit	h	Hour
Counter reading	13485,0	
Difference	+ ,	
TotalCtrReading	13482,00	
Valuation code		
Text		Long text

Detailed information

Read by	D046323
ProcessStatus	

Counter reading smaller than in previous document 1314

FIGURE 33: CREATION OF NEW MEASUREMENT DOCUMENT

6.3 DSS Algorithm

1. Notations

i	index for maintenance options, 1: inspection, 2: repair, 3: change
j	index for class of weld, $j \in \{W, G, F_2, F, E, D\}$
t	index for time
T	additional delayed time
t'	Current time
C_{it}	Maintenance cost for maintenance option *i* at time *t*
C_P	Part change cost
C_R	Risk cost
C_B	Machine breakdown cost
$f_{I_j}^{k_t}(t)$	Ideal fatigue damage at time *t* for the class type *j* of weld in the application profile *k* at time *t*, $t = t',\ t''$
$g_{I_j}^{k_t}(t)$	Inverse function of $f_{I_j}^{k_t}(t)$, $t = t',\ t''$
$f_{C_j}^{k_t}(t)$	Crack fatigue damage data at time *t* for the class type *j* of weld in the application profile *k* at time *t*, $t = t',\ t''$
$g_{C_j}^{k_t}(t)$	Inverse function of $f_{C_j}^{k}(t)$, $t = t',\ t''$
$p_B(x)$	Probability of machine breakdown at damage level *x*
P	Risk factor
T_D	Designed target lifetime (=10,000 hrs)
T_R	Estimated remaining lifetime (based on real trend)
T_I	Theoretical remaining lifetime (based on designed target life time)
λ_j	Inspection threshold value of the class type *j* of weld
E_j	Equivalent fatigue damage of the class type *j* of weld

2. Step 1: Status data collection

Collect status data of a structural part of a heavy construction equipment
Gather machine serial number, location, class of weld, crack propagation data, latest sensoring time, current time, fuel consumption rate, and type of attached bucket size. We assume that the data is gathered per every 960 hours.

3. Step 2: Remaining life time estimation

(1) Identify application profiles.

First, identify application profile based on fuel consumption types (light, standard, and heavy applications) and attachment types (small, standard, and big bucket). In the fuel consumption case, the change of fuel consumption type is automatically identified by the reference data (fuel consumption rate). Attachment types can be identified manually.

Criteria of application profile	Example
Attachment type	Small size bucket (0.8m^3)
	Standard EAME size bucket (1m^3)
	Large size bucket (1.3m^3)
Range of fuel consumption rate	Light application (OPG)
	Standard application (quarry)
	Heavy application (hard soil or rocks)

The following shows the rule to classify application profiles according to fuel consumption rate.

- If Fuel consumption rate < 0.227 Gal/hr, then, fuel consumption type is "light application".
- If 0.227 Gal/hr < Fuel consumption rate < 0.312 Gal/hr, then fuel consumption type is "standard application".
- If Fuel consumption rate > 0.312 Gal/hr, then, fuel consumption type is "heavy application".

(2) In this case example, the sensoring data are based on the F class of weld. Depending on the class of weld, the fatigue damages are different. For each class of weld, we estimate the equivalent fatigue damage function as follows. Note that all values of parameters are empirical, which have been gotten from the company.

$$E_W = 4.69 \cdot f_{C_W}(t)$$
$$E_G = 3.05 \cdot f_{C_G}(t)$$
$$E_{F_2} = 1.4 \cdot f_{C_{F_2}}(t)$$
$$E_F = f_{C_F}(t)$$
$$E_E = 0.52 \cdot f_{C_E}(t)$$
$$E_D = 0.43 \cdot f_{C_D}(t)$$

(3) Calculate the regression model based on gathered field data.
For example, ideal degradation of fatigue damage at *F* class of weld, standard bucket, and standard fuel consumption application can be expressed in the form.

$$f_{I_j}^{k_t}(t) = t/T_D = t/10000$$

And with the following regression model, we can express the real degradation of fatigue damage by estimating a_0, a_1, and a_2.

$$f_{C_j}^{k_t}(t) = a_0 + a_1 \cdot t + a_2 \cdot t^2,$$

For each identified application profile (total 9 profiles), we assume that we have reference models for regression models of ideal degradation of fatigue damage as follows.

- Standard bucket
  - Light application

Machine running hours	Equivalent fatigue damage
0	0
5000 hrs	0.433 (43.3%)
10000 hrs	0.866 (86.6%)

- Standard application

Machine running hours	Equivalent fatigue damage
0	0
5000 hrs	0.5 (50%)
10000 hrs	1 (100%)

- Heavy application

Machine running hours	Equivalent fatigue damage
0	0
5000 hrs	0.833 (83.3%)
10000 hrs	1.667 (166.7%)

- Small bucket
  - Light application

Machine running hours	Equivalent fatigue damage
0	0
5000 hrs	0.353 (35.3%)
10000 hrs	0.705 (70.5%)

  - Standard application

Machine running hours	Equivalent fatigue damage
0	0
5000 hrs	0.407 (40.7%)
10000 hrs	0.815 (81.5%)

  - Heavy application

Machine running hours	Equivalent fatigue damage
0	0
5000 hrs	0.68 (68%)
10000 hrs	1.36 (136%)

- Large bucket
  - Light application

Machine running hours	Equivalent fatigue damage
0	0
5000 hrs	0.66 (66%)
10000 hrs	1.32 (132%)

  - Standard application

Machine running hours	Equivalent fatigue damage
0	0
5000 hrs	0.76 (76%)
10000 hrs	1.53 (153%)

  - Heavy application

Machine running hours	Equivalent fatigue damage
0	0
5000 hrs	1.27 (127%)
10000 hrs	2.545 (254.5%)

(4) Calculate the estimated remaining life time and theoretical remaining life time.

The estimated remaining life time and theoretical remaining life time can be calculated with the formulae below.

$$T_R = \begin{cases} g_{C_j}^{k_{t'}}(100) - g_{C_j}^{k_{t'}}(f_{C_j}^{k_{t'}}(t')) & \text{if } g_{C_j}^{k_{t'}}(100) \geq g_{C_j}^{k_{t'}}(f_{C_j}^{k_{t'}}(t')) \\ 0, \text{ otherwise} \end{cases}$$

$$T_I = \begin{cases} g_{I_j}^{k_{t'}}(100) - g_{C_j}^{k_{t'}}(f_{C_j}^{k_{t'}}(t')) & \text{if } g_{I_j}^{k_{t'}}(100) \geq g_{C_j}^{k_{t'}}(f_{C_j}^{k_{t'}}(t')) \\ 0, \text{ otherwise} \end{cases}$$

4. Step 3: Asset status check

(1) Check inspection option.
If $\lambda_j - 0.1 \leq f_{C_j}^{k_{t'}}(t') \leq \lambda_j$, schedule inspection and go to (2). Inspection should be scheduled at λ_j. In this case example, $\lambda_W = 0.167$, $\lambda_G = 0.25$, $\lambda_{F_2} = 0.583$, and $\lambda_F = 0.833$ are given by the company. Inspection cost (C_{1t}) is also given as 30.

(2) Compare estimated remaining lifetime with theoretical remaining life time.
While theoretical remaining life time (T_I) is based on designed target life time, estimated remaining life time (T_R) is based on real usage operation. If $T_R < T_I - \varepsilon$ where ε is a threshold value, in other words, when the machine is over-used than expected, then go to step 4.

5. Step 4: Maintenance option generation

For each option, alternatives can be generated from time t' to $t'+T_R$ by increasing T for each time. For each option, we can calculate maintenance cost, equivalent fatigue damage, and estimated remaining lifetime.

For t, from $t = t'$ to $t = t' + T_R = g_{C_j}^{k_t}(100)$ by increasing T each time

1. Calculate repair cost and estimated remaining time
   Repair cost depends on damage status of machine. The repair cost can be calculated as follows:

$$C_{2t} = \begin{cases} -42.748 + 297.710 \cdot f_{C_j}^{k_t}(t), & 0 \leq f_{C_j}^{k_t}(t) \leq 0.8 \\ -11400 + 14500 \cdot f_{C_j}^{k_t}(t), & 0.8 < f_{C_j}^{k_t}(t) \leq 1 \end{cases}$$

   Estimated remaining lifetime after repair is $0.8 \cdot T_D = 0.8 \cdot 10000 = 8000$ hrs.

2. Calculate change cost
   The change cost (C_{3t}) includes part change cost and risk cost. If the maintenance is not taken at appropriate time, the possibility of machine failure will increase. Hence, the risk cost for delaying maintenance should be considered in the cost model. The risk cost (C_R) depends on damage status and theoretical remaining life time.

$$C_{3t} = C_P + C_R$$

$$C_R = p_B(f_{C_j}^{k_t}(t)) \cdot C_B \cdot P$$

Where $p_B(f_{C_j}^{k_t}(t)) = c_0 + c_1 \cdot f_{C_j}^{k_t}(t) + c_2 \cdot \left[f_{C_j}^{k_t}(t) \right]^2 = -0.023 \cdot f_{C_j}^{k_t}(t) + 0.123 \cdot \left[f_{C_j}^{k_t}(t) \right]^2$,

$C_B = 10000$, $C_P = 3000$, and $P = \dfrac{\int_{t'}^{t} f_{C_j}^{k_t}(t)dt}{\int_{t'}^{t} f_{I_j}^{k_t}(t)dt}$, $t < T_R$

The estimated remaining lifetime after change is T_D=10000 hrs.

Using the above equations, we can generate some maintenance alternatives