DR9.14: Specification of necessary PLM system enhancements

Written by:
Andreas Edler, InMediasP
Altuğ Metin, InMediasP

DELIVERABLE NO	DR9.14: Specification of necessary PLM system enhancements
DISSEMINATION LEVEL	**CONFIDENTIAL**
DATE	15. Mai 2008
WORK PACKAGE NO	WP R9: Development of PROMISE Information management system
VERSION NO.	0.1
ELECTRONIC FILE CODE	DR9.14-0.1
CONTRACT NO	507100 PROMISE A Project of the 6th Framework Programme Information Society Technologies (IST)
ABSTRACT	In this deliverable a specification is given that describes on a general level the system enhancements which are necessary in order to enable a PLM system to support the PROMISE PDKM approach. Due to the fact that commercially available PLM systems vary sometimes considerably, basic PLM functions may be implemented differently. However, basic PLM functions are taken for granted and thus are not covered by this deliverable.

STATUS OF DELIVERABLE		
ACTION	**BY**	**DATE** (dd.mm.yyyy)
SUBMITTED (author(s))	Andreas Edler, Altuğ Metin	15.05.2008
VU (WP Leader)	Andreas Edler	15.05.2008
APPROVED (QIM)	Dimitris Kiritsis	15.05.2008

Revision History

Date (dd.mm.yyyy)	Version	Author	Comments
25.03.2008	0.1	Andreas Edler	Structure and chapter analysis. First version.
13.05.2008	0.2	Altuğ Metin	Final version

Author(s)' contact information

Name	Organisation	E-mail	Tel	Fax
Andreas Edler	InMediasP	promise@inmediasp.de	+49-3302-559-420	-
Altuğ Metin	InMediasP	promise@inmediasp.de	+49-3302-559-420	-

Table of Contents

List of figures

List of Tables

Abbreviations

Abbreviations used in this document:

BOL	Beginning Of Life phase of the product lifecycle
BOM	Bill Of Materials
DoW	Description of Work document
DSS	Decision Support System
EOL	End Of Life phase of the product lifecycle
MOL	Middle Of Life phase of the product lifecycle
MW	PROMISE Middleware
PDKM	Product Data and Knowledge Management system
PDM	Product Data Management
PLM	Product Lifecycle Management

PEID	Product Embedded Device
UML	Unified Modelling Language
WP	Work Package

1 Introduction

This deliverable specifies the enhancements of a commercially available PDM system that are necessary to support the PROMISE concepts. The enhancements are derived from the functional gap between the Product Data and Knowledge Management (PDKM) system and typical PDM/PLM systems. The PDKM system as such is described and specified in the following deliverables:

- *DR9.1 (Architecture design)*: This deliverable focuses on the development of the overall concept and architecture of the PDKM system for representation and distribution of field and application-specific engineering knowledge. System components, functionalities, and interfaces are to be identified.
- *DR9.2 (Object model design)*: This deliverable specifies an object model describing the attributes of the required objects and their relations in detail. The object model needs to address both semantic issues, i.e. comprehensible representation of domain knowledge for communication between users and applications, as well as technical issues, i.e. the efficient storage of data in physical databases.
- *DR9.3 (Specification of system functions)*: This deliverable describes the functions for creation, update, and change of items and structures as well as for life cycle, change and configuration management and handling and distribution of engineering knowledge in detail. Furthermore, user interfaces, export, import and report functionality will be defined.
- *DR9.4 (Specification of application interface)*: This deliverable specifies an application interface to third-party systems, such as, PEID, PDM, as well as decision making tools.

The following subsections describe necessary enhancements of the data model, the system functions, and interfaces in order to enable a commercially available PDM system to support the data management concepts developed in PROMISE and implemented largely in the PDKM prototype.

2 Object Model

2.1 Product Item Instances

Unlike typical PDM and PLM systems the PROMISE approach to PLM is a "product instance-centric" one. Each instance of a certain product type should be followed all along its life cycle in order to close the desired information loops, thereby creating value. The concept of PEID is capable of enabling the link to all these product items and their related information. A central portion of the object model should thus reflect this approach and properly represent the information on each product at the item level. An object similar to the PHYSICAL_PRODUCT described below has to be implemented in order to make a PLM system PROMISE-capable.

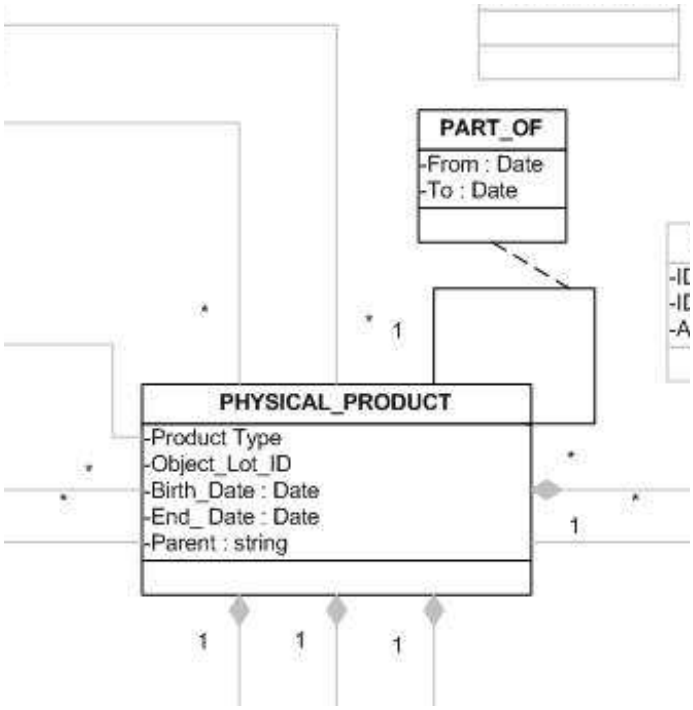

Figure 1: The product structure of physically existing products and the PHYSICAL_PRODUCT class

The PHYSICAL_PRODUCT class (figure 1) is intended to cover this issue by modelling some basic information on each product item, as reflected by its attributes:

- *Product_Type*. This is the name of the product type/model/variant (depending on the meaning given to these words in different industrial contexts) to which the product item belongs. It directly corresponds to the value of the *Product_Type_ID* attribute of one and only one object of the AS_DESIGNED_PRODUCT class. More precisely, it states the name of the object of the AS_DESIGNED_PRODUCT class from which the general attributes of the product as an instance of a product type are derived, as indicated in the diagram by the classification linking the AS_DESIGNED_PRODUCT class and the PHYSICAL_PRODUCT class.
- *Object_Lot_ID*. This shows the ID of the production lot to which the product item belongs, in case it is important to store this information in the specific application considered. For instance, the typical example of situation in which this information turns out to be important can be that of a typical recall campaign a company must carry out when a big defect has been detected in one or more products of the same lot, such as some cases happened in the last years in the automotive industry.
- *Birth_Date*. This models the date at which the existence of the product item actually starts. This can correspond to different time instants in different applications, e.g. the moment at which the product item exits the last station of the production line, or the moment when the product item enters the first stage of that line, etc. The first can be the case of a company which is not interested in following specifically each product item in the different phases of the production cycle, but e.g. is interested in tracking and tracing the product inside the warehouse or/and across its supply chain. The second can be instead the case of a company which gives importance to the tracking and tracing of the product all along the production cycle, such as those companies which daily apply RFID technologies for this reason at the shop floor level.
- *End_Date*. This represents the date at which the product item reaches its end of life. The cardinality, differently from the previous attributes which have cardinality of one and only

one, is in this case zero to one. This reflects the fact that this attribute is not instantiated until the product item reaches its end of life.

- *Parent*. This attribute gives the name, i.e. the ID of the father node in the tree corresponding to the structure of the product as a whole. Each node of this tree corresponds to one and only one product item. So it is important to notice that at this level only the product structure for the phases BOL supply, MOL, and EOL, i.e. the structure of physically existing products is actually addressed (see in the following pages for an explanation of these terms; refer also to DR9.1 [2]), while the BOL as-designed structure of the product (see again DR9.1 document) is treated in an analogue way by the AS_DESIGNED_PRODUCT class. The cardinality of this attribute is zero to one. It is zero when the product structure is the simplest one, i.e. the product is considered as a one-piece product, or if the according product is at the top of the assembly-part hierarchy. It is one when on the contrary the product is modelled as a part of an assembly.

In order to support all relevant PROMISE scenarios it is inevitable to track the history concerning the physical components/subassemblies belonging to a single physical product entity. At the beginning of a product's life, the *From* attribute of the PART_OF association class related to the link between each of these components/subassemblies and the whole product is set to the current date. Then, if at a certain point in the products life, a component has to be replaced by another one of the same type, e.g. because its residual life has expired, its *To* attribute is set to the new current date and, as soon as the new component is attached to the complex product, a new object of the PART_OF class is created and the *From* attribute related to the new component is set to the proper value. This enables the PDKM system to keep an updated list of components/ subassemblies of each product item, as well as a list of old components/subassemblies, together with the related periods of relative belonging.

- *From* (Date): This attribute keeps record of the date and time when the component/subassembly is attached to the product. The cardinality of 1 says that there can only be one value for this attribute.
- *To* (Date): This attribute keeps record of the date and time when the component/subassembly is detached from the product. The cardinality of 0...1 is intended in the sense that this attribute is set to no value at all for the whole time span during which the component/subassembly is attached to the product, and then is set to its value when it is detached from it.

2.2 Life Cycle Phases and Related Field Data

The basis for all PROMISE-related concepts are information of all product life cycle phases. In the system object model specification (DR9.2) three different classes, namely the PRODUCT_BOL_SUPPLY class, the PRODUCT_MOL class and the PRODUCT_EOL class, have been defined for this purpose and are essential in a PROMISE-enabled PLM system.

The PRODUCT_BOL_SUPPLY class (figure 2) contains the pieces of information which are specific to the SUPPLY phase of the product and which cannot be described by using the LIFE_CYCLE_PHASE class or the EVENT, RESOURCE or ACTIVITY classes (figure 3). The life cycle phases covered by PRODUCT_BOL_SUPPLY refer to product item instances (physical products), while the design phase is related to the product type concept which is addressed by the AS_DESIGNED_PRODUCT class in DR9.2. This object is implemented in all PDM and PLM systems.

These two categories essentially differ in the sense that field data gathered during the product use and service must be related to product item instances. Field data can be retrieved in the sense of

measurements, events that the physically existing product is involved in, and similar, which is a core input in all PROMISE application scenarios.

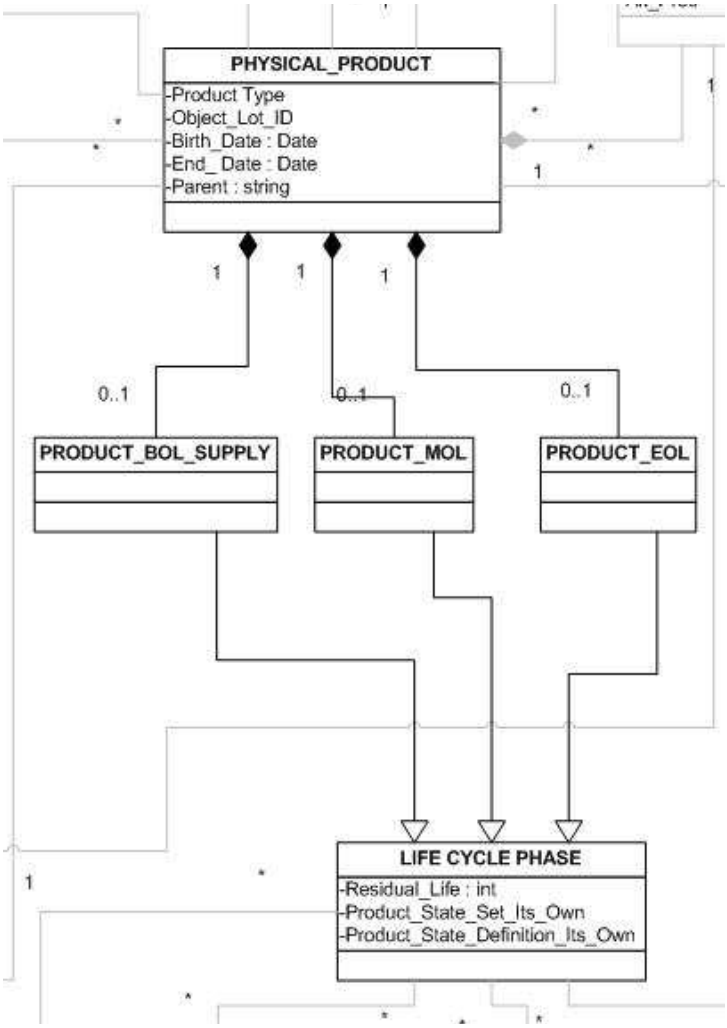

Figure 2: Product life cycle information: different classes for different purposes

In order to collect all this data that is created during the life of a physically existing product the LIFE_CYCLE_PHASE class with its sub-classes has to be implemented in a similar way in order to make PDM/PLM systems PROMISE-compliant.

The PRODUCT_MOL class and the PRODUCT_EOL class play the same role as the PRODUCT_BOL_SUPPLY class, but are related to respectively the MOL and to the EOL phase of a product's life.

The FIELD_DATA class (figure 3) is another crucial class in the semantic model, in the sense that it enables the overall PROMISE approach to Product Life Cycle Management by collecting data from the field, thus also enabling the improvement of product performance and in general the creation of economic value from PLM activities.
Field data can be of different types (VALID_FD_TYPE class), and is collected by means of sources like e.g. sensors (FD_SOURCE class). It might be organized in documents (DOCUMENT class) with attached physical files (FILE class). The associations among these classes show the most important existing links; the cardinalities are all zero to many, to take into account the most general cases.

The detailed description of all attributes mentioned in the figure can be found in the deliverable DR9.2 as well as examples for the instantiation of these semantic object models in order to represent data from real applications.

In order to support PROMISE concepts the field data class with its main attributes has to be implemented in PDM/PLM systems. However, the FIELD_DATA class is a very generic class, which can be utilized in different applications, for different purposes and different meanings. The relevant attributes strongly depend on the actual application and may differ notable.

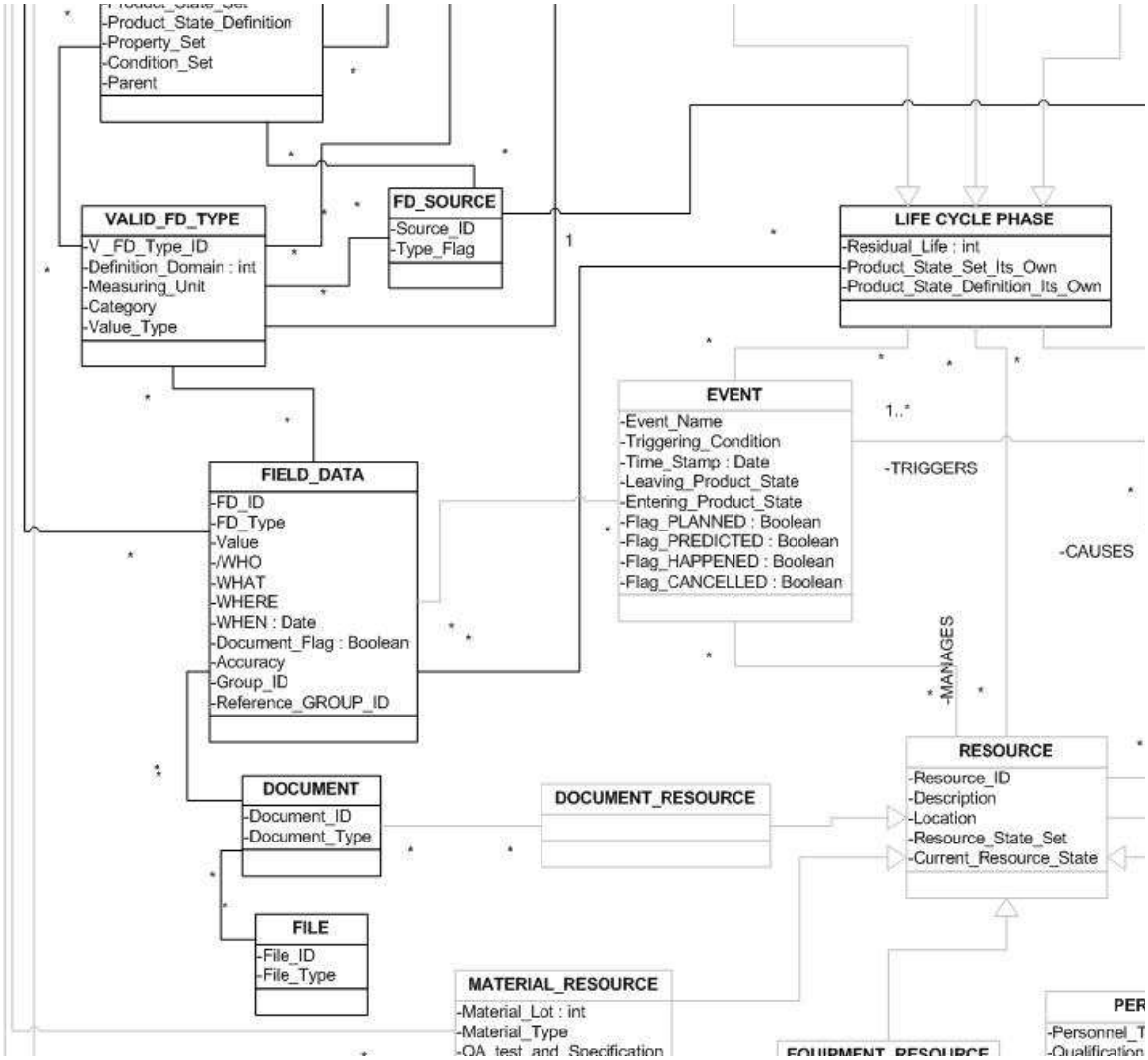

Figure 3: The FIELD_DATA class and its relationships with other model components

3 PROMISE PLM Functions

In this chapter the functions are described which are either not part of typical PDM/PLM systems or which are part of typical PDM/PLM systems but that are usually not implemented in an extent that is needed in order to support the PROMISE concepts. Standard PLM functions are not described in this deliverable. All functions are described in more detail in DR9.3.

3.1 Web Portal

Although the web portal is rather a technical component than a PLM function it is mentioned here since the different participants in different places all over the world have to work simultaneously with a PDKM system over the internet via the web interface, even with mobile clients. Any web browser can act as a client and there is no installation on client computers needed. All user functionality described in the subsequent sections should be offered in a web portal of the PLM.

To protect the communication between servers and clients technically the communication should be handled via the SSL (Secure Sockets Layer) security protocol to protect the system from misuse.

3.2 Product and Product Structure Management

Product and product structure management in PDKM starts with the creation of an as-designed product which contains design information about a *specific product type* and an *as-designed product structure* (Bill of Material, BoM) which covers the components and parts structure of that *Material*.

After the *Material* and/or the *BoM* is created, an *individual product* (SAP: Equipment with Serial Number, shortly Equipment; DR9.2: PHYSICAL_PRODUCT) and its respective *as-build/as-used* (in the following shortly "as-used") *product structure* (SAP: Installed Base) can be generated on this basis. An *Installed Base* reflects the actual bill of material (in the common sense, not the SAP-term) of the *Equipment*. There is maximally one *Installed Base* for each *Equipment* in the field but an *Equipment* does not need to have an associated *Installed Base* if it is not required to track its structure. Parts of an *Equipment* are itself represented by *Equipments* or a *Material* with associated amount. The latter might be the preferred solution if e.g. 100 screws of a certain type shall be modelled as parts of an *Equipment*. An *Equipment* can be identified by its unique ID or its serial number (in the common sense, not the SAP-term) and the ID of the respective *Material*.

In SAP ECC *Equipment* and *Serial Number* are two independent objects. Especially *Serial Number* is not only a number but a complex object representing a physical product with a serial number (in the common sense). For a *Serial Number* there can be activated the *Equipment*-view which allows additional operations on that *Serial Number*. By activating this view the *Serial Number* is uniquely associated with an *Equipment*. In the PDKM-context *Equipment* and *Serial Number* are usually used together therefore in this text the term *"Equipment"* will be used instead of *"Equipment* with *Serial Number"*. If *"Equipment"* is meant without *Serial Number* this will be indicated explicitly or is clear from the context. When using the PDKM there might be nevertheless the need to create *Equipments* without *Serial Numbers* e.g. when modelling wheels of a truck. This is possible and does not limit the functionality of the PDKM.

For each *Material* and *Equipment* a set of metadata is stored in the form of attributes to these objects. A respective *subset of this metadata* (SAP: Classification; DR9.2: PROPERTY) is customisable. Besides this option metadata can also be attached to *Materials* and *Equipments* in the form of *documents* (SAP: Document Info Record; DR9.2: DOCUMENT_RESOURCE with associated DOCUMENT; for Document Management functions see section 3.7). PDKM enables users browsing through product structures or search for products using metadata and thereby generating different kinds of reports like where-used-reports[1].

Besides the *Materials* and *Equipments* with its corresponding structures there are *product templates* (PDKM: Equipment Template) with a *corresponding structure* (PDKM: Installed Base

[1] "Report" in this context means the display of a list and summarising data in the user interface.

Template). From a system point of view an *Equipment Template* is an *Equipment* eventually associated with a respective *Installed Base* that can be semantically recognised as template. This means especially that all operations that are possible for *Equipments*/*Installed Bases* are also possible for *Equipment*/*Installed Base Templates*. This fact is not mentioned explicitly for all functions described in this document but only for those where it has special importance. *Equipments* can be instantiated either directly from a *Material* or from an *Equipment Template* while it is possible to trace back from the *Equipment* to its corresponding *Material* and eventually the respective *Equipment Template*. This intermediate layer *Equipment Template* was introduced since it is assumed that it might be more convenient and clearly structured for the user not to have to instantiate for every variant of a product a new *Material* but to instantiate an associated *Equipment Template*. An example of its application is given in section 3.4.

3.3 Configuration Management

The PDKM system is capable of tracking the history of *Equipments*, i.e. it is traceable in which *Installed Bases* an *Equipment* has been installed before. This information also allows the reconstruction of the *Installed Base* for a certain point of time in the product lifecycle.

3.4 Field Data Functions

There has to be distinguished between

- field data in the form of single values like the current mileage of a locomotive and
- field data in the form of documents like a specific report.

With respect to the first the PDKM system offers functionality to manage this kind of data that is gathered during the usage of *Equipments* by using *Measuring Documents*[2] (SAP-term; DR9.2: FIELD_DATA). In order to manage different *types of Measuring Documents* (SAP: Characteristic; DR9.2: VALID_FD_TYPE) and thereby both allowing a classification of *Measuring Documents* and controlling the application of *Measuring Documents* a respective *Characteristic* has to be defined prior to attaching *Measuring Documents* to *Equipments*. *Characteristics* can then be assigned to *Equipments* or *Equipment Templates* by defining for those a respective *Measurement Point* (SAP-term; DR9.2: FD_SOURCE) using this *Characteristic*. A *Measurement Point* corresponds to a sensor/an info item of a PEID (compare DR6.3 for the term info item). When *Equipments* are instantiated from an *Equipment Template*, *Measurement Points* associated to the *Equipment Template* will be made available for the *Equipment* respectively copies of these *Measurement Points*. Finally, *Measuring Documents* containing a timestamp can be stored for the *Equipment*.

In many cases *Measurement Points* respectively the corresponding *Measuring Documents* are semantically bundled to a record of measurements. This information can also be stored since PDKM offers functionality to model groups of *Measurement Points* respectively their associated *Measuring Documents*. A group of *Measurement Points* might e.g. correspond to all sensors/info items of 1 PEID or a subset of sensors of 1 *Equipment*. The values of *Measuring Documents* of different *Measurement Points* for a given time period can be shown graphically for an *Equipment*.

With respect to field data documents it is possible to associate *Document Info Records* (see section 3.7) to *Equipments* that can semantically be recognized as field data.
Both *Measuring Documents* and field data documents can also be associated to *Equipment Templates* as foreseen in DR9.2.

[2] A *Measuring Document* has nothing to do with a document in the common sense.

3.5 Field Data Input

The PDKM system receives field data records using XML and web services technology. Field data records are usually submitted to the PDKM via the PROMISE Middleware (compare DR9.4). Furthermore the user has the option to import field data records contained in a XML-file.
Version 1 of the PDKM prototype does not yet support the interface to the Middleware but only the upload via XML-file.

By using additional tools which are not seen as part of the PDKM and which re-format e.g. XLS-files into the required XML-structure it is already now possible to upload test data from the application scenarios into the PDKM. These tools might be re-used/recycled in the PROMISE context e.g. for development of adapters connecting *Equipments* to the PROMISE Middleware.

When realising the interface to the Middleware there will besides the data delivery via the Middleware also offered user functions that allow users to regulate the PDKM – Middleware interaction like triggering the data import. In the same course the user will be enabled to send information to PEIDs using the PDKM. Details about user functionality in the context of the PDKM – Middleware interaction will be described in DR9.6b.

3.6 Incidents Management

Information about the usage of *Equipments* can also be captured in form of *incidents* (SAP: Notification; DR9.2: EVENT). The PDKM will offer different PROMISE-specific *Notification Types* (SAP-term) like Breakdown Events, (general) Notifications, Malfunction Reports, and Maintenance Requests. BT-LOC-Incidents and MTS-Incidents are *Notification Types* that can already be instantiated to *Notifications* in version 1 of the PDKM prototype. *Notifications* are attached to *Equipments* or *Equipment Templates* and can contain any *additional* (*field*) *data that describe the Notification* (SAP: Item; DR9.2: FIELD_DATA). In order to control which *Items* can be associated to a *Notification*, *types of possible Items* (SAP: Code; DR9.2: VALID_FD_TYPE) are grouped in *Code Groups* (SAP-term) that are associated with the respective *Notification Type*. Furthermore *Notification Types* can be configured to allow the user to associate a *Document Info Record* (see section 3.7) to a *Notification*.

Additionally there is the possibility to associate *Measuring Documents* via *Measurement Points* to a *Notification.* But when analysing the test data provided so far by the PROMISE application work packages it turned out that the use of *Items* is more feasible than using *Measuring Documents.*

Notification reports can be generated in PDKM for a certain *Notification Type*, the status of a *Notification*, range of *Equipments*, range of *Materials*, or a time period. “Report” in this context means the display of a list and summarising data in the user interface. *Notifications* can be entered manually into the system but also bulk loaded from XML-formatted files.

3.7 Document Management

In the lifecycle of a product there are many documents that are related to it. PDKM offers functionality to attach to

- elements in a product structure,
- the product structure itself,
- *Notifications* of specific types,

and some other objects *Document Info Records* containing metadata about associated physical files (SAP: Original; DR9.2: FILE). To a *Document Info Record* there are associated 0 to many *Originals*[3].

As an application example *Document Info Records* containing design information might be attached to a *Material* and affect the BOL phase in the product lifecycle. Whereas *Document Info Records* containing usage information (e.g. usage profiles, analysis documents etc.) might be attached to an *Equipment* or an *Equipment Template* and affect the BOL or EOL phase in the lifecycle. Since the *Equipment* is instantiated from a *Material* design information provided via a *Document Info Record* is also accessible via the *Equipment* and information from the MOL and EOL phases stored in *Document Info Records* respectively the *Originals* associated to it is accessible via the *Material*.

For *Document Info Records* and *Originals* there is provided the "standard" document management functionality expected from a "standard" document management system e.g. version tracking, check in/check out of *Originals* etc.

3.8 Knowledge Management

There has to be clearly distinguished between knowledge generation and the management of generated or otherwise provided knowledge. The first one is subject of section 3.9 while the latter is described here.
The knowledge management uses the functions described for field data (section 3.4), incidents (section 3.6), and document management (section 3.7) while the respective objects can be semantically recognised as knowledge by respective indicators that are set during the creation of these objects.

3.9 Data Analysis and Knowledge Generation Functions

Most of the analysis and knowledge generation functions are provided by the PROMISE DSS developed in WP R8. In DR9.4 is described in detail how the DSS is integrated into the PDKM.
Besides that there has been realised so far only some basic but already useful functions for browsing and displaying data and knowledge (see previous sections) like displaying *Measuring Documents* in one graphic allowing to compare the data from different *Measurement Points*.

3.10 Access Control

The users of the PDKM system logging on to the web interface are authenticated by a login name and password.
According to the needs of the different application scenarios the system administrator can define complex user roles by grouping elementary user roles that grant access to creating, changing, or viewing the different objects in the system. Thereby these elementary user roles are in a reasonable granularity. E.g. it does not make sense to allow somebody to create a specific object without allowing him at the same time to view it.
In the revision of this deliverable, DR9.6b, will be defined a default set of such complex user roles that might be readily used by the application scenarios.

[3] Using a *Document Info Record* without associating an *Original* to it offers principally the opportunity to store additional metadata to the mentioned objects. This option is not used so far in the PDKM.

3.11 Additional functions

Since the PDKM is build upon an existing PLM system the users of the PDKM have access to all functions offered by this system within the limitations of their assigned role (compare section 3.10) and possible restrictions set by the PDKM-specific functions described above.
An overview of the functions of the actually used system, SAP ECC 5.0, is available on http://help.sap.com.
Some of the above mentioned functions are readily provided by the underlying PLM system. They have been explicitly described because of their importance in the PROMISE context.

3.12 Translation of terms

In the previous sections there have been used terms of the different domains:

- The semantic domain that non-expert users use when talking about the objects.
- The SAP-domain that is relevant since the PDKM is implemented based on SAP software.
- The object model domain that references to the respective objects defined in the object model of deliverable DR9.2.

Table 1 summarises the translation between the terms of the different domains. Thereby it should be considered that not all translations are indeed a 1:1 mapping. For the semantic domain as a matter of course not all respective terms that correspond to the terms of the other domains are given but only some examples.

Semantically (examples)	SAP	Object model (DR9.2)	Comment
specific product type	Material	AS_DESIGNED_PRODUCT	
as-designed product structure	Bill of Material, BoM	self-association of AS_DESIGNED_PRODUCT	
individual product	Equipment with Serial Number, shortly Equipment	PHYSICAL_PRODUCT	
as-build/as-used (shortly “as-used”) product structure	Installed Base	self-association of PHYSICAL_PRODUCT	
product templates	Equipment Template (PDKM-term)	AS_DESIGNED_PRODUCT	From SAP-system point of view an Equipment Template is an instance of an Equipment semantically recognizable as Template.
product template structure	Installed Base Template (PDKM-term)	self-association of AS_DESIGNED_PRODUCT	analogue to Equipment Template
customisable subset of metadata of products	Classification	PROPERTY	
document metadata	Document Info	DOCUMENT or (depending	

Semantically (examples)	SAP	Object model (DR9.2)	Comment
	Record	on the context) DOCUMENT_RESOURCE with associated DOCUMENT	
document (in contrary to document metadata), (physical) file	Original	FILE	
field data in the form of a single value	Measuring Document	FIELD_DATA	
field data in the form of a document	Document Info Record semantically recognisable as field data with associated Original	FIELD_DATA with associated DOCUMENT with associated FILE	
type of field data	Characteristic	VALID_FD_TYPE	
a sensor/an info item of a PEID	Measurement Point	FD_SOURCE	
incident, event	Notification	EVENT	
type of incident, event	Notification Type	(no correspondence)	
additional (field) data describing an incident, event	Item	FIELD_DATA	
types of possible additional (field) data describing an incident, event	Code	VALID_FD_TYPE	
classes of types of possible additional (field) data describing an incident, event	Code Groups	(no correspondence)	
knowledge	depending on the context Notification, Measuring Document, or Document Info Record with associated Original(s) semantically recognisable as Knowledge	depending on the context FIELD_DATA or EVENT with eventual associations	Certainly all product-related data stored in the system is accessible for the user. Context-dependent the methodical access to it might result in knowledge.

Table 1: Translation between terms of different domains

4 Extended Field Data Functions

The PDKM system has been developed by customising and further development of a commercially available PDM system. After a comprehensive study and evaluation of PDM systems on the market (see DR7.3, chapter 3), it has been ascertained that these tools do not offer field data functions that can support a PROMISE application scenario optimally (see DR7.3 chapter 2). According to the requirements analysis from the application scenarios, a specification for the PDKM system has been derived.
The current implementation of the PDKM system as described in DR9.3 covers field data functionality that enables definition of different field data types and their storage and management. Field data can be attached to product instances and analysed based on different views. Furthermore it is possible to introduce an event based workflow management, i.e. to specify certain processes according to field data actions and content.

5 Middleware Integration

The ability of the PDKM system to communicate with other system using the ISC architecture clearly distinguishes it from commercially available PDM system. It is easily possible to integrate new data sources and expand the data schema. This has been possible by implementing PROMISE messaging interface (PMI) as described in the Deliverable DR6.5.
Thanks to the PMI, PDKM is not only capable of integrating information submitted by PEIDs, but also from any other system that implements this interface, e.g. another system which acts as a PDKM as well. Using this feature, the integration of the systems PDKM and DSS has also been based on communication via PMI partially.

6 Conclusions

After an analysis of the gaps between the requirements of PROMISE application scenarios and the functions of commercially available PDM tools, a specification for the PDKM system has been developed. This specification includes aspects of extended field data functions and middleware integration. The current implementation of the PDKM system and its usage in PROMISE application scenarios demonstrate enhancements to the commercial systems.

7 References

PROMISE DR9.1 „Design of PROMISE Information Management System“
PROMISE DR9.2 „Specification of the System Object Model“
PROMISE DR9.3 „Specification of System Functions“
PROMISE DR9.4 „Specification of interfaces“
PROMISE DR7.3 „Selection of Tools and an existing PDM System to support PROMISE specific Knowledge and Information Management Processes“
PROMISE DR6.5 „Fully Functional Device Controller“

