DR12.2: PROMISE Architecture Guide (M42)

Written by:
David Potter, INDYON GmbH

DELIVERABLE NO	DR12.2: PROMISE Architecture Guide (M42)
DISSEMINATION LEVEL	**CONFIDENTIAL**
DATE	08 May 2008
WORK PACKAGE NO	WP R12: Architecture Integration
VERSION NO.	3.1
ELECTRONIC FILE CODE	dr12.2 architecture guide v3.1.doc
CONTRACT NO	507100 PROMISE A Project of the 6th Framework Programme Information Society Technologies (IST)
ABSTRACT	This document charts and explains the significant steps in the evolution of the elements of the PROMISE architecture. It also explains the justification for the development of the Architecture Series reference documents, and positions the PROMISE project demonstrators in relation to the evolving architecture levels. This is the M42 version of this document, recording the state of the PROMISE architecture at the end of the project.

STATUS OF DELIVERABLE		
ACTION	**BY**	**DATE** (dd.mm.yyyy)
SUBMITTED (author(s))	David Potter	08.05.2008
VU (WP Leader)	Guido Stromberg	08.05.2008
APPROVED (QIM)	Dimitris Kiritsis	09.05.2008

Revision History

Date (dd.mm.yyyy)	Version	Author	Comments
31.07.2007	0.0	David Potter	Initial draft.
08.08.2007	0.1	David Potter	Minor editorial changes, and incorporation of comments from SAP.
13.08.2007	0.2	David Potter	Merged comments from SAP and Cambridge.
17.08.2007	0.3	David Potter	Merged HUT comments plus two of my own
23.08.2007	0.4	David Potter	Merged most of InMediasP comments (full commented version available in eRoom), Also removed Appendix A.
04.09.2007	0.5	David Potter	Accept all changes to date.
14.09.2007	0.6	David Potter	AS volume name updates.
03.10.2007	1.0	David Potter	Final version for M30.
15.10.2007	1.1	David Potter	Peer review feedback incorporated.
06.12.2007	2.0	David Potter	Revised version for M36.
01.05.2008	3.0	David Potter	Revised version for M42.
08.05.2008	3.1	David Potter	Added table of DSS algorithms.

Author(s)' contact information

Name	Organisation	E-mail	Tel	Fax
David Potter	INDYON	david.potter@indyon.de	+44 23 9234 5152	+44 23 9259 2327

Table of Contents

List of Figures

List of Tables

Abbreviations

Abbreviations used in this document:

AS	Architecture Series
BOL	Beginning Of Life
DC	Device Controller
DSS	Decision Support System
ECP	Embedded Core PEID
EOL	End Of Life
ERP	Enterprise Resource Planning
IOCI	Inter-Organisational Communications Interface
ISC	Inter System Communication
ITA	Interim Technical Assessment
MOL	Middle Of Life
MW	Middleware
OBC	On-Board Computer
PAC	PEID Access Container
PDKM	Product Data and Knowledge Management
PEID	Product Embedded Information Device
PLM	Product Lifecycle Management
PMI	PROMISE Messaging Interface
PSB	Project Steering Board
TIW	Technical Integration Workshop
UPnP	Universal Plug and Play
XML	Extensible Markup Language

1 Introduction

1.1 Background

During the first 12-15 months of the PROMISE project, the structure and detail of the elements of the PROMISE architecture began to emerge. This evolution was documented in a number of Research Cluster deliverables published at different times during that period.

Also during that time, and since, further requirements began to emerge from the PROMISE demonstrators that required revision or extension to the initial architecture elements. Some of these revisions were included in an ad hoc manner into later deliverables. However they were not incorporated in a consistent manner, since there was no plan or provision for revision and updating of the earlier research deliverables.

As a result some parts of the architecture and interface specifications were updated in later deliverables, while other parts remained unchanged and mixed with now obsolete information. This made it difficult for a user, such as a demonstrator owner, to find a definitive point of reference.

Therefore, during the PROMISE Project Meeting held in Cambridge in April 2006, it was proposed during the Project Steering Board meeting, discussed and unanimously endorsed, that a separate architecture reference library should be created. Into this could be extracted the current and definitive architecture-related material from all existing research deliverables. Thereafter changes and revisions could be made to these reference documents, thus providing a single point of reference to the latest published version of the PROMISE Architecture.

In parallel, the technology providers and research partners were working together to ensure that the requirements of the PROMISE demonstrators would be met. At the first Technical Integration Workshop (TIW-1) in Hennigsdorf, Berlin in March 2006 a number of challenges were recognised. Most of these were due to a more detailed understanding of requirements as the demonstrator owners' more detailed designs began to emerge, but others were due to some gaps that had been identified in the architecture specifications. Therefore, work was initiated to address these challenges.

As a consequence, the second Technical Integration Workshop (TIW-2) was held in Munich in July 2006 to review the results of that work. It was during TIW-2 that the decision was made to revise in particular the PROMISE middleware interface specifications using a completely XML-based approach. This led to the definition of the first version of what was initially called the *PROMISE Middleware Interface* or PMI.

It is also important to recognise the influence of work on standards on the stages of the PROMISE architecture evolution. At the second Interim Technical Assessment (ITA-2), Brussels, June 2006, the reviewers called for a more aggressive effort on standards (ITA-2 Recommendation 8). So by the time of the second Project Review Meeting in Turin, January 2007, the revised standardisation plan was fundamentally tied to significant elements of the PROMISE architecture, namely the PDKM System Object Model and the PLM Event subset of the middleware interface specification (PMI).

As a result of the Turin Review, a new review recommendation (also Recommendation 8) urged that the entire PMI specification should be proposed as a candidate for standardisation. The partners involved in the architecture and standards effort readily agreed and so this factor also has a considerable influence in the on-going development of the architecture concepts and the PMI specification in particular.

The PROMISE Project Meeting in Galway, May 2007, endorsed the consolidated proposals for architecture, standards and exploitation, recognising that success in all three of these areas requires a fully integrated and inter-dependent set of activities that support each other.

Finally, a PROMISE architecture review was held in Hennigsdorf, Berlin on June 8th 2007, where all the partners actively involved in the development of the PROMISE architecture met, discussed and endorsed many new architecture propositions, only some of the concepts of which will be introduced in this document, although described in full in *PROMISE Architecture Volume 1: Architecture Overview*.

1.2 Approach

In order to attract the maximum appeal and acceptance as elements for standardisation and exploitation beyond month 42 of the project (May 2008), the innovation and flexibility of the PROMISE architecture and standards needed to go far beyond the extent of their definition at month 15.

As the momentum for the standards initiatives and exploitation increased, so the vision for the PROMISE architecture began to greatly accelerate beyond the immediate needs of the PROMISE demonstrators. Therefore we have taken great care to ensure that any architecture changes did not enforce any changes on existing implementation plans for the demonstrators.

When the first level of the PMI was defined during 2006, the contributing partners were very conscious of the need to protect technology developments already made within the project by some partners following the initial architecture specifications. It remained a key objective to minimise impact on existing developments as we extend the scope of the architecture and interfaces.

It was necessary to maintain a clear distinction between the base architecture requirements of the PROMISE project demonstrators and the much greater scope of changes to support post-M42 objectives in architecture, standards and exploitation.

Therefore we have defined levels of certain architecture elements which allow a clear separation of function which will also be explained later in this document.

1.3 Purpose of document

This document is intended for **internal PROMISE consortium use** having three main objectives:

1. To chart the evolution of the PROMISE architecture and specifications clearly defining the different levels of the architecture and specifications. It will especially differentiate between those levels that have existed inside the PROMISE project itself as steps of the evolution and the initial level which will be published externally in promoting standards and exploitation, (Section 2).
2. To introduce the PROMISE architecture reference library, the so-called ***PROMISE Architecture Series***, explain its purpose, document its previously existing Research Cluster sources, and summarise its extensions (Section 3).
3. To position the actual implementation levels of the PROMISE demonstrators and the technology implementations that they use to show how they comply with the PROMISE architecture and specifications (Section 4).

The following sections of this document will deal with each objective in turn.

2 Evolution of the PROMISE Architecture

2.1 Overview

The initial definition of the PROMISE architecture comprised a layered concept as represented by the following simplified diagram:

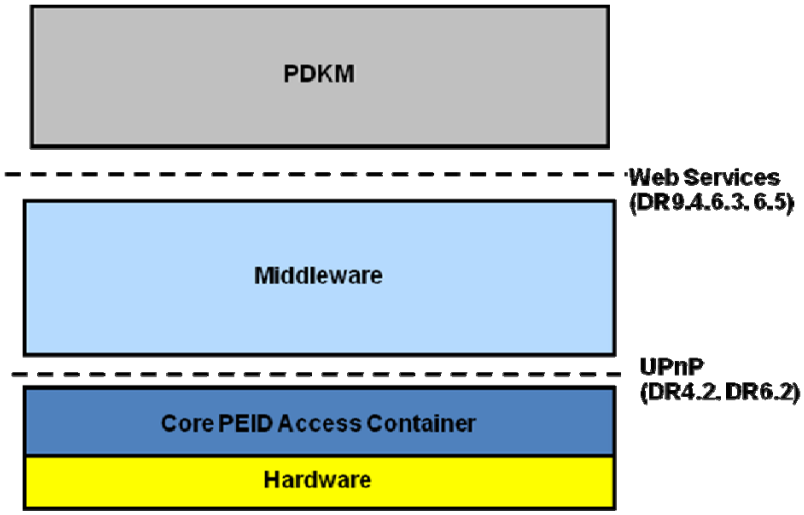

Figure 1: Initial Layered Architecture

This simplified diagram identifies the major layers in the architecture and indicates the Research Cluster documents in which their respective interface definitions were first defined. These references will of course be superseded by the Architecture Series volumes once their preparation is complete.

It was possible to conclude from this diagram that a PROMISE implementation should use all layer elements. Later project experience showed this was not necessarily true, e.g. Bombardier (A10) uses neither PEID nor middleware. Earlier in the project this seemed to be a problem to position A10 as a true PROMISE demonstrator. However, when it later became clear that A10 would use the PMI as the direct interface between field databases and the PDKM, this started to make clear that elements of the PROMISE architecture could be used in more flexible ways.

It also highlighted some limitations in the initial demonstrator requirements gathering in as much as there was no real consideration of how to integrate other data sources, back-end systems such as ERP, production or manufacturing control, or warehouse management systems, or as in the case of A10, already existing databases. These are the kinds of gaps that had begun to emerge at TIW-1 in March 2006 as demonstrator plans began to become more detailed.

As the involved partners made more progress with the detailed definition of the PMI, it became clear that it could be regarded as more than an interface between PROMISE middleware (MW) and the PROMISE PDKM/DSS.

The first step was to imagine the PMI as the generic interface for all types of user connecting to PROMISE middleware as depicted in Figure 2: A revised PROMISE component concept on page 6.

This sample component diagram shows the PMI as the common interface not only between the PDKM/DSS and the middleware, and between other so-called back-end systems and the middleware, but also between certain classes of PEID and device controllers and the middleware.

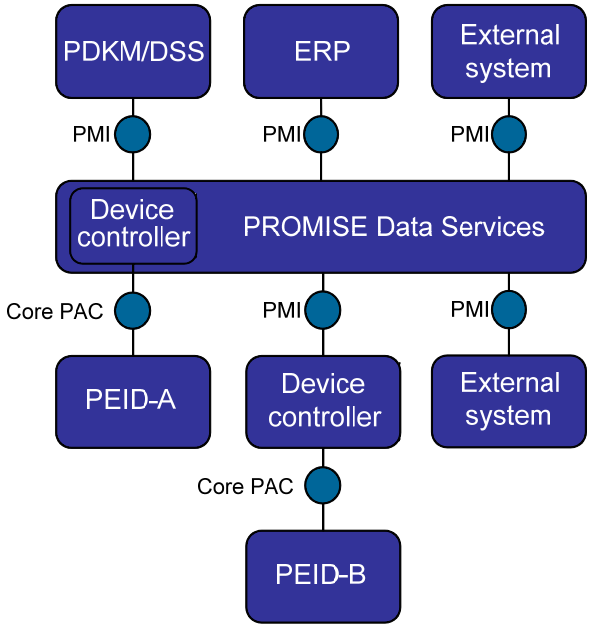

Figure 2: A revised PROMISE component concept

In the context of the above diagram, the term "PMI" still retains its original conception as the "middleware" interface. By comparison in the following example component configurations, Figure 3 through Figure 5, the alternative, and now preferred term PROMISE ***Messaging*** Interface becomes more appropriate.

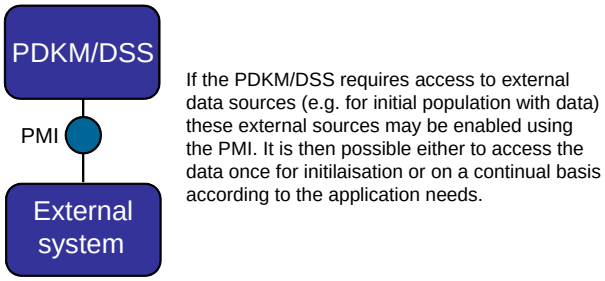

Figure 3: PDKM directly connected to data source

Figure 3 depicts the use of the PMI, specifically the XML schema, as a data exchange mechanism between the PDKM/DSS and either an external system or database, without the need for the services of a middleware component. This is the configuration used by the Bombardier (A10) demonstrator and is also considered for use by the Teksid demonstrator (A11).

It should be emphasised that the possibility to connect PROMISE components directly to each other according to the needs of any specific application is a major strength in flexibility of the architecture. It does not in any way lessen the value of a PROMISE middleware implementation. Simply the Bombardier application does not require a sophisticated middleware which can discover, buffer, filter and aggregate, while applications which may have multiple, non-persistent and less predictable data exchanges with PEIDs and other data sources can benefit from a middleware rich in service functions to manage those exchanges.

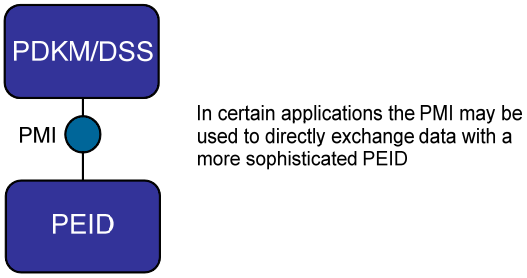

Figure 4: PEID directly connected to PDKM/DSS

In an application case where there are few, high-function PEIDs, it may be desirable to connect the PEID directly to the PDKM/DSS without an intervening middleware layer, as shown in Figure 4 above.

This kind of possibility was not considered in the early stages of the PROMISE architecture, since it only became apparent during the later stages of Work Package R5 leading to the concept of different levels of PEID capability as documented in *DR5.4 Generic PEID roadmap for each group*. This expanded the PEID concept in particular to include the possibility of implementing the then so-called "middleware interface" directly in the PEID itself. This is attractive in the case of a particularly sophisticated PEID such as an on-board computer (OBC) which has the capacity and system flexibility to integrate the PMI functionality rather than use an external proxy

Once again, it is important to emphasise that this kind of sophisticated PEID approach does not invalidate in any way the PROMISE Core PAC or Embedded Core PEID (ECP) concepts or their use as a proxy. It merely strengthens the PROMISE architecture by allowing appropriate flexibility in implementation.

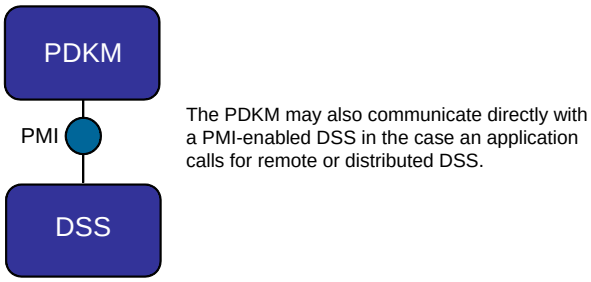

Figure 5: DSS decoupled from PDKM

Finally for the purposes of this overview, Figure 5 shows a DSS implementation which is decoupled from the PDKM using the PMI for data access. If a middleware component intervened, then this access could also be asynchronous. This shows that the flexibility of the PROMISE architecture can be applied according to the needs of individual applications.

Figure 5 is not intended to imply that the PDKM/DSS should not be closely coupled. In the project implementation, the greatest strength of the close coupling of PDKM and DSS is the much higher performance that can be obtained when the DSS has direct read access to the database, which allows formulating queries that are more complex as it would be possible when using the PMI. However, the flexibility to decouple the DSS from the PDKM enables the possibility of distributed DSS functionality, which would have been applicable for example in the FIDIA (A6) demonstrator case.

In the early architecture concept, where it was assumed that the DSS would be always closely coupled to the PDKM, and would read directly from the PDKM database, it was initially proposed that the DSS would use still-to-be-specified web services to write data to the PDKM. In the process of revision, it became obvious that it was unnecessary to specify additional web service

functions since the DSS could use the PMI in order to write to the PDKM, and this would function equally whether closely or loosely coupled. Therefore this approach has been adopted.

In summary then, this section has sought, using just a few examples, to give an overview of the evolution of the PROMISE architecture concepts, the full extent of which may be found in *PROMISE Architecture Volume 1: Architecture Overview*. The following sections will detail the significant milestones in the development of the PROMISE architecture and the corresponding specification levels.

2.2 Architecture Chronology

The chronology in Table 1 below charts the most significant steps in the evolution of the PROMISE architecture. It helps to understand why there has been the need for the re-examination and revision of elements of the architecture as over time more and more detailed work items have come to fruition.

Table 1: PROMISE architecture chronology

Date	Architecture Event
13.05.2005	Initial architecture and component overviews DR6.1: *Functional Specification of the PROMISE Middleware*
15.08.2005	Interface between Core PEID Prototype and Device Controller DR4.2: *PEID Core Prototype* and DR6.2: *PEID - Device Controller Specification*
14.11.2005	Initial DSS architecture views DR8.1: *Design of the decision support system*
15.11.2005	Initial PDKM architecture views DR9.1: *Design of PROMISE Information Management System (PDKM)*
15.02.2006	Specification of Application-facing Interface (initial middleware interface) DR6.3: *Fully Functional Device Controller*
15.02.2006	DR9.2: *Specification of the System Object Model*
06-07.03.2006	Technical Integration Workshop (TIW-1) Hennigsdorf: Architecture gaps highlighted by more detailed demonstrator requirements and feedback from initial technology implementations.
28.04.2006	PSB Meeting No.4 Cambridge (Item 05/06): Decision to create Architecture Series
12.05.2006	Multiple PEID levels defined DR5.4: *Generic PEID roadmap for each group*
15.05.2006	DR9.4: *Specification of interfaces (PDKM)*
20.07.2006	Technical Integration Workshop (TIW-2) Munich: Decision to merge and revise existing middleware and proposed ISC interfaces to create PMI
16.11.2006	Initial PMI specification published DR6.5: *Interface definition and design of enterprise communication infrastructure*
02.05.2007	PROMISE Project Meeting Galway: Architecture and Standards direction endorsed
08.06.2007	PROMISE Architecture Review Hennigsdorf: approval of several revisions and extensions to the architecture concepts and PMI specification
05.07.2007	PROMISE Project Management Meeting Brussels: Confirmed results from Architecture Review in Hennigsdorf
06.07.2007	PROMISE project architecture change control process established
17.08.2007	PMI Version 2.0 finalised. Documented only in the eRoom for use in the project demonstrators.
14.04.2008	PMI Version 3.0 finalised. Documented in Architecture Series Volume 3.

This chronology also helps to highlight the challenge of keeping architecture concepts and specifications current when they have evolved and been documented in a sequential manner. This has been the strongest driver for creation of the PROMISE Architecture Series.

2.3 Currency of Specifications

The decision to create the PROMISE Architecture Series as a reference library of current concepts, specifications and developer's guides, was taken to establish a consolidated and authoritative set of reference material. However owing to the high rate of revision it was decided to defer the work of extraction of material from existing Research Cluster documents until a clear position regarding the future of the PROMISE architecture and standards had been established. This milestone was reached in July 2007.

The following table defines the currency of the PROMISE architecture specifications.

Table 2: Currency of Specifications

Specification	Location	Availability
Architecture concepts, diagrams and overview	AS Volume 1	M42
Multiple PEID levels defined	AS Volume 1	M42
Interface between Core PEID Prototype and Device Controller	AS Volume 2	M39
PMI specification (Version 2)[1]	eRoom[2]	M33
PMI specification (Version 3)[3]	AS Volume 3	M42
PDKM architecture	AS Volume 4	M42
Specification of the System Object Model	AS Volume 4	M42
Specification of interfaces (PDKM)	AS Volume 4	M42
DSS architecture	AS Volume 5	M42

[1] PMI Version 2 is for internal project use only
[2] PMI Version 2 will only be published in the PROMISE eRoom for the benefit of technology developers in the PROMISE consortium
[3] PMI Version 3 will be the earliest public level submitted for consideration as a standard.

2.4 PROMISE Messaging Interface (PMI)

The emergence of the PMI as the major interface in PROMISE resulted from the recognition leading up to the second Technical Integration Workshop (TIW-2), that was held in Munich in July 2006, that not only were multiple interfaces being investigated, but also that they could actually be developed.

Figure 6 below shows an intermediate, though short-lived, stage in the evolution of the architecture, combining the presentation of components (Application, PDKM, DC, Hardware) and interfaces (IOCI, Core PAC). The initial middleware interface, the Device Controller (DC) interface, had already been defined in DR6.3, and even mostly implemented by some of the PROMISE technology developers. The next planned step was to extend the middleware functionality with the so-called Inter-Organisational Communications Interface (IOCI), which function is now known as Inter System Communication (ISC).

As the diagram below clearly shows, the IOCI interface was conceived as an additional and separate interface to the already defined middleware Device Controller (DC) interface.

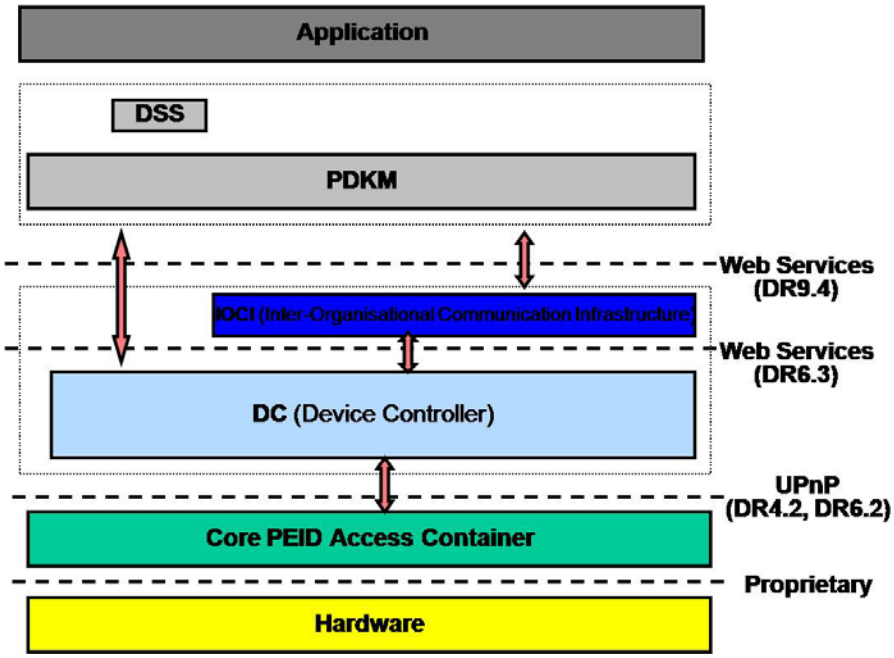

Figure 6: Intermediate architecture view July 2006

If the evolution and development of these interfaces had continued according to the original vision and plan, then back-end systems like the PDKM would have been obliged to implement 2 separate interfaces together with logic to determine when to use each one.

The preparatory work leading up to TIW-2 concluded that it would be better to consolidate the DC and ISC interfaces into a single interface to be used by the PDKM and other back-end systems, and to provide the logic to decide if the request was a “DC request” or an “ISC request” within the middleware itself. It was also recommended that the whole interface be redefined as an XML schema. These conclusions were endorsed at TIW-2, leading to the definition of Version 1 of the then so-called PROMISE Middleware Interface (PMI).

Version 1 of the PMI was published in November 2006 in DR6.5, and initially became the reference level to which the PROMISE project technology providers developed their middleware and PDKM implementations. It was superseded by Version 2 in August 2007.

As work continued on the detailed definition of the PMI, further analysis of more detailed demonstrator requirements and apparent gaps in the architecture, additional opportunities for consolidation on the PMI became clear.

As early as November 2005, the following diagram, Figure 7, taken from DR9.1 *Design of PROMISE Information Management System (PDKM)*, showed the consideration of two different middleware channels into the PDKM, one from PEIDs via PROMISE Middleware and the second from operational data sources via other middleware implementations.

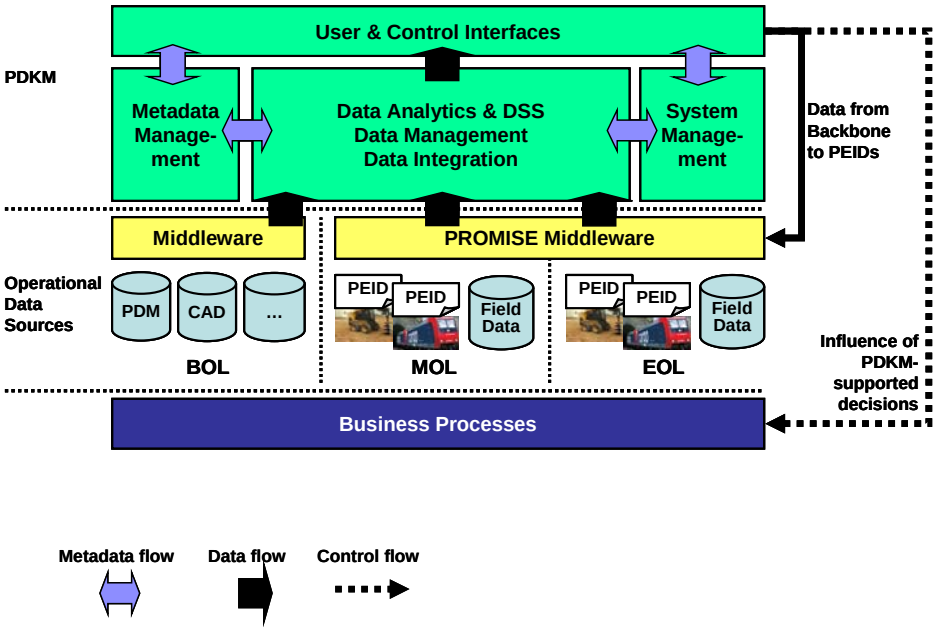

Figure 7: PROMISE architecture (view from DR9.1)

However, while working on the integration of the PDKM with other data sources for the A10 Bombardier demonstrator, Bombardier and InMediasP together realised that using the PMI XML schema would be the logical solution. In reality, this meant using the PROMISE PMI interface specification, but without any PROMISE middleware implementation. This was an important step in the recognition of the PMI as being more than a middleware interface.

Additional important steps came through:

1. The recognition by InMediasP that the PMI would offer best solution for the DSS to write to the PDKM rather than by using a separate web services interface as had been originally proposed.
2. The recognition that the PMI would be an appropriate interface that could be used by many kinds of existing or new “non-PROMISE” systems, including systems associated with servicing or monitoring of products, logistics systems, fleet and warehouse management systems, and even ERP systems, etc.. All of these could exploit the PMI to request data from products whenever it should become available, or provide data to other systems or PEIDs connected to the PROMISE infrastructure.
3. A proposal that PEIDs or their proxies might in the future use the PMI to communicate directly with other PEIDs. Although there is no PROMISE demonstrator requirement for such function, it should not be excluded from the architecture concepts.
4. A proposal that the PMI would be an appropriate back-end interface for an external Device Controller, i.e. a DC implemented outside of a PROMISE middleware instance.

Therefore the transition from the “middleware interface” was completed as the PMI became the more generic “PROMISE Messaging Interface”.

PMI Version 1 was for PROMISE project internal use only; it was made obsolete in August 2007 by Version 2, and will not be published outside of the project consortium.

In order to protect the technology providers and the demonstrator owners from repeated re-working of their implementations, it was necessary to freeze the internal PMI specification. This was done at the end of August 2007 only after essential additions explicitly required in order to ensure the success of the PROMISE demonstrators had been incorporated.

An architecture change control process was established, and changes were considered to Version 2 but only if they met emergency criteria because one or more demonstrators could not be implemented without the change. The objective was to maintain extremely rigid control in order to avoid unnecessary re-implementation work. Changes to the PMI affect not only the middleware implementations but also users of the PMI such as the PDKM and DSS, and in some demonstrator cases other systems and databases.

Therefore the reference level of the PMI for all PROMISE demonstrator implementations has been PMI Version 2.

The public PMI version to be submitted as a candidate for standardisation will be called Version 3. It will be based upon Version 2 and any emergency changes required to Version 2 will be automatically applied also to Version 3.

PMI Version 3 has been extended to further develop the PMI functions and the PROMISE architecture in general to extend their appeal to a broader marketplace.

Changes implemented in the Version 3 PMI specification include:

- Migration from a multi functions Web Service communication interface to a one function Web Service or HTTP POST communication interface. Removing unnecessary complexity from the interface itself makes the PMI easier to use and adapt to different scenarios. Allowing simple HTTP POST communication allows for devices with limited Web Service capabilities to send and receive PMI messages.
- Use of same interface for call-back messages as for placing requests. The PMI became a bidirectional interface. (Systems are still allowed to only support one direction communication, for example, not all systems can and should support handling of subscription requests.)
- Moved from a PEID centric approach to a more generic approach where data is communicated about ‘targets’, which are more general and not limited to UPnP or RFID based PEIDs.
- Introduction of function type, content type and subject type definitions to define the content of a PMI message more clearly.
- Introduction of generic subscriptions to allow easier subscription management. Subscriptions can be made to messages types (content type or subject type), id ranges, etc. Removed limitations in establishing subscriptions (such as separate subscription required per target) to allow for more generic subscriptions. The subscriptions became a mean for a system to tell what type of data it desires, and is not limited to asking for specific, predefined data instances.
- Allowed for communication between any nodes connected to a PROMISE middleware to use PMI. PMI is no longer dedicated to the communication from PDKM towards the DC handling the PEID.

- Modified metadata content to allow for various properties about targets and infoItems to be communicated.
- Introduction of new message types, PLM events, system events, device management events, alarm events.
- Made the PMI structure more general, but still specific enough to communicate data in a unanimously way. Allowed for communication of metadata, PLM event, system management events, device events and alarm events using same interfaces and message structure as for field data. The same base data structure suits all the above event types.
- Introduced metadata for message types (PLM events, system events, device management events, alarm events) for targets using same metadata structure as for field events. Targets can have metadata defined for the events they generate, or systems generate about them.
- Clarification of subscription parameters such as time-to-live and subscription interval.
- Defined a base set of PLM events common in PLM applications.
- Improved and clarified generic requirements and guidelines on how systems should handle subscriptions and requests.
- Defined basic PMI based Middleware functionality feature requirements, without defining and limiting implementations. Such functionality as discovery service, device management (keeping track of devices to make sure data messages reach the devices), system management (keeping track of other PMI nodes and their roles in the application), metadata management and storing, routing of messages between PMI nodes. The features are defined on a level of "what service they should provide", but actual implementation is outside the scope of PMI.
- Introduced PMI node information in the PMI schema to direct the PMI messages to predefined nodes where applicable.
- General improvements to the PMI XSD schema. Unnecessary complexity and limitations removed.
- Introduction of PMI XSD name space.
- Capability to extend the PMI XSD schema to allow application specific data structures.
- Initial definition of PMI level error messages.

The full specification of the PROMISE Messaging Interface Version 3 can be found in the *PROMISE Architecture Series Volume 3: Architecture Reference: PROMISE Messaging Interface (PMI).*

2.5 PROMISE PEID and the Core PAC Interface

2.5.1 The PROMISE PEID

During the early life of Work Packages R4 and R5, there was a working assumption that PROMISE would be able to conceive a Product Embedded Identification Device (PEID) which would eventually be realised in a hardware package and could be deployed in a flexible manner in all PROMISE demonstrators and in each of the life cycle stages, BOL, MOL and EOL.

As demonstrator requirements became progressively more concrete, so it also became clear that it would not be practical to apply a common PEID implementation to all demonstrators. This was also deemed to be indicative of its suitability for the wide variety of industry applications in general.

Furthermore as work progressed in WP R5, it also became clear that its original objective to specify a PEID prototype for each of the life cycle stages did not make sense since it was more likely that once the PEID was applied or embedded in a product, that it would remain with the product throughout its life, since this was important to the concept of whole-of-life data gathering on board of the PEID.

Therefore the final deliverable from WP R5, DR5.4: *Generic PEID roadmap for each group*, instead defined five different levels of PEID capability (PEID:0 through PEID:4) based on the relative computing capabilities of the identification device and ranging from a simple RFID tag or barcode to a highly-sophisticated on-board computer (OBC).

Integration of the different PEID levels differs according to the complexity of the PEID. In the case of PEID:0 and PEID:1, a reader suitable for simple RFID or barcode identifications can be integrated using a Device Controller (DC) which is either an integral part of a PROMISE middleware implementation or connected to the middleware via the Core PAC or the PMI.

In the case of PEID:2 and PEID:3, which are suitable candidates for the PROMISE Embedded Core PEID approach, the integration can be via the UPnP-based Core PAC Interface using a UPnP-enabled Device Controller, which again may be either an integral part of a PROMISE middleware implementation or connected to the middleware via the PMI.

In the case of the high-end PEID:3 or PEID:4, where an on-board computer has the power and interfaces to integrate data from a multitude of identification and sensor devices and also a system environment which permits the direct integration of the PMI, these PEID types are able to connect directly to their partner systems or via a PROMISE middleware implementation.

The PEID levels are now included in this document, Appendix A: PEID Grouping, and this information is also included in *PROMISE Architecture Series Volume 1: Architecture Overview*.

2.5.2 The PROMISE Core PAC Interface

The PROMISE Core PAC interface defines the connection between PROMISE Core PEID and a middleware Device Controller.

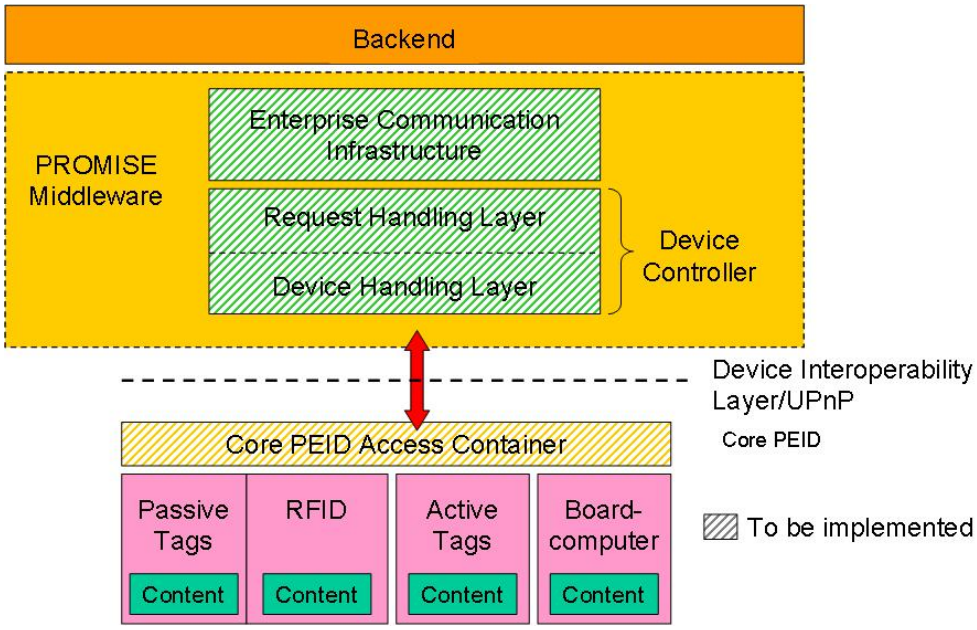

Figure 8: Location of the Core PAC Interface

Figure 8 shows an early architecture diagram from DR4.2 and DR6.2 which has not yet been formally superseded. The exposure of the internal components of the middleware layer as shown in this diagram is not desirable in an architecture representation since such details can vary greatly from one middleware implementation to another.

The important point is that a PROMISE middleware implementation should implement the PMI and Core PAC interfaces; the internal structure of a middleware implementation is not so important, and later architecture diagrams avoid this level of detail.

The documentation of the Core PAC interface itself, between PEID and Device Controller, will be migrated to *PROMISE Architecture Series Volume 2: Architecture Reference: PROMISE Core PAC Interface*. The Device Controller is a middleware concept which may be implemented as an integral part of a middleware implementation or as an external component which may use the Core PAC as its device-facing interface and the PMI as its back-end interface. This concept will be developed in greater detail in *PROMISE Architecture Series Volume 1: Architecture Overview*.

2.6 PROMISE PDKM and PDKM System Object Model

2.6.1 PROMISE PDKM

The architecture of the PROMISE PDKM, represented by the diagram in Figure 9 below, is published in Section 5 of DR9.1, which remains the authoritative reference until the publication of *PROMISE Architecture Series Volume 4: Architecture Reference: PROMISE PDKM System Object Model and Interfaces*.

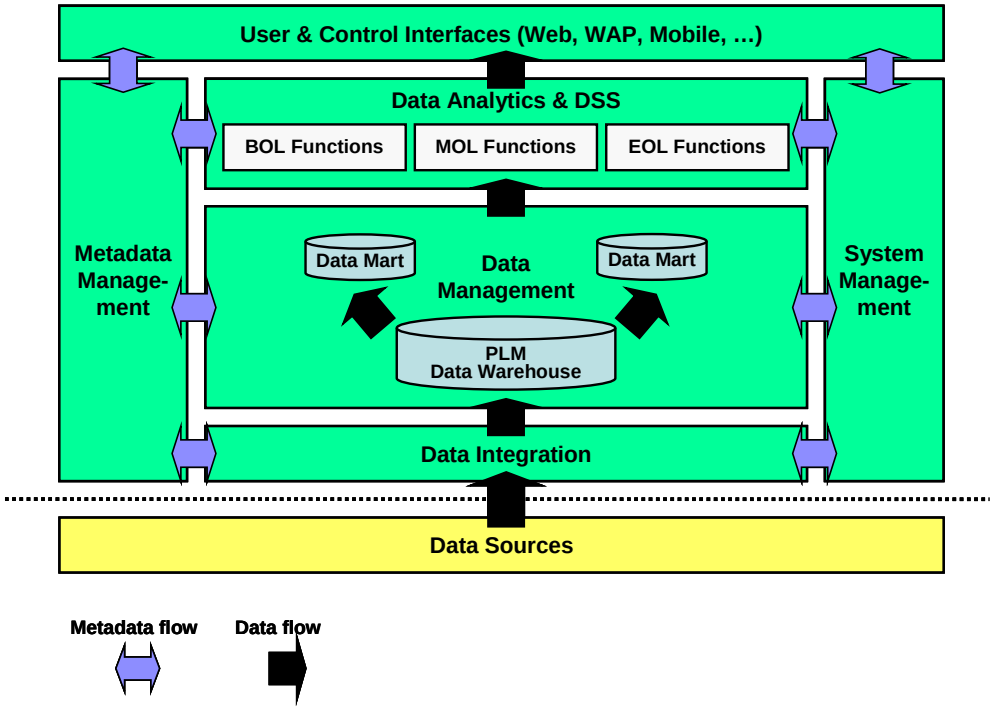

Figure 9: Architecture overview of the PDKM system

2.6.2 PDKM System Object Model

The General System Object Model of the PDKM system is outlined in Section 4 of DR9.6a, which also remains the authoritative reference until the publication of *PROMISE Architecture Series Volume 4: Architecture Reference: PROMISE PDKM System Object Model and Interfaces*.

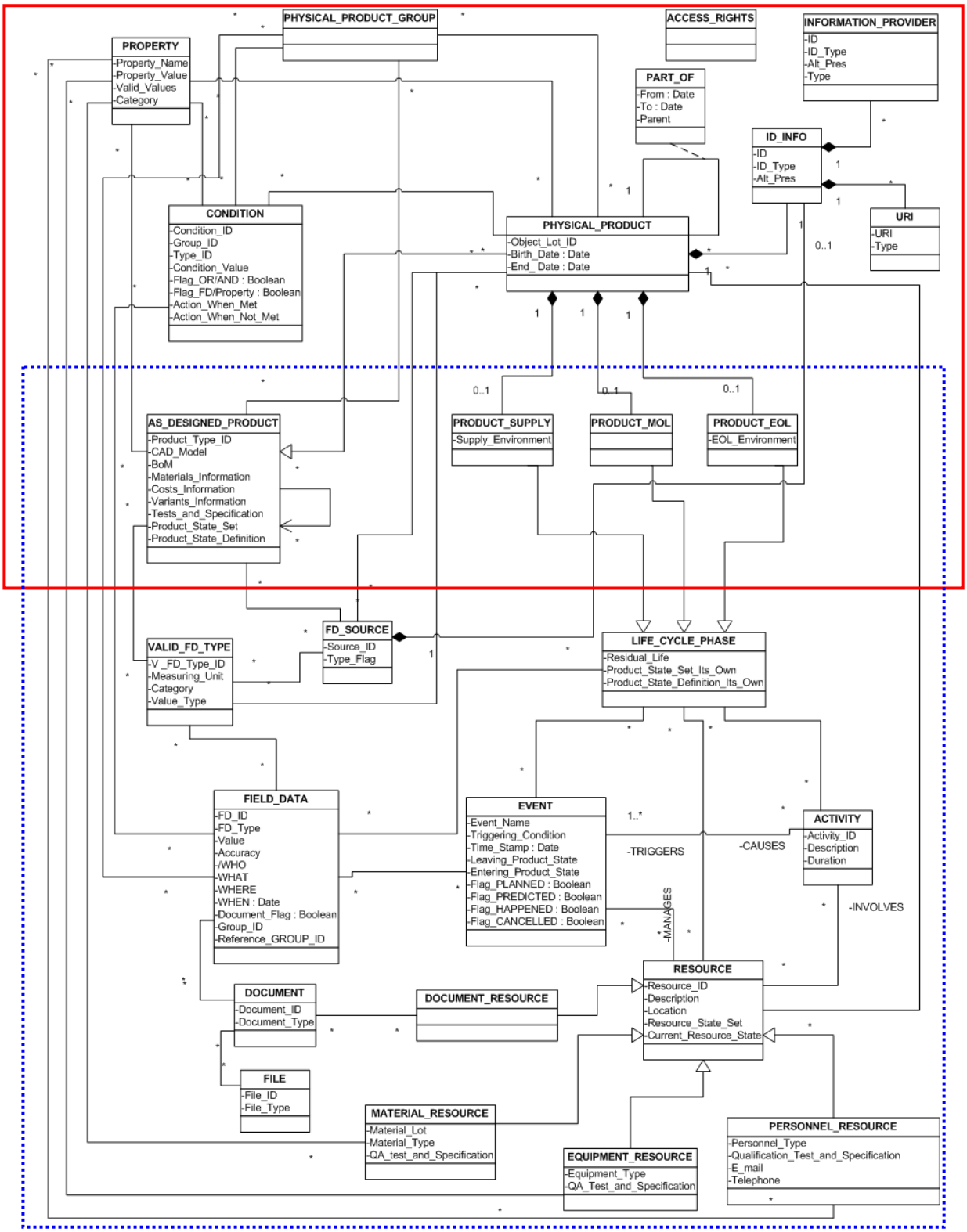

Figure 10: Complete schema of the semantic object model

2.7 PROMISE Decision Support System (DSS)

The architecture of the PROMISE DSS, as represented here by the generic DSS structure shown in Figure 11, is defined in sections 17.1 and 17.2 of DR8.1 *Design of the decision support system*, which remains the definitive source until it is superseded by *PROMISE Architecture Series Volume 5: Architecture Reference: PROMISE Decision Support System (DSS)* planned for month 37.

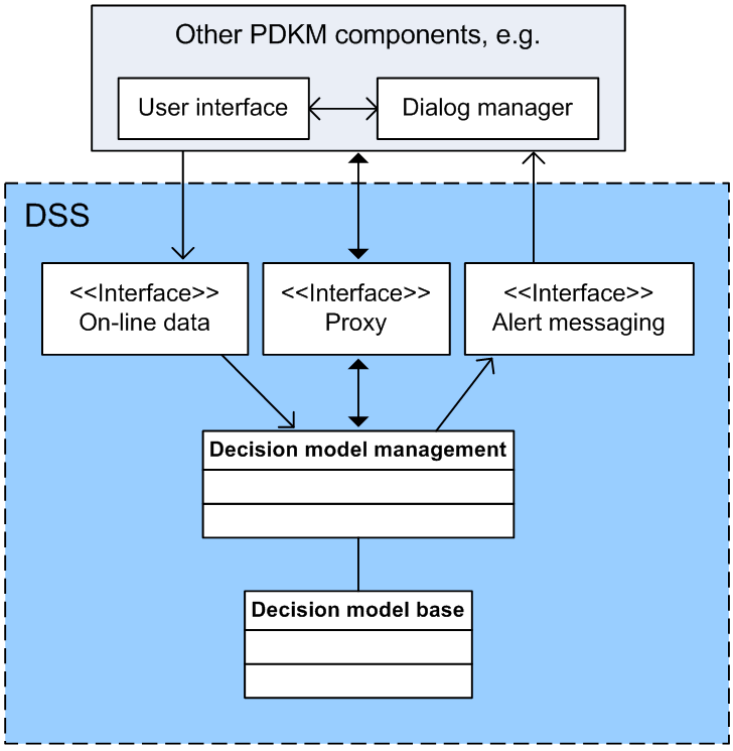

Figure 11: Generic DSS architecture

2.8 Standards Promotion

The PROMISE consortium is pursuing a standards initiative in order to promote the following:

1. The PROMISE PDKM System Object Model
2. The PROMISE Messaging Interface (PMI)

The most recent developments and status relating to proposals and promotion of PROMISE standards are documented in the WP I1 Standardisation deliverables for M42, DI1.6c and DI1.7c.

3 The PROMISE Architecture Series

3.1 Introduction

The purpose of the PROMISE Architecture Series is to provide a consolidated and definitive reference library for the concepts and interface specifications, which have resulted from the PROMISE project.

It allows the periodic revision of the reference volumes as continuing work both during and after the project leads to the need to supersede technical information, which would otherwise be impossible if it remained only in the Research Cluster deliverables.

At the time of writing of this M42 version of DR12.2 *PROMISE Architecture Guide*, the preparation status of the Architecture Series is as shown in the table below. The actual volumes available, their titles and content have been subject to change as anticipated.

Promise Innovation International Oy proposes to take responsibility for the future maintenance and development of the PROMISE Architecture Series in collaboration with other partners from the consortium who are motivated to give continued support to the development of the PROMISE architecture. However this is dependent on a clear resolution regarding intellectual property rights (IPR) and whether the consortium is able to give the required approval to Promise Innovation International Oy for this to happen.

3.2 Structure of the PROMISE Architecture Series

Table 3: TR12.4 Architecture Series: Structure

Title	Classification	Availability
Volume 1: PROMISE Architecture Overview	PUBLIC	M42
Volume 2: Architecture Reference: PROMISE Core PAC Interface	PUBLIC	M39
Volume 3: Architecture Reference: PROMISE Messaging Interface (PMI)	PUBLIC	M42
Volume 4: Architecture Reference: PROMISE PDKM System Object Model and Interfaces	PUBLIC	M42
Volume 5: Architecture Reference: PROMISE Decision Support System (DSS)	PUBLIC	M42
Volume 6: Developer's Guide: PROMISE Product Embedded Information Device (PEID)	LICENSED	TBD
Volume 7: Developer's Guide: PROMISE Data Services	LICENSED	TBD
Volume 8: Developer's Guide: PROMISE Product Data and Knowledge Management (PDKM)	LICENSED	TBD
Volume 9: Developer's Guide: PROMISE Decision Support System (DSS)	LICENSED	TBD

Table 3 presents both the available and planned volumes of the Architecture Series in 3 logical groupings: the Architecture Overview, the Architecture Reference and the Developer's Guide. The scope of each of these will be described in more detail shortly.

The table also proposes a possible system of classification to distinguish between reference materials which is considered public and freely available, and material which comprises valuable intellectual property of the consortium partners. Individual partners and groups of partners wishing to exploit commercially results from PROMISE after the end of the project may wish to protect certain volumes and license those volumes to other parties.

The classifications will be influenced by further discussions amongst the PROMISE partners. This may also lead to changes being made to the proposed structure to further separate material which should be made public and that for which it is desirable to keep appropriate control of intellectual property rights.

3.2.1 Architecture Overview

Volume 1 of the Architecture Series gives an overview of all aspects of the architecture and concepts of the PROMISE architecture and technologies. Its purpose is to position and explain the structure of PROMISE and the relationships between the architectural components and to present options for their exploitation.

3.2.2 Architecture Reference

There are four separate Architecture Reference volumes, each one focussed on a major element of the PROMISE architecture:

Volume 2: Architecture Reference: PROMISE Core PAC Interface
Volume 3: Architecture Reference: PROMISE Messaging Interface (PMI)
Volume 4: Architecture Reference: PROMISE PDKM System Object Model and Interfaces
Volume 5: Architecture Reference: PROMISE Decision Support System (DSS)

Each of these volumes will contain a description of the rationale and concepts of its respective element, together with its most current fully detailed interface specifications.

3.2.3 Developer's Guide

Four separate Developer's Guide volumes are also proposed, and each one will also be focussed on the relevant major element of the PROMISE architecture:

Volume 6: Developer's Guide: PROMISE Product Embedded Information Device (PEID)
Volume 7: Developer's Guide: PROMISE Data Services
Volume 8: Developer's Guide: PROMISE Product Data and Knowledge Management (PDKM)
Volume 9: Developer's Guide: PROMISE Decision Support System (DSS)

The value of the Developer's Guides will be seen after the end of the formal PROMISE project. Their use is mainly intended for technology providers outside of the original consortium who would like to consider developing new products or adapting existing ones that conform to PROMISE architecture concepts and specifications.

4 Compliance of the PROMISE Demonstrators

4.1 Components employed for each Demonstrator

We discovered during the course of the project, that it is neither appropriate nor essential for a single demonstrator to implement all layers of the architecture or all technology elements.

The following table, which is included from DA0.1 Integration Report (M42), contains an overview on which specific component has been customized for and integrated into each of the demonstrators. As can be seen, all demonstrators are based on the interface definitions that have been developed in PROMISE, and that various implementations of each of these components have been implemented and are used.

Table 4: Components employed for each Demonstrator

	A1 (CRF)	**A2** (CAT)	**A3** (Indyon)	**A4** (CRF)	**A5** (CAT)	**A6** (Fidia)	**A8** (Indesit)	**A9** (ICOM)	**A10** (BT)	**A11** (Polimi)
DSS status	Cognidata's part of the DSS has been developed as a generic framework in which solutions of all PROMSE application scenarios can be embedded. The embedding of the algorithms was successful. The integration of scenarios with database and GUI is still under construction except for A3. The reason for that is that the algorithms underlie a steady course of development and change quite frequently. The A3 demonstrator has its own stand-alone DSS framework into which all of its algorithms are integrated, and this framework has its own GUI. This has been fully completed.									
	SAP Research has developed the DSS GUI for all application scenarios and deployed them on the central PDKM instance for demonstration.									
			ready							
PDKM status	The PDKM system has been developed as a generic framework in order to support all promise application scenarios. The functional scope of the PDKM system has been successfully utilized in all application scenarios with the exception of A3. A3 has successfully implemented the PDKM model and application specific data model subset using open source components.									
	ready	ready	ready	ready	ready	ready	ready	ready	ready	ready
MW status	MW (DC+ISC) with PMI	SAP RHL & DHL with PMI, MOMA	INDYON MW with PMI & external DCs	MW (DC+ ISC) with PMI	SAP RHL & DHL with PMI, MOMA	MW (DC+ISC) with PMI	PMI mpl., DC, additional GUI	SAP RHL & DHL with PMI, MOMA	File based import	
	ready	ready	ready	ready	ready	ready	ready	ready	ready	N/A
PEID status	Blue & Me, Blue & Me sensors, ECU	RFID Tags, ECM	IFX ECP, RFID Tags, Track and Race®	Blue & Me, Blue & Me sensors, ECU	Crack First™ + IFX ECP	ECU on Machine	Refrigerator ECU, Smart Indesit Adapter	RFID tags, Protocol Convert.+ IFX ECP	ECU data integration via file and BT infrastructure	
	ready	ready	ready	ready	ready	ready	ready	ready	N/A	N/A

4.2 Exploitation of PEIDs in PROMISE Demonstrators

The following table, also included from DA0.1 Integration Report (M42), shows the deployment of different PEID technologies and types across the different PROMISE Demonstrator applications:

Table 5: Deployment of PEID types and technologies

No.	PEID	Type	Description
A1	**Blue&Me**	**PEID:3**	Central device for in-car data acquisition
	Blue&Me Sensors	**PEID:2**	Sensory for in-car data acquisition
	ECU	**PEID:3**	On board unit
A2	**RFID Tags**	**PEID:1**	Storage and retrieval of engine component information
	ECM	**PEID:3**	On board Electronic Control Module
A3	**RFID Tags**	**PEID:1**	Storage and retrieval of container and plastic items information
	Track and Race®	**PEID:1**	RFID transponder coordinate position matrix mounted in the floor
	Track and Race®	**PEID:4**	Calculation of position coordinates in T+R On Board Computer
	ECP with sensor	**PEID:2**	Temperature monitoring of goods
A4	**Blue&Me**	**PEID:3**	Central device for in-car data acquisition
	Blue&Me Sensors	**PEID:2**	Sensory for in-car data acquisition
	ECU	**PEID:3**	On board unit
A5	**Crack First PEID**	**PEID:1**	Measuring fatigue damage, integrated via ECP
	ECP	**PEID:2**	Storage and modification of structural part information and fatigue damage
A6	**ECU on Machine**	**PEID:4**	Local acquisition of data and direct PMI connectivity
A8	**Smart adapter**	**PEID:2**	Acquisition of status information of the refrigerator
	Internet Gateway	**PEID:4**	Communication to smart adapter and middleware connectivity
A9	**RFID tags**	**PEID:1**	Identification and data storage for line cards
	ECP	**PEID:2**	Communication of monitoring information
	Protocol Converter	**PEID:2**	Conversion of SNMP information and forwarding to ECP
	Monitoring Assembly	**PEID:3**	Combination of APU + ECP + Protocol converter as comprehensive PEID solution for the A9 needs
A10	**ECU data integration via file and BT infrastructure**	**PEID:3**	Exact up-to-datedness of field data in PDKM is secondary since the scenario deals with BOL decision support and focuses on clustering algorithms.
A11	None	N/A	N/A

4.3 Table of DSS Algorithms

Table 6 Table of DSS Algorithms

Demonstrator	Type	Notes
A1	Bayesian Network	
A2	Bayesian Network	
A3	Decision Tree	
A4	"Life Cycle Residual Cost optimisation"	A4 computes the optimal calendar for maintenance of trucks, based on the estimated wear-out of components, the vehicles and garage operator availability, the cost of maintenance and of stoppage. The DSS scheduling algorithm then leads the user (the fleet manager) to the identification of the proper calendar of maintenance actions for all his/ her vehicles.
A5	Regression Analysis (polynomial / exponential)	
A6	Fuzzy Expert System	
A8	Fuzzy Expert System	
A9	Weighted sum, normalization	
A10	1. Failure code event rate evaluation + multi linear regression model 2. Clustering	1. Failure code event rate are evaluated based on the historical data. Multi linear regression model is used to relate evaluated failure code event rate with field data (environmental and operational data) to find correlation between them. 2. Clustering is applied into the field data (environmental and operational data) to provide general overview of field data.
A11a	"What...If? Analysis"	1. Discrete-Time (Finite-State) Decomposition - as physical performance evaluation algorithm 2. Continuous-State Buffer Allocation Optimization - as the buffer allocation optimization algorithm.
A11b	"Optimal System Reconfiguration"	1. Finite-State/Finite-Time Dynamic Programming

Appendix A: PEID Grouping

Table 7: PEID Grouping

Group	Examples	Network connectivity	Computation power	Sensors, other connections?	Description
PEID:0	Barcodes, low-range passive RFID tags	Usually intermittent, by "proxy" device	Only gives read access to GUID	None	Identifier-only PEID. The PEID only contains a GUID (Globally Unique ID). The GUID is usually of write-once-read-many (WORM) type. Examples: barcode or RFID tag or any information device for which only the GUID is accessible, no matter how "computationally powerful" the PEID is.
PEID:1	Barcodes, passive RFID tags	Usually intermittent, by "proxy" device	Read access to GUID, data storage capabilities (read and possibly write)	None	Only identifier and data storage capabilities, no computation capabilities. Data storage can also be re-writable. Examples: barcode and passive RFID tag with data contents in addition to GUID. Intermittent network connectivity through proxy device (e.g. barcode reader, RFID tag-reader)
PEID:2	Active RFID, WiFi-enabled devices	Can be activated by PEID if in range	Sufficient for collecting data from external devices	Yes	Limited computation power, possibly including sensors and other "measuring" capabilities. Wireless network connectivity when "in range". Examples: ECP based PEID, active RFID, WiFi-enabled devices etc.
PEID:3	On-board computers, embedded micro-controllers, laptops	Can be activated by PEID, range depends on embedded communication technology	Sufficient for collecting data from external devices, controlling actuators, collecting statistics etc.	Yes	Medium-level computation power, sensor connectivity, data processing power. Wireless network connectivity when "in range". Example: car ECUs (as in PROMISE prototype described in deliverable DR5.2), embedded controller in general. UPnP is good for these, but for some it might be simpler to embed the PROMISE middleware connectivity with themselves.
PEID:4	PC, printers, mobile switches, user-operated PDAs or other user terminals	Always on wired or wireless connection or possible to activate anytime "when needed" either by PEID itself or human operator	Typically high	Yes	PEIDs with "sufficient" computation power e.g. for implementing "client" connectivity to PROMISE middleware Web Services or even hosting them. The A7 and A8 demonstrators could be examples of PEID:4 cases. Still, PEID:4 is to be considered as the "spare" group where PEIDs that, for some reason, do not fit into any other group will end up. Therefore it is also likely that the PEID:4 group will evolve in the future as PROMISE demonstrators are being implemented and non-PROMISE application scenarios are being considered