SAP ARCHITECTURE BLUEPRINT

VERSION: 1.1

DATE: JULY 3, 2008

AUTHORS:

Manfred Bergdolt

Srivathsan Govindarajan

Wolfgang Lindner

Dealer Business Management 7.0

PROJECT NAME/CPROJECT TITLE: PI07_DBM700_110161

SPONSOR/PROJECT INITIATOR: KERSTIN GEIGER / KLAUS WEBER

PROGRAM/PROJECT LEAD: GERALD HOPPE / STEPHAN BANTLIN

LEAD ARCHITECT: SEBASTIAN SPECK

DEVELOPMENT: ☑ SAPLabs, mainly in Hallbergmoos / Budapest

☐ Partner/ISV ____________

Document History

Version	Date	Status (Comments)
0.1	06.11.2007	Document created
0.9	18.04.2008	Draft Version
1.0	24.04.2008	Final Version
1.1	13.05.2008	After Development Project Review
1.2	18.06.2008	After final review

I MARKET AND PRODUCT BACKGROUND OF PROJECT/PROGRAM

Planned release date:	Q4 / 2009	
Underlying SAP NetWeaver release:	7.2	
Used SAP NetWeaver stacks:	☑ ABAP	☐ J2EE/Java EE 5

Use cases targeted by the project/program:

SAP Dealer Business Management (DBM) enhances the portfolio of SAP for Automotive into a new segment. The solution is based on SAP ERP, and our goal is to offer a complete and comprehensive management of all core business scenarios for dealerships, national and international dealer groups, as well as OEM owned sales and service companies.

SAP is the only vendor of a DMS package which is based on a world class ERP offering, and integrates dealer-specific processes such as customer management, vehicle service, vehicle sales & administration, parts management & sales as well as comprehensive HR and FI/CO in real-time. Moreover, SAP Dealer Business Management is fully scalable and handles the requirements of multi-brand as well as multi-location dealer businesses. It accommodates the needs of a dynamically changing business environment globally.

Target Market

- Passenger Car Segment
- Bigger Dealer groups, Integrated Importer & Dealers
- Markets & Brands less complexity in Interfaces and brands layers

Strategic goals SAP wants to achieve with the project/program:

- Mass market readiness
- Establish DBM as a coherent platform for partners to configure and build upon

Mandatory software capabilities to address goals, use cases, and target market:

- Flexible, adaptable and extensible platform as a coherent solution for a dealer's business
- Easy-to-use
- Low TCO
    - Easy-to-implement
    - Easy-to-maintain
    - High performance
- Multi-site, multi-brand, multi-company

II. ARCHITECTURE

This architecture blueprint reflects the architecture of SAP Dealer Business Management 7.0. DBM is a non-modifying Add-On product based on ERP DIMP 6.0 Enhancement Package 5 and subsequent.

Architecture Goals for SAP DBM 7.0

The Architecture of DBM 7.0

- Shall strengthen the approach of DBM as a platform for dealer processes, with inbuilt adaptability to customer needs, open integration ability to ISV components, using enterprise SOA where applicable ;
- Shall focus on reducing the overall costs for SAP and TCO for customer ;
- Shall reinforce the demarcation between and encapsulation of software components;
- Shall enable a user centric design ;
- Shall further align with SAP strategy and guidelines.

Main Architecture Characteristics of DBM

DBM was developed as a classic ERP based Industry Add-On and implementing an open architecture in a pragmatic way. It does not require any additional system installations as any additional applications like CRM did not seem to be sellable and therefore does not integrate different applications (SAP or non-SAP). It is tightly coupled to ERP (> 9 000 DBM objects uses more than 1500 ERP objects) and does not support any other backend systems. It uses backward integration to ensure consistency with ERP Business Objects and does minor enhancements to the backend like new fields.

The main objective of DBM development is the implementation of Business Objects and the integration of these with different modules of ERP. This is achieved via a central and generic object control. The focus is clearly on reuse rather than enhancing core ERP Business Objects by new functionality.

It is a highly configurable, extensible and malleable solution.

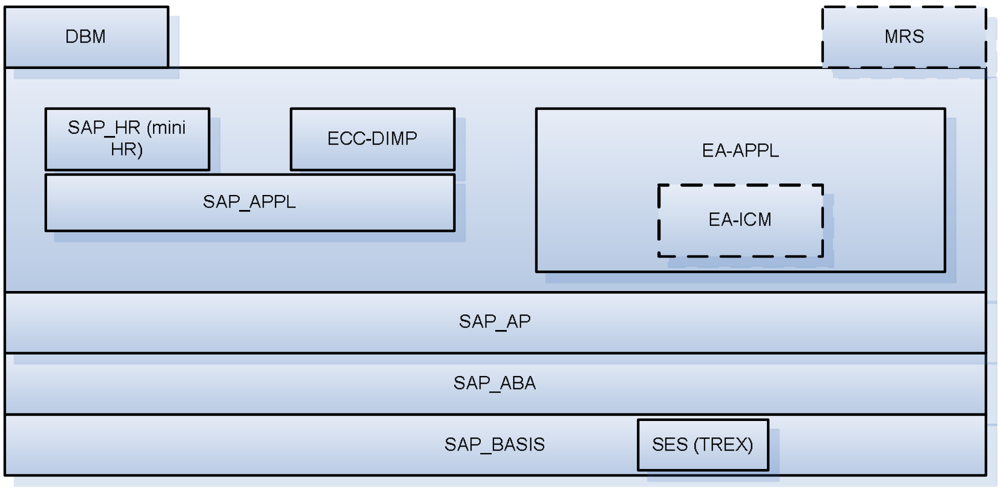

Figure 1

Figure 2 - DBM Software Stack

Main Architecture Layers

The main architectural challenge for DBM 7.0 is to implement a clear layering of the overall application and to draw clear boundaries between the different software layers. The different layers are described in the following sections.

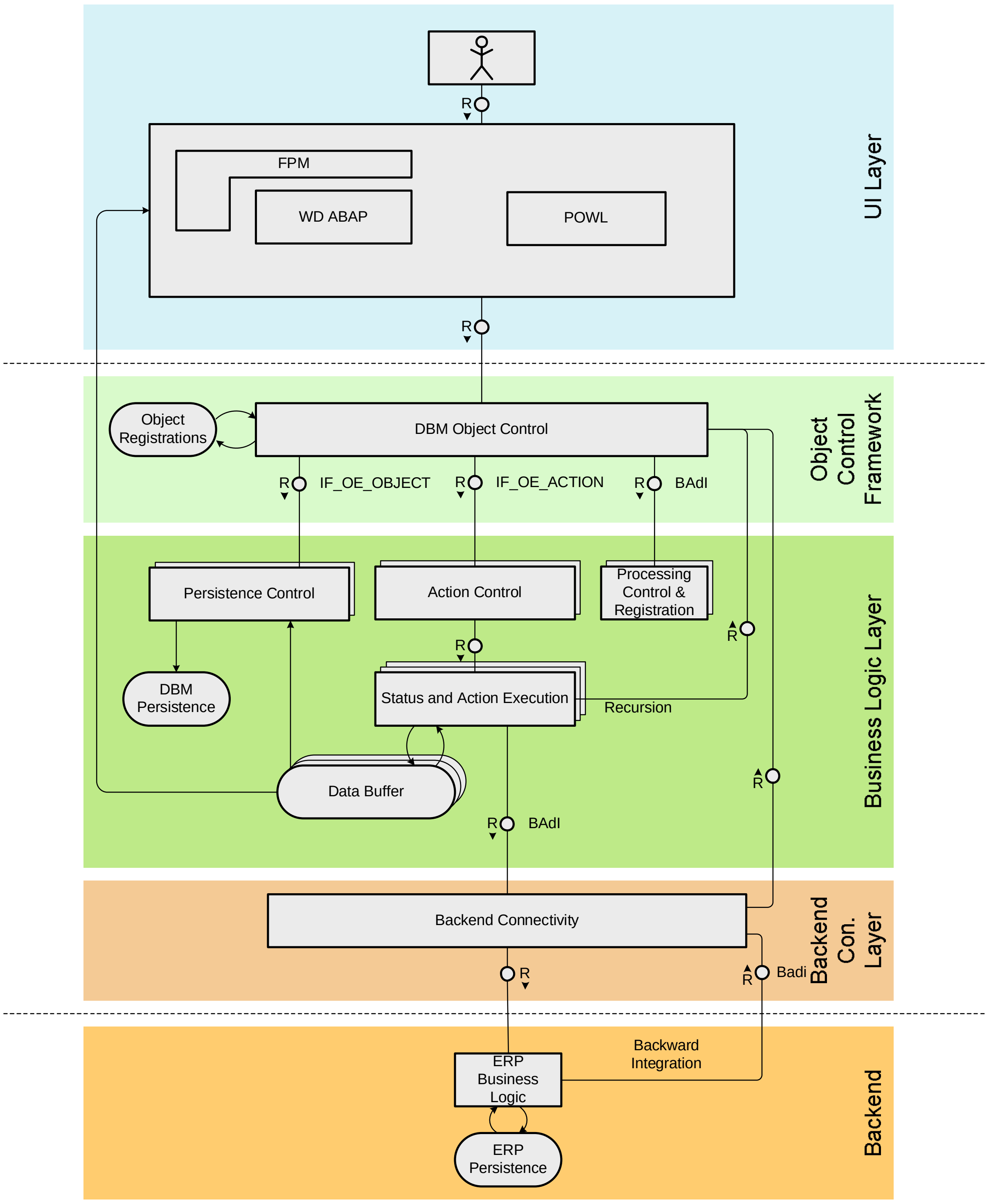

Figure 3 – DBM Layers

User Interface

The User Interface for the central roles Service Advisor, Parts Advisor and Workshop Supervisor (German: Werkstattmeister) shall be redesigned using the UCD process. The new User-Centric UI will be implemented with Web Dynpro for ABAP and Floor Plan Manager and Power Lists (POWL). The NetWeaver Business Client shall be the preferred UI client for these roles, but all applications must run in the portal as well. The Order, Vehicle Master, Storage Goods Manager and Customer Master transactions are expected to be redesigned. Work centers will be developed for the individual roles.

All roles will be delivered as PCD roles. If necessary, PFCG roles will also be delivered depending on the front-end client.

Beside this the classic SAPGUI will remain for other roles, master data and customizing. More information can be found in the DBM Wiki.

Dependencies to other layers

All business functions shall be executed via the framework and will be encapsulated. Therefore a clear separation of Business and UI logic shall be enforced. The framework will also provide information to the calling application (UI) regarding, which process steps are executable at a given point in time.

The UI can directly read all business object data and display them on the UI, but it cannot change it directly.

Additional UI services are required for certain purposes (e.g. valid field entries, value proposals). Therefore special UI service classes belonging to the Business Logic Layer can be called. The use access shall be controlled via a special package interface.

UI functions cannot be called from any other layer.

Object Control Framework

The Object Control Framework is designed for interacting with individual Business Objects.

- Competing frameworks in the predecessor release increased (and would even more increase) the complexity of functional implementations. The Order Engine Framework that implements all Order Business Objects and all Order related processes shall be enhanced to be used for other Business Objects as well. It is implemented with a strong separation of framework, processing control, Business Object implementation, business process implementation and status handling. All parts except the framework itself are replaceable by kernel-based BAdIs or an ABAP Interface.

The framework operates as generic service provider of all business object functionality and implements the steps

- Preparation
- Auto Save in case an ERP function requires data on the database
- Call the determination of object references and actions (BAdI)
- Pre-check if all object actions can be processed (e.g. authorization)
- Call to Business Object Action Processing by using the object reference determined before
- Transaction Control and processing of subsequent events.

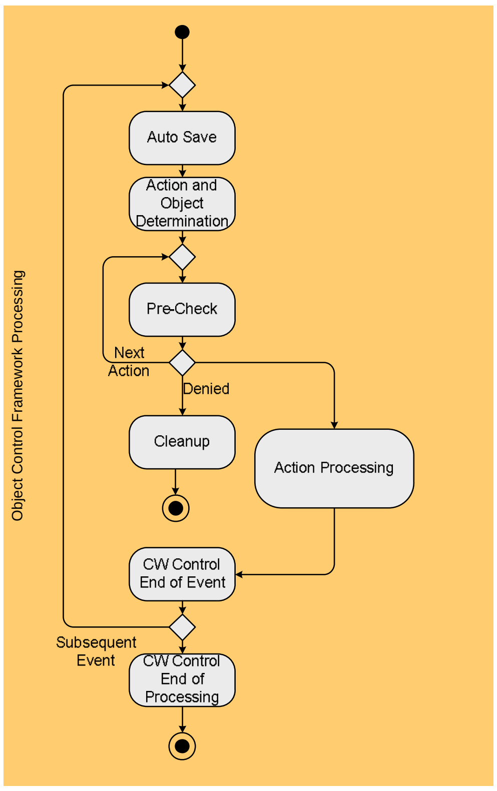

Figure 4 - Object Control Flow

The framework provides:

- Central transaction control including different commit levels to support the update of multiple business objects in one step. Even if a Business Object cannot handle multiple documents in one Logical Unit of Work (LUW) or depends on database updates of other business objects, the transaction control together with the status management ensures consistency. It offers the following steps:
    - Load
    - Save
        - Consistency Check
        - Posting
        - Synchronization after Save
    - Exit
- Consistency Control of object buffers by cloning of class instances and rolling back to the state of the objects to the start of the current logical processing step (event) or the last database commit.
- APIs for all business functions
- Different services like application log (BAL) or logbook

The framework should be used to implement and control all business objects that have the characteristics of transactional data. The usage for DBM master data that are normally not changed as a part of the process (original parts master, labor value, service packages and vehicle model) are not in focus.

There are two kinds of usages for the Object Control Framework. The first usage is to implement business objects and leverage the complete functionality of the framework as an OCF object. These objects do not necessarily require the existence of any other object, but have a flexible action control. Additional Extension Objects can be implemented that only requires the basic operations (create, change, delete) but no action control. These Extension Objects are compulsorily associated to an OCF Object.

Every OCF Object needs to implement the BAdIs of one enhancement spot to register the object in the framework and control the processing of actions. The action control can be implemented by using an OCF interface. All OCF Objects and Extension Objects need to implement OCF interface for persistence control.

For more information please refer to the DBM Wiki.

Dependencies to other layers

The Object Control Framework can be called by all DBM layers, but the framework cannot access any other layers. The framework can only call its own interfaces and BAdIs.

Business Logic Layer

All DBM business functions are implemented in this layer.

The OCF provides an interface for individual business objects to implement their processes and functionality.

The Business Logic Layer includes:

- Persistence of the DBM Business Objects
- Implementation of the persistence control for OCF Objects and Extension Objects
- Action and status control or integration to backend action and status control. Actions are executed for OCF objects and represent its services.
- Implementation of processing control for OCF objects
- Implementation of actions for OCF objects

Dependencies to other layers

The BAdIs for the Backend Connectivity Layer are defined and called in the Business Logic Layer. The Business Logic calls the framework to implement recursive calls for object control. The recursion depth is tracked by the Object Control Framework.

During recursive calls the Object Control Framework's transaction control ensures consistency with respect to persistence and buffer.

Back end Connectivity Layer

The back end Connectivity Layer consists only of BAdI Implementations that encapsulate the back end calls. The BAdI definitions belong to the Business Logic Layer. It enforces the demarcation of software components by package interfaces and use access.

The definition of the Connectivity Layer delivers additionally the interfaces that would be required for a decoupling of DBM. This demarcation of DBM and back end supports the maintenance process.

Having a clearly defined 'Backend Connectivity Layer', wherein all communication between DBM and the back end is defined and implemented, is easier for maintenance of the solution in the long term.

A back end call can be stateless or stateful. A stateless call maintains no state information between calls, whereas stateful calls keep state information in main memory across multiple invocations. The interaction with the back end is based mainly on synchronous communication.

Calls to the backend will be made via BAPI calls and RFC function modules. Enterprise Services will not be used for this purpose.

Back end

Historically DBM was and further evolved into an Add-On that is tightly coupled to the ERP and DIMP objects and business logic. Therefore it is not in focus to decouple DBM from the back end and converting it into an ABAP Composite.

As DBM is an Add-On, the transaction control of the framework also controls the update of all backend business objects. As described in note 921445 the consistency of back end data is not ensured against loss or damage. As DBM is an Add-On it allows implementation of BAdIs of backend business objects that call APIs of the Object Control Framework. With these back end enhancements it is possible to ensure consistency between backend and DBM even in non-DBM transactions.

The implementations are in a separate package for backward integration that has access to the Business Logic Layer and the Object Control Framework Layer.

Package Concept

The demarcation between different software layers is ensured by assigning the objects to appropriate packages. A more detailed package concept can be found in the Wiki.

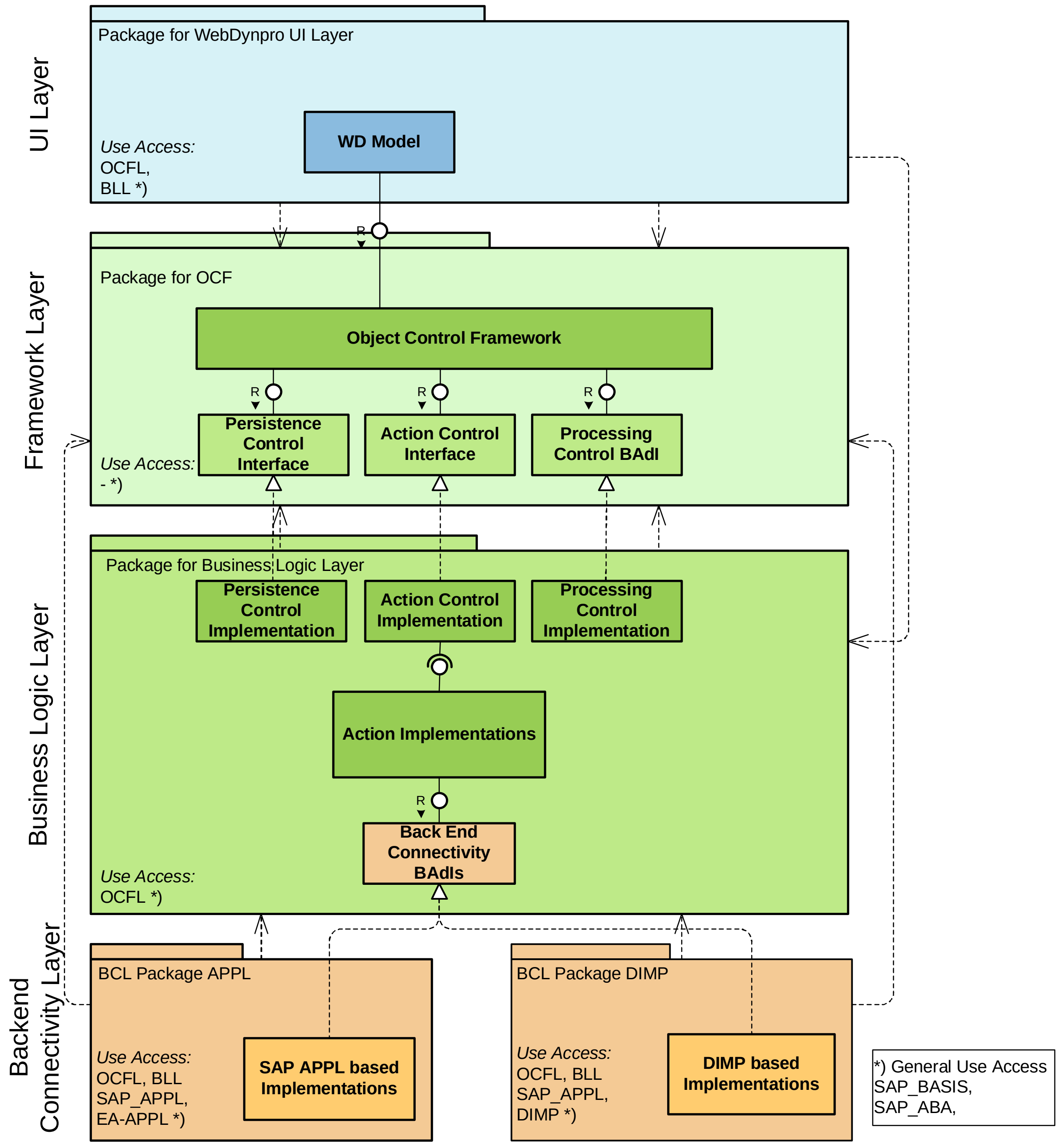
UI Layer
Package for WebDynpro UI Layer
WD Model
Use Access: OCFL, BLL *)
R
Framework Layer
Package for OCF
Object Control Framework
R
R
R
Persistence Control Interface
Action Control Interface
Processing Control BAdI
Use Access: - *)
Business Logic Layer
Package for Business Logic Layer
Persistence Control Implementation
Action Control Implementation
Processing Control Implementation
Action Implementations
R
Back End Connectivity BAdIs
Use Access: OCFL *)
Backend Connectivity Layer
BCL Package APPL
Use Access: OCFL, BLL SAP_APPL, EA-APPL *)
SAP APPL based Implementations
BCL Package DIMP
Use Access: OCFL, BLL SAP_APPL, DIMP *)
DIMP based Implementations
*) General Use Access SAP_BASIS, SAP_ABA,

Figure 5 – Package Concept for DBM Layers

The connection to MRS (or any other optional backend component in the future) will be connected similar to the packages SAP_APPL or DIMP above.

The DDIC objects are part of the individual components, the usage control will be managed via package interfaces (e.g. the UI Layer will access the Business Logic Layer only via a certain DDIC package interface.

DBM Business Objects

The term Business Object as referred to in this document is a definition based on the design of the application, wherein core objects are identified as a representation of a self-contained business entity. These are not modeled objects or objects in an Enterprise Services Repository.

The following provides an overview of the DBM Business Objects:

- The **DBM Order** is the central order object in DBM, which used the Order Engine (predecessor of the Object Control Framework) for executing the processes related to the business object. It has a hierarchical status concept and phased action management (using sorted multi-use kernel based BAdIs):
    - Check possibility of execution
    - Check to skip action depending on status
    - Execution
    - Post processing (e.g. status update, document flow)

  The choice and selection of the requested nodes of Order (Header, Job, Task, Item and Split Header) are done in the Processing Control of the Order Engine.
- The **DBM Vehicle** consists of the DIMP application Vehicle Management System (VMS) and the Individual Object (IObject). Even though it uses the Vehicle Management System, it is not completely compatible with VMS, i.e. with the current architecture it is not possible to leverage all functionalities of both vehicle solutions for the same individual vehicle. More information can be found in the Wiki.
  As DBM is still an Add-On component and the Vehicle Management System does not support the required functionality, the existing DBM Vehicle Object shall henceforth use the new Object Control Framework. This will enable decoupling from VMS, offers the possibility to replace the VMS framework by the DBM Object Control and opens up the possibility to use equipments or one-time vehicles.
- **Storage Goods** are objects that are stored at a dealership but owned by the customer. The placement or removal from storage is mostly related to other main processes like Service or Vehicle Sales. Therefore it is essential that all functionalities are available via API and can be integrated in the Object Control.
  Even though the implementation is based on IObjects, it cannot be enhanced to new storage types or functions due to limitations in the implementation. With the use of the Object Control Framework for Storage Goods, these limitations can be overcome.
- **Other Objects**(Only for completeness' sake): Labor Value, Original Parts Master, Service Package, Vehicle Model

ERP Business Objects

Notifications are used for sales activities and customer complaints and related to other main processes like Service or Vehicle Sales. Therefore notifications shall be integrated in the Object Control Framework.

In it envisaged that other Business Objects (Business Partner, Customer Master, Vendor Master, Employee, Warranty Claim, Recall, SD Billing, Manual Reservation, Purchase Requisition, Material Document, Internal Order, Activity Allocation, Primary Costs, Time Sheet, Asset, Down Payment, CO Production Order, Cash Transactions, Commission Case) are only integrated as Extension Objects.

Total Cost of Ownership

DBM targets both the SME segment as well as bigger dealer groups (Large Enterprises) and therefore must fulfill both their TCO requirements.

Deployment

The only deployment target of DBM 7.0 is to be deployed as a pure ABAP Add-On on top of ERP 6.0 (Enhancement Package 05 and following) with activated Business Function Set DIMP. The Business Function IS_AD_MPN (A&D Manufacturer Part Number) is optional. It has no requirement of the Java stack.

TREX will be used as the search technology for selected business objects like the DBM Order and Vehicle. TREX will be an optional component, and the application will carry out a normal search in the absence of TREX.

It shall be possible to deploy it on the same instance as the Adaptable Customer Solution MRS (Multi-Resource-Scheduling). It will be possible to install additional Partner Add-Ons (e.g. Time Recording) on the same instance.

The preferred deployment scenario is an All-in-One solution by partners, wherein interfaces, brand and country specifics are added.

Architecture Documentation

All Architecture Documents including this can be found in the DBM Wiki.

External Interfaces

DBM will offer interfaces as required in the product standards as RFC's, eSOA is not in focus. ISV partners or customer projects will implement OEM interfaces based on our delivered RFC's.

Glossary

Term	**Definition**
Object Control	DBM framework that centrally implements and controls the business objects, by communicating with the action control, maintaining data consistency and handling of transaction control.
Action Control	The 'Action Control' controls the execution of the individual actions for the business objects.
Processing Control	The 'Process Controller' identifies and prepares the individual business objects for a certain process step.
Status and Action Execution	A Processing block that executes the individual action on a business object, tightly bound together with a flexible status concept.
Extension	Objects that are related to the main business object and cannot exist

Objects	independently as a separate business object in the current context.

III. OPEN ISSUES, OUTLOOK AND RISKS

Open Issues

- Different scenarios for the Solution Architecture need to be investigated and considered from the perspective of TCO. The long term architecture roadmap is not defined.

Risks

- New requirements because of new deployment scenarios (SaaS) coming from a global partnership (GLAD).
- Delay in and misalignment of different work streams in development might cause rework of already developed functionality and might have an impact on the degree of realization of the described architecture.
- Application components exist in DBM which require reengineering. Currently this is not in scope due to the identification of such applications as ISV components. These components could also be identified and expected as a core DBM offering.
- The postponed retrofit of DBM into an ERP enhancement package for a latter release increases overall effort, TCO and does not provide additional functionality for other ERP customers. No hard facts as of today are available that contradicts this approach.

Outlook

- Currently it is decided that the following release to DBM 7.0 will be retrofitted into an enhancement package of ERP.
- A future transformation of DBM into an ABAP composite that can also be connected to AP is not recommended from architectural and effort point of view. Nevertheless the provided architecture would support a ‘Composition by Evolution’ process.
- A Business Suite offering of DBM as ERP component together with CRM and additional Composites could strengthen the DBM solution for large dealer groups or other customer segments.
- There is a prospect of an exclusive global partnership (> 10 000 SAP customers) for providing DBM in a Software as a Service model. Therefore a new development of a new composite application on top of AP could be envisioned.