Editor:
Jochen Boeder
Office of the CTO
Architecture Excellence

Authors:
Goyal, Ashwani
T, Kalaimani

SAP Architecture Bluebook

SAP SRM 6.0

Version 1.0

September 2007

Acknowledgement

The following colleagues contributed in various ways to make this or previous editions of the document possible: Ramakrishna Potluri, Padma Prasad Munirathinam, Hari Krishna L, Santosh Kumar Addanki, Andreas Stolz, Reinhard Sudmeier, Oliver Hilss, Erik Dick, Patrick Karcher, Peter Neumayer

Table of Contents

Table of Figures

1 Architecture Summary

SAP Supplier Relationship Management (SAP SRM) is SAP software solution to support modern corporate or institutional buying by automating procurement processes and collaborating online with suppliers. SAP SRM embraces multiple applications. The core is the SAP SRM server, which runs on the ABAP stack of SAP NetWeaver Application Server. In addition SAP SRM includes the stand-alone applications supplier self services, catalog management, internet pricing and configurator and live auction cockpit.

The SRM server application is divided into the following architecture layers:

- Database layer which persists the business and configuration data
- Procurement document layer (PD layer) which contains the core business logic
- Procurement document objects layer (PDO layer) which provides an object-oriented access layer to the business logic.
- Channel logic layer which prepares the business object data for multiple output channels, for example user interface or composite applications

The user interface is implemented in Web Dynpro ABAP and presented using SAP NetWeaver Portal.

2 Introduction of SAP SRM

Supplier relationship management (SRM) is a term which describes all processes, policies, and procedures to support modern corporate or institutional buying. SAP Supplier Relationship Management (SAP SRM) is a software application which automates procurement processes and collaboration with suppliers. It supports both planned and ad hoc procurement.

SAP SRM is one application of SAP Business Suite, which embraces in addition SAP Enterprise Resource Planning (SAP ERP), SAP Customer Relationship Management (SAP CRM), SAP Supply Chain Management (SAP SCM), and SAP Product Lifecycle Management (SAP PLM). These applications are integrated with each other to support end-to-end business processes.

2.1 Supported Business Processes

SAP SRM helps to streamline the procurement process by empowering employees to create their own requisitions, and automating the purchase order approval and generation process. Purchases can be restricted to approved vendors and business rules can be established to enforce individual and departmental purchasing limits. Special requests and one-time purchases can be handled online, while connectivity with e-marketplaces allows companies to find new suppliers and obtain more competitive prices.

Overall SAP SRM includes the following main functions:

- Sourcing:
  - Category management
  - Supplier qualification
  - Supplier negotiation
  - Contract management
- Procurement
  - Requisitioning
  - Order management
  - Receiving
  - Financial settlement
- Supplier enablement
  - Document exchange
  - Supplier network support
  - Supplier portal management
  - Supplier collaboration

For supplier enablement SAP SRM includes Supplier Self Services (SUS) as a stand-alone application. Further information on the business process supported by SAP SRM can be found in [Set07].

2.2 Focus of Document

This document describes SAP SRM release 6.0, which is based on SAP NetWeaver 7.0 (formerly known as SAP NetWeaver 2004s). With SAP SRM 6.0 the architecture has been changed to a large extent to migrate the existing ITS-based user interface to Web Dynpro ABAP and include a new workflow concept (see chapter 5).

2.3 Architecture Overview

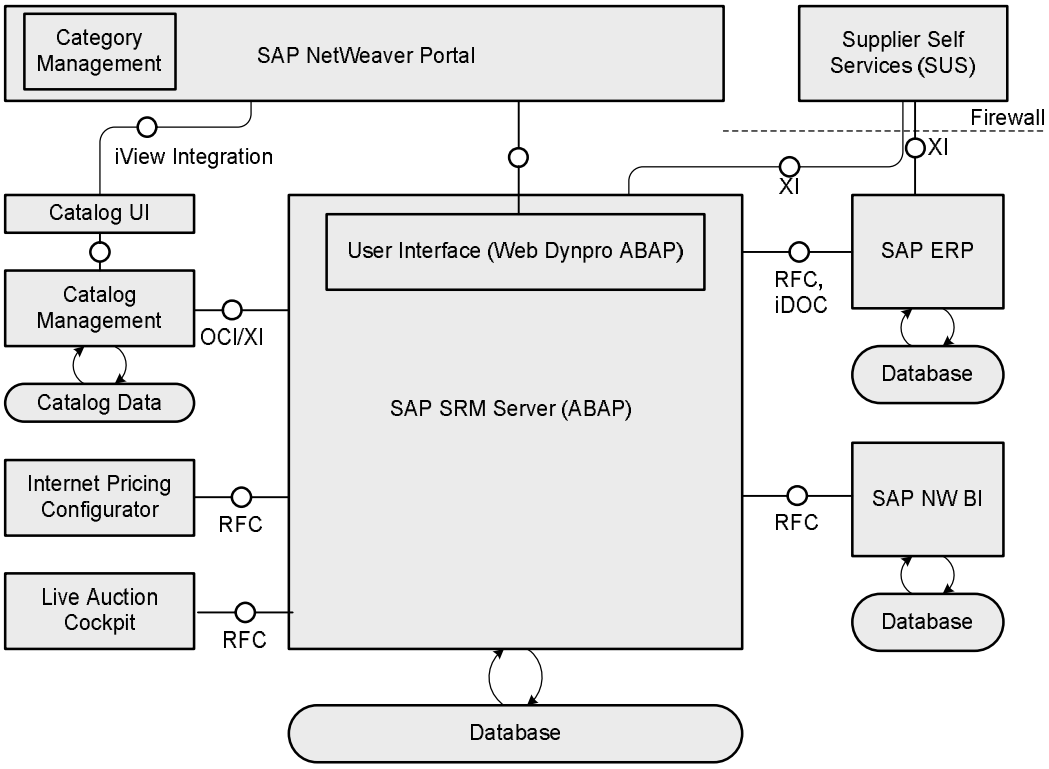

Figure 2-1 Architecture Overview

The complete SAP SRM 6.0 solution embraces multiple applications (see figure 2-1). The core is the SAP SRM server, which runs on the ABAP stack of SAP NetWeaver Application Server and utilizes Web Dynpro ABAP for user interface rendering. SAP SRM has its own database. In addition the following stand-alone applications are part of the SAP SRM solution (see figure 2-1):

- **SAP NetWeaver Portal** provides role-specific access to the SAP SRM user interface. SAP SRM uses the Knowledge Management (KM) component of the portal to store unstructured data, for example document attachments of business objects. The portal also integrates cRoom.
- **Catalog management** is a standalone application, which provides a catalog of products for purchasing. It is also used for product master data management. As default SAP SRM uses the new MDM catalog for catalog management instead of the CCM catalog application, which has been used in former releases. As the communication between SAP SRM and catalog management takes place via the standardized open catalog interface (OCI) any compliant catalog can be integrated with SAP SRM.
- **Supplier Self Service (SUS)** is an application for supplier collaborations. The user interface is implemented using Business Server Pages (BSP). SUS has its own database and communicates with SAP SRM and SAP ERP using SAP NetWeaver Exchange Infrastructure (SAP NetWeaver XI). SUS can be installed on the SAP SRM server or on a separate server. Typically it is operated on a separate server to protect SAP SRM server by an extra firewall from attacks from the Web (see figure 2-1).
- The **Internet Pricing and Configurator (SAP IPC)** is a stand-alone Java application which is used by SAP SRM for determining prices of materials and services.

- For analytics and reporting, business data is replicated to **SAP NetWeaver Business Intelligence (SAP NetWeaver BI)**
- **Live Auction Cockpit** is a Java application which allows conducting online auctions in real time with many vendors participating.
- **Category management** is a Java application which allows the user to manage categories (commodity groups) and their spends in SRM.

The various applications are mainly integrated using RFC. SAP NetWeaver Exchange Infrastructure (SAP NetWeaver XI) is used for communicating asynchronously between SAP SRM server and SUS in order to decouple both applications. SUS is also integrated with SAP ERP materials management (SAP ERP MM) using SAP NetWeaver XI. In addition SAP NetWeaver XI is used to upload contract data to catalog management.

2.3.1 Integration with SAP Business Suite

SAP SRM can be integrated with the other applications of SAP Business Suite to support end-to-end business processes. SAP SRM uses SAP ERP as backend for financial checks and postings, as depicted in figure 2-1. SAP SRM communicates with different releases of SAP ERP using meta BAPI (see 3.3.1.4). The communication is based on RFC. CRM middleware is used to synchronize master data between SAP ERP and SAP SRM.

The SAP SCM application advanced planner and optimizer (APO) is used for supply chain planning. Resulting demand for material and services can be posted to SAP SRM via SAP ERP for further processing of the purchase requisition.

SAP CRM is integrated with SAP SRM using SAP NetWeaver XI within the leasing and asset management scenario.

SAP SRM can be integrated with cProjects Suite (cProjects, cFolders) of SAP PLM to enable purchasing goods and services directly from project management applications.

2.3.2 Architecture Layers of SAP SRM Server

The architecture of SAP SRM 6.0 separates the SRM server application into the following layers (see figure 2-2):

- Database layer which persists the business and configuration data
- Procurement document layer (PD layer) which contains the core business logic
- Procurement document objects layer (PDO layer) which provides an object-oriented access layer to the business logic.
- Channel logic layer which prepares the business object data for multiple output channels, for example user interface or composite applications

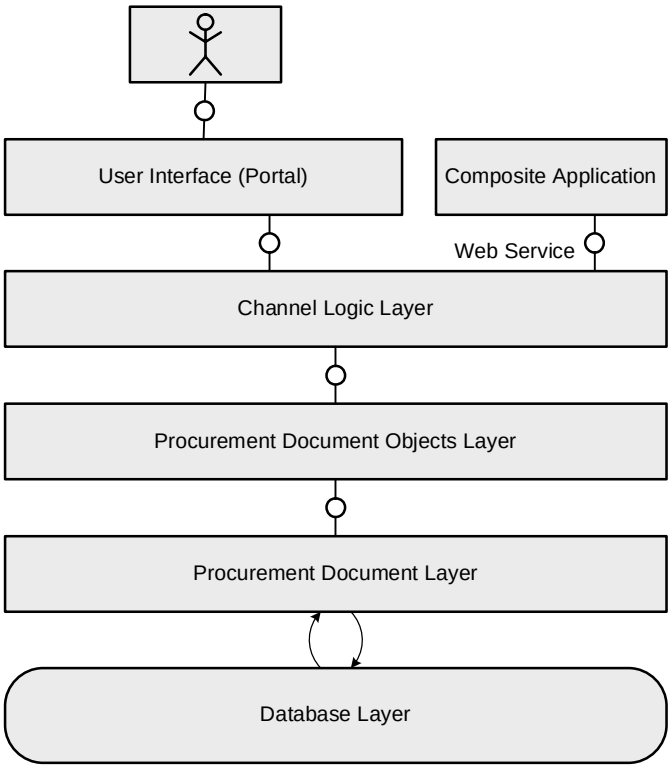

Figure 2-2 Layers of SAP SRM 6.0

Due to the layering approach each output channel, in especially the user interface is clearly separated from business logic. Additional output channels can be added or existing output channels can be changed without affecting the business logic. With the procurement document objects layer an object oriented wrapper of the business logic has been introduced. It provides business objects which can be accessed from the channel layer using standardized methods, for example to create, retrieve, and update business object data. With this, SAP SRM evolutes towards a service-oriented architecture as the business logic is decoupled from its consumers using standardized interfaces. This approach is similar to the business object and core service approach of the AP/A1S architecture, which eases a potential migration.

3 Architecture of SAP SRM 6.0

Figure 2-2 shows the architecture layers of SAP SRM and their main components. In the following each layer is described in more detail.

3.1 Channel Logic Layer

The channel logic layer prepares business object data from the PDO layer for multiple output channels. Currently the channel logic layer supports the output channels Web Dynpro and enterprise SOA (see figure 3-1).

3.1.1 Web Dynpro Channel

The user interface of SAP SRM 6.0 follows a pattern-based approach which is implemented using Web Dynpro ABAP. Floor plans define the arrangement of the different UI elements on the screen and the basic layout. Each application UI is composed from multiple Web Dynpro components according to one floor plan and the specific configuration. This ensures a homogenous and consistent user interface and allows reusing Web Dynpro components. Currently the following floor plans are used within SAP SRM 6.0:

- Object instance floor plan (OIF) which enables user to view and edit a business object.
- Guided activity floor plan (GAF) which provides step-by-step wizard to accomplish a task sequence.
- Application monitors floor plan which displays error messages from application monitors.

At runtime the application UI is created by the UI framework (see figure 3-1), which retrieves required business object and meta data from PDO layer using object mappers. An object mapper controls the user interface for a business object. Object mappers map data between user interface and business objects, update the user interface, and trigger and handle UI actions.

The meta data defines the properties of the UI fields, such as mandatory and read-only. When the application UI is instantiated, the UI framework enables the communication between floor plan and Web Dynpro components. The framework provides actions for refreshing and uupdating data on the screen which can be triggered by the object mappers.

Besides meta data management the UI framework provides the following supporting functionality (see figure 3-1):

- Event handling: events triggered from the user interface, for example triggering a button or selecting a tab, are handeled by the UI framework which updates table buffers and user interface data.
- Error and exception handling: the UI framework uses the Web Dynpro message manager encapsulated by a SRM-specific wrapper as message handler. Error messages are displayed in a predefined area of the floor plan. Exception messages are displayed in a pop-up.

Both are controlled by the object mappers.

3.1.2 Enterprise Service Channel

The enterprise SOA channel is used to provide enterprise services. These enterprise services are Web services which can be used to build composite applications on top of SAP SRM or to integrate SAP SRM with external systems. Within the enterprise SOA channel for each Web service a proxy is generated based on the meta data defined in enterprise services repository.

The proxy calls the referring object mappers to map the service parameters to the business object or dependent object structure. The mapper update the business object instances of PDO layer directly by calling the corresponding methods.

For error and message handling the enterprise SOA channel includes a log handler, which collects all messages from PDO and PD layer and provides them for the service consumer.

For further details on service enablement of SAP Business Suite applications see [SAP06].

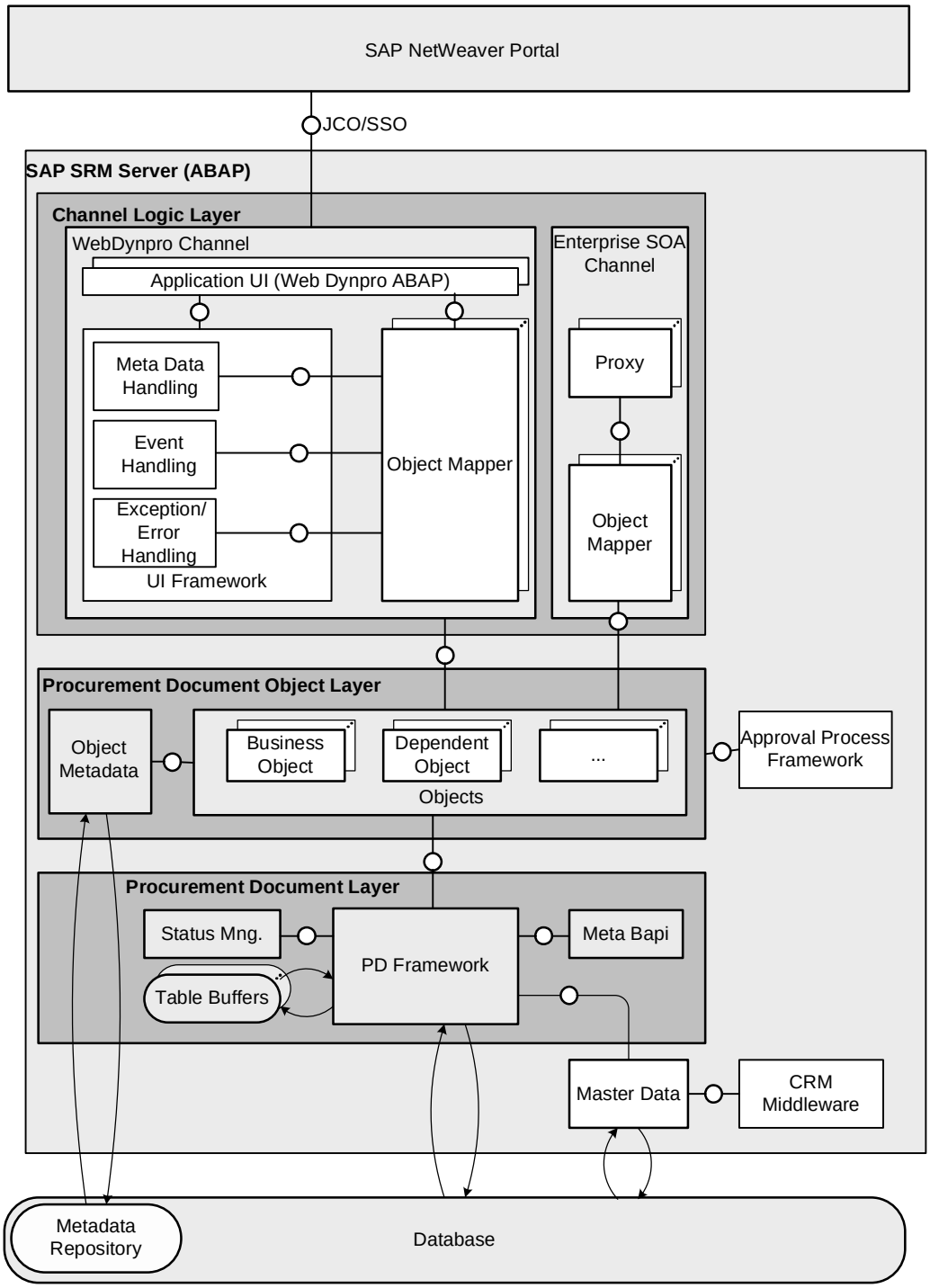

Figure 3-1 Architecture of SAP SRM Server

3.2 Procurement Document Object Layer

The main purpose of PDO layer is to give different output and input channels a uniform access to business functionality independent of the presentation layer. To do so, the business logic contained in the PD layer is encapsulated in a homogenous way using business objects and supporting objects (see section 3.1.2). Each business object is implemented as one ABAP OO class. The business objects provide a standard set of methods for creating, retrieving and updating business object data to the channel logic layer. The business objects are implemented independent of specific output mediums, such as user interface, message-based integration, or MS Office integration.

Business objects take care about authorizations and enqueues. They can access configurable meta data of fields and actions. Business objects are also integrated with the approval process framework of SAP SRM (see figure 3-1).

The introduction of the PDO layer was required to provide a standardized backend interface for the new user interface. It also absorbed business logic which was contained in the user interface or the application layer in previous SAP SRM releases.

3.2.1 Object Model of SAP SRM

The central object type of the SAP SRM object model are business objects. They represent entities with significance to the SRM business, for example purchase order, shopping cart, supplier, and material. In general a business object encapsulates business logic and data and is accessible only via its interface. In PDO layer each business object is a wrapper of a procurement document defined in PD framework (see section 3.3.1). The procurement document defines the data structure of the business object.

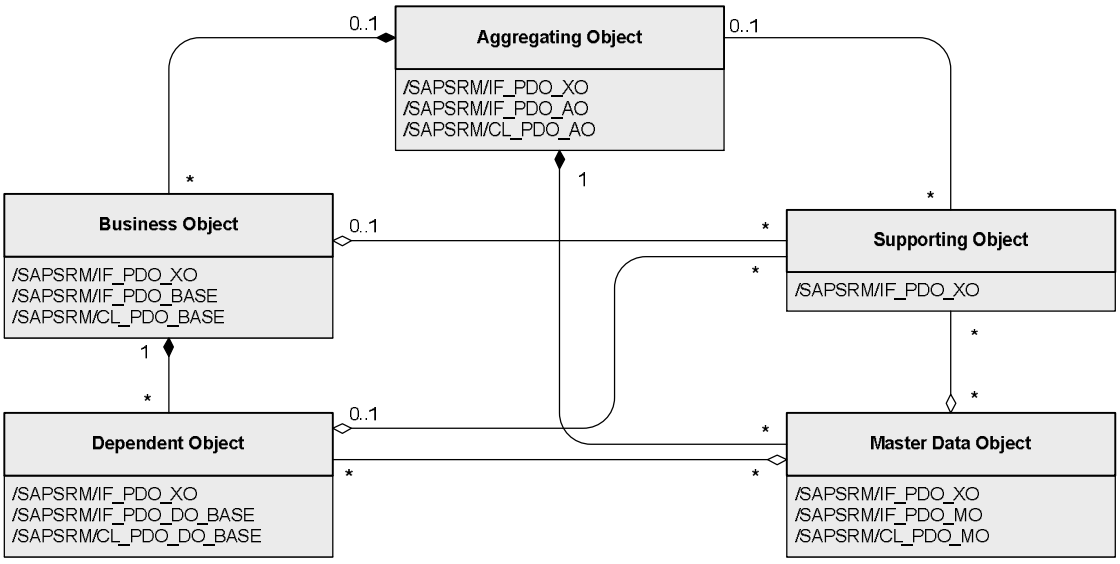

Figure 3-2 Object Types with Relationships

To facilitate reuse of functionality, additional object types are introduced, as there are aggregating object, dependent object, master data object and supporting object (see figure 3-2). Each object provides common methods for action and meta data handling, which are declared in the x-object interface (IF_PDO_XO). Examples of these common methods are handlers for framework events such as refresh user interface, update data, and retrieve meta data handler. Object type specific methods are inherited from a corresponding base class. In addition the base class provides implementations of certain common methods, such as check, get item list, save, and submit updates. Business objects inherit the interface IF_PDO_BASE

as well as implemented methods from their base class CL_PDO_BASE; dependent objects inherit IF_PDO_DO_BASE as well as implemented methods from their base class CL_PDO_DO_BASE, and so on.

In the following the different object types are described in more detail.

3.2.1.1 Business Object

The data structure of a business object is defined by the DDIC structure of the corresponding procurement document. Each procurement document is structured in sets, which embrace semantically related data (see figure 3-3). Common to all business objects are the header and the item set. Each of them embraces further sets. At PDO layer the business object itself controls the data of the header set. The data of all other sets are handled by dependent objects (see section 3.2.1.2).

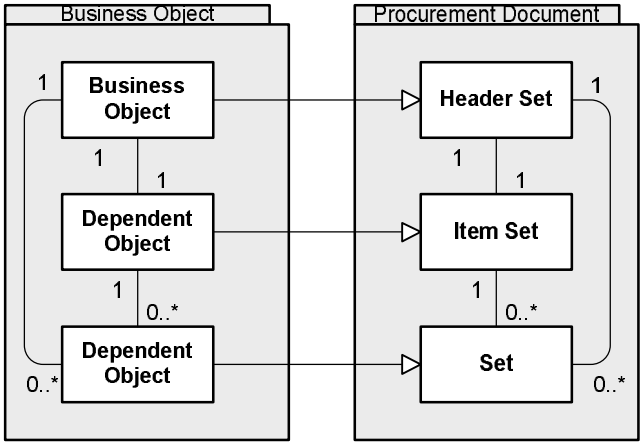

Figure 3-3 Relationship of Business Object and Procurement Document

Each business object is implemented by an ABAP OO class. A single base class is provided, which defines a common set of methods for retrieving, updating, and deleting business object data. All business object classes inherit the corresponding interfaces from the base class.

Each business object has a corresponding application UI which is controlled by its object mapper.

3.2.1.2 Dependent Object

Dependent objects control a subset of semantically related business object data, for example purchase order item or purchaser. So dependent objects do not exist on their own, but are assigned to at least one business object.

Dependent objects are implemented as classes which are instantiated inside the constructors of business objects. The business object stores the reference to its dependent objects as class member variables. Dependent objects are persisted using PD framework. A specific base class for dependent objects defines common methods, for example get related actions, and get meta data handler.

Dependent objects can be reused by multiple business objects. For examples account assignment and “notes and attachments” are dependent objects, which are assigned to purchase order, confirmation, and invoice.

3.2.1.3 Supporting Object

Supporting objects are used to provide additional functionality for business objects within PDO layer. They have no persistency, but can have a user interface. For example cross catalog

search is a supporting object which provides a user interface and the corresponding functionality for searching across multiple catalogs. So it has its own object mapper within Web Dynpro channel to control its user interface. Workflow is an example for a supporting object without user interface. It controls the workflow status of business objects.

Each supporting object is implemented as ABAP OO class and can be instantiated independent of a business object.

3.2.1.4 Aggregating Object

Aggregating objects allow to display and edit multiple business objects within the same application UI. To do so, an aggregating object references multiple business objects, master data objects, and supporting objects. For example the aggregating object “contract mass changes” has references to all contracts it has to change in a mass change transaction. Other examples are sourcing cockpit and bid comparison.

3.2.1.5 Master Data Object

Master data objects control data which is changed rarely but read often, for example, employee, vendor, and purchasing organization. Subsets of attributes are structured in dependent objects, for example address. Master data objects are implemented as classes, too, which encapsulate underlying business logic. The implementation of employee and vendor is based on the business partner of SAP ABA. Product master data is maintained in SAP ERP and imported from there using CRM middleware (see chapter 3.3.2.2). Product is not defined as a separate master data object, but product data is included in business objects where needed.

3.2.2 Accessing Business Objects from the User Interface

Within Web Dynpro channel for each business object a corresponding object mapper exists, which controls the business object’s application UI. On request the object mapper instantiates the business object using the business object factory. Each business object itself instantiates referenced dependent objects.

3.3 Procurement Document Layer

Procurement documents represent SRM business documents such as shopping cart and purchase order within the SAP SRM system. Each business object of PDO layer is assigned to one procurement document. The core of the procurement document layer (PD layer) is the PD framework for managing and persisting procurement documents. It provides access to status and action management and master data. PD layer includes also meta BAPI for integration with SAP ERP backend (see figure 3-1).

3.3.1 Procurement Document Framework

The PD framework is an evolution of the one order framework of SAP CRM and is not implemented in an object oriented way. It provides a library of generic functionalities for creation, modification, and persistency of procurement documents, in detail:

- Create of a procurement document: a new instance is created and stored on the database. PD framework enriches the document with further data, such as date, product id for a product-GUID, and address data
- Check consistency and correctness of procurement documents before persisting them. In case the check fails, an error message is raised
- Provide interfaces to call further components, such as price determination and tax calculation

- Retrieve procurement document data from database,
- Buffer procurement document data in main memory,
- Store procurement document on the database
- Maintain multiple versions of procurement documents, in especially provide the possibility to save draft versions (see section 3.3.1.2)

In addition PD framework includes procurement document-specific business logic, for example specific checks for fields of a business document For setting and managing of status values in procurement documents it uses status management. The business objects and dependent objects of PDO layer call the corresponding interfaces of PD framework.

The strength of PD framework is the good support of document flows. SAP SRM is based on the following two standard document flows:

- Self service procurement: Shopping cart – purchase order – purchase order response – purchase order confirmation – invoice
- Plan-driven procurement: Purchase requisition – purchase order – purchase order response – purchase order confirmation – invoice

PD framework allows creating a subsequent procurement document from its predecessor easily.

3.3.1.1 Procurement Documents

A procurement document is structured in sets which combine semantically related attributes. The header set defines all attributes that are valid for the whole procurement document, such as the type of document, and the creation date. Item set defines subsets of data, which can exist multiple times within a document, such as positions. Header and item can embrace further sets. A set can contain data from other sources, such as master data or accounting data. Sets can be attached both to the header and the item. As an example the contract uses the partner set attached to the header to store the vendor for which the contract is valid as well as the partner set attached to the item to store the location the item is valid for.

Documents containing errors can be saved on the database (put on hold) without starting follow-up processes (approval, output).

3.3.1.2 Status Management

A status represents a milestone in the life cycle of a procurement document. Actions are triggered to change the value of a status. The status management allows defining status variables, status values and actions. At runtime when an action is triggered PD framework accesses the status management to evaluate if the status value of a procurement document can be changed. In addition to the status management evaluation, PD framework checks the consistency and completeness of the business object before actually changing the status value.

SAP SRM reuses the status management implemented by SAP CRM.

3.3.1.3 Buffering Data

PD framework keeps the state of procurement documents using four levels of temporary data stores. These buffers are the following (see figure 3-4):

- Y tables contain the original procurement document that was read from the database. Data stored in Y tables is read-only. Y tables are used to cache data and to compare updated data from X tables with original data.
- W table, which store the current state of the procurement documents during the transaction. They are working tables with unchecked data and include also some control data, which is not stored on the database. These tables act as a mapper between the T table and the application programming logic.

- T table: When save is triggered by the user, data is copied from W tables to T tables. Data in T tables is checked for consistency and correctness. If the checks are successful data is passed on to X tables for storing it on the database. If checks are not successful the procurement document can be stored with errors as save draft.
- X table: After successful check data is copied from T table to X table. Thereby it is compared with original instance of Y table to define the required database operations insert and update. At commit work procurement document from X tables are persisted to database using “call function in update task”.

In SAP SRM no procurement document is physically deleted from the database. Deleted procurement documents are only marked by a flag as deleted.

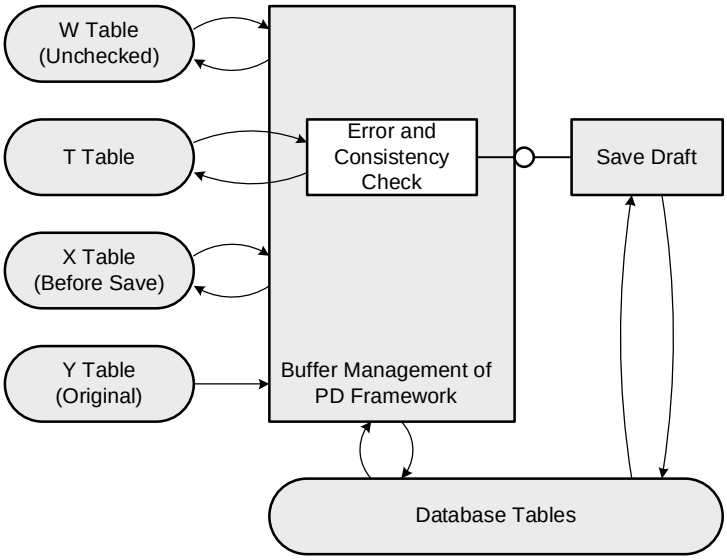

Figure 3-4 Buffer Layers of PD Framework

3.3.1.4 Organizational Management

The organizational model displays the functional structure of an enterprise. It is a hierarchical structure that is maintained in the SAP SRM system. Its elements include organizational units (for example, departments), user attributes (for example, country, cost center), and user master records. The organizational model defines who is allowed to buy which products. For this products and catalogs are assigned to the corresponding entities of the organizational model. Workflow rules utilize the organizational model to decide who receives a workflow item and who executes a task.

The organizational management functionality of SAP SRM reuses the SAP CRM implementation.

3.3.2 Integration to ERP

Between SAP SRM and SAP ERP the following integration scenarios are supported:

- In the classic scenario, shopping carts are created in SAP SRM, but the resulting purchase orders are created only in SAP ERP. In the classic scenario SAP SRM can be integrated with multiple SAP ERP systems.
- In the extended classic scenario, the complete procurement process takes place in SAP SRM. The documents are transferred to SAP ERP. However the leading documents reside in SAP SRM.

- In the decoupled scenario, the purchasing documents resulting from shopping carts are created either in SAP SRM or in SAP ERP in dependence of the product category (commodity group) of the ordered item. Customers can adapt the selection criteria using a Business Add In (BAdI).
- In the standalone scenario, the purchasing documents resulting from shopping carts are created only in SAP SRM system. They are not replicated to SAP ERP.

In all scenarios financial checks and postings are performed in SAP ERP Financials (FI). For communication with SAP ERP, SAP SRM uses meta BAPI. Master data is uploaded and synchronized using RFC and CRM middleware.

3.3.2.1 Meta BAPI

Meta BAPI enables backend integration of SAP SRM with an ERP backend (SAP ERP or third-party) using BAPI. As SAP SRM supports the integration with multiple releases of SAP ERP, also different interface versions of SAP ERP needs to be supported in parallel. To do so, meta BAPI stores mappings between SRM business objects and different BAPI versions of different SAP ERP releases (table BBP_FUNCTION_MAP).

At runtime the meta BAPI dispatcher is called from PD framework, when data has to be transferred to SAP ERP. According to the release and version of the connected SAP ERP system the meta BAPI dispatcher selects the right BAPI interface and mapping from the table. Meta BAPI maps the data to the corresponding interface and calls it via RFC.

Meta BAPI can be enhanced to support additional interfaces by using the corresponding BAdI.

3.3.2.2 Master Data Exchange

SAP SRM uses different techniques to import and synchronize master data with SAP ERP.

3.3.2.2.1 Product

SAP SRM uses integrated CRM middleware for product master data synchronization with SAP ERP. To do so, SAP ERP needs to install a corresponding plug-in. Two scenarios are supported:

- Initial download of product master data SAP ERP.
- Delta download: after initial load only changes of master data are downloaded from SAP ERP

In both scenarios SAP ERP is the leading system for master data maintenance. CRM middleware initiates the download of product categories (material groups), product subtypes (material types), and product masters (material masters and service masters). The data is transferred asynchronously via RFC.

Product master data cannot be maintained in SAP SRM.

3.3.2.2.2 Business Partners

Business partner data can either be imported from SAP ERP using RFC or can be created and maintained directly in the SAP SRM system. Business partner records cover persons as well as organizations. Additionally, they are categorized as internal or external business partners.

3.3.2.2.3 Configuration Data and Organization Data

Common configuration settings such as unit of measure can be imported from SAP ERP using RFC. The organizational model can be imported from SAP ERP HCM using iDOC interfaces. For regular synchronization of organizational data an ALE scenario exists.

4 Program Flow

To explain the runtime architecture in more detail, we describe the program flow for creating and updating a purchase order.

4.1 Requesting a User Interface

The user accesses SAP SRM using a Web browser. The user interface displays a portal page with various iViews provided by SAP NetWeaver Portal. The iViews can integrate user interfaces from different backend systems. In case of SAP SRM the iViews point to ABAP Web Dynpro applications.

When a user requests a user interface to create or update a purchase order, SAP NetWeaver Portal triggers the purchase order business object through object-based navigation which in turn launches the corresponding iView. The iViews act as containers to the underlying Web Dynpro applications. When launching a task in the work center a new browser window is opened which renders the assigned floor plan according to the related Web Dynpro application configuration? For the purchase order object, the object instance floor plan (OIF) application is invoked with the required parameters, in especially business object type purchaser order, and the application configuration for purchase order. If the user wants to create a new purchase order the URL contains the parameter transaction mode = create. If the user wants to display or edit an existing purchase order the URL includes the corresponding transaction mode (display or edit) and the GUID of the purchase order.

The UI framework instantiates the requested object instance floor plan (see figure 4-1). The floor plan defines the frame, which is filled by different UI components. One is the identification area component, which displays header information of the purchase order; the others are content area components, which display data of a purchase order set. Both are implemented as Web Dynpro components.

According to floor plan and business object type the UI framework looks up the purchase order specific UI configuration, which defines which identification area component and which content area components should be instantiated.

For each UI component the UI framework instantiates an object mapper which controls UI interactions and provides backend access (see figure 4-1). For each identification area component the UI framework instantiates one business objects mapper. For the content area components dependent object mappers or supporting object mappers are instantiated. Dependent object mappers control the UI interaction of their content area component and provide the corresponding set of data. Dependent object mappers have the reference of the business object mapper stored inside them, as they can only communicate with each other via the business object mapper. On instantiation of the identification component, the business object mapper for purchase order is initialized. In addition the dependent object mappers for items, notes and attachments, account assignment, and so on are instantiated. If the transaction mode is create, an initial application UI is displayed. Otherwise the requested purchase order instance is retrieved from the database and displayed.

Within the Web Dynpro application further navigation can be triggered by the user leading to:

- replacing the view(s) in the content area (for example switching from overview to item view of the business object)
- extending the view(s) in the content area (for example displaying item details)
- displaying a modal view while disabling the parent view (for example enter data via value help screens)
- launching of an external view in the same browser window (for example catalog product selection)

The navigation and the corresponding changes of the displayed user interface are provided by the Web Dynpro application.

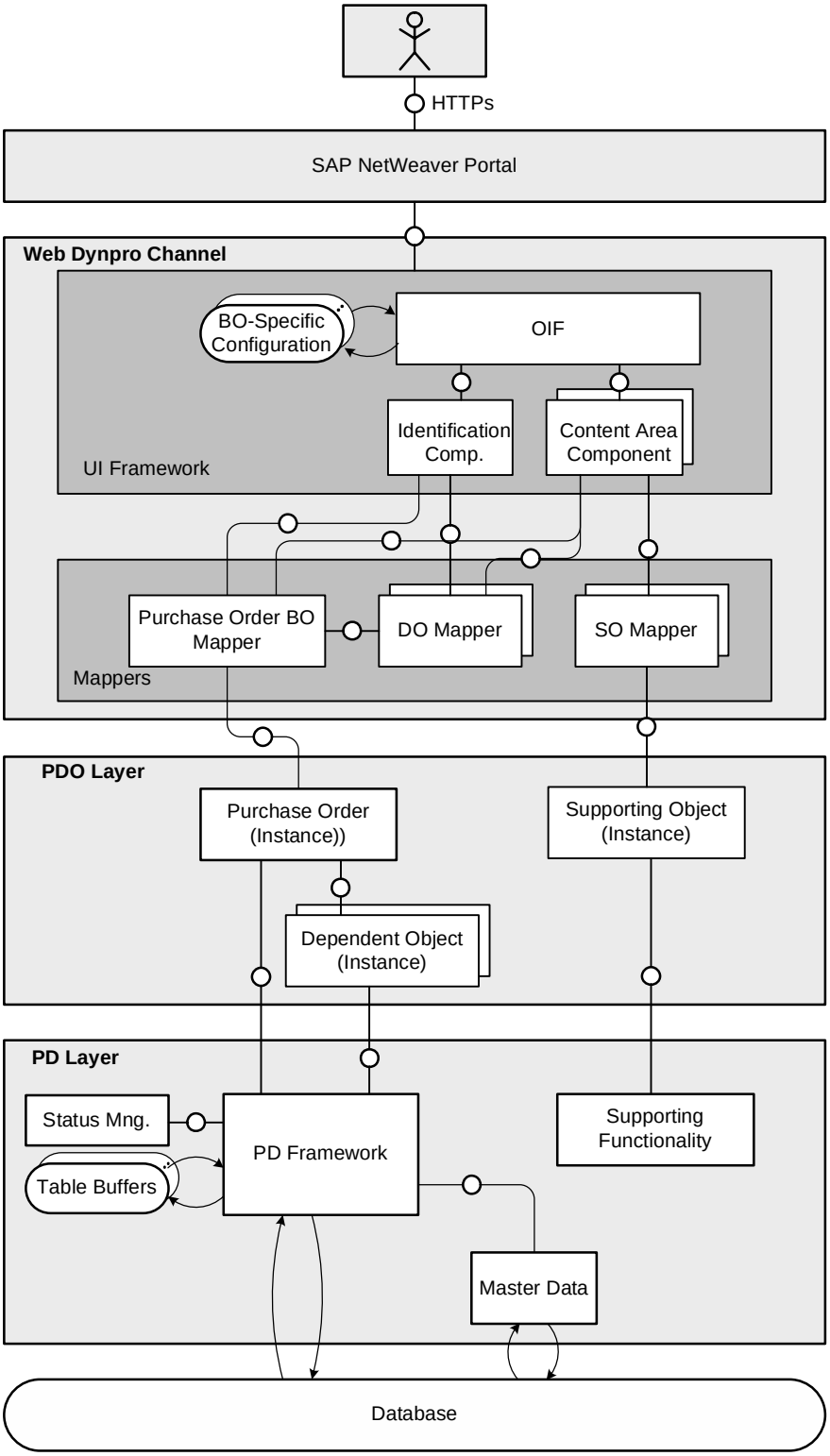

Figure 4-1 Program Flow Taking Example of Purchase Order Business Object

4.2 Creating a Purchase Order

If the transaction mode is create, the purchase order mapper creates a new purchase order instance in PDO layer. This instance creates a new procurement document instance of purchase order by calling the create function of PD framework. PD framework generates a new GUID and creates a new purchase order instance in main memory.

While retrieving the purchase order object for first display the PDO layer calls the GetDetail function of PD framework to read the data of the created document, which includes now business object ID, creation date and so on. The newly created procurement document is then displayed on the user interface and the user can edit the document.

4.3 Editing a Purchase Order

If the transaction mode is edit the purchase order mapper instantiates a purchase order object in PDO layer. This business object instantiates all its related dependent objects and stores the references in member variables. The references are used later for delegating calls.

Business object and dependent objects call PD framework to retrieve their data sets from the database. PD framework reads the data from the database, sets the required enqueues, stores it in the Y table buffer, and passes it to the objects. The business object provides the procurement document data together with meta data, which defines for example field properties (read-only, mandatory and so on, see chapter 4.4) to the object mapper. The retrieved meta data is passed to the UI framework which binds it to the corresponding UI elements. Calls to retrieve dependent object data are delegated from the business object to the respective dependent objects.

Now the user can edit the requested purchase order on the application UI. When the user leaves the current user interface screen, which displays the purchase order, for example by navigating from the header details to the list of positions or presses the refresh button, the current state of the purchase order instance is transferred from the object mappers to the corresponding objects in PDO layer. The objects call the update function of PD framework to process the changes. The update performs certain preliminary checks on the changed data such as UI validations but does not check the consistency of the whole purchase order. Afterwards the business object calls the GetDetail function to retrieve the updated data for displaying it on the application UI.

On the user interface the user can check the purchase orders for consistency and correctness by pressing a corresponding button. In this case, the business object calls the check function of PD framework which stores the complete purchase order in T table and validates the data (see also chapter 3.3.1.3).

If the user finishes the transaction by triggering save, the update function is called for a last time. The purchase order is stored in T table and validated. If all checks are successful, the purchase order is stored in the X table. When the business object in PDO layer calls commit work, PD framework stores the procurement document on the database.

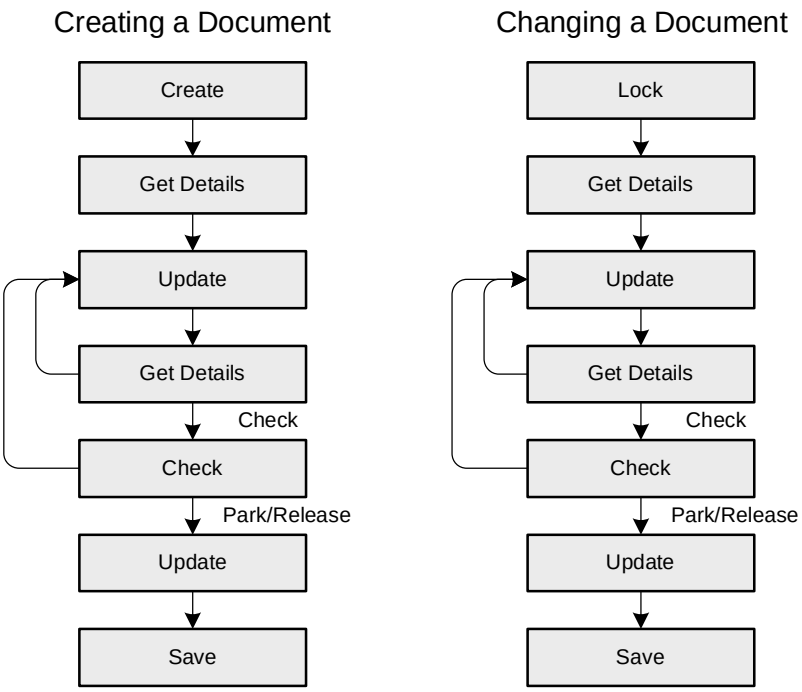

Figure 4-2 Program Flow of PD Framework

4.4 Meta Data Management

In SAP SRM the UI and interface properties of fields and actions are defined using meta data. For example meta data specifies if a field is visible on the UI, if it is read-only, mandatory, and so on, but as well which actions can be executed on a certain set of data.

Meta data is defined using configuration tables, which store static meta data for fields and actions. They also allow specification of methods and classes for dynamic metadata determination at runtime. Customers can adapt the meta data to their requirements.

At runtime meta data for fields and actions is determined on different levels. Thereby the higher level can only restrict the settings of the lower level, for example optional can be changed to mandatory but not the other way round. First the dependent objects evaluate the meta data settings. These values can be restricted by the business object. Further restriction is possible by custom settings and then within the UI layer, for example via personalization.

For meta data management each object implements a meta data provider interface.

4.5 Error handling

In Web Dynpro ABAP you can create and display messages that contain important information for the end user of the Web Dynpro application. Messages are language-dependent texts that are displayed on the screen if, for example, an error occurs or the user has entered data in the wrong format.

The SRM business logic can return application messages in a predefined structure or raise exceptions of error or abort level. The Web Dynpro applications handle this messages and exceptions appropriately by:

- rendering the messages in a prominent message area that is part of all floor plans that allow maintaining business objects

- displaying a popup window on error condition and return to the application
- displaying a popup window on abort condition and terminate the application

Beyond this PDO related message handling, the channel implementation is enabled to provide additional messages, for example as a result of an extended field syntax validation. Moreover, the implicit Web Dynpro context attribute validation is reflected in the message handler design and the message area content at runtime. The behavior of the message handling is outlined in figure 4-3.

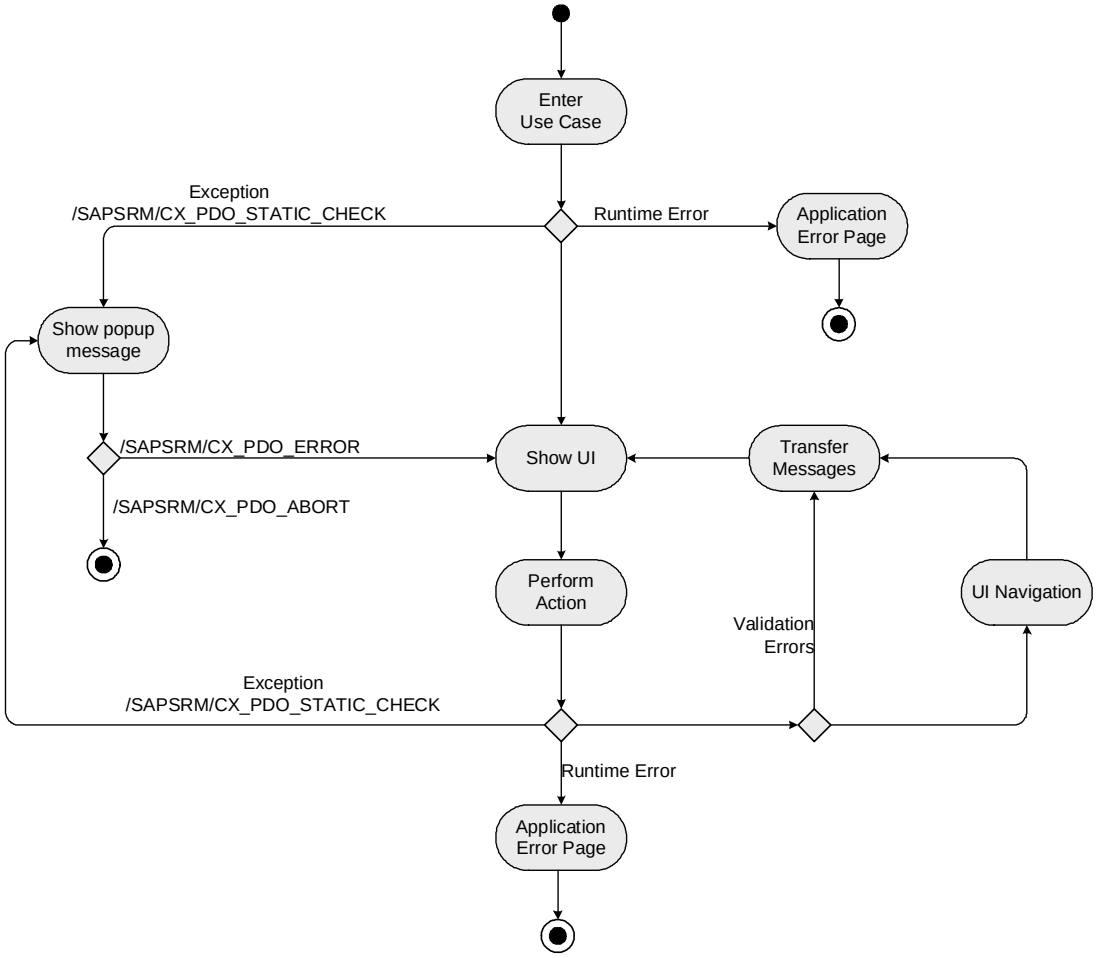

Figure 4-3 Application Message and Exception Handling

The messages are displayed in a popup works always on demand, regardless of what is set in the application. For application development the UI element MessageArea is provided for positioning the message display on the screen. Messages are integrated into the message log of a component using the message manager (interface IF_WD_MESSAGE_MANAGER).

Application messages of the PDO layer are buffered in the message manager that is associated with the business object mapper. The return structure for message allows a field level assignment of error messages. If supplied by the PDO layer, the Web Dynpro UI interprets and maps the field assignment in the according message.

The exception handling is achieved by launching popup with the regarding error text. Depending on the exception class the behavior is either to terminate the current task (abort exception) or to cancel the current action and navigation and enable the user to correct the error condition (error exception).

4.6 Enhancement Technology

SAP SRM provides enhancement possibilities to adapt the solution to the specific requirements of an industry or an individual enterprise.

SRM data structures can be extended by additional fields and tables. To enable this, many SRM data structures provide an extension include. Additional fields and tables are defined as append structures in ABAP dictionary and assigned to the extension include. Extensions are then automatically persisted on the database. If the meta data for the corresponding fields is maintained, the fields are also propagated to the user interface. To ensure that extensions of an industry solution do not interfere with customer extensions, different includes are provided for each of them.

Business logic can be enhanced without modifying the source code by using BAdIs. SAP SRM provides different BAdIs for industry enhancements and customer enhancements to avoid conflicts. For adding UI elements, like input or display-only fields, UI logic and UI checks enhancements to Web Dynpro components can be used.

The switch framework provided by SAP NetWeaver allows supporting multiple different industry solutions and other extensions delivered with the SRM core application. Industry-specific program code and extensions can be activated by the customer via switch framework. SAP SRM uses the switch framework to support government procurement, which is the SRM solution for the public sector, a leasing scenario, and country specific coding in addition to standard SAP SRM.

5 Approval Process Framework

Approval processes play an important role in SRM processes. To ease the definition and processing of approval workflows the approval process framework of SAP SRM provides a simplified process layer on top of SAP Business Workflow and business rules framework. This framework allows defining approval processes by defining a small set of configuration settings.

5.1 Process Definition

All approval processes are executed and controlled by SAP Business Workflow. SAP SRM uses a set of generic workflow templates to model the underlying execution process, the event coupling, the creation, delivery and tracking of work items. These templates include no SRM semantics. On top the approval process framework establishes its own simplified process schema. The process schema is defined at design time, specifying all steps that may appear in an approval process together with the corresponding configuration.

For each business object that is linked to an approval process a specific process definition exists. A business object can have multiple process definitions, but only one can be active at runtime.

5.2 Process Execution

When a business object triggers an approval process the approval process framework sends a request to SAP Business Workflow asynchronously (see figure 5-1). The process control of SAP Business Workflow, which is defined by one generic template, creates now a workflow instance. During level control which is defined by a separate generic template SAP Business Workflow calls approval process framework to check if a decision level should be executed. To do so, approval process framework evaluates configuration and business rules, which are defined and controlled by the business rules framework.

Work item control in SAP Business Workflow is defined by two generic templates, which can be replaced by customers according to their specific requirements. During work item control the approval process framework determines the responsible approver. The responsibility determination is based on the organizational model, where responsibility areas are defined and assigned to organizational units. All users which are assigned to the same responsibility area get the same work item.

The assignment of approvers to work items is controlled by BAdIs which can be adapted by customers.

Example of an approval process: One typical process level is the reporting-line approval, where a line manager approves shopping carts of his employees. In this case, the rule based evaluation activates the level only if for example the total value of the shopping cart exceeds the spending limit of the requester. Next, the context function determines the first organizational unit of the reporting-line hierarchy that has a line manager and identifies that unit as the manager's responsibility area. The whole shopping cart belongs to the same requester and is thus assigned to one responsibility area as one decision set. Finally, a work item is created for the shopping cart and is sent to the manager(s) of the organizational unit.

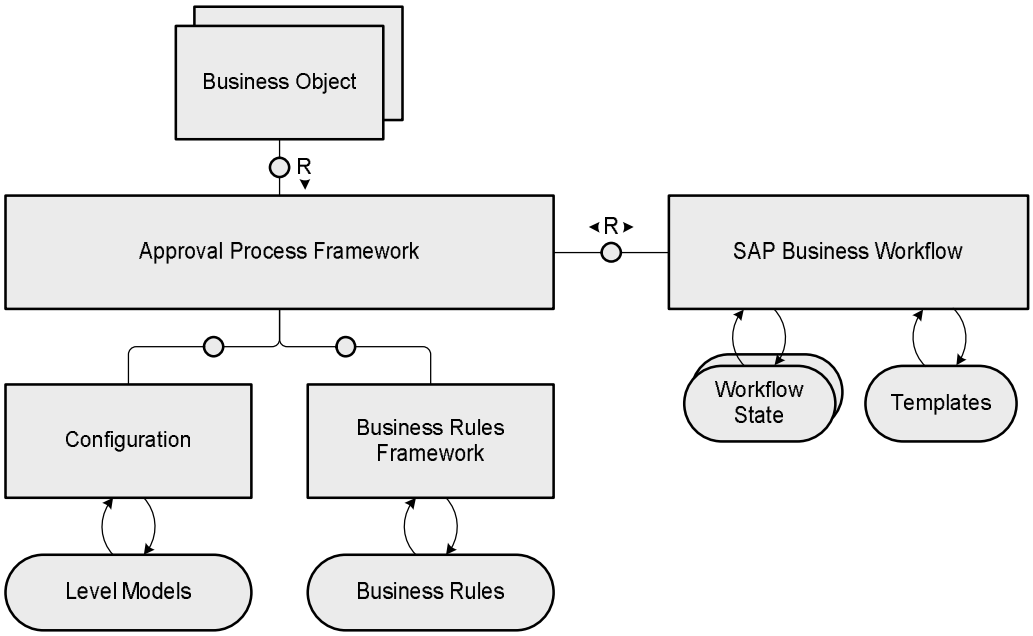

Figure 5-1 Approval Process Framework

For displaying work items SAP SRM uses the universal work list (UWL) of SAP NetWeaver Portal. In addition approval process framework offers buyers an approval process view, which displays the process progress including the history, current state and a forecast. To support the preview of the current applicable process definition, a dedicated service of approval process framework receives the current business object and evaluates the configuration completely in main memory as the corresponding document might not be saved at this point of time.

As soon as the document is stored and ready for the process, a persistent copy of the design-time configuration will be created. At the same time a process instance is established which again references the configuration copy. The current process instance works on this copy for its entire life-cycle. Thus, concurrent manipulations of the design-time configuration do not have any impact on currently active process instances.

6 Further Reading

[Set07] Sachin Sethi: Enhancing Supplier Relationship Management with SAP SRM, SAP Press, 2007

[SAP06] Office of the CTO, mySAP Business Suite Service Provisioning, SAP Architecture Bluebook, SAP AG, 2006

7 Glossary

Term	Definition
APO	Advanced Planning and Optimization
BAPI	Business Application Programming Interface
BW	Business Warehouse
CLL	Channel Logic Layer
CRM	Customer Relationship Management
GUID	Globally Unique Identifier
IPC	Internet Pricing and Configurator
KM	Knowledge Management
LAC	Live Auction Cockpit
MDM	Master Data Management
OCI	Open Catalog Interface
OPI	Open Partner Interface
PDO	Procurement Document Object
PD	Procurement Document
SRM	Supplier Relationship Management
SUS	Supplier Self Service
UI	User Interface
UWL	Universal Worklist
UOM	Unit of Measure
WD	Web Dynpro
WDA	Web Dynpro ABAP
SAP XI	SAP NetWeaver Exchange Infrastructure

