Author:
Axel Schuller

Architecture Guideline Series for Composite Applications

Portal and Process Layer

Version 1.0

October 2007

Table of Contents:

Acknowledgement

The contributions of the Industry Composite Development team made the writing of this document possible

1 Scope of this document

This part of the 'Architecture Guideline series for Composite Applications' covers briefly the portal and in detail the process layer of composite applications or short 'composites'. It is done by mainly guiding through the process layer, discuss possible options and give recommendations for architectural decisions based on experience and best practices acquired in first composite implementation projects.

One benefits most from this part of the guideline series if one has already a general overview of composites as provided in the 'Introduction and Basic Overview' part of this guideline series. It also helps to have a fair amount of knowledge of composite specifications. These specifications are written by either product managers or business experts and explain the composite functionality. For more information on composite specifications, one can review the paper 'Guidelines for Specifying Composite Applications' (GSCA).

2 Architecture Overview

Figure 1 provides an overview of the layers of a composite, the technologies to be used in each layer, and the communication between the layers.

The following chapters provide detailed information especially about the process layer. The approach taken is to answer the following questions for each layer:

- What is the layer about?
- What are the technology options?
- Link to 'Guidelines for Specifying Composite Applications' (GSCA) for the relevant high level information

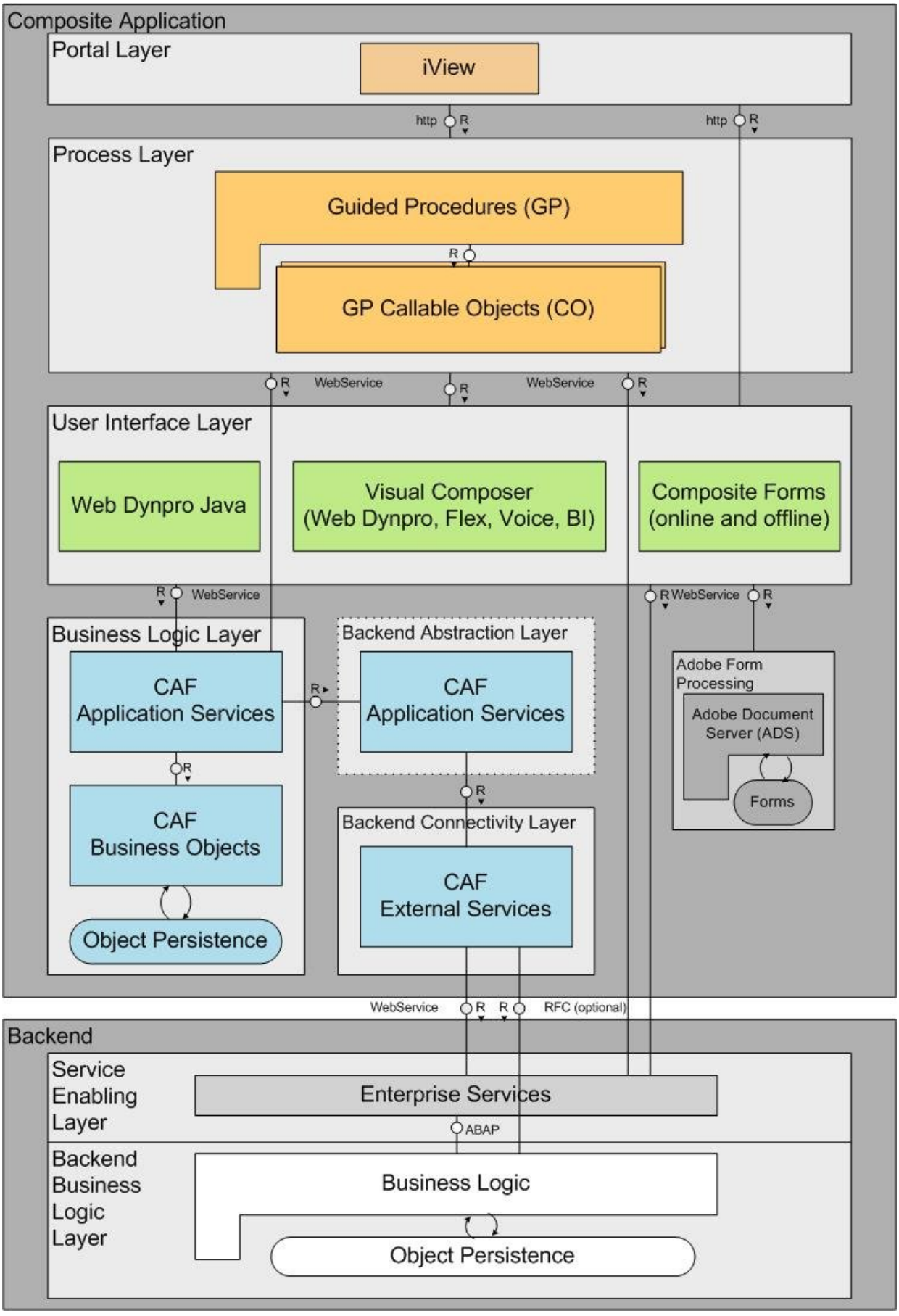

Composite Application
Portal Layer
iView
http
R
http
R
Process Layer
Guided Procedures (GP)
R
GP Callable Objects (CO)
R
WebService
R
WebService
R
User Interface Layer
Web Dynpro Java
Visual Composer
(Web Dynpro, Flex, Voice, BI)
Composite Forms
(online and offline)
R
WebService
R
WebService
R
Business Logic Layer
CAF
Application Services
R
CAF
Application Services
Backend Abstraction Layer
Adobe Form
Processing
Adobe Document
Server (ADS)
Forms
R
CAF
Business Objects
Object Persistence
R
Backend Connectivity Layer
CAF
External Services
WebService
R
R
RFC (optional)
Backend
Service
Enabling
Layer
Enterprise Services
ABAP
Backend
Business
Logic
Layer
Business Logic
Object Persistence

Figure 1

3 Portal Layer

The smallest visible unit in the portal is an iView (portlet), the basic element of a portal page. It is a small application that retrieves information, processes it, and displays it in the portal. One can use iViews to integrate composite applications into the portal pages.

☑ **In the Portal Layer, the SAP Enterprise Portal is used as the single choice of technology.**

Within the portal layer the user interfaces and processes of a composite are provided in a role-based manner using the work- and control-center concept. This means that the roles described in GSCA chapter '4.4 Roles' need to be mapped to the roles maintained in the portal. Let's have a closer look at this concept: A user may have different roles. For each role a work center exists. The work center provides a role and task-oriented view of data and activities (e.g. customer service care, employee self service, purchase order management). It consists of a set of pages organizing and supporting the user's activities in a certain area (work set), e.g. the work set 'employee self service' with the activities e.g. leave request and address change. The control center provides an overview of all work centers a user is assigned to.

The portal layer serves as the central entry point for work items the user has to take care of. This is covered by the so-called Universal Work List (UWL) which is needed to forward follow-up process steps between users.

3.1 Federated Portal Network (FPN)

The federated portal network capabilities of SAP NetWeaver make it easy to share content between distributed portal installations, both SAP and non-SAP; thus providing a single portal access point to portal information, services and applications distributed on portals throughout the entire organizational network.

The SAP NetWeaver CE portal serves as a platform for composites. Through federation, these composites can be exposed to the central portal. Customers benefit by taking advantage of the advanced composition capabilities offered in SAP NetWeaver CE, while keeping their corporate portal in a stable and less frequently updated environment, ensuring a consistent end-user experience.

A federated portal network (FPN) allows organizations with multiple portals to share content between them by the usage of:

- Remote Role Assignment (RRA) enables the consumer administrator to assign complete roles offered by a SAP producer. RRA interoperability between SAP NetWeaver CE 7.1 and SAP NetWeaver 7.0 requires SAP NetWeaver 7.0 SP9 and above.
- Remote Delta Links (RDL) enable the consumer to embed remote portal content (iViews, Pages, Work sets etc.) offered by a SAP producer into local content. RDL Interoperability between SAP NetWeaver CE 7.1 and SAP NetWeaver 7.0 requires SAP NetWeaver 7.0 SP12 and above

Figure 2 illustrates a potential landscape setup.

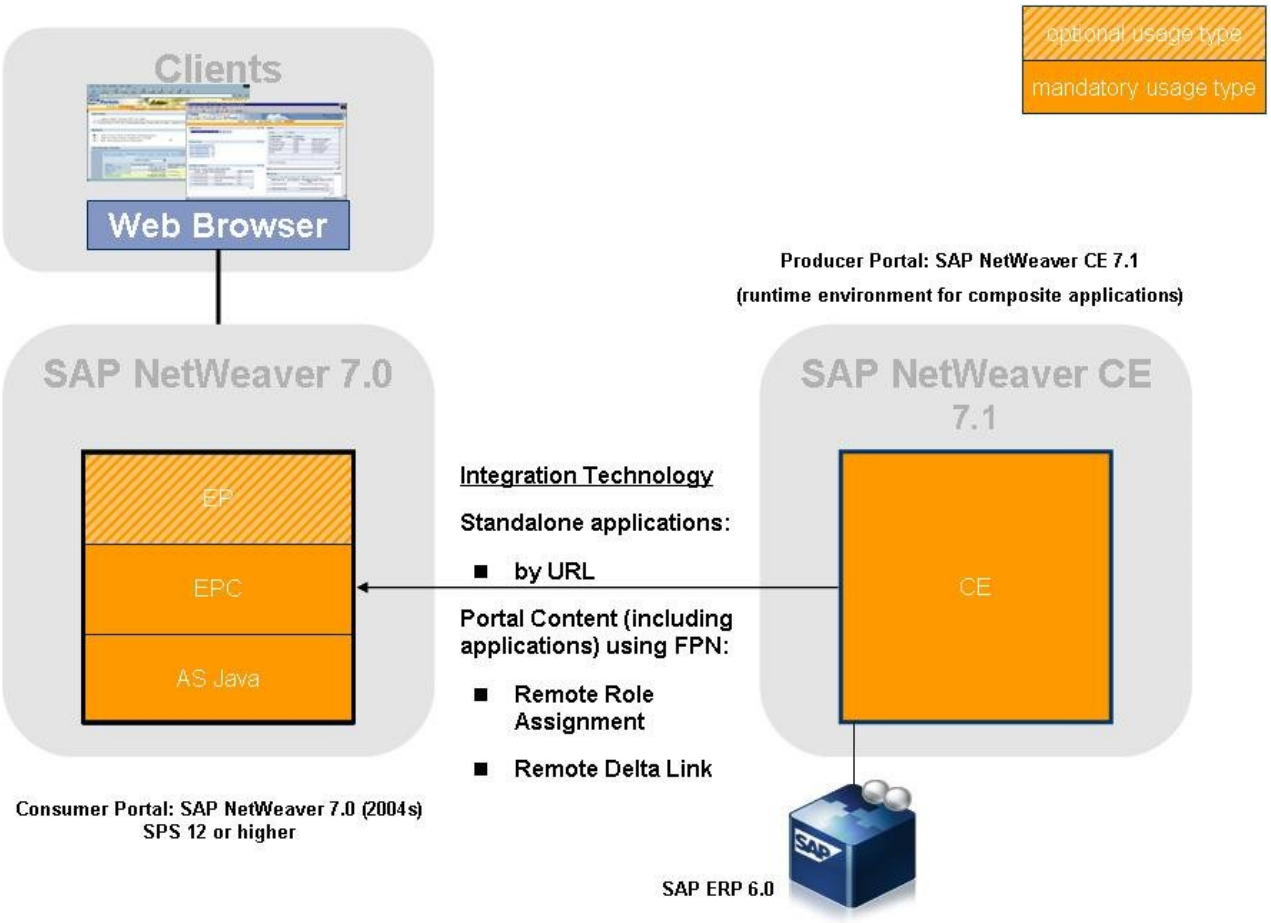


Figure 2

In Figure 2 SAP NetWeaver 7.0 provides the central corporate portal. Here is at least release 7.0 SPS 12 required with Enterprise Portal Core (EPC) and optionally the full Enterprise Portal (EP). Composite applications that have been built with SAP NetWeaver CE 7.1 can be integrated into the corporate portal either via URL or FPN.

FPN offers two major content usage modes:

- Remote Role Assignment (SAP NetWeaver CE 7.1 SP1): Allows the remote consumption of entire portal roles (including all embedded content). The consumer portal cannot modify the role's content.
- Remote Delta Links (SAP NetWeaver CE 7.1 SP3): Allows the remote consumption of specific content objects (iViews, pages, roles etc.) and their reuse and modification (in the same way as local delta link objects can be manipulated) without affecting their source objects on the remote portal.

4 Process Layer

Within the Process Layer the process steps are defined, the sequence in which they are executed by which roles and how the context data of a process is passed between the process steps. The process, including required process steps, is described in GSCA chapter '4.2 Processes Overview – 4.3 General Process Information'. The complete process specification can be used for the process architecture.

A process should be defined by using a:

- ☑ Guided Activity if:
  - o Only one user/role is involved
  - o It is a simple sequential process that will be finished 'in one go'
  - o All steps are part of one transaction
- ☑ Guided Procedure if:
  - o Several users/roles are involved
  - o Each step is one transaction

4.1 Guided Activity

A Guided Activity basically consists in Web Dynpro of a RoadMap control and in SAP NetWeaver Visual Composer of a Wizard indicating the current step of a sequential process and buttons for navigating back and forth between the steps or completing/canceling the whole sequence.

Unlike a Guided Procedure a Guided Activity is limited to a single user transient task flow without own context or persistence since there is no underlying workflow engine like in the case of Guided Procedures. On the other hand, if it's clear that there will be only one contributing processor, there is less overhead (in terms of development effort and performance) compared to Guided Procedures as well as the ability to navigate backwards through the steps.

A possible indicator for such 'exactly-one-contributor' processes is: The UIs of the individual steps could be combined into one UI, to be processed by one user 'in one go'.

4.2 Guided Procedures (GP)

Guided Procedures is a framework consisting of two main parts: the GP Design time for modeling user-centric processes (i.e. it's not suited for A2A scenarios) and the GP runtime that executes these process models.

GP makes it easy to bond diverse backend applications and services into a single business workflow. It provides role-based access to resources and guidance through the workflows at runtime, thus helping end users to identify and complete their tasks easily. A Dashboard allows to monitor the progress of the whole process by users assigned to the relevant role (for more information about roles see chapter 4.2.3 Roles). The individual work items of each involved user are available in their GP Inbox.

Figure 3 provides an overview of Business Process Management (BPM) and Guided Procedures (GP) terminology.

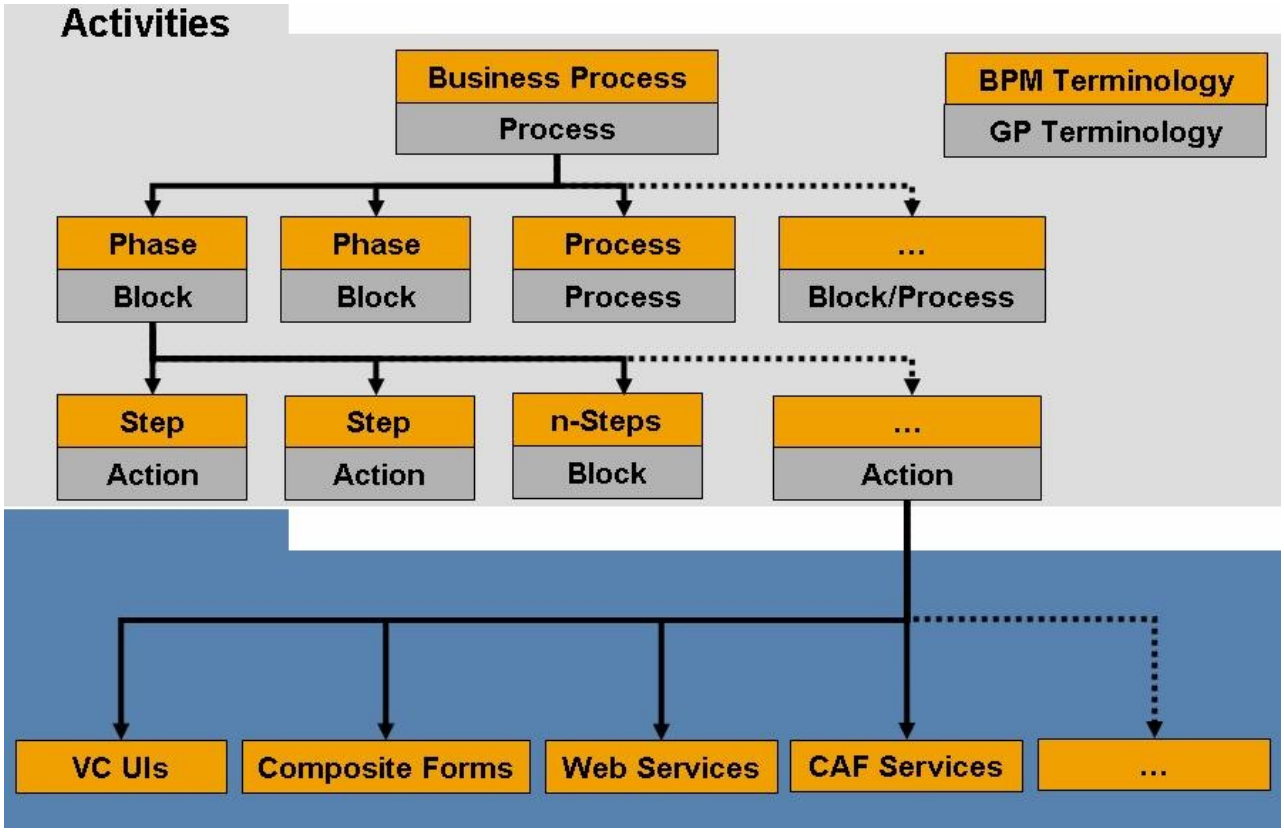


Figure 3 BPM – GP Terminology

The process modeling can be approached:

- Top-down: Create the design time objects (Process template, Blocks and Actions) first and finally create and add the Callable Objects.
- Bottom-up: Create the Callable Objects first and insert them into the embedding design time objects later.

Blocks are the structural units that build a process in Guided Procedures. They are re-usable and may contain actions, nested blocks or processes. The block design defines how the items in the block are executed. One can choose to execute them sequentially, in parallel, in a loop or let the user choose between several alternatives. This is defined by the block type. Blocks have input and output contexts, as well as role contexts. The parameters and the roles are inherited from the items in the block, and can be consolidated at design time.

Actions decouple process steps from services and user interfaces to allow business experts to model processes at a non technical level. They are executable units that define a single process step. Each action can refer to either one or two callable objects – one for execution and an optional one for display. Attaching a callable object to an action is a prerequisite for using it. The definition of the action adds metadata to the callable object functions, thereby defining how it is invoked and executed within the process.

Callable objects enable the execution of external applications or services within the GP framework.

For performance reasons it is recommended to:

- ☑ **Focus on straight forward, sequential process flow**
- ✘ **Try to avoid**
  - ✘ **Dynamic parallel blocks**
  - ✘ **Nested blocks and deep hierarchies (block in block in block)**
- ✘ **Use not more than about 15 process steps – this does not mean that the overall process can not be more complex! Just organize it better (sub-processes).**

To design a Guided Procedure it is recommended to follow the step by step approach in the 'Guidelines for Specifying Composite Applications' (GSCA) chapter 'Guideline for Writing a Composite Specification':

- Read step 1 from 'General Application Information (Executive Summary)' for an overview.
- Read steps 2-6 for information about the process. They provide the information for the process model as illustrated in GSCA chapter '4.5. Visualization of Process Flow'. This process model is a good starting point for composite design.

Figure 4 shows a part of the process model of the customer service care example as described in GSCA:

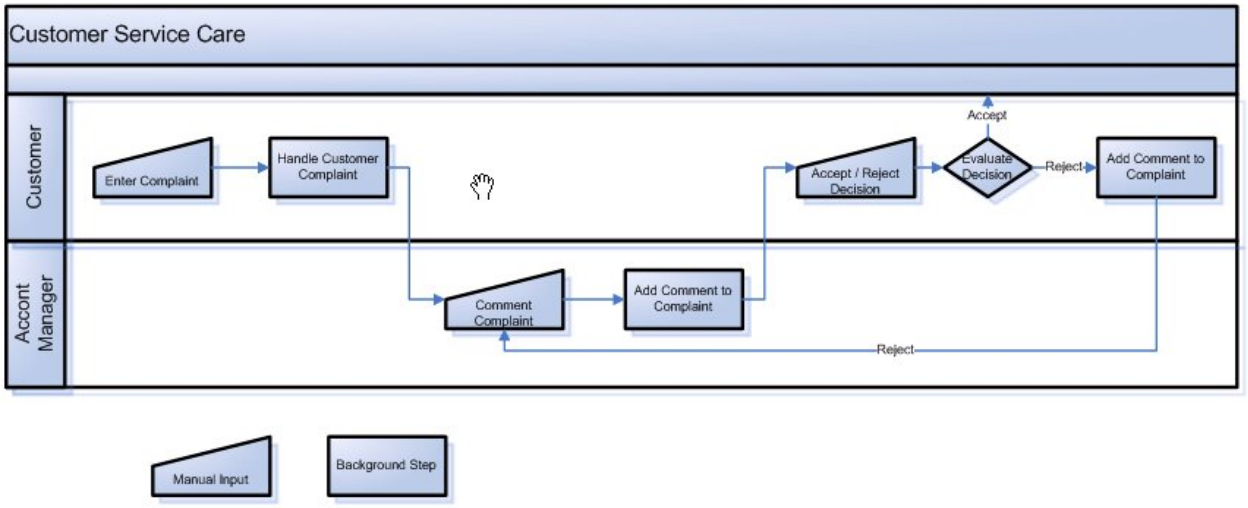


Figure 4

1. Customer enters a complaint
2. Complaint is created in the background
3. Account Manager reviews complaint and enters comment
4. Application service is called to update the complaint
5. Customer reviews comment from account manager and chooses either to accept or reject the decision by the Account Manager
6. If the customer rejects the decision, a customer comment is added to the complaint and it is returned to the Account Manager
7. Order Information is updated

Guided Procedures is designed to implement such workflows. It enables users to set up and execute collaborative business processes easily by seamlessly integrating backend system transactions and services into the business process context.

4.2.1 Flow Logic

In Guided Procedures flow logic can be modeled by using different kinds of blocks and result states, see GSCA chapter '4.5. Visualization of Process Flow'.

Each of the **3 block types** has a different logic for executing its actions. Generally, actions are executed in the following ways:

- Sequential: In the sequential block, child items are executed in sequence.

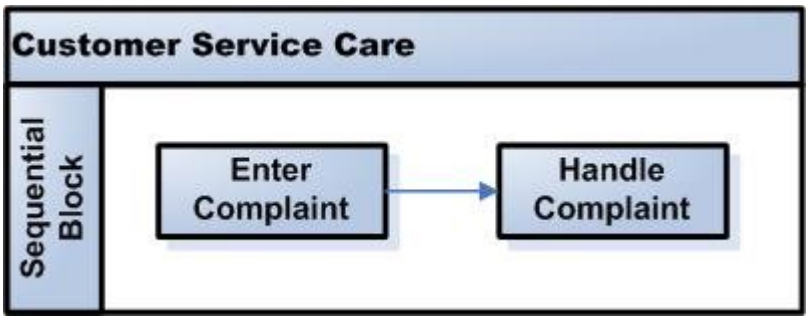


Figure 5

- Parallel: When the parallel block is executed all its items are started simultaneously. The block is completed once all items have been executed.

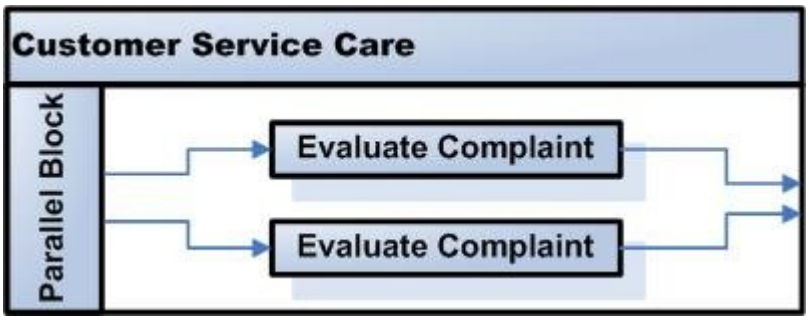


Figure 6

- Loop: There are Precondition and Postcondition Loops possible. The following Figure 7 is a Postcondition example. The actions in this loop block can be executed again if the decision action executes with a result that requires the loop to continue.

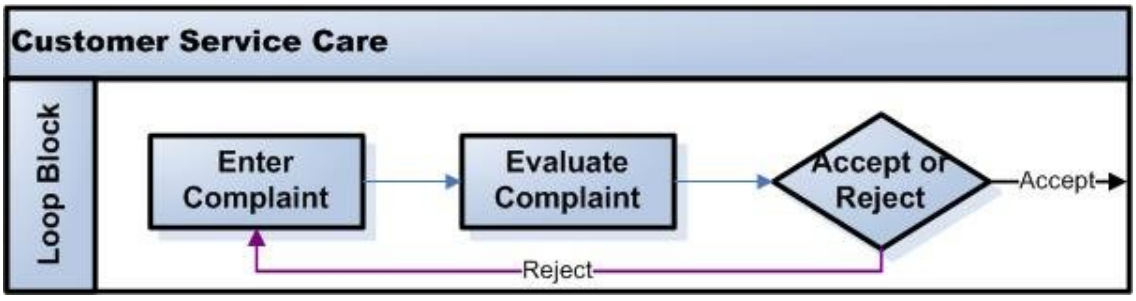


Figure 7

Result states indicate the result of the operation that an action has carried out. For example, an approval action will have 'Approve' and 'Reject' result states. In the design time, result states can be mapped to a previous or subsequent item or used to exit the current block. This means that steps can be skipped or returned to later. At runtime when the action executes it returns a previously defined result state. The GP

engine redirects the logic according to the mapping. If the result states of an action are not mapped explicitly then the default is that the next element is executed.

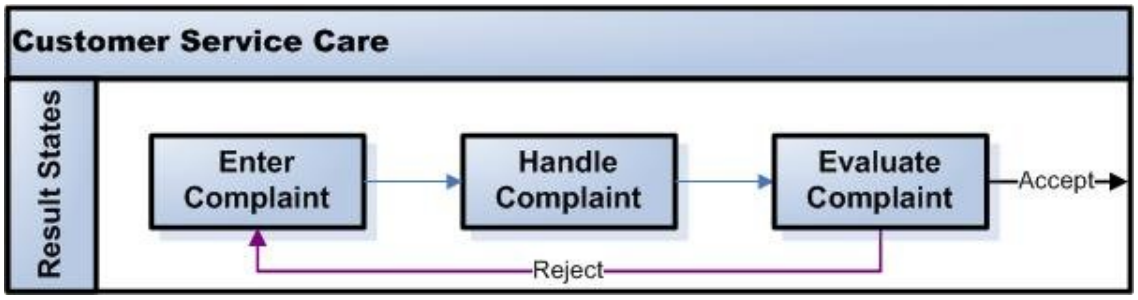


Figure 8

4.2.2 Data Flow / GP Context

At runtime, passing of data between process steps is done via the so-called GP context, which is in a nutshell a data persistency with a structure that can be modeled and mapped in the GP design time for COs, actions, blocks and process templates.

- ☑ **GP context stores all intermediate data by default.**
- ☑ **Save 'simple' / 'low amount of' intermediate data in GP (Process) context (see Figure 9)**
  'Simple' intermediate data that needs to be exchanged between actions, should be stored in the GP context. This has the major advantage that in case the running process instance is cancelled somehow, the GP framework will perform all clean-up; moreover, it's easier for a customer to enhance the process template if all data is 'at hand' in the context.
  This is easier than the following method recommended for 'complex' intermediate data
- ☑ **Save 'complex' / 'high amount of' intermediate data in CAF business objects but separate them from operational data**
  Try to avoid 'complex' / 'high amount of' intermediate data.
  For performance reasons it is recommended to store 'complex' / 'high amount of' intermediate data in SAP CAF business objects. The intermediate data should be stored separately from operational data in order to prevent inconsistent data.

These measures will help to prevent inconsistencies:

- ☑ **Save only consistent business objects in the business logic layer**
- ☑ **Parameters in the GP context should be restricted to keys. Texts should be included in the GP context only if texts for keys cannot be determined via F4 help values in the callable object UIs.**
- ☑ **Only expose parameters to the GP context that must be used within the process.**
- ✘ **No F4 help values should be stored in the GP context as they could change**

Variables can be made available to subsequent process steps by exposing them to the process context (Figure 9).

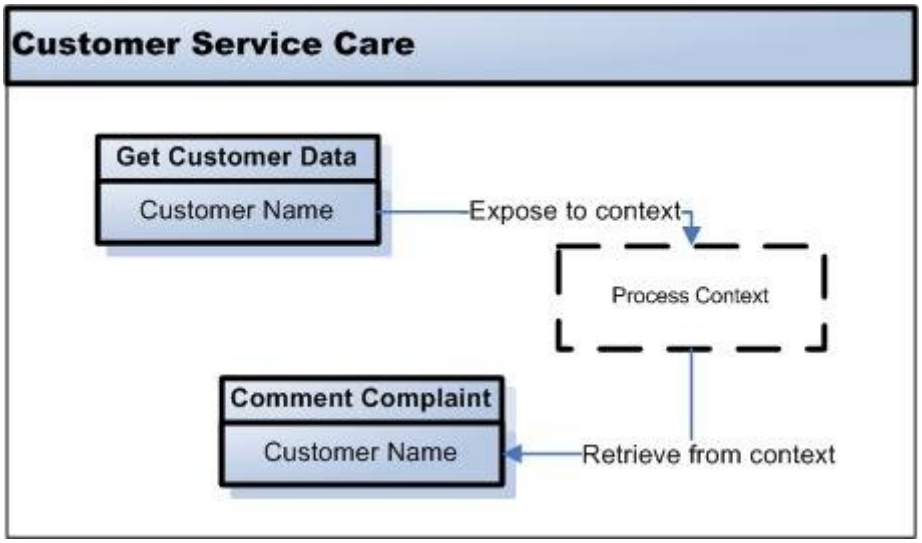


Figure 9

An overview of the customer service care example is provided in Figure 10

The data is identified by keys such as Customer ID and Order ID to transfer the Complaint data

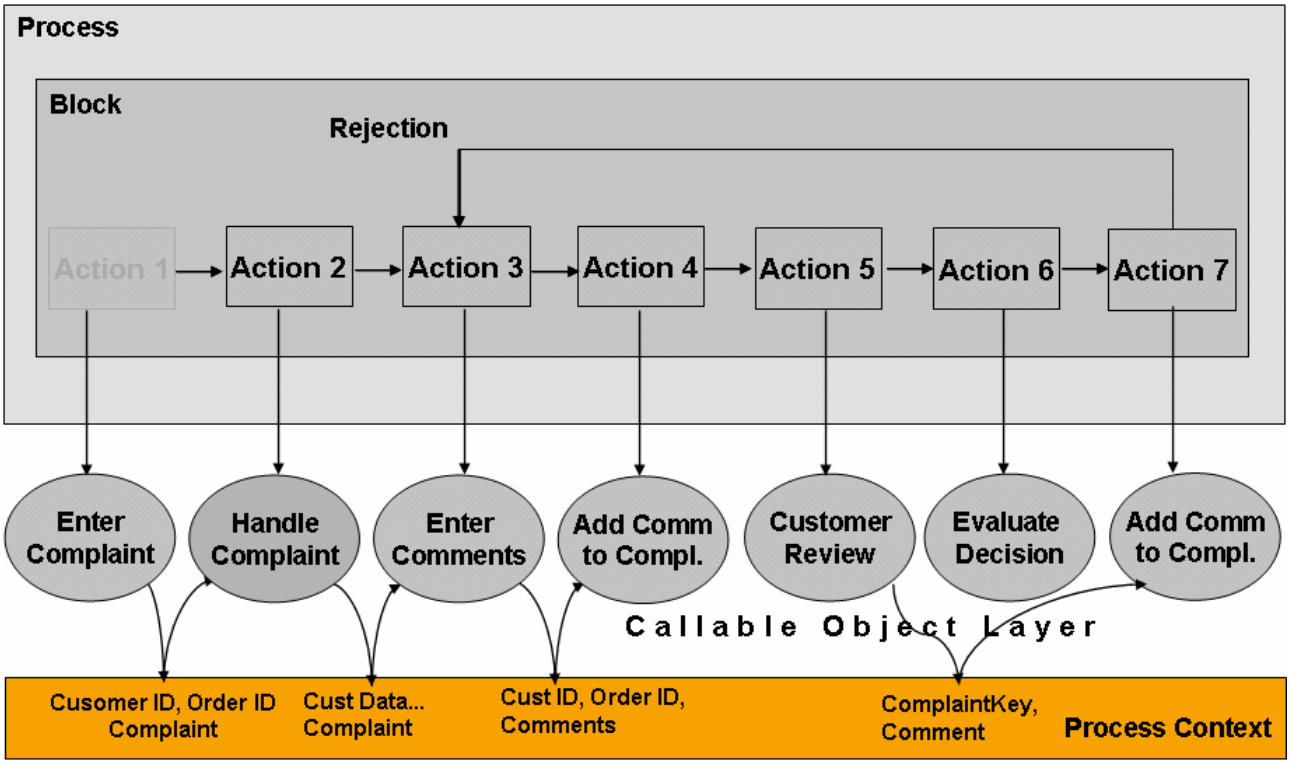


Figure 10

Processes, blocks and actions have input and output parameters. Output parameters can be mapped to input parameters. This is done by using consolidation. Consolidated parameters always hold the same value. This is displayed in Figure 10

4.2.3 Roles

Before a GP process is initiated, it needs to be configured appropriately. One of the most important process setup aspects is the configuration of the process roles since user assignments are managed using roles, see Figure 11**.** They specify which user executes each process step at runtime.

Configuring process roles includes consolidating roles into groups with the same user assignments, setting up defaults, assigning users, and so on. For details on how to do this, see Process Role Configuration.

More about the built-in and additional GP process roles can be found here: Process Roles.

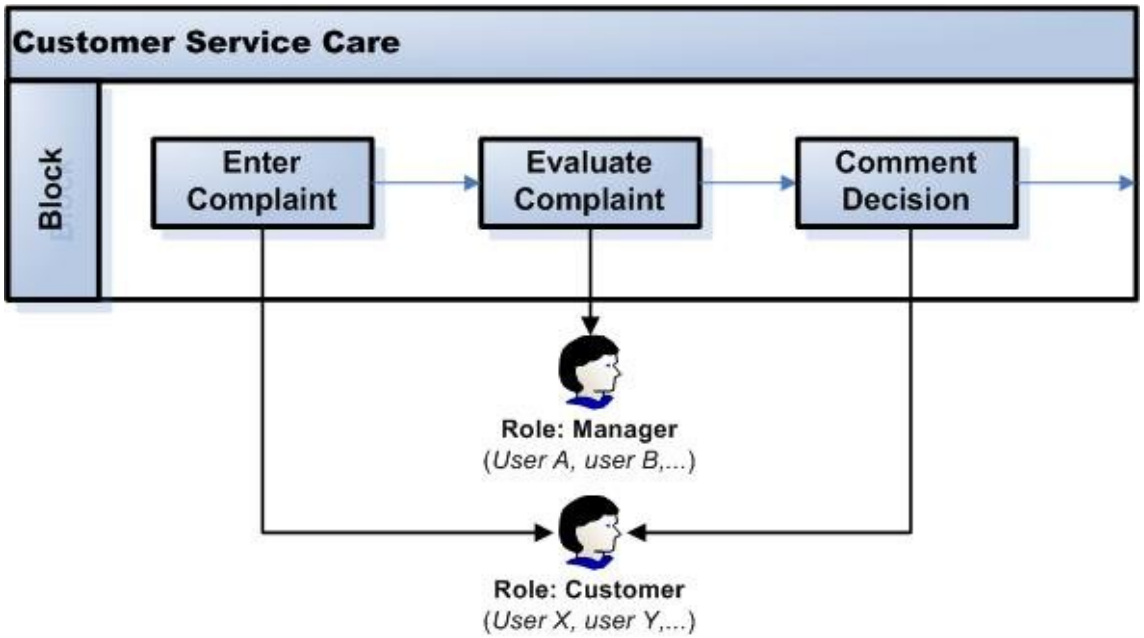


Figure 11

Roles in GP procedures control who is entitled to execute a specific action. When one puts an action in a block a separate role is created in the block for the new action. This role propagates up to the parent process. Roles for several actions can be consolidated so that only one role will appear in the parent block. One can:

- consolidate roles
- define role defaults
- add role restrictions

4.2.4 Callable Object (CO) / Action Design

- ☑ **When using the Callable Object type 'Web Service', only use logical destinations**
  Logical destinations are required for all Web services that need authentication.
- ☑ **Callable Objects offering an UI should ensure data consistency by calling a check service**
  This will help to avoid inconsistent data being persisted
- ☑ **If consistency checks depend on data entered in different callable objects, the corresponding actions including the update service will be called within a loop block. Then actions can be executed again depending on the results of the check service.**
- ☑ **Unnecessary process steps can be avoided by:**
  - ✘ **No splits of UIs only for re-use purposes**
  - ✘ **Do only separate UI and update callable objects, do not do further separation, e.g. check service in separate callable object**
- ✘ **Don't split up updates to SAP CAF business objects: there should only be one CO/action calling one SAP CAF application service combining all necessary SAP CAF BO updates**
  Combining SAP CAF BO updates into one SAP CAF application service operation allows to roll back changes if one of these BO updates fails.
- ☑ After the update service call step, add a process control CO of type ‚Decision (comparison to pre-defined value)' or ‚Business Logic' to handle service call error messages
  Since in the guideline part 'Business Logic Layer' the recommendation will be given that all exceptions (from the backend as well as from the composite business logic itself) are to be caught and passed as GDT Log part of the (Web Service-exposed) SAP CAF Application Service output structure, handling error messages only based on GDT Log should be sufficient here.

Figure 12 and Figure 13 illustrate the enhancement and re-use possibilities by separating the UI from backend calls. The principles are valid for all enhancements and reuse cases, e.g. a standard / core object could be used by several industries.

4.2.4.1 Separation of UI and Update:

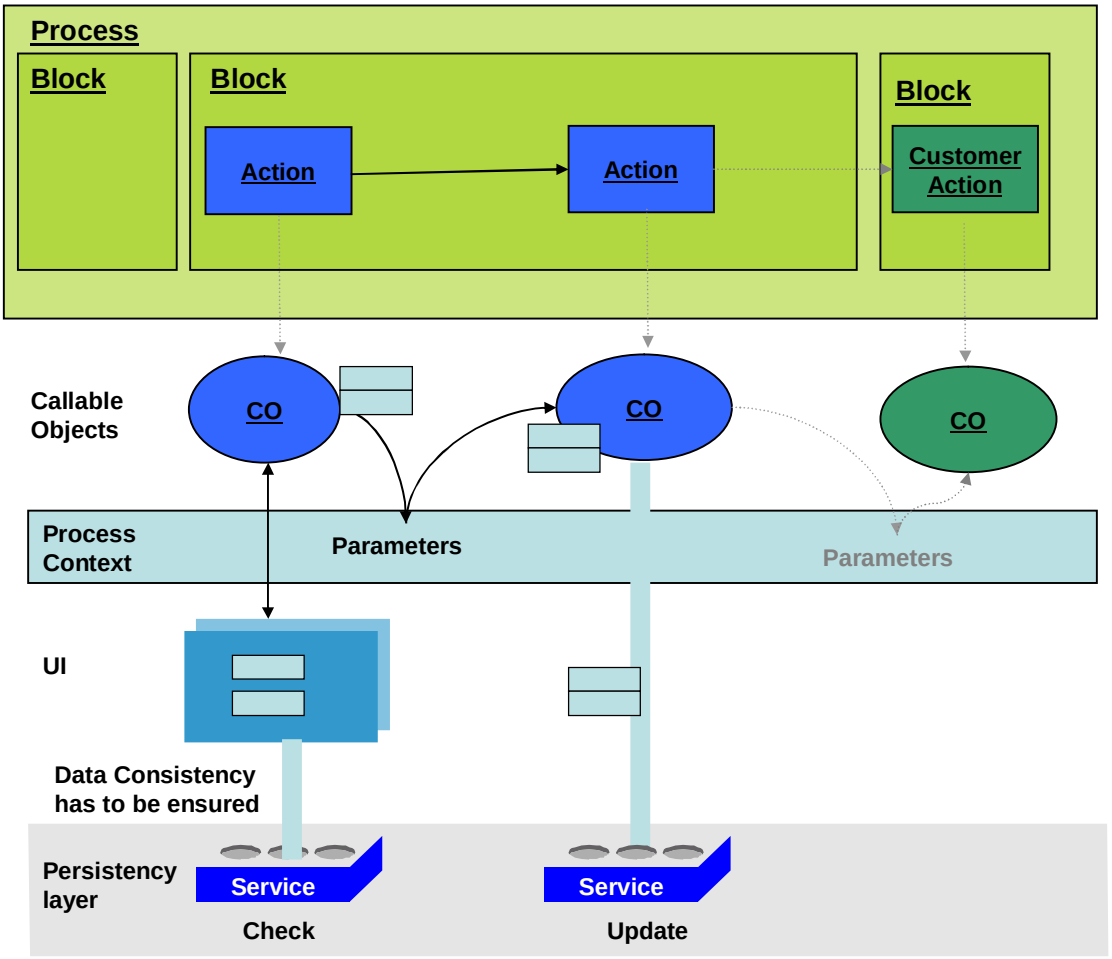


Figure 12

Figure 12 shows that user interactions can be divided into two callable objects / actions, one handling just the UI and performing the appropriate check services, the other one handling the backend update service calls (background action). This separation offers high flexibility of the process flow definition, e.g. it would be very easy to add an approval step between data maintenance and the update service call.

Calling appropriate check services before the update service call minimizes the possibility that the update will fail, but this cannot be excluded obviously. For example in highly parallel scenarios where many users are accessing one Business object at the same time, it might happen sometimes that a version ID was retrieved by calling Read<BO> service (that we also count as 'check service' in this context) in the UI action but this version is already out-of-date when the process continues with the Update<BO> service call in the update action.

These error cases are typically expressed by means of a dedicated service return parameter (in case of Enterprise Services, this parameter adheres to the GDT Log, in case of BAPIs, it has the type BAPIRET2 defined in ABAP DDIC). Based on this parameter, one can add an additional process control callable object of type 'String comparison' or 'Business Logic' to define a result state fired in case of the respective error condition. The screenshot below illustrates that for the CO type 'Business Logic'.

4.2.4.2 Reuse of an action / callable object:

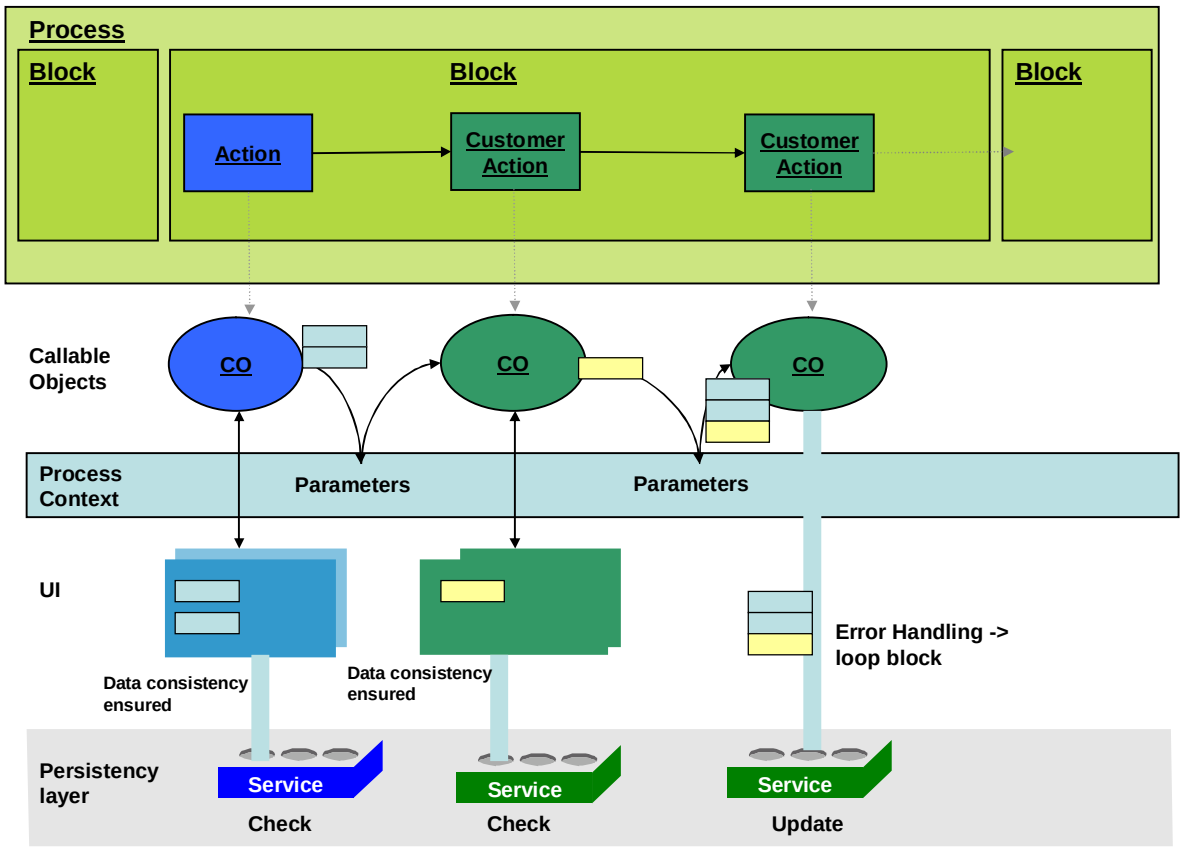


Figure 13

Figure 13 shows an action that is reused in a customer's process flow. The original UI step collecting and checking the data is still present, but after that, the customer has added another UI step that puts additional data to the GP context in the end. After that, both the data from the original UI step as well as the data from the customer UI step is consolidated into the backend update step.

Again, all data consistency checks have to be carried out in the callable objects offering the UIs. If data is changed in several callable objects and there are dependencies between the data, errors could occur during the service call, which could require that input data to be changed again. This repetition until all data is consistent can be realized with a decision callable object used within a loop block, creating decision dialogs. The service calls have to return a value indicating whether the call was successful or not. This parameter value can be interpreted within the process flow definition.

4.2.5 Starting a Guided Procedure (GP)

Guided Procedures can be started in the following ways:

- Via Web Service:

  For each (active) process template, there is a dedicated 'start web service' generated and registered to the central GP wsil document /gpcore/GPProcessDiscoveryWSIL. This means: creating a web service destination in SAP NetWeaver Administrator pointing to /gpcore/GPProcessDiscoveryWSIL will give access to all GP process-start web services from all SAP tools that can handle web service destinations.

  This is the most preferable way to start GP processes, especially in case one needs to supply parameters only known at runtime.

  Further information is available here: Starting a Process Using Web Services
  For a sample, see the following tutorial: Importing a Web Service in Guided Procedures

- Via API: The GP framework offers a Java-based API for various tasks, one of them is starting a process - also parameterized if needed. Only use the GP APIs in case the Web Service based process instantiation doesn't offer the needed functionality.
- Via URL: A process can be started via a so-called instantiation URL that can be retrieved for each (active) process template from the GP design time, looking like this:

  ../../sap.com/caf~eu~gp~ui~inst/AInstantiation?process.template.id=DA479C30036411DAA699000BCD3B046D

  This way of process instantiation makes only sense in case there are only parameters that can be set at design time (to append all input parameters of your process template with their respective default values to the URL, set the checkbox 'Include default parameters' when retrieving the instantiation URL in the GP design time).

  It is recommended to only choose this instantiation type in case it's not feasible to use the process start web service or the GP APIs.
- Via Composite Form: A process can be started by submitting a Composite Form, supplying parameters to the process – or better mapping form fields to the input parameters of the process. It's also possible to create the necessary form template directly from the GP design time.
- GP runtime: The GP runtime application has its own native entry UI that also allows to instantiate processes from all available process templates.

☑ **It is recommended to add a relative due date/duration (e.g. '3 weeks from process start') to each process template, which terminates processes after a sensible timeout to prevent orphaned instances**

One example for the origin of orphaned processes: when a GP Process is integrated into the portal via an URL iView (pointing to the process instantiation URL), it is possible to instantiate it by simply navigating to this iView. However, users may be unaware of this more or less implicit instantiation, which leads to unexpected processes/entries in their or other users' workflow inbox.

4.2.6 Workflow / UWL

The Universal Worklist is used as the central access point to tasks, alerts and notifications. You can access the UWL to manage and track your tasks.

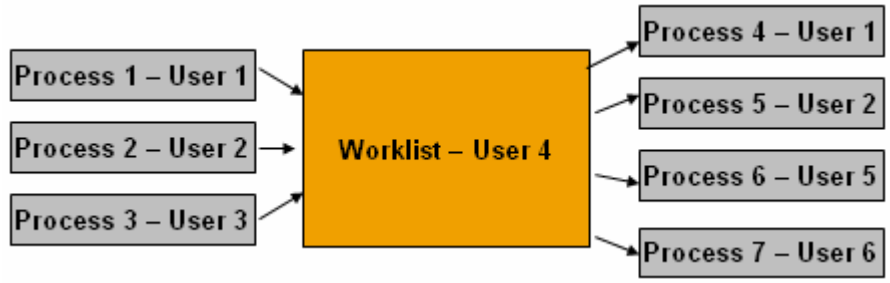
- ☑ **The UWL shall be used as workflow inbox for all processors of Guided Procedures process actions.**
- ☑ **UWL is part of the CE 7.1 SP3 delivery. Before SP3 the UWL only works in a Federated Portal Network (FPN) scenario with a SAP NetWeaver 7.0 consumer portal hosting the UWL.**

  See the chapter 3.1 Federated Portal Network (FPN) for more details on FPN**.**
- ☑ **The Guided Procedures runtime and administration portal worksets (and thus the Guided Procedures inbox) should only be accessible by administrators.**

  The UWL has to be configured to retrieve workflow items generated by the Guided Procedures framework.
- ✘ **UWL cannot be used as mass processing tool, each work item has to be processed individually**

Within the UWL each work item has to be processed individually, no mass functions on several work items are possible, which means that in such cases, a custom solution is needed.

The following example illustrates that:



User 4 in the second step should be able to 'batch assign' approvers to some business objects, each one provided via one individual process instance Process 1-Process 3.

As said, Guided Procedures doesn't have a functionality to work on several process instances/workitems in parallel, so it's not sufficient to simply have some 'Assign approver' action in the corresponding process template (which would give User 4 one separate workitem for each business object).

User 4 in the second step should have custom multi-selection enabled worklist UI –which is not part of the GP process– that displays the business objects provided by Process 1-Process 3 and allows for assigning approvers to them. After this has been done, follow-up Guided Procedures processes are started for each business object, sending respective workitems to the defined approvers (User 1, User 2, User 5 and User 6.in the example).

It should also be possible to use the Guided Procedures APIs to collect and complete running process instances for such a custom worklist.

- ✘ **UWL is 'inbox only' which means that the execution of actions in the UWL is not possible**

Guided Procedures process steps cannot be completed within the UWL directly, e.g. no ad-hoc Approve/Reject function in the UWL is supported. It is also not supported to forward the Guided Procedure workitem to another processor.

- ☑ **In case forwarding of Guided Procedures workitems is desired, use the process control item 'Delegate Role'**

The process control item 'Delegate Role' allows the respective processor to assign his current workitem to other users from the Guided Procedures runtime UI (where he automatically navigates to when clicking a Guided Procedures workflow item in the UWL).

5 Copyright