SAP ARCHITECTURE BLUEPRINT

SAP Netweaver Composition Environment

VERSION: 0.1

DATE: 07.09.2007

AUTHORS:

Ulf Fildebrandt

PROJECT NAME/cPROJECT TITLE:

SPONSOR/PROJECT INITIATOR: KLAUS KREPLIN

PROGRAM/PROJECT LEAD: PRASAD KOMPALLI

LEAD ARCHITECT: ULF FILDEBRANDT

DEVELOPMENT: ☑ SAPLabs (Walldorf, Sofia, Bangalore, Tel Aviv)

☐ Partner/ISV ____________

Document History		
Version	Date	Status (Comments)
0.1	07.09.2007	First draft
0.2	8.11.2007	Feedback incorporated

I MARKET AND PRODUCT BACKGROUND OF PROJECT/PROGRAM

Planned release date:	SAP NetWeaver 2007	
Underlying SAP NetWeaver release:	SAP NetWeaver New York	
Used SAP NetWeaver stacks:	☐ ABAP	☑ J2EE/Java EE 5

Use cases targeted by the project/program:

- Enable development on top of the Business Process Platform (BPP) following the SOA architecture
- Customers can run Java applications developed by ISVs on BEA, Websphere etc. on SAP NW CE

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

- Provide a high productive development and runtime environment for composition
- Accelerated release cycles – delivering innovative technologies/content faster
- Upgrade to new CE releases without the need of a full upgrade across the customer landscape
- Attract partners to build composite applications
- Standards support and interoperable SAP tools, concepts and guidelines
- Straightforward building of composite application which can be integrated into SAP and non-SAP systems supported by model driven technologies
- Translation of business needs into delivered solution is supported by model driven technologies

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

- Standard Compliance
  - Competitive J2EE offering
  - Low cost switch to SAP Java EE 5
  - Clean interoperability between standards and SAP concepts (e.g. Web Dynpro, CAF support of EJB)
  - Driving necessary standards to SAP's advantage (e.g. SDO, SCA)
  - Integration of standard tools in Eclipse
- Lean consumption
  - Increase productivity by at least factor 10
  - One single consistent stack and programming model
  - Decrease download and installation to less than 1 hour
  - Smooth learning curve by fewer concepts and easy to understand programming model
  - Reference implementation oriented rollout and knowledge transfer
- Development productivity
  - Integrate existing tools to improve the development efficiency (e.g. Web Dynpro, CAF, EJB interoperability)
  - Provide SAP Visual Composer kits for various classes of application (e.g. Voice, Analytics, etc.) as model driven, highly productive design environment

II. ARCHITECTURE

This architecture blueprint reflects the intended architecture of the SAP NetWeaver Composite Environment for Composition Environment 7.1.1.

The main idea is to lean up the NW java stack for composition. Therefore, the composition environment is designed and implemented as a usage type of the NW Java stack. The required consolidations happen in the same stack as the NetWeaver standalone release. Nevertheless not all NetWeaver components are taken into Composition Environment, at some places the Composition Environment plugs into the customer landscape and reuses the already existing components (see Figure 1: CE in the landscape). The most important components that are used on the customer side are:

- Portal: Composite Applications are federated into a customer Portal via Federated Portal Network (FPN).
- Knowledge Management (KM): the Composition Environment frameworks connect to a remote KM and get the content from these servers.
- BI: data can be retrieved from remote servers and used in Composite Applications.
- NetWeaver Development Infrastructure (NWDI): Composition Environment could host an NWDI, but it is also possible to configure a Composition Environment to use a remote NWDI.

Beside a lean runtime, a design time is very important to the ISVs. To provide a compelling offering it is important to reduce the knowledge entry barrier for ISV developers by:

- embrace community standards & best practices
- make SAP NW development infrastructure an optional offering
- provide a good tool support for leveraging the SAP application through web services

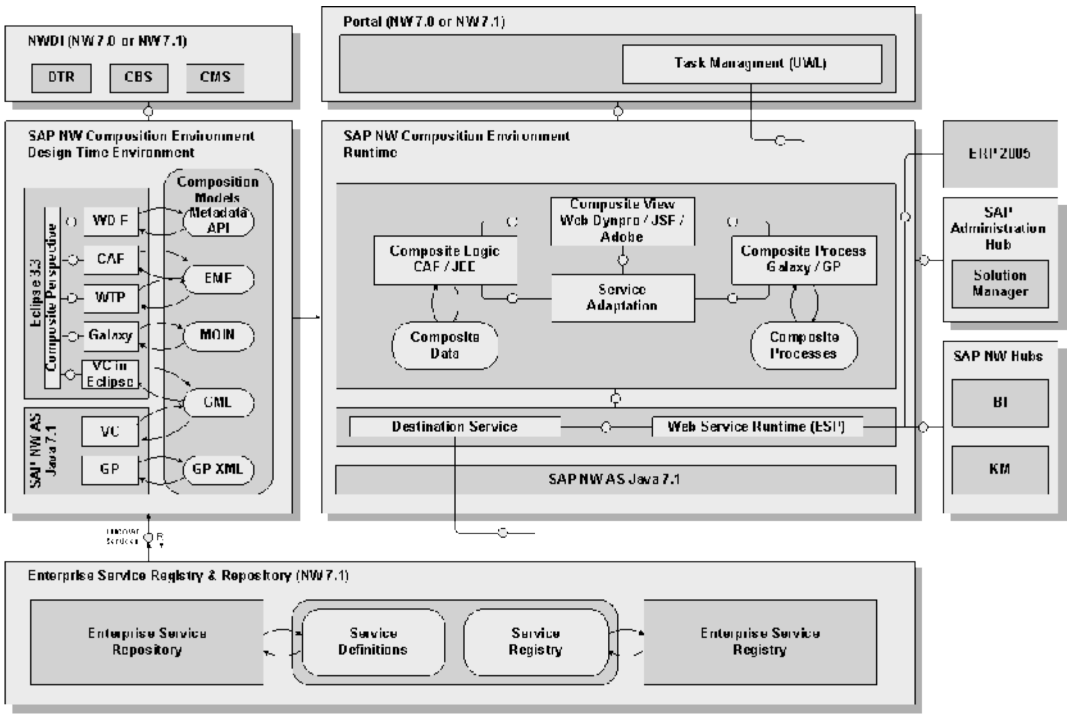

Figure 1: CE in the landscape

Main Architecture Concepts and Decisions

Overview of layers

A Composite Application is structured in a way that it contains content for specific purposes. Some parts are UI related, some parts defining the process flow, other parts are specific to the business logic. The common functionality that all frameworks provide is the consumption of Enterprise Services.

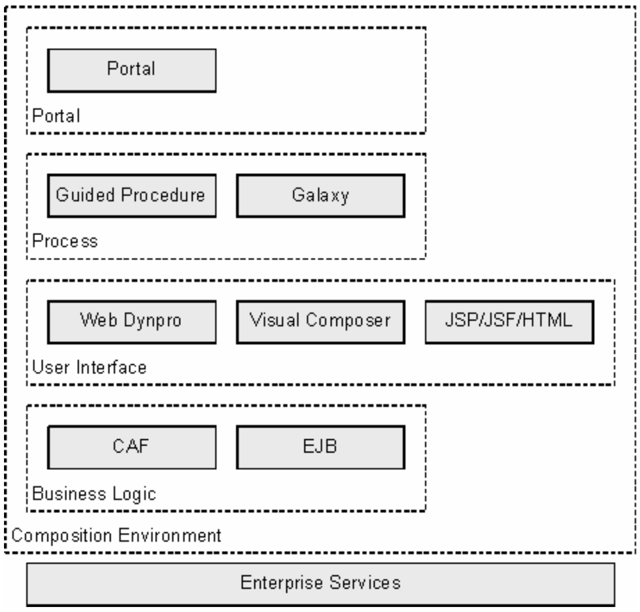

Figure 2: Layers in the Composition Environment

The main Composition Environment frameworks are:

- J2EE frameworks (EJB, JSP/JSF): This is the basic framework for Enterprise Java application and the Composition Environment supports all applications that are Java EE 5 compliant.
- Composite Application Framework (CAF): on top of the EJB framework SAP provides the functionality to define business objects and services in a model-driven way.
- Web Dynpro Foundation: Most of the UI applications of SAP are running with the Web Dynpro runtime.
- Visual Composer: Model-driven applications can be developed very efficiently with the Visual Composer in a completely model-driven way. The designtime is completely located on the server.
- Guided Procedure: Processes are currently developed with the Guided Procedure framework, because it provides the full feature-set that is expected for process definition on Composition Environment, including Adobe Interactive Forms for standardized user interfaces.
- Galaxy: Model-driven process definitions are done with the Galaxy framework. Currently it is not a full replacement of Guided Procedure, but it definitely a goal to be feature-complete to Guided Procedure.
- Portal: Normally applications are triggered via the Portal UI and therefore a Composite Application contains Portal entities, too.

Server architecture

The Composition Environment can not be divided into a runtime stack that runs on a server and a designtime that runs in Eclipse. For sure the Eclipse parts are all related to designtime purposes, but on the other side not all components on the server are only relevant for runtime. Complete designtime solutions are running on the server, mainly:

- Visual Composer
- Guided Procedures

Nevertheless the applications are running on the Java EE 5 stack and utilizing the standards. This means that the runtimes on top of the Java EE 5 standard are using the concepts of the Java server. Some examples are:

- The Web Dynpro runtime is integrated with the Web Container and every Web Dynpro application runs in the Web Dynpro servlet.
- The Guided Procedure runtime is on the one hand a set of Web Dynpro applications, on the other hand a set of Java EE 5 services.
- The Galaxy runtime is using the EJB container and is also running some services on the Java EE 5 server. Especially for execution of a process the cluster capabilities of the server are used.

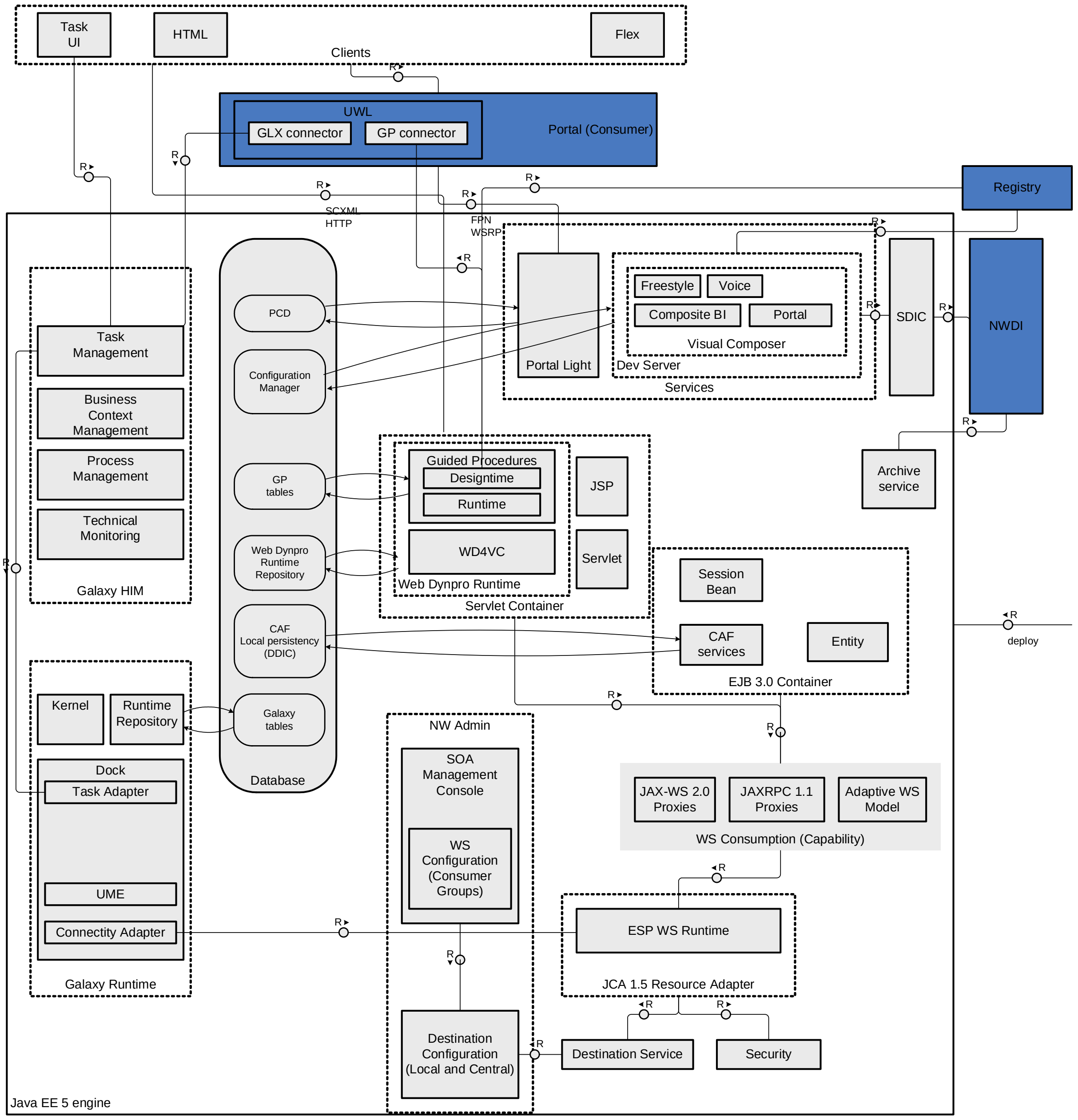

Figure 3: Architecture Overview Diagram (Static Structure at Server)

Eclipse architecture

Composite Applications are mainly developed in the Eclipse environment; Guided Procedure and Visual Composer is developed on the server. The available frameworks are used heavily to leverage as much as possible from the community. The specific Composition Environment frameworks have tools that suite best for their use cases to reduce the development time of an application.

As explained before there are some designtimes on the server, but the goal is to bring these toolsets to Eclipse. There are some improvements in this area:

- VC-in-Eclipse: The Visual Composer is running locally in Eclipse and it is not required any more connect to a server to model a UI or Portal content.
- Galaxy Workbench: The process modeling tool for Galaxy is completely Eclipse-based, in contrast to the Guided Procedure designtime before.

The toolset for the domain specific models are bundle via the Composite Designer. This tool provides a consistent overview of a Composite Application, showing the dependencies and check, if the contracts between the different objects in the different domains are violated.

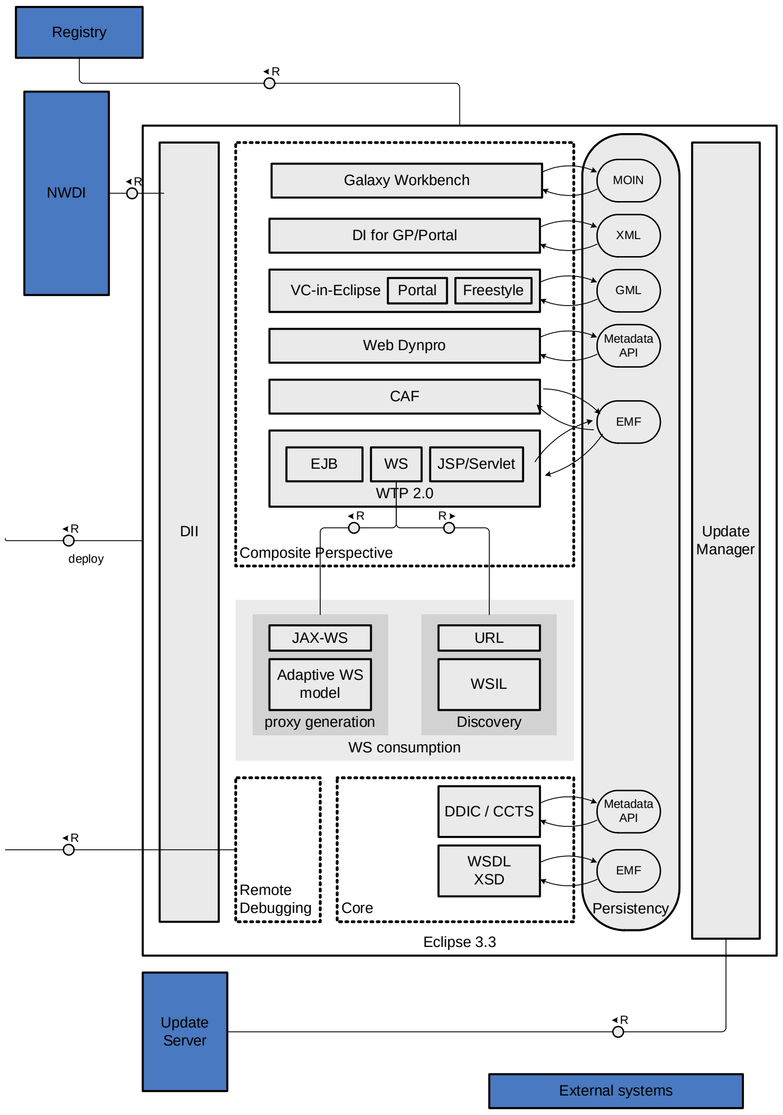

Figure 4: Architecture Overview Diagram (Static Structure in Eclipse)

It will also be possible to run Web Dynpro applications (Foundation and Visual Composer) in Eclipse in a specific sandbox, so that the overhead of deployment is not required on every roundtrip. This improves the developer productivity by reducing the turn-around cycles.

Lifecycle Management in the Composition Environment

The Composition Environment is a Java-stack solution, therefore the basic infrastructure is based on the SAP Component Model, e.g. Software Components, Development Components etc. It is possible to use the full NWDI of SAP including DTR, CBS, and CMS.

There are different phases regarding lifecycle management

- Setting up the Composition Environment: includes installation and basic configuration to be able to consume Enterprise Services.
- Developing a Composite Application: involves not only the tools for the domain models, it also contains the test cycles and infrastructure operations like check-in/check-out.
- Building/Shipping a Composite Application: is supported by NWDI and the Composite Environment leverages all investments that are done in this area, e.g. building SCAs.

- Consuming a Composite Application: The typical tools like Update Manager, Solution Manager etc. are used to bring a Composite Application to the customer system. Additionally a Composite Application has to be connected to the Enterprise Services in the customer backend system, and because of that the configuration can be done either manually or automatically based on some additional information that is deployed with the application.
- Maintaining a Composite Application: The Composition Environment contains a local Netweaver Administrator (NWA) to monitor and to trace the execution of applications, but it is also possible to connect the Composition Environment to a central NWA to monitor it centrally.

SOA in the Composition Environment

The Composition Environment supports SOA in two ways:

- Service Consumption: Enterprise Services can be consumed in all Composition Environment frameworks. This can be done directly or via the Services Registry to benefit from predefined taxonomies (based on application semantics). Additional information can be used to bundle the service usages of a Composite Application to configure them later in the customer landscape in one step (reducing TCO).
- Service Provisioning: Service definitions are done in the Enterprise Service Repository. The Composition Environments contains a browser to get these service definitions and generate the skeletons for the services.

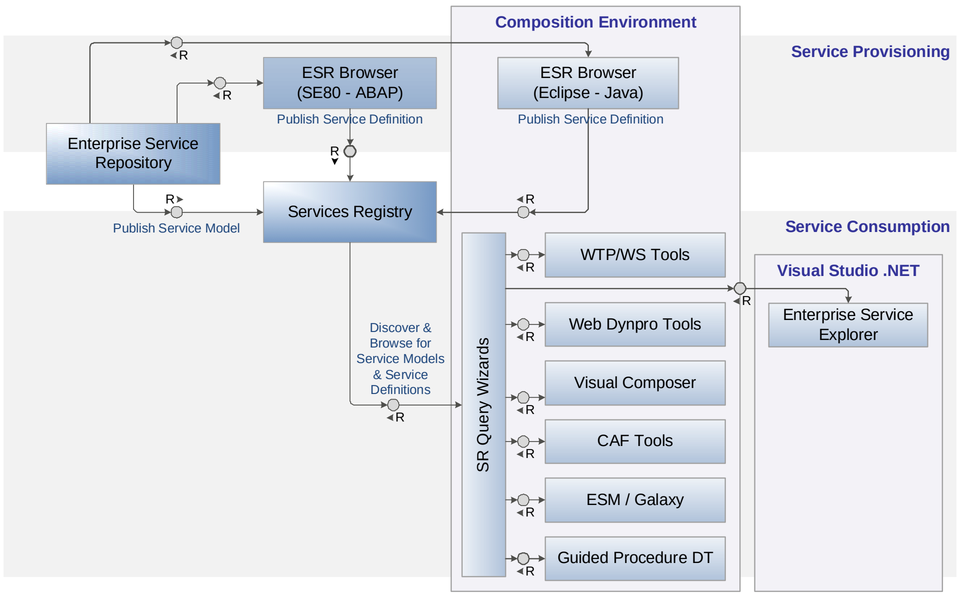
Composition Environment
Service Provisioning
ESR Browser (SE80 - ABAP)
ESR Browser (Eclipse - Java)
Publish Service Definition
Publish Service Definition
Enterprise Service Repository
Services Registry
Publish Service Model
Service Consumption
WTP/WS Tools
Visual Studio .NET
Enterprise Service Explorer
Web Dynpro Tools
Discover & Browse for Service Models & Service Definitions
SR Query Wizards
Visual Composer
CAF Tools
ESM / Galaxy
Guided Procedure DT
R

Figure 5: SOA in the Composition Environment

Decoupling of components

A very important goal of the Composition Environment is to decouple the frameworks in order to reduce the start-up time of the server and applications and reduce the memory-consumption. This is an ongoing effort and leads to architectural changes by introducing abstractions.

The most effort is done on the server to reduce the dependencies of components and use other frameworks only on demand, not via a static reference. This paradigm is also applied in the designtime in Eclipse, where the feature concept of Eclipse is used to structure the composition frameworks into features to download and download them on demand.

Total Cost of Ownership

- Architectural consistent, aligned programming model is the basis for reducing complexity, increasing the overall quality, and delivering high productivity.
- The approach to leaning up starts with standard compliance, which results in manageability, unification and finally reveals missing concepts.
- Alignment or loose coupling of all SAP concepts with industry standards
- Significant architecture changes where needed to achieve lean consumption. This is an on-going effort that is part of the usual project management.
- Integration-first approach for development
- Start from small core (JEE, WTP) with controlled, value-driven grow
- Introduction of a lean consumption quality gate for components to enter CE

Deployment

SAP NetWeaver CE supports the following installation options:

- Java Application Server (including NWA and Web Dynpro)
- Composite Views (including Visual Composer and Portal)
- Composite Processes (including Guided Procedures)
- Composite Applications (including CAF)
- Composite Voice (including Voice kit and runtime)
- CE Adobe Document Service (including Adobe Document Server)
- IDE Update Site (including Eclipse Update Server)

CE & Enterprise Services Registry:

- Publishing from Enterprise Services Repository into Enterprise Service Registry is not supported yet
- CE design time tools can consume multiple registries (e.g. hosted and local registry)
- At ISV side, local registry is initialized with content from hosted registry
- In addition, local registry is used to support CE development

Portal Federation:

- SAP proprietary ‘Server-to-Server’ protocol is implemented to provide personalization and configuration features
- Remote role assignment and role integration
- Remote content sharing

Maintenance

The Composite Environment is part of the overall Netweaver release, so the maintenance support is inherited from Netweaver.

Architecture Documentation

For further details and information, see Composition Program on SAP Corporate Portal