TUHH
Technische Universität Hamburg-Harburg

Enterprise Services Architecture and Composite Application Framework

Master Thesis

Mariusz Chechelski
Information and Media Technologies

Hamburg-Harburg University of Technology
Germany

Supervised by
Prof. Dr. J.W. Schmidt
Prof. Dr. V. Turau

STS Arbeitsbereich Softwaresysteme

Supervised by
Klaus-Georg Lemke
Rudi Grom

hp invent

Hamburg, April 2005

Declaration

Hereby I declare that I am the author of this thesis, titled “Enterprise Services Architecture and Composite Application Framework“. All literally or content related quotations from other sources are clearly pointed out and no other sources rather than the ones declared are used.

Mariusz Chechelski

Hamburg, April 25, 2005

Acknowledgement

I would like to thank Professor Schmidt and Rainer Marrone of Software System Group for supervising this thesis and giving valuable feedback for continuously enhancing the scope of this work. Thanks also to Professor V. Turau for being the co-supervisor of this thesis. Furthermore, I am very grateful to Klaus Lemke and Rudi Grom for offering the opportunity to work closely with HP SAP Center of Expertise and giving continuous support. Furthermore, I am very grateful to Mario Herger of SAP AG for offering the opportunity to work closely with CAF team in Walldorf.

Contents

List of Figures

1 Introduction

“It’s not the strongest of the species that survives, nor the most intelligent; but the one most responsive to change”

- Charles Darwin

The fragmented world of data and software would really not be much of a problem if business conditions remained constant, as they did in the early 1990s before the arrival of the Internet as a primary force of making business. Back then, most companies were self-contained units, impenetrable to the outside world, and IT systems were an internal affair. Information was essentially "dis-integrated" among various systems. Very often these systems defined the same business concepts like Customer in a Human Resource Management System (HRMS) application, Payroll application, and Benefits application, thus creating dissonance between these systems.

At the present time the network has brought suppliers and customers right inside that infrastructure. Customers frequently use computer interfaces to enterprise systems, often through the Internet. This computer-mediated interaction with customers applies not just in a few industries and contexts, like customers interacting with a bank through an ATM or with an airline reservations system through a web site, but it spread out to almost every company.

At the business-to-business (B2B) level, creating the most efficient supply chain involves opening up the core systems of the company and integrating them with vendor’s systems at key points. Therefore there is a pressure on business to expose their internal systems to the outside world as services. IT systems are thus no longer an internal affair. There is a shift under way in the role of information technology (IT) in the enterprise. It’s an evolution in how IT will enable companies to become truly adaptive, flexible, and able to both respond to and take advantage of change on a business market. IT organizations try to adopt step-by-step approach to evolve their current IT infrastructures so that it can deliver services, reliability and stability, speed and agility, and to achieve a better return on investment (ROI).

IT organizations must cope with the growing scope of services and its change needed to support a variety of business initiatives – information security, application integration, the development and deployment of services (web services) – while at the same time supporting vital everyday business needs ranging from compliance with regulatory changes to mergers and acquisitions, to changes in supply chain. In the quest to increase efficiency and gain competitive business advantage, IT organizations have embraced numerous methodologies and technologies over the past two decades. One among them, which aspire to address all of these above-mentioned challenges, is Enterprise Services Architecture (ESA) concept.

1.1 Objectives of this thesis

Enterprise Services Architecture (ESA) is being developed to address business drivers such as flexibility of IT systems, computer-mediated interaction (interoperability) with consumers, suppliers, regulators, financial institutions etc. in order to enable business innovation while lowering total cost of ownership (TCO).

Therefore, the aims of this thesis are to identify the major changes concerning the organizational and implementations methodologies when using the ESA's approach and to identify the areas that might give business benefits using the new approach of interfacing with Package Composite Applications. This thesis will concentrate on methodologies that enable to build process-based applications like package composite applications that try to reflect business processes within a company. Furthermore, there will be executed an investigation in order to position and find out about technical aspects for enterprise services, which are brought with ESA concept.

In order to accomplish the aims, the objective of this thesis is the following:

- To build an application with a Web Service Interface with an application of SAP NetWeaver and SAP xApps technologies those are SAP's technical foundations for Enterprise Services Architecture concept
- To discuss ESA concepts regarding the Service-Oriented Architecture (SOA) approach

1.2 Structure of this thesis

This thesis is organized as follows:

Chapter 2 will present important and related concepts aimed to organize business functions and IT infrastructure in order to increase efficiency and to gain competitive business advantage of an enterprise taken up by IT organizations over the past two decades. This chapter will look briefly at the Web Services concept, which as a first technology brought a piece of functionality taken from a company's business processes or infrastructure and made it accessible over the Internet. Then this chapter will look briefly also at the Service-Oriented Architecture (SOA), which goes beyond Web Services concept and not only expose single services over a network but also focuses on organizing business systems as reusable components, not fixed processes. In some depth this chapter will introduce Enterprise Services Architecture that is an application of a service-oriented architecture and sound principles of object-oriented design applied to the current heterogeneous world of IT architecture focused by economic reality. This chapter also presents a comparison of SOA and ESA concepts. Finally, the Package Composite Applications (PCAs) concept as part of ESA will be described. PCAs reflect a certain process that exist within enterprise infrastructure and presents it as new business applications.

Chapter 3 contains the most relevant technologies and techniques applied for this thesis. This chapter will describe the application of Enterprise Services Architecture with the latest SAP Technology named SAP NetWeaver that is designed to integrate the current legacy applications and to build package composite applications called SAP xApps (Collaboration Cross Applications) with a use of the SAP Composite Application Framework (CAF).

Based on a real life business scenario for maintaining the customer master data at Hewlett-Packard Company, chapter 4 describes the design of a custom composite application with the SAP Composite Application Framework (CAF). This chapter will outline the technical aspects of designing package composite applications with the SAP NetWeaver technology that offers a new design approach for user web interfaces.

Chapter 5 contains a discussion about the results obtained during the research phase of this project. It will be described the major changes concerning the organizational and implementations technologies while using the ESA approach. Among others things it will be discussed a new Web services paradigm, namely Enterprise Services that want to enhance Web services technology in order to support enterprise-level business functionality.

Finally, this thesis ends with a conclusion in chapter 6, providing conclusion remarks, summarizing the key results of the presented thesis with regard to Enterprise Services Architecture as a service-based architecture.

2 Concepts and Philosophies

"Ancient civilizations made three principal contributions to the development of architecture. One was the perfection of two structural systems, the post and lintel and the arch, and their use as decorative as well as structural elements. Another was a multitude of decorative forms and patterns, many of which passed into the architectural heritage of Western Civilization and are still in use today. The third was the concept of orderly planning… the most important of these contributions is perhaps the last, a plan is fundamental in architecture"

- Joseph Watterson

In the first chapter it was underlined that companies (IT organizations) are becoming aware that IT should help by automating and optimizing the processes of a business. As a result, IT infrastructures (enterprise applications) have evolved from mainframe systems that offered rock-solid stability and reliability to client/server systems that emphasis speed since companies began automating front office towards partners and customers because of the Internet and cooperation with them through the network.

This chapter presents the concepts and philosophies (Figure 1) taken up by IT organizations over the past two decades that aim to establish a tight partnership between business and IT, and in turn delivers greater business agility and a greater return on investments of an enterprise. Also this chapter will focus primarily on Enterprise Services Architecture, which at the present time is regarded by most enterprises and software vendors as the more promising approach for future enterprise applications. Therefore, in this chapter will be described the general principles, advantages and drawbacks of Enterprise Services Architecture.

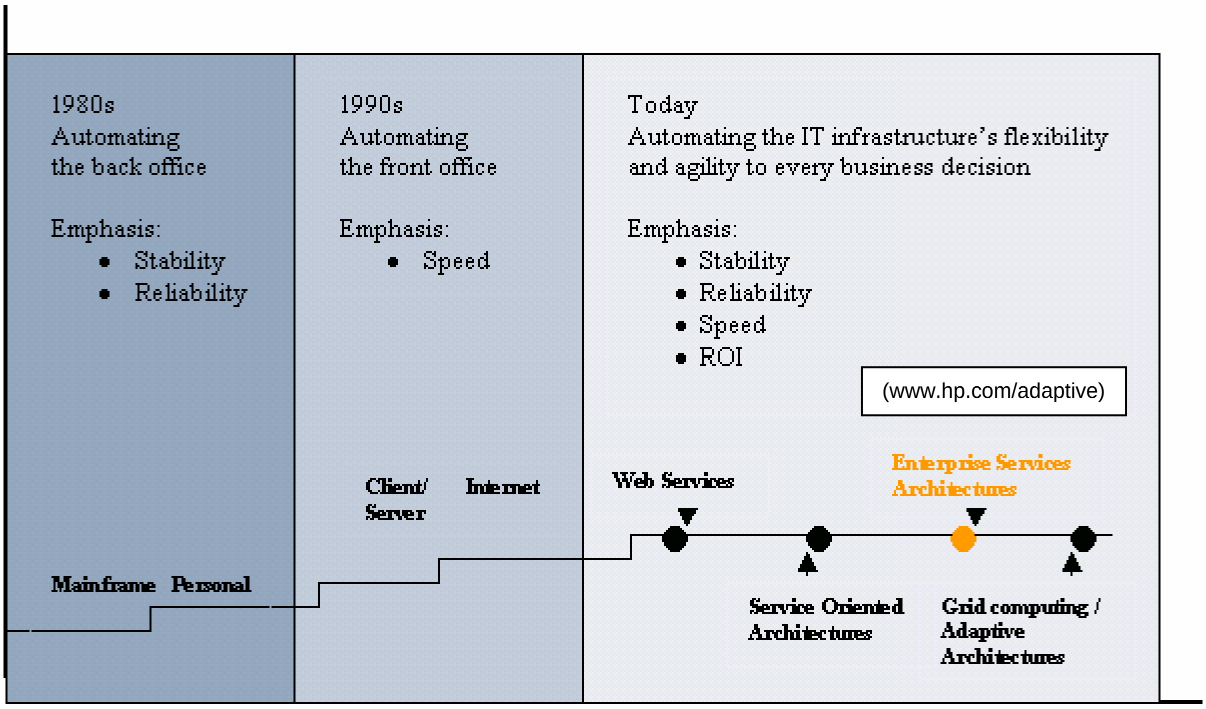

1980s
Automating the back office
Emphasis:
• Stability
• Reliability
1990s
Automating the front office
Emphasis:
• Speed
Today
Automating the IT infrastructure's flexibility and agility to every business decision
Emphasis:
• Stability
• Reliability
• Speed
• ROI
(www.hp.com/adaptive)
Mainframe
Personal
Client/ Server
Internet
Web Services
Service Oriented Architectures
Enterprise Services Architectures
Grid computing / Adaptive Architectures

Figure 1 Evolution of IT infrastructures leaded through ROI demand

2.1 Web Services

This section outlines basis of Web Service technology and its contribution to evolution of IT technologies against business values.

The main concept of this technology is the redefinition of a software component as a Service. The World Wide Web Consortium (W3C) defines:

"*A service is an abstract resource that represents a capability of performing tasks that represents a coherent functionality*" [W3C].

Web Services represents a new platform on which developers can build the same distributed applications they have always built, but this time with interoperability as the highest priority. Interoperability has always been a major concern for organizations, but over the past decade it has become a much bigger priority across the industry. There are two main areas where interoperability is a significant challenge: Enterprise Application Integration (EAI) and Business-to-Business Integration (B2Bi).

EAI represents the challenge most enterprises face in integrating their various applications with each other. B2Bi represents the business interactions between different enterprises. If one business wants to purchase supplies from another, they have to interact and exchange information—and they rarely happen to be using the same technology suites. Many organizations want to extend their reach to users, so interoperability becomes an even bigger challenge than ever before.

Protocols for connecting enterprise applications have been around for quite some time, but developers have typically encountered obstacles when designing according to these models. Proprietary protocols force developers to think in a product-specific way and learn special programming APIs in order to achieve interoperability. Therefore, there was a force on IT to provide a standardized way for connecting enterprise applications that should be on wide world scale adopted by all IT organizations, software vendors and so on. Web services have met unexpectedly in comparison with other technologies like RFC, CORBA/DCOM this requirement over the past few years and have been standardized in committees including WS-I[1] and W3C[2].

Web Services allow the integration of multiple software platforms and any types of networks. This is due to the fact that Web Services are based on a standardized set of technologies, such as: eXtensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Service Description Language (WSDL) and Hyper Text transport Protocol (HTTP). The usage of widely spread and platform independent XML format and SOAP protocol offers the possibility of accessing enterprise applications through services, which are requested by web clients from wherever in the world (i.e. interoperability). While traditional applications interacting with services in the Internet know those services deductively and need

[1] The Web Services Interoperability Organization (WS-I) is an open, industry organization chartered to promote Web services interoperability across platforms, operating systems, and programming languages. http://www.ws-i.org/

[2] The World Wide Web Consortium (W3C) develops interoperable technologies (specifications, guidelines, software, and tools) to lead the Web to its full potential. W3C is a forum for information, commerce, communication, and collective understanding. http://www.w3.org/

to be pointed to them manually, Web services let applications find Web services in a standardized directory structure and bind to the services with minimal human interaction. Figure 2 presents the Web Services concept.

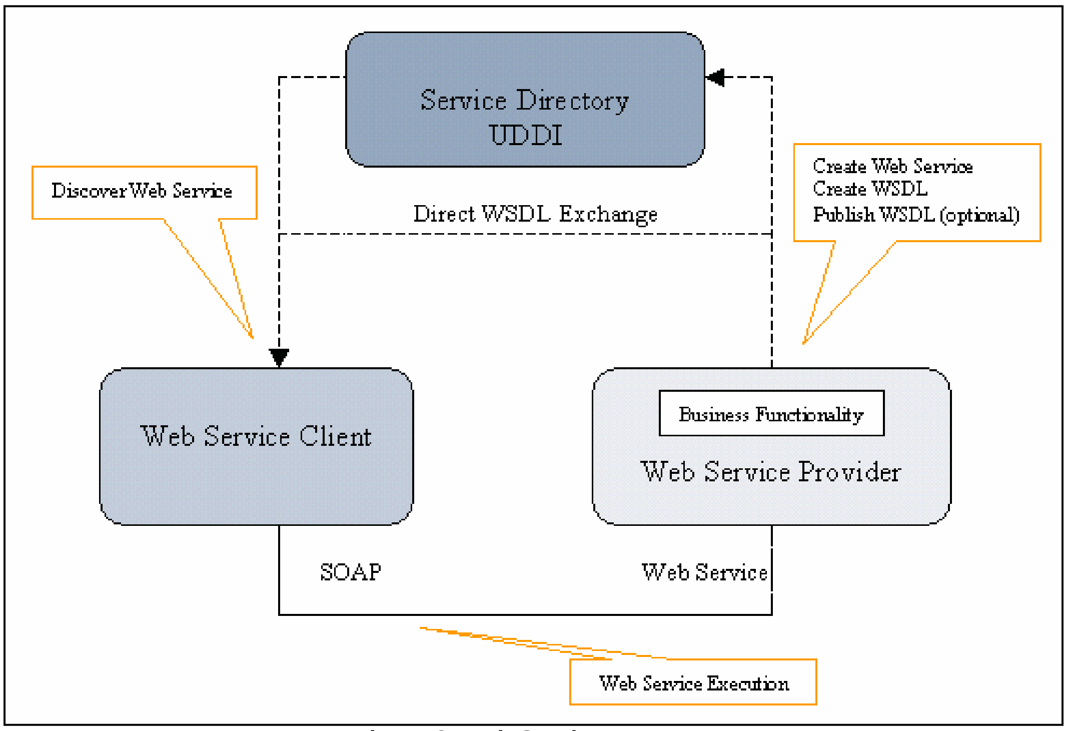


Figure 2 Web Services concept.

In the Web services concept, providers and consumers of services represent the world. The Web Service provider develops a Web service in a certain programming language and deploys it to its own server runtime environment. The service is described in the Web Services Description Language (WSDL) using special XML tags. The service description is published in a common service directory. Web Services directories are generally organized following the UDDI[3] specification. A developer on the Web Service client (consumer) side can browse the UDDI directory and look for applicable services. The client (consumer) may then download the WSDL document of a selected Web service and trigger the execution of the Web service over the communication link that is established between the client and the provider. Web service invocations are standardized using SOAP, while SOAP request contains the name of the Web service plus its actual parameters. A SOAP response contains the result parameters based on the signature that is exposed in the WSDL. It is worth to note that in a Web service scenario, the use of the service directory is optional; if client knows where a Web service runs and client obtains the description directly from the Web service provider as a result client can invoke the Web service without using the service directory.

[3] Universal Description, Discovery, and Integration (UDDI) is a protocol for registration and discovery of Web services. http://www.uddi.org

The goal of Web services is to achieve universal interoperability between applications by using Web standards. So far, we have described the Web services concept and its most related standards like XML, SOAP, WSDL, and UDDI. Figure 3 presents a Web services architecture that involves much more layered and interrelated technologies. The most important standards like services coordination, transactions or security in the context of building distributed systems will be presented in the following part of this section.

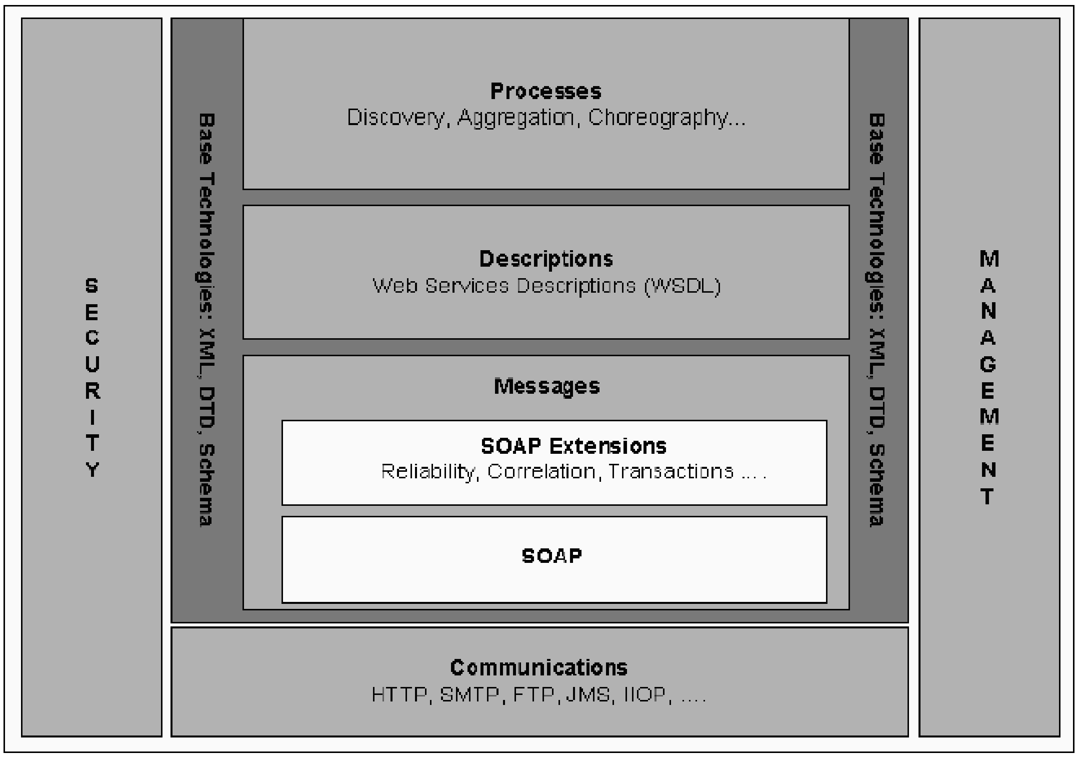

Processes
Discovery, Aggregation, Choreography...
Descriptions
Web Services Descriptions (WSDL)
Messages
SOAP Extensions
Reliability, Correlation, Transactions ....
SOAP
Communications
HTTP, SMTP, FTP, JMS, IIOP, ...
SECURITY
MANAGEMENT
Base Technologies: XML, DTD, Schema
Base Technologies: XML, DTD, Schema

Figure 3 Web Services Architecture (W3C)

Let’s have a look at a small practical Web Service example and how WSDL works with SOAP. Assume you are the client behind the imaginary company snowboard-info.com, a snowboarding industry database providing a service that allows others to query endorsements from snowboard manufacturers. You can as a client send a request which professional snowboarder endorses the K2 FatBob model. (SOAP 1.1 message – Listing1) to retrieve this information from a server.

```
POST /EndorsementSearch HTTP/1.1
Host: www.snowboard-info.com
Content-Type: text/xml; charset="utf-8"
Content-Length: 261
SOAPAction: "http://www.snowboard-info.com/EndorsementSearch"
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
<m:GetEndorsingBoarder xmlns:m="http://namespaces.snowboard-info.com">
<manufacturer>K2</manufacturer>
<model>Fatbob</model>
</m:GetEndorsingBoarder>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listing 1

In response, the server can send the SOAP 1.1 response (HTTP header) message for the foregoing request as shown in Listing 2. In natural language, it encapsulates the simple string response "Chris Englesmann" who is the professional snowboarder that endorses the K2 FatBob model.

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
   <m:GetEndorsingBoarderResponse xmlns:m="http://namespaces.snowboard-info.com">
     <endorsingBoarder>Chris Englesmann</endorsingBoarder>
   </m:GetEndorsingBoarderResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listing 2

Now the overall structure of requests, the relevant data types, the schema of the XML elements used, and other such matter are left to the trading partners by the SOAP specification itself. The communications between these trading partners can be described in structured way with use of WSDL standard. WSDL document provides a standard for service specification that unites the types of requests and the requirements needed to process them (Listing 3). Its meaning is as follows:

- The <document> element describes a set of related services.
- The <types> element allows the specification of low-level data-typing for the message or procedure contents.
- The <message> element defines the data format of each individual transmission in the communication.
- The <portType> element groups messages that form a single logical operation. For instance, in our case, we can have an *EndorsingBoarder* request which triggers an *EndorsingBoarder* response, or in case of error or exception, an *EndorsingBoarderFault*.
- The <binding> element is the bit that firmly provides the connection between logical and physical model.
- The final element, <service>, defines a physical location for a communication end-point. It uses the port type and binding specified earlier, and basically gives the Web address or URI for a particular provider of the described service.

```
<?xml version="1.0"?>
<definitions name="EndorsementSearch"
  targetNamespace="http://namespaces.snowboard-info.com"
  xmlns:es="http://www.snowboard-info.com/EndorsementSearch.wsdl"
  xmlns:esxsd="http://schemas.snowboard-info.com/EndorsementSearch.xsd"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
  <types>
    <schema targetNamespace="http://namespaces.snowboard-info.com"
        xmlns="http://www.w3.org/1999/XMLSchema">
      <element name="GetEndorsingBoarder">
        <complexType>
         <sequence>
            <element name="manufacturer" type="string"/>
            <element name="model" type="string"/>
          </sequence>
        </complexType>
      </element>
      <element name="GetEndorsingBoarderResponse">
        <complexType>
           <all>
            <element name="endorsingBoarder" type="string"/>
           </all>
        </complexType>
      </element>
      <element name="GetEndorsingBoarderFault">
        <complexType>
           <all>
            <element name="errorMessage" type="string"/>
           </all>
        </complexType>
      </element>
    </schema>
  </types>
  <message name="GetEndorsingBoarderRequest">
    <part name="body" element="esxsd:GetEndorsingBoarder"/>
  </message>
  <message name="GetEndorsingBoarderResponse">
    <part name="body" element="esxsd:GetEndorsingBoarderResponse"/>
  </message>
  <portType name="GetEndorsingBoarderPortType">
    <operation name="GetEndorsingBoarder">
      <input message="es:GetEndorsingBoarderRequest"/>
      <output message="es:GetEndorsingBoarderResponse"/>
      <fault message="es:GetEndorsingBoarderFault"/>
    </operation>
  </portType>
  <binding name="EndorsementSearchSoapBinding"
           type="es:GetEndorsingBoarderPortType">
    <soap:binding style="document"
                  transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="GetEndorsingBoarder">
      <soap:operation
        soapAction="http://www.snowboard-info.com/EndorsementSearch"/>
      <input>
        <soap:body use="literal"
namespace="http://schemas.snowboard-info.com/EndorsementSearch.xsd"/>
      </input>
      <output>
        <soap:body use="literal"
namespace="http://schemas.snowboard-info.com/EndorsementSearch.xsd"/>
      </output>
      <fault>
        <soap:body use="literal"
namespace="http://schemas.snowboard-info.com/EndorsementSearch.xsd"/>
      </fault>
    </operation>
  </binding>
  <service name="EndorsementSearchService">
    <documentation>snowboarding-info.com Endorsement Service</documentation>
    <port name="GetEndorsingBoarderPort"
          binding="es:EndorsementSearchSoapBinding">
      <soap:address
location="http://www.snowboard-info.com/EndorsementSearch"/>
    </port>
  </service>
</definitions>
```

Listing 3

The basic Web services infrastructure with standards such as XML, SOAP, HTTP, and WSDL suffices to implement simple interactions as that one in the example presented above. In particular, it supports interactions where the client invokes a single operation on a Web service. In real business applications, interactions are typically more complex than single, independent invocations. Using a particular service typically involves performing sequences of operations in a particular order. Sometimes, these sequences of operations might even involve more than one Web service. For example, consider a *supplier* Web service where clients can connect to buy certain items. As part of the purchasing procedure, clients typically have to identify themselves, request a quote for prices and delivery time, place the order according to the quote received, and submit the payment (Figure 4). All of these operations are necessary and they must be performed in a given order.

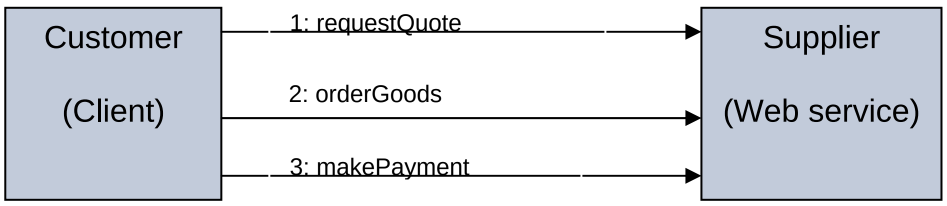


Figure 4 A sample conversations between a client and a Web service.

This presented simple scenario addresses important features that must be realized by Web services technologies. Firstly, the coordination among a number of Web services must be assured through introduction of additional protocol. Than additional requirements for consistency i.e. transactions and security feature must be supported through definition of new protocols. Adoption such new protocols for coordination, transactions and security may leverage web services technology to a prime technology for e-business commercial application. Currently there are in progress specification that initially propose standards for a coordination, transactions and security of web services, namely

- WS-Coordination protocol
- WS-Transactions protocol
- and WS-Security protocol

The primary goal of WS-Coordination is to create a framework for supporting coordination protocols. In this regard, it is intended as a meta-specification that will govern specifications that implement concrete forms of coordination e.g. transactional coordination. This specification describes a framework for a coordination service (or coordinator) which consists of these component services:

- An Activation service with an operation that enables an application to create a coordination instance or context.
- A Registration service with an operation that enables an application to register for coordination protocols
- A coordination type specific set of coordination protocols.

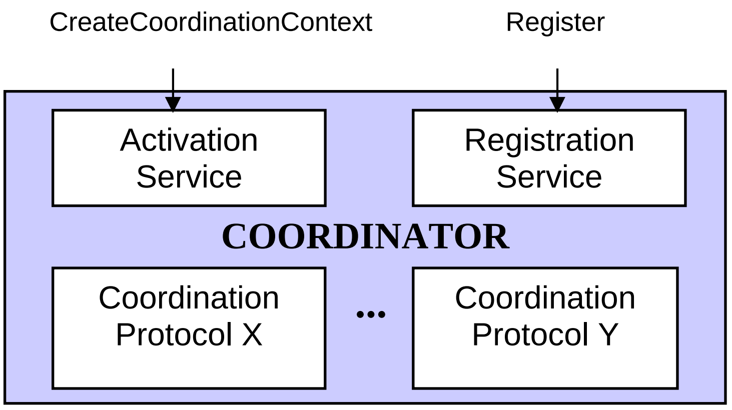


Figure 5 WS-Coordination protocol[4]

Applications use the Activation service (Listing 4) to create the coordination context for an activity. Once a coordination context is acquired by an application, it is then sent by whatever appropriate means to another application. The context contains the necessary information to register into the activity specifying the coordination behavior that the application will follow. Additionally, an application that receives a coordination context may use the Registration service of the original application or may use one that is specified by an interposing, trusted, coordinator. In this manner an arbitrary collection of web services may coordinate their joint operation.

```
ACTIVATION SERVICE:

<wsdl:portType name="ActivationCoordinatorPortType">
    <wsdl:operation name="CreateCoordinationContext">
      <wsdl:input message="wscoor:CreateCoordinationContext"/>
    </wsdl:operation>
  </wsdl:portType>
```

Listing 4

```
RESPONSE ACTIVATION SERVICE:

<wsdl:portType name="ActivationRequesterPortType">
    <wsdl:operation name="CreateCoordinationContextResponse">
      <wsdl:input message="wscoor:CreateCoordinationContextResponse"/>
    </wsdl:operation>
    <wsdl:operation name="Error">
      <wsdl:input message="wscoor:Error"/>
    </wsdl:operation>
  </wsdl:portType>
```

Listing 5

WS-Coordination addresses several fundamental issues in coordination among Web services:

- It defines SOAP extensions that are necessary to achieve coordination
- It defines meta-protocols for creating coordination context (Activation) and for binding coordinators and participants context (Registration)

[4] [WebServices-Alonso]

Similar forms of coordination are specified in traditional middleware systems such as CORBA. For example, the coordination scheme in WS-Coordination is very similar to the one described for CORBA's OTS.

WS-Transaction defines a set of protocols that require coordination among multiple parties. Therefore, it naturally builds upon the WS-Coordination framework (Figure 6). This specification provides the definition of two coordination types including their respective protocols for:

- An atomic transaction (AT) is used to coordinate activities having a short duration and executed within limited trust domains. They are called atomic transactions because they have an "all or nothing" property. The Atomic Transaction specification defines protocols that enable existing transaction processing systems to wrap their proprietary protocols and interoperate across different hardware and software vendors. Atomic Transaction protocol is built out of five protocols: *Completion, 2PC, ComplietionWithAck, PhaseZero, and OutcomeNotification*.
- A business activity (BA) is used to coordinate activities that are long in duration and desire to apply business logic to handle business exceptions. The long duration prohibits locking data resources to make actions tentative and hidden from other applications. Instead, actions are applied immediately and are permanent. The Business Activity specification defines two protocols: *BusinessAgreement* and *BusinessAgreementWithComplete* that enable existing business process and work flow systems to wrap their proprietary mechanisms and interoperate across trust boundaries and different vendor implementations.

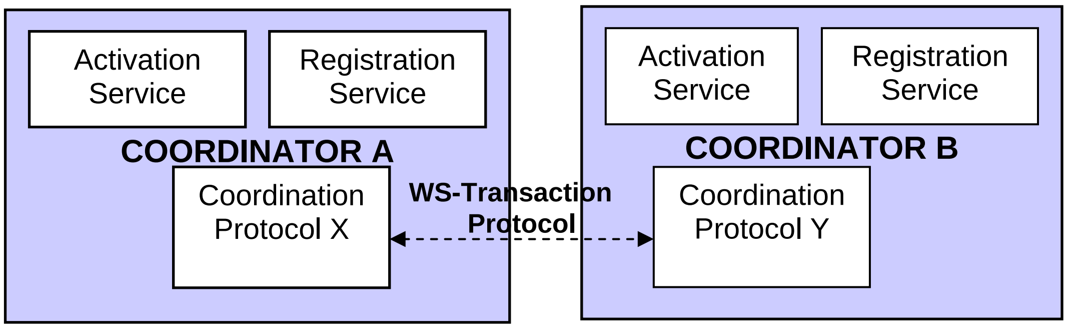


Figure 6 WS-Transaction protocol

Web services technology can enable building loosely coupled applications that can be assembled from a set of web services that are distributed over a connected infrastructure. The distributed nature of service-oriented applications addresses security concerns as critical success factor. The primary concern is to establish an interoperable framework that enables security for services, applications, and users in a trusted environment and complies with established corporate policies. Since a security paradigm is a wide range area this thesis only will outline the ongoing security standards for web services security issues.

WS-Security is an OASIS standard. WS-Security describes enhancements to SOAP messaging to provide message integrity, message confidentiality, and message authentication. WS Security uses XML Signature to provide message integrity and message authentication and uses XML Encryption to provide confidentiality. WS Security also provides a general-purpose mechanism for associating security tokens with messages. Examples of security tokens are X.509

certificate, SAML assertion. JSR 183 (Web Services Message Security APIs) defines a standard set of APIs for web services message security.

XML Encryption is a W3C recommendation. It ensures confidentiality of XML information transfers. XML Encryption allows the parts of an XML document to be encrypted while leaving other parts open. WS Security provides processing rules for using XML Signature for SOAP messages. JSR 106 (XML Digital Encryption APIs) defines a standard set of APIs for XML digital encryption services.

XML Signature is a W3C recommendation. It ensures message integrity and authentication. WS Security provides processing rules for using XML Signature for SOAP messages. Signature can be applied over parts of an XML document. JSR 105 (XML Digital Signature APIs) defines a standard set of high-level implementation-independent APIs for XML digital signature services.

Having presented Web Services concept, let's have a discussion how both business and IT departments can potentially benefit from it in terms of ROI. Web Services have brought a simplified mechanism to connect enterprise applications regardless of the technology. Thus, web services potentially improve business process efficiency by reducing cost and particularly time to connect enterprise applications i.e. they enable remote access to core source of information (real time business). Also, Web services can reduce the high cost of private networks, coupled with the cost of proprietary EDI/B2B solutions and as a result small, medium companies can enter a business market on the network (globalization). Since today organizations use different technologies for distributed computing, EAI, EDI, B2B, Websites, Portals so this results in n-times products, tools, skills and cost. Web Services provides an opportunity to radically reduce this by supporting these different scenarios with the same basic protocol stack thus IT benefits in cost savings through consolidation. Whilst Web Services remove many of the technology constraints of communication between applications providing flexibility at the implementation layer, the business agility that is promised is more a factor of service design than protocol adoption. Therefore, IT organizations has started a quest for a regulation by which we ensure that services (web services) are the right services, delivered at appropriate levels of granularity, abstraction and generality that makes sense to both Service Provider and Service Consumer, reduces the effort (particularly on the client) to use a set of services to perform a particular objective. The new approach is known as Service-Oriented Architecture (SOA) and the following section 2.2 will present it.

2.2 SOA – Service Oriented Architecture

This section outlines basis of Services-Oriented Architectures (SOA) concepts and its contribution to evolution of IT technologies against business values. Since SOA is ongoing research how to align IT infrastructure with business demands, in this section it will be made endeavor to describe SOA concepts as a layered architecture.

A service-oriented architecture is a style of design that guides all aspects of creating and using business services throughout their lifecycle (from conception to retirement). An SOA is also a way to define and provision an IT infrastructure to allow different applications to exchange data and participate in business processes, regardless of the operating systems or programming languages underlying those applications.

SOA is the architectural style that supports loosely coupled services to enable business flexibility in an interoperable, technology-agnostic manner. SOA consists of a composite set of business-aligned services that support a flexible and dynamically re-configurable end-to-end business processes realization using interface-based service descriptions.

In contrast, earlier approaches to building IT systems tended to directly use specific implementation environments such as object orientation, procedure orientation, and message orientation to solve business problems, resulting in systems that were often tied to the features and functions of a particular execution environment technology such as CICS, IMS, CORBA, J2EE, and COM/DCOM.

Service-oriented development, which SOA enables, is an evolutionary software engineering approach enabled by component-based and object-oriented development. The concepts behind SOA are not new. The idea of separating an interface from its implementation to create a software service definition has been well proven in J2EE, CORBA, and COM. But the ability to more cleanly and completely separate a service description from its execution environment is new. This ability is part of what Web concepts and technologies bring to Web services. The traditional implementations of the interface concept might not have considered such a "loose" separation because the performance implications are negative. However, in many cases, the performance issue is less important than the ability to more easily achieve interoperability, something the industry has long strived for but only partially achieved until now.

The major advantages of implementing an SOA using Web services are that Web services are pervasive, simple, and platform-neutral. As shown in following figure, the basic Web services architecture consists of specifications (SOAP, WSDL, and UDDI) that support the interaction of a Web service requester with a Web service provider and the potential discovery of the Web service description. The provider typically publishes a WSDL description of its Web service, and the requester accesses the description using a UDDI or other type of registry, and requests the execution of the provider's service by sending a SOAP message to it.

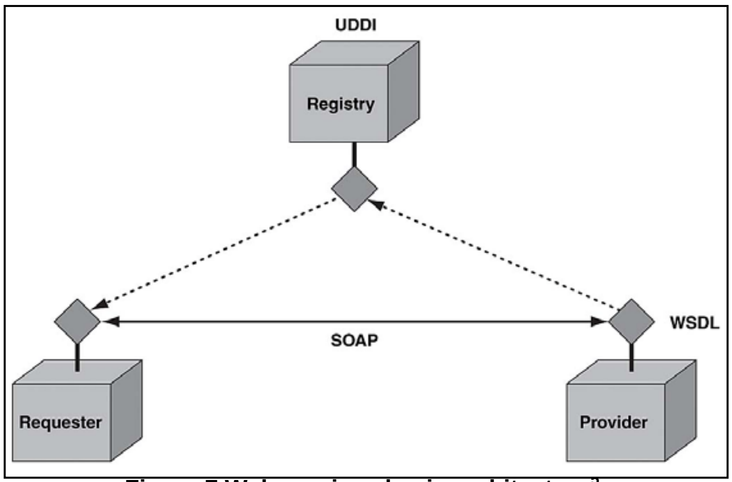


Figure 7 Web services basic architecture[5]

5 http://www.sys-con.com/webservices/

The major components of an SOA are:

- Services portfolio: Describes the business services in SOA. This includes a list, classification and hierarchy of services defined through the technique of service-oriented analysis and design.
- Components: Provide the functional realization of the services.
- Service providers, service consumers, and optionally, the service broker(s): With their service registries where service definitions and descriptions are published.
- SOA layers: Where software components and services reside.

Figure 8 illustrates the Service-Oriented Architecture (Services layer, Business Process layer, Presentation layer).

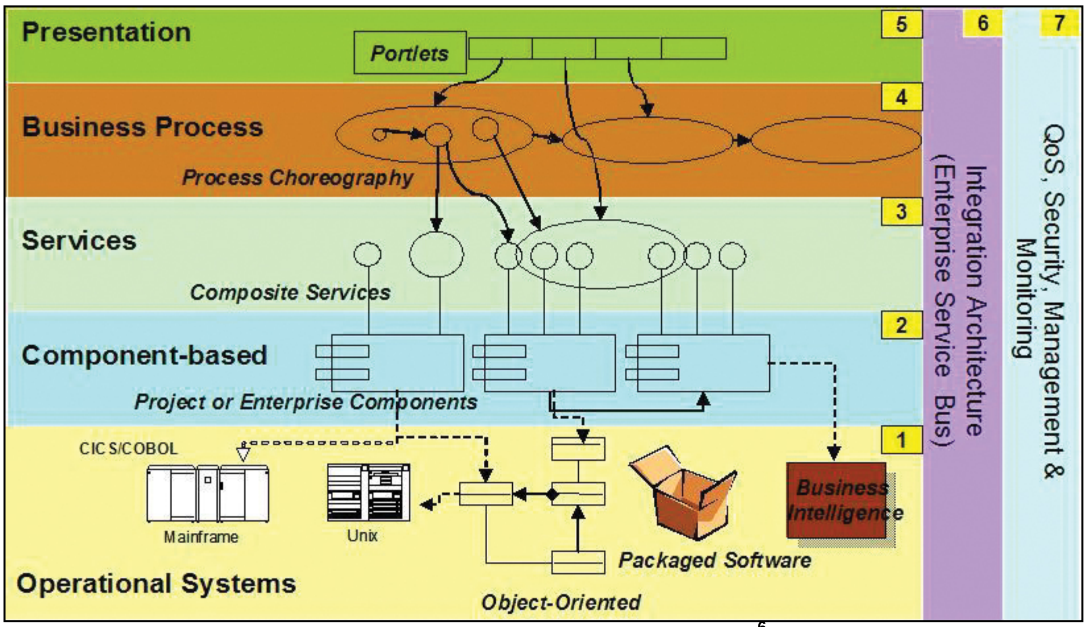


Figure 8 Service-Oriented Architecture [6]

Operational Systems Layer describes operational systems. This layer contains existing systems or applications, including existing CRM and ERP packaged applications, legacy applications, and (non-) object-oriented system implementations, as well as business-intelligence applications. The composite layered architecture of an SOA can leverage existing systems, integrate them using service-oriented integration.

Component Layer uses container-based technologies and designs in typical component-based development. Today, component-based technologies such as Enterprise Java Beans (EJB), .NET and CORBA are effective ways of implementing and managing software components.

Services Layer presents enterprise-scale components, business unit specific components, and in some cases project-specific components and provides services through their interfaces. The interfaces get exported out as service descriptions in this layer, where services exist in isolation or as composite services.

[6] http://www.sys-con.com/webservices/

Business Process Layer is an evolution of service composition into flows or choreographies of services bundled into a flow to act as an application. These applications support specific use cases and business processes. Here, visual flow composition tools (e.g. SAP XI Integration Builder) can be used for design of application flow and thus process-based application can be build.

Presentation Layer is usually out of scope for an SOA. However, it is depicted because some recent standards such as Web Services for Remote Portlets version 2.0 may leverage Web services at the application interface or presentation level. It is also important to note that SOA decouples the user interface from the components.

Integration Architecture Layer enables the integration of services through the introduction of reliable and intelligent routing, protocol mediation, and other transformation mechanisms, described as the Enterprise Service Bus.

The last layer ensures quality of service through sense-and-respond mechanisms and tools that monitor the reliability, security of SOA applications, including the all-important standards implementations of WS-Management.

SOA leverages the Web Services technology. The SOA design approach focuses on organizing business systems as reusable components and not as fixed processes, as it is in the case of Web Services. Business processes in SOA are defined in the Business Process layer.

In the context of Service-Oriented Architectures, business process specifies the potential execution order of operations from a collection of Web services, the data shared between Web services, which partners are involved and how they are involved in the business process, joint exception handling for collections of Web services, and other issues involving how multiple services and organizations participate. Business processes in a Web services world can be described by Business Process Executive Language for Web Services (BPEL4WS). It is an initiative of the industry leaders BEA Systems, Microsoft, IBM, SAP AG and Siebel Systems to drive and ensure interoperability for description and communication of business processes based on Web services. Processes in BPEL4WS are exported and imported functionalities through using Web services exclusively. BPEL4WS is layered on top of several XML specifications:

- WSDL 1.1
- XML Schema 1.0(XSLD)
- XPath 1.0

WSDL messages and XML Schema type definitions (XSLD) provide the data model used by BPEL4WS processes. XPath provides support for data manipulation. All external resources and partners are represented as WSDL services. BPEL4WS provides extensibility to accommodate future versions of these standards, specifically the XPath and related standards used in XML computation.

The BPEL4WS model is built upon a number of layers, with each layer building on the facilities of the previous. This is shown in Figure 9.

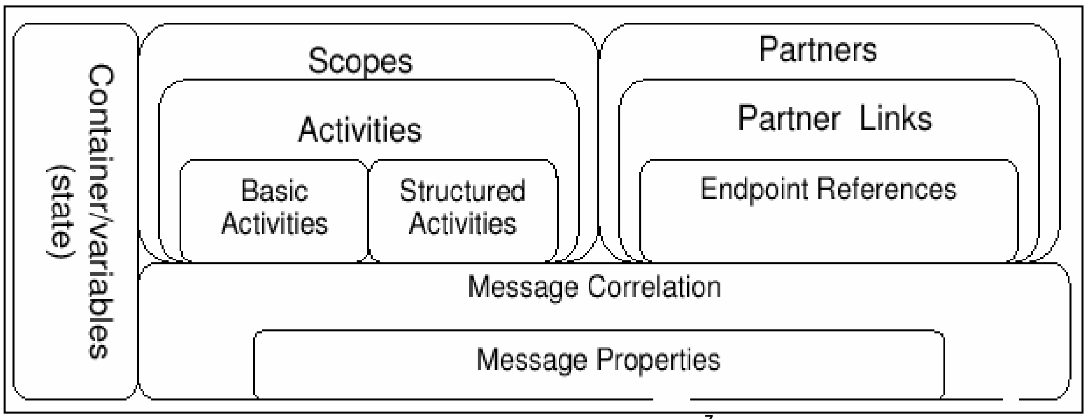


Figure 9 BPEL4WS logical view[7]

Figure 9 shows the fundamental components of the BPEL4WS architecture, which consists of the following:

- A means of capturing enterprise interdependencies with partners and associated partner links;
- Message correlation layer that ties together messages and specific workflow instances;
- State management features to maintain, update, and interrogate parts of process state as a workflow progress;
- Scopes where individual activities (workflow stages) are composed to form actual algorithmic workflows.

Service-Oriented Architecture also enables a new managing way of a heterogeneous IT infrastructure. It introduces the Enterprise Service Bus (ESB) that is a Web Services aware reincarnation of traditional Enterprise Application Integration (EAI) solutions. The architecture of an ESB (Figure 10) is centered on a bus. The bus provides message delivery services. The services might be based on open standards such as SOAP, HTTP, and Java™ Messaging Service (JMS). The ESB enables the use of multiple protocols (such as synchronous and asynchronous) and performs transformation and routing of service requests. The ESB enables services to interact with each other based on the quality of service requirements of the individual transactions.

Components types that can connect to ESB are:

- Custom applications like for instance web applications based on standards like J2EE and Struts, which plug into the ESB to provide a user interface to enterprise services.

- Service orchestration engine, which hosts long running business processes, based on standards like Business Process Execution Language for Web Services (BPEL4WS).

- Adapters that can be with Java Connector Architecture (JCA) or .NET Connector specification enable integration with a wide variety of enterprise applications.

[7] http://www.sys-con.com/webservices

- Presentation and portals enable the creation of personalized portals that aggregate services from multiple sources.
- Data services which provides real time view of data from heterogeneous data sources.
- Web services provide a standard means of connectivity to legacy and proprietary integration technologies.

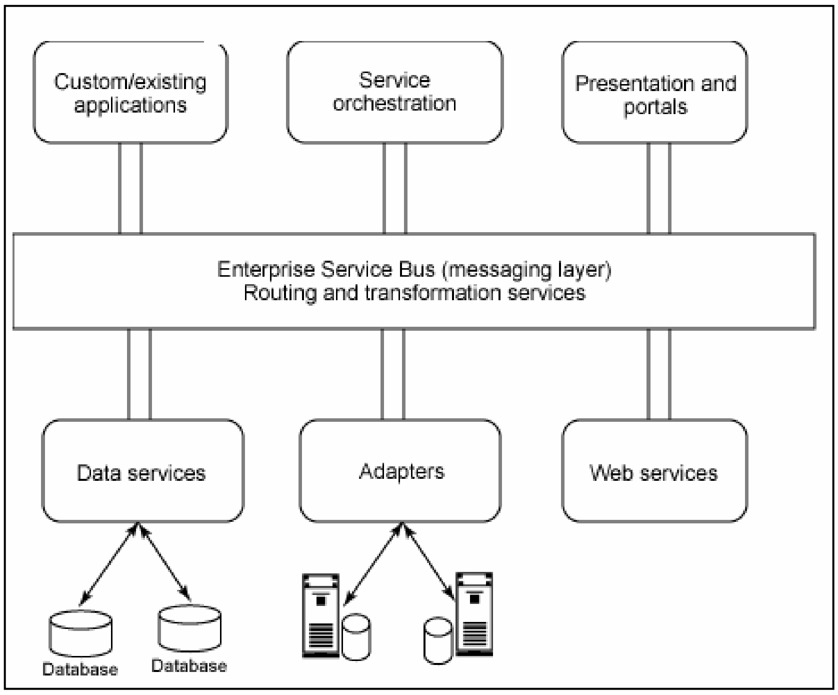

Custom/existing applications
Service orchestration
Presentation and portals
Enterprise Service Bus (messaging layer)
Routing and transformation services
Data services
Adapters
Web services
Database
Database

Figure 10 ESB architecture[8]

In conclusion, SOA provides benefits in three basic categories: reducing integration expense, increasing asset reuse, increasing business agility, and reduction of business risk.

Service-Oriented Architecture Benefits	
• Reducing Integration Expense	• loosely-coupled integration reduces the complexity and hence the cost of integrating and managing distributed computing environments.
• Increasing Asset Reuse	• One of the most important benefits of SOA is that users can create new business processes and composite applications from existing services. In other words, service reuse becomes the major formula rather than application integration.
• Increasing Business Agility	• Through Service-Oriented Process, companies can delegate parts of their overall business process flows to different parts of the organization, each of which have direct and immediate control of the actual operation of the business

[8] IBM source

2.3 ESA – Enterprise Services Architecture

This section describes concepts of Enterprise Services Architecture (ESA) and its contribution to evolution of IT technologies against business values. Also this chapter compares SOA and ESA and presents the Package Composite Applications concept. Package Composite Applications as a part of ESA concept are core of this thesis. Chapter 3 will present a particular technical architecture of software products that enable to realize ESA and PCAs concepts.

Service-Oriented Architecture is one of the fundamental concepts at the root of the ESA. The crux of this concept is that computing resources can be defined as a set of services that encapsulate the functionality of underlying applications and allow access to that functionality through a relatively simple interface. Properly defined, the services should be able to be loosely coupled; that is, they should be able to be combined over and over again to solve different problems in a way that avoids dependencies between services or unintended side effects from invoking a service.

Enterprise Services Architecture (ESA) differs from concepts like Service-Oriented Architecture in two important ways. Firstly, business concerns are always fundamental. ESA is not a goal to be achieved for its own sake or a methodology to be applied everywhere without a concern for costs or results. Second, ESA assumes the starting point is an existing base of enterprise applications, platform component systems, and customized legacy applications, where IT infrastructure entirely is reconstituted into a set of components. The communication between systems components is based on services interoperability. Finally, package composite applications, which reflect a certain process, are offered by ESA concept.

The goal of the Enterprise Services Architecture is to break current enterprise applications into components and services so that user interfaces are no longer linked to the silo of current monolithic applications and Package Composite Applications can be assembled from these services to bring functionality to new groups of users and to extend automation further into the company. Thus, the functionality of current enterprise applications can be more effectively reused. Enterprise Services Architecture aims to extend Web services to Enterprise services. Enterprise services borrow the syntax and standards of Web services to implement business-level requirements, such as scalability, robustness, security, and manageability to fulfill all enterprise requirements. Thus, enterprise services are high-level components that aggregate Web services into reusable elements with business value[9].

Figure 11 presents the layers of Enterprise Services Architecture. The bottom layer presents existing systems such as ERP systems, CRM applications, SCM applications and all enterprise applications. The middle layer is the ESA platform. It is an application and integration platform. The major task of ESA platform is to provide a unified, homogenized view of the enterprise in order to enable building new process-based applications such us Packaged Composite Applications. The highest layer presents a business process as an software application. Package Composite Applications are a new breed of business applications that consume services and data, and are orchestrated to reflect new business processes.

[9] [ESA-Woods]

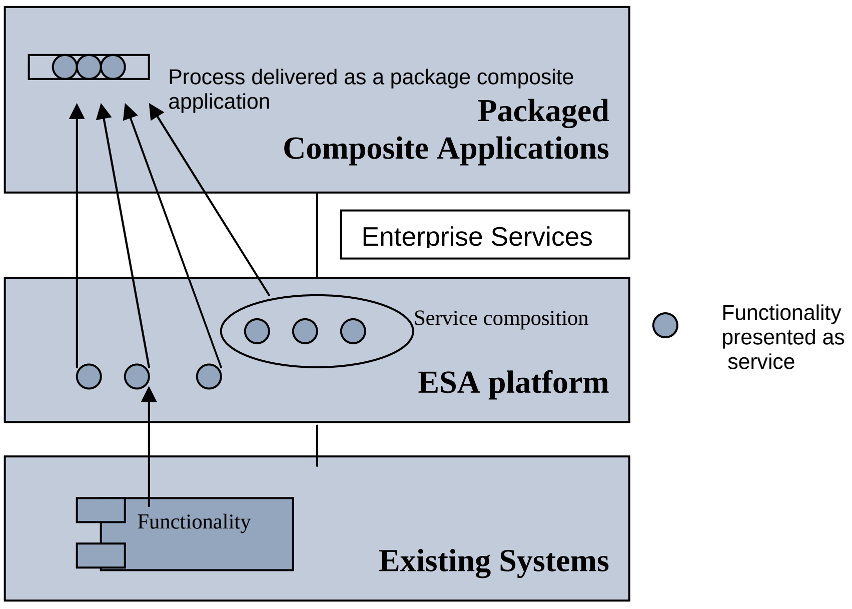

Process delivered as a package composite application
Packaged Composite Applications
Enterprise Services
Service composition
Functionality presented as service
ESA platform
Functionality
Existing Systems

Figure 11 Overview of Enterprise Services Architecture

ESA platform as integration and application platform can be seen from the idea of the application stack. The application stack is the name for the different layers of functionality that exist in most software programs[10]. The layers are:

The user interface layer

This is the part of the software that controls interaction with the users. Here, are included also Content- and Knowledge Management Systems.

The process layer

The part of the software that automates business processes, moving end users from one step to another as they complete tasks and coordinating the execution of underlying services. This layer presents Business Process Management

The services layer

The part of the software that contains the logic to perform the transformations, calculations, and other required processing to do the work of the application.

The object layer

The data of the application and the associated service functions that perform basic manipulations on the data and move the data back and forth between the object and persistence. Component-based platforms included are here.

The persistence layer

Usually it is a database, although it is possible to store or persist data in many other ways.

[10] [ESA-Woods]

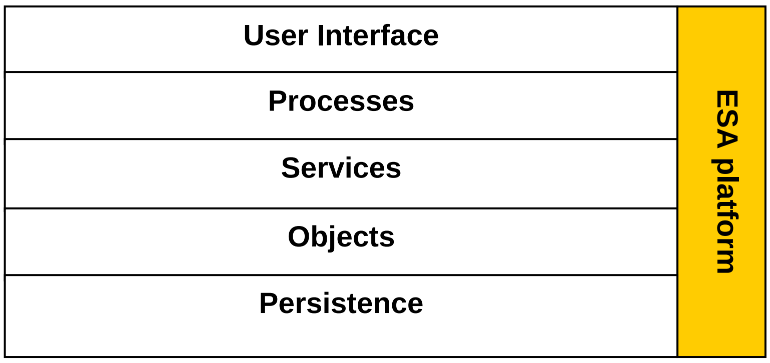


Figure 12 ESA platform stack

Figure 13 presents the complete Enterprise Services Architecture overview.

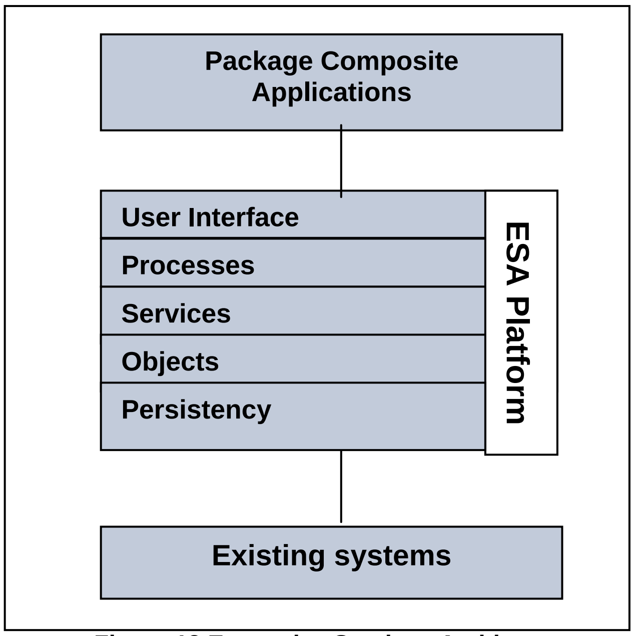


Figure 13 Enterprise Services Architecture

2.3.1 PCA - Packaged Composite Applications

This section presents a concept of Packaged Composite Applications (PCAs) which are brought with ESA concept. It will be described what the reason for building new composite applications is. I will define demands that must be met by a development environment for composite applications.

Nowadays, many companies are faced with challenge of quickly reacting to their customer's needs – not only locally but around the whole world. This situation forces companies to improve communication and collaboration in order to guarantee improved decision-making and increased productivity. Therefore current enterprise applications should:

- Serve business processes that cross multiple functions, meaning new applications are able to drive end-to-end business processes not only across different applications, but also across heterogeneous IT systems and organizations. (cross-functionality)

- Target multiple users across the enterprise, what plays a great role in ensuring vital business processes and in facilitating decision-making (collaboration)
- Integrate all of company's generic applications and systems in compliance with company's strategy. Should the strategy change, the company can easily reconfigure its processes (composition)

Existing enterprise applications can no longer keep up with above-mentioned demands as they are designed for specific functions and users. They are designed for:

- Fit and fulfill particular generic business functions or processes
- Fulfill a generic need, such as managing, storing and creating documents, sending messages or searching for information
- integration process (EAI approach) of them is to expensive and still provides native API approach

Composite applications that are built on top of existing applications and heterogeneous systems can fulfill the needs for cross-functionality, collaboration and composition. Packaged Composite Applications (an example of composite applications) are applications that are built on top of existing applications and heterogeneous systems (cross). Thus, can consume services and data from existing sources (functionality) and can orchestrate them to fulfill a particular business process (composition). The service composition takes place in ESA platform. ESA platform acts as a point-to-point connector between participating applications. Package Composite Applications rely upon underlying existing systems and cannot function as applications on their own[11] (Figure 14).

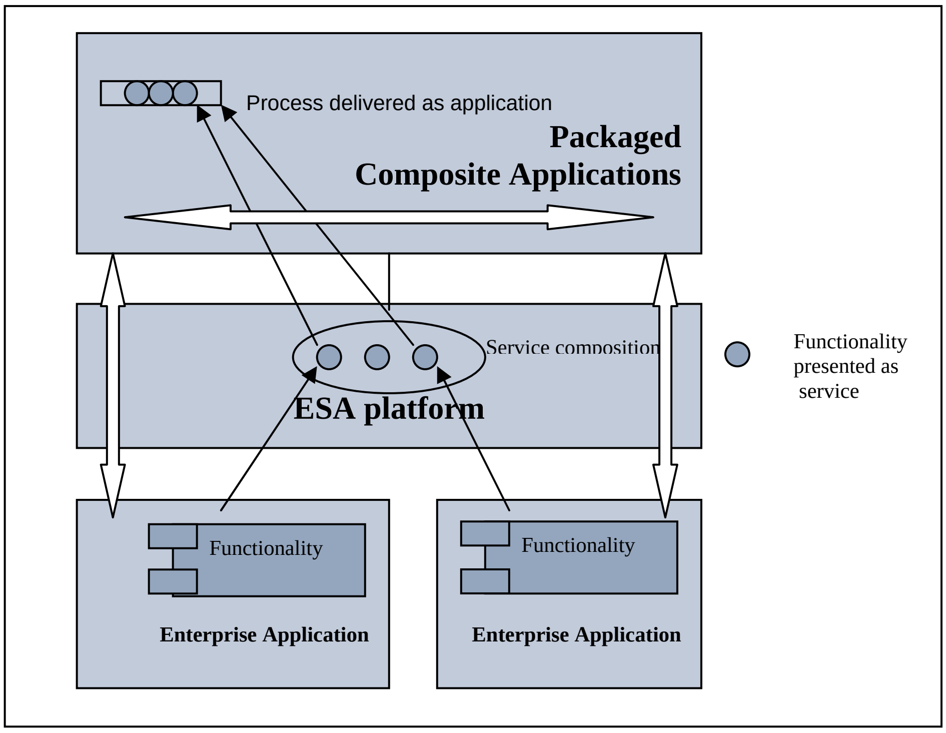


Figure 14 Concept of Package Composite Applications

[11] [PCA-Woods]

ESA concept includes the PCA concept itself. The ESA platform must enable a point-to-point connection through adapters between existing applications. Also ESA platform must provide integration of collaborating systems and orchestrate their services in order to build a business process that can be delivered as a package composite application. Such requirements create a need to provide a unified development environment within ESA platform that addresses them and enables building package composite applications.

Such a unified development environment must reflect changes in building business applications that introduces service-based architecture like ESA. In ESA concept communication between all collaborating systems is based on services. Package Composite Applications cannot function as application on their own. As a result, Package Composite Applications must be build by grabbing various services and exposing them to end users.

Also the first major requirement for a new unified development environment for PCA is to import services from existing systems and expose them as user interface to users that participate in a business process presented by a PCA. Moreover, it must enable a creation of additional new services for business objects.

Furthermore, in a context of application development, package composite applications already are provided with functionality and data encapsulated as services. Therefore, while building package composite applications the designers are freed from data modeling and have to coordinate their functionalities to solve business problems. Thus, the user interface designers must focus on the end user needs. The user interface must support end users with a task and process-based user interface to center him in a process. This can be achieved by introducing workflow tools that from user interface level enable end users to define a specific process. Moreover, repeated across many UIs user interface interactions can be defined as UI patterns.   Below are once again summarized needs and challenges to be considered while building a unified development environment for packaged composite applications are summarized.

Integration of heterogeneous system landscapes	• separate business objects from persistency • remote data access to backend systems
Leverage user interface development	• encapsulate user interface from application data • introduce user interface common patterns
Leverage process development	• define workflow tools • introduce workflows patterns

Having discussed the ESA concept let's consider the following Order-to-Cash scenario and analyze the impact of this approach.

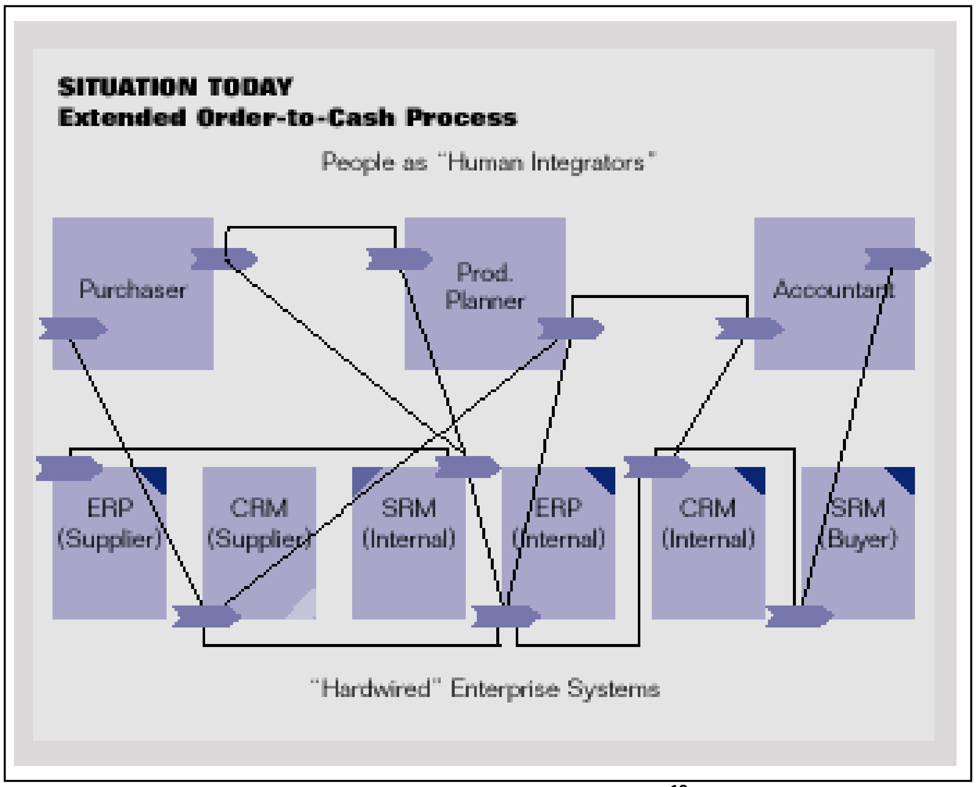


Figure 15 Order-to-Cash scenario[12]

In the past the order-to-cash process involved interaction of different software components such as ERP, CRM where the orders are stored (Figure 15). Usually this process required human intervention to process and forward information manually. The Enterprise Services Architecture aims to provide a concept to automate and coordinate this Order-to-Cash scenario. Using service interfaces users and existing systems are integrated to one composite application that defines and manages the respective integration. As a result the process is more flexibly adaptable to changes based on future business requirements (Figure 16).

[12] SAP source

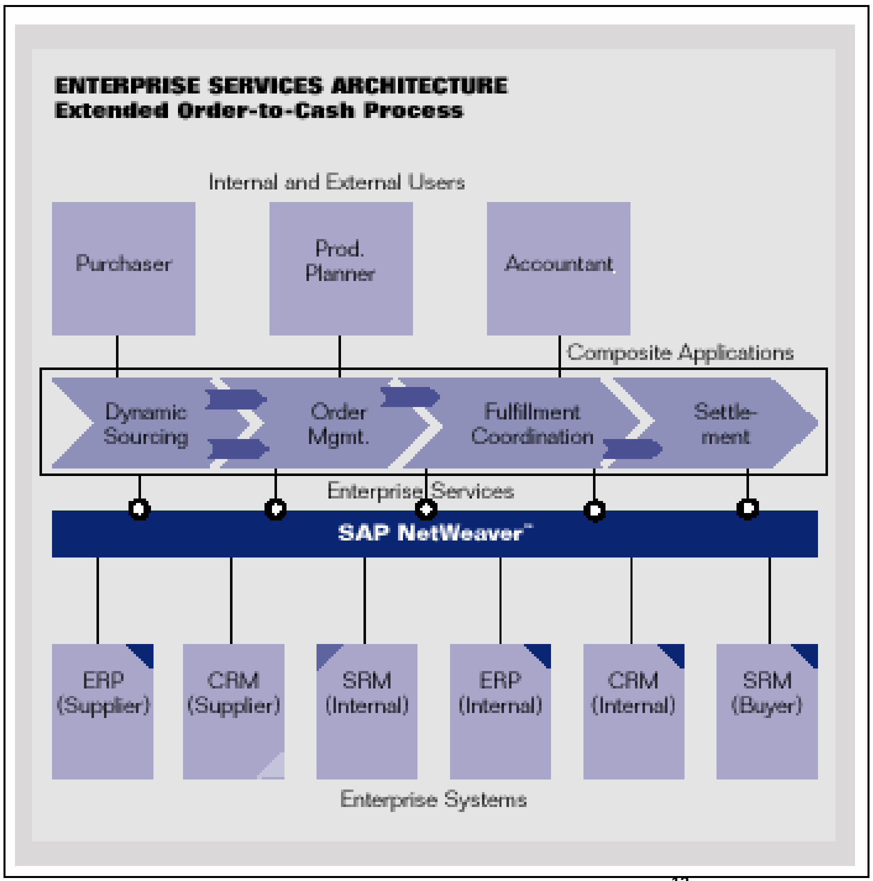


Figure 16 Order-to-cash scenario with ESA[13]

2.3.2 ESA and SOA

Enterprise Services Architecture concept is very similar to Service-Oriented Architecture paradigm since SOA is the crux of ESA. Both concepts (re)-use functionalities (components) of existing systems that are encapsulated into services. The communication between system components is based on services interoperability. Also, SOA enables services integration within its landscape that is based on the reconstituted traditional EAI approach called Enterprise Services Bus (ESB). The bus is a centralized place for routing and transformation of services. In both concepts services are orchestrated in order to reflect business processes within enterprises. ESA concept introduces an additional layer Package Composite Applications that represent a process delivered as a product (solution). Package Composite Applications are itself a concept as well. PCA is a new type of business applications that consume services and data from service-based architectures like SOA or ESA and orchestrate them to reflect a process. The table below summarizes differences and similarities between ESA and SOA.

[13] SAP source

Similarities	***Differences***
Services play major role in reuse functionalities of existing systems and interoperability between heterogeneous systems.	In ESA, particular process delivered as a software solution (PCAs)
Business processes are composed of services, process-centric approach	For ESA the web services paradigm is crucial, but technologies like traditional EAI and APIs are vital features as well[14].

2.4 Summary

Chapter 2 presented some important and related concepts aimed to organize business functions and IT infrastructure in order to increase efficiency and to gain competitive business advantage of an enterprise taken up by IT organizations over the past two decades. In particular, in this chapter it was described Enterprise Services Architecture and SOA. Both are at present most promising IT solutions for lowering TCO. Chapter this concentrated was as well as on Packaged Composite Applications concept and definition of challenges for a development environment in the context of composite applications.

As already outlined, the goal of Enterprise Services Architecture is to break the monolith applications into services so that user interfaces are no longer linked to the monolith applications and packaged composite applications can be assembled from these services to bring functionality to new groups of users and to extend automation further into the enterprise.

The Enterprise Services Architecture concept has its roots in Service-Oriented Architectures. However, for ESA the Web services paradigm is no crucial, technologies like traditional EAI and APIs are vital features as well. Reason for that are immatureness standards for such features like transactions, security, and guaranteed delivery.

The ESA concepts are technically realized as SAP xApps (packaged composite applications) and SAP NetWeaver platform (ESA platform) technologies. These technologies enable a company to make progress toward an architecture that embodies Enterprise Services Architecture principles. These technologies will be a subject for the next chapter 3. Briefly, it will be presented the architecture of SAP NetWeaver integration platform. The major focus will be on SAP's vision how to build and develop composite applications such as SAP xApps with a use of Composite Application Framework.

[14] [ESA-Woods]

3 SAP NetWeaver Technologies

According to Merriam-Webster Dictionary[15], one of the many descriptions defines a technology as "*the practical application of knowledge especially in a particular area*" or as "*a manner of accomplishing a task especially using technical processes, methods, or knowledge*", and methodology as "*the analysis of the principles of inquiry in a particular fields*". When executing an investigation project, both technologies and methodologies are required to curry out it with the aim to discover and interpret potential benefits and drawbacks of practical applications.

This chapter describes the important technologies and methodologies required for this master thesis. The section describes SAP NetWeaver platform that is SAP's blueprint of Enterprise Services Architecture concept. In particular, it will be described SAP NetWeaver solutions that enable to realize the Package Composite Applications concept.

The ESA concept is technically realized as a SAP NetWeaver platform and SAP xApps Technology (Figure 17). SAP NetWeaver presents the technical realization of ESA platform concept. SAP xApps (Collaborative Cross Applications) are SAP's product that embodies the Package Composite Applications concept.

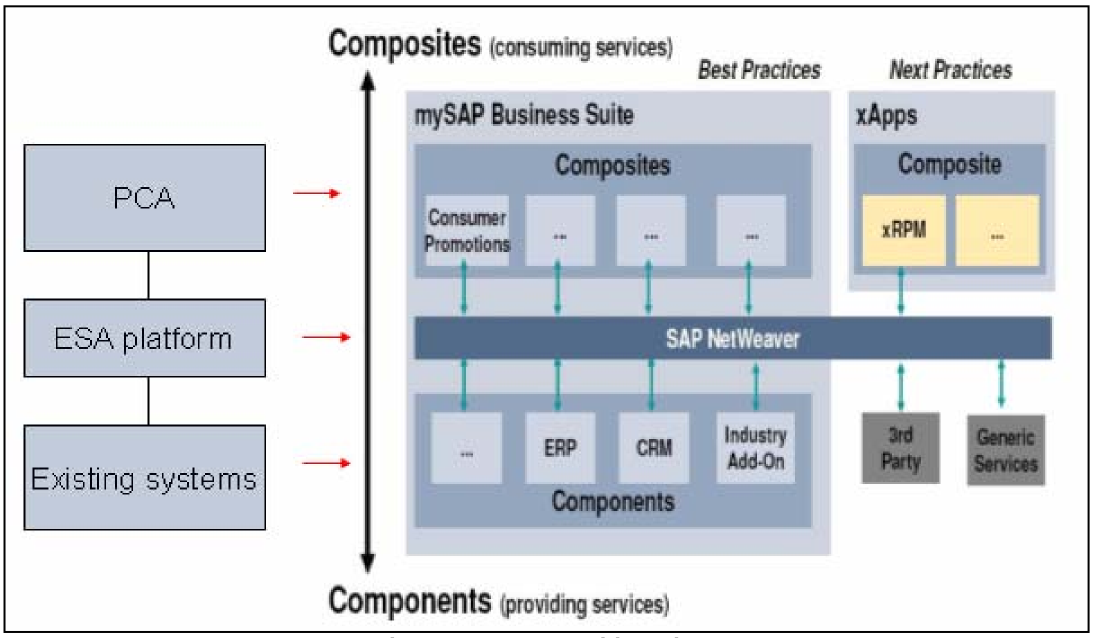


Figure 17 SAP ESA blueprint

SAP NetWeaver is the technical foundation of Enterprise Services Architecture and its ESA platform. It combines SAP's experience in enterprise applications with the flexibility of Web services and other open standards. Simply putting, SAP NetWeaver is a set of capabilities that are provided by many different SAP products constructed to work with each other to make applications work together, build new applications on top of existing enterprise applications, and lower the

[15] Merriam-Webster is America's foremost publisher of language-related reference works. http://www.m-w.com

total cost of owning applications[16]. Figure 18 presents overview of SAP NetWeaver platform with notation of products that enable building Package Composite Applications. SAP distinguish two kinds of PCAs: SAP xApps are package composite applications that are delivered by SAP AG and its partners and custom composite applications can be built by customers with application of Composite Application Framework, which is the unified environment for building PCAs in SAP NetWeaver platform.

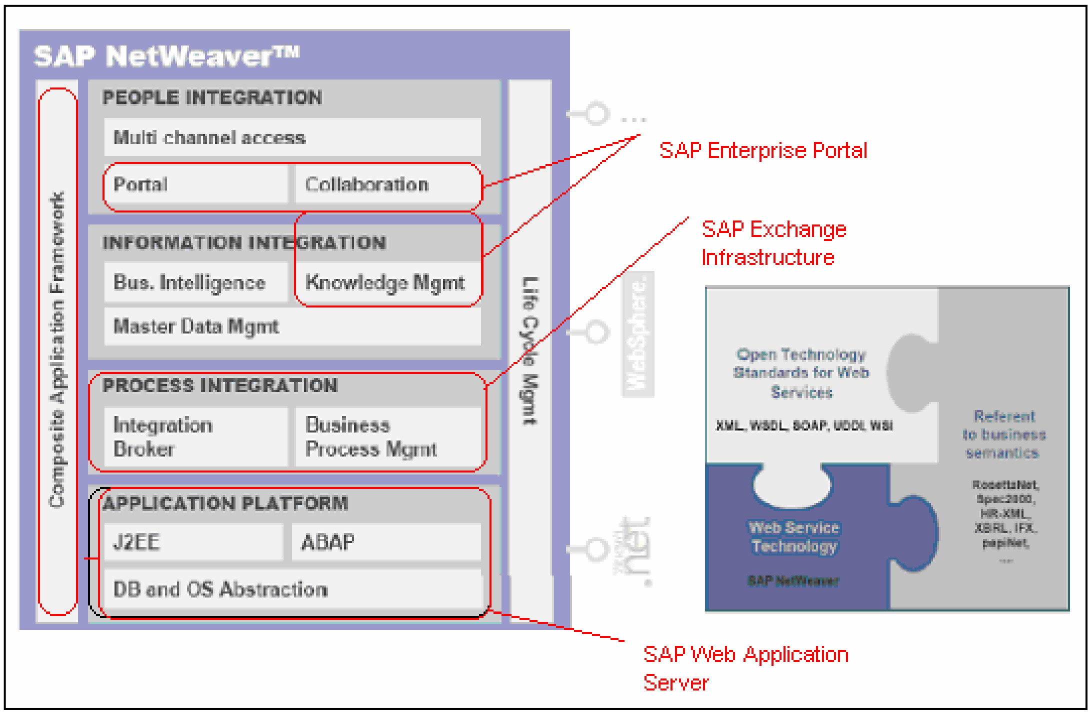


Figure 18 Overview of SAP NetWeaver[17]

Within SAP NetWeaver technical software components that are used to build and run custom composite applications as well as SAP xApps application on top of other business applications are:

- SAP Web Application Server (SAP Web AS) on which runs all SAP NetWeaver solutions. This is the application platform.
- Composite Application Framework (CAF) that provide development infrastructure (design time) and tools to build PCAs. Also CAF is a runtime for PCAs.
- SAP Exchange Infrastructure (SAP XI) forms the basis for the integration of business processes. SAP XI provides a technical infrastructure for XML-based message exchange to enable the integration of SAP systems with other systems.
- SAP Enterprise Portal (SAP EP) unifies enterprise applications, information, and services from SAP and non-SAP sources into one system to support business processes. Hence, SAP EP is the user-interface layer for all SAP´s and non-SAP`s applications.

[16] www.sap.com/netweaver
[17] www.sap.com/netweaver

Figure 19 presents the simples landscape (without SAP XI) of SAP xApps that can function on top of existing systems and run on SAP NetWeaver platform.

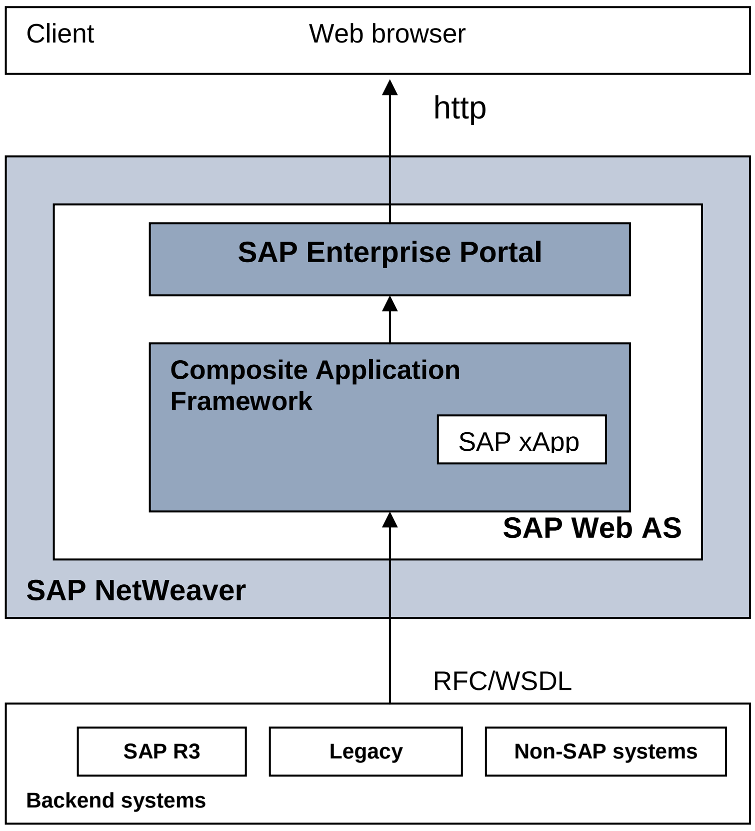


Figure 19 Architecture of SAP xApps[18]

In the context of SAP xApps, Composite Application Framework (CAF) acts the major role as a unified environment for designing and running such composite applications. CAF itself is a development environment that is a part of SAP Web AS. Services from backend systems can be imported to CAF via web services, RFC, and BAPIS interfaces. Within CAF are integrated collaborating systems and new functionalities as services can be added as well. SAP xApps application can be integrated within SAP Enterprise Portal content management system to bring all users participating in a process presented by a particular SAP xApps. Following chapters will describe architecture of Composite Application Framework, SAP Web AS, and SAP Exchange infrastructure.

SAP and SAP xApps partners provide already packaged composite applications that drive specialized processes across a variety of industries like:

- *SAP xApps Cost and Quotation Management (SAP xCQM)* – helps to turn business expertise and supplier relationships into revenue
- *SAP xApps Resource and Portfolio Management (SAP xRPM)* – helps to improve ability to manage enterprise wide project portfolios
- *SAP xApps Product Definition (SAP xPD)* – helps to translate ideas into innovation
- *SAP xApps Emissions Management (SAP xEM)* – helps to enable energy-consuming and carbon dioxide-producing businesses to comply with environmental regulations

[18] [xApps-Herger]

- *SAP xApps Global Trade Services (SAP xGTS)* – helps to drive regulatory compliance for international trade

3.1 SAP Web Application Server

SAP Web Application Server (SAP Web AS) is an open, scalable, and high-availability infrastructure for developing dynamic and company-wide Internet applications. It provides a full support for platform-independent Web services, business Web applications, and open standards-based development built on J2EE and ABAP technologies. As a result, customers can leverage existing technology assets while building and deploying new dynamic e-business applications[19].

As figure 20 depicts SAP Web AS like a typical web application server is build out of a connectivity layer, presentation layer, business layer, integration layer, and persistence layer. The connectivity layer represented by the Internet Communication Manager ensures TCP/IP connection to the outside world. A variety of standardized protocols such as HTTP, HTTPS, SOAP and SMTP are supplied as standard. The presentation layer creates the graphical user interface (GUI). SAP Web AS contains two personalities: ABAP and J2EE. These complementary environments produce the options available for developing user interfaces either with Business Server Pages (BSPs) for ABAP applications or JSPs, Java Servlets, and JSP tag libraries for Java-based web applications. The business layer provides the necessary business logic of a web application. Depending on the personality, the business logic is implemented in either the ABAP world or the J2EE world and Enterprise Java Beans. The integration layer ensures access to business functionalities from external sources. This is done using various interfaces, connectors, communication protocols, and support for general data exchange formats like XML. The SAP Web AS provides necessary mechanisms for connecting the SAP world like RFC, BAPI, IDoc and the non-SAP world via a variety of connectors like Java- and .NET-connectors, WSDL. Finally, the persistence layer uses a general valid abstracted database interface to ensure a maximum level of database-independence and efficiency of access to datasets. Open SQL interface is used by ABAP world for integration various databases. The Java side provides database-specific links to different databases using JDBC.

[19] www.sap.com/netweaver

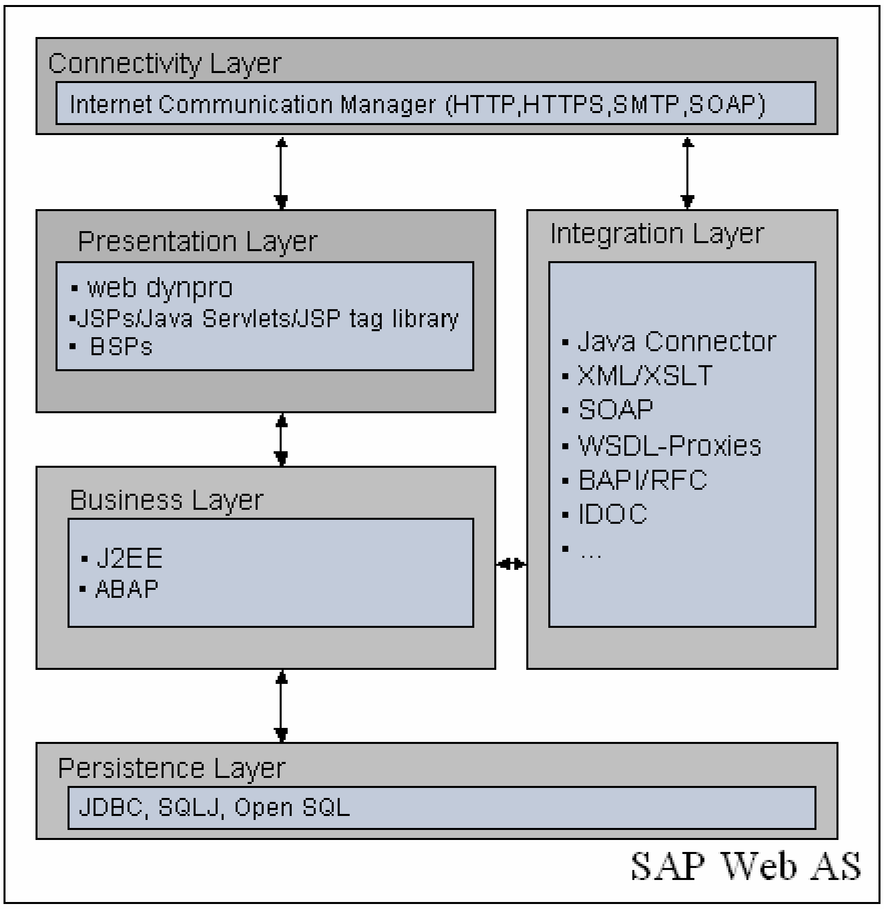


Figure 20 The architecture of SAP Web AS[20]

The SAP Web Application Server implements the basic Web services standards eXtensible Markup Language (XML), SOAP, Web Service Definition Language (WSDL) and Universal Description, Discovery, and Integration (UDDI) by providing web services framework (Figure 21). Software components can be developed in both the ABAP and J2EE personality. Such components can be encapsulated to services via such open standard like web services. On the SAP Web Application Server session beans are the preferred kind of J2EE component for an implementation of a Web service. Using the Web service framework, any session bean can be easily transformed into a Web service, for simple ones it is not necessary to write code. In addition, all functionality that is available through BAPIs can be provided as Web services. In the ABAP personality, each remote-compatible function can be released as web service.

Access to external web services is also supported for both personalities. A uniform SOAP framework and a proxy generator are available for this purpose. The proxy classes use the SOAP framework for communication via web services. Here the SOAP Framework provides an object-oriented framework for the Integration Engine. The Integration Engine carries out the low-level communication via the Internet Communication Manager.

[20] [WPWAS-Rau]

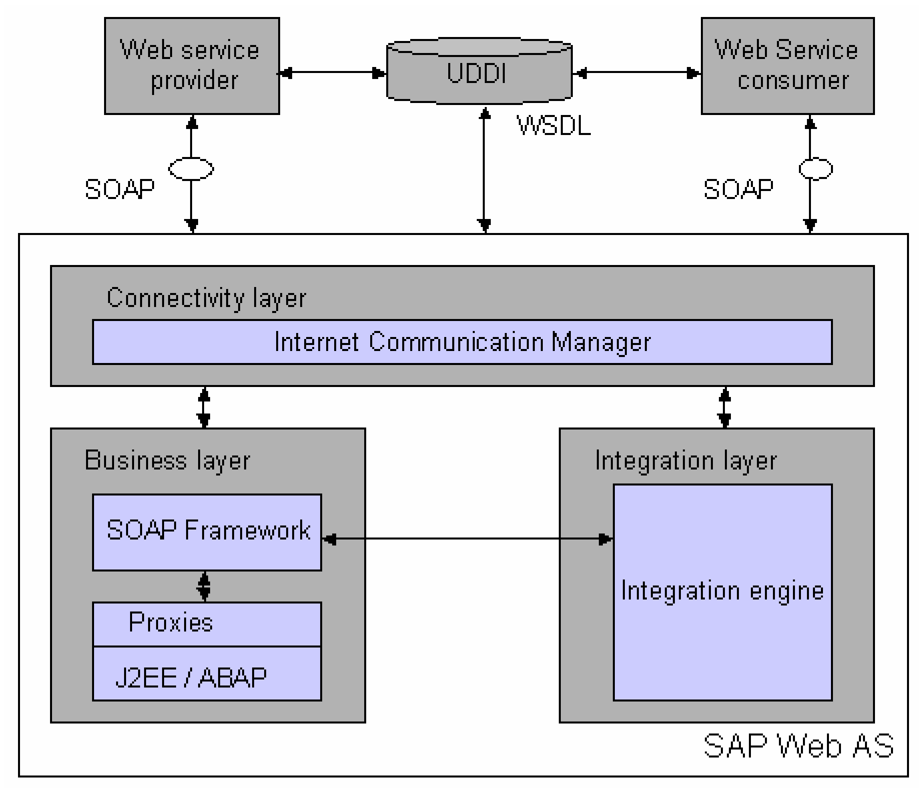


Figure 21 Web Services Framework[21]

The SAP Web AS supports the WS-Security 1.0 standard. Regarding security SAP Web AS provides additionally support for security standards like Secure Sockets Layer (SSL) on the basis of HTTP (HTTPS). Certificates (X.509, SAML), tickets, and cookies are used for authentication purposes. Therefore, SAP Web AS supports the single-sign-on (SSO) authorization and can be integrated into SSO environments as well as can itself create SSO environments. Open standards for web services coordination and transaction (WS-Coordination, WS-Transaction) are not supported.

Interoperable Web services are key to integration in SAP application and a key component of the ESA vision. As mentioned above, all of existing core interfaces, RFC, BAPIs, and IDocs, are available as Web services. In addition to peer-to-peer connectivity between SAP and non-SAP applications using Web services. SAP Exchange Infrastructure (SAP XI) provides additional mechanisms on top of pure technical connectivity, for example, logical addressing, message mapping, business process modeling, and execution. SAP XI also supports other protocols by means of adapters, for example, Java Message Service (JMS), Java DataBase Connectivity (JDBC), and RosettaNet Implementation Framework (RNIF). Therefore, in the following chapter it will be presented architecture of SAP XI.

[21] [WPWAS-Rau]

3.2 SAP Exchange Infrastructure

SAP Exchange Infrastructure (SAP XI) forms the basis for the integration of business processes. SAP XI provides a technical infrastructure for XML-based message exchange to enable the integration of SAP systems with each other on the one hand, and SAP and non-SAP systems on the other hand. Furthermore, SAP XI provides a set of integrated tools for creating and managing all integration-relevant information into business process with use of BPEL4WS. SAP Exchange Infrastructure (XI) is a product that is positioned in the Enterprise Application Integration (EAI) area. Figure 22 presents SAP XI architecture.

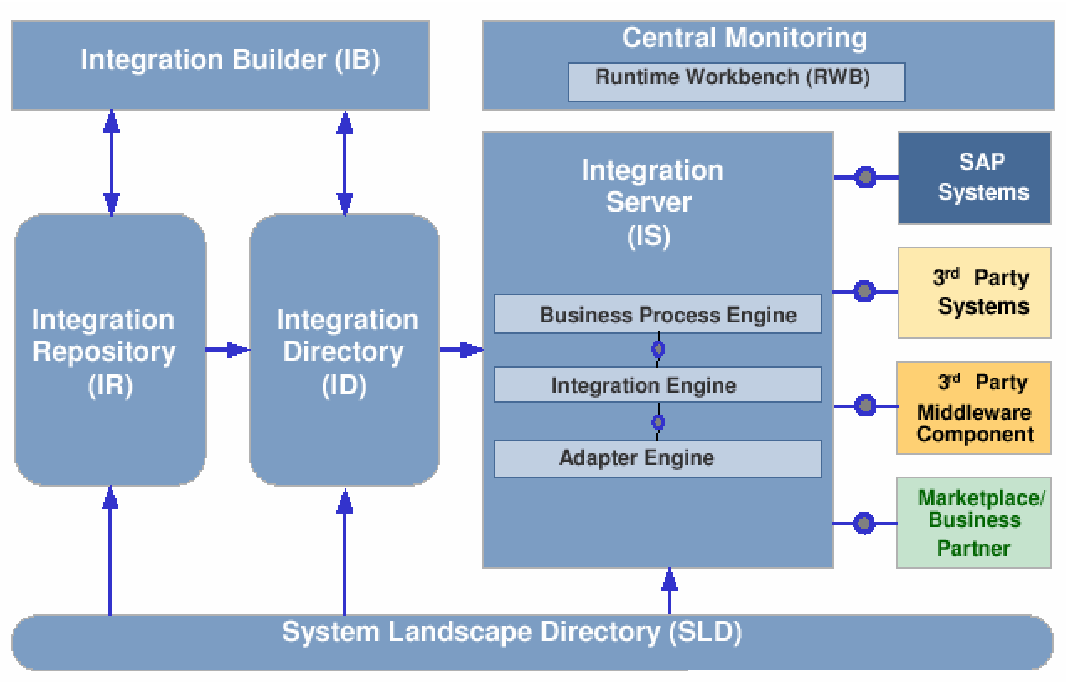


Figure 22 Architecture of SAP XI.[22]

SAP XI has the following components:

- Integration builder (IB)
  - Integration repository (IR)
  - Integration directory (ID)
- System landscape directory (SLD)
- Integration server (IS)
- Central monitoring

The system landscape directory is directory of the technical information about the programs (software or applications) and computers (technical or business systems) being connected by SAP XI as a central information provider.

Central monitoring monitors and assess whether the messages are successfully flowing between systems. This centralized access point gives a view of the whole integration scenario and presents the constrains, the technical end-to-end monitoring, and everything you need so that you can see the path that messages take.

[22] SAP source

Integration Builder (IB) is the central development environment for the following:

- o Development of all design objects in the Integration Repository at design time
- o Definition of all configuration objects in the Integration Directory at configuration time

In the Integration Repository (IR) we can define following objects through the graphics design tools

- o Define *data, message type* and *interfaces* with an interface editor
- o Create *rules* using the condition editor
- o Define mapping rules between two different message formats with the mapping editor
- o Define the cross component Business Process management (ccBPM) with the process editor
- o Knit all objects we need for one business scenario in the scenario editor

The Integration Directory is where the message types and processes described in the integration repository are connected to the real word. All metadata that describe all connections are stored in the integration directory, for example, the business systems and applications involved in the business scenario and the routing rules for the systems. Thus, in Integration Repository business processes are defined. Figure 23 presents the architecture of Integration Directory. In the *collaboration profile* you document the technical options available to the communications parties for exchanging messages. You specify the potential senders and receivers of messages and the technical details for communication path. In *routing rules* you define the flow of messages in a system. In *collaboration agreements* you define the technical details for message processing like adapter configuration or security settings for senders and receivers.

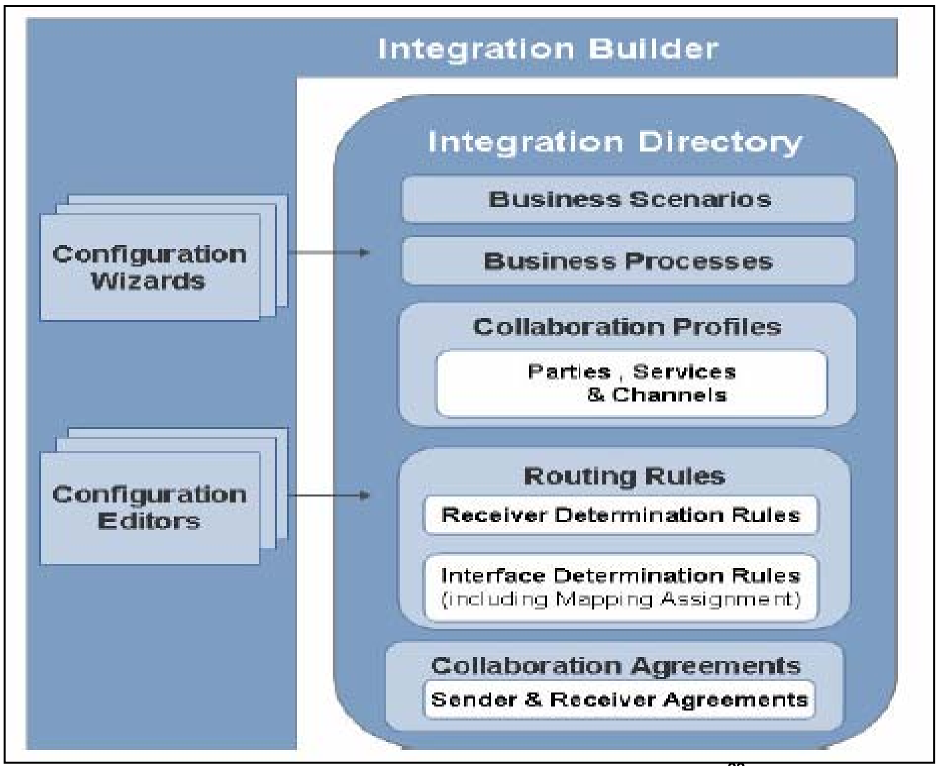


Figure 23 Architecture of Integration Directory.[23]

[23] SAP source

The Integration Server works as a brain where it comes all together at runtime. The message is processed here based upon the information what is configured in the Integration Directory. Inside Integration Server, the Integration Engine is the runtime of Exchange Infrastructure that receives processes and transfers XML messages. Adapter Engine connects the Integration Engine to SAP and external systems. Various adapters convert XML and HTTP-based messages to the specific protocol and format required by these systems.

Figure 24 presents how the business process with Business Process Engine is integrated into Exchange Infrastructure.

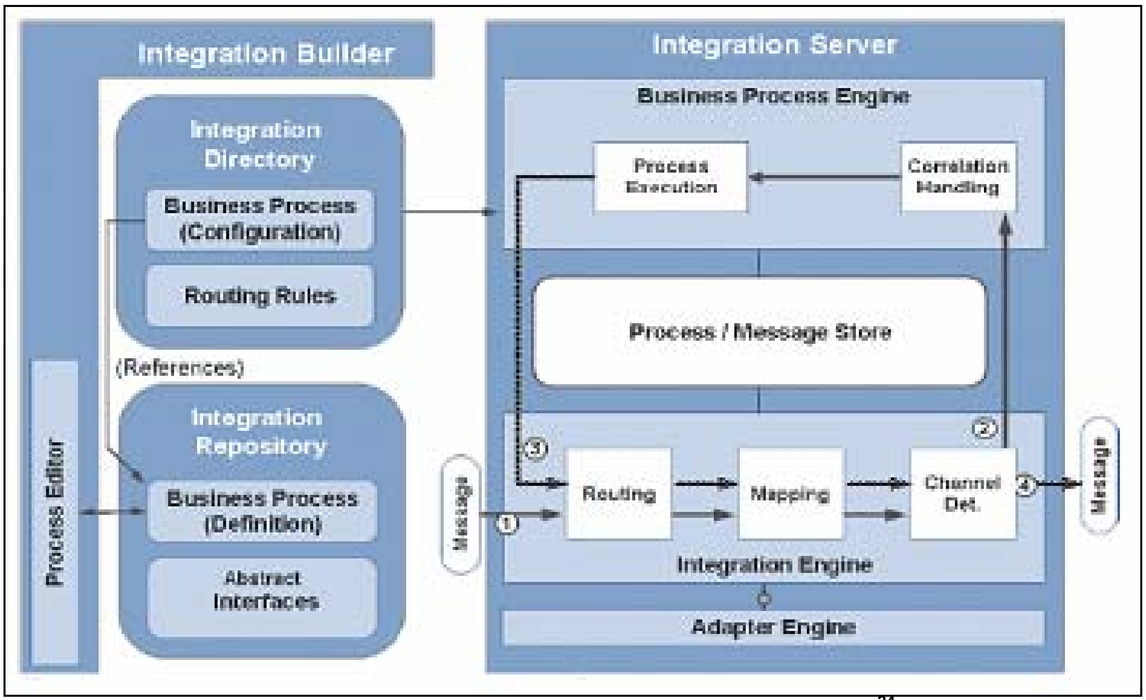


Figure 24 Business process within Business Process Engine[24]

Figure 25 presents a scenario of where client sends a request (Web service) to a banks system. The request performs a credit limit check and contains the amount as one of the input parameters. Based on a routing rule in SAP XI, the Web service request is directed to an SAP R/3 system (for small amounts) or to a .NET application (for larger amounts).

[24] SAP source

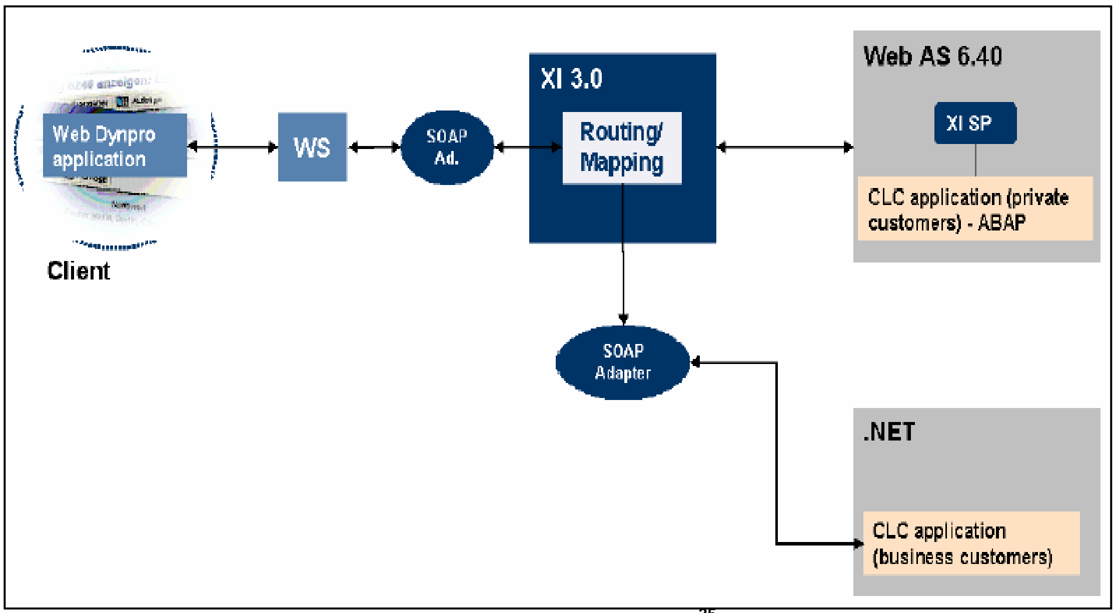


Figure 25 Scenario with SAP XI[25]

3.3 Composite Application Framework (CAF)

The SAP Composite Application Framework (SAP CAF) as one of the building blocks of SAP NetWeaver platform is an environment for a development and running package composite applications (PCAs) such as SAP Collaborative Cross Applications (xApps) based on Enterprise Services Architecture.

The SAP CAF is a SAP's vision how to build future business transactional enterprise applications like SAP xApps in the SAP landscape. The Composite Application Framework uses SAP NetWeaver to encapsulate the functionality of the current generation of enterprise applications as components and services. These components and services are described in metadata and implement standard frameworks for application layers, such as the user interface and process control. The combination of functionality abstraction, standard services and metadata descriptions enable the application designer to assemble an application primarily through modeling rather than through programming. The modeling technique is leveraged further through the identification and reuse of patterns that are common to many application scenarios. As presents the Figure 26, the Composite Application Framework supports developers and business analysts with patterns, in particular, with patterns for user interfaces as well as process workflow patterns.

[25] SAP source

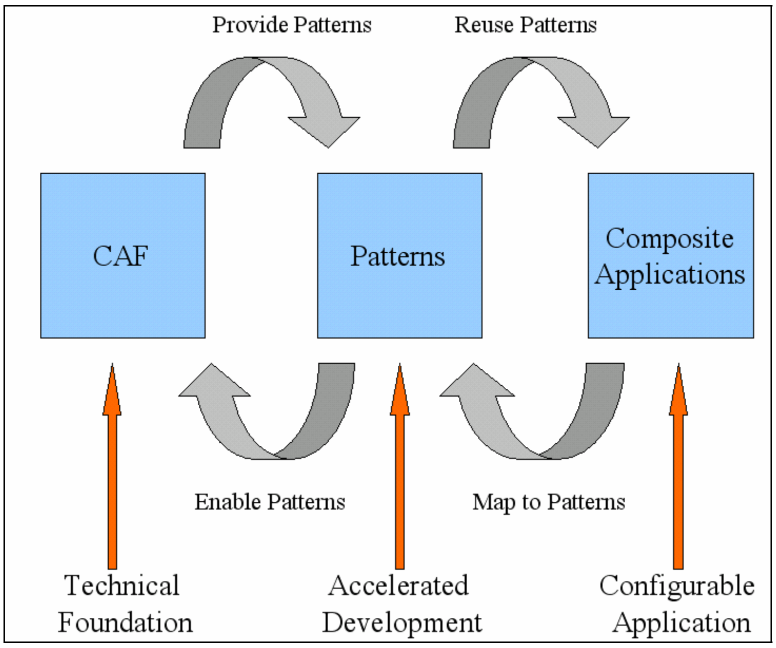


Figure 26 CAF and Patterns

Composite Application Framework includes:

- *Model-driven architecture* – supports model-driven application composition, so applications can be built with as little programming as possible. That means reduced development time for new applications and business-pattern-oriented integration.
- *Services access layer* – the services layer decouples repositories of underlying systems from business objects and processes. Thus, SAP xApps can access any existing system landscape. The service access layer is a central interface that controls communication with participating systems via Web Services and SAP Exchange Infrastructure. SAP xApps do not need to know whether service is provided by another SAP NetWeaver component or an external service provider. Tools such as service modeler allow creating back-end-independent object models for SAP xApps.
- *Collaborative business context* – a service framework allows relating any service of SAP NetWeaver components with any business object. Collaboration objects such as task, document and meeting are accessible within the service access layer. Hence, all composite applications based on SAP CAF can have built-in collaboration functionality.
- *User interface patterns and Guided Procedures* – these tools accelerate application design and collaborative process execution through reusability and automatic configuration. Guided Procedures are like best practices patterns, with an easy-to-use, design user interface and run-time process visualization. The business objects and services of SAP CAF are foundation for the design of Guided Procedures. Predefined workflow patterns support the process definition of Guided Procedures. Users can make ad-hoc modifications to the business process.

3.3.1 CAF Architecture

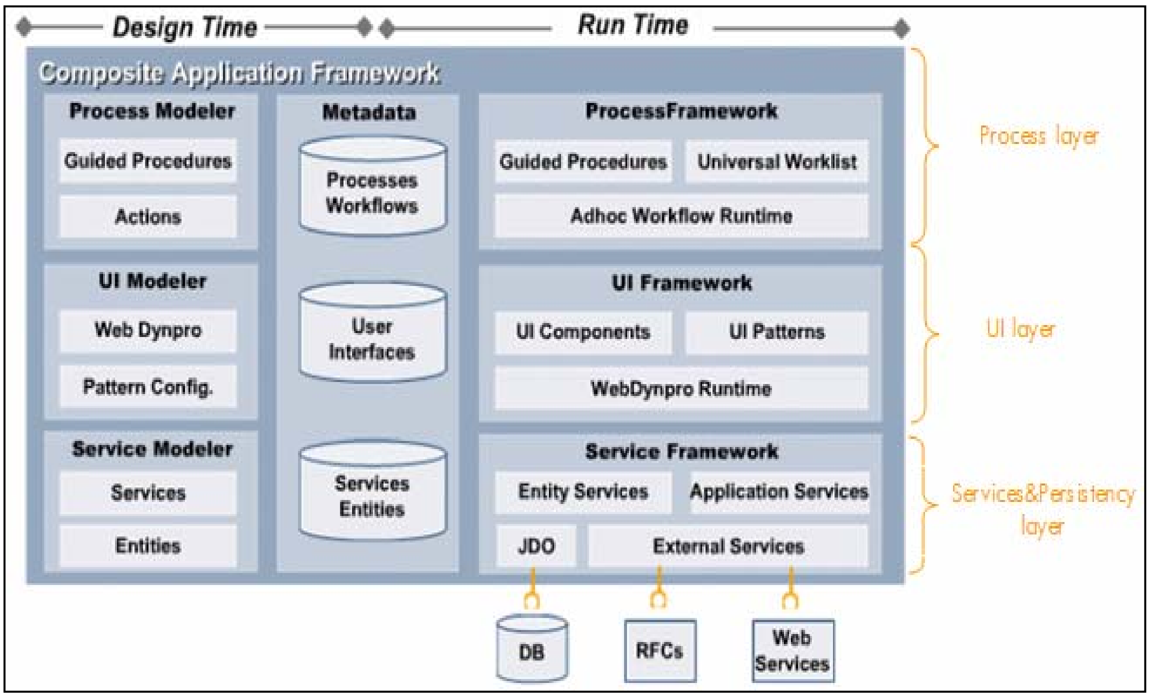


Figure 27 CAF Architecture[26]

The Composite Application Framework is a development as well as a runtime environment for packaged composite applications. Thus, the CAF architecture is split into two parts Design Time and Runtime.

The Design Time

The Design Time part is used to model services and/or map external services (i.e. web services, RFC, BAPI). Also to assemble user interfaces from UI patterns as well as to assemble a process workflow. The Design Time is provided within SAP NetWeaver Developer Studio.

The Runtime

The Runtime part, so called CAF-Runtime is running on the SAP Web AS J2EE Engine. It includes:

- The Business Intelligence Meta Model Repository (BI MMR) as storage for metadata. Metadata are XML-files which describe attributes and relations of services modeled in the CAF design time
- CAF libraries that are responsible for authorization, logging and tracing, and many others system processes

The Design Time and the Runtime, both access the same metadata layer (metadata repository) where metadata of services, user interfaces (UI), and processes are stored.

[26] Training materials from Workshop “SAP Composite Application Framework / SAP NetWeaver” at SAP AG in Walldorf

The architecture of the Composite Application Framework (CAF) consists of three layers: Service Layer, UI layer, and Process layer.

Service Layer

An abstraction layer which exposes business objects that exist within an enterprise to the outside world as services. CAF provides the following kinds of services;

- application services for implementation of application's business logic,
- entity services describe existing business objects in a company
- external services that provide access to external services like remote function call (RFC) or web services.

UI Layer

An abstraction layer that allows users to access services through user interfaces. User interfaces are built based on the Model-View-Control concept from generic predefined UI patterns and/or foundation Web Dynpro applications.

Process Layer

The Process layer is an abstraction layer that defines step-by-step collaboration workflow patterns (guided procedures) for composite applications in order to create specific process within a company.

The following figure presents steps in designing package composite applications with Composite Application Framework.

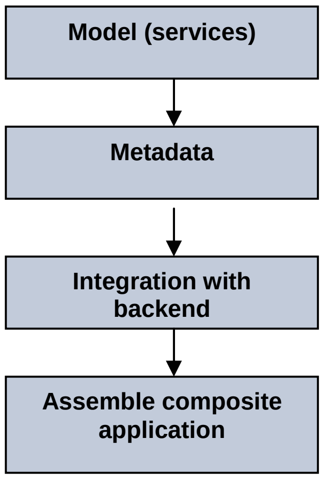


Figure 28 CAF: Workflow of building composite applications

Step 1
From a modeling tool level a model based on services is created. This model includes services that describe business objects that will present data needed by a composite application. New business objects are defined as entity services. Composite applications' business logic is represented through application services that can be published and reused as web services. Application services are used by user interface framework to show functionalities to users involved in a process. Technically, currently application and entity services are modeled as a session bean according to the EJB (Enterprise Java Bean) 2.0 specifications. Following figure presents programming model available in CAF framework.

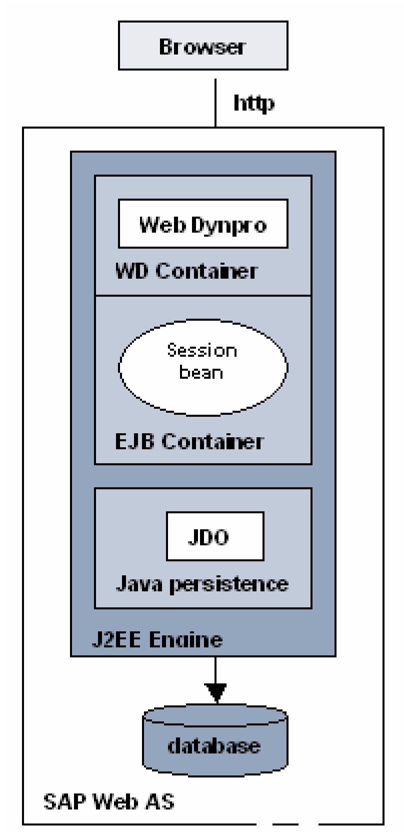


Figure 29 CAF: Programming model

Step 2
From metadata automatically is generated code of java classes, tables and data dictionary components.

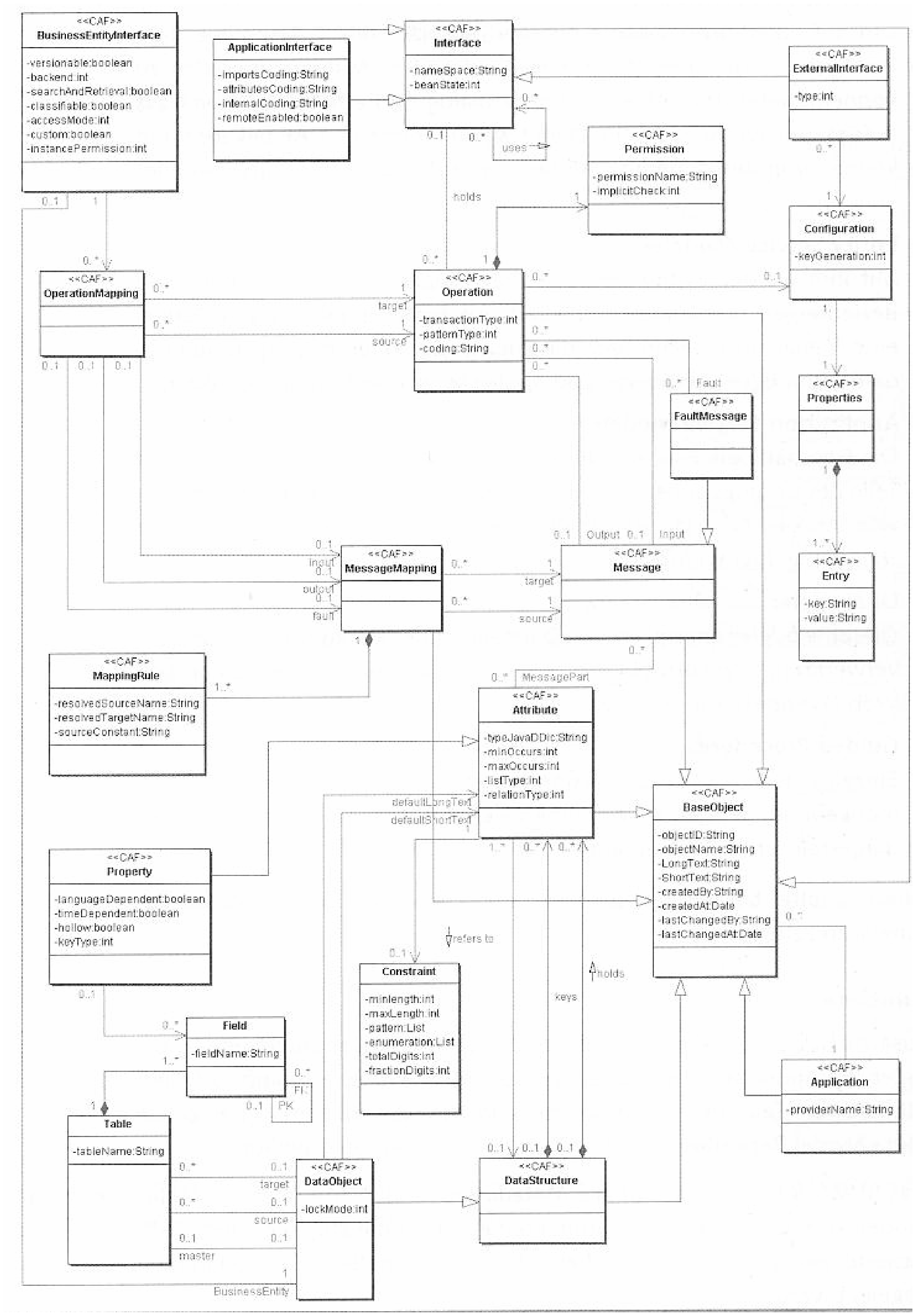


Figure 30 CAF-Metamodel[27]**.**

27 [xApps-Herger]

Step 3
The 3rd step comprises of an integration of backend systems through mapping dependencies between entity services and external services.

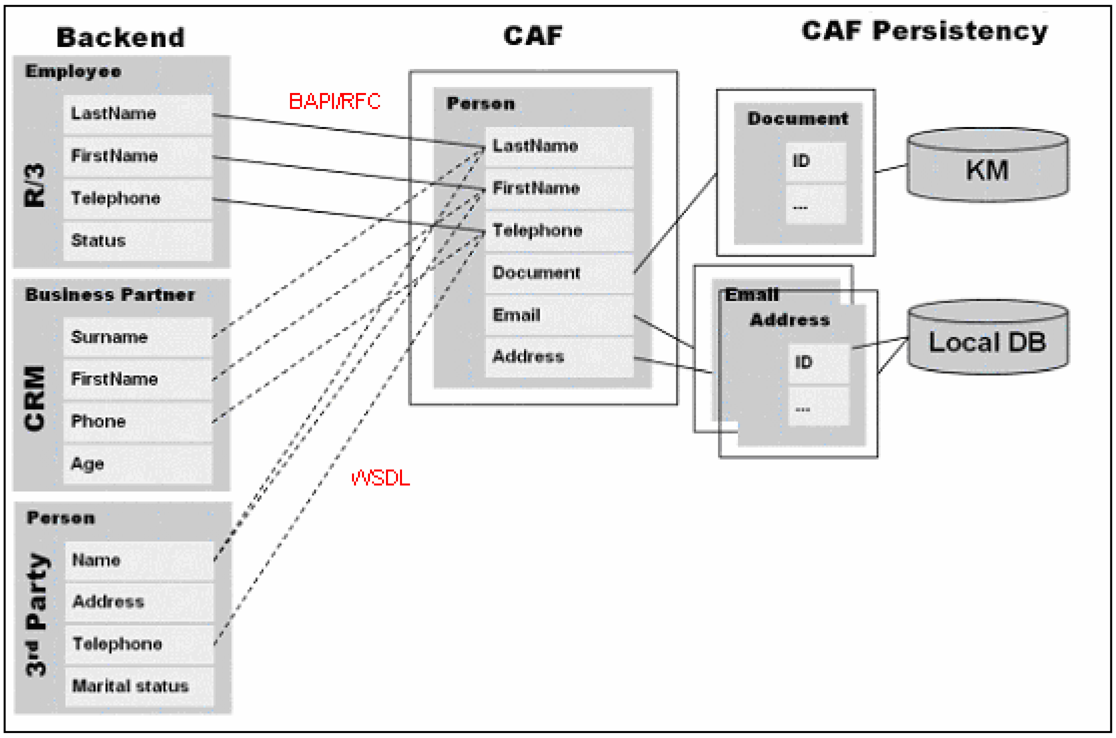


Figure 31 Mapping between different backend systems[28]

Step 4

Assemble composite application from user interfaces patterns, services and process workflows. Here, we configure UI patterns and define process applying CAF-web based tools. Following figure presents synergy of composition within Composite Application Framework. External services are mapped to entity services or directly used by user interfaces framework.

[28] SAP training materials

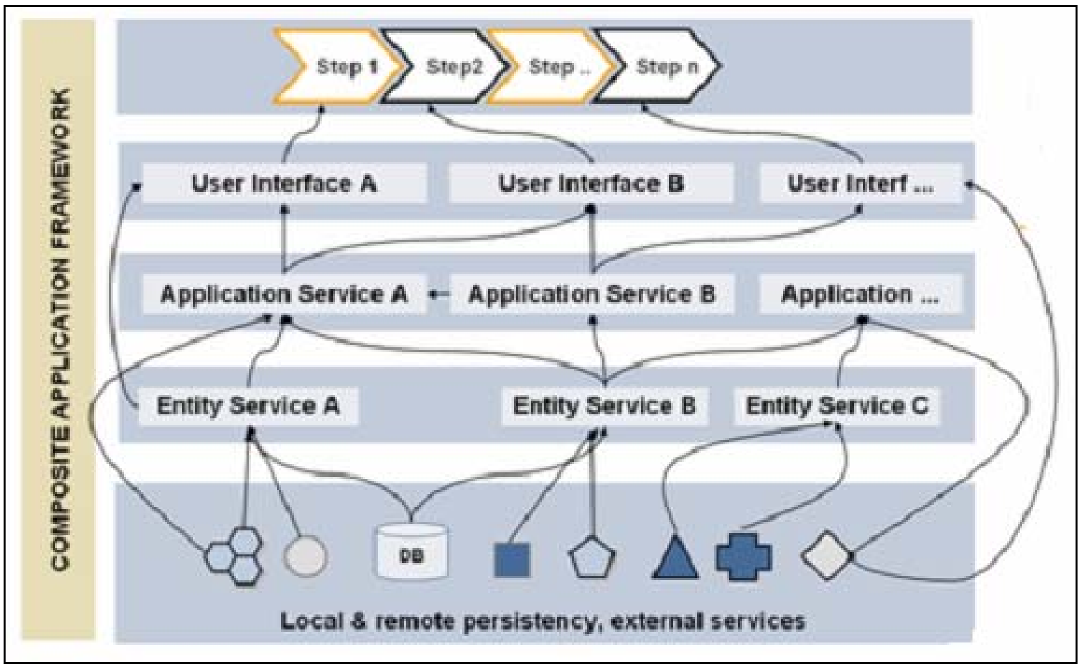


Figure 32 Synergy within composite applications[29].

3.3.2 Services Layer

The Service layer is an abstraction layer which exposes business objects that exist within a company to the outside world as services. Such services afterward show data of business objects through user interfaces (UI) to the end users or to other services.

The Service Modeler is used to model and generate services i.e. application services, entity, and external services. Following figure presents architecture of Service Modeler. The Service Modeler consist of Object Access Layer (OAL) Design time and OAL Runtime components. OAL Design time is build from Composite Application Services component which is a plug-in for SAP NetWeaver Developer Studio (figure 34). It includes a set of modelers for entity and application services, and Metadata API component. In the Generator Framework for modeled services are generated EJB session beans, web services, tables, and data-dictionary.

[29] SAP training materials

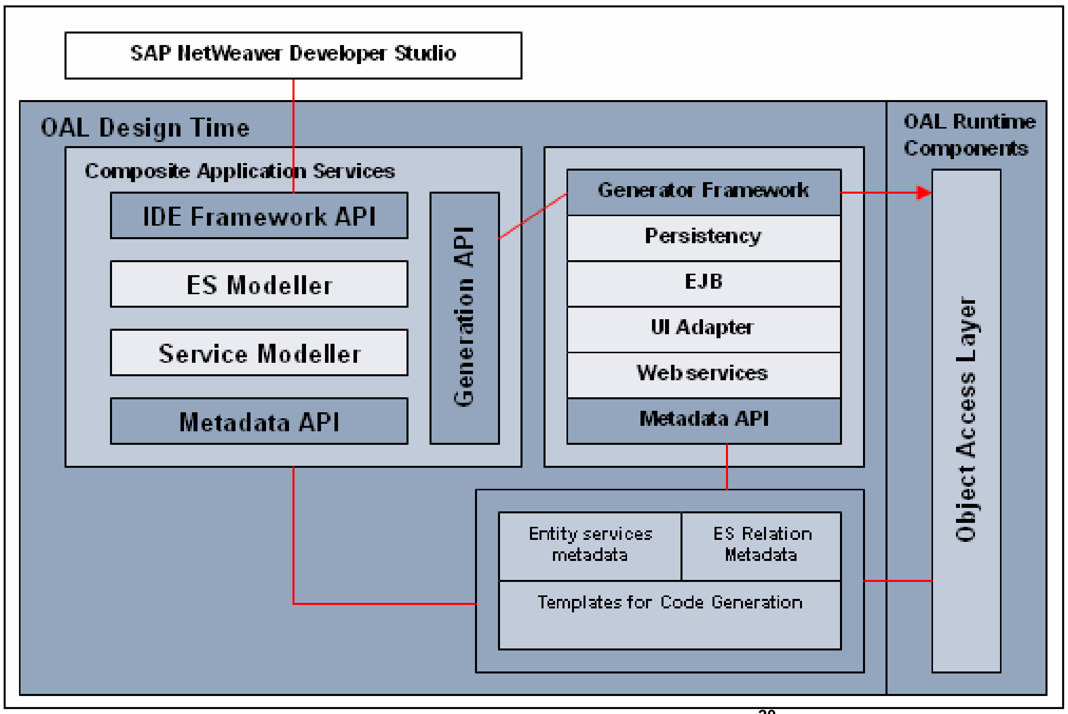
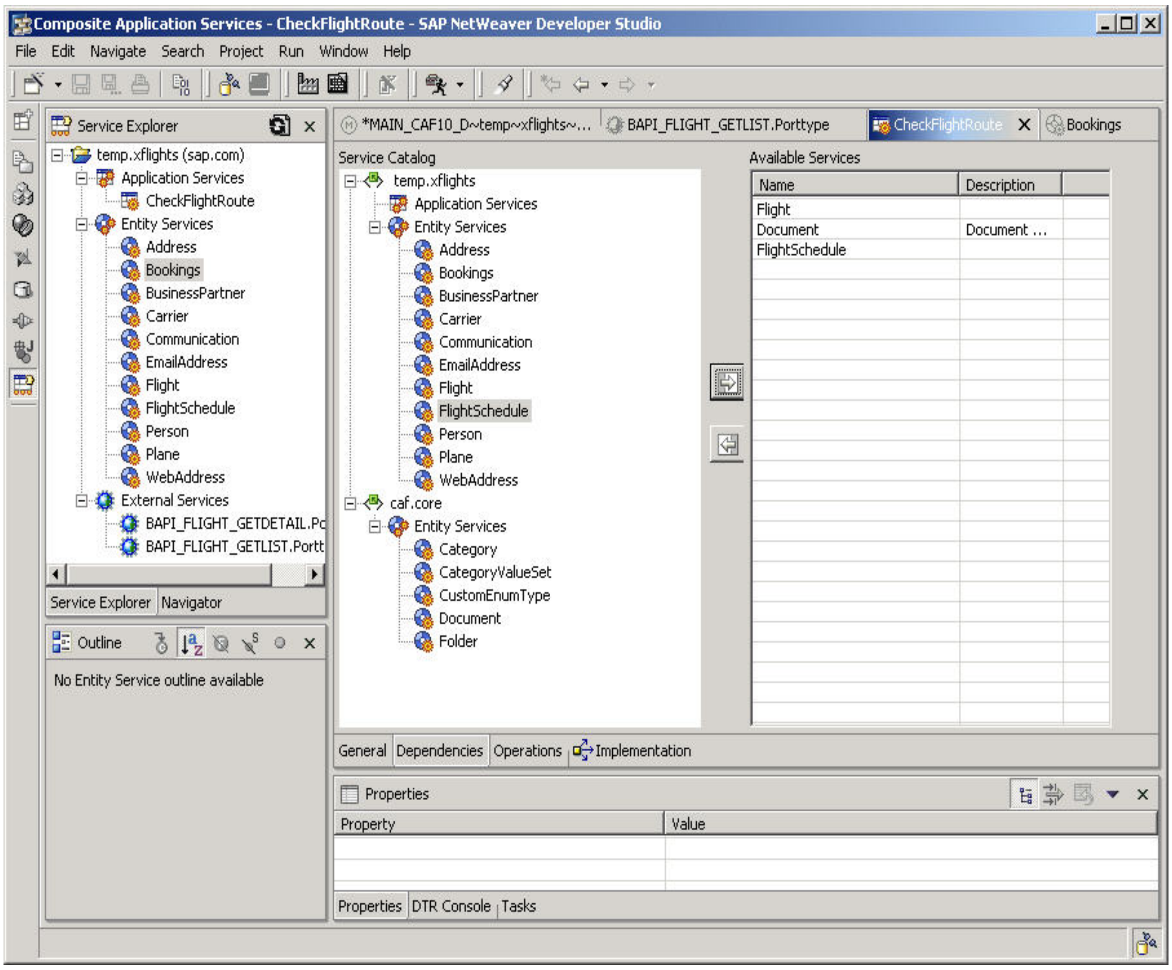


Figure 33 Architecture of Service Modeler[30]


Composite Application Services - CheckFlightRoute - SAP NetWeaver Developer Studio

File Edit Navigate Search Project Run Window Help

Service Explorer

```
temp.xflights (sap.com)
  Application Services
    CheckFlightRoute
  Entity Services
    Address
    Bookings
    BusinessPartner
    Carrier
    Communication
    EmailAddress
    Flight
    FlightSchedule
    Person
    Plane
    WebAddress
  External Services
    BAPI_FLIGHT_GETDETAIL.Pc
    BAPI_FLIGHT_GETLIST.Portt
```

Service Explorer | Navigator

Outline

No Entity Service outline available

*MAIN_CAF10_D~temp~xflights~... | BAPI_FLIGHT_GETLIST.Porttype | CheckFlightRoute | Bookings

Service Catalog

```
temp.xflights
  Application Services
  Entity Services
    Address
    Bookings
    BusinessPartner
    Carrier
    Communication
    EmailAddress
    Flight
    FlightSchedule
    Person
    Plane
    WebAddress
caf.core
  Entity Services
    Category
    CategoryValueSet
    CustomEnumType
    Document
    Folder
```

Available Services

Name	Description
Flight	
Document	Document ...
FlightSchedule	

General | Dependencies | Operations | Implementation

Properties

Property	Value

Properties | DTR Console | Tasks

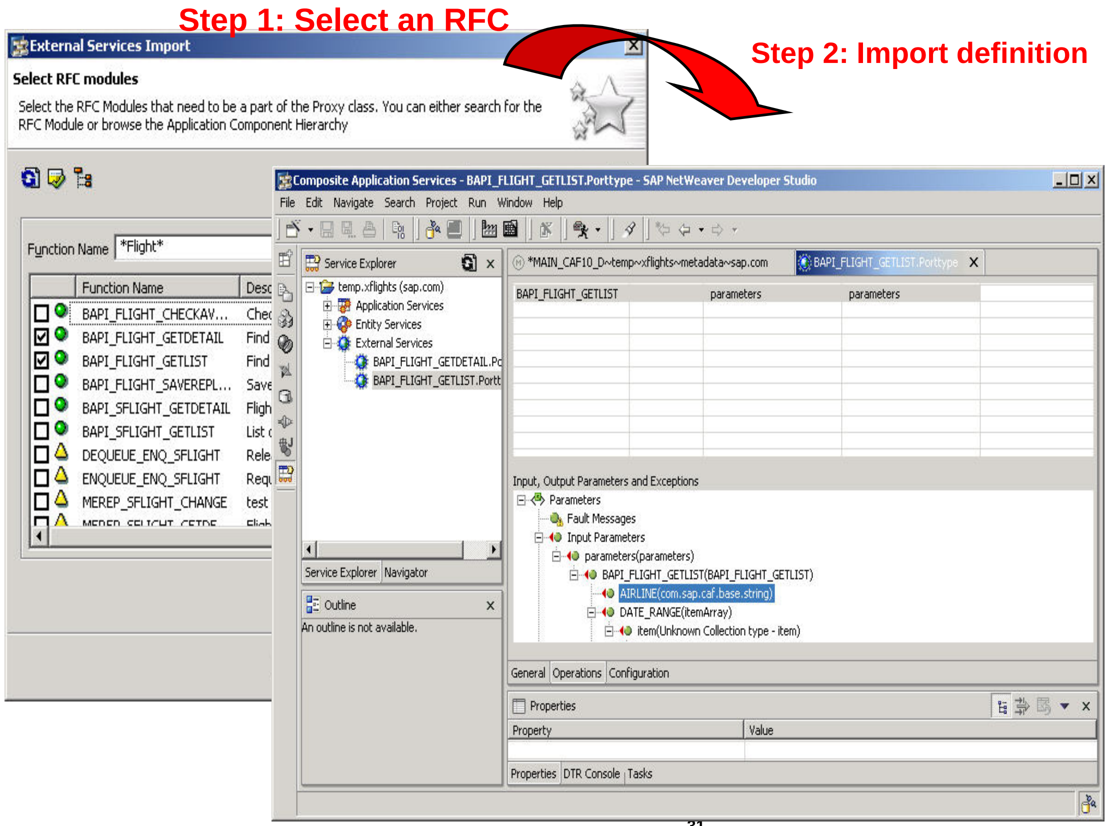
Figure 34 SAP NetWeaver Developer Studio.

30 [sap-caf-ug]

In SAP NetWeaver Developer Studio you can model the external, entity and application services.

External Services

Within CAF, functionality of backend systems can be imported to composite applications as services either through Remote Functional Calls (RFCs) or web services (WSDL). Such imported services are called External Services. External services cannot be changed during Design Time or Runtime in CAF. Currently there is one restriction that web services must have one namespace only. The definition of an external service technically specifies operations with arguments, argument types, and return type as well as remote system and its type. External services can be either reused by an entity service and/or application services or directly connected to UI patterns in order to access and expose data of business objects to end users. Following figure presents steps to import a service from external systems.


Step 1: Select an RFC
Step 2: Import definition
External Services Import
Select RFC modules
Select the RFC Modules that need to be a part of the Proxy class. You can either search for the RFC Module or browse the Application Component Hierarchy
Function Name *Flight*
Composite Application Services - BAPI_FLIGHT_GETLIST.Porttype - SAP NetWeaver Developer Studio
File Edit Navigate Search Project Run Window Help
Service Explorer
temp.xflights (sap.com)
Application Services
Entity Services
External Services
BAPI_FLIGHT_GETDETAIL.Pc
BAPI_FLIGHT_GETLIST.Portt
Service Explorer Navigator
Outline
An outline is not available.
*MAIN_CAF10_D~temp~xflights~metadata~sap.com
BAPI_FLIGHT_GETLIST.Porttype
BAPI_FLIGHT_GETLIST parameters parameters
Input, Output Parameters and Exceptions
Parameters
Fault Messages
Input Parameters
parameters(parameters)
BAPI_FLIGHT_GETLIST(BAPI_FLIGHT_GETLIST)
AIRLINE(com.sap.caf.base.string)
DATE_RANGE(itemArray)
item(Unknown Collection type - item)
General Operations Configuration
Properties
Property Value
Properties DTR Console Tasks

	Function Name	Desc
☐	BAPI_FLIGHT_CHECKAV...	Chec
☑	BAPI_FLIGHT_GETDETAIL	Find
☑	BAPI_FLIGHT_GETLIST	Find
☐	BAPI_FLIGHT_SAVEREPL...	Save
☐	BAPI_SFLIGHT_GETDETAIL	Fligh
☐	BAPI_SFLIGHT_GETLIST	List (
☐	DEQUEUE_ENQ_SFLIGHT	Rele
☐	ENQUEUE_ENQ_SFLIGHT	Requ
☐	MEREP_SFLIGHT_CHANGE	test

Figure 35 External Services[31]

Entity Services

Entity Services are models of simple business objects that exist in a company. They are used to store and access data of a business object. Theses data can be stored in a local backend system such as R/3 system. An example of an entity service can be a customer, invoice, address, document business object, etc. An Entity service is the basic part of an application.

[31] SAP training materials

The definition of an Entity Service technically specifies:

- Table(s) – attributes and relations
- Lifecycle methods (CRUD) – Create, Read, Update, and Delete
- Mappings – proxy or external services
- Special modeling – language dependencies, attribute properties
- Special functions – logging, tracing, permission checks, and interfaces.

For an entity service you can specify the following properties:

- General
- Attributes
- Operations
- Persistency
- Data source
- Permissions
- Implementation

General property displays and allows changing some general administrative information of an entity service. Here are defined five standard simple attributes that cannot be changed or deleted:

- Key (GUID)
- createdAt (timestamp)
- createdBy (User)
- lastChangedAt (timestamp)
- lastChangedBy (User)

Attributes property maintain attributes and their properties of an entity service. Three types of attributes can be created: simple attribute, complex attribute that is build from at least two simple types, and entity service attribute that is a relation to another entity service.

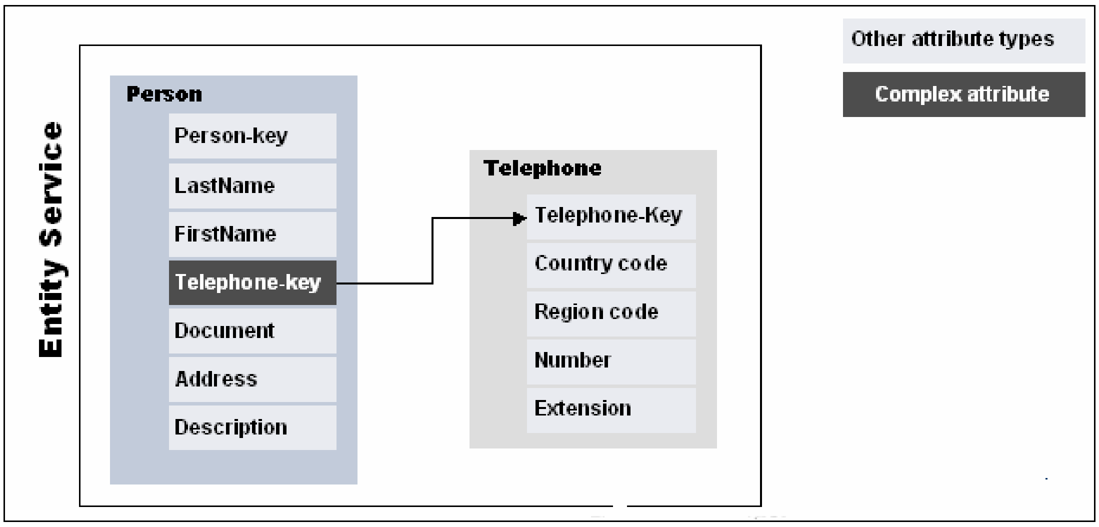


Figure 36 Complex attribute[32]

[32] SAP Training materials

Operations property has findBy-operations for an entity service. CAF distinguish between default operations and custom operations. Default operations are lifecycle methods (CRUD) and *findByPrimaryKey*. Custom operations are any other findBy-operations. If an attribute is set as key or as mandatory, it will be an argument in the CREATE-method.

Persistency property displays tables and allows changing local persistency settings for an entity service. Tables are created as follows. One table is created for language dependent attributes. Tables are created for each complex attribute as well as for each entity service attribute. Per default, the persistency is always set to “Local Persistency”. In order to change that the checkbox has to be unchecked and on the tabstrip “Datasource” and external service has to be assigned to the corresponding methods.

Datasource property displays all mappings to a remote persistency of an entity service. Here are defined the corresponding external services for each operation-type, field-mappings for each operation, and parameters to be passed.

Permissions property defines permissions settings for an entity service. Permission check means whether a user is allowed to see data presented by an entity service. Permissions on instance level are checked for every instance of an entity service. The Permission settings have an effect on the coding, as permission check calls to the User Management Engine (UME) will be executed.

Implementation property displays the coding. This coding is purely generated and cannot be manually changed. It can only be changed from changing the definitions and properties on the other properties of the Entity Service while modeling.

Application Services

The flexibility of being able to code for specific applications needs (business logic) is supported by Application Services. Application services use both entity and external services in an application and present their data to the user interface. Application services can also become itself services (web services).

The definition of an application service technically specifies:

- State full or stateless session bean (EJB)
- Web service enabled or disabled
- Dependencies to other application, entity and external services
- Custom operations and custom code

Application Service has the following properties:

- General
- Dependencies
- Operations
- Implementation

General property specifies standard information for identifying the application service. You can gather information about the service creation date, global unique identifier, or when the object was last changed. Here you can determinate whether an application service will be stateful or stateless service and enable it as web service.

Dependencies property is used to maintain dependencies to external, entity and other application services. Dependencies are necessary to be able to use the operations of those services.

Operations property tab page is used to create operations in application services. Only operations with return parameters and arguments can be created in application services. There are 6 operations-types

- CREATE
- READ
- UPDATE
- DELETE
- FINDBY
- CUSTOM

Depending on the type chosen, the operation requires certain input parameters, return-types and exceptions. The operations-wizard supports a developer in doing this.

Implementation property displays the generated and implemented coding used in an application service. You can change or manipulate coding of an application service, however only in designed areas – between custom tags.

3.3.3 UI Layer

User Interface (UI) Layer is an abstraction layer that consumes all services and its data, and exposes them to the end user through user interfaces. Within CAF, user interfaces can be built by applying generic UI patterns and/or by implementing foundation Web Dynpro applications. Such user interfaces can afterward be combined through workflow-patterns (Guided Procedures) in order to create a specific business process within a company.

User interface (UI) components are key elements in the development of user interfaces for browser-based business applications. UI components are the cornerstones in creating reusable user interface patterns in Web Dynpro technology.

The UI development environment for SAP, Web Dynpro technology, has a model-driven approach that minimizes coding and uses visual tools to design and reuse UI components. It ensures a clear separation of user interfaces and backend services, and gives developers a full control of the generated code at all stages of the development process.

User interface (UI) patterns are configurable templates designed to simplify the creation of screen layouts in Web Dynpro technology. The use of UI patterns supports the uniform layout and navigation paradigms of user interfaces and can be individually configured for use in different applications as per the intentions of developers or users. By reusing existing patterns, developers take advantage of already configured functions without needing additional coding. This pattern-based approach is therefore ideal for user interface generation.

There are two types of user interface patterns:

- Component patterns
- Page patterns

A component pattern is a UI pattern that is fully embedded in a Web Dynpro application. Component patterns are embedded for use in various other UI patterns such as page patterns. Page patterns provide a user interface layout of combined component patterns. Page patterns are themselves composite applications, which contain interfaces that plug into other components. Below are listed all available UI patterns

Component patterns	**Page patterns**
Attachment	Object Editor
Classification Assignment	Object Selector
FlexTree	
History Log	
Knowledge Management File Select	
Object Browser	
Search Bar	
User Assignment	

The Attachment pattern can be used to attach any type of a document or file to a business object instance. These documents are stored in Knowledge Warehouse. The Attachment pattern is a UI component pattern that can be embedded in any composite application.

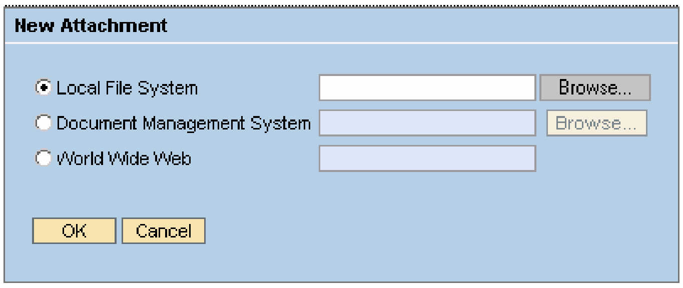
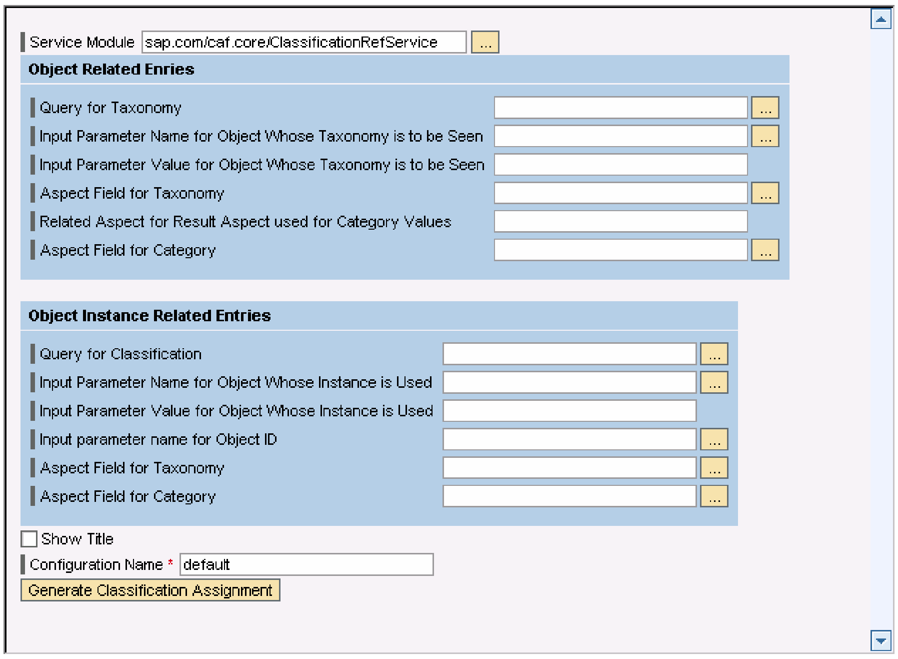


Figure 37 Attachment Pattern

The Classification Assignment pattern can be used to add classification values to an object's instance. It is a UI component pattern that can be embedded in any composite application. It can be configured at implementation time to display classification taxonomies assigned to an object.



Service Module	sap.com/caf.core/ClassificationRefService

Object Related Enries

Query for Taxonomy	
Input Parameter Name for Object Whose Taxonomy is to be Seen	
Input Parameter Value for Object Whose Taxonomy is to be Seen	
Aspect Field for Taxonomy	
Related Aspect for Result Aspect used for Category Values	
Aspect Field for Category	

Object Instance Related Entries

Query for Classification	
Input Parameter Name for Object Whose Instance is Used	
Input Parameter Value for Object Whose Instance is Used	
Input parameter name for Object ID	
Aspect Field for Taxonomy	
Aspect Field for Category	

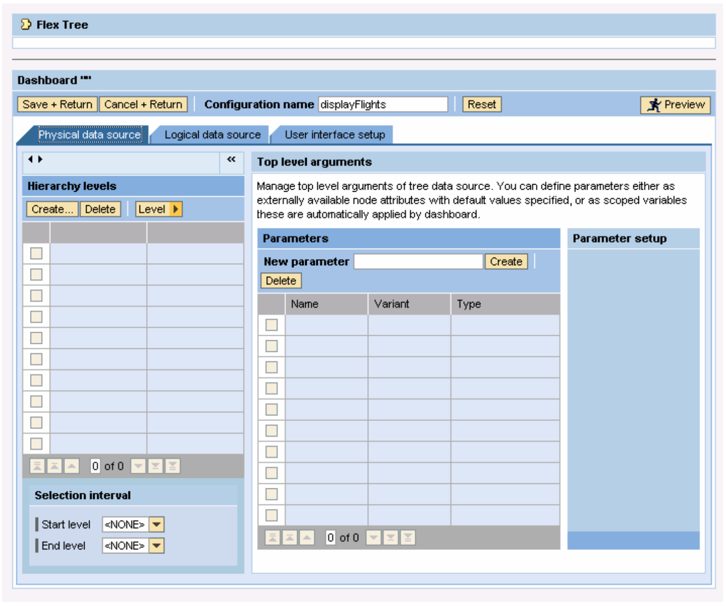
☐ Show Title

Configuration Name * default

Generate Classification Assignment

Figure 38 Classification Assignment Pattern

The FlexTree pattern enables developers to display lists of data in a table according to predefined hierarchical levels. The table uses data from different application services to populate the table. The data can appear as text, hyperlinks, graphics, or icons.



Configuration name displayFlights

Top level arguments

Manage top level arguments of tree data source. You can define parameters either as externally available node attributes with default values specified, or as scoped variables these are automatically applied by dashboard.

New parameter

	Name	Variant	Type

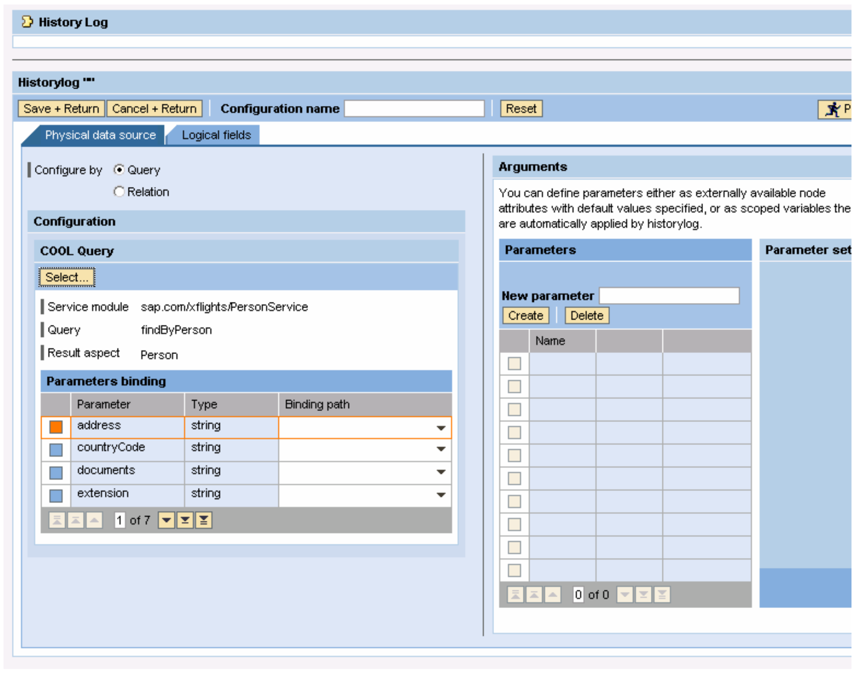
Selection interval

Start level <NONE>

End level <NONE>

Figure 39 FlexTree Pattern

History Log Pattern enables developers to display additional comments about an entity service during application runtime. Comments are read-only and displayed in chronological order. As a result, developers can see information about any entity without having to look into the entity service itself.



History Log

Historylog ""

Save + Return | Cancel + Return | Configuration name | Reset

Physical data source | Logical fields

Configure by (● Query) ○ Relation

Configuration

COOL Query

Select...

Service module	sap.com/xflights/PersonService
Query	findByPerson
Result aspect	Person

Parameters binding

	Parameter	Type	Binding path
🟧	address	string	
🟦	countryCode	string	
🟦	documents	string	
🟦	extension	string	

1 of 7

Arguments

You can define parameters either as externally available node attributes with default values specified, or as scoped variables the are automatically applied by historylog.

Parameters | **Parameter set**

New parameter

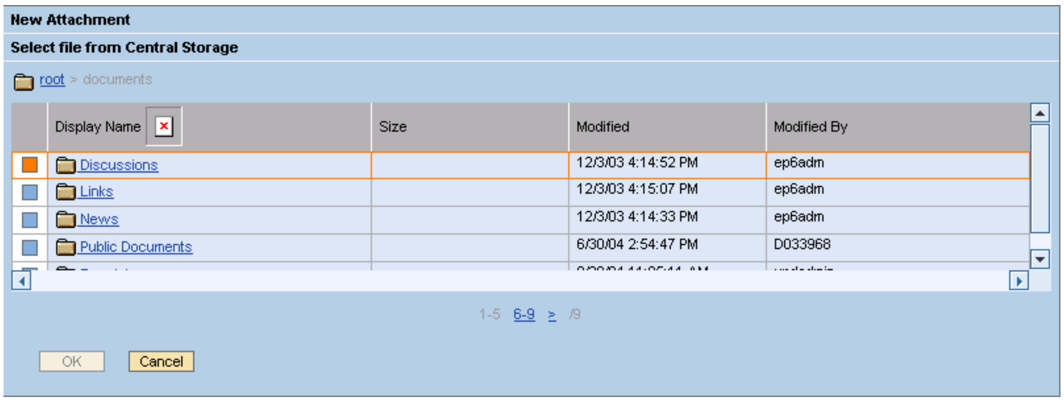
Create | Delete

	Name		

0 of 0

Figure 40 History Log Pattern

The KM File Select pattern can be used to select directories and data files from the KM repository. It is a UI component that can be embedded in any composite application.



New Attachment

Select file from Central Storage

root > documents

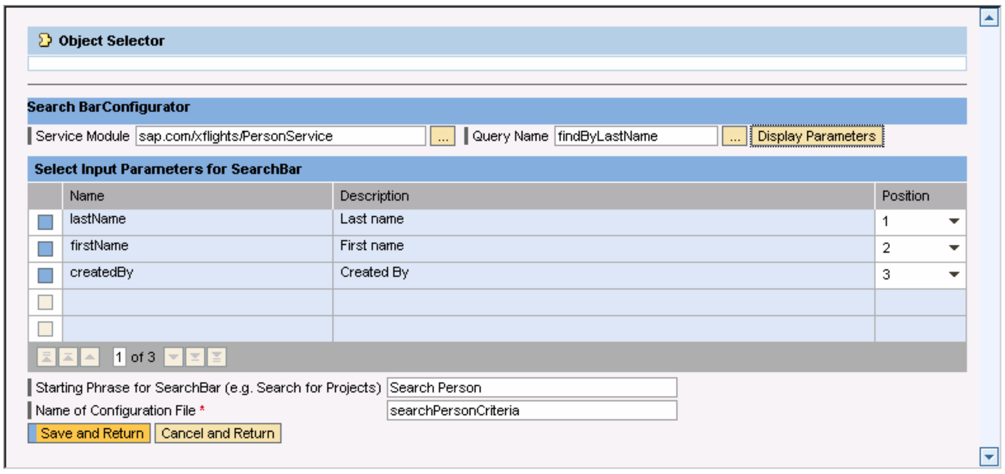
	Display Name	Size	Modified	Modified By
🟧	Discussions		12/3/03 4:14:52 PM	ep6adm
🟦	Links		12/3/03 4:15:07 PM	ep6adm
🟦	News		12/3/03 4:14:33 PM	ep6adm
🟦	Public Documents		6/30/04 2:54:47 PM	D033968

1-5 6-9 ≥ /9

OK | Cancel

Figure 41 KM File Select Pattern

The Search Bar pattern can be used to perform simple searches for specified objects. It can be embedded into any composite application and used in combination with other UI patterns. The Search Bar pattern is also embedded into the Object Selector and Object Browser UI patterns.



Object Selector

Search BarConfigurator

Service Module: sap.com/xflights/PersonService ... Query Name: findByLastName ... Display Parameters

Select Input Parameters for SearchBar

	Name	Description	Position
☑	lastName	Last name	1
☑	firstName	First name	2
☑	createdBy	Created By	3
☐			
☐			

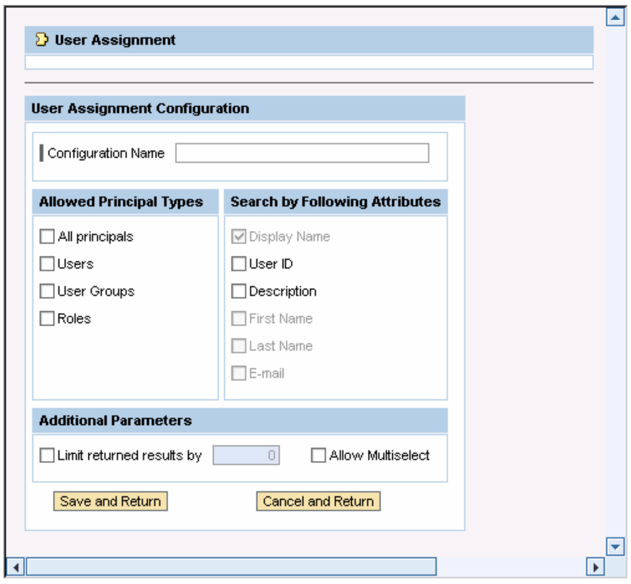
1 of 3

Starting Phrase for SearchBar (e.g. Search for Projects)	Search Person
Name of Configuration File *	searchPersonCriteria

Save and Return | Cancel and Return

Figure 42 Search Bar Pattern

The User Assignment pattern can be used to assign users to specific roles. The data basis for the users is the User Management Extension (UME). Generic user information is retrieved by the UME API.



User Assignment

User Assignment Configuration

Configuration Name:

Allowed Principal Types	Search by Following Attributes
☐ All principals	☑ Display Name
☐ Users	☐ User ID
☐ User Groups	☐ Description
☐ Roles	☐ First Name
	☐ Last Name
	☐ E-mail

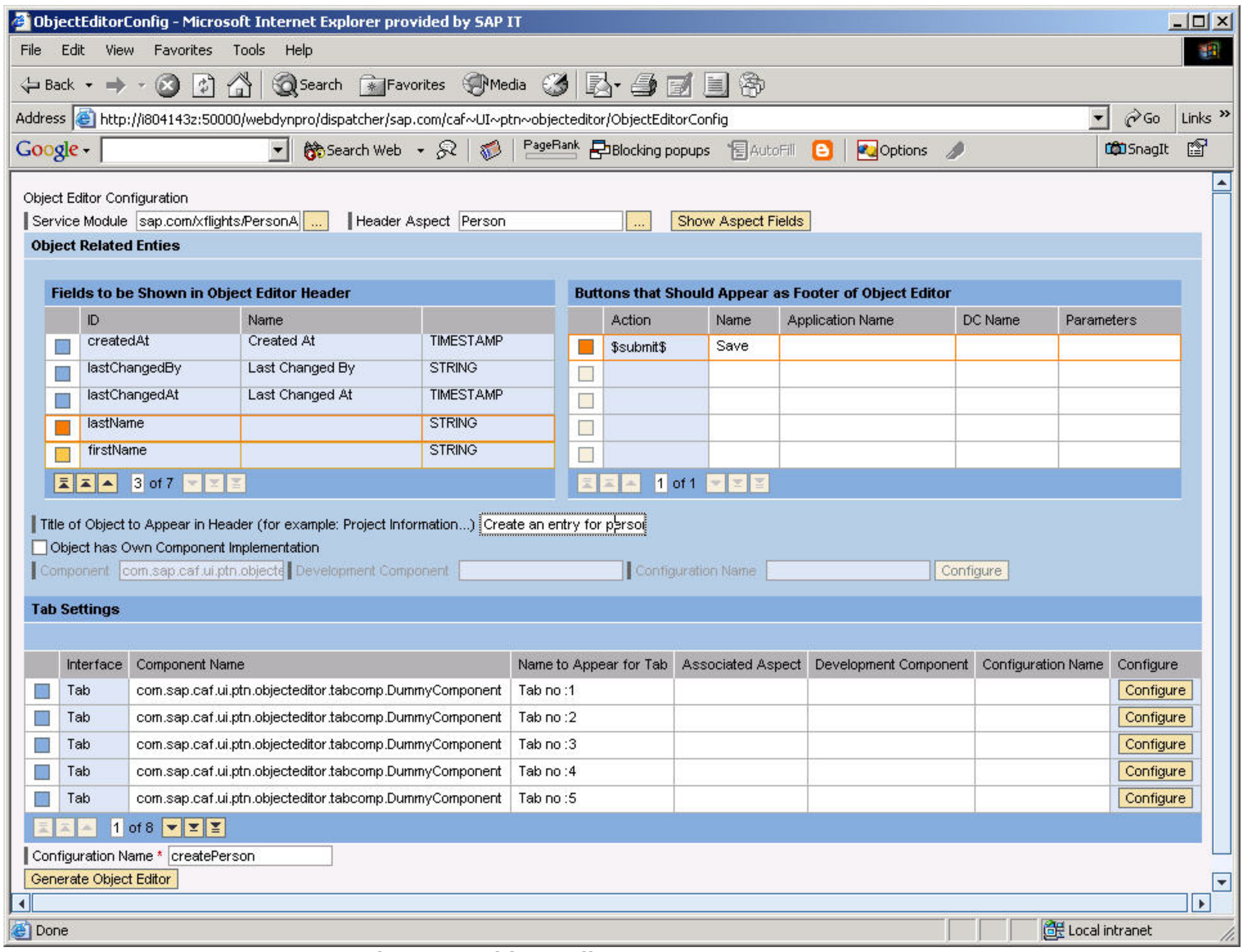
Additional Parameters

☐ Limit returned results by 0 ☐ Allow Multiselect

Save and Return | Cancel and Return

Figure 43 User Assignment Pattern

The Object Editor Pattern can be used to design, create, or edit objects attributes within an application. The resulting editor, whose components are provided as component interface implementations for interface definitions in the Object Editor, contains the Attachments and Classification Assignment patterns.


ObjectEditorConfig - Microsoft Internet Explorer provided by SAP IT
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://i804143z:50000/webdynpro/dispatcher/sap.com/caf~UI~ptn~objecteditor/ObjectEditorConfig
Go Links
Google Search Web PageRank Blocking popups AutoFill Options SnagIt
Done
Local intranet

Object Editor Configuration

Service Module sap.com/xflights/PersonA ... Header Aspect Person ... Show Aspect Fields

Object Related Enties

Fields to be Shown in Object Editor Header

	ID	Name	
	createdAt	Created At	TIMESTAMP
	lastChangedBy	Last Changed By	STRING
	lastChangedAt	Last Changed At	TIMESTAMP
	lastName		STRING
	firstName		STRING

3 of 7

Buttons that Should Appear as Footer of Object Editor

	Action	Name	Application Name	DC Name	Parameters
	$submit$	Save			

1 of 1

Title of Object to Appear in Header (for example: Project Information...) Create an entry for perso

☐ Object has Own Component Implementation

Component com.sap.caf.ui.ptn.objecte Development Component Configuration Name Configure

Tab Settings

	Interface	Component Name	Name to Appear for Tab	Associated Aspect	Development Component	Configuration Name	Configure
	Tab	com.sap.caf.ui.ptn.objecteditor.tabcomp.DummyComponent	Tab no :1				Configure
	Tab	com.sap.caf.ui.ptn.objecteditor.tabcomp.DummyComponent	Tab no :2				Configure
	Tab	com.sap.caf.ui.ptn.objecteditor.tabcomp.DummyComponent	Tab no :3				Configure
	Tab	com.sap.caf.ui.ptn.objecteditor.tabcomp.DummyComponent	Tab no :4				Configure
	Tab	com.sap.caf.ui.ptn.objecteditor.tabcomp.DummyComponent	Tab no :5				Configure

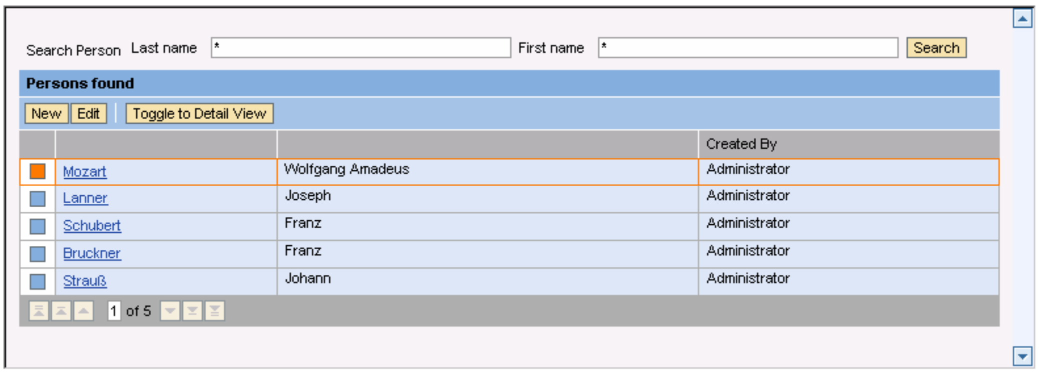
1 of 8

Configuration Name * createPerson

Generate Object Editor

Figure 44 Object Editor Pattern

The Object Selector pattern can be used to search for and list aspect attributes of an object. The pattern is a complete user interface layout, which encompasses the component patterns; the Search Bar and the Object List. The Object Selector pattern uses the Search Bar to search for single objects from a list of items based on specific criteria. You can use the Object List pattern to list objects in either a tabular form or list layout. The resulting layout can be used to create new or edit existing objects.



Search Person Last name * First name * Search

Persons found

New Edit Toggle to Detail View

			Created By
	Mozart	Wolfgang Amadeus	Administrator
	Lanner	Joseph	Administrator
	Schubert	Franz	Administrator
	Bruckner	Franz	Administrator
	Strauß	Johann	Administrator

1 of 5

Figure 45 Object Selector Pattern – runtime

3.3.4 Process Layer

The Process Layer is an abstraction layer that defines step-by-step collaboration workflow patterns (guided procedures) for composite applications in order to create specific business processes within a company.

SAP Guided Procedures (SAP GP) are not released to clients. The following material comes from SAP marketplace and it does not cover the practical aspects of process framework within CAF environment.

SAP Guided Procedures (SAP GP) framework is designed to implement workflows with greater ease and speed across multiple applications. It enables users to easily set up and execute collaborative processes by seamlessly integrating backend system transactions and services into the process context. SAP GP differentiates between process templates and process instances. The process templates can be multiply instantiated, the process instances can be started with set parameters. Processes in SAP GP consist of phases and steps.

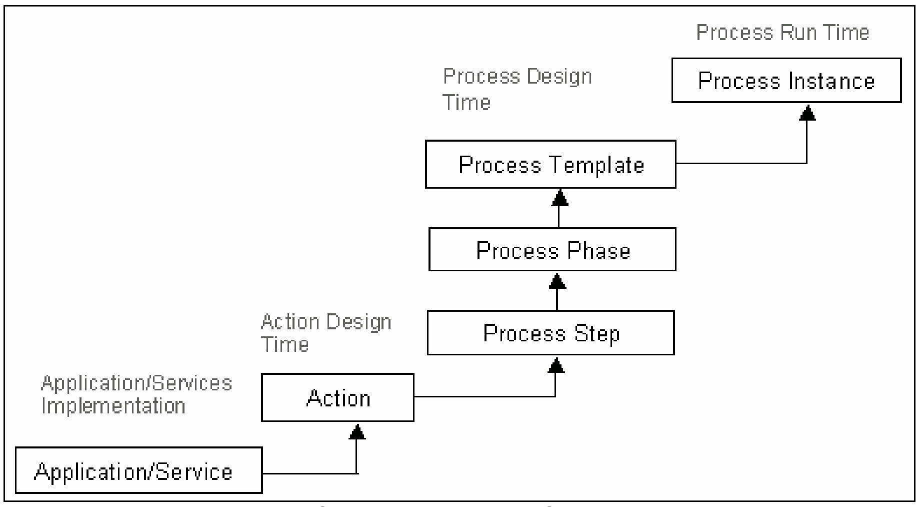


Figure 46 SAP GP - Phases

Phases are sequential – one phase must have been completed before a new one is started. A step is defined by the action assigned to it. This action describes which application or service the application calls, for example a Web Dynpro component, an iView or a form. In addition, input and output parameters are transferred and the context-dependent, ad-hoc actions are determined. Actions can be reused in various process steps. Every process has a process context. This context consists of the input parameters (resources) and output parameters (deliverables) of the individual steps. This ensures that the output parameters of preceding steps are taken into consideration as input parameters of subsequent steps. The actions can also access the process context and read data. At process design time, the user defines how the context is mapped. The design time, the configuration part of SAP GP, is a Web-based tool for modeling processes by editing predefined templates or by creating new templates. The part of SAP GP that supports the execution of business processes is called runtime. Lifecycle of Guided procedures is presented below.

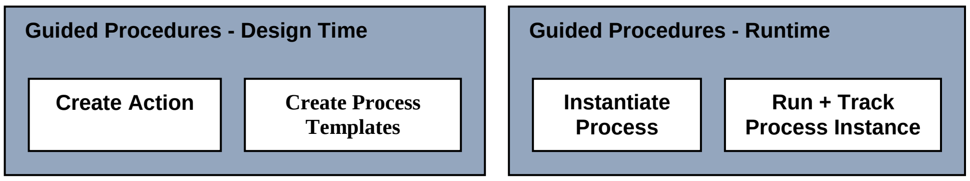

Guided Procedures - Design Time
Create Action
Create Process Templates
Guided Procedures - Runtime
Instantiate Process
Run + Track Process Instance

Figure 47 Guided Procedures - Lifecycle

SAP GP, which is a Web application (Web Dynpro application) running on the SAP Enterprise Portal (SAP EP), can describe a collaborative business process as a thread of tasks that need to be performed by the processors (that is, the end users) in a certain order to obtain the overall process goal. Some tasks, however, may be more complex than others and contain a series of subordinate tasks. Thus, in order to maintain a clear outline of the process flow, business analysts need a higher structure level which bundles associated subtasks under an upper-level task. To meet these requirements, Guided Procedures (GP) models business processes out of phases and steps. Once business analyst has designed and published a process template, authorized users can initiate process instances that need to be completed by the users or user groups assigned. It can be run several process instances of the same type independently. The following figure illustrates how a GP process can be structured.

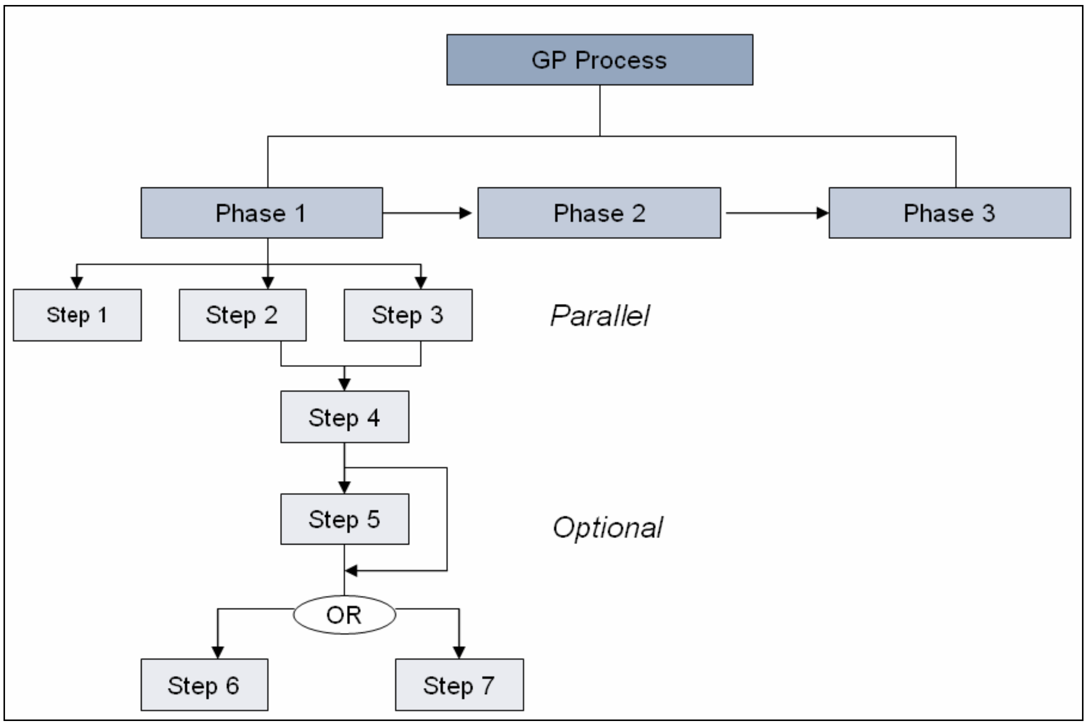


Figure 48 SAP GP process structure – example

3.3.5 CAF landscape

Following figure presents landscape of Composite Application Framework. On a client side must be installed SAP NetWeaver Studio and Composite Application Services plug-in for designing services model as well as Web browser in order to work with CAF-Web tools to configure user interfaces in order to present data to end users. On the side of SAP Web AS the Composite Application Framework must be installed. Additionally, the SAP Guided Procedure framework must be installed for implementing workflows between different applications. In SAP Enterprise Portal are all applications integrated through iViews.

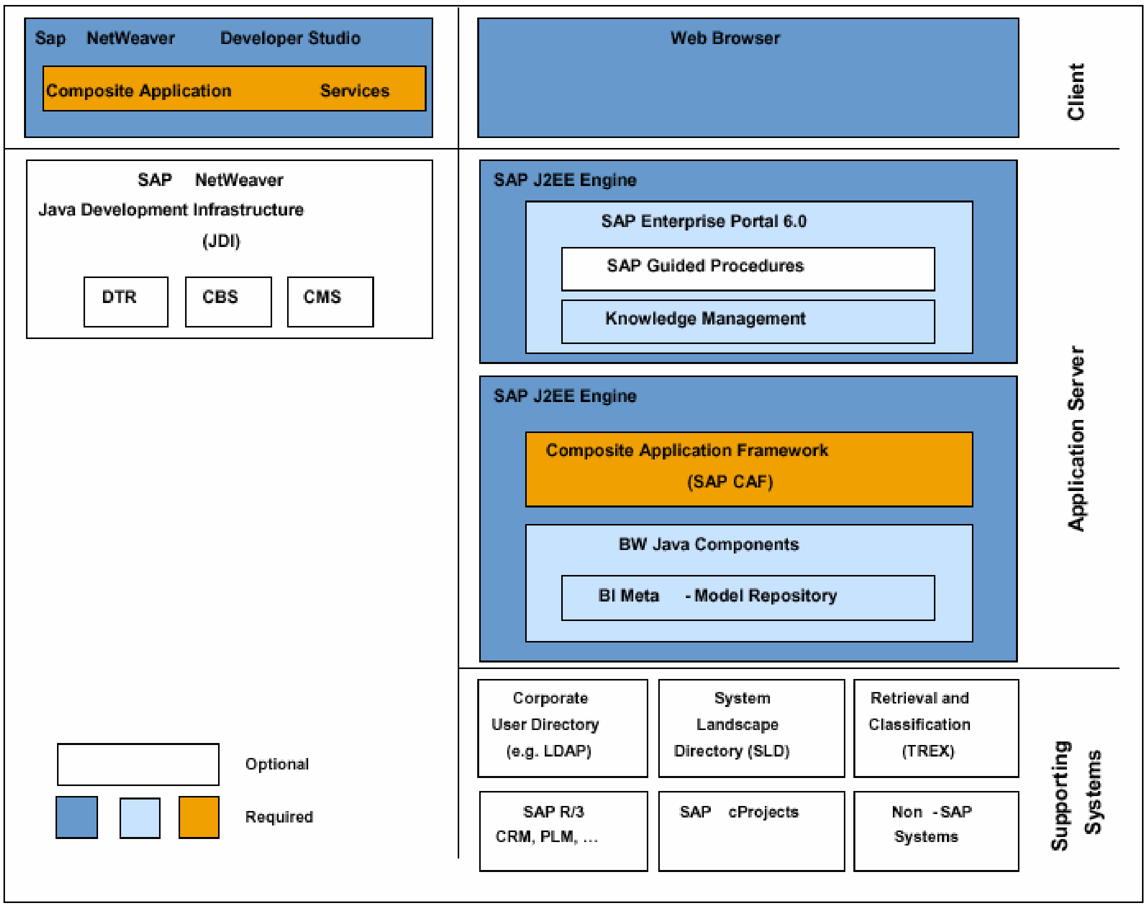

Sap NetWeaver Developer Studio
Composite Application Services
Web Browser
Client
SAP NetWeaver Java Development Infrastructure (JDI)
DTR
CBS
CMS
SAP J2EE Engine
SAP Enterprise Portal 6.0
SAP Guided Procedures
Knowledge Management
SAP J2EE Engine
Composite Application Framework (SAP CAF)
BW Java Components
BI Meta - Model Repository
Application Server
Corporate User Directory (e.g. LDAP)
System Landscape Directory (SLD)
Retrieval and Classification (TREX)
SAP R/3 CRM, PLM, ...
SAP cProjects
Non - SAP Systems
Supporting Systems
Optional
Required

Figure 49 SAP CAF landscape

3.4 Summary

Chapter 3 presented the most important and related technologies for this master thesis. The SAP NetWeaver integration and application platform is a technical foundation for the ESA platform. SAP NetWeaver supports open standards like Web services technologies as well as RFC, BAPI, IDoc interfaces. Therefore, services play a major role to provide interoperability between SAP NetWeaver platform and other systems. SAP Web AS provides Web Services Framework that offers the basic web services standards like XML, SOAP, WSDL. Also, Web Services Framework supports the WS-Security 1.0 standard. SAP Exchange Infrastructure (SAP XI) forms the basis for the integration of business processes. SAP XI provides a technical infrastructure for XML-based message exchange to enable the integration of all systems to SAP systems. Furthermore, SAP XI provides a set of integrated tools for creating and managing all integration-relevant information into business process with use of BPEL4WS standard. The SAP xApps technology is a blueprint for the package composite applications paradigm. This thesis, presents in detail the SAP Composite Application Framework. It is a unified environment for a development and running package composite applications. It encapsulates the functionality of current enterprise applications as components and services (WSDL, and RFC/BAPI). These components and services are described in metadata and implement standard frameworks for application layers such as user interface and process control. It offers the model-driven architecture so an application developer assembles an application primarily through modeling rather than through programming. The modeling technique is leveraged further through the identification and reuse of patterns that are common to many application scenarios. Below you can find summary of supported open standards by SAP NetWeaver platform that are relevant for this thesis.

SAP NetWeaver and open standards	
SAP Web AS	• XML, SOAP, WSDL, UDDI • RFC, BAPI, IDoc • JMS, JDBC, JCO, .NET Connector
SAP Exchange Infrastructure	• BPEL4WS • RosettaNet Implementation Framework (RNIF)
CAF	• WSDL, RFC, BAPI

4 Custom Composite Application

“Most problems have already been solved in this world; we just keep forgetting the solutions”

- Melissa A. Cook
Hewlett-Packard Company

This chapter outlines technical aspects of a development of custom composite applications with a use of the SAP Composite Application Framework. Firstly, it will be presented a scenario that makes it possible to apply composite applications concept as a practical solution. In the second part, it will be evaluated a methodology and practical development aspects that are offered by the Composite Application Framework. The developed model of composite application is a contribution to overall first assessment of SAP CAF product as a part of the Restricted First Customer Shipment program that is SAP’s initiative.

4.1 Customer master data at HP

One of the most important values of HP is its customers. HP is a worldwide company and has a vast number of customers, which are divided in different categories such as those purchasing direct from HP its products, those purchasing its products through some agents/brokers etc. Moreover, HP provides a great number of products, which are produced by independent HP business units (PSG, IPG). As a result over the time, there have been developed many order-management IT systems for HP business units in order to proceed the information regarding the customers of HP (Figure 48).

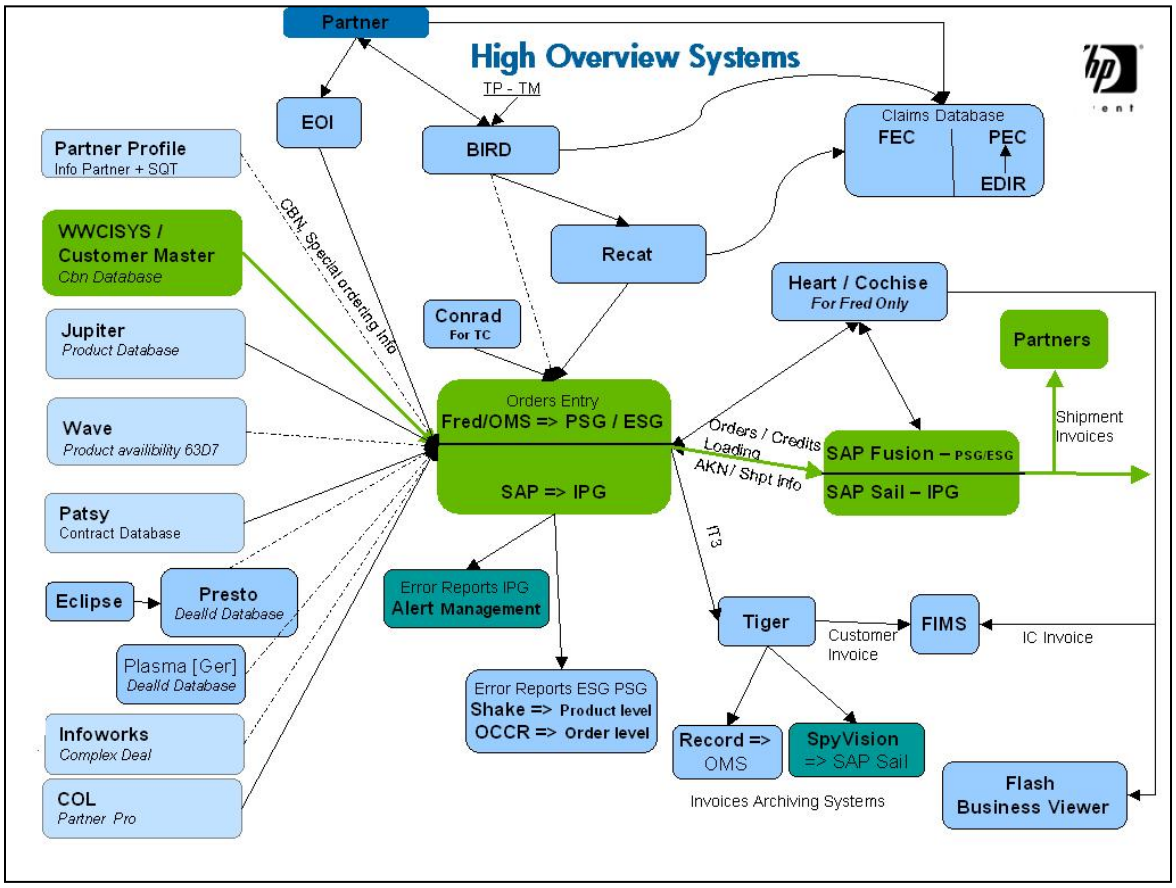


Figure 50 Customer Master Data flow

WWCISys is the common platform, used by all the HP business, to store the information regarding the customers of HP. In fact, WWCISys means World Wide Customer Information System.

The Customer Reference Server CRS supports enterprise-wide sharing of customer information required for the end-to-end Order Fulfillment processes (quote to collections) and HP's goal of improving customer relationship management. This is done through the definition of business processes and data standards, as well as through the implementation of a mechanism, which supports timely distribution of customer information to those sites where that information is required. The business processes and mechanism work together to ensure the quality and consistency of customer information.

SAP Fusion instance is used to generate the transaction in US$ but converting it into EURO before sending it to HPFO systems (conversion using the finance valid exchange rate). The Fusion program enables SAP Fusion for Value Order Management to obsolete the legacy platform (WWOMS, Tiger, PIM) within the Value Solution Order Management Environment. The main benefits are the elimination of a dual process environment and associated overhead in cost for training and support along with improvement in order entry times in defined areas.

Globalizations effects and today’s business collaboration between HP and the customers indicates that there is a need to be addressed, namely a seamless, comprehensive view on the customer data among various HP business units (thus, among numerous IT storage systems as well). The one comprehensive view should allow for sharing, updating, synchronization-replication of the customer data irrespectively of HP business units (thus, IT storage systems as well).

These needs for the current IT storage systems could be fulfilled through building composite applications that reside upon existing already applications. Composite applications called "apps on apps" allow for an aggregation of loosely-coupled various components of different systems. Therefore, composite applications can provide a common view of the customer data through consuming services (Figure 51).

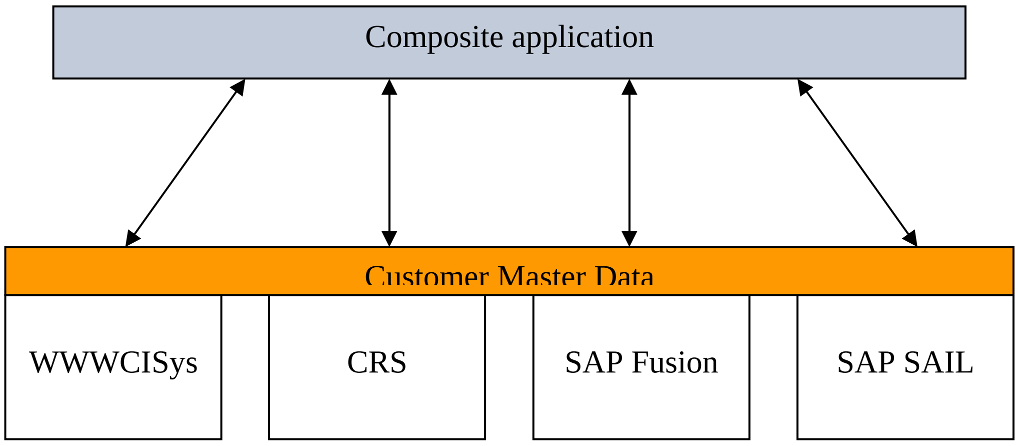


Figure 51 Unified view on customer master data

At HP the customer data mostly are processed in SAP's landscape. Since around two years SAP promotes its new brand technologies SAP NetWeaver, which allow creating custom composite applications upon existing applications. The SAP NetWeaver platform provides the Composite Application Framework (CAF) that facilitates the creation of composite applications. Therefore, SAP CoE[33] at HP has taken up an initiative to build a prototype of a custom composite application with use of SAP CAF product. Such a custom composite application should present a comprehensive view on the customer data at HP business units, in particular among WWCISys, CRS, SAP Fusion and SAP SAIL storage systems (Figure 52).

[33] SAP CoE – SAP Center of Expertise department is responsible for a research and analyze of the latest SAP technical solutions. Additionally, SAP CoE executes a number of pilot projects with regard to the researches. Based on acquired knowledge and experiences, SAP CoE teams support afterward technical teams at HP in an implementation of IT-solutions.

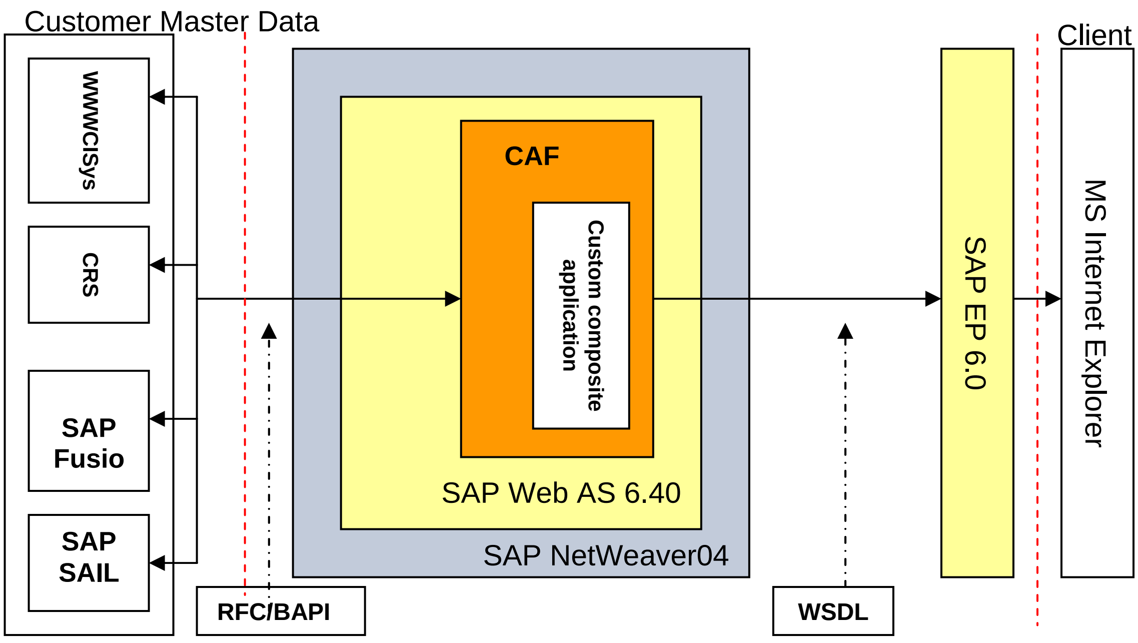


Figure 52 Prototype

Figure 52 presents a technical architecture of composite applications for our prototype. A communication between SAP R/3 systems and SAP NetWeaver platform is based on BAPIs interfaces. With application of Composite Application Framework tools the customer data between collaborating systems are integrated through a model that describes the customer data. From CAF the customer data can be accessed through web services from outside world. The Composite application can be integrated within a content management system (SAP EP 6.0) and available thus for external users through a web browser.

Firstly, I started with installation of all SAP NetWeaver components which are needed to run and develop composite applications on SAP NetWeaver platform

- SAP Web AS 6.40
- Composite Application Framework SP 2.0

Implementation of custom composite application is foreseen in two scenarios.

- SAP CAF framework with own repository (test phase)
- SAP CAF connected with backend systems (integration phase)

In the first scenario it will be implemented a custom composite application that enables maintaining Customer Data and store them in a local persistent system (CAF Repository System) (Figure 51). The Customer Data will be described by service model (entity service “customer” and one “customerAS” application entity) that exposes data through user interfaces to end users. Feed data that represent General Customer Data will be provided from plain text file or in an XML-format. The User Interface of the custom composite application will be defined with an application of available UI patterns within SAP CAF framework. Initially, for the prototype we wanted to provide basic functionalities for creating and searching of customer data. Therefore, the goal is not to build a composite application which reflects processes while working with customer data. As a result, I will design the model and configure user interfaces patterns to build a user interface. Process model with an application of SAP Guided Procedures will not be executed for this prototype.

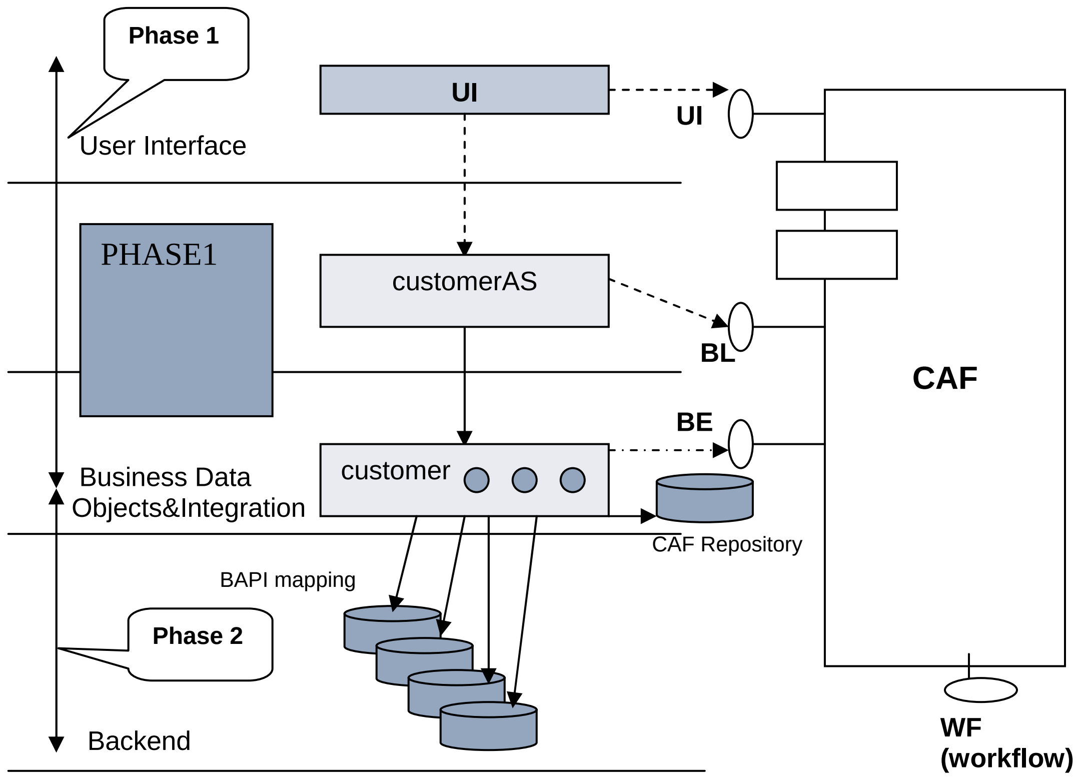


Figure 53 Phases 1&2 of Prototype

In the second scenario, the model of custom composite application will switch to the original scenario with backend-systems integration through BAPIs. A reason for that was that I wanted firstly to identify the factual capabilities of SAP CAF. Therefore, I defined simplified model for business objects of Customer Master Data. After identification of factual capabilities of SAP CAF and modeling the suitable service model the integration to backend systems should be performed.

According with the CAF concept I have started with modeling of model. This required from me to analyze business objects of all four participating systems and make an attempt to create a unified model of customer data.

4.1.1 Model

A simplified Customer Master Data view from all involved systems can be defined as (Figure 54):

- *General Data* – the General Data applies to one customer in all Company Codes and all Sales Areas. Example of General Data is address, information with transportation zone, general control data, and unloading points.
- *Sales Area Data* – the Sales Data applies to one customer only within a specified Sales Area. Example of Sales Data are pricing control data, shipping preferences, billing data, payment terms, partner functions.
- *Company Code Data* – the Company Code Data applies to one customer, only within a specified Company Code. Example of Company Code Data; account management, payment transactions.

- *Other Data* – the Other Data applies to one customer, only within a specified Company Code. Example of Other Data; sales organization data, division data, distribution channel.

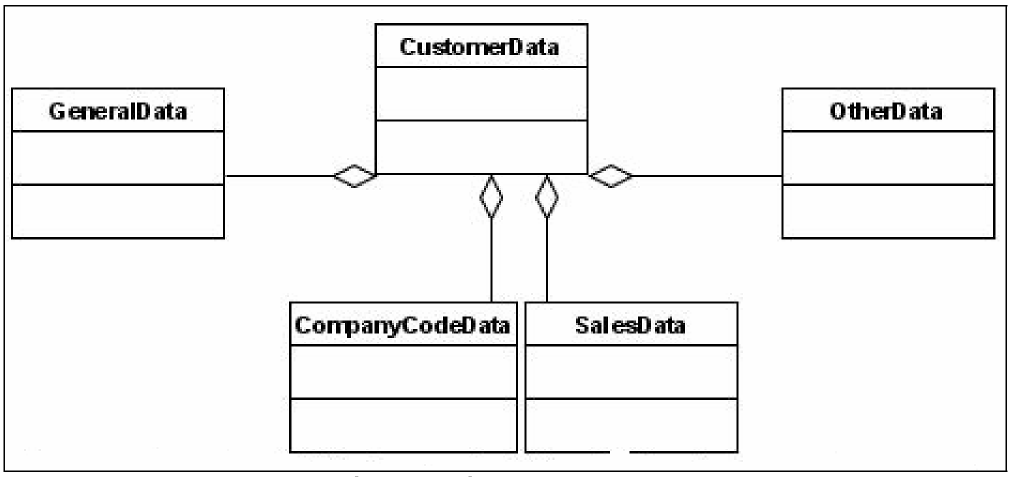


Figure 54 The Customer Data

At the very beginning, I have modeled one application service "customerAS". It represented the business logic for creating and searching data of customer data. The application service "customerAS" presents the customer data to end users through user interface. It was also enabled as a Web service that exposed the customer data to other systems. The service "customer" with a use of other entity services describes all business objects that present the Customer Master Data. Underneath you will find the first designed simplified service model of Customer Master Data.

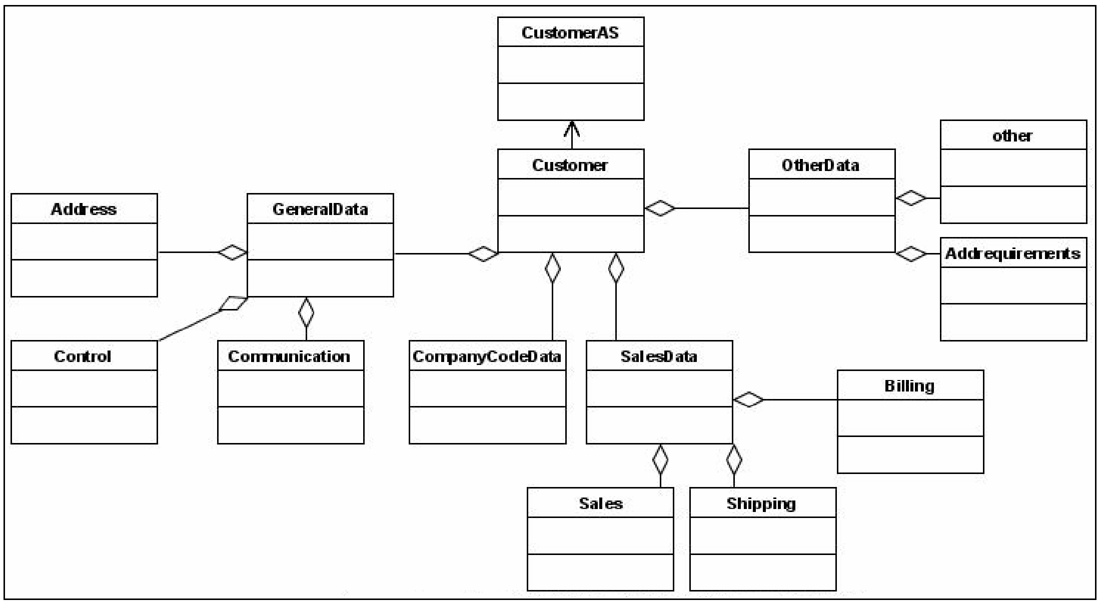


Figure 55 Simplified model of Customer Data

CAF framework based on the model generates metadata for EJB session beans, database tables, dictionary data-types. A simplified CAF-Metamodel for "CustomerAS" application service is depicted below. Also figure this explains how the basic business logic for CRUD-methods is implemented in Application Service. Within CRUD-methods of Application Service we call the CRUD-methods of Entity Service.

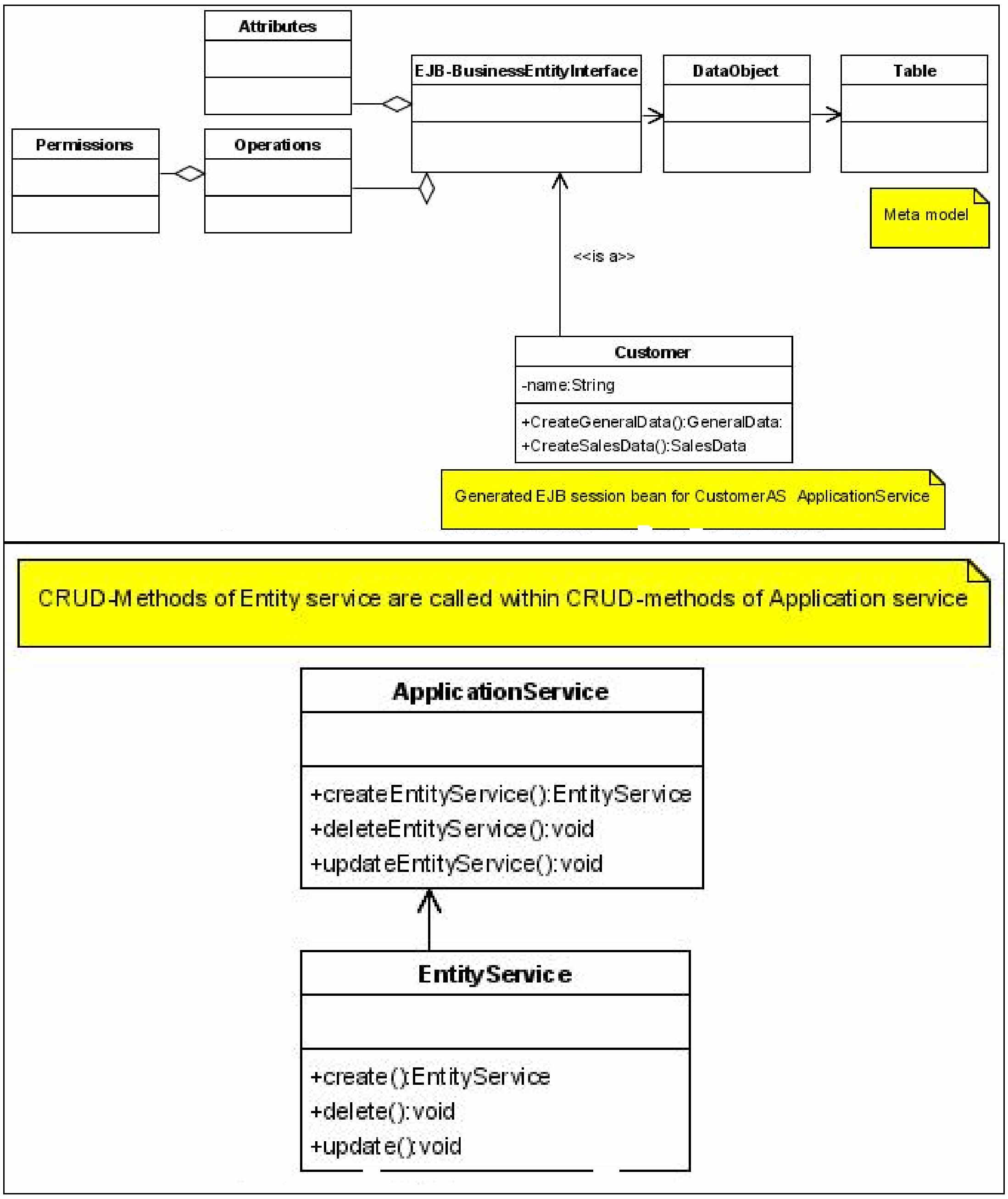

Attributes
EJB-BusinessEntityInterface
DataObject
Table
Permissions
Operations
Meta model
<<is a>>
Customer
-name:String
+CreateGeneralData():GeneralData:
+CreateSalesData():SalesData
Generated EJB session bean for CustomerAS ApplicationService
CRUD-Methods of Entity service are called within CRUD-methods of Application service
ApplicationService
+createEntityService():EntityService
+deleteEntityService():void
+updateEntityService():void
EntityService
+create():EntityService
+delete():void
+update():void

Figure 56 CAF simplified Metamodel

Presented model of business objects on figure 55 has resulted in incapability within UI patterns offered by SAP CAF framework which cannot present services which reference level is more than one, for example it cannot be showed data of entity service “Address”. The UI patterns can have embedded several component patterns (e.g. Object Editor pattern can have at maximum 8 component patterns). Component pattern works only with one service. Thus, we have this limitation of entity services dependencies.
As a result, I have redesigned the service model of customer data. I have abandoned the “customer” entity service. In the new model, “CustomerAS” application service has dependencies to “GeneralData”, “SalesData”, “OtherData”, and “CompanyCodeData” entity services. Beneath you can find the new customer data model with entity services and its attributes. The business logic for creating and searching within data was implemented in application service “CustomerAS”

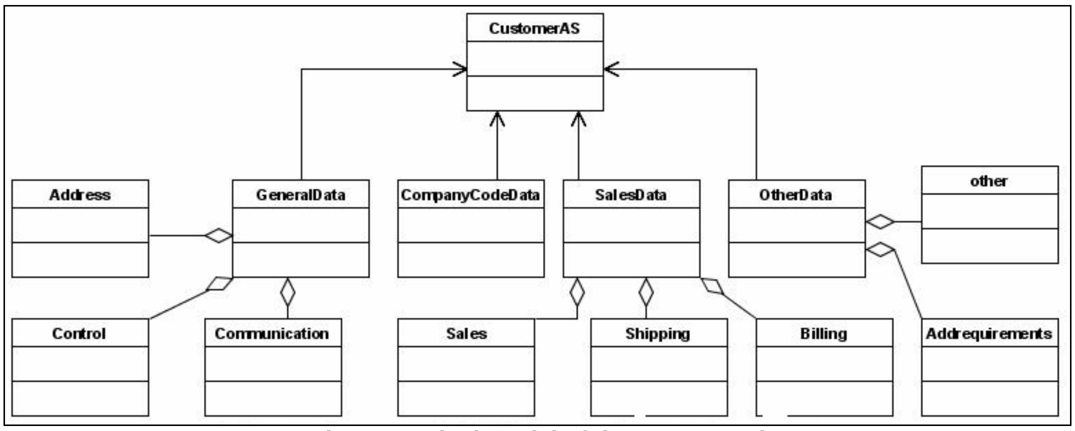


Figure 57 Final model of the customer data

4.1.2 Configuring User Interfaces

After a definition of a model of business objects, the next step in a development of the custom composite application is a definition of user interface. Here, as I implemented second scenario with CAF Repository, I did not defined mappings to backend systems. As we already know, in SAP CAF framework we can model user interfaces through an application of generic UI patterns and/or foundation Web Dynpro applications. As far as the investigation of the SAP CAF product is concerned, I have concentrated firstly only on a (re)-use of UI patterns.

Since the scope of this custom composite application is to provide functionalities for creating/modifying and searching of customer data the following UI patterns were of our interest.

Functionality of custom composite application	UI patterns
Creating/modifying of customer data	*Object Editor Pattern* – page pattern used to design, create, or edit attributes of entity services for General Customer Data, Sales Data, Company Code Data, and Other Data.
	Property Editor Pattern – component pattern used to create and edit attributes of entity services. This pattern can be embedded within Object Editor Pattern and give access to data of referenced entity patterns of an entity pattern which works with Object Editor Pattern
Searching for customer data	*Object Selector Pattern* – page pattern used to search for and list of customer data presented by entity service "customer".
	Search Bar Pattern – component pattern used to search for within customer data. This patterns enables simple text searching for specified objects(services). This pattern can be embedded within Object Selector Pattern
other	*Navigation Pattern* – enables to build a navigation menu within custom composite application i.e. navigation within configurations of Object Editor Pattern and Object Selector pattern.

In order to build our custom composite application at the beginning we have started with configurations of Object Editor and Property Editor Patterns. A presentation of Customer Master Data we have defined as four Object Editor pattern configurations for General Data, Company Code Data, Sales Data, and Other Data. Each Object Editor configuration embeds a configuration of Property Editor Pattern that enables to work with dependent entity services. Figure 58 presents UI patterns configurations.

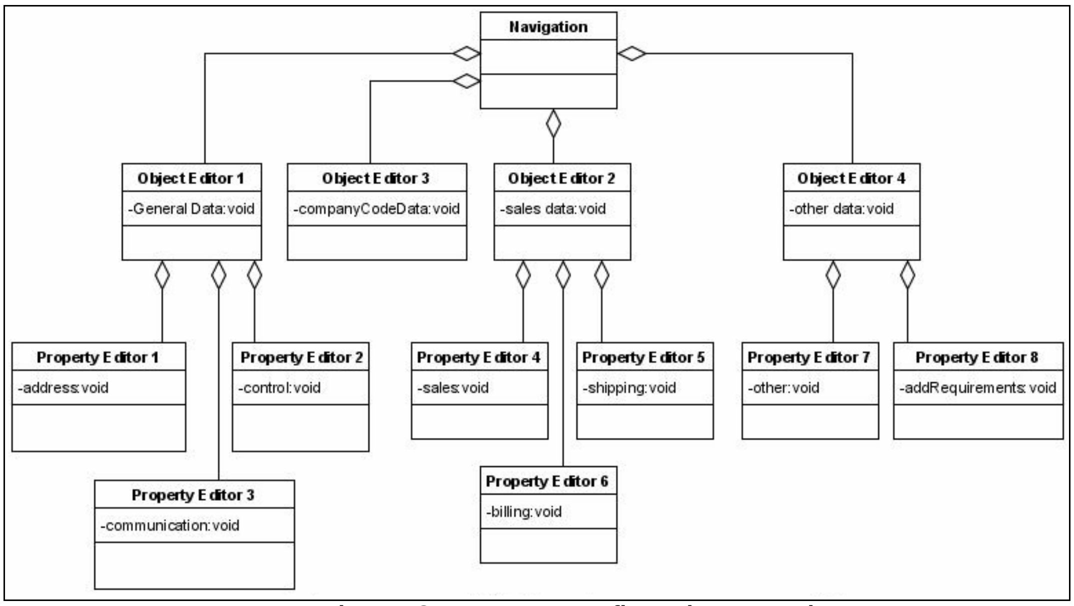
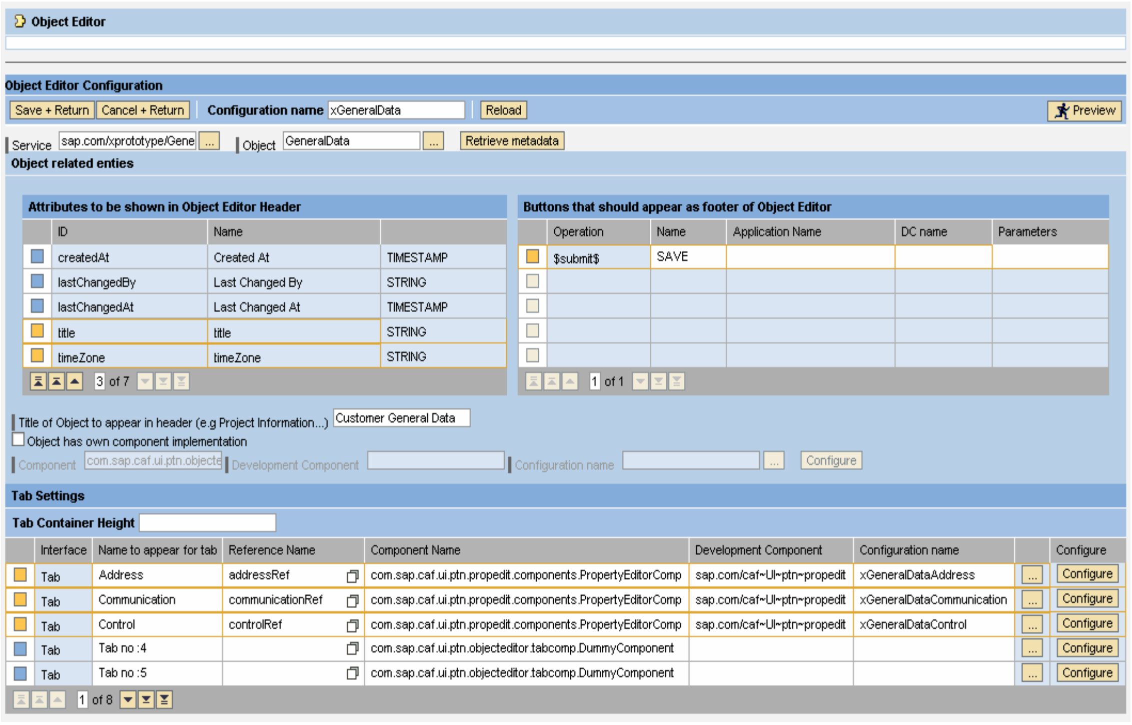


Figure 58 UI patterns configuration - overview

Below it is presented a detailed view of configuration of Object Editor Pattern for General Data and a configuration of Property Editor Pattern for "Address", "Communication", and "Control" entity service (see Figures 59 and 60). The runtime view of UI patterns configurations for entity service "general data" is presented on the figure 61.



Object Editor Configuration

Configuration name: xGeneralData

Service: sap.com/xprototype/Gene | Object: GeneralData

Object related enties

Attributes to be shown in Object Editor Header

	ID	Name	
	createdAt	Created At	TIMESTAMP
	lastChangedBy	Last Changed By	STRING
	lastChangedAt	Last Changed At	TIMESTAMP
	title	title	STRING
	timeZone	timeZone	STRING

3 of 7

Buttons that should appear as footer of Object Editor

	Operation	Name	Application Name	DC name	Parameters
	$submit$	SAVE			

1 of 1

Title of Object to appear in header (e.g Project Information...): Customer General Data

Object has own component implementation

Component: com.sap.caf.ui.ptn.objecte | Development Component: | Configuration name:

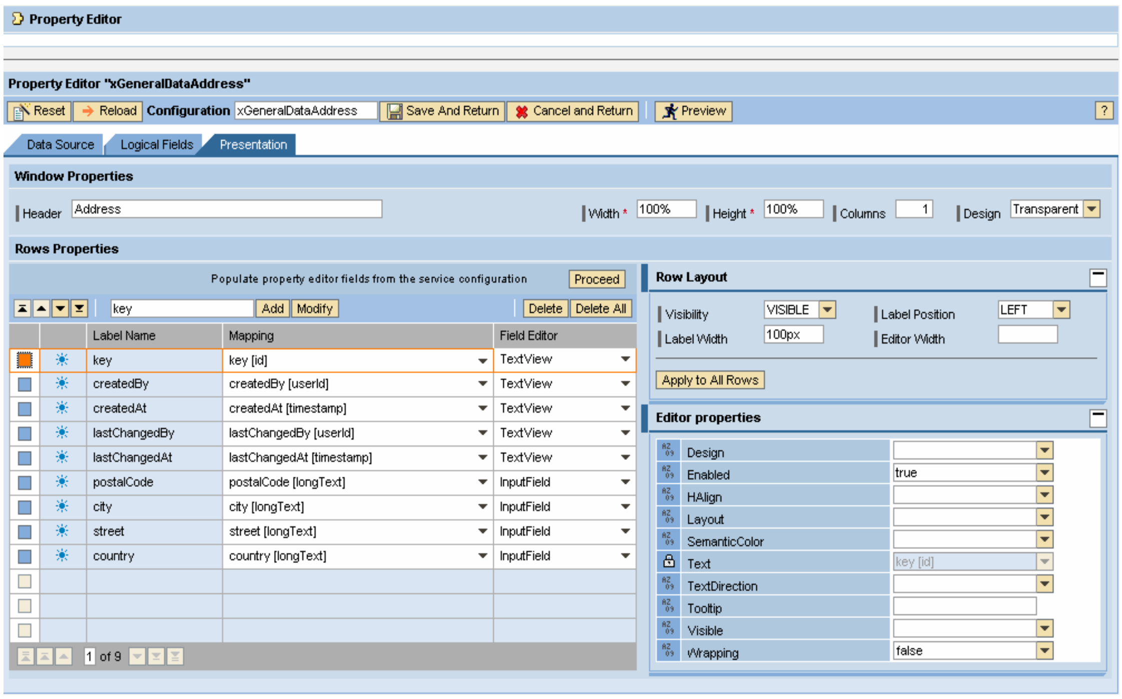
Tab Settings

Tab Container Height:

	Interface	Name to appear for tab	Reference Name	Component Name	Development Component	Configuration name		Configure
	Tab	Address	addressRef	com.sap.caf.ui.ptn.propedit.components.PropertyEditorComp	sap.com/caf~UI~ptn~propedit	xGeneralDataAddress	...	Configure
	Tab	Communication	communicationRef	com.sap.caf.ui.ptn.propedit.components.PropertyEditorComp	sap.com/caf~UI~ptn~propedit	xGeneralDataCommunication	...	Configure
	Tab	Control	controlRef	com.sap.caf.ui.ptn.propedit.components.PropertyEditorComp	sap.com/caf~UI~ptn~propedit	xGeneralDataControl	...	Configure
	Tab	Tab no :4		com.sap.caf.ui.ptn.objecteditor.tabcomp.DummyComponent			...	Configure
	Tab	Tab no :5		com.sap.caf.ui.ptn.objecteditor.tabcomp.DummyComponent			...	Configure

1 of 8

Figure 59 Object Editor Pattern – configuration



Property Editor "xGeneralDataAddress"

Configuration: xGeneralDataAddress

Window Properties

Header: Address | Width *: 100% | Height *: 100% | Columns: 1 | Design: Transparent

Rows Properties

Populate property editor fields from the service configuration

key

		Label Name	Mapping	Field Editor
		key	key [id]	TextView
		createdBy	createdBy [userId]	TextView
		createdAt	createdAt [timestamp]	TextView
		lastChangedBy	lastChangedBy [userId]	TextView
		lastChangedAt	lastChangedAt [timestamp]	TextView
		postalCode	postalCode [longText]	InputField
		city	city [longText]	InputField
		street	street [longText]	InputField
		country	country [longText]	InputField

1 of 9

Row Layout

Visibility: VISIBLE | Label Position: LEFT

Label Width: 100px | Editor Width:

Editor properties

Design	
Enabled	true
HAlign	
Layout	
SemanticColor	
Text	key [id]
TextDirection	
Tooltip	
Visible	
Wrapping	false

Figure 60 Property Editor Pattern - configuration

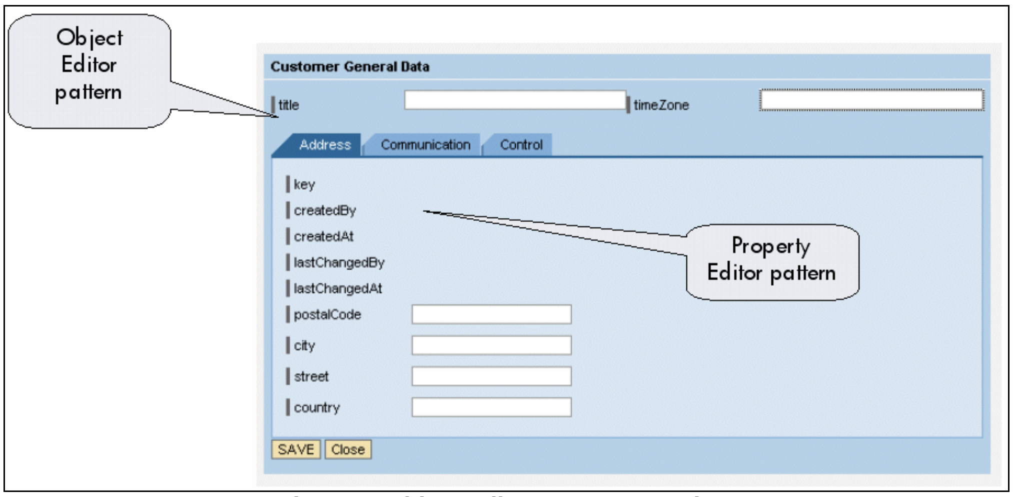


Figure61 Object Editor Pattern - runtime

All four configurations of Object Editor pattern for General Data, Company Code Data, Sales Data, and Other Data finally should be put into the Navigation Pattern. Hence, from a context menu an end user should be able to switch his/her view between all sorts of the customer data.

As far as SAP CAF SP2 is concerned configurations of UI patterns do not provide full functionalities. For instance, the combination of Object Editor and Property Editor Pattern cannot save inputted data for entity services presented through the Property Editor pattern. The Navigation Pattern is foreseen to provide its functionality from release SP3. In conclusion, SAP CAF SP2 provides immature UI patterns that forces developers to use still foundation Web Dynpro applications.

Concerning experiences while developing custom composite applications with a use of SAP CAF SP2 we encountered a number of obstacles. The summary in a table below presents general overview of SAP CAF framework.

Area	Advantages	Disadvantages
Software Installation		- SAP CAF is not a part of SAP Web AS, it needs to be installed additionally
		- At runtime SAP CAF is unstable and requires JVM memory parameter settings on the SAP Web AS site
		- SAP NetWeaver Developer Studio and plug-in for services development needs patching in order to enable services modeling
Application development - Service modeling	- Modeling process fully automated (Java EJB session beans code and web services are generated) thus application development is accelerated	- If UI patterns are used, service modeling depends on knowledge of UI patterns in advance. This obstacle can be avoid while developing foundation Web Dynpro application. Also this requires additional skills in Web Dynpro development. - web services WSDL with only 1 namespace can be integrated within CAF
Application development – UI modeling	- UI patterns provided thus application development can be accelerated	- At runtime not all UI patterns configurations provide full functionality

To begin with, the primary installation of SAP NetWeaver Developer Studio on a local computer does not provide full functionalities. While modeling entity services or application services the “Implementation” tab page could not provide view of a generated source code, we could not specify data types and exceptions handling for entity attributes. Therefore, we could not design services. Reason for that was that some CAF libraries were not provided with the primary installation. The first patches for CAF runtimes we received, forced us to reinstall the CAF runtime on the SAP J2EE engine. Unfortunately, despite the fact that the above mentioned problems were solved with this new CAF installation, we encountered further troubles while deploying our services models on the SAP J2EE engine. The CAF subprojects for metadata and Web Dynpro could not be deployed successfully on the SAP Web AS. The reason for that were inconsistencies in a package com.sap.ip.mmr.ant of SAP NetWeaver Developer Studio installation. Also we were provided with a patch for this subject. It took us approximately seven weeks to solve these above-mentioned problems and to be able to design and to deploy services within CAF environment.

After modeling services the next step was to model user interfaces within an application of UI patterns. Firstly, we started with an initiative to familiarize ourselves with CAF runtime tools. We wanted to test our modeled services with available tools and then implement user interfaces. Within these activities we experienced some obstacles as well. The test tools do not provide possibility to test defined operations (methods) of application services. You can only test entity services functionalities. Initial configuration of Object Editor and Object Selector patterns were done. But they did not work. After an investigation and knowledge exchange with SAP Developers it turned out that meanwhile SAP has change internal mappings (operations mappings between operations of application and entity services) for metadata of services. In principle, application services should have defined all CRUD methods with the same names like in entity services. This matter forced us to redesign our services model from scratch. However, after that some problems with UI patterns still remained. For instance, configuration of Object Editor Pattern with Property Editor Pattern does not work i.e. referenced entity services cannot be referenced during runtime. In case of Object Selector Pattern the functionality of “Edit” or “New” buttons that call configurations of another patterns (Object Editor Pattern) resulted in runtime errors too. According to SAP claims, mentioned UI patterns configurations problems are solved with SP3 patch level. As a result, we performed an upgrade to CAF SP3. With the new CAF installation initially you have still problems with building CAF projects within SAP NetWeaver Developer Studio (NWDS). There is a need to patch the following package com.sap.tc.ap of NWDS. What is more, with SP3 the NWDS must be upgraded to SP10. Unfortunately, exported CAF projects from NWDS SP9 could not be successfully imported to NWDS SP10; therefore services model had to be implemented manually from a scratch once again.

At present SAP Guided Procedures is not a part of the CAF environment. It is a standalone workflow tool. Moreover, SAP GP currently does not provide capability of binding UI pattern configurations within it in order to build a process. Therefore SAP Guided Procedures was out of scope in this project.

Last but not least, we have also learned that additional settings for Java Virtual Memory on the SAP J2EE engine suite need to be done in order to be able to work with CAF web-based tools which launch UI patterns configurations. In particular, a parameter for permanent space has to be increased. The permanent space is responsible for a handling of java packages and its memory allocation.

4.2 Summary

The SAP Composite Application Framework (SAP CAF) is a unified environment for a development and running of package composite applications (PCAs) such as SAP xApps based on Enterprise Services Architecture. In general, SAP CAF offers a methodology for building composite applications that will be the future business applications in the SAP landscape. The composite application framework uses SAP NetWeaver to encapsulate the functionality of the current enterprise applications as components and services. These components and services are described in metadata and implement standard frameworks for application layers, such as the user interface and process control. The combination of functionality abstraction, standard services and metadata descriptions enables the application designer to assemble an application mainly through modeling rather than programming (model-driven technology). But the real advantage that CAF brings is that the modeling technique is leveraged further through the identification and reuse of patterns for user interfaces as well as for a business processes workflow. (Figure 26)

Based on experiences acquired while developing the prototype for customer master data with the Composite Application Framework, it must be underlined that the SAP CAF as software product is in the very early stage of a development and therefore it is not yet a mature product. Firstly, CAF currently is a separate add on to the SAP Web AS. Also CAF provides a support only for J2EE Technology. In the future releases the CAF will be expanded to support ABAP stack and .NET Technology. The Process Layer of the Composite Application Framework is implemented as SAP Guided Procedures (SAP GP). Currently SAP GP is not a part of CAF architecture and therefore does not have a connection to underling CAF layers. It is a standalone workflow tool that will be integrated into CAF in the future releases. The user interfaces (UI) layer provides 10 generic patterns that can be assembled in order to build composite applications. There is also a possibility to integrate foundation Web Dynpro applications if the generic UI patterns do not offer enough functionality. In fact, at present some UI patterns configurations can be only launched. Therefore, in order to build composite applications developers are forced to implement foundation Web Dynpro applications. Furthermore, we have found out that while doing services modeling, knowledge of UI patterns in advance is required. The Service Layer turned out to be the most mature part of the SAP CAF architecture. It is integrated in SAP NetWeaver Developer Studio as Composite Application Services tools. It gives developers the possibility of reusing/importing functionalities of already existing services through external services. Also it enables developers to implement new services through specification of entity services. The business logic, which is required to implement checks and transformation of an application, is provided through application services.

5 Status and future of ESA

In this thesis, the aim was to identify the major changes concerning the organizational and implementation technologies when using an approach of the Enterprise Services Architecture (ESA). This thesis was first focused on the Packaged Composite Applications that are build out of different services with SAP Composite Application Framework. This chapter will highlight the organizational impact of Enterprise Services Architecture for a company. Furthermore, there will be discussion about a new web services paradigm – enterprise services which will be brought with the ESA. Enterprise services are Web services that provide enterprise-level business functionality[34]. Enterprise services are used to compose service-oriented applications like composite applications that reflect a business process within a company.

The Enterprise Services Architecture (ESA) is a high-level blueprint for how a customer can build a service-oriented landscape, benefiting from Web services technology to increase the value of the IT platform while dramatically reducing the total cost of ownership. However, the introduction of the ESA brought with it the introduction of various types of Web services and the ESA also blurs the lines between individual applications and between services producers and providers. In the Enterprise Services Architecture are two important kinds of services:

- *Enterprise Services* – an enterprise service corresponds to a business event, independent of any applications. The business event is described in business terms and is typically stable over a very long time. Example: receiving an order from a customer is a business event that can be described using terms such as “customer”, “order”, “product” etc.
- *Application Services* – an application service is a service offered by an application, i.e. calling it corresponds to events inside the application. The application service is described using exact software terms such as data elements and state transitions and it changes when the application changes. Example: creating a customer order in an application is an application service that uses data elements such as “customerID”, “orderID”, “productID” etc.

Moving towards the Enterprise Services Architecture creates the need for new roles and responsibilities within the IT organization, to some extent requiring new skills profiles compared to what is common in many traditionally organized IT departments. These some new roles and associated skills are presented on the figure 62. [paper-vandeLoo]

[34] [ESA-Woods]

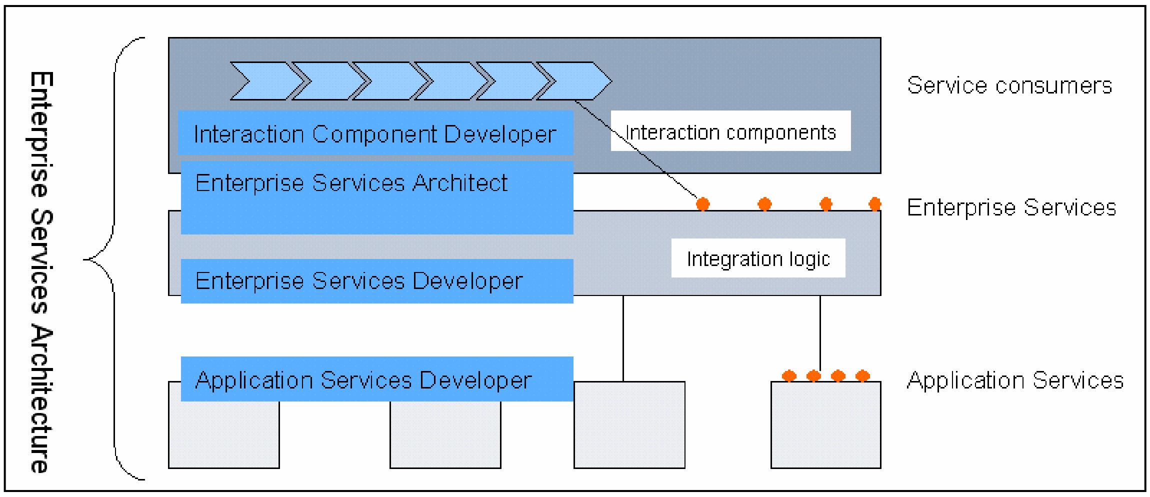

Enterprise Services Architecture
Interaction Component Developer
Interaction components
Service consumers
Enterprise Services Architect
Enterprise Services
Integration logic
Enterprise Services Developer
Application Services Developer
Application Services

Figure 62 Organizational structure for the ESA

The *enterprise services architect* is working with the business units to identify the most important business events to be supported by enterprise services. He decides on the appropriate sets of services needed to support certain business scenarios and on the granularity and interfaces of these services. He also defines the enterprise data standards that will allow integration across heterogeneous systems. The enterprise services architect must be characterized by a good understanding of business processes and the ability to extract process and service models from information given by business experts.

The *enterprise services developer* is the person who actually implements the enterprise services and creates all the integration logic necessary to map them to the underlying application services. This requires skills in using the tools of an integration platform (SAP NetWeaver) such as SAP Composite Application Framework/SAP Guided Procedures, SAP Exchange Infrastructure and understanding of the specific of the underlying applications and the technical possibilities and limitations for integrating to them.

In many cases the underlying applications will not provide the services needed by the enterprise services developer. This is a task of an *application services developer* to develop the missing services.

A development of new user interaction components meaning everything from fairly simple UI components to complete composite applications is conducted by the *interaction component developer*. The skills needed are focused on selection and use of the appropriate UI technologies, which could range from web page development tools and highly graphics tools such as Macromedia Flash to professional UI development tools like SAP Web Dynpro or Microsoft Office technologies. Ideally, the interaction component developer is working very closely with one or more professional user interaction designers.

Organizationally, *the enterprise services architect* and *developer* should be part of a central enterprise architecture team reporting to business departments. They are owners of the integration knowledge and primarily responsible for the alignment of IT with the needs of the business. The *application services developer* may be part of the organization that owns the respective application. The *interaction component developers* have to be very close to the end users, so it is often advisable to place them in local IT departments. In that case, they need architectural guidance from the central architecture team. The SAP Composite

Application Framework as a unified environment for development composite applications will provide suitable tools for definition of application services and enterprise services.

Another important topic to be investigated for the purposes of this thesis was to position and find out about technical standards for enterprise services. The investigation focused on aspects as technical requirements, designing and standardization for enterprise services. These all aspects are very important for IT departments, which want to start implementing ESA, since enterprise services are the core of the Enterprise Services Architecture.

As already given in this thesis, the Enterprise Services Architecture (ESA) is a high-level blueprint for how a customer can build a service oriented landscape, benefiting from Web services technology to increase the value of the IT platform while dramatically reducing the total cost of ownership. The introduction of the ESA brought with it the introduction of various types of Web services, namely application and enterprise services.

Web services are an ideal option for automating complex business processes, since they are not bound to a specific platform or programming language, and – more importantly – they provide the potential for worldwide collaboration across systems and companies based on the Internet. In order to handle enterprise-level, business-scale processes Web services also need to be as reliable, maintainable, and secure as current business software. Also to keep maintenance costs down, web services must achieve the right level of scalability and reuse. As a result, there are already discussions about “enterprise services” that do have all the features mentioned-above.

Enterprise Services are web services that will provide enterprise-level business functionality. They can be simple lookup services (like finding a company’s location or product offerings) or complex and composite services. What do they have in common is that they are integrated into process or application. Typically enterprise services are high-level components that take more granular Web services and aggregate them into reusable elements with business value. [ESA-Woods]

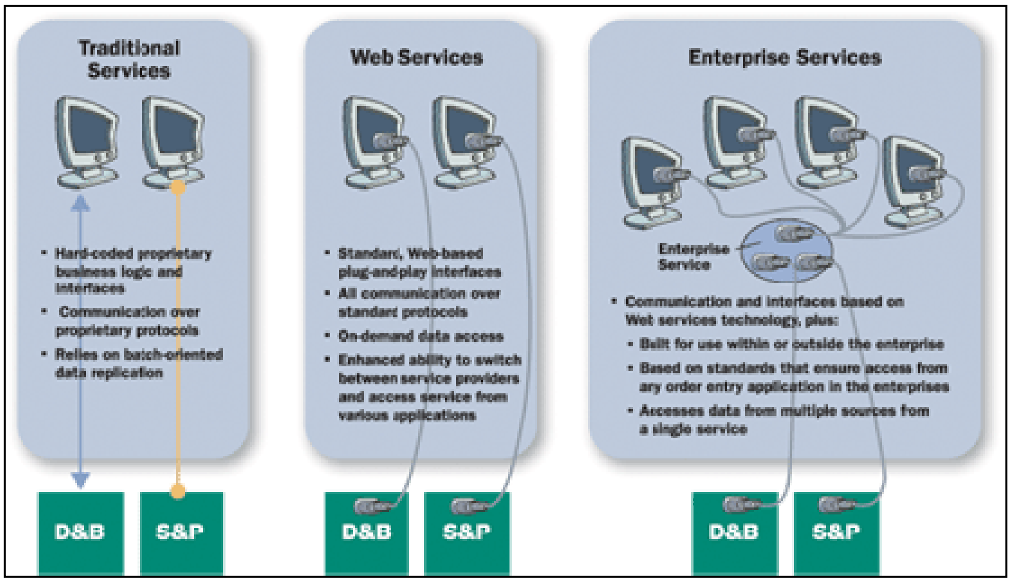

Traditional Services
Hard-coded proprietary business logic and interfaces
Communication over proprietary protocols
Relies on batch-oriented data replication
Web Services
Standard, Web-based plug-and-play interfaces
All communication over standard protocols
On-demand data access
Enhanced ability to switch between service providers and access service from various applications
Enterprise Services
Enterprise Service
Communication and interfaces based on Web services technology, plus:
Built for use within or outside the enterprise
Based on standards that ensure access from any order entry application in the enterprises
Accesses data from multiple sources from a single service
D&B
S&P

Figure 63 From traditional services to Enterprise Services[35]

As far as technical requirements for Enterprise Services are concerned, they must be as robust and reliable as business applications that rely on their own set of exclusive and local data sources. Technically, Enterprise Services must be built on established and emerging Web service standards for the big benefit of openness and interoperability. They must cope with requirements in terms of scalability to support thousands or even millions of calls and in terms of security to provide authentication as well as end-to-end message integrity and data confidentiality which are features generally not granted on the Internet. Furthermore, when any problems arise, an infrastructure must be in place to analyze and debug what will almost certainly be a multitude of interconnected components. This is where Enterprise Services Architecture and Enterprise Services Repository comes in. [paper-Fritz]

While designing Enterprise Services, an important thing to remember about the Enterprise Services Architecture is that it attempts to build a platform out of a number of independently designed applications. Hence, it cannot be assumed that all applications are applications of a particular software vendor or even that the applications have been designed to work together. This fact has a significant impact how the services have to be defined and gives some criteria to be considered. Therefore, Enterprise Services should cover a complete business event and only use business terms in its interface so that it can be called from any application. Moreover, when designing Application Services, an application service should perform a specific business function, i.e. avoid for example a generic “change” interface, and use only such data elements that can be mapped to business terms, i.e. avoid data elements that are specific to the particular application. While designing Enterprise Services, it is very important to understand that enterprise and application services are not mutually exclusive. Application service can be very well enterprise service. This is a case as soon as the application service covers all aspects of a business event that are relevant for a particular company. This may often be the case for comprehensive services offered by large, integrated applications such as mySAP CRM. However, whether an application service is an enterprise service or not depends on the system landscape within each enterprise. An application vendor cannot “build” enterprise

[35] [paper-Fritz]

services; it can only strive to build application services that will exactly match customer's enterprise services. If the underlying applications offer a wide array of application services well suited to be promoted to enterprise services, the definition of enterprise services is a fairly straightforward task. When enterprise services cannot be created by simply promoting application services, they will have to be composed out of several application services, provided by one or more applications. As a result, it is imperative for the success of the Enterprise Services Architecture that the composition of enterprise services on one hand is easy to perform and maintain, and on the other hand results in adequate performance, stability, reliability, security etc. The major challenge is to define and provide web service standards as well as tools for composing enterprise services.

As already presented in this thesis the Composite Application Framework is a unified environment for designing and running composite applications. Composite applications are applications that consume services like enterprise services and orchestrate them to reflect new business processes within a company. The Composite Application Framework offers a really important user benefit through the fact that the interface layer of composite applications is constructed from process-based perspective. The process-based perspective shows the flow of work through the steps of a process as the work proceeds from task to task and user to user. As figure 64 presents, with process-based modeling, the process is at the top of the tree. The main container for the process in the SAP CAF is a business scenario, which has procedures, actions, and services as its components. The idea is that some applications should be described with a set of steps, each with components, with several levels of containment. A business scenario could represent a process for product lunch or project to be managed. Then, the specific processes are the next level, with the lowest level representing steps in a process.

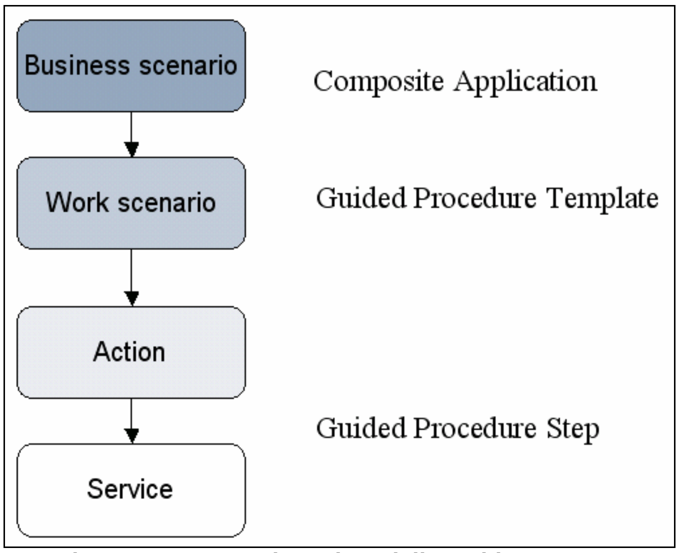


Figure 64 Process-based modeling with SAP CAF

The SAP Composite Application Framework provides SAP Guided Procedures framework for defining processes. A guided procedure includes a collection of steps called phases. Each phase might have several actions, steps such as filling out an evaluation form or performing a specific unit of work. Behind each action is an enterprise service that performs some sort of a function. An envisioned action model is presented underneath.

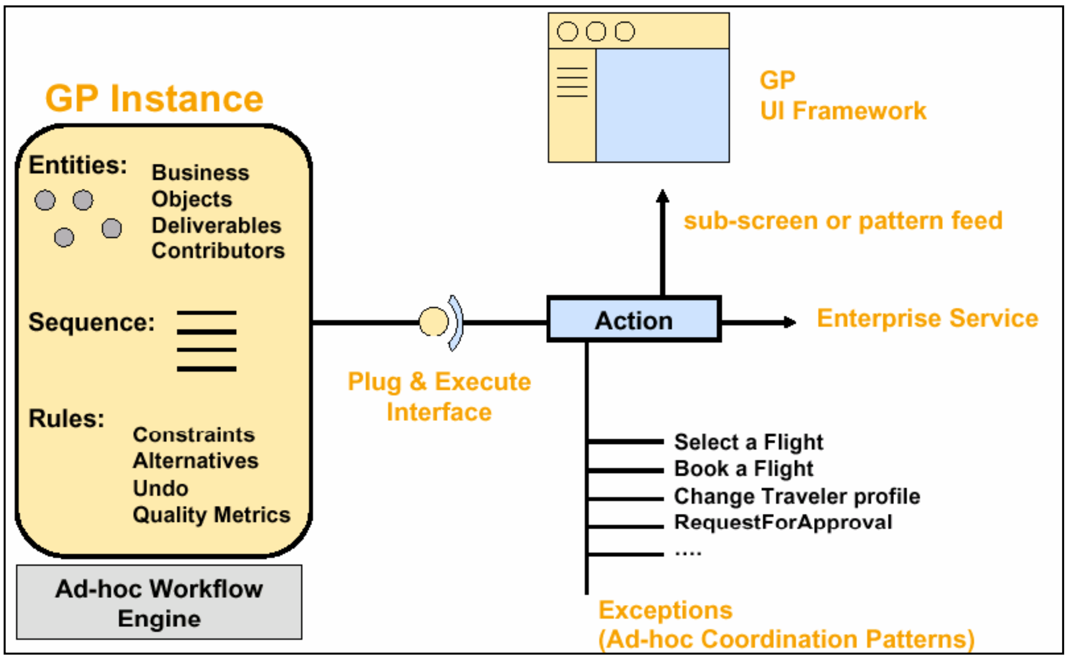


Figure 65 Envisioned Action Model[36]

At present there is no a technical description standard for enterprise service. Instead the SAP Composite Application Framework and guided procedures propose currently a composite scenarios technique and interactive forms.
Composite scenarios orchestrate the behavior of many different applications that have not been made available as services. In this technique, a guided procedure moves a user from one user-interface screen to another. The user has impression of a single workflow, even though the screens are provided by different applications. Sometimes a process can be effectively automated as a path through several different applications without modifying them. This approach has its limits. The screens do not really know that they are participating in the composite scenario, data may be inconsistent across each screen, and implementing a composite scenario in an environment with many different vendors is difficult. The biggest limitation may be the assumption that the business logic of each application has to be accepted as is. With composite scenario, you have no chance to modify the applications.

Interactive forms are another way that guided procedures communicate with the outside world. Interactive forms are standalone documents build to collect information offline from a user. The form is then submitted to a guided procedure, which processes the information. They transfer the data back and forth from the guided procedure by using XML. Interactive forms solve an important piece of the automation puzzle because they provide a mechanism for collecting data that can be used anywhere, on many devices such as phones, PDAs regardless of connectivity.

[36] Training materials from Workshop “SAP Composite Application Framework / SAP NetWeaver” at SAP AG in Walldorf

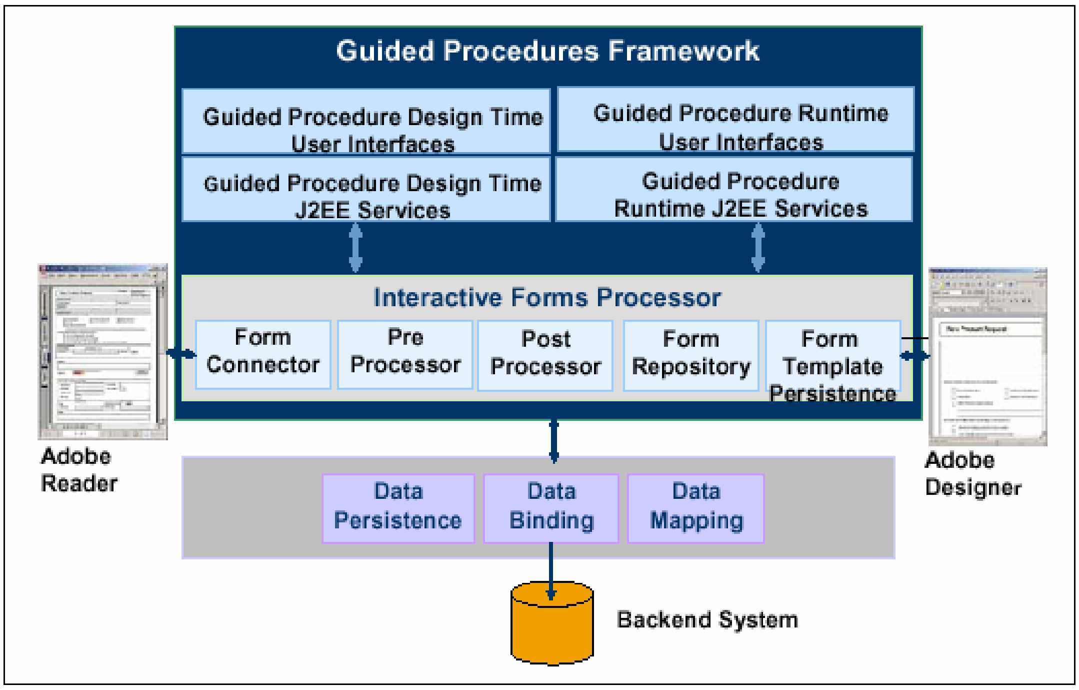

Guided Procedures Framework
Guided Procedure Design Time User Interfaces
Guided Procedure Runtime User Interfaces
Guided Procedure Design Time J2EE Services
Guided Procedure Runtime J2EE Services
Interactive Forms Processor
Form Connector
Pre Processor
Post Processor
Form Repository
Form Template Persistence
Adobe Reader
Adobe Designer
Data Persistence
Data Binding
Data Mapping
Backend System

Figure 66 Architecture of SAP GP with active forms[37].

[37] SAP marketplace source

6 Conclusion

This thesis investigates the major changes concerning organizational and modeling methodologies when using the Enterprise Services Architecture approach that aims to enable a business innovation for a company while lowering total cost of application ownership. Through a development of the custom composite application prototype, this thesis also makes a contribution to the subject of proving business benefits while using the new approach of a web service interfacing with composite applications. Finally, this thesis investigates the new web services paradigm called enterprise services that provide enterprise-level business functionality for a company. And as a result, gives a comprehensible answer of this subject for IT departments that want to use Enterprise Services Architecture approach in the future.

The goal of the Enterprise Services Architecture is to break a silo of current enterprise applications into services so that user interfaces are no longer linked to the silo and composite applications can be assembled from these services to bring functionality to new groups of users and to extend automation further into the company. Thus, the functionality of current enterprise applications can be more effectively reused. The most persuasive arguments for the Enterprise Services Architecture are specific to a particular business. The Enterprise Services Architecture is most compelling when the executive team has a shared vision of a business strategy that will crush the competition and make customers joyfully send in more orders. The core assumptions of the ESA are that:

- Conscious design of a comprehensive architecture is based on enterprise's current needs and predictions about future business conditions
- Current systems are used as a foundation
- Flexibility is created where we need it using loosely coupled components and services as the fundamental building blocks
- The resulting architecture will decrease the cost of change, which will in turn expand tactical and strategic possibilities that should enable business advantage.[38]

This thesis clearly states that Enterprise Services Architecture is an example of Service-Oriented Architecture. The Enterprise Services Architecture (ESA) as a high-level blueprint for how a customer can build a service-oriented landscape benefiting from Web services technology introduces various types of Web services like Enterprise services (description of business functionalities). As a result, moving towards the Enterprise Services Architecture creates the need for new roles and responsibilities within the IT organization, to some extent requiring new skills profiles compared to what is common in many traditionally organized IT departments. These some new roles and associated skills are presented on the figure 62. [paper-vandeLoo]

As for companies organizationally ESA approach imposes that *the enterprise services architect* and *developer* should be part of a central enterprise architecture team reporting to business departments. They are owners of the integration knowledge and primarily responsible for the alignment of IT with the needs of the business. The *application services developer* may be part of the organization that owns the respective application. The *interaction component developers* have to be

[38] [ESA-Woods]

very close to the end users, so it is often advisable to place them in local IT departments. In that case, they need architectural guidance from the central architecture team.

Another investigated subject for the ESA is the modeling approach for future business applications that consume services and build user web interfaces that reflect a particular business process within a company. Such new applications are called composite applications. Composite applications as a new breed of applications endeavor to; serve business processes that cross multiple functions, target multiple users even across inter-enterprise boundaries, and integrate functions that are currently supported by independent generic applications. Composite applications are built on top of the company's heterogeneous technology landscape, thus enabling cross-functional business processes and securing existing software investments. By integrating all of a company's applications and systems, cross applications help to maximize the value of the existing IT investments. Composite applications are content-driven also they enhance business processes by relating knowledge and structured information. As a result, business can be run with corporate-wide business intelligence instead of disaggregated information.

The SAP Composite Application Framework (SAP CAF) offers a methodology for building composite applications which will be the future business applications in the SAP landscape. The Composite Application Framework uses SAP NetWeaver to encapsulate the functionality of the current enterprise applications as components and services. These components and services are described in metadata and implement standard frameworks for application layers, such as the user interface and process control. The combination of functionality abstraction, standard services and metadata descriptions enables the application designer to assemble an application mainly through modeling rather than programming (model-driven technology). But the real advantage that CAF brings is that the modeling technique is leveraged further through the identification and reuse of patterns for user interfaces as well as for business processes workflows. The SAP CAF as a solution proposes an approach to build composite applications through assembling three layers: service layer, user interfaces UI layer, and process layer. The service layer is an abstraction layer where data models of business objects are build or imported from existing systems, and exported to the outside world as services. Such services are consumed by UI layer that represents its data to end users through user interfaces. User interfaces are built based on the Model-View-Controller concept from generic predefined UI patterns and foundation Web Dynpro applications. Such defined user interfaces are used in the process layer to define a collaboration workflow in order to reflect a specific business process within a company.

Through a development of the prototype of a custom composite application for customer master data it must be underlined that the Composite Application Framework as a software product is in the very early stage of a development and therefore it is not yet a mature product. The Service layer turned out to be the most mature part of the SAP CAF architecture. It enables a software developer to reuse applications functionalities by importing existing services. Also gives the possibility to extend composite application by new services. The UI layer provides currently about 10 generic UI patterns and also a possibility to integrate foundation Web Dynpro applications if the generic UI patterns do not offer enough functionality. At present some UI patterns configurations can be only launched. Therefore, in order to build composite applications with complex user interfaces, developers are forced to implement foundation Web Dynpro applications.

However, in a scope of the prototype we did not work with the Process Layer implemented as SAP Guided Procedures (SAP GP), our investigation determined that currently SAP GP is a standalone workflow tool and do not provide possibility to create Enterprise services. Moreover, the investigation points out that there are no technical description standards for Enterprise services. Instead at present the SAP Composite Application Framework and guided procedures propose a composite scenarios technique and interactive forms. Composite scenarios orchestrate the behavior of many different applications that have not been made available as services. In this technique, a guided procedure moves a user from one user-interface screen to another. The user has impression of a single workflow, even though the screens are provided by different applications. The limitation of this approach is that screens do not really know that they are participating in the composite scenario, data may be inconsistent across each screen, and implementing a composite scenario in an environment with many different vendors is difficult. Another way that guided procedures use to communicate with outside world are interactive forms. Interactive forms are standalone documents build to collect information offline from a user. The form is then submitted to a guided procedure, which processes the information. They transfer the data back and forth from the guided procedure by using XML.

7 Glossary

ABAP	*Advanced Business and Application Programming* is a programming language for developing SAP applications.
Application Service	Application Services contain the business logic of the application and provide an interface to other services e.g. interface to end users through user interfaces.
BAPI	A *Business API* is an interface to one of SAP's R/3 applications. It enables third-party developers to write enhancements that interact with the R/3 modules. Technically a BAPI is a RFC-enabled Function Module (RFM) that follows the rules defined in the SAP BAPI Programming Guide and is defined as a method in the SAP Business Object Repository.
BI	mySAP *Business Intelligence* enables a complete view of all business operations and information. It provides the tools to use information to make the right decisions, set strategy, and measure the results of business tactics.
BMP	If *Bean-Managed-Persistence* is used methods to find, create and delete EJB objects have to be implemented by the developer and are not provided by the EJB container.
CAF	Composite Application Framework is a unified environment for a development and running package composite applications PCAs such as SAP xApps based on Enterprise Services Architecture. The CAF uses SAP NetWeaver to encapsulate the functionality of the current enterprise applications as components and services. These components and services are described in metadata and implement standard frameworks for application layers, such as the user interface and process control.
CMP	If *Container-Managed-Persistence* is used the EJB container provides persistence mechanisms to store entity beans in a database.
EJB	*Enterprise JavaBeans* can be described as a component software architecture from Sun that is used to build Java applications that run in the server. It uses a container layer that provides common functions such as security and transaction support and delivers a consistent interface to the applications regardless of the type of server. CORBA is the infrastructure for EJBs, and at the wire level, EJBs look like CORBA components. EJBs are the backbone of Sun's J2EE platform, which provides a pure Java environment for developing and running Web-based applications.
Entity Service	Entity Services contain the life-cycle methods and table definitions for defining business objects. Entity Services expose business objects data to application services.
ESA	Enterprise Services Architecture (ESA) outlines a disciplined and structured approach to understanding how today's enterprise applications will make use of web services. ESA is an application of service-oriented architecture applied to the current heterogeneous world of IT architecture.
External Service	External services only contain metadata information and provide access to external services such as web services and remote function call (RFC).
J2EE	The *Java 2 Platform, Enterprise Edition* is a platform from Sun for building Web-based enterprise applications. J2EE services are performed in the middle tier between the user's browser and the enterprise's databases and legacy information systems. J2EE comprises a specification, reference implementation and set of testing suites. Its core components are EJBs, JSPs and Java Servlets as well as a variety of interfaces for linking to the information resources in the enterprise.

Master Data	Data that remains unchanged for an extended period. It contains information that is frequently required for use in the same way.
Metadata	Data describing other data. Metadata are data definitions normally stored in a data dictionary.
MVC	*Model-View-Controller* is a design pattern to efficiently relate a user interface to underlying data models in object-oriented programming languages. The *Model* represents the underlying, logical structure of data in a software application and the high-level class associated with it. A *View* is a collection of classes representing the elements in the user interface A *Controller* represents the classes connecting the model and the view, and is used to communicate between classes in the model and view.
mySAP	The *mySAP.com* e-business integration platform is a family of software and services that empowers customers, partners, and employees to collaborate successfully. It delivers content to the user based on their role in different business areas like customer relationship management, supply chain management, e-procurement, business intelligence, product lifecycle management, human resources, or finance.
PCA	Packaged Composite Applications (PCAs) sit on top of an Enterprise Services Architecture layer, a software product that creates components out of existing enterprise applications. PCAs represent a new architectural paradigm for enterprise applications. Using web services, they combine new functionality with services from existing applications to enable flexible cross-functional automation.
RFC	A *transactional Remote Function Call* (tRFC) calls a function module in R/3 indirectly using a transactional interface in R/3. If an error occurs, the RFC client program has to reconnect to R/3 later and repeat the call with a specific transaction ID. A queued RFC (qRFC) guarantees that data is transferred immediately. In case of an error the changes aren't lost but put in a queue until the data is transferred correctly.
SAP Service Marketplace	Worldwide information and communication network from SAP AG.
SAP xApps	Collaborative Cross Applications (xApps) are SAP's fulfillment of the PCA vision. xApps are products focused on solving specific business problems that build on existing systems to provide new functionality, automating cross-functional processes.
SOAP	The *Simple Object Access Protocol* is a lightweight message-based protocol using HTTP for accessing services on the Web and for the exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined data types and a convention for representing remote procedure calls and responses.
UDDI	*Universal Description, Discovery and Integration* is a XML-based specification using the SOAP protocol. It provides a registry designed to enable software to automatically discover services on the Web and to integrate with them by providing the necessary translations.
URL	The *Uniform Resource Locator* defines the route to a file on the Web or any other Internet facility. It contains the protocol prefix, port number, domain name, subdirectory names and file name.
WAS	The SAP *Web Application Server* is the e-business platform for mySAP.com solutions. It provides Web services through platform-independent, easy-to-maintain business Web applications and technologies, including key support for J2EE and ABAP.
Web Service	A *Web Service* is a self-contained, modularized functionality, which can be published, discovered, and accessed across a network using open standards.

WSDL	The *Web Services Description Language* is a language to describe the capabilities, the protocols and formats used by a Web service. WSDL descriptions can be housed in a UDDI directory and they are always independent from the used protocol (SOAP, XML) and coding (MIME).
XML	The e*Xtensible Markup Language* is an open standard for describing data from the W3C. It is used for defining data elements on a Web page and business-to-business documents. It uses a similar tag structure as HTML; however, whereas HTML defines how elements are displayed, XML defines what those elements contain. HTML uses predefined tags, but XML allows tags to be defined by the developer of the page. Thus, virtually any data items, such as product, sales rep and amount due, can be identified, allowing Web pages to function like database records. By providing a common method for identifying data, XML supports business-to-business transactions and is expected to become the dominant format for electronic data interchange.

8 Bibliography

Introduction Literature:

[xApps-Herger] "SAP xApps und das Composite Application Framework"	Jo Weilbach, Mario Herger	Galileo Press, 2005
[NWfD-Woods] "SAP NetWeaver for Dummies"	Dan Woods, Jeffrey Word	Wiley Publishing, 2004
[ESA-Woods] "Enterprise Services Architecture"	Dan Woods	O'Reilly, Sep. 2003
[PCA-Woods] "Packaged Composite Applications"	Dan Woods	O'Reilly, Jun. 2003
[WPWAS-Rau] "Web Programming with the SAP Web Application Server"	Frederic Heinemann, Christian Rau	SAP PRESS, 2003
Service-oriented computing: Introduction	M. P. Papazoglou, D. Georgakopoulos	Communications of the ACM, Volume 46, Issue 10 ;2003,
Adoption Challenges in Migrating to Web Services	*Scott Tilley, John Gerdes, Terrance Hamilton,*	IEEE Proceedings Fourth International Workshop on Web Site Evolution (WSE'02); 2002,
[WebServices-Alonso] "Web Services, concepts, architectures and applications"	Gustavo Alsonso	Springer, 2004
[mySAPT-Farber] "mySAP Technology"	Günter Farber, Julia Kirchner	Galileo Press, 2002

Papers:

[paper-vandeLoo] "Organizational impact of Enterprise Services Architecture" by Kaj van de Loo December 21, 2004
http://www.sdn.sap.com
[paper-Varhol] "*SOA: Get it Right the First Time*" by Peter Varhol May 24, 2004
http://www.ftponline.com/weblogicpro/2004%5F05/magazine/features/pvarhol/

[paper-Mattern] "Build a Services-Based Infrastructure that enables Business Change while containing costs" by Thomas Mattern, SAP AG; October, 2004
http://www.sapinsideronline.com/spijsp/article.jsp?article_id=40097&volume_id=5427

[paper-Fritz] "When Does a Web Service Become an Enterprise Service?" by Dr. Franz-Josef Fritz, SAP AG; April 2004
http://www.sapinsideronline.com/searchspi/search.htm?page=article&query_text=base&key=37906

[paper-SAP-NW] "*SAP NetWeaver*^{TM}" by SAP AG
http://www.sap.com/solutions/netweaver/pdf/BWP_NetWeaver_Overview.pdf

[paper-Eisenberg] "Service-Oriented Architecture: The Future is Now" by Robert Eisenberg, REassociates.Net;
http://www.intelligenteai.com/print_article.jhtml?articleID=18900111

SAP documentation:

[sap-caf-ug] SAP Composite Application Framework User Guide Release 1.0
[sap-caf-gp] Guided Procedures (CAF – GP) Release 1.0

Online documentation:

[sap-doc-nw] http://service.sap.com/netweaver
[sap-doc-xapps] http://service.sap.com/xapps
[sap-doc] http://help.sap.com
[grid-doc] http://gridcafe.web.cern.ch/gridcafe/