Technical White Paper

On BRM Framework

October 15, 2003
By Tony Hansen

The purpose of this paper is to give an overview of Presys' enterprise-class framework for business relationship management (BRM) and its role in the Web service stack and the Service Oriented Architecture(SOA). In order to do this the Web service stack and its layers and components are described detailed. How On BRM and its components fit into the Web services stack's layers perfectly is also described. The architecture of On BRM, including its many layers, is explained in technical detail and it is described how On BRM uses best practises and reputable patterns with regards to designing, coding, and integrating enterprise applications.
Owing to these facts an organization can achieve a structured integration of internal and external resources using On BRM and the Web service stack. This well designed integration is compared to the traditional IT architecture.
Therefore the consequences of using SOA are shown and a distinction is made between systems built on business processes, such as On BRM, and traditional workflow systems using proprietary workflow process execution models and APIs. Using On BRM it is not necessary to reengineer the organization; it is only necessary to adaptively tailor On BRM to the organization's business processes to respond quickly to changing business conditions.
The paper also explains why On BRM's architecture makes way for providing seamless support to both .Net and J2EE, excellent scalability, and state-of-the art security.
Lots of patterns and anti-patterns exist that do not provide optimal solutions and these are also described to show the validity of pattern choices in On BRM.
The paper describes the job profiles and competencies required to fully utilize On BRM's potential.
Owing to all these circumstances the On BRM Framework will significantly reduce time to market for new business opportunities, lever IT assets by reusing existing applications, reduce the expenses to payloads from external resources such as Web services and applications and minimize cost to change management and converting of data.

Figure list

Executive Summary

On BRM (Business Relationship Management) is a framework for enterprise applications built to:

- Lever existing IT assets
- Enable responsive and efficient collaboration between external and internal business processes despite an uneven technological development
- Cover the entire business process life cycle from end to end in a global world, but on a local basis
- Respond adaptively and rapidly to business changes
- Offer responsiveness with real-time visibility and access 24x7
- Provide scalability
- Provide state-of-the-art security.

These points are the seven main imperatives for the choice of IT architecture and design of On BRM.

Large corporations have made significant investments in three main areas of assets: business process modelling, workforce management and training, and IT system development and deployment. Often IT projects are budgeted as one-time expenses but the costs of software maintenance such as extending, integrating, and managing applications are significant. Over time these costs even tend to increase to a bigger percentage of the IT budget than new application development. When maintenance costs exceeds new software investments, productivity growth decreases and an organization's competitive power also decreases. But such investments can also yield unprecedented returns when combined and utilized effectively and therefore On BRM is designed to support the execution of best-in-class business process models in collaboration with existing IT systems. In order to fulfil the imperatives On BRM offers:

- Interface-centric architecture based on SOA and the Web service stack
- Process- and task-oriented approaches based on project management components
- Several hundred best-in-class designed business components, utilizing more than 175 database tables, containing all the important business services required for business relationship management in a modern organization, whether private, public or governmental
- A design of the components that is focused on re-use of existing IT and business assets
- Adaptive scalability across different platforms.

On BRM is a multi-tier J2EE framework that runs on an application server. On BRM is built upon the emerging standard Web service stack at the business process layer and uses best practices for business processes at the top layers. Therefore it is designed for distributed computing, where both internal and external existing data sources such as applications and databases are seamlessly integrated with each other using On BRM's components. On BRM breaks the opaqueness and isolation of existing applications by specifying constraints on how the operations of a collection of existing applications and

their joint behaviour can be used. This turns out to be similar to specify business processes and business relationships. On BRM can also be used as a traditional application running on an application server without using the components of the Web service stack. On BRM is .Net compatible.

On BRM's extreme scalability ensures that an organization's deployment configuration can be adapted to its business needs easily. Simple business process needs may result in the use of only a single application server, while more demanding needs may give rise to a large cluster of application servers that collaborate for

- Increased reliability
- Seamless integration of external and internal resources demanded by customers, partners and employees
- Cost-effective use of external resources such as Web services.

Ultimately On BRM may become the foundation on which multiple departments and external business-to-business connections collaborate throughout the world. All locations have the same opportunities, working as one centralized application, such as access to shared company data resources while providing specific, location-based functionality and interfaces, like local languages, currencies and taxes. Also business partners such as suppliers, who might all use different systems, can be connected to the application as local external resources with real-time visibility, access and personalization.
This deployment configuration enables support of high-volume transactions on a global scale.

On BRM is multi-layered and can be seen as a general three-layered architecture that supports the separation of presentation, domain logic and data connectivity:

- The Presentation layer is responsible for implementing the business process logic managing user interfaces, including thin and rich clients, browsers, corporate portals, and wireless/pervasive devices
- The Business layer contains the domain logic and business components that can be modelled to sustain specific business processes
- The Data source layer provides connectivity with internal and external resources.

This three layer architecture supports agile, stepwise and separated development far away from the 'big bang´ approach used by some ERP and CRM vendors as each layer is well defined and requires different set of skills:

- Interface developers, who deal with presentation, coding with tools like VB, C#, Delphi, Java, JSP, ASP, or Flash
- Business analysts, who analyze and design the business processes that define transactions, services, and operations. Developers who code the business processes and have knowledge about the components of On BRM and best practice design and code patterns
- Architects, developers, and other IT staff who build, install, maintain, and customize the application based upon On BRM. They have knowledge about their own IT infrastructure, infrastructure patterns, and On BRM's architecture.

BRM (Business Relationship Management) is a broader concept than the traditional CRM (Customer Relationship Management). The On BRM Framework offers several hundred enterprise-class business components that can be used as process templates to support all business processes and business connections, such as citizens, employees, partners, suppliers, clients and customers. The framework is not database-centric but interface-centric, so the organization will not be locked into its own departmental or organizational data, but get two-way connections to all internal and external resources such as:

- Traditional legacy applications with proprietary protocols and APIs
- Resources such as databases and applications as soon as they are published as a Web service
- J2CA enabled applications
- Applications supporting transaction managers such as CICS and Tuxedo
- Applications supporting MOM (Message Oriented Middleware) such as MQSeries
- J2EE based adapters. On BRM is therefore seamlessly integrated among others with SAP, Siebel, Documentum, and Lotus Domino
- Web services from products such as MS Office, and Lotus Domino among others
- Security technologies such as digital signature.

On BRM offers means to efficiently and rapidly develop highly responsive solutions built upon first class end-to-end business processes based on internal and external resources providing real-time visibility and access of information for employees, customers, and partners. An organization's business processes can even be a part of other organizations' business processes as a data source because On BRM supports Web services, J2CA, transaction managers, MOM, and J2EE based adapters - hence On BRM enables business agility and collaboration.

All business components including tailored components can be used as business process templates. These business process templates are interfaced through WSDL or Business Delegates. Thereby On BRM can be integrated and combined with both thin and rich clients such as MS Office or internally developed applications. New client applications can be developed with well known products such as MS Office.Net, VB.Net, C#, JSP, BPEL, Flash, Delphi and data modelling tools to give an organization's business processes the user interface that is suitable for a specific scenario. Therefore based on the organization's own and its partner's business processes, its developers can quickly, with well-known tools and with only small training expenses develop new client applications.

As On BRM supports user interfaces such as Windows, portals including corporate and trading portals, and HTML, an organization will reduce the training and change management costs.

The On BRM enterprise-class components support a multitude of end-to-end business processes in a two-way dialog, including sales and marketing, customer support, and distribution by handling:

- CRM

- PRM (Partner Relationship Management)
- Address Management
- Organization Management
- Document handling (EDHS in Danish)
- Service Management
- HR including competency management, time reporting and recruitment
- Project management supporting tasks such as procedures and campaigns
- Supply Chain Management

A variety of enterprise-class business applications can, with small costs and a stepwise approach dependent on competitive advantages, be build using the components. This can be done re-using existing legacy systems, ERP systems, workflow systems, EDI, CRM systems, SCM systems, project management systems, and HR systems - thus enabling the functions of these applications together with On BRM's components to be used in the new business opportunities without the removal of existing data, and without intensive and costly training of users.

With On BRM the integrated applications' data can be published in a fine-grained standard way tailored for an organization's and its partners' business processes based upon all or some of the components of the Web services application stack. The publishing can run across many channels such as Web services, wireless communication, MS Office, Portals including corporate and trading portals.

The On BRM Framework is a programmable business service and integration platform, containing hundreds of business components that can be used as building blocks.

On BRM's adaptive architecture and components offers means to reduce expenses to external resources such as Web services and applications and to document earlier business experiences with external resources.

Being designed and developed with the aid of well known design patterns and proven technology, the framework is very reliable with a high availability and with failover. It is highly scaleable and can therefore be deployed in a variety of architectures. It has an excellent performance and supports load balancing.

Installing On BRM, an organization does not get a new isolated application requiring the keying in of all necessary information such as employees and customers once more. On BRM can just be connected to legacy applications containing these data. If it is appropriate redundant applications can later be phased out stepwise in an order and with a speed according to the organization's business strategy.

The On BRM Framework supports the OIOXML (Infostrukturbasen) specifications and OCES (digital signature).

As On BRM supports many security technologies, such as SSL (Secure Socket Layer), WS-security, SAML (The Security Assertion Markup Language), digital signature

including OCES (Offentlige Certifikater til Elektronisk Service), encryption, and DMZ (Demilitarized Zone), the security of On BRM is state-of-the-art.

Currently, among others, a car sales and financing system, and an order/booking system for handling of patients and their transportation to and from hospitals have been built upon the framework.

Overview of the framework's architecture

On BRM is a middle-tier, distributed, and multi-layered J2EE enterprise-class framework comprised of hundreds of cooperating components that an organization can use as process templates and building blocks for its applications. On BRM is based on the Service-Oriented Architecture (SOA - see appendix A).
An overview of the framework and its surroundings is given in figure 1. An alternative way of describing On BRM's separation into layers might be based on the Service Layer pattern (see appendix B).
The framework consists of a Presentation layer, a Business layer and Back-end integration layer. The Presentation layer is based upon the Model View Controller pattern (Glenn E. Krasner et al.) and has two layers: The Controller layer and the View layer (see figure 3). The Presentation layer can run on a free of charge web server such as Apache with Tomcat. The Business layer contains a Business process template layer running on the web server and a Domain logic layer running on an application server such as WebSphere or WebLogic. The Domain logic layer contains a Session Façade and business objects. The Back-end integration layer is capable of communicating with resources such as workflow systems (appendix C), Web services, databases, and legacy systems.

Most systems that are developed for the enterprise today rely on the J2EE platform, therefore On BRM offers seamless integration within today's IT environments, while relying on the standards-based Web service stack for integration with the .NET platform. On BRM can communicate with the .Net platform without using the Web service stack but the use of the Web service stack offers many advantages as seen below.
Most organizations' infrastructure has something in common with figure 4. This architecture is very complex, and the communication between the different components is sequential and based on proprietary point-to-point protocols. The transactions will often be coarse-grained.

Even though there are integration services such as application servers which the legacy systems can communicate with through adapters, the legacy systems require different tools and deployment runtime configurations in the Gateway service to support distributed transactions. This makes it practically impossible to integrate the different processes in distributed transactions because it leads to a split of the business process into business process logic and transactions. This is also true of commercial off-the-shelf applications (COTS) and custom applications.

In figure 4 most of the communication is based on XML but replacing proprietary data-formats and protocols with XML and HTTP does not automatically lead to integration. It only provides certain advantages over e.g. comma-separated files:

- Comma-separated files are very compact, containing almost nothing except the data; by comparison XML files are much larger. On modern networks this is rarely a problem
- Both types of files are easy to extend if new fields of data are needed. Programs using comma-separated files for data transfer need to be updated to be able to use

the new files, even if the program does not use the new fields. In the case of XML old programs continue to work with the new files, the programs need only to be extended if they use the new fields. This makes XML more flexible than comma-separated files

- The tags in the XML files often make the XML file more human readable than a comma-separated file
- The XML specifications include standardised methods for validating the content of the XML-files; a similar method does not exist for comma-separated files.

A structured alternative to the traditional architecture in figure 4 is a simple and well designed architecture based on the Web service stack (see figure 5).

Web service stack

The On BRM Framework supports SOA and is therefore built upon the emerging Web service stack. Lawrence Wilkes has written a paper about the Web services stack and given a time schedule for the expected implementation of the Web service stack.

This paper is not a detailed introduction to the Web service stack or the functional requirements of Web service stack-based application development. However the paper will in the following paragraphs explain, in brief, definitions, technologies, layers, and the components of the Web service stack.

Web services

Gartner says that 'Web services are loosely coupled software components that interact asynchronously with one another dynamically via standard Internet technologies'. Forrester says Web services are 'automated connections between people, systems and applications that expose elements of business functionality as a software service and create new business value'. Gartner's definition is more technical than Forrester's and this paper is more in accordance with Gartner's definition.

To get a Web Service to speak to another Web service either an application server or other middleware, such as Common Object Request Broker Architecture (CORBA), Java 2 Enterprise Edition (J2EE) or .NET is required. A stack of Web service interfaces is also required. A standard for the stack of these Web service interfaces is emerging called the Web service stack. An overview of the Web service stack and how On BRM builds upon the Web service stack is given in figure 6.

Web services are built around XML, XML Schema, SOAP (Simple Object Access Protocol), and WSDL (Web Services Description Language). XML is used as the common description between systems, Schema is used to describe the data carried in the XML payload, SOAP is used (among other things) to provide a way of framing XML messages, and a WSDL document describes a Web service in detail.

XML Schema

W3C defines XML Schemas as schemas that express shared vocabularies and allow machines to carry out rules made by people. They provide a means for defining the structure, content and semantics of XML documents.

WSDL

A WSDL document describes a Web service in detail:

- Location of the service
- Messages the Web service will send
- Data types used in those messages
- Style of the messages.

When a Web service is created the WSDL should be the starting point as it provides the contract that both the client and the service must adhere to. A WSDL document can be used to generate server-side skeletons and/or client-side stubs. On BRM contains hundreds of fine-grained WSDLs and a developer can easily write new WSDLs based upon the building blocks of On BRM.

Web service transport layer

From the bottom up (follow stack on figure 6), standard transport protocols such as HTTP, HTTPS, FTP, and SMTP serve as transport protocols for Web services. HTTPS uses SSL (Secure Socket Layer) as security. SSL is designed to provide point-to-point security at transport level, which falls short for Web services because they need an end-to-end security at message level, where multiple intermediary nodes could exist between the two endpoints. As described later, WS-Security addresses these needs.

SOAP (Simple Object Access Protocol) serves as a transport protocol for the services exposed through the WSDL for the next level.

Presys supports Axis (see appendix D) as the implementation of SOAP and Axis allows On BRM's components to be deployed as Web services.

Web service communication layer

The Web service communication layer contains several components as you can see in figure 6. They will be described in the following paragraphs.

UDDI

UDDI (Universal Description, Discovery, and Integration) allows businesses and trading partners to find, discover and inspect one another and their Web services in a directory. For business relationships where a more detailed knowledge about the partner is not required UDDI is relevant – e.g. a Web service for weather reports. But for business

relationships where a deeper knowledge about the partner is required, before an organization wants to do business, UDDI is not sufficient. Therefore On BRM supports UDDI while it at the same time maintains its own UDDI tables, where an organization can track its relationship to its business partners. For instance:

- How is a partner's service management
- What partner status has the other organization obtained
- How often has our organization done business with the other organization
- Who is the other organization's competitors
- How well does the partner perform.

BPQL

BPQL (Business Process Query Language) is a language for managing a business process management infrastructure that includes a process execution facility (process server) and a process deployment facility (process repository). The BPQL enables business analysts to query the state and control the execution of process instances. BPQL is supported by BPMI.ORG.

WS-Security

WS-Security describes enhancements to SOAP's end-to-end messaging to provide protection through message integrity, message confidentiality, and single message authentication. These mechanisms and technologies can be used to accommodate a wide variety of security models and encryption technologies such as PKI used by OCES (Offentlige Certifikater til Elektronisk Service), Kerberos (see appendix E), and SSL in a platform- and language-neutral manner. WS-Security is the de facto standard supported by OASIS for secure Web services. The specification was published by IBM, Microsoft, and VeriSign a couple of years ago.

WS-Security specifies:

- A general-purpose mechanism for associating security tokens also called tickets with messages
- The rules for using and processing XML Signature
- XML Encryption (see appendix F).

Although the processing rules must be followed using any type of security token including XML-based tokens, it is not necessary that XML-based tokens be signed or encrypted.

WS-security does not require a specific type of security token. Therefore WS-security's design is extensible i.e. support multiple security token formats. For instance, a client might provide proof of identity together with a proof that he or she has a particular business certification.
XML-based tokens use the Security Assertion Markup Language (SAML - see appendix G) as a standard framework.

WS-Security also specifies how to encode binary security tokens. Specifically, the specification describes how to encode X.509 (see appendix H) certificates and Kerberos tickets as well as how to include opaque encrypted keys.
It also includes extensibility mechanisms that can be used to further describe the characteristics of the credentials that are included with a message.

WS-Transaction

The WS-Transaction specification is developed by Microsoft, IBM, and BEA. It defines how Web services coordinate their activities in order to ensure the integrity of underlying data source operations. WS-Transaction specifies two architectures for transactions over Web services based upon two-phase commit (2PC):

- Atomic transactions where the highest level of protection is required.
  An atomic transaction coordinates activities having a short duration and being executed within limited trust domains. They are called atomic transactions because they are all or nothing transactions. They are either committed or aborted. 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.
- Business activity transactions also called long running transactions where a lower level of protection offers better scalability.
  A business activity is used to coordinate activities that:
  - are long in duration
  - need logic to handle business exceptions.

  Data sources could be locked for a long duration and therefore locking is prohibited and the business activity transactions are tentative.
  The Business Activity specification defines protocols that enable transaction processing systems and workflow systems to wrap their proprietary mechanisms and interoperate across trust boundaries and different vendors' hardware and software implementations.

Long running transactions/business activity transactions are in BPEL (see appendix I) centered on scopes. A scope can be nested.
WS-Transaction specifies what will constitute the completion of the long running transaction. An agreement protocol between a scope and its parent scope analyzes and determines the result of a long running transaction.
For the moment WS-Transaction only supports a scope and all its nested scopes contained within a single process, which is hosted at a single BPEL engine. In the future BPEL might support long running transactions that are distributed across processes and even across BPEL engines.

WS-Coordination

WS-Coordination is both a specification and a framework for how individual Web services can interact in order to accomplish an application task. The framework enables

existing transaction processing systems and workflow systems to hide their proprietary protocols and to operate in a heterogeneous environment by providing protocols to coordinate distributed transactions. The framework enables an application service to create the context for propagating a transaction/activity to other services and registering for coordination protocols. The coordination protocols are used to support a number of applications which need to reach consistent agreement on what will constitute the completion of distributed transactions.
The WS-Coordination specifies:

- A context for coordination
- The specific items of data that are to be exchanged in order for transactions to complete successfully, as part of an overall business process defined in for instance On BRM or BPEL (see the paragraph Business process modelling layer).

WS-Coordination is developed by IBM, Microsoft, and BEA Systems.

WSCI

WSCI (Web service Choreography Interface) is another standard used for:

- Orchestration of atomic Web services' transactions based on a business process
- The definition of interfaces between multiple processes, especially for processes that are defined in different languages such as BPML and BPEL (see appendix I).

WCSI has taken a step towards standardization through a submission to W3C.
On BRM does not enforce the use of either of the two standards for Web service coordination/choreography and transactions, since On BRM's business components support both standards depending on the used application server.

Application server layer

The next layer is the Application server layer. The application server (see appendix J) communicates with the Web service communication layer and the Back-end integration layer in On BRM. The best-of-breed application servers such as WebLogic and WebSphere already support or will in the near future support the above mentioned lower layer protocols and specifications of the Web service stack.

Back-end integration layer

The next layer in the Web service stack is the Back-end integration layer. In On BRM the entity beans, DAOs and Gateways (see appendix K) handle this communication. This layer communicates with the Business process template layer across the Domain logic layer in On BRM.

Business layer

The next layer in the Web service stack is the Business layer. This layer contains several sub layers. The first of these sub layers is the Domain logic layer. The components in the Back-end integration layer communicate with the business object in the Domain logic layer. On BRM's business objects perform and support in the context of the Web service stack many business services and some of these services are:

- Single sign-on
- Identification and authentication
- Security
- Database transaction management
- Load balancing
- Failover
- Integration and management of internal and external applications
- Document handling
- Search engine
- Personalization
- Business process modelling and execution
- Handling of XML-standards
- Data cleaning.

Single sign-on

This component handles the identification and authentication mechanism to establish an entity's identity to the On BRM. The entities might be persons, companies, applications, Web services, and many other entities. In any case, the entity must be identified and authenticated before granted appropriate access rights. The Java Authentication and Authorization Service (JAAS) provides this service. It implements a Java version of the standard Pluggable Authentication Module (PAM) framework, and supports user-based authorization. With PAM, multiple authentication technologies can be added without changing any sign-on services, thereby preserving existing system environments. PAM can be used to integrate sign-on services with different authentication technologies, such as Kerberos, DCE, RSA, S/KEY and smart card. Therefore the users are not forced to type multiple passwords even if the users have to use multiple authentication mechanisms. So On BRM offers single sign-on just like SAML but at another level of the Web service stack. This pluggable architecture enables On BRM to be plugged into existing environments for instance corporate portals (because many corporate portals support PAM) or to be a portal for an integrating/unified sign-on of an organization's existing environment. An organization can choose which of the single sign-on mechanism it wants depending on its needs.

Identification and authentication

On BRM can establish the identification and authentication mechanism for documents using digital signature (see appendix L) like the Danish digital signature OCES

(Offentlige Certifikater til Elektronisk Service). On BRM can store and get the public key from TDC or EuroTrust PKI Services A/S, for each external entity, such as partners, customers, or citizens and store the private key for internal entities such as ones own organization. As mentioned above On BRM supports WS-Security, and all the components included in WS-Security.

Security

As a frontline to protect the On BRM Framework from direct exposure from an untrusted environment, On BRM supports DMZ (Demilitarized Zone) (see figure 13). Internally On BRM supports an XML firewall (see appendix M). Since On BRM can use WS-security, DMZ and XML firewall at the same time, the security of On BRM is state-of-the-art.

Database transaction management

On BRM uses entity beans and session beans to perform many of its services.
An entity bean models a business entity and performs activities within a business process. An entity bean can be considered as long living business data. Actually an entity bean maps data in a row of a table of a data source, and that is why they are persistent. Domain logic can be implemented in an entity bean e.g. to retrieve and perform computation on line items within a purchase order, but in On BRM entity beans are only used for data persistence, not for business or domain logic. Using entity beans for data persistence only is often described as a Row Data Gateway pattern (Martin Fowler). If an entity bean contains domain logic it is built using the Active Record pattern (Martin Fowler). Entity beans survive beyond the end of an application session, even a server crash or a network failure, because the data is stored by the container (application server) in some form of data storage system, such as a database. The EJB specification defines the following persistence mechanisms for entity beans:

- Bean-managed persistence (BMP), where data is stored and retrieved through methods implemented by the bean developer. These methods typically use JDBC or SQLJ to manage persistence.
- Container-managed persistence (CMP), where the container handles the persistence issues automatically without any programming efforts.

As container-managed persistence keeps an entity bean class separate from the methods enabling persistence, the developers might change a bean's data source without affecting its implementation. Instead of writing Java code for enabling BMP, CMP is described declaratively in a deployment descriptor file. At runtime, the container manages the bean's data by interacting with the data source, typically a relational database or an external resource such as SAP's Netweaver, designated in the deployment descriptor. On BRM is based on CMP and all data source calls are performed through entity beans and not session beans as you will see later. To enhance performance, read-only queries involving many records can act through DAOs (see paragraph Data Access Object). Any use of DAOs is encapsulated within the container and is therefore managed by the container.

As business flourishes very rapidly, business logic is also changed day by day. That is why On BRM does not contain business logic in the entity beans. Instead On BRM uses session beans and 'plain old java objects' (POJOs) to provide a mechanism for developing an enterprise level business system. In On BRM the session beans manage the business objects, which are POJOs, in a grid of interconnected objects. These business objects contain all of On BRM's domain logic. The use of POJOs ensures a high level of maintainability as well as provides a more dynamic system, since different POJOs with different domain logic can easily be plugged in or out. This also ensures that changes will be highly localized when On BRM is upgraded so it will have no impact upon the rest of On BRM. The users of On BRM will not be disturbed by such changes.

There are two types of session beans:

- Stateful session bean
- Stateless session bean.

Stateful session beans

Stateful session beans manage transaction and can process the business logic that is both the business process logic and the domain logic. Stateful session beans are usually used to act as agents for the client, managing the interaction of other beans and performing work on behalf of the client application while maintaining a conversational state with the client. On BRM currently does not use stateful session beans since applications based on stateless session beans scale better and tend to be easier to perform load balancing on.

Stateless Session beans

Stateless Session beans are EJBs that complete a task in a single step. They are the simplest of all EJB components. They make small demands on the EJB component developer and are used to implement very simple operations. The term stateless refers to the fact that stateless session beans cannot hold any information for an EJB client between method invocations for that client. They are held in a shared pool on one or several containers between each method invocation on the bean.

A typical usage of both stateful and stateless session beans has been to access a database directly via SQL, including inserts, updates and domain logic. This pattern is called Transaction Script (see Appendix N). One problem with this practice is that it binds the system to a database, making it impossible to work with data source entities from other systems and applications such as SAP or Siebel. Other problems are that the system quickly becomes dependent on one specific database vendor and that it ruins the advantages of the layers. Transaction scripts tend to yield more coarse-grained interfaces resulting in a Blob antipattern (see Appendix O), which is practically impossible to present as a Web service. On BRM avoids all these problems by not using the Transaction Script pattern but instead moving the business process logic to the Presentation layer and all domain logic to a grid of interconnected plain java classes (see the paragraph about Business Objects) in the Domain logic layer, leaving only the transaction management to the stateless session beans based on CMP.

By using stateless session beans without any domain or business process logic On BRM supports distributed transaction even without using BPEL (se paragraph Business process modelling and execution).

Load Balancing

Load balancing is the ability to alter the location where similar requests are handled. An example is a request for a method on a stateless session bean which might be handled on any of the stateless session beans in any of the pools on the available application servers. When a client invokes a method on a stateless session bean, it is removed from the pool, the method is executed, and the bean is immediately returned to the pool. Consequently, if two concurrent methods on a stateless session bean are executed, different beans in one or more EJB containers would probably service them. Since all instances of a stateless session bean are identical, a home or remote stub can have its request serviced by any available stateless session bean on any server. The remote stub doesn't care whether an instance in the first or the second server handles the request, as long as the request gets handled. On BRM uses stateless session beans for enabling load balancing.
Another example of load balancing is a request for a database connection that might be handled on any server in a cluster. A cluster is a loosely coupled group of application servers that collaborate to provide shared access to On BRM's services. The cluster aims to balance resources requests, and failover of business logic to provide overall scalability.

Failover

Due to the fact that On BRM is based on CMP and uses stateless session beans On BRM supports failover and clustering. Failover is the ability for a request that is being serviced to have a high availability switchover to another server without disruption of the service. How these mechanisms are implemented depends on the application server that is used.

Integration and management of internal and external resources

As many internal and external resources are fairly unsophisticated and only support a sequential proprietary protocol but still bring a lot of value, when being published to other applications, On BRM has many layers in order to integrate these resources in the organization's internal and external business processes, and figure 28 shows how On BRM can:

- Integrate internal resources that are published as Web services (see the paragraph 'On BRM architecture' about On BRM and the four tiers: Client tier, Web tier, EJB tier, and EIS tier)
- Combine and process these Web services based upon an organization's business processes
- Publish the new combined information in a personalized manner to other organizations such as divisions, business partners, and customers.

The uneven architecture where some applications support the Web service stack in full scale, some an application server, some XML and some are still based upon sockets, is handled by On BRM through Gateways (appendix K). For each resource there is a

gateway. All external resources behave as domain logic from the view of the above layers. These external resources can be published as Web services (SOAP, WSDL, and UDDI), that other applications can use as templates (see figure 1) or in one of many other protocols such as J2CA. Note here the use of the Gateway pattern (Martin Fowler) internally in the system and the Business process template layer to provide an external façade (Erich Gamma et al.) to the services (via Business Delegates and WSDL).

If several external services logically constitute the same service, for example a social security service that differs in each country, it will be put behind the same gateway. As Larry Constantine et al. describes in Structured Design, this ensures maximum cohesion in the gateway and therefore lowers the coupling between business objects and gateways. In the specific example in figure 14 the gateway can lookup information based on social security number and country. The gateway finds the right external service and builds the response. A standard common response is built and the Builder pattern (Erich Gamma et al.) is used to provide internationalization by building country specific responses that the business objects can choose to use. In figure 15 you can see a diagram of the Social Security Number component in On BRM.
Another way to provide the internationalization is to provide a general response and let the Presentation layer sort out what to present. This strategy is used on addresses where a common object that contains all relevant fields is returned. The Presentation layer then decides for example if a region is to be considered a state.

It is easy to change the presentation of information so it is not necessary to have a one-to-one relationship between the way the external resource publishes its information and the way On BRM based Web service publishes its information. On BRM can act as an intermediate between several internal and external resources so that only one set of data is presented even if more than one of the sub systems returns data.

Besides Web services, Presys also handle J2CA, J2EE, and transaction managers such as CICS and Tuxedo (see figure 16). Presys' components can be integrated seamlessly and flexibly with other applications supporting J2EE and J2CA such as SAP's, Documentum's, SAS's, or Siebel's different modules. Presys' components can also be integrated seamlessly and flexibly with legacy applications through CICS, Tuxedo, and MOM (message oriented middleware) such as MQSeries.

It is of course possible in the business layer to develop and integrate new components in J2EE as plug-ins to the On BRM Framework, for tailoring On BRM to new business process logic and domain logic. These components/plug-ins can enhance the business logic components included in On BRM. This enhancement can be done flexibly by inheritance, wrapping, or developing a new vertical solution using On BRM's Session Facade and using On BRM's Business Delegates.

Document handling

Traditional workflow systems have usually hard-coded document management capabilities using proprietary business process execution models and APIs. Such an

approach has proved itself successful for document-intensive applications that required integration with various scanning devices or storage systems, but it lacks the flexibility to adapt to the requirements of heterogeneous environments that need a more standard-based approach. Traditional workflow architecture (see appendix C) has gained a substantial market penetration in vertical markets such as financial services (especially insurance and retail banking), government, and healthcare, but the penetration of other markets has been small. The reason for this lies in the nature of the business processes that are predominant within these vertical markets. These processes are often very document intensive, e.g. tax filling and insurance claims, and they can be designed in a very sequential manner. The documents go from one participant to another and tasks are assigned to participants for the processing of these documents. Even though very attractive for its ability to streamline otherwise manual-intensive processes, this approach does not reflect business processes which are made of multiple independently managed divisions and relationships with external participants (suppliers, partners, customers), and for which multiple independently managed processes must collaborate in a parallel rather than sequential manner (see figure 17). An example of this is the building industry. Many documents are here used by many participants at the same time especially in the design phase, and the documents comes from many sources like the government, architects, suppliers, and contractors.

Following a pure process-oriented approach, On BRM defines a set of services for document management by providing standards-based interfaces such as the Web Distributed Authoring and Versioning protocol (WebDAV - see appendix P) and takes advantage of its flexible business process logic to overlay business process-oriented document management capabilities on top of document- and task-oriented document management systems. As a result, existing document management systems can be used as document repositories through standard interfaces, and the most complex document-oriented workflow processes can be defined as part of end-to-end processes. The file management features in WebDAV makes it a cost-effective alternative to traditional document management system. Therefore all stakeholders, who produce relevant information and post it, are members of a virtual information center (see figure 18). Each member has appropriate access rights and can manipulate information on demand. The virtual information center supports mixed technologies such as EDI, workflow systems, and private mail systems.
A sales team, which is working from remote offices or travelling, may need to access information about their company's latest product release. If information such as price sheets, screenshots and Flash demos is stored in a common location on a WebDAV-compatible server used by On BRM, the sales team members can each use HTTP and WebDAV to view and customize the sales materials for their own purposes through On BRM. In this way they can increase their productivity.

So based on an organization's business processes On BRM lets it flexibly manage rich business assets such as orders, forms, emails, contracts, journals, documents of a case, product information, and reports.

Search Engine

On BRM's search techniques are based on Jakarta's Lucene. Lucene is a high-performance and full-featured text search engine. Lucene's main features are:

- Scalable, high-performance indexing:
    - Rapid
    - Incremental indexing as fast as batch indexing
    - Small RAM requirements
    - Index size roughly 30% the size of text indexed
- Powerful, accurate and efficient search algorithms:
    - Ranked search (best results returned first)
    - Boolean and phrase queries
    - Date-range searching
    - Field search e.g., contents, title, and author.

Personalization

A distinction is often made between customization and personalization. Customization occurs when the user configures an interface and creates a profile manually by adding and removing elements to the profile. The control of the look and/or content is explicit and is user-driven. In personalization the user is seen as being somewhat less in control. On BRM supports personalization by assigning user specific roles and supports customization through client products such as MS Office.Net.

The user has attributes, interests, needs, some or all of which need to be captured and processed. The techniques used to complete the profile of the user are varied and, as mentioned, may engage the user to different degrees. Furthermore, there are other differences in how the 'appropriate content' that matches the user's needs is determined and delivered.

On BRM enables a profile to be built through active involvement of the user through fill-in forms. Users can control the type of content provided, by indicating their choices through their profile. The picture of the user, built through the profile, may consist of generic information (such as job, job position, role in a business process, partner type, customer type, country or area code). It may also include explicitly stated wishes of specific content, such as a general area of interest (e.g. genre of products), or campaign management such as particular leads or particular customers for the organization's sales people and partners. The personalization may therefore be based upon a particular role or a collection of roles. This style of personalization requires the users to exert most effort and make the initial investment; it depends on the motivation and the ability of the user to set up complex customization features. If users are reluctant to spend time setting up complex personalization features the service may remain underutilised. Therefore On BRM enables a centralized way to fill in a profile where the information is available upfront. However the profile may remain static and not change with the user's changing needs (unless the user puts in the effort to update it and it is easy to do so). On BRM's HR components support these techniques and these components can be integrated with an organization's existing HR software.

Business process modelling and execution

The On BRM enterprise framework handles a multitude of business processes, including:

- CRM
- PRM (Partner Relationship Management)
- Service Management
- HR
- Project management
- Supply Chain Management.

The On BRM Framework is a programmable business service and integration platform, containing hundreds of business components that can be used as building blocks. Therefore a variety of business applications can easily be build using the framework. In figure 19 you can see how Presys' CRM differs from traditional client/server based CRM systems from a technologic and development perspective. As you can se it is a much more complex and expensive development process to tailor a CRM application based on a client/server pattern than it is to tailor one based on a SOA pattern.

Handling of XML standards

On BRM handles the many different business specific XML standards and the transcription between them as well as the conversion to Java objects. For example different applications and different services may provide an address in many different XML formats. Another example is that the same item e.g. social security number has a different XML format in different countries.

Data cleaning

Databases often contain dirty data due to:

- spelling, phonetic and typing errors
- word transpositions
- extra words
- missing data
- inconsistent values
- illegal values
- synonyms and nicknames
- abbreviations, truncation and initials.

Due to insufficient updating many databases often need cleaning. To help reduce mailing costs, search for names, addresses, telephone numbers, e-mail addresses, products, documents and to avoid embarrassment, On BRM offers a range of data cleaning services, including de-duplication, for persons, organizations, products, and documents. On BRM also handles the identification of those who have moved from the address recorded, changed a telephone number and those who have passed away. The identification of dirty data is handled at item level and many data sources can be

compared at a time. The dirty data may come from both internal and external resources such as applications, databases and Web services.

Business process modelling layer

The next layer offers a standard for defining how to send XML messages to remote services, manipulate XML data structures, receive XML messages asynchronously from remote services, manage events and exceptions, define parallel sequences of execution and undo parts of processes when exceptions occur. These are the elements needed to compose a set of services into collaborative and transactional business processes. BPEL and BPML (Business Process Modelling Language – see appendix I) are languages that provide these services by means of establishing a bilateral partnership, correlating messages and processes, defining the order of the activities of a business process, and handling exceptions.

As you have seen above, On BRM offers mechanisms enabling flexible procedures for handling an organization's uneven architecture, where some internal and external resources support the Web service stack in full scale, some an application server, some XML and some is still based upon individual protocols using sockets. All the data sources are collected and processed through On BRM and therefore behave as integrated and collaborative business components. Since they can be accessed through Business Delegates and WSDLs, it is possible to manage all On BRM's components by the means of BPEL and BPML.

As mentioned above WS-Transaction supports the scope and all the nested scopes contained within a single process, which is hosted by a single BPEL engine. In the future BPEL may support long running transactions that are distributed across processes and even across BPEL engines. For the moment On BRM supports distributed transactions through application servers.

Many visual modelling tools are being developed at the moment, and On BRM supports most of them because they are based on WSDL.

Presentation layer

The last layer is the Presentation layer. This layer enables On BRM's business components or individually tailored components to be used as business process templates. On BRM is interfaced through WSDL or Business Delegates and therefore supports several user interfaces (see figure 22) such as

- Windows
- Portals including corporate portals and e-commerce portals
- Browsers
- Highly visual interactive content such as Flash
- Wireless/pervasive devices.

Windows

Since many organizations want a uniform user interface to reduce the training and change management costs, On BRM supports a Windows user interface. It is possible to use Microsoft's tools to develop Microsoft smart clients (rich and thin) or other Windows interfaces where On BRM's business process templates are used as the building blocks for MS Office, MS Project, VB, C#, or VBA. On BRM supports both .Net and non-.Net versions and technologies. Therefore migrating and upgrading costs can be reduced. For MS products On BRM acts as a web service which the MS products connect to by means of VBA code connecting to On BRM's WSDLs. .Net programs can also connect directly to the Business Delegates via IIOP. So with On BRM it is possible to personalize the Windows user interfaces so each user interface reflects a business process.

Portals

Corporate portals serve a wide range of uses in the enterprise, from internal self-service and knowledge management portals to external business-to-business catalogues and e-commerce platforms. They are aggregating information and content (both static and dynamic), integrating functionality from internal and external resources, and providing unifying services such as user access control, single sign on, personalization, and a common user interface.
A portlet is a Java technology based web component, managed by a portlet container that processes requests and generates dynamic content. Portlets are used by portals as pluggable user interface components that provide a Presentation layer to the business process template layer. The content generated by a portlet is called a fragment. A fragment is a piece of markup (e.g., HTML, XHTML, WML) adhering to certain rules and can be aggregated with other fragments to form a complete document. A portal normally adds a title, control buttons and other decorations to the fragment generated by the portlet; this new fragment is called a portlet window. The portal then aggregates the portlet windows into a complete document, the portal page. On BRM's components or individually tailored components can be seamlessly connected as portlets in the organization's portal environment because Presys support the Portlet API called JSR 168 (see appendix Q).
A Web services standard for the 'plug-n-play' of portals has also been advanced by members of the OASIS Web Services for Remote Portlets TC (Technical Committee). The goal of this specification (see appendix R) is to develop a web services standard that will allow for the 'plug-n-play' of portals, other intermediary web applications that aggregate content, and applications from disparate sources. The OASIS TC recently voted to approve its latest WSRP deliverables as a Committee Specification and to submit a request for approval of the WSRP specification as an OASIS standard.

Many vendors of portals such as BEA and IBM have an integrated connection to Struts (see appendix S) and they often provide a visual environment to build rich clients based on Struts.

Presys supports many portals such as WebSphere Portal, WebLogic Portal, SAP Netweaver (mySAP Enterprise Portal is built on SAP Netweaver), and Oracle9iAS Portal.

Browsers

On BRM supports browsers such as Internet Explorer, Netscape, Mozilla and Opera.

Highly visual interactive content such as Flash

On BRM supports products such as Flash. With Flash it is possible, as many organizations already do, to develop Internet solutions with highly visual interactive content.

Wireless/pervasive devices

Mobile employees often need access to information from their organization's database during a meeting or on the road, while away from their desktop. A common practice today is to use wireless/pervasive devices such as Personal Digital Assistants or cell phones. Their interfaces are very varied in screen size, processing power, connection speed etc. Using Struts and Cocoon (see appendix S and T), On BRM can adapt its interface to different clients, such as WAP browsers or HTML browsers with limited screen sizes.
On BRM also supports SMS. Therefore it is possible to synchronize calendars and other applications in cell phones. This synchronization is a bi-directional synchronization, so a personal Outlook calendar can be synchronized with a personal calendar in a cell phone. The calendar in On BRM will also be synchronized.

On BRM Architecture

In the sections above On BRM is described in connection with the Web service stack architecture (see figure 5). But On BRM can run using only some or even none of the other components which constitute the Web service stack (see figure 30). These components may be contained in an application server or be separate components which On BRM access. The following describes the architecture and environment of On BRM without the use of the components of the Web service stack. On BRM has from a logical perspective a general three-layered architecture (see figure 23) that supports the separation of Presentation layer, Business layer, and a Data source layer:

- The Presentation layer is responsible for implementing the business process logic and managing user interfaces, including thin and rich clients such as portlets, MS Office or tailored applications running on Windows, browsers, corporate portals, or wireless/pervasive devices. Components such as Struts and Cocoon support these services.
- The Business layer. The Business layer is comprised of the Business process template layer and the Domain logic layer which consists of the Session Facade and Business objects. The Domain logic layer contains enterprise-class business components to sustain an organization's business processes. By designing Business Delegates and WSDLs or using IIOP On BRM's business components can be modelled to sustain specific business processes that can be accessed by tailored user interfaces.
- The Data source layer. The Back-end integration layer provides connectivity with internal and external resources (for instance components such as Web services and legacy systems) through entity beans, DAOs and other kinds of Gateways.

This three layer architecture supports stepwise and separated development as each layer requires different set of skills:

- Presentation layer: Interface developers, who deal with presentation coding with tools like VB.Net, C#, Delphi, Java, JSP/portlets, ASP.Net, or Flash
- Business layer: Business analysts, who design the business processes that define transactions, services, and operations. Developers who code the business processes and have knowledge of the components of On BRM
- Data source layer: Architects, developers, and other IT staff who build, install, maintain, and customize the internal and external resources that On BRM integrates and communicates with.

This logical three layer architecture is not identical to a traditional three layer architectures such as Microsoft's DNA (see appendix U).

From a physical perspective On BRM can be viewed as a four tiers application (see figure 11):

- The Client tier that consists of rich and thin clients including portlets, MS Office or tailored applications running on Windows, browsers, corporate portals, e-commerce portals, and wireless/pervasive devices
- The Web tier that consists of presentation technologies such as Cocoon and JSP/Struts and On BRM's Business process template layer

- The EJB tier that consists of On BRM's Domain logic layer and the Back-end integration layer
- The EIS (Enterprise Information System) tier that consist of internal and external resources including On BRM's database or parts of it.

The Web tier and the EJB tier combined are called the Middle tier.

As you can see in the figure 11 and figure 23 the two models describing On BRM are on a superficial view conflicting as the physical layers and the logical layers do not match. But in fact it is a great strength as On BRM thereby achieves a scalability and a performance that are ahead of the scalability and performance that are encapsulated in applications servers' best practises, that have sustained distributed computing for decades such as resource pooling (see paragraph Load balancing).

The Presentation layer

Following the stack in figure 23 we begin top-down with the Presentation layer. The Presentation layer implements the business process logic. On BRM is accessed by rich and thin clients running in the Client tier. The clients can be developed in any language that the customer wants. If the customer prefers clients developed with Microsoft or Microsoft compatible tools On BRM offers WSDLs and IIOP. If the clients are developed with Java based tools such as portals/portlets On BRM offers Struts, Cocoon, WSDLs, and IIOP (see figure 1 and figure 28). The business process logic can be coded directly into the clients or be implemented in helper objects that can be reused in different clients.

Cocoon

A large number of technologies such as PDF files, MS Excel files, WML are easily implemented using the JSP pages by generating XML, and parsing the result through Apache Cocoon (appendix T), which is a framework for XML web publishing. Cocoon runs on a Web server in the Web server tier.

Struts

The JSP/Struts framework (see appendix S) is designed as a Model View Controller Model 2, also known as MVC2-pattern (Erich Gamma et al.). The pattern prescribes a Controller that receives requests from clients, validates these, and begins the appropriate interactions with the model, which in this case is the Business Delegate in the Business process template layer (see figure 1) and thereby implements the business process logic. The Business Delegates in the business process template layer handle all communication with the Controller on behalf of the business processes. The Views present the data provided by the business processes.

The On BRM Framework uses the Apache Struts Framework to implement MVC2. On BRM's MVC2 is a set of cooperating classes and JSP tags designed to control requests, validate input data and represent resulting views. New classes and JSP tags are easily written and implemented when new pages are demanded. Struts runs on a Web server in the Web tier.

Even the most server-oriented system needs a session ID on the client to manage and identify the state of the specific client because there are more clients than servers. In this way servers keep track of which client is which. A session is a single client's conversation which typically involves many request/response roundtrips, with a logical server, which may be formed of one or more physical servers. Session state is client specific data that is accumulated during the session. On BRM uses the Client Session State pattern to maintain the session state between the client and On BRM (see appendix V).

The Business layer

The Business layer is separated into:

- The Business process template layer that is responsible for the integration between the clients and/or Struts in the Presentation layer and the Session façade in the Domain logic layer
- The Domain logic layer that handles transactions and domain logic. The Domain logic layer consists of
    - The Session Facade that provides a simple and standardized interface to the Business objects
    - Business objects that implement domain logic data in order to provide business services to sustain enterprise-class business processes.

The Business process template layer

The Business process template layer is responsible for the integration between the clients and/or Struts in the Presentation layer and Session façade in the Domain logic layer. The Business process template layer runs on a Web server in the Web tier. On BRM's business components are accessed from the clients and Struts through a number of templates. These templates can be WSDLs or Business Delegates. WSDLs are automatically generated from the Business Delegates so what is valid for a Business Delegate is also valid for a WSDL. A Business Delegate is an external view (see figure 21) or an individual client's view of the domain. It may be thought of as a restriction of the domain - which is the total community client view - to just that portion that is of interest to that particular client. A Business Delegate/WSDL defines a client's view of the domain where the Session façade (Domain View) conceptually defines all business services. The Business Delegate may be a collection of entities such as companies and employees. It doesn't have to show all properties of an entity. The Business Delegates are designed as an application of the Business Delegate Pattern (see appendix X), the Data Transfer Object Pattern (see appendix Y) and the Remote Façade Pattern (see appendix Z). Significant advantages are obtained using this design:

- Changes to the Presentation layer caused by possible changes to the Domain logic layer or the internal and external resources contained in the EIS tier are vastly reduced. The On BRM Framework are therefore highly customizable and it is easy to tailor the framework to new needs and business processes
- The Business Delegates reduce the coupling between the Presentation layer and the Session Facades
- The Business Delegates hide the underlying implementation details of On BRM's business components by acting as a client side business abstraction. It is an external view of the domain model that is of particular interest for a client. Therefore developers working with the Presentation layer do not need knowledge of the underlying system
- The Business Delegates cache results and references to remote business components in the EJB tier. This significantly improves performance, because it limits unnecessary and potentially costly round trips over the network. With a high volume throughput it is very costly if the Web tier, where the Business Delegates reside, does not act as a cache because either the EJB tier or the EIS tier would have to do the work. Struts doesn't solve this problem because Struts doesn't handle caching
- The Business Delegates provide automatic failure recovery without exposing the problem to the client, for example in case of short network related errors
- The Business Delegates centralize communication with the underlying business components, thereby making the Client tier independent of the physical location of the application server or application servers running the Business objects, thus increasing the scalability of On BRM.
- Business Delegates are easily converted to WSDLs allowing:
    - On BRM Framework to be seamlessly accessed by .Net clients or other MS compatible tools thereby providing a standard Windows interface to the user
    - other Web service enabled systems to access all external and internal resources made available through the On BRM Framework from either On BRM's own components or external applications placed in the EIS tier
- The Business Delegates reduce the network traffic thus increasing performance. The Web tier and the EJB tier will typically be physically separated and connected through one or more networks, so data requests should be bundled. This is accomplished with Data Transfer Objects that, when transmitted between the Web tier, placed on a Web server, and the EJB tier, placed on an application server, are automatically bundled together into more coarse-grained objects, according to the frequency of use. This drastically reduces the number of network requests and therefore minimizes the amount of overhead transmitted. In order to connect from the Web tier to the EJB tier, knowledge of the interfaces of the Business Delegate classes and the Data Transfer Object classes is the only prerequisite (Appendix X provides an example of the company Business Delegate to illustrate the ease of use).

Tailored vertical solutions based on the On BRM Framework are easily developed by combining existing services into new Business Delegates. These existing services can be

separate legacy systems that can be combined into one simple interface through the Business Delegate. If the system requires transaction management, a vertical solution can be build up by constructing new Session Facades before exposing a combined service in a Business Delegate.

Developer Competence

A typical J2EE system consists of a servlet/JSP front that accesses stateless session beans directly. These stateless session beans communicate directly with a database using SQL (see figure 24 and appendix N). J2EE provides an easy to use transaction management scheme so a typical system could be programmed by developers whose main focus is the business domain. The detailed technical competencies are of lesser importance. On BRM makes it even simpler for the developers since integration through the Business Delegates does not require pattern, EJB or other J2EE expertise. Through the Business Delegates, On BRM provides a black box with a bundle of business components containing business processes easily accessed through a simple and easy to understand API (see figure 25).

Earlier in this paper the Service Layer pattern was mentioned (see appendix B) as an alternative description of On BRM's architecture. The Business Delegates is the outer part of the Service Layer in the Service Layer pattern.

The Domain logic layer

The Domain logic layer handles transactions and domain logic. It contains the Session Façade and Business objects. The Domain logic layer belongs to the EJB tier and runs on an application server.

Session Facade

The Session Façade provides a simple and standardized interface to the Business objects. The layer implements the Session Façade pattern (Deepak, Alur et al.) which is also called the Remote Façade pattern (see appendix Z). Each Session bean provides access to one or more business objects, and the related domain logic.

It is not required that the Business Delegates have a one-to-one mapping to the Session Façade (see figure 21). As mentioned in the paragraph about Business Delegates a Business Delegate specifies an external model or view of the domain presented by the Session Façade, in figure 21 called a Domain View. The Session Façade defines the total community client view of the domain. The domain is a view of the total content of business services and the Session Façade is a conceptual definition of this view. The third level in the developer's view of On BRM is the internal model which is a representation of low level Business objects, entity beans, DAOs, and Gateways. The internal model is a black box for the client side developer containing Business objects, entity beans, DAOs, and Gateways.
The mapping between the Business Delegates and the Session Facade specifies the rules for extracting the external model from the domain model (see appendix W). If On BRM only had provided the Session Façade the resulting system would not have been as

flexible, and resilient to changes in the business services. A change in the Session Façade would result in a change in all clients using the business service. The system would also have ended up with a tight coupling between the Client tier and the EJB tier resulting in decreased scalability because each client would have to know the physical location of the application server.

The Session Façade handles the transaction management in On BRM. The Session Facades provides the following advantages:

- The Session Facade abstracts the underlying internal model and provides a service layer that exposes only the required interfaces of On BRM's business services
- Transaction management for the underlying business services
- Greater flexibility
- Greater ability to cope with changes in the underlying business services
- Avoids exposing the underlying business objects directly to the client in order to keep tight coupling between the Presentation layer and the Business Objects to a minimum
- Provides a thin layer that handles the switch from a fine-grained to a coarse-grained interface.

The Business Objects

The Business objects are responsible for implementing domain logic and they provide business services to sustain enterprise-class business processes. The Business objects are implemented using standard plain java (POJO) to avoid unnecessary remote invocations and increase the flexibility of On BRM. Avoiding remote invocations in the Business Objects increases the performance of On BRM by providing a lean and more efficient system core. The transaction management is left to the session façade. For this reason On BRM does not use the Transaction Script pattern (see appendix N) in this layer but the Domain Model (see appendix W) pattern, which creates a grid of interconnected objects, where each object represents an entity. A Business Object implements domain logic and works on coarse-grained objects containing the data. But the Business Object itself is fine-grained with fine-grained interfaces. By delegating the domain logic to POJOs following the Domain Model pattern, On BRM avoids the bad performance and duplication resulting from using remote objects such as Session Beans for managing domain logic and thereby promotes reusability. The choice of objects in On BRM is not arbitrary. The manner in which the system is divided into parts significantly affects the structural complexity of the resulting system, as well as the total interconnected references. For this reason strong cohesion of each object in isolation – how tightly bound or related its internal elements are to one another – is a design principle. For the greater the cohesion of individual objects in the system, the lower the coupling between objects will be (Larry Constantine).

The Back-end integration layer

The last layer is the Back-end integration layer. It provides connectivity with internal and external resource for instance components such as Web services and legacy systems through entity beans, DAOs and other kinds of Gateways.

Entity Beans

As described in the paragraph about database transaction management, On BRM does not use entity beans for handling business or domain logic, and the entity beans are therefore not a part of the Domain Model pattern (see appendix W). On BRM implements entity beans using the Row Data Gateway pattern (Martin Fowler). An entity bean acts as a gateway to a single record in a data source, which is not necessarily a database, with one instance per row.

On BRM avoids coupling between the record layout and the code because only one entity bean access each table, and all business objects use the same entity bean.

The use of entity beans ensures easy replacement of database systems. The On BRM Framework is fully portable and supports Oracle and DB2 and other databases, and can furthermore run across many different databases at the same time. It is fully integrateable with legacy systems and ERP-systems such as SAP and People Soft through the J2EE Connector Architecture (J2CA) and the connection is encapsulated in a Gateway based on the Row Data Gateway pattern (see appendix K).

Data Access Object

In On BRM all updates and inserts to the data sources are performed by entity beans. To enhance performance, read-only queries involving many records or multiple tables can act through DAOs (Data Access Object). Any use of DAO is encapsulated within the application server using the same data sources as the container. DAOs can use SQL calls or navigation calls using entity beans.
By using SQL calls the following advantages are obtained:

- Enhanced performance compared to performing the same queries with only entity beans
- A quicker and easier development process

By using navigation and entity beans:

- Transparency to the actual data source implementation to provide easy migration to different vendor products or storage types
- The calls can traverse many physical separated databases or other data sources such as Web services

The performance can be rather low if using entity beans to access the data sources especially if using older versions of entity beans, without local interfaces (EJB 1.x). In this case it may be necessary to use SQL based DAOs to increase performance to an acceptable level, when performing searches which might result in large amounts of entity beans being initialized. The use of SQL based DAOs should be kept to an absolute

minimum as it also decreases the cohesion of the code and increases the coupling between code and data source. Presys only uses SQL based DAOs for enhancing speed of presentation for instance in data intensive searches such as finding companies. The results from the DAOs are presented as Data Transfer Objects (see appendix Y), so that it will be possible to replace the SQL based specific code of the DAOs with calls to the entity beans instead, returning the same results. If the table structure is changed it will only affect the SQL call. Not only will the DAOs be reusable but On BRM will be easy to maintain.
DAOs are a subtype of Gateways (see appendix K).

Adaptive scalability based upon a flexible architecture

On BRM is highly scalable and, because of On BRM's flexible infrastructure, an organization can adapt the level of scalability it wants considering costs, use, business processes and platforms:

- It can run on a single machine which is connected to the EIS tier and the Client tier.
  On BRM can support different platforms and their architectures at the same time in the Client tier (see figure 22)
- As the use of On BRM grows On BRM can be separated into parts (see figure 11). First the EJB tier can be placed on one machine where the application server resides and the Web tier on another machine where the Web server is. The Middle tier is therefore separated across two machines with the Back-end integration layer connected to the EIS Tier and the Business process template layer running on a Web server.
- Data can be cached in the Middle tier (see figure 28). Cached data is typically held for a temporary amount of time, and can be lost if the Middle tier is restarted. Examples of cached data is frequently used look-up tables e.g. ZIP codes. Instead of repeatedly asking a database to return a static table of valid ZIP codes, the Business template process layer may get it once during initialization and provide a static reference to a collection that all threads use.
- Intermediate data is cached data that must not be lost if the Middle tier is restarted. Shopping carts are an example of intermediate data because they are complex calculations that rely on large, expensive to re-compute data sources. If the system loses intermediate data it can either retrieve the lost data from the EIS tier or use cached persistence that restart the processing without disturbing the EIS tier. The persistent caches are cached data held in a database in the Middle tier. The last solution, placing persistent caches in the Middle tier, offloads demand from the EIS tier. Another reason for making Middle tier caches persistent is to reduce or eliminate the amount of time required to restart On BRM in the event of a failure. Even if best-of-breed application servers support fault-tolerant clusters, the possibility of a failure exists. By maintaining a persistent Middle tier cache, it is possible to restart On BRM from the cached data and prevent a flood of requests to the EIS tier when On BRM comes up again. Synchronization is directly related to the dynamic nature of cached data. Therefore it must be determined how often it is necessary to synchronize the cached data with the EIS tier, and this decision will influence the implementation of cache data.
- As the use of On BRM further grows and/or the business processes get more integrated between the internal and external users, On BRM can be further separated so the Business Process Template Layer runs on many local Web servers in local centers. The rest of the middle Tier still runs on a single machine.
- When an organization's use of On BRM becomes global the architecture may look as in figure 12. An extra benefit of having persistent caches is that now individual application server instances with On BRM's EJB tier can have in-memory caches that are subsets of the disk cache. This is an efficient design because the in-memory cache for each application server can be specified to a

certain use. For example, one application server might hold cached information for Denmark, another for Sweden and Norway. If Danish customer requests are routed to the Danish application server, Danish customers will see a decreased response time. If the number of Swedish customers rapidly increase, it is possible to start a new dedicated application server. This lets On BRM scale in the Middle tier and it will not affect the internal and external resources in the EIS tier. It is possible to make the caches persistent by providing Middle tier persistence by using On BRM's databases in the Middle tier. By using On BRM's databases in the Middle tier the cache implementation automatically gains the benefits of locking and data sharing across application server instances. Since On BRM uses entity beans it is easy to reuse the mapping code for those beans to manage the data's persistence in the cache.

- Many other architectural settings of On BRM are possible.

Cost and scalability

On BRM's scalability improves the use of external resources such as Web services in different ways. As many external resources' content is local for a country or region it gives no meaning to use them across these boundaries. Examples of this kind of external resources are government Web services, ZIP codes, maps, route schedules, and credit rating. Using On BRM an organization can connect into the appropriate local external resources in the regional centers through On BRM's Back-end integration layer running on a regional application server, while the HQ still runs the main version of On BRM. All this can be done with or without the Web services application stack. Of course On BRM still support an uneven development so the local external resources can be proprietary.

Even if letting the Client tier handle the local external resources looks attractive at first, this solution has a drawback. For thousands of clients represent a tremendous amount of duplication. If the clients use their own databases as caches the data will be replicated in thousands of places. Maintaining data integrity will be impossible so there will be inconsistency between the different databases. All the problems most organizations have experienced with decentralized departmental databases (e.g. the same customer in many databases) will be increased.

On BRM also offers efficient and adaptive collaboration with regional partners, customers and suppliers built on the same technology and architecture as mentioned above.

Since many external resources have a fee per CPU, in addition to a payload per message, it is cost effective to connect the external resource to one regional CPU instead of decentralising the CPUs in the Client tier. Using a centralistic approach also offers a possibility for a cost effective use of the external resources by only allowing use to the appropriate users.

By separating the servers running the Web tier and the EJB tier an organization can efficiently minimize the costs to:

- Standard software

- Databases
- Operating systems
- Hardware and operative systems costs
- Network.

Benefits of On BRM's architecture

Some of the benefits of On BRM's flexible architecture can be summarized:

- No bottlenecks to scaling. New servers can be easily and dynamically added to the configuration as necessary to meet new market demands, optimize business processes and deployment strategies. The overall request load can be optimally distributed among the servers running the Web tier and the EJB tier so that resources are fully utilized. Internal resources – legacy, department, and branch systems – and new external resources such as Web services and proprietary APIs can dynamically be added and integrated to support new business partners' participation and collaboration in the organization's business processes
- No single point of failure that could impact availability. Requests are automatically transferred from non-working machines to working machines (failover). User state is protected via caching to ensure that any failures that occur (e.g. a server crash) is fully transparent from the user and clients
- Transparency to the programmers. An organization's programmers should not have to deal with difficulties of replication, caching, request routing, load balancing, failover, components of the Web service stack and all components in the internal model. They should only deal with Business Delegates and WSDLs together with business applications in the Client tier and the EIS tier. This requires domain competence instead of detailed technical competence. They will be able to rapidly develop new features. Moreover the organization should be able to integrate and deploy commercial off-the-shelf Web services at small costs
- Single system image to the administrators. They are empowered to manage the clustered services and servers as a single logical resource. This simplifies operations and helps prevent any introduction of inconsistencies between peers in the cluster
- Personalization. Every user has a personalized view based upon the role and participation in a business process, language, and appropriate content that matches the user's needs and interests
- Low costs to change management. The training and change management costs will be reduced with a stepwise deployment of On BRM. The users will still use legacy systems, MS Office, a corporate portal or other standard software after a deployment of On BRM. They will all be seamlessly integrated with On BRM
- Hardware and operating system independence. Using J2EE based clustering, replicas can be established across disparate hardware and operating systems platforms. By not relying on specific platform features, an organization's investment is protected as it moves On BRM's components from one platform to another.

Security

As a frontline to protect the On BRM enterprise framework from direct exposure from an untrusted environment, On BRM supports DMZ (Demilitarized Zone) (see figure 13). It does this by isolating the On BRM from being directly accessed from other machines, applications or the Internet by adding a DMZ layer between the intranet running the On BRM and the extranet running the clients. Two firewalls make up the DMZ: one firewall in front of the DMZ and one behind it. The two firewalls separate the internal and external environment. In the DMZ it is possible to have Web services (WSDL and UDDI), JSP, On BRM Business Delegates, and non-sensitive data. The On BRM Business Delegates communicate with On BRM and sensitive data behind the second firewall through On BRM's Session Façade. Internally On BRM supports an XML firewall (see appendix M). Using at the same time WS-security, DMZ, and XML firewall, the security of On BRM is state-of-the-art.

Benefits of using On BRM

As mentioned in the Management summary above, On BRM is a framework for enterprise applications built to:

- Lever existing IT assets
- Enable responsive and efficient collaboration between external and internal business processes despite an uneven technological development
- Cover the entire global business process life cycle from end to end in a global world, but on a local basis
- Respond adaptively and rapidly to business changes
- Offer responsiveness with real-time visibility and access 24x7
- Provide scalability
- Provide state-of-the-art security.

Large corporations have made significant investments in three main areas of assets: business process modelling, workforce management and training, and IT system development and deployment. Because such investments can yield unprecedented returns when combined and utilized efficiently, On BRM is designed to support the execution of best-in-class business process models in collaboration with existing IT systems seamlessly directed by its users.

In order to successfully fulfil the seven main imperatives, On BRM offers a process-oriented architecture based on SOA, including the Web service stack and On BRM's components.

Based on the seven main imperatives and designed with the aid of well-known patterns On BRM ensures the following qualities:

- Levers IT assets by reusing existing applications
- Reduces time to market for new business opportunities
- Minimizes cost to change management

- Minimizes cost to data converting
- High reliability
- High level of security
- High performance
- Adaptive scalability
- Rapid integration with existing internal and external resources
- Reduction of expenses to external Web services and applications
- Easy adaptability of new technologies such as Web services
- Tiers and layers are easily changed or modified without impacting other parts of the system
- Easy maintenance
- Communication between the development team and users in standard vocabulary
- Low training costs for both developers and users
- Change of focus for programmers. From a detailed technical competence to domain competence
- Rapid development of new features
- Interface to all leading development tools.

Bibliography

Alur Deepak, John Crupi, and Dan Malks. Core J2EE Patterns: Best Practices and Design Strategies. Sun Microsystems Press (Prentice Hall), 2001.

Axis. http://ws.apache.org/axis/

Beck Kent. Extreme Programming Explained. Addison-Wesley, 2000

Beck Kent & Martin Fowler. Planning Extreme Programming. Addison-Wesley, 2001

Booch Grady, James Rumbaugh, and Ivar Jacobson. The Unified Modeling Language User Guide. Addison-Wesley, 1999

Bott Ed and Woody Leonhard. Using Microsoft Office XP, 2001

Brown Kyle et al. Enterprise Java Programming with IBM WebSphere. Addison-Wesley, 2001

Brown William H., Raphael C. Malveau, Hays W. "Skip" McCormick III, and Thomas J. Mowbray. AntiPatterns, 1998

Cockburn Alistar in Vlissides, Coplien, and Kerth (eds.). Pattern Languages of Program Design 2. Addison-Wesley, 1996

Cocoon http://cocoon.apache.org/

CORBA http://www.corba.org/

Date C. J. An Introduction to Database Systems. Addison-Wesley, 1979

Det offentliges it-arkitektur. Hvidbog.
http://www.oio.dk/index.php?o=bdb7047655b10902b3bcd7ac10bc3a76

Dettelback Bill. Scale J2EE Applications with Middle-Tier Caching. Advisor Magazine, 2002. http://websphere.advisor.com/doc/08848

Dettelback Bill. Take Your Middle-Tier Caches to the Next Level. Advisor Magazine, 2002. http://www.advisor.com/Articles.nsf/aid/DETTB02

Digital signatur. http://www.digitalsignatur.dk/visForside.asp?artikelID=588

Duffey Kevin et al. Professional JSP Site Design. Wrox, 2001

Java Authentication and Authorization Service (JAAS) http://java.sun.com/products/jaas/

Jeffries Ron, Ann Anderson, and Chet Hendrickson. Extreme Programming Installed, Addison-Wesley, 2001

Fowler Martin. Analysis Patterns. Addison-Wesley, 1997

Fowler Martin. Patterns of Enterprise Application Architecture. Addison-Wesley, 2003

Fowler Martin with Kendall Scott. UML Distilled. Addison-Wesley, 1997

Fowler Martin. Refactoring. Addison-Wesley, 2000

Gamma Erich, Richard Helm, Ralph Johnson, John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995

Infostrukturbasen. http://isb.oio.dk/info/

Gartner om Infostrukturbasen (Gartner about the Danish Infostructurebase) http://isb.oio.dk/info/news/gartner+on+isb.htm

Gates William H. Business@the speed of Thought. Warner Books, 1999.

Halter Steven and Steven Munroe. Enterprise Java Performance. Prentice Hall, 2001

Hanna Phil. JSP: The Complete Reference. Osborne/McGraw-Hill, 2001

Husted Ted, Cedric Durmoulin, George Franciscus, and David Winderfeldt. Struts in Action, Manning, 2003

Kerberos http://web.mit.edu/kerberos/www/

Krasner Glenn E. and Stephen T. Pope. A Cookbook for Using the Model-View Controller User Interface Paradigm in Smalltalk-80. Journal of Object-Oriented Programming, 1(3):26-49, August/September 1988.

Kuhn Thomas. The Structure of Scientific Revolution. Second Edition, Enlarged. The University of Chicago Press, 1962

Leymann Frank and Dieter Roller. Business Processes in a Web Services World. DeveloperWorks, 2002 http://www-106.ibm.com/developerworks/library/ws-bpelwp/

Lucene http://jakarta.apache.org/lucene/docs/index.html

Microsoft DNA http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndna/html/dnasolutions.asp

Myerson Judith. Web Service Architectures. Tect , 2002 http://www.webservicesarchitect.com/content/articles/myerson01.asp

OASIS http://www.oasis-open.org/home/index.php

PAM http://wwws.sun.com/software/solaris/pam/

Portlet specification JSR 168 http://www.jcp.org/en/jsr/detail?id=168

Rockwell Westy. XML, XLST, Java and JSP. New Riders, 2001

Roman Ed. Mastering Enterprise JavaBeans and the Java2 Platform, Enterprise Edition, Wiley, 2002

Samar Vipin. Making Login Services Independent of Authentication Technologies. http://java.sun.com/security/jaas/doc/pam.html

SAML http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security

Shin Sang. Secure Web services. JavaWorld, 2003. http://www.javaworld.com/javaworld/jw-03-2003/jw-0321-wssecurity.html

Struts http://jakarta.apache.org/struts

Struts and WebSphere Portal http://www7b.software.ibm.com/wsdd/techjournal/0303_hanis/hanis.html

Texel Putnam and Charles Williams. Use Cases Combined with Booch/OMT/UML. Prentice Hall, 1997

The Stencil Group. Web Services Rules: Real-World Lessons from Early Adopters, 2003 http://www.stencilgroup.com/ideas/reports/2003/wsrules/wsrules.pdf

The Stencil Group and A. T. Kearny. The emerging Web Services Market, 2002 http://www.stencilgroup.com/ideas_scope_200203atkws.html

The Stencil Group. The Laws of Evolution: A Pragmatic Analysis of the Emerging Web Services Market, 2002 http://www.stencilgroup.com/ideas_scope_200204evolution.html

Virdell Margie. Business Processes and workflow in the Web Services World, 2003 http://www-106.ibm.com/developerworks/webservices/library/ws-work.html

Vlissides John. Pattern Hatching, Addison-Wesley, 1998

Yuordon Edward & Larry Constantine. Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design. Prentice Hall, 1975

WebDav http://www.webdav.org/

WfMC http://www.wfmc.org/

W3C http://www.w3.org/

Wilkes Lawrence. The Web Services Protocol Stack http://roadmap.cbdiforum.com/reports/protocols/

WSRP http://xml.coverpages.org/ni2003-09-11-a.html

WSRP Specifications 1.0 http://xml.coverpages.org/WSRP-CSv10-Rev2.pdf

X.509 http://www.ietf.org/html.charters/pkix-charter.html

Appendix

Appendix A: SOA

Service Oriented Architecture (SOA) is the new wave in the IT architectures according to most experts. Each major shift in architecture has always been predicated by changes in the underlying capabilities and premises of the role of IT in the organizations. A Service-Oriented Architecture is a collection and combination of loosely coupled services that are distributed.

Through On BRM an organization can combine and consume the available services on an as-needed basis. Since many services still have proprietary protocols the organization needs a way to connect these services, and On BRM does just that. With On BRM the organization can harvest the power of combining existing and new services.

The communication between On BRM and the services can involve either simple data passing or it can involve two or more services coordinating some activity. This combination of services - internal and external to an organization - makes up a Service-Oriented Architecture.

Service-Oriented Architectures are not a new thing. For many people the first experiences of Service-Oriented Architecture in the past were with the use DCOM or Object Request Brokers (ORBs) based on the CORBA specification.

The major characteristics of the new architectural framework are:

- Distributed computing. Applications or functional elements of applications deployed on multiple sites collaborate across intranet or extranet as a single application making use of existing, ubiquitous protocols like HTTP. The focus is not the isolated database but the interface
- Loosely coupled interfaces. Traditional architecture for communication and connection depends upon a tight interconnection of all components. The complexity of these connections requires a thorough understanding of both ends of the connections, and it is difficult to replace one element with another. Most often the information is represented as coarse-grained transactions.
  Loosely couple systems that interact dynamically and asynchronously with one another via standard Internet technologies such as the Web service stack, require a much simpler understanding, and it is much easier to replace an element as the information most often is represented as a fine-grained transaction. The data formats are not binary and proprietary, but shared and broadly accepted as an open standard
- Open accepted standards. All major vendors and standards organizations support the open standards. For instance applications based upon .Net can seamlessly communicate with applications based upon J2EE. Even legacy applications can communicate with other applications by means of loosely coupled interfaces either through applications such as On BRM or Web services. Most previous attempts to do distributed computing have been based upon proprietary standards

- Process-oriented. The services are designed with a process-orientation. They are modelled for specific business processes and larger distributed transactions, and are not all-in-one applications with a standard user interface.

Appendix B: Service layer

The Service layer pattern (see figure 2) is used for organizing business logic (Alistair Cockburn). There are two kinds of business logic in On BRM: Domain logic and business process logic. Domain logic is purely about a specific domain such as credit rating, calculation of interest, and storing a record. The business objects contain the domain logic. The business process logic is about syntax validation and business process responsibilities such as notifying managers and administrators in the credit department and sales people about an organization's credit policy and credit rating in a specific scenario. The business process logic is formed by the Presentation layer using the Business process template layer. Sometimes business process logic is referred to as workflow (see appendix C). In On BRM these two kinds of business logic are factored into separate layers, yielding the usual benefits of layering and rendering the pure domain object classes more reusable. On BRM has implemented the Service layer in the Service layer pattern as a façade called the Business process template layer, where the facade does not implement any business logic. The Domain logic layer in On BRM implements the domain logic.
Enterprise applications typically require different interfaces to their data and domain logic. In figure 2 you see different kinds of interfaces: Data loaders, user interfaces, integration gateways, and others. Despite their different purposes, these interfaces often need common interactions with the application to access and manipulate its data and invoke its domain logic. The interactions may involve transactions across multiple resources and the coordination of several responses. Programming the interactions separately in each interface causes a lot of duplication. The use of the layer is a result of the wish to avoid duplication and promote reusability. For instance this, together with the wish to avoid the performance problems (owing to too many remote invocations), motivates the Session Façade in On BRM.

Appendix C: Workflow

In 1996, The Workflow Management Coalition (WfMC) defined a workflow as 'The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules'.

The most popular workflow systems currently available on the market have been developed over the past ten years using C and C++. They usually have hard-coded work management capabilities that are based on proprietary workflow execution models and APIs for task creation, assignment, acceptance, delegation, delaying, and fulfilment. They do not have as many layers as On BRM and they do not distinguish between domain logic and business process logic (see appendix B). Most often they are based on the simple and inexpedient three layer architecture (see appendix U). This increases development, deployment, and maintenance costs, as well as increasing the integration challenges.

Some workflow systems today offer interfaces to the J2EE platform, but they have not been re-written in Java in order to take full advantage of J2EE application servers (see appendix J). Therefore they are not capable of benefiting from the services offered by best-of-breed J2EE application servers such as load balancing, failover, scalability, distributed transactions, Web services support, and administration. Because of this, organizations that have standardized on the J2EE platform for the development of their enterprise applications usually find it difficult from an architectural perspective to deploy or maintain these workflow systems.

Process Orientation vs. Document- or task Orientation

Traditional workflow systems usually rely on a document-oriented model or a task-oriented model. Originally many of these products were document management systems or groupware systems designed to automate manual activities. This limits their relevance to a category of processes that can adapt either a document-oriented or task-oriented view of the world. This limitation becomes clear when business processes must be integrated with internal and external data sources, which are usually data source-centric rather than document-centric or task-oriented.

Some workflow systems have found ways to bypass such limitations, but usually at the expense of forcing one model onto another in a very abnormal fashion, leading to complexity from a development standpoint, and performance overhead from an execution standpoint.

On BRM removes these limitations by having a process-oriented architecture and a separation between business process logic and domain logic. Owing to this fact On BRM supports document-orientation and task-orientation through the use of dedicated document and task management processes, for instance using the project management's components.

Interoperability

Traditional workflow systems are usually based on proprietary workflow execution models. Even those that support the Workflow Reference Model defined by the WfMC, and therefore share the same definitions for workflow processes, participants, tasks, and documents, do not always share the same model from an execution standpoint. This is the result of workflow systems that have evolved before any standardization took place. The WfMC recently developed the XML Process Definition Language (XPDL), but only very few workflow products actually support this specification. That is why the WfMC now focuses on the definition of a Workflow Reference Model and interoperability protocols, such as Wf-XML, allowing one workflow system to interoperate with another without a common understanding of a workflow execution model. The Web service stack architecture facilitates this interoperability and with On BRM's components the interoperability can be further promoted.

Distributed Business Processes

As mentioned above workflow systems were developed for the automation of manual activities without any consideration of distributed business processes requiring distributed transactions. This limits the workflow's applicability to business processes that do not require the use of distributed transactions.
Even though some workflow systems have recently offered integration with transaction management systems and J2EE application servers through adapters, they require different tools and deployment runtime configurations to support distributed transactions (see figure 4). This leads to a split of the business process into workflow and transactions.

On BRM takes a different approach to distributed transactions in taking full advantage of J2EE application servers. In On BRM distributed transactions are considered a fundamental basis of business processes and their definition is a part of the business process definition. This leads to lower development, deployment, and maintenance costs and decreases the integration challenges. On BRM therefore offers a tailored solution adapted to the organizations' business processes that can accommodate process-driven task management, multiple interaction channels, customizable policies for user authentication and authorization, and integration with existing document management processes.

Appendix D: Axis

Axis is an open source implementation of SOAP. SOAP serves as a transport protocol for the services exposed through WSDL. Axis consists of several subsystems working together with the aim of separating responsibilities cleanly and making Axis modular. Properly layered subsystems enable parts of the system to be used without using the whole of it. Figure 7 shows the layering of Axis' subsystems. The lower layers are independent of the higher layers. The 'stacked' boxes represent mutually independent, although not necessarily mutually exclusive, alternatives. For example, the HTTP, SMTP, and JMS transports are independent of each other but may be combined and used together.

Appendix E: Kerberos

Kerberos is a network authentication protocol. Kerberos was developed by Massachusetts Institute of Technology (MIT) as a solution to network security problems. It provides strong authentication for client/server applications by using secret-key cryptography. A free of charge implementation of this protocol is available from MIT. Kerberos is also available in many commercial products such as Microsoft's Active Directory. Kerberos is under a copyright permission notice. MIT provides Kerberos in source form, so that anyone can look at it for examining its trustworthiness.
Kerberos protocol provides strong cryptography. Therefore a client can prove its identity to a server (and vice versa) across insecure networks. After a client and server have proved their identity, they can also encrypt all of their communications to assure privacy and data integrity during their business communication.

Appendix F: XML Encryption

The W3C is coordinating XML Encryption. The goal is to develop XML syntax for representing encrypted data and to establish procedures for encrypting and decrypting such data. Unlike SSL the only information that is encrypted is the data that need to be encrypted, for example, only the credit card information in a purchase order XML document.

Appendix G: SAML

SAML

The Security Assertion Markup Language (SAML) is an XML-based framework for exchanging security information. The framework handles three issues:

- Syntax and semantics of XML-encoded assertion messages
- Request and response protocols between requesting and asserting parties for exchanging security information
- Rules for using assertions in connection with standard transport and message frameworks.

The security information is expressed in assertions about an entity which is either a human or a piece of hardware and has an identity in a security domain. An example of an entity is a person, identified by an email address in a particular Internet DNS domain. Assertions can transfer information about:

- Authentication acts performed by an entity
- An entity's attributes
- Authorization decisions about whether en entity is allowed to access certain resources.

SAML assertions are XML documents containing security information and they are expressed as XML constructs which may be nested. Therefore a single assertion may contain several different internal statements about authentication, authorization, and attributes.
SAML assertions may be used in connection with long running transactions (see paragraph WS-Transaction).
Assertions are issued by SAML authorities, such as authentication authorities and attribute authorities.
SAML defines a protocol by which clients can request assertions from SAML authorities and get a response from them. This protocol consists of XML-based request and response message formats. These request and responses can be bound to many different underlying communications and transport protocols. For the moment the SAML specification group is defining a binding to SOAP over HTTP.
SAML authorities can be both producers and consumers of assertions because SAML authorities can use received input from other sources as output, when they are creating their responses.

SAML use-cases

The SAML specification group uses three use-case scenarios to show its tasks:

- Single sign-on
- Distributed transaction
- Authorization service.

Single sign-on

A user logs in to Cars.com and is authenticated (see figure 8). Later, the same user accesses Financing.com. Without single sign-on, the user would typically have to re-enter his user identity information to Financing.com. Using the SAML scheme Financing.com asks Cars.com if the user has already been authenticated by sending an SAML assertion request message. Cars.com then sends back an SAML assertion statement indicating that the user in fact has been authenticated. When Financing.com receives the SAML assertion statement, it allows the user to access its resources without asking the user to re-enter his identity information.

Distributed transaction

A user buys a car from Cars.com (see figure 9). The user then decides to get a car loan from Financing.com. When the user goes to Financing.com in order to get the loan, the user's profile such as name and address, which Cars.com has already collected, can be passed to Financing.com. The Protocol will be:

- Financing.com sends a SAML assertion request to Cars.com where it ask Cars.com to send a user profile information
- Cars.com sends all the user profile information it knows to Financing.com in SAML assertion statements.

Authorization service

An employee from Company.com wants to buy a company car from Cars.com, which has a company arrangement with Company.com about company cars (see figure 10). When Cars.com receives the purchase order from the employee it wants to know if the employee is authorized to complete the order. If so it wants to know the maximum amount limit he can spend. So when Cars.com receives a purchase order from the employee, it sends a SAML assertion request message to Company.com. Company.com then sends back an SAML assertion indicating whether the employee is allowed to order the car together with his maximum amount limit.

Appendix H: X.509

X.509 defines a framework that establishes the condition of authentication services, under a central control represented by a directory. It describes two levels of authentication:

- Simple authentication, using a password as a verification of a claimed identity
- Strong authentication, involving credentials by using cryptographic technologies.

The Danish digital signature project called OCES (Offentlige Certifikater til Elektronisk Service) supports X.509.

Appendix I: BPEL and BPML

BPML and BPEL share roots in Web services and take advantage of the same XML technologies (XPath, XSDL), and are both designed to integrate applications and partners. BPML makes use of WSCI.
BPEL replaces IBM WSFL and Microsoft XLANG by combining and extending the functions of these technologies. It has been submitted to OASIS with royalty-free terms. At the core of the BPEL process model is the notion of peer-to-peer interaction between services described in WSDL and both the process and its partners are modelled as WSDL services.

Appendix J: Application server

Today most new enterprise applications are built upon J2EE application servers. An application server is a component-based product that resides in the middle-tier of a server centric architecture or next to Web service communication layer in the Web service stack architecture. It provides middleware services for security and state maintenance, along with data access and persistence. On BRM communicates with the Application server layer over the Back-end integration layer. Application servers are typically used for complex transaction-based applications such as On BRM. To support high-end needs, an application server has built-in redundancy, monitors for high-availability, high-performance distributed application services, and support for complex database access. J2EE application servers offer load balancing, failover, scalability, distributed transactions, Web services support, and administration.
J2EE supports a multi-tier distributed architecture. A simple deployment of this architecture generally includes a Client Tier, a Middle Tier, and an EIS Tier (see figure 13). The Client Tier can be one or more thin clients and rich clients. The Middle Tier consists of the Web tier containing one or more Web servers and the EJB tier containing one or more EJB Servers. These servers are also called containers. The Enterprise Information System (EIS) tier is formed by internal resources such as the existing applications, files, databases, and external resources.
A more sophisticated deployment configuration may involve a cluster of application servers that collaborate for increased scalability and reliability (see figure 12).

Appendix K: Gateway

A Gateway is an object that encapsulates access to an external system or resource (Martin Fowler). Many external systems and resources are integrated with On BRM. When accessing these external systems, On BRM has to translate these systems' proprietary APIs to a simple class. In order to do this, On BRM wraps the proprietary API into a class whose interface looks like a regular object. In this way the proprietary code is not spread through the whole On BRM. The API may also be a standard interface, such as WSDL or J2CA, which is much simpler for On BRM to handle.

One of the benefits of using this pattern is that when changes occur in the external resource, only the gateway will have to be modified. After the external service is incorporated into On BRM and processed in the business objects, it can be published in the usual way through a Business Delegate and WSDL. Therefore no clients using On BRM have to know any proprietary formats or interfaces of external resources; they just access On BRM's standard interfaces and retrieve data in a standard coherent format. Note the use of the Gateway pattern internally in the system and the Business process template layer to provide an external façade to the services (via the Business Delegates and the WSDL) in figure 1.

Appendix L: XML digital signature

XML digital signature provides authentication, data integrity, and non-repudiation. Non-repudiation is critical for both Web services and any other business transaction because non-repudiation means that a communicating entity can prove that the other party has performed a particular transaction. For example, if a financing company received a car transaction order from one of its clients and performed the transaction on behalf of that client, the financing company can prove that it completed that transaction to an arbitration committee if a dispute arises. Often a timestamp for a transaction is needed if a dispute arises. TDC will in a short time offer timestamps along with digital signature in Denmark. Currently the XML digital signature initiative is at the front of all XML-based security initiatives. The IETF (Internet Engineering Task Force) and The W3C (World Wide Web Consortium) jointly coordinate this initiative. The initiative's goal is to develop XML syntax for representing digital signatures for any data type. The specification also defines procedures for computing and verifying digital signatures. The XML digital signature initiative also addresses the canonicalization of XML documents. The purpose of canonicalization is to enable the generation of identical message digests and thereby identical digital signatures for XML documents which are equivalent in content but different in appearance. For example, a different number of carriage returns in two documents will make the documents different in appearance. XML digital signature also provides means of supporting different distributed transaction models. For example, it is possible to sign individual items or multiple items of an XML document. The signed document can be local or remote as long as the document can be referenced through a URI (Uniform Resource Identifier). A signature can be either embedded in a document (enveloped) or reside outside the document (enveloping).

XML digital signature also allows multiple signing levels for the same content e.g. signed, witnessed, co-signed, and notarized by different people.

Appendix M: XML Firewall

An XML firewall is an application-level firewall. It inspects SOAP headers and XML content, and only allows authorized content to get through. This is in contrast to a traditional firewall, which offers protection at the packet level. Some XML firewalls are also able to map SAML security attributes between different security domains. XML is easier to forge than traditional binary protocols, and the XML firewall helps prevent internal users from getting unauthorized access to data and systems.

Appendix N: Transaction Script

A Transaction Script (Martin Fowler) organizes all business logic (that is both business process logic and domain logic) as a single procedure, making calls directly to the database or through a thin database wrapper. The business logic is primarily organized by the transactions which the system performs. The Transaction Script's strength is its simplicity (see figure 24). As the business logic gets more complicated it gets both more and more difficult to keep it in a well-designed state and to maintain it. Often the code will be duplicated because it is used in other transactions. For these reasons On BRM does not use Transaction Scripts.

Appendix O: The Blob antipattern

The Blob antipattern (W. H. Brown et. al.) is found in systems where one class monopolizes the processing of data. A single large complex controller class surrounded by simple data classes characterizes this antipattern. The Blob class is typically too complex for testing and reuse, making the system hard or impossible to extend, and it typically makes maintenance very difficult.
The Blob class often includes unnecessary code that is no longer used.

Appendix P: WebDAV

The WebDAV protocol defines the HTTP extensions that enables distributed web authoring tools. It is an extension of the HTTP/1.1 protocol and adds new HTTP methods and headers. WebDAV specifies how to use the new extensions, how to format request and response bodies. Major features are:

- Locking (concurrency control): long running exclusive and shared write locks prevent the overwrite problem, where two or more collaborators write to the same resource. To achieve robust Internet collaboration, where network connections may be disconnected arbitrarily, the duration of WebDAV locks is independent of any individual network connection.
- Properties: XML properties provide storage for metadata, for instance a list of authors on Web resources. The WebDAV protocol called DASL (DAV Search and Locate protocol) can efficiently retrieve, set, and delete these properties. DASL also enables searches based on property values to locate Web resources.
- Namespace manipulation: WebDAV supports operations to copy and move a resource. Collections, similar to file system directories, may be created and listed.

IETF (Internet Engineering Task Force) approved it as a specification in 1998. Since then it has been deployed widely on multiple platforms and in applications from many vendors. WebDAV can be found in Web servers such as Apache and Microsoft Internet Information Server. It is also supported by leading document and content management vendors such as Documentum.

Appendix Q: Portlet API

The Portlet Specification, defined in Java Specification Request 168 (JSR 168), standardizes how components for portal servers should be developed. This standard has

industry backing from major portal server vendors such as BEA, IBM, SAS Institute, and Sun.

Portlets are web components like Java servlets, but they are specially designed to exist in the context of a composite page. Each portlet produces a fragment of a page which can be combined with fragments of other portlets, all within the portal page. Portlets rely on a portlet container infrastructure. This container is accessible via the Portlet API which is an interface to functions providing security, user customization and layout management. Via the Portlet API portlets can access functionality such as:

- Access to user profile information for the current user
- Participation in the portal window and action event model
- Access to web client information
- Sharing of information with other portlets
- A standard way of storing and retrieving per-user/per-instance portlet data persistently.

The Portlet specification is designed leveraging the following technologies: XML, JAXP, Servlet/JSP, JAAS and other J2EE technologies.

Appendix R: WSRP

The Web Services for Remote Portlets Specification is a joint effort of two OASIS technical committees: Web Services for Interactive Applications (WSIA) and Web Services for Remote Portals (WSRP).

The aim of this specification is to simplify the efforts of integration to portals through a standard set of web service interfaces. These interfaces should allow portals to quickly exploit new web services as they become available. The interfaces are defined using WSDL. It thereby standardizes the way Web services are integrated into portal front ends and the way in which content providers can write Web services for portals. Portals supporting WSRP can therefore seamlessly consume On BRM's Web services published in the Business process template layer. It also allows maximum reuse of presentation-oriented, interactive web services processed through On BRM thereby allowing the applications to access a much richer set of standardized web services. WSRP lies on top of the existing web services stack.

Appendix S: Struts

The core of the Struts framework is a flexible controller layer based on standard technologies like Java Servlets, JavaBeans, Resource Bundles, and XML, as well as various Jakarta Commons packages. Struts supports application architectures based on the Model 2 approach (see figure 3), a variation of the classic Model-View-Controller (MVC) design paradigm.

Struts provides its own Controller component and integrates with other technologies to provide the Model and the View. On BRM's Business Delegates represent the Model for Struts. For the View, Struts works well with Java Server Pages (JSP), including JSTL and JSF, as well as Velocity Templates, XSLT, and other presentation systems.

The Controller

The Controller is an event handler. Requests from either client or the model are processed with respect to the general application scheme as well as the basic validation rules implemented in each single control unit, called an Action. Up front validation ensures minimal network traffic and fast and friendly dialogue with users. All Actions are tied to a single servlet, called ActionServlet, which is the core of the Controller layer. The ActionServlet lives throughout the entire application cycle, and is responsible for calling the correct Actions and passing responses to the View layer.

The View

The View layer renders the actual response according to the Controller layer's decisions. Each type of response is encapsulated in a unit, called an ActionForm.

The output from ActionForms can be presented in multiple formats, depending on the implementation of the view. Traditionally, the view layer generates HTML pages by the use of TagLibs, JSP templates and predefined resources, which makes it easy to implement for example internationalisation. A large number of technologies such as PDF files, MS Excel files, WML are easily implemented using the JSP pages by generating XML, and parsing the result through Apache Cocoon (see appendix T).

New Views can be developed without much Java expertise, due to the fact that the View layer can produce among other things, XML, HTML, WML (Wireless Markup Language), PDF, and so on. Addition of new Views and screen layout are thus turned into technologically trivial tasks. This enables the use of technologies such as mobile phones, browsers, Web services, Windows, 3270 displays, etc. The On BRM Framework will handle all the necessary details behind the scenes and will allow the organization to fully concentrate on its business. On BRM contains several hundreds pre-written Views.

Struts is part of the Apache Jakarta Project, sponsored by the Apache Software Foundation.

Appendix T: Cocoon

Apache Cocoon is a web development framework that has been designed to coexist and interoperate side-by-side with existing J2EE solutions or to give them new functionality without requiring any change in the existing infrastructure. It is an XML publishing framework for processing content that adds XML and XSLT technologies to server applications. It provides content delivery in different formats such as HTML, WML, PDF, SVG, and RTF, to name just a few.

Cocoon interacts with many data sources, including filesystems, RDBMS, LDAP, native XML databases, SAP systems and network-based data sources.

Appendix U: Three layer pattern

Many applications use a three layer pattern. The three layers are: Presentation, business, and data access (see figure 27). All three layers operate on record sets that result from SQL queries issued by the data access layer. This records set is looked upon as Data Transfer Objects (see appendix Y) between the tree layers. This gives a strong coupling as both the business and the Presentation layer know about the database. This strong coupling makes it difficult to change the business process logic and the domain logic. Many tools like Visual Basic and Delphi use this pattern in order to allow the Presentation layer to use data-aware GUI controls. In this case the Business layer is structured as Table Modules (Martin Fowler). A Table Module is a single instance that handles the domain logic for all rows in a database or view. Many systems divide the Presentation layer into two new layers in order to support both HTML and Windows. Sometimes these systems use adapters in connection with the Data Access layer to avoid dependence to a single database vendor. On BRM supports a three layer pattern as you can see in figure 23. But On BRM contains many layers inside both the Presentation layer, the Business layer, and the Data source layer and On BRM's architecture therefore offers modularization and avoids coupling.
On BRM does not use the Table Module pattern. If On BRM did use the Table Module pattern, it would for instance not be possible for On BRM to be as scalable as it is. Microsoft calls its version of the three layer pattern DNA (Distributed interNet Applications Architecture).

Appendix V: Client and server session state

The Client session state

The Client Session State pattern stores the session state on the client (Martin Fowler). Even the most server-oriented system needs a session ID on the client to manage and identify the state of the specific client.
On BRM supports two architectures:

- For rich clients the Business Process Template Layer in On BRM's Presentation layer sends a data transfer object (see appendix Y) to the client.
- For HTML the web client holds a session ID used by the web server/servlet engine to identify the server side session object. This ID may be stored in a cookie or as a parameter to the URL, depending on the configuration of the web server and whether the user allows cookies. This is part of the servlet specification, and is handled automatically by the servlet engine.

The Server session state

The Server Session State pattern keeps the session state on a server system in a serialized form. The server side servlet HTTP session object holds the volatile session state, such as any unsaved data. Many J2EE based application servers such as WebSphere and WebLogic implements a shared HTTP session capability that allows storing the object in some persistent form (XML, database etc.) between user requests. This can be used to enable load balancing and failover clustering. On BRM can handle persistence of session objects internally if it is not supported by the implementation of the application server.

Appendix W: Domain Model

The Domain Model pattern (Martin Fowler) is an object model of the domain that incorporates both behaviour and data. The Domain Model should have the minimum coupling to other layers in the system, to enhance the ability to modify, build and test this layer. In On BRM we use a rich Domain Model, which mean that the objects in the Domain Model do not directly map to tables in a database. The rich Domain Model maps more complex business objects with inheritance as a grid of small interconnected objects.

Appendix X: The Company Business Delegate

The Business Delegates are designed as an application of the Business Delegate Pattern and the Data Transfer Object Pattern.

Table 1 shows a short description of some Business Delegates in On BRM:

Company	The Company Business Delegate implements a subtype of the Organization Business Delegate. It handles changes to the company, company structure and employee information. The Organization business object in the business tier offers very complex company structures. A company can for instance be a supplier, a partner and a customer, or even your own company at the same time.
ContactPerson	Subtype of the Person Business Delegate. It offers access to persons who have roles as contact persons for companies. This object handles changes to a contact person. Contact persons roles are much like employees roles (see below) but without information about job description or salary.
Employee	Subtype of the Person Business Delegate. An Employee is a subtype of a Person with attributes such as job, job description, job level, and competencies.
Person	The Person Business Delegate provides access to the elementary person entity. This object handles changes to persons, their social security numbers, names and aliases, addresses, emails, marital status, citizenship, housing, etc. A person object can also be accessed through its roles where it has additional information (such as an employee role with job description and salary information). A multi-layered manager/subordinate relationship provides easy creation of organizational hierarchies among the different roles such as employees and contacts.
Order	It is possible to create, update, validate or otherwise access orders through the Order Business Delegate. Orders can be pipelined - from prospect to the final accepted order. Orders can be used in the traditional sense, or can be used by consultant companies to manage their time-reports.
Organization	The Organization Business Delegate provides access to the organization object in the business tier. This Business Delegate provides methods for handling of changes to organization data such as business codes (Danish

	"CVR nr."), name and aliases, addresses. A multi-layered parent/child relationship provides easy creation of entities such as departments or subsidiary companies, even across countries.
Product	The Product Business Delegate provides access to the product object in the business tier. The Business Delegate provides methods for handling product data, including product categories, names, product numbers and product descriptions. A multi-layered product/part relationship provides easy creation of complex products.
Etc…	Several hundred others.

Table 1: Business Delegates examples.

The Business Delegates can easily be tailored to provide more fine-grained data, or to provide the data in different forms such as XML. Figure 20 displays how the On BRM Framework provides a Business Delegate for an elementary person as well as delegates for a person in certain roles (such as employees, contact persons, tax payers). The coarse-grained Business Delegates can be used through inheritance to provide a more fine-grained Business Delegate (see EmployeeDelegate and ContactDelegate in figure 20). These subtypes of the main Business Delegates can be provided by the On BRM Framework, or by an organization's own programmers. The Business Delegates can gather and combine methods from many Session Facades in the EJB tier into one object.

As it is difficult to transfer complex data types between a Java environment and a .Net environment using SOAP, Business Delegates can provide a simpler interface for the .Net environment by converting the Data Transfer Objects to simple objects such as strings. This allows On BRM to be connected with both Java and .Net Web services clients. On BRM also use IIOP and Janeva to solve the problems of incompatibilities between .Net and J2EE (see appendix AA).

Using JBuilder, WebSphere Studio Application Developer or any other modern Java development environment WSDLs can easily be generated using the Business Delegates as templates. In this way On BRM provides an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information for .Net applications, J2EE application, or transaction managers.

An example of a Business Delegate

In table 2 the CompanyDelegate is used as an example of a Business Delegate:

CompanyDTO getCompany(long id)	Will retrieve a Data Transfer Object representing information of the company specified by the parameter.
void setCompany(CompanyDTO value)	Will update information of a given company according to the values specified by the parameter.

List getChildCompanies()	Will retrieve a list of the child companies one level below.
List getParentCompanies()	Will retrieve a list of the parent companies one level above.
List getCompanyList(String search)	Will retrieve a list of companies containing the search string as a sub string of the company name.
Void deleteCompany(CompanyDTO value)	Deletes a company.

Table 2: The CompanyDelegate

Business Delegates send and receive data as Data Transfer Objects (see appendix Y). The CompanyDelegates' Data Transfer Objects are objects that carry data between the CompanyDelegate and the client. It holds all the data for the call. The fields in the CompanyDataTransferObject are fairly simple, being primitives and simple classes like strings. They represent the most commonly used values for a company. A part of the CompanyDataTransferObject is shown in table 3. Note how On BRM is UDDI-prepared.

long id	
String name	
AddressDataTransferObject list addresses	List of addresses for the company
String authorizedName	According to the UDDI-specification
String businessCode	According to the UDDI-specification
String businessType	
String operator	

Table 3: The CompanyDataTransferObject.

Appendix Y: Data Transfer Object

A Data Transfer Object is a simple object for carrying data that needs to be transferred across a process or network boundary. It contains no logic and limits its behavior to activities such as internal consistency checking. In On BRM they are used for transporting data between the business objects and in connection with the Session Façade and the Business Delegates which are Remote Facades (see appendix Z). When dealing with remote calls the aim is to reduce the number of calls. Aggregating data into an object can replace multiple remote calls with one remote call.

Data Transfer Objects were previously called Value Objects in Alur et al., but Value Objects are something slightly different (see Martin Fowler).

Appendix Z: Remote Facade

Remotes Facades provides a coarse grained façade on fine grained objects to improve efficiency over a network (Erich Gamma et al). Remote Facades handle the distribution problem of separating distinct responsibilities into different objects. As fine grained objects are the best solution for complex logic, On BRM's complex logic is placed in fine

grained objects. For efficient access to them On BRM uses Remote Facades such Session Façades and Business Delegates.

Appendix AA: SOAP and Complex data types

The SOAP definition does not specify the encoding of complex data types (records) as parameters or results of the request. This is handled differently by Java and .NET implementations. Often the solution of this problem is encoding the data to strings at the sender and decoding the strings at the recipient. On BRM supports this solution. This solution however requires a specialized client library used in the .NET clients. The developer of the client application develops to an API that hides all the decoding and encoding details. Another more neat solution is to generate a .NET interface using tools such as Borlands Janeva (see figure 29), which connects the client to On BRM using the secure and scalable Internet Inter-ORB Protocol (IIOP). On BRM supports the Janeva IIOP connection.

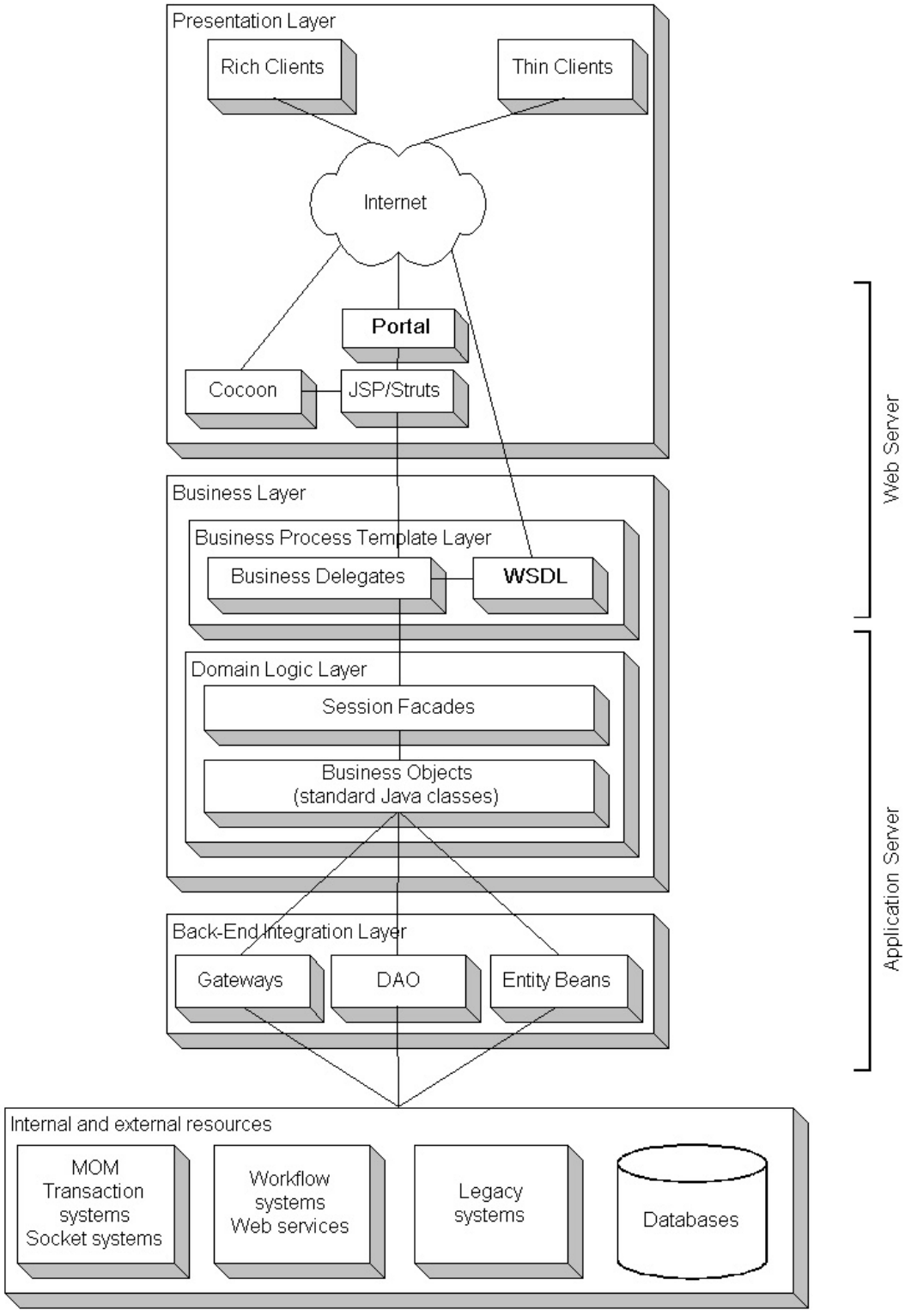

Figure 1: The On BRM Framework and its surroundings.

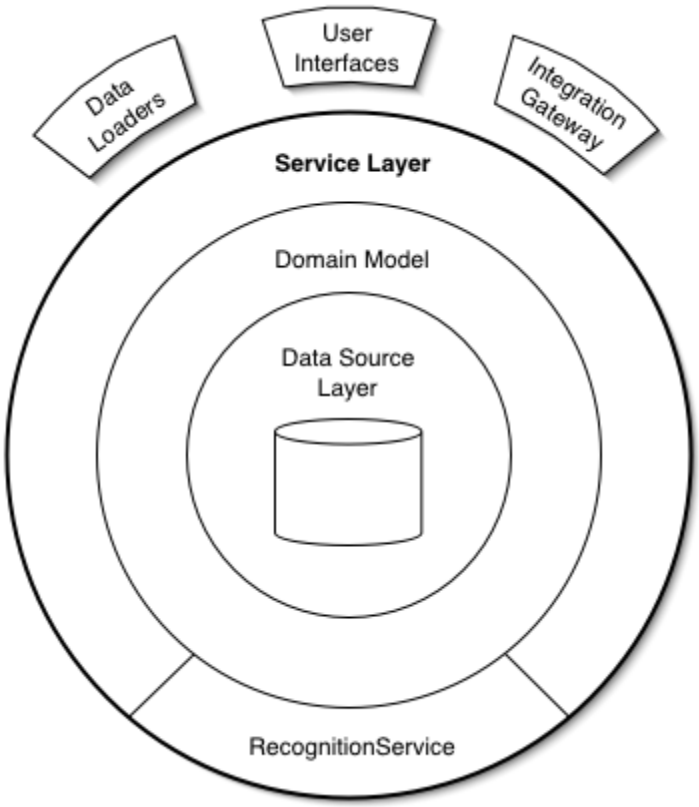

Figure 2: Service Layer Pattern

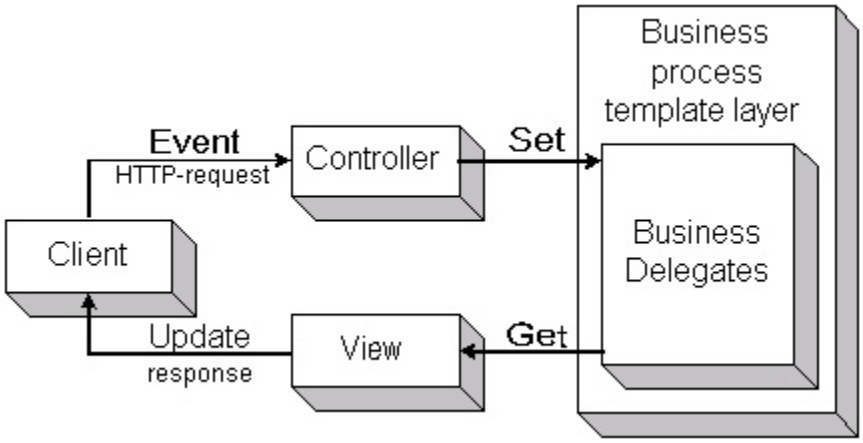

Figure 3: The Presys MVC Model 2.

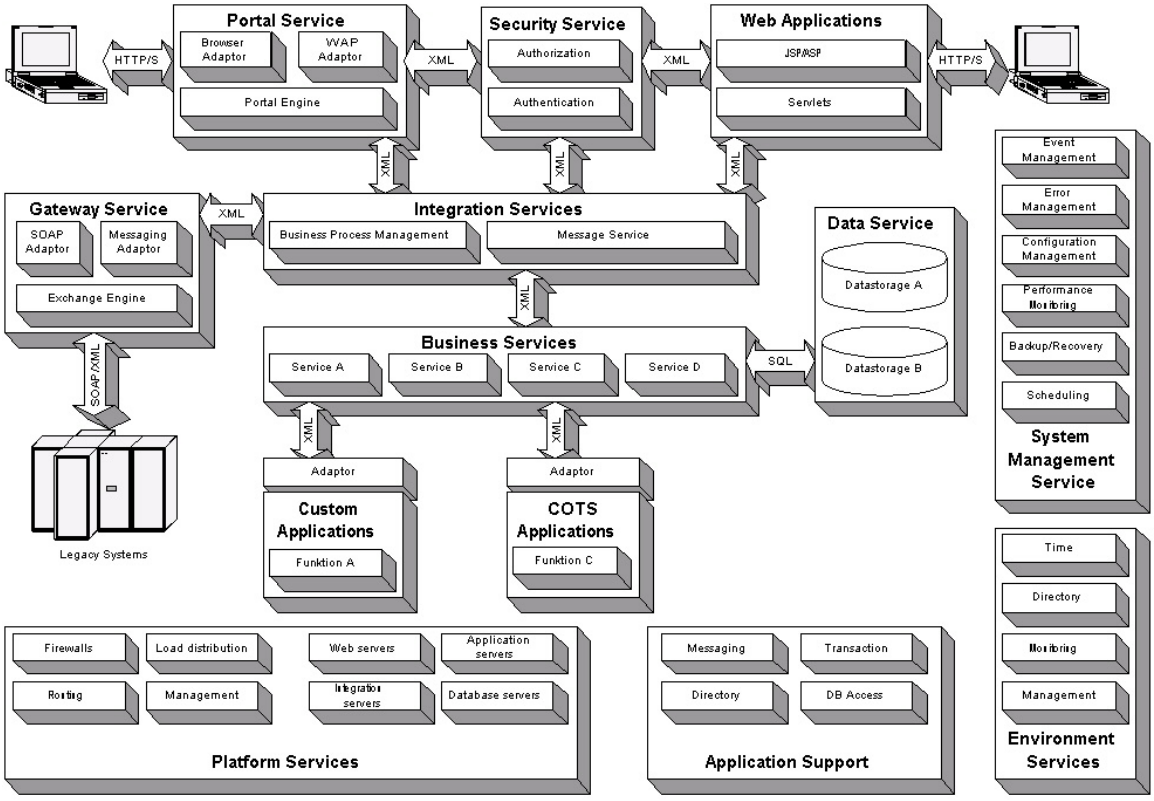

Figure 4: Traditional infrastructure

COTS Applications: Commercial off the shelf applications

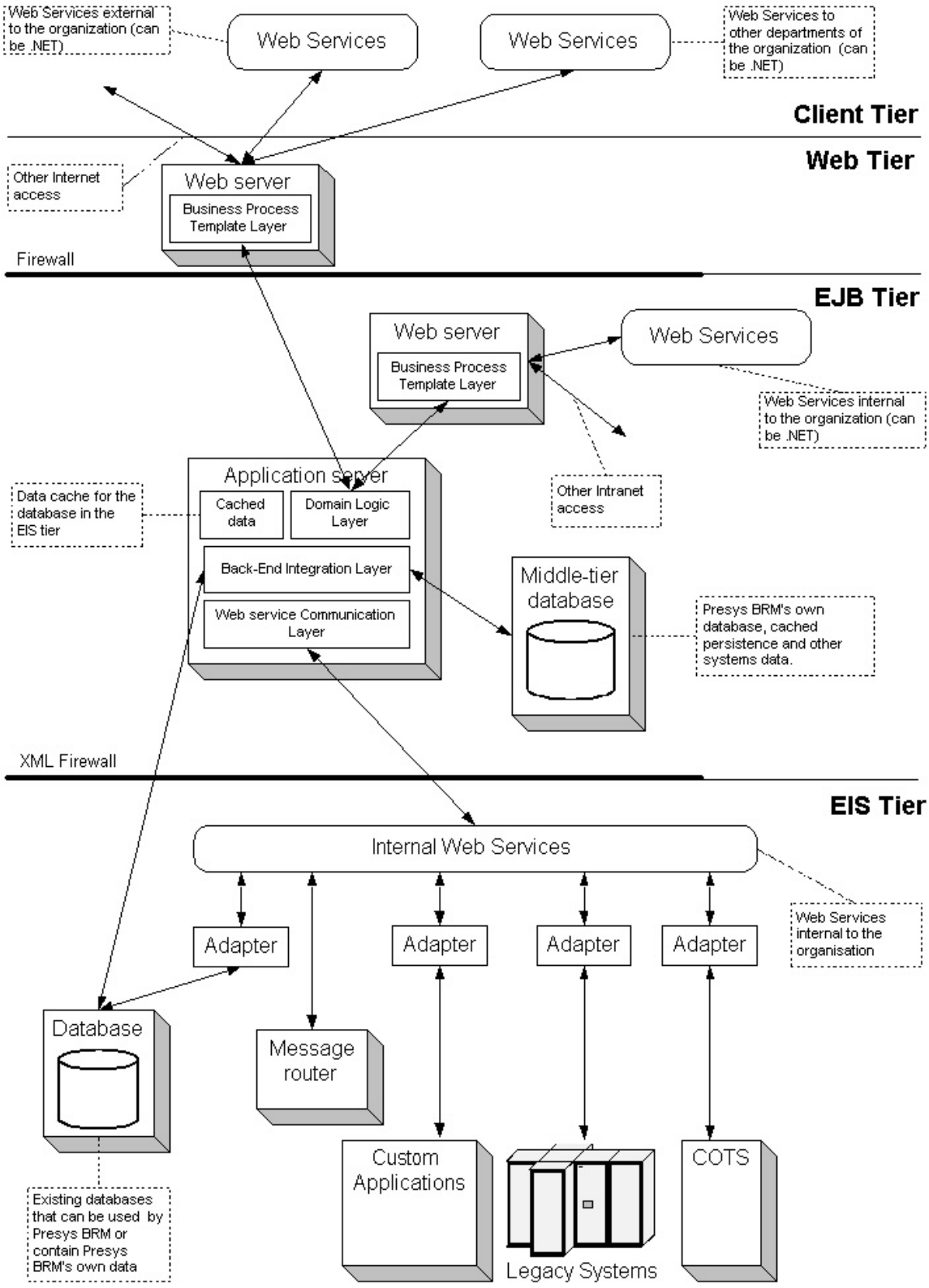

Figure 5: Integration with On BRM using the Web service stack

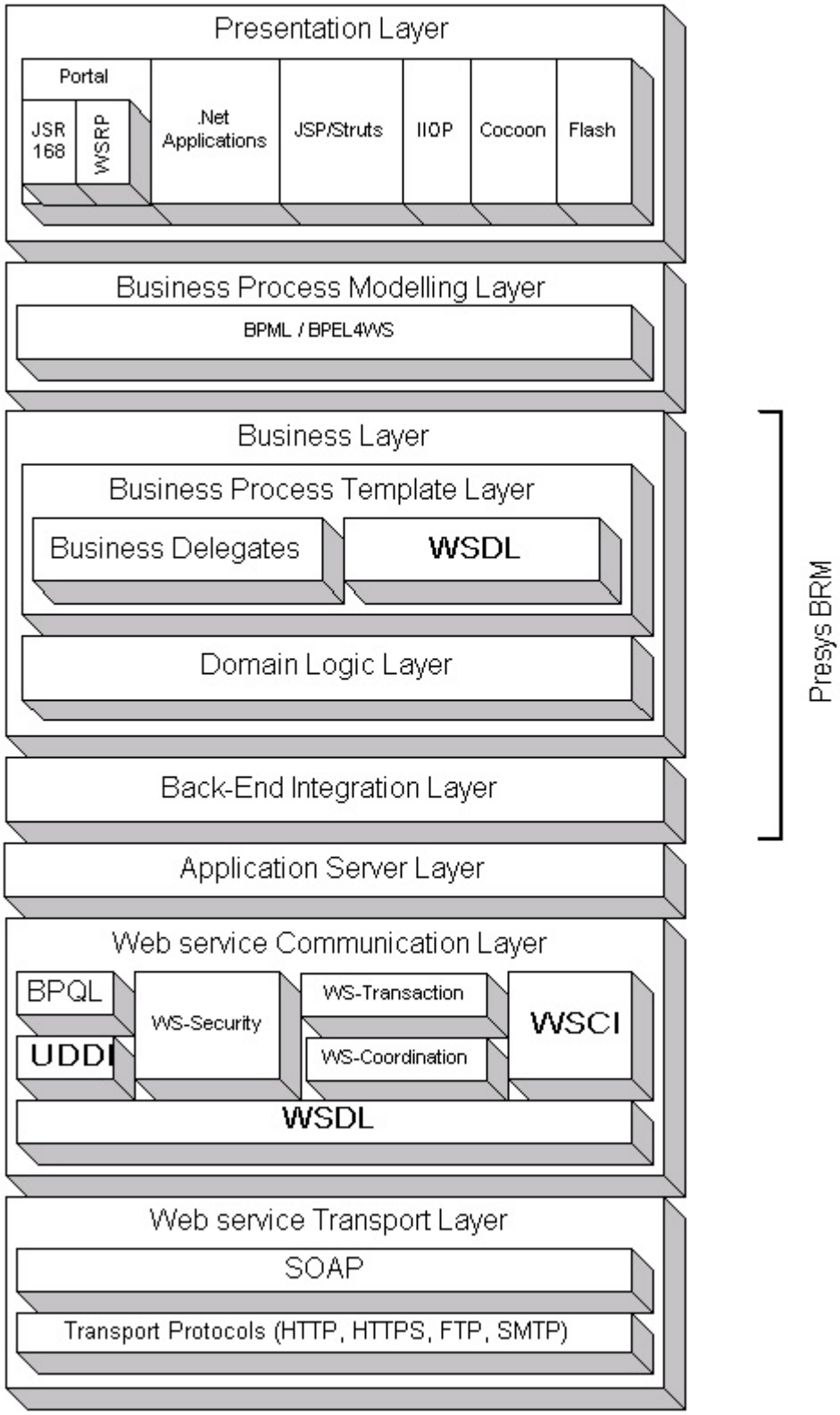

Figure 6: Web service stack including On BRM

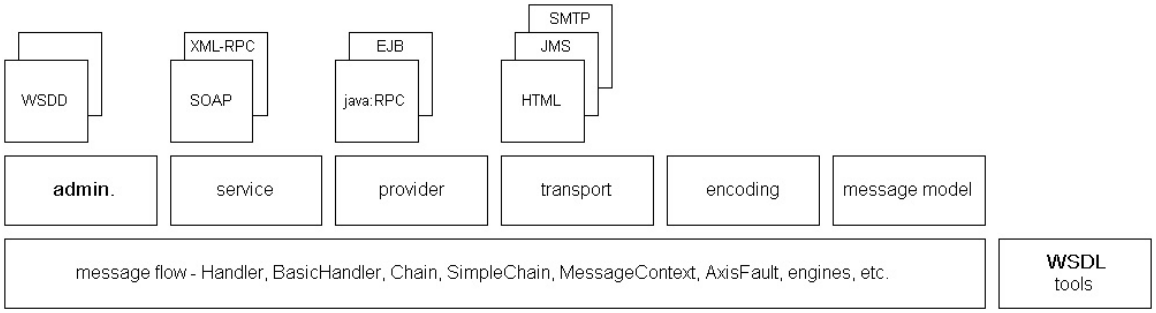

Figure 7: The layering of Axis' subsystems

WSDD: Web Service Deployment Descriptor
XML-RPC: XML-Remote Procedure Calling
java:RPC: Java Remote Procedure Calling
JMS: Java Messenger Service
SMTP: Simple Mail Transfer Protocol

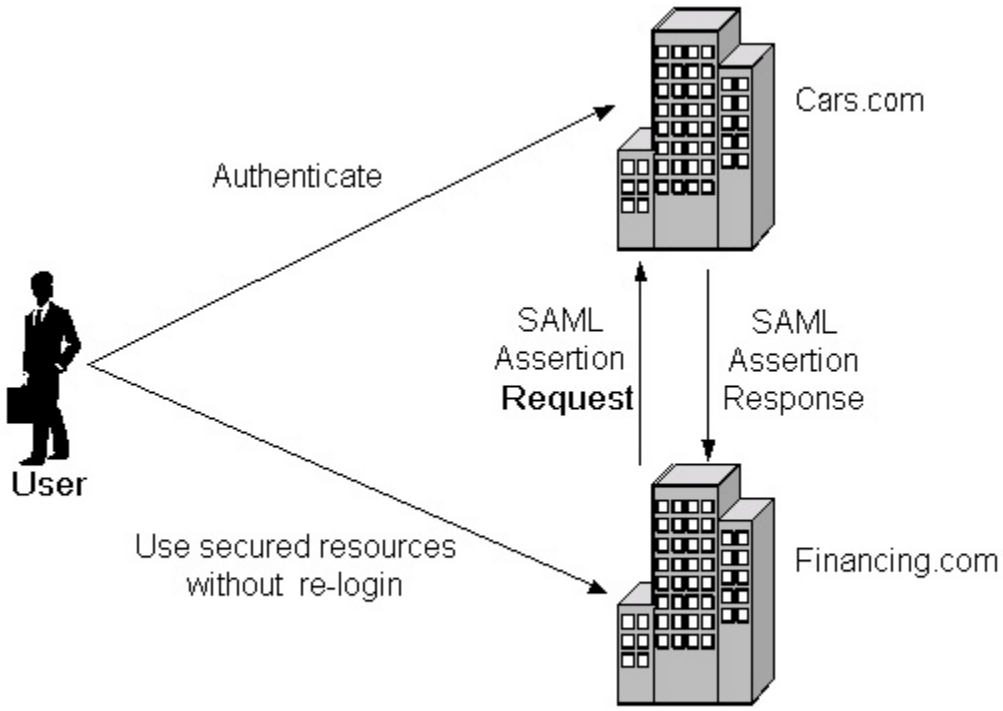

Figure 8: Single sign-on

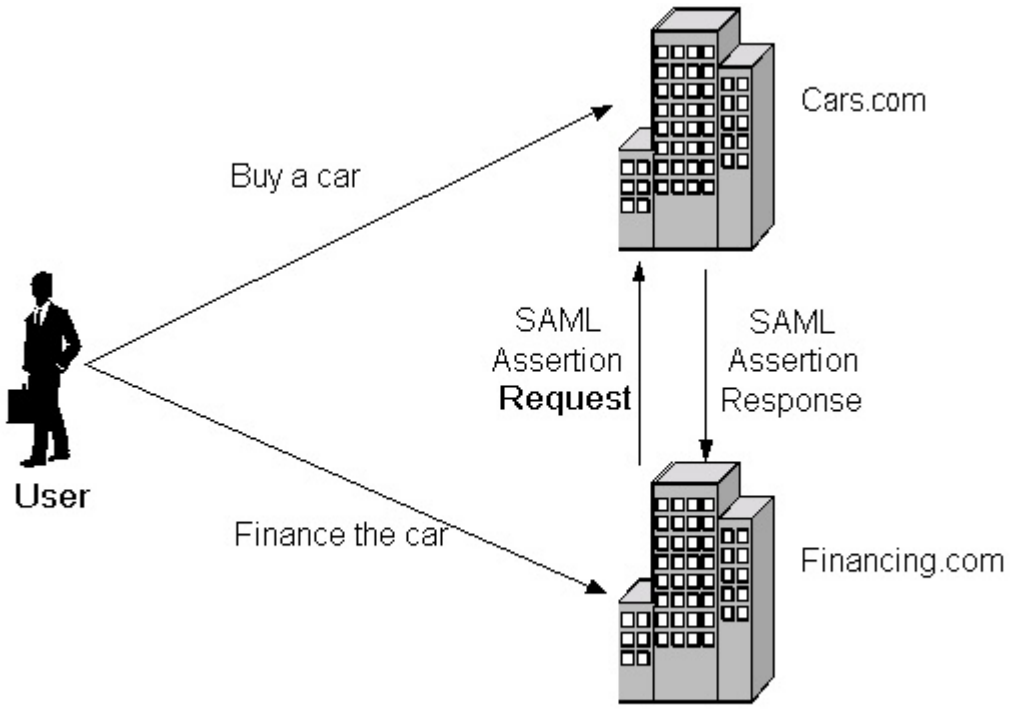

Figure 9: Distributed transaction

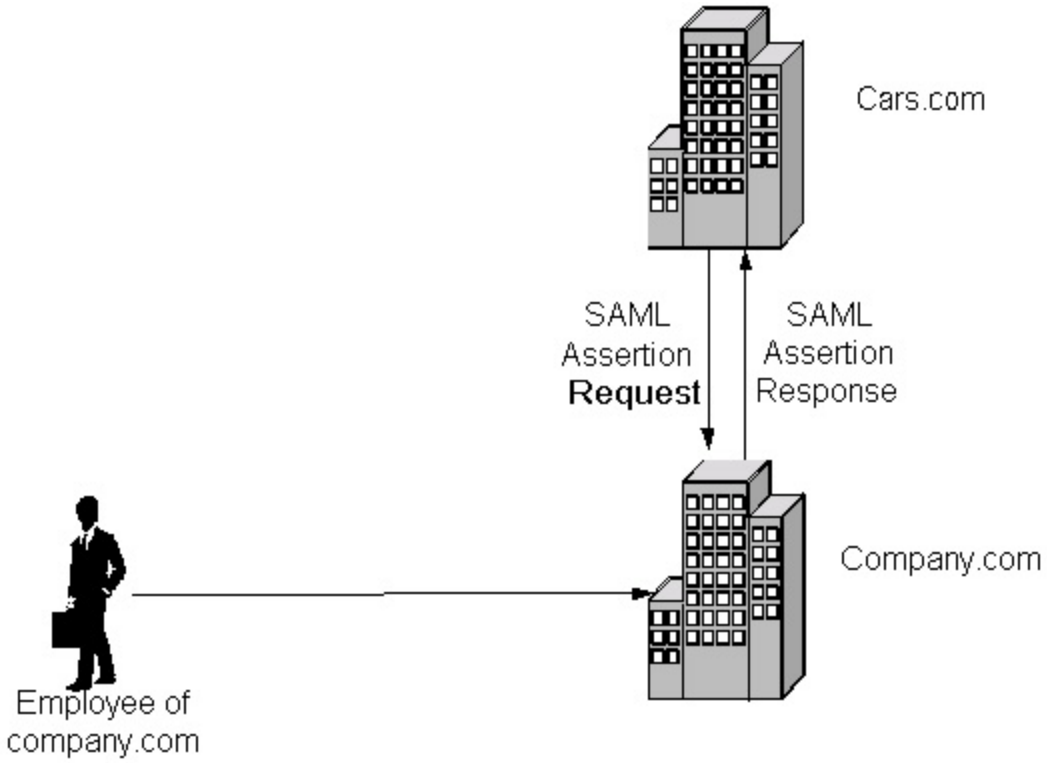

Figure 10: Authorization service

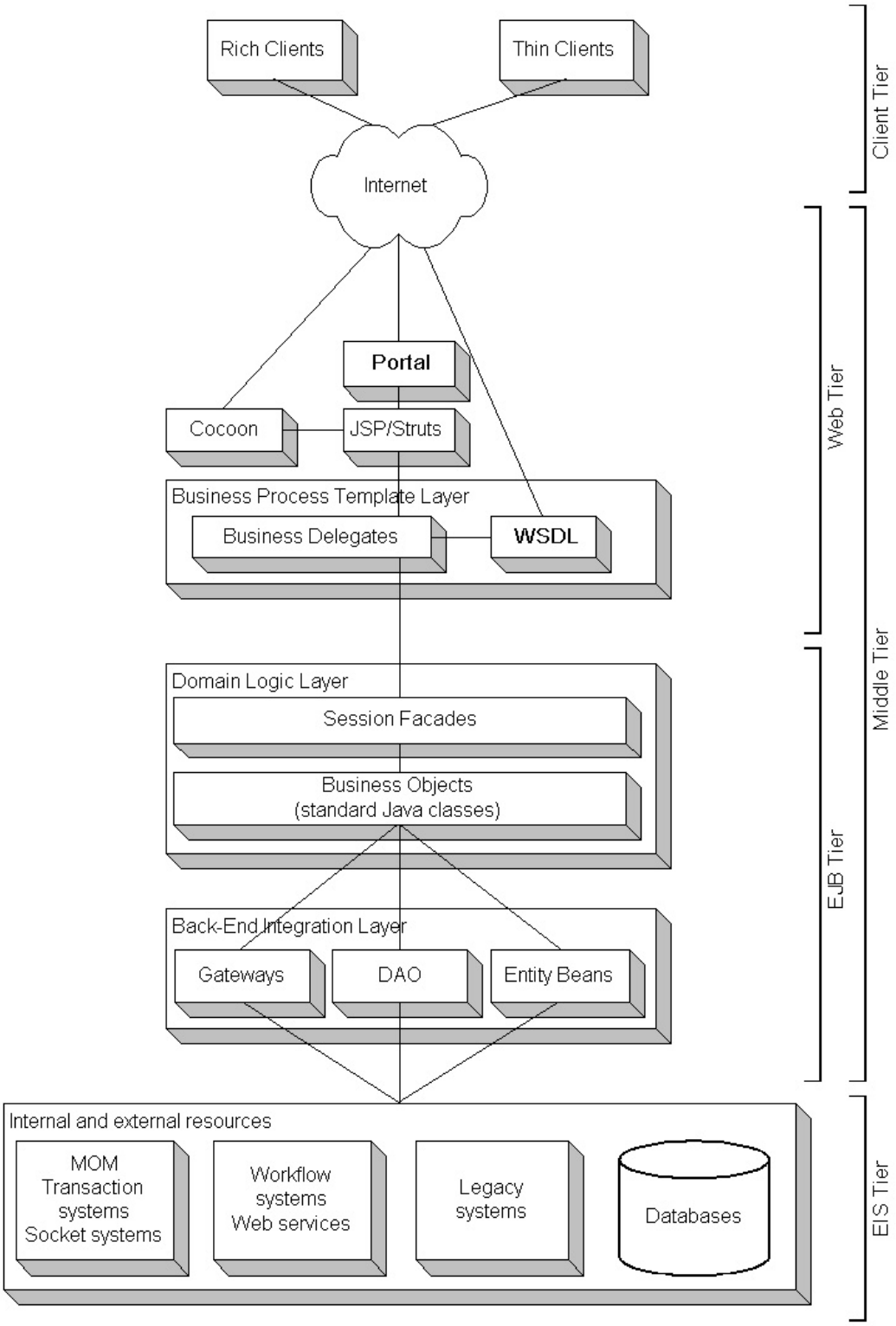

Figure 11: Tiers in On BRM

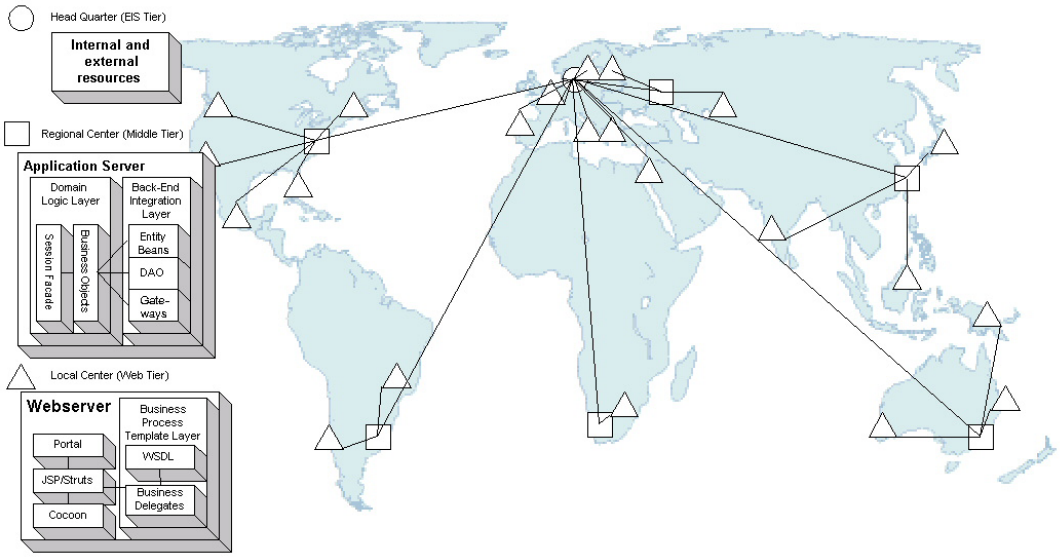

Figure 12: On BRM based on a distributed architecture

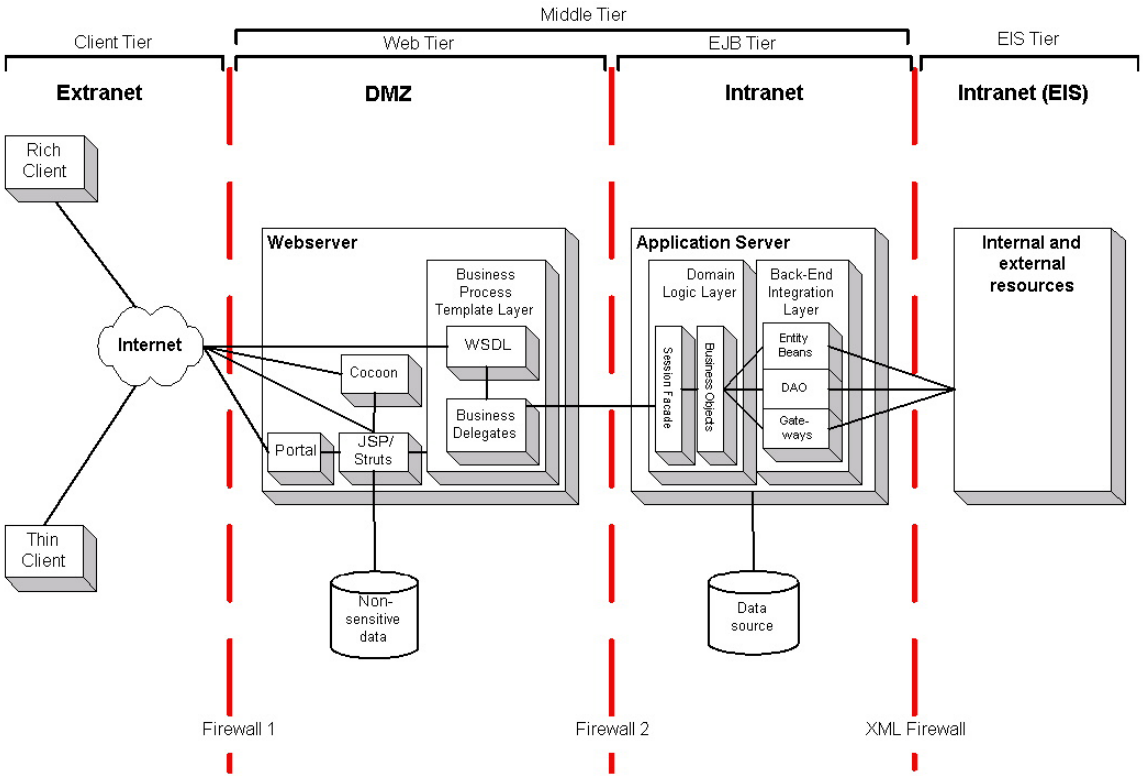

Figure 13: Presys and DMZ

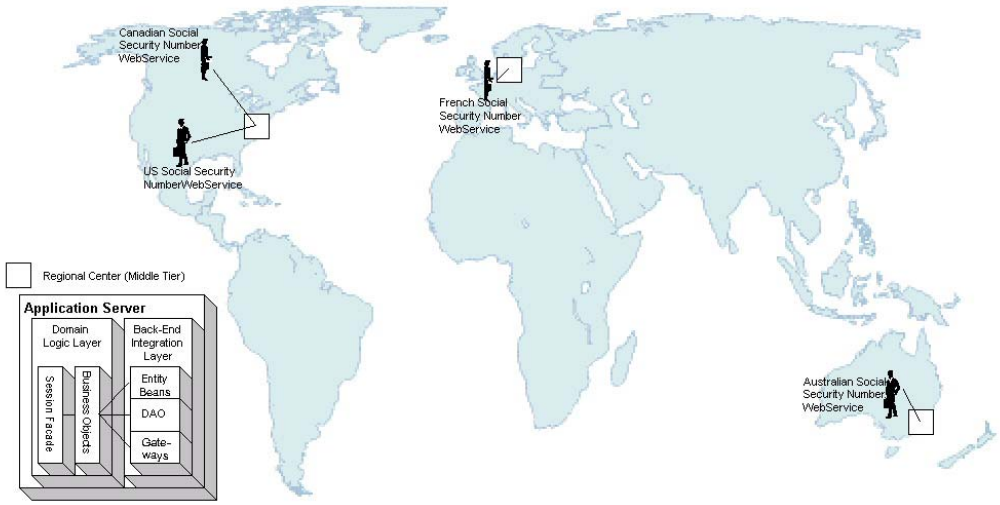

Figure 14: Social security number

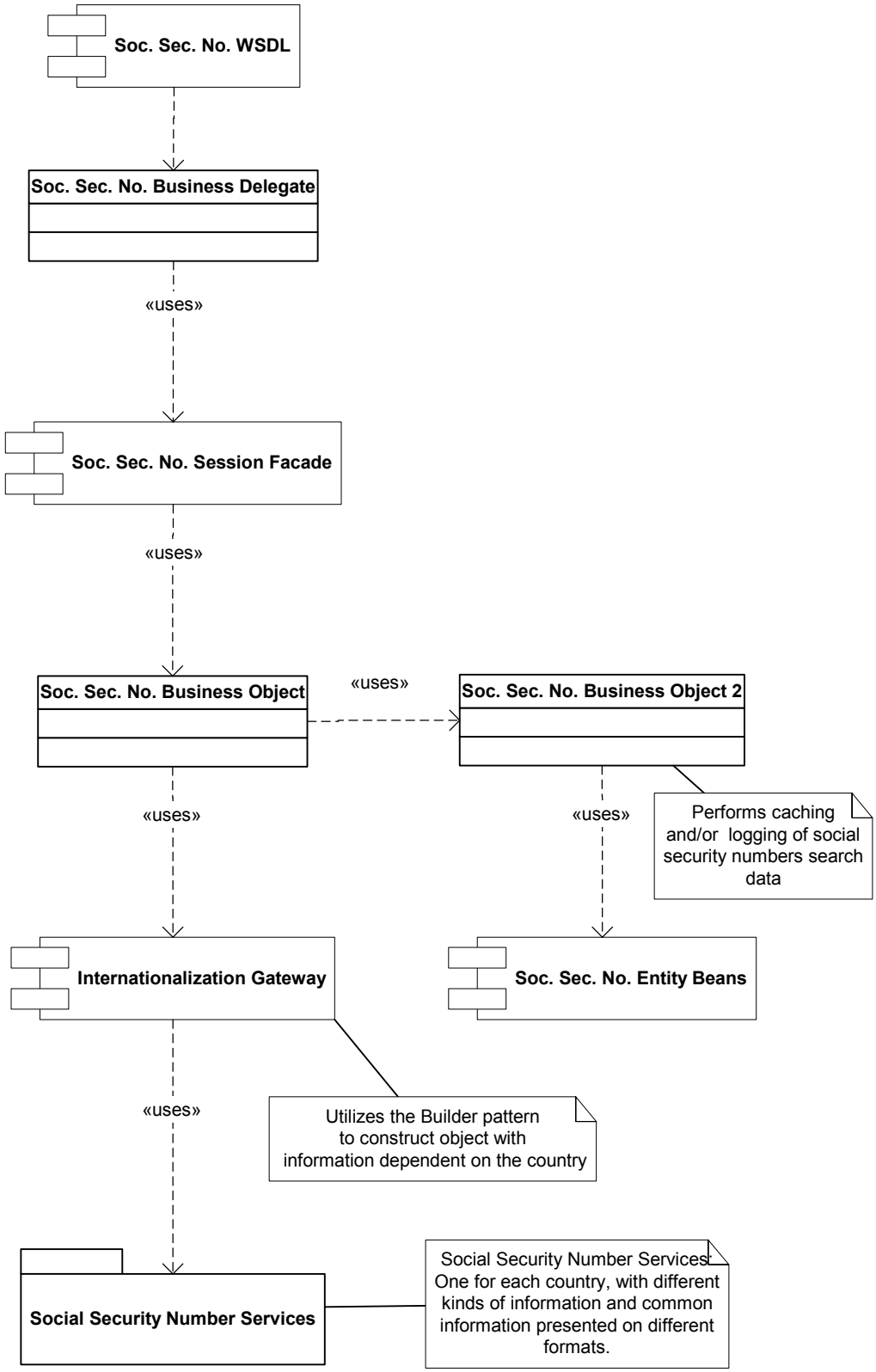

Figure 15: Social security number gateway

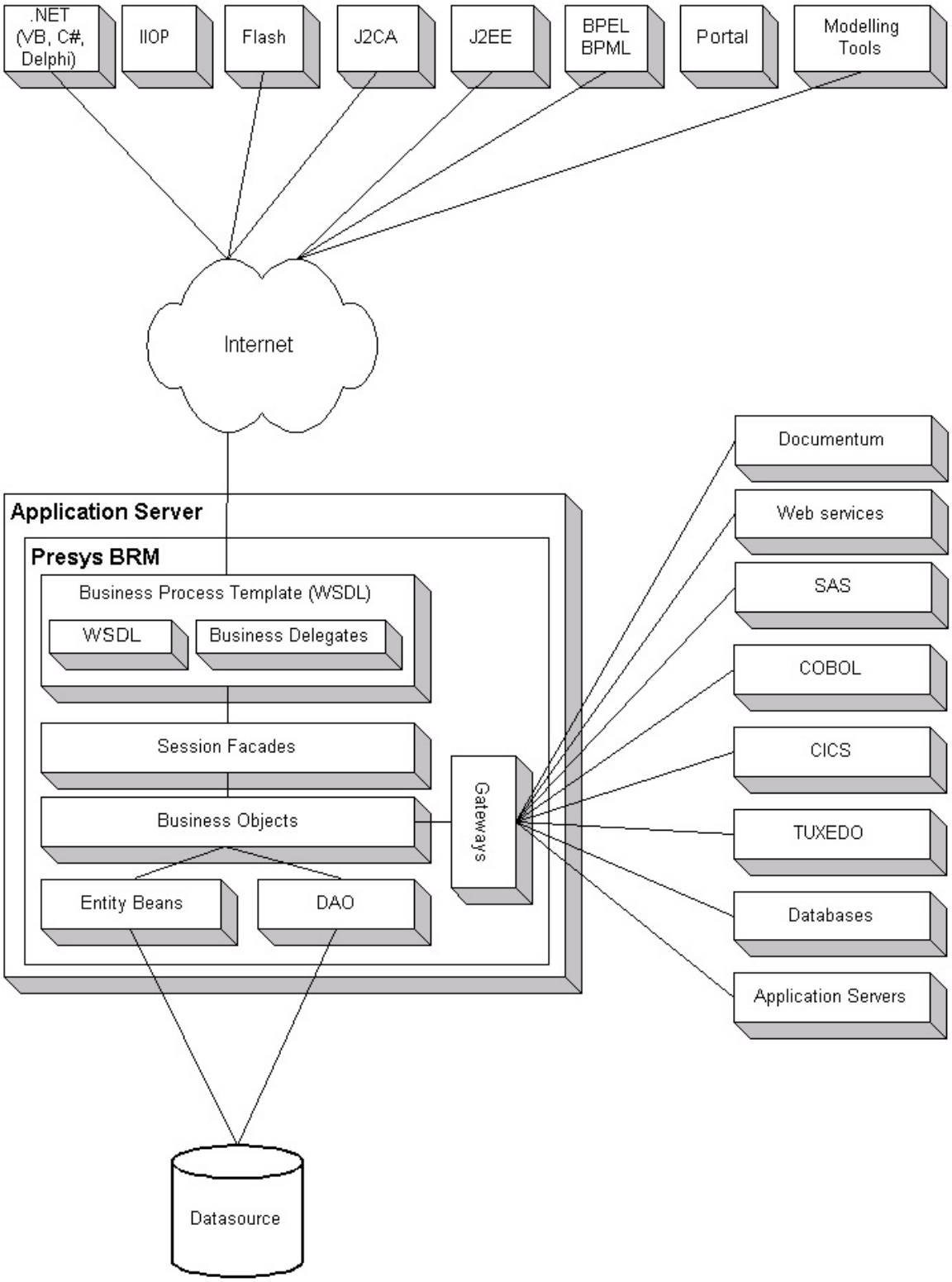

Figure 16: On BRM and integration possibilities

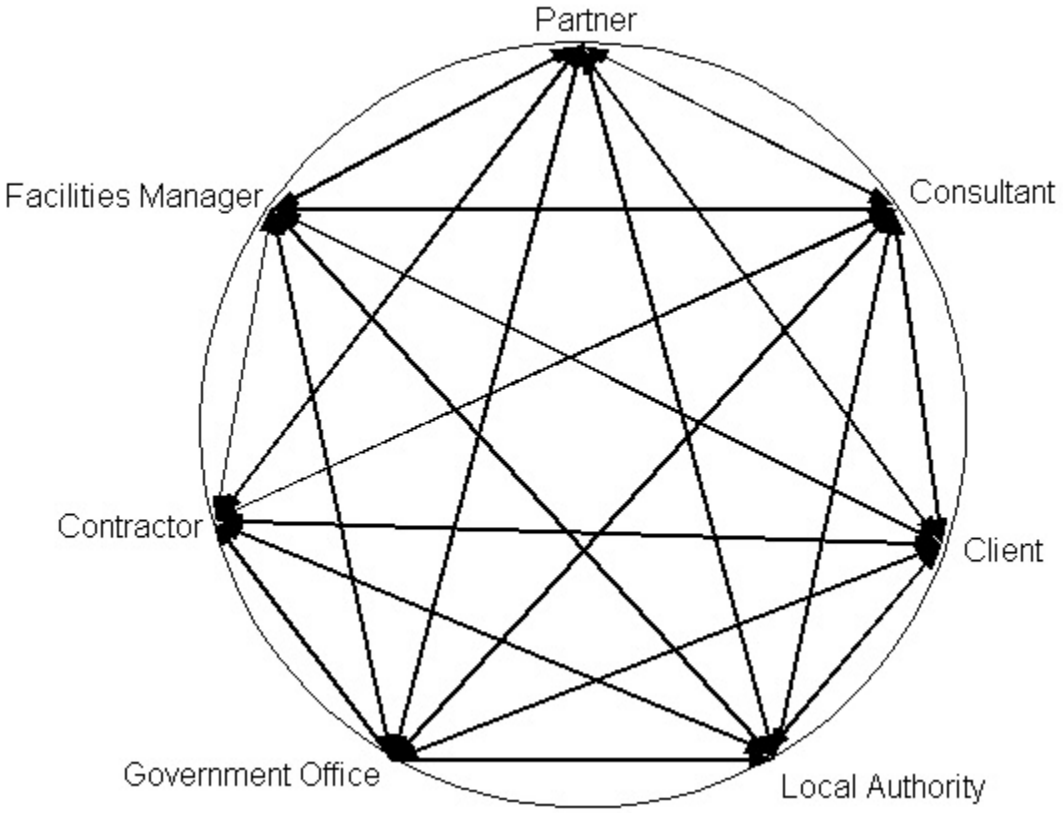

Figure 17: Sequential processes

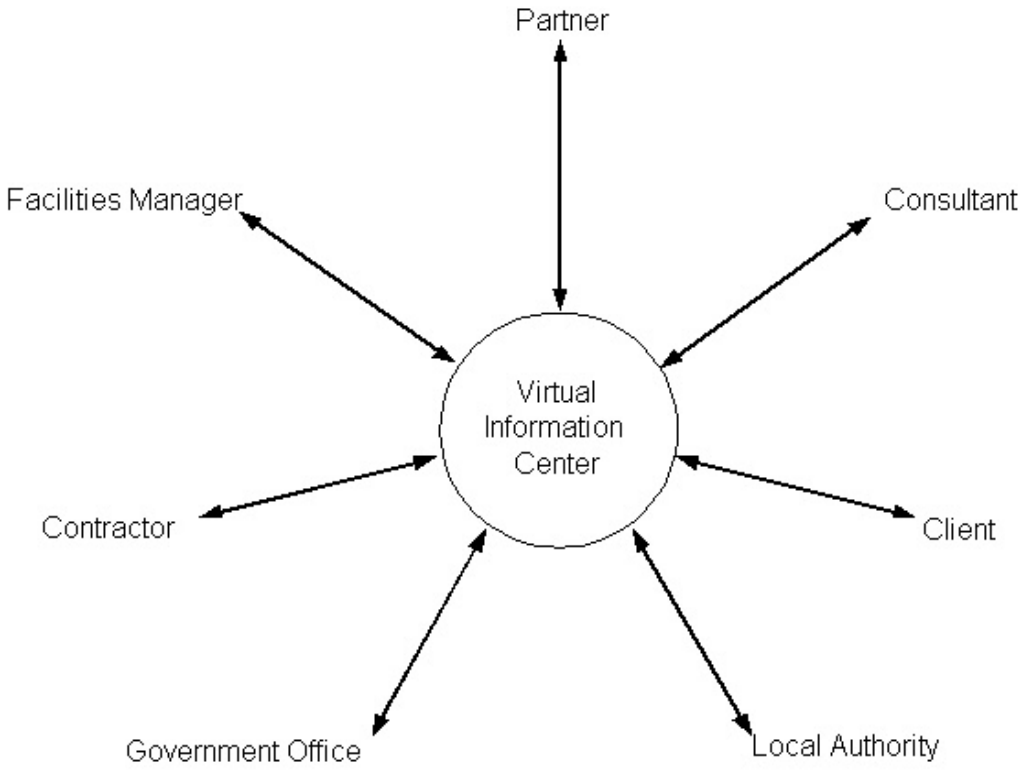

Figure 18: Virtual Information Center

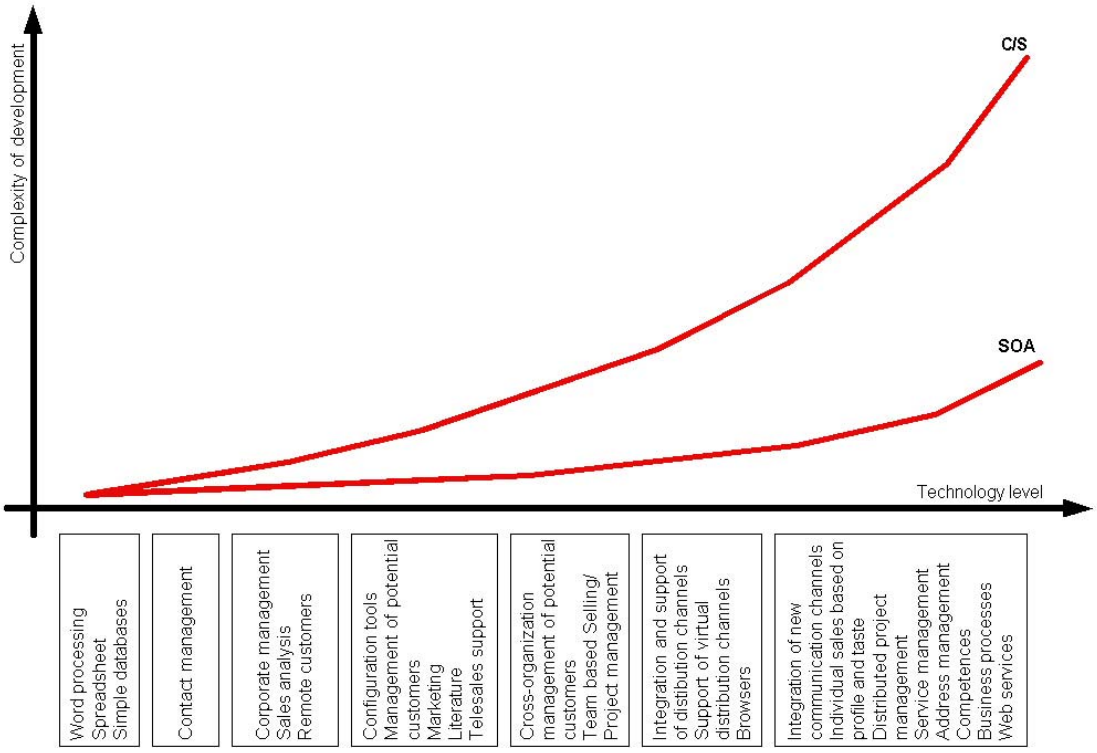

Figure 19: Complexity of CRM systems

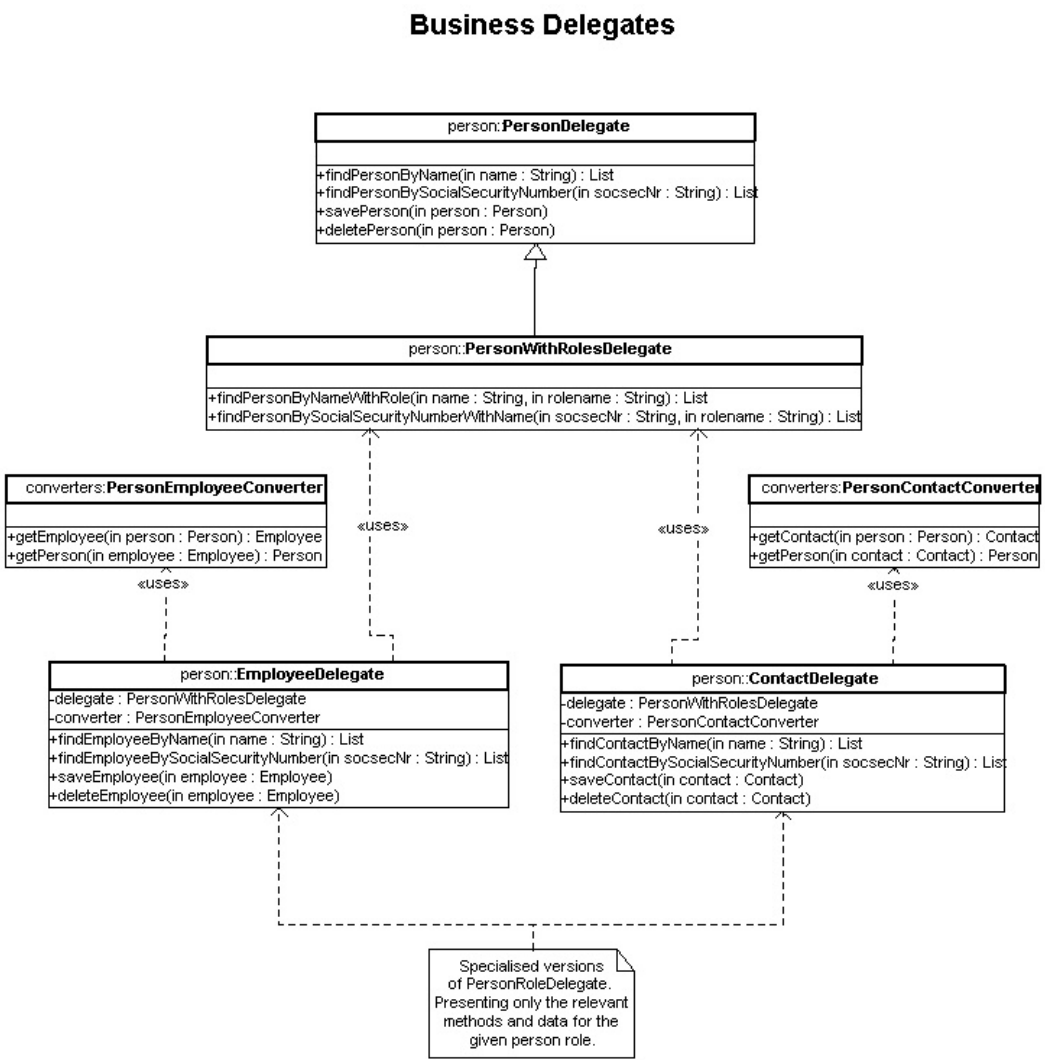

Figure 20: On BRM Business delegate example

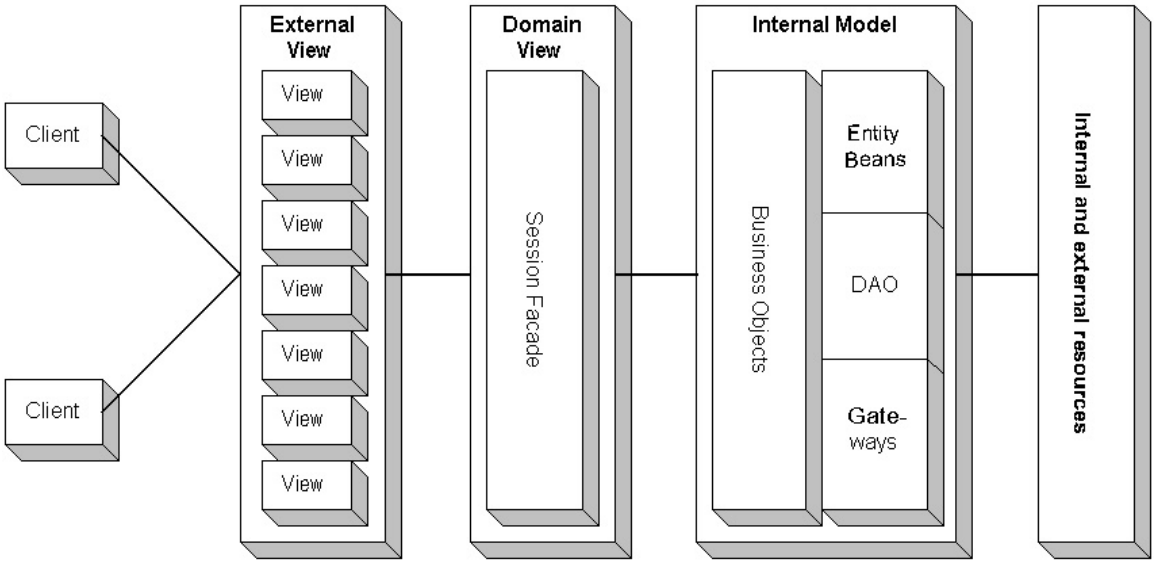

Figure 21: Developer view of On BRM's architecture

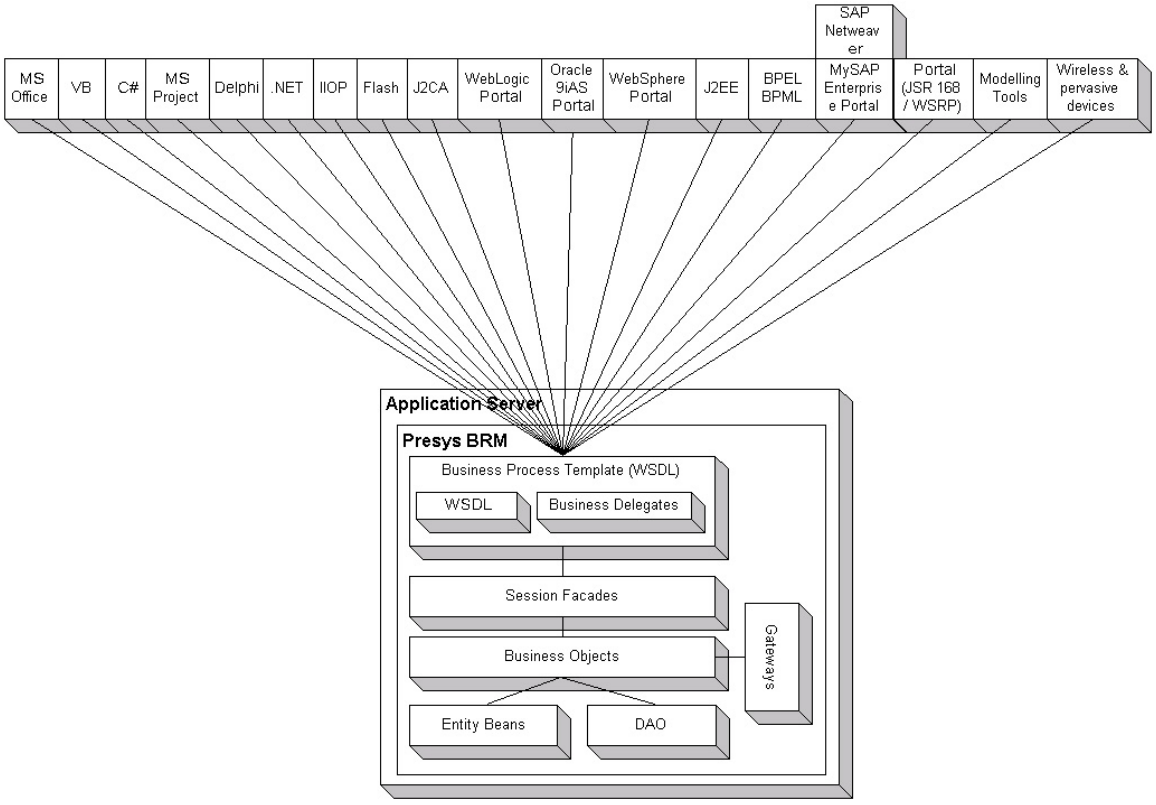

Figure 22: On BRM's possible user interfaces

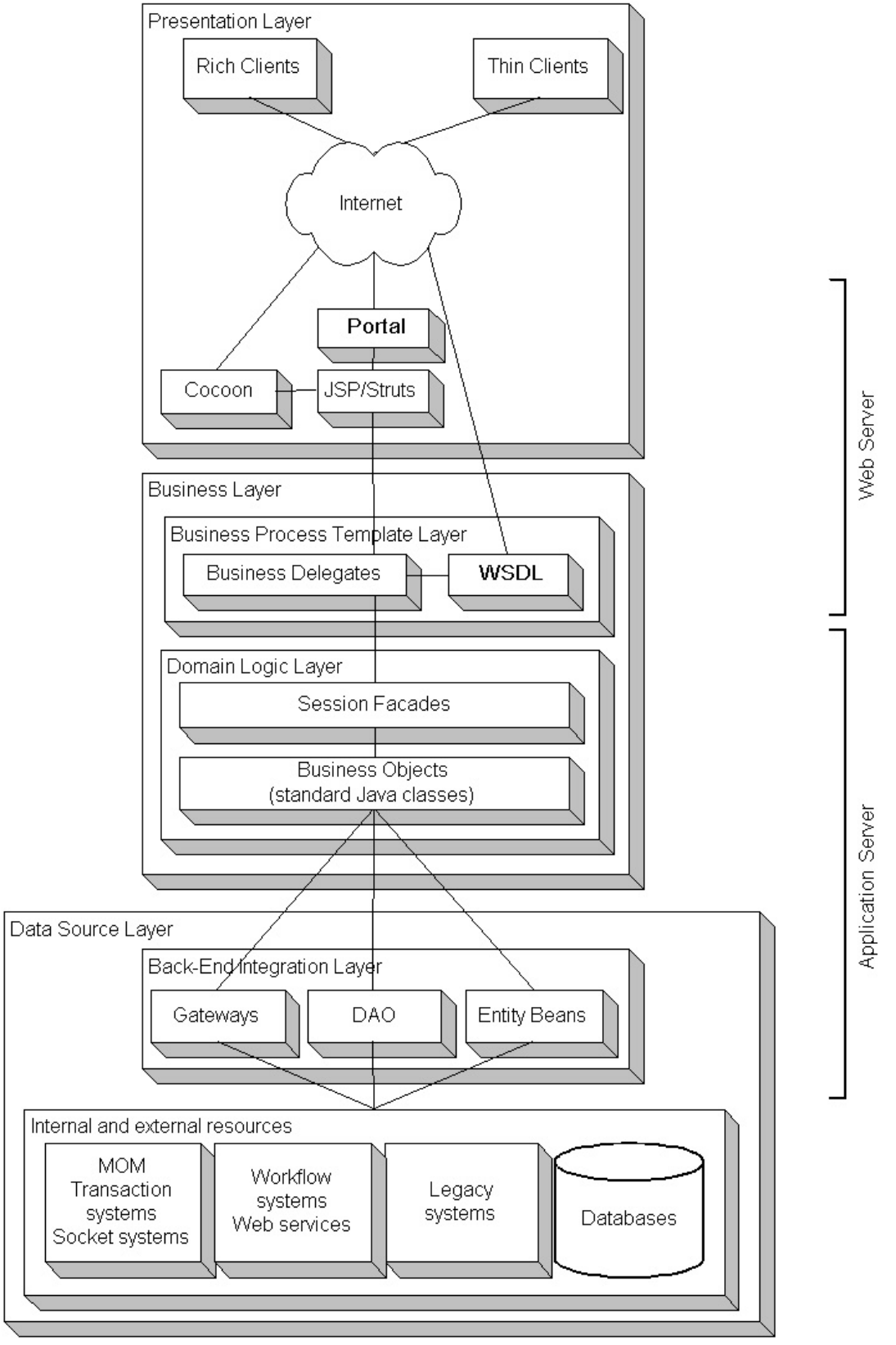

Figure 23: On BRM in a three layer perspective

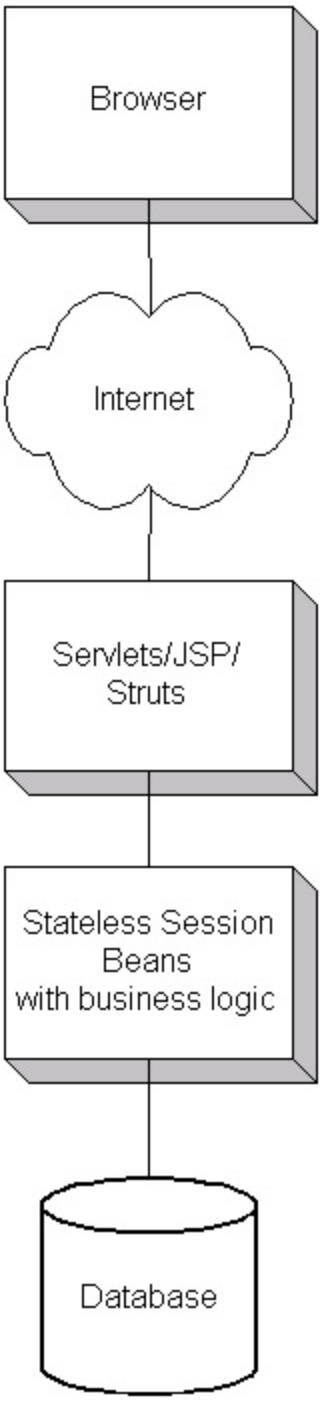

Figure 24: Transaction scripts using J2EE

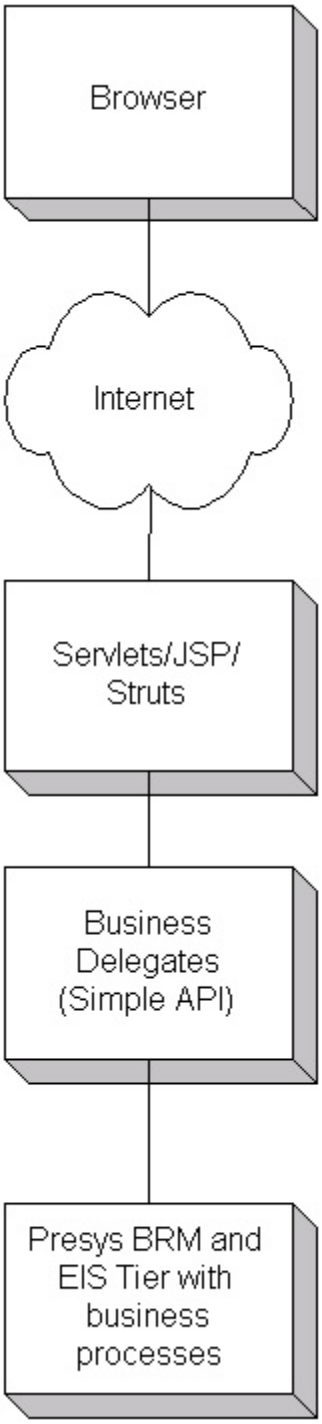

Figure 25: On BRM as seen by a Java developer

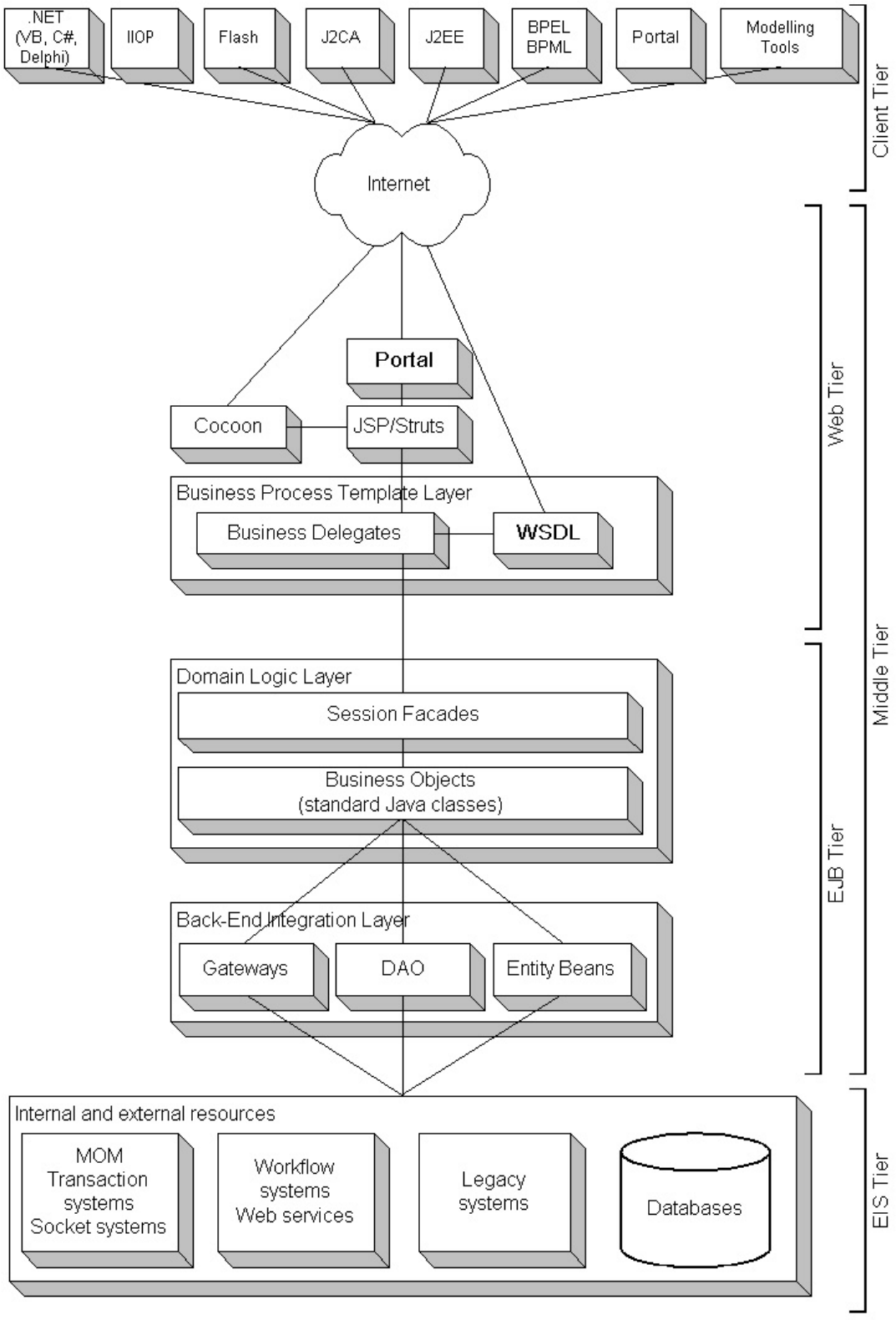

Figure 26: On BRM running on a single machine

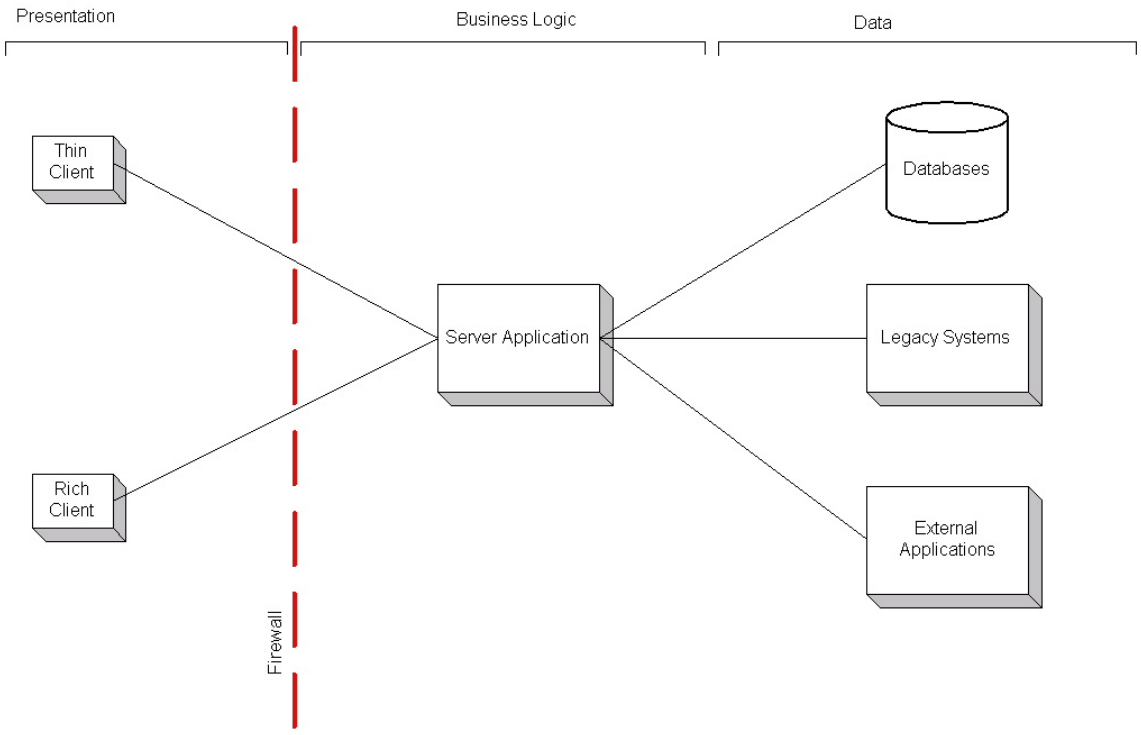

Figure 27: Three layer pattern

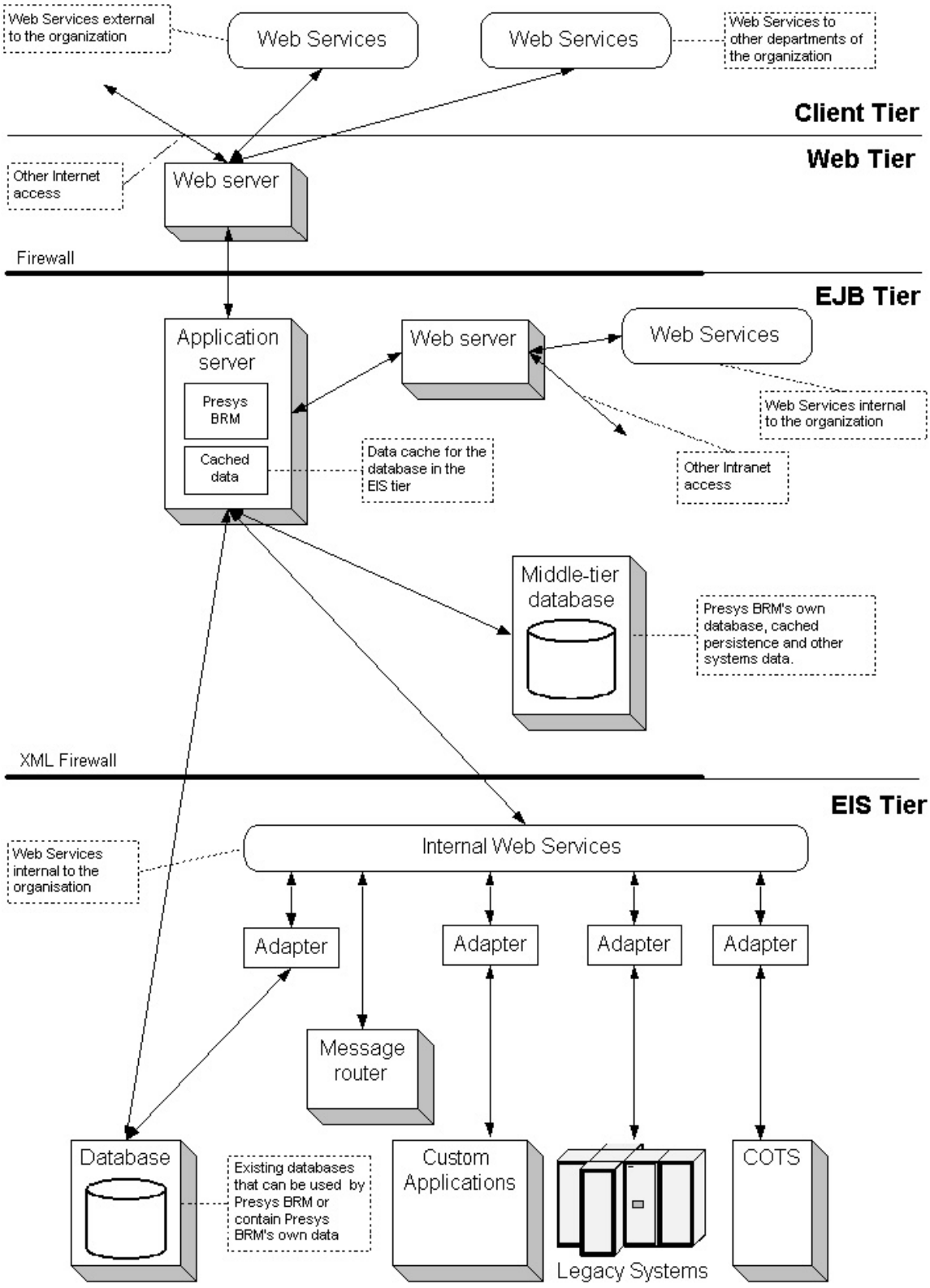

Figure 28: On BRM and Web services.

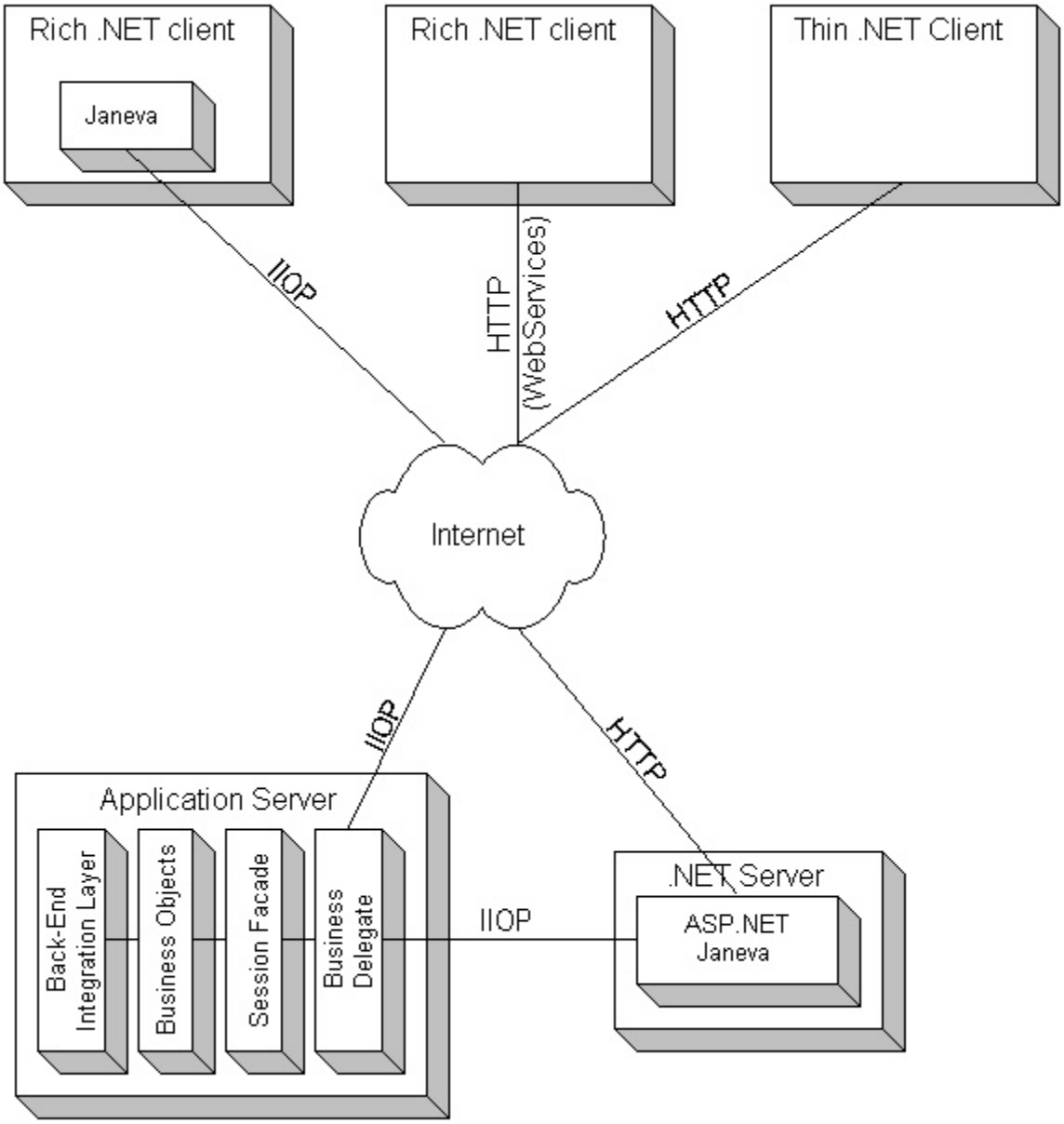

Figure 29: On BRM integration to .NET through Janeva

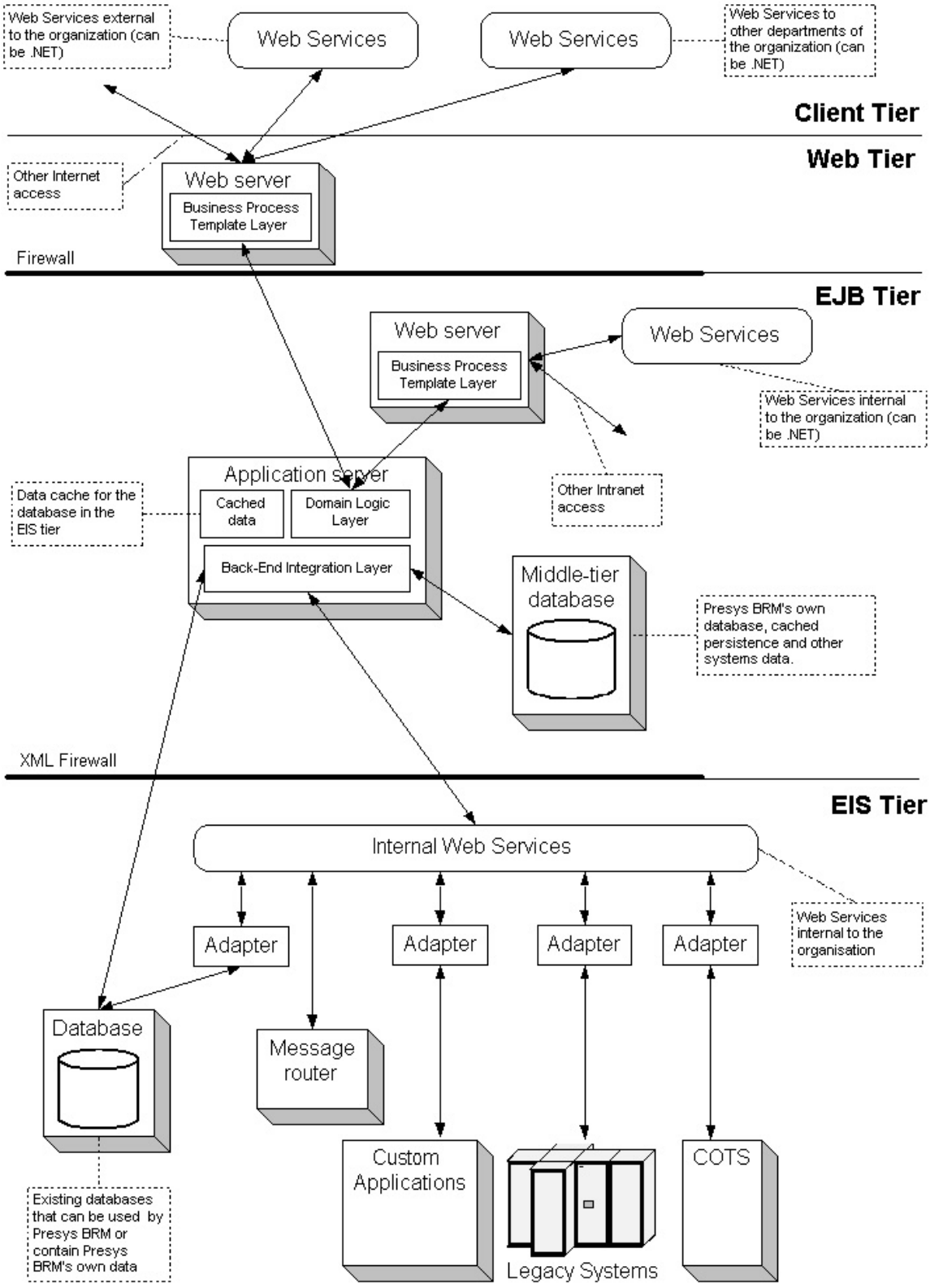

Figure 30: Integration with On BRM with Web services without the Web service stack