February 2005

Realization of Service-Oriented Architecture (SOA) Using Enterprise Portal Platforms Taking the Example of Multi-Channel Sales in Banking Domains

Rainer von Ammon
Upper Austria University of Applied Sciences

Wolfgang Pausch
Regensburg University of Applied Sciences

Markus Schimmer
ibi research at the University of Regensburg

In: Ferstl, Otto K, u.a. (Hg) 2005. *Wirtschaftsinformatik 2005: eEconomy, eGovernment, eSociety*; 7. Internationale Tagung Wirtschaftsinformatik 2005. Heidelberg: Physica-Verlag

ISBN: 3-7908-1574-8

Realization of Service-Oriented Architecture (SOA) Using Enterprise Portal Platforms Taking the Example of Multi-Channel Sales in Banking Domains

Rainer von Ammon
Upper Austria University of Applied Sciences

Wolfgang Pausch
Regensburg University of Applied Sciences

Markus Schimmer
ibi research at the University of Regensburg

Abstract: Multi-channel sales make high demands on IT infrastructures. Service-Oriented Architectures (SOAs) are to enable reusable software-based services and flexible, adaptable business processes. To realize this, the use of enterprise portal platforms is planned. The state of the art has been tested in a relevant project that focused on typical use cases of automotive credit banks in retail banking applications. Initial findings indicate that these platforms are ideal for realizing an SOA. In addition to the technical implementation of an SOA, the planning of suitable services, their level of granularity and reusability is another challenge.

Keywords: Service-Oriented Architecture, Enterprise Portal Platform, Multi-channel sales

1 Multi-Channel Sales in Banks

In addition to the industrialization of banking operations, the main area of action currently being discussed is the reorganization of financial institutions as multi-channel and all-finance sales and service providers [Bart+03, p. 14]. Multi-channel sales in this context means that customers can use different channels for their entire customer process (information or initiative, consultation, conclusion, and after sales) when they access financial service providers (FSP). While from a technical point of view the "channel" is primarily seen as being various terminals, in the financial sector it can be understood as an organizational unit from the vendor's

standpoint or as a "customer touch point" from the customer's. The most important channels are currently branches, automated teller machines (ATM), the Internet, call centers, and, most recently, consultants in the field.

In setting up these channels, it must be possible for the vendor to monitor and support customer processes in all channels to prevent any breakdown in the flow of processes or information when the point of contact changes. After obtaining information on his own via the Internet, the customer should, for example, be able to conclude the transaction in the branch office and contact the call center for further dealings with the institution. Communication between the customer and the FSP must thus always take up where the current level of information from the previous channel left off. The strategic questions facing a bank include

- which parts of the customer's process
- for what products and services
- for which customers
- on which channels

are to be offered (cf. in detail [Gron03; GrVo03]).

Another requirement in multi-channel sales is to realize interactions with the customer via all channels in the same way so that same activities lead to identical results. A budgetary accounting for a consumer credit must record the same data and be calculated in the same way via the Internet as during a personal meeting in the branch office or with the mobile consultant.

2 Multi-Channel Architectures for Multi-Channel Sales

2.1 Requirements for Multi-Channel Architectures

An application architecture for multi-channel sales that meets the aforementioned requirements must have several "target properties" (with reference to [Felt03, p. 5; Lese+02, p. 6-15]).

- As a rule, different channels and their specific terminals must be supported by the user interface.
- The application architecture must guarantee consistency between the channels at the process, function and data levels.

- Transparency between channels is mandatory; exchange of information must take place. The FSP needs to obtain an integral picture of the customer through which all activities on all channels are comprehensible.
- The application systems (APS) need to provide real-time capabilities to handle business processes without back-office batching. Manual steps and media discontinuity must be minimized or, if possible, completely removed.
- Finally, the flexibility and scalability of the application architecture must be guaranteed. Process modifications must be rapidly implementable and not obstructed by "hard-wired" IT components. New channels or external partners should be able to be added rapidly, with the IT environment continuing to function without error or losses in performance.

2.2 Obstacles to Implementation

For all practical purposes, the demands of an implementation are still frequently hindered by technical problems and mandatory infrastructures. Host systems are still very important to FSPs. On these mainframes monolithic and difficult to modify systems for accounts, securities, or payment transactions are implemented holding the entire legally relevant data stock [DeSc00]. The application architecture is oriented towards branches or accounts instead of business processes and customer needs [Moor99, p. 12]. This makes consistent workflows guided by the customer process, as defined above, impossible to achieve.

Today's multi-channel architectures often grew in an opportunistic way, with new channels added according to demand [Felt03, p. 4]. There are a variety of independent, sales-channel specific application blocks for branch offices, the Internet, call centers, etc. with redundant business logic [DeSc00]. Consequently, there is no standardized function or process view. Additionally, there is often no standardized data view since distributed and redundant data is stored in different branch or channel-specific databases [Moor99, p. 12].

Another obstacle is the complexity involved in the coupling of the application systems. FSPs have a very heterogeneous system environment employing numerous proprietary systems and interfaces [Bran+04, p. 138; ITVe04]. The point-to-point integration often used in the past is, however, too inflexible since many proprietary interfaces and protocols by different manufacturers have to be supported [LaWe03, p. 15].

2.3 New Approach Implementing Service-Oriented Architecture

FSPs' current application architectures are frequently a collection of multi-layered applications with the main user interfaces usually being desktop applications or

web browsers [Mehl01, p. 55]. These application blocks are the fundamental problem since their user interfaces are still linked to these monoliths [Wood03, pp. 17-24]. It is indeed possible to reuse parts of them by linking one monolith with another through portals and Application Programming Interfaces (API). However, this may be limited in that portals under certain circumstances may only be able to access a fraction of the monoliths or, if the interface is inappropriate, not at all. On the other hand, a connection via the API directly in the client's coding is difficult and expensive. The problem may be solved ideally by further dividing up the program logic into components that are then linked with one another using a Service-Oriented Architecture (SOA) (cf. Figure 1).

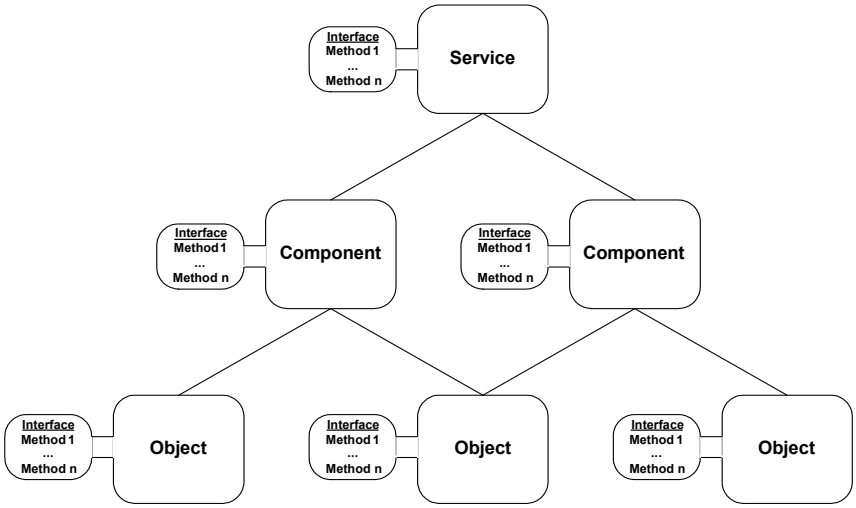

Figure 1: Aggregation of categories, components and services and their interfaces on different levels of abstraction and granularity

One of the most important tasks is to identify and develop components in order to create new and flexible services in the shape of a network of components [Mein04, p. 33]. During the first stage, it is important to internally break down the applications of an enterprise into components. For an effective breakdown, the FSP applications must be looked at from an overview perspective to identify shared functionalities which should later be made available as services. If functionality is redundant, e.g. contained in several applications or parts of applications, it indicates that these parts can be merged and combined in a single component.

Once possible services have been identified, their correct granularity must be determined prior to breaking down the application system. A level of granularity that is too coarse limits reusability, if the components bundle too much functionality

and are specialized for certain applications. By contrast, too fine a level of granularity may result, particularly in large systems, in an uncontrollable set of services that each contains too little functionality.

If reusable components have been identified and the appropriate granularity has been determined, the architecture of the old application system is broken down. This makes it possible to model the central elements of an enterprise architecture based on service (cf. Figure 2).

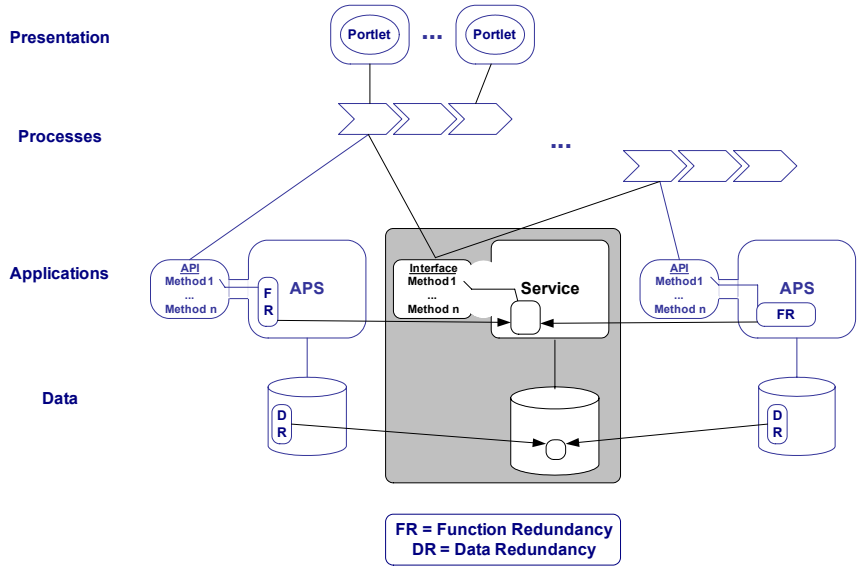

Figure 2: Redundant functionality and data, their modeling in suitable general components, and their reuse in one or several services

An SOA is based on three basic roles: that of the "Service Provider" making services available, that of a "Service Consumer" taking advantage of the services, and that of a "Service Broker" acting as an intermediary between "provider" and "consumer" in the form of a public or corporate "registry" (Figure 3).

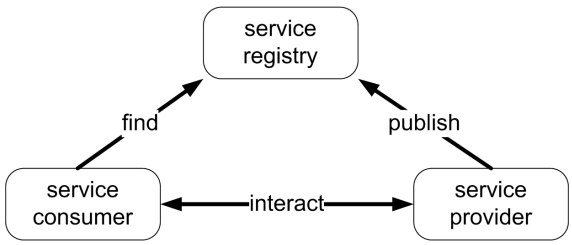

Figure 3: Interaction of the roles in an SOA (with reference to [KuWö02, p. 79])

As the means of communication, platform-independent service interfaces should be available over which the services can then be loosely coupled. One way of coupling service providers and service consumers is to use web service interfaces, such as the "Web Services Definition Language" (WSDL, [W3C04]). The service registry can be realized using "Universal Description, Discover and Integration" (UDDI, [OASI04a]). The data exchange format XML can be used for communication between the services.

SOA is expressly *not* synonymous to web services and XML but can contain many other technologies and standards, such as the "J2EE-Connector Architecture" (J2EE-CA, [Sun04]), "Java Process Description" (JPD, [JCP04]), and similar competing standards for orchestration and workflows such as the "Business Process Execution Language for Web Services" (BPEL4WS, [OASI04b]).

The services are either joined horizontally or vertically (cf. Figure 4). Horizontal in this context means that the components are loosely coupled to business processes ("workflows") that are positioned horizontally on top of the application system's old functional organization.

Services in these business processes can be accessed serially (serial orchestration) as well as in parallel (parallel orchestration). Serial orchestration refers to a process which is used to access web services one after the other. This means that the subsequent service waits for the previous service to terminate and then starts after its predecessor terminates so that the required part of a "workflow" is run. By contrast, in parallel orchestration, services are launched simultaneously. Before the business process will continue at the point of synchronization, all services invoked in parallel must be terminated.

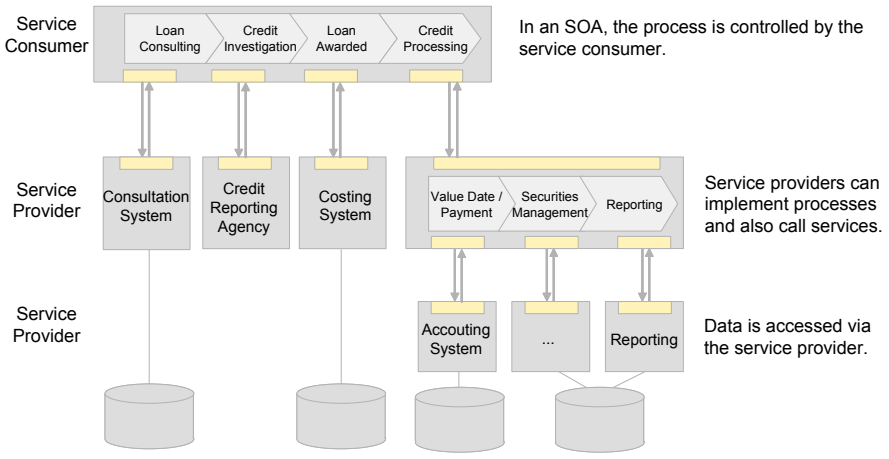

Figure 4: Horizontal and vertical coupling of services (with reference to [LaWe03, p. 12])

In contrast to horizontal joins, vertical joins mean that a vertical cut runs through the old architecture. In a complete cut, for example, a (very course-grained) "credit consulting" service would be cut out of the old architecture starting at the presentation level in a portlet over the presentation logic level in a "Java Server Page" (JSP) and the business logic level for processing data to the database access level. This can produce reusable portlet galleries that may be flexibly installed in company portals.

3 SOA Using Enterprise Portal Platforms

This section is intended to show how an SOA can be realized using an enterprise portal platform. In the process, services with varying levels of granularity are either assigned to the portal level, the integration platform level, the application server level or several levels (Figure 5).

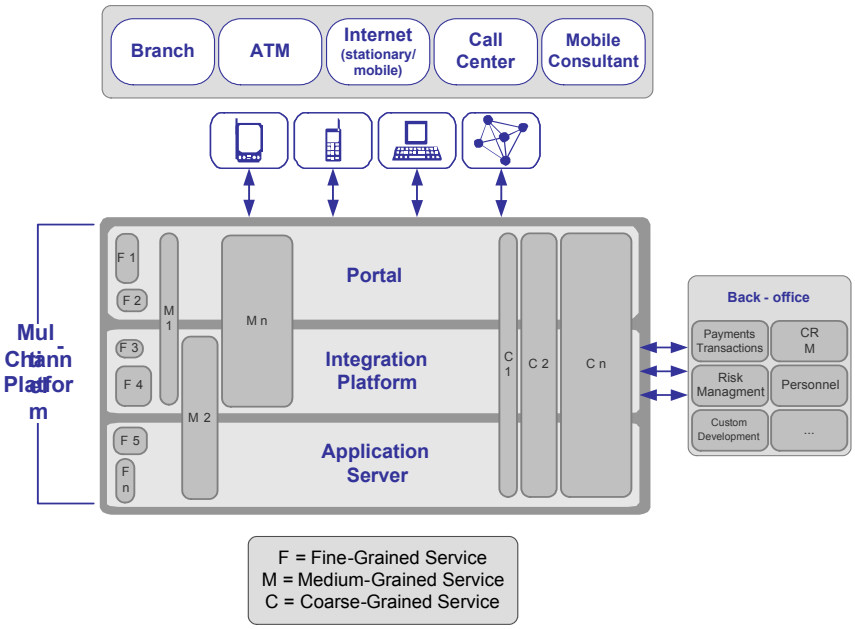

Figure 5: Basic architecture of enterprise portal platforms and diagram of the levels of granularity in service: fine, medium, coarse

In addition to their granularity and allocation to functional levels, services can be classified as *generic* services made available by the enterprise portal platform (customer management, product catalogs, etc.; see the following description of the different platform levels for more details) or as *domain-specific* services (for the

application domains “Finance”, “Logistics”, “Healthcare”, etc.) which can increasingly be purchased from specialized manufacturers on the “componentware” market (e.g. services provided by abaXX, a vendor specializing in finance portals [abax04]). A further differentiation are *application-specific* services which are custom-developed by the users for specific business targets. The classification of services for banking applications with respect to SOAs is the subject of an ongoing study conducted by ibi research at the University of Regensburg.

3.1 The Portal Level

In today’s distributed application systems, different groups of users such as customers or employees access the information and applications relevant to their needs using different user interfaces (e.g. various web browser technologies, such as Perl, JSP, or PHP, or native desktop applications). This is referred to as “multiple single-point-of-access”. These solutions are incompatible and can only be combined and modified with great difficulty. Portals solve this problem. A single point-of-access guarantees precisely one point of entry for an FSP’s customers, business partners, and employees, as well as a standardized and personalized view of the data and applications integrated in the portal.

A portal consists of several portlets. In the context at hand, this “view” refers to the view of an application or a service within the meaning of SOA, e.g. the querying of deposit holdings or the purchase of securities. The view of this kind of service is driven by a page flow logic that, depending on the progress within the business process, queries the deposit number or shows the content of the security deposit in the next process step. Different users or user groups in their various roles, e.g. as the employee of a specific sales channel, are allowed to see and execute only certain business processes. This is managed by a personalization component dependent on permissions and roles assigned to a particular user or a user group.

In addition to personalized access to the business processes offered in the portal, the portal level takes care of the adaptation of content to different terminals. This is “multi-channeling” from a technical point of view and means that users can communicate with the portal via cell phones, PDAs, notebooks, and so on.

The data basis for personalization and for further user-specific use of the corporate portal’s functionality are profiles generated from the user properties. These user properties are flexible and adaptable data structures in the enterprise portal platform which are used to identify and distinguish the user. Properties can be static values, such as the user’s date of birth, or include dynamic values such as the last bank product viewed. To obtain user properties, the enterprise portal platform offers the capture of dynamic data using “behavior tracking”, i.e. by recording user behavior in the portal. This makes it possible to analyze areas of interest, customer behavior, and the web site’s effectiveness.

Additional services that are already provided for by portal platforms are those offered by so-called campaign management systems. These systems enable sales campaigns to be activated or deactivated depending on a certain time period, a business target, individual actions performed by the user on the site or depending on events in the customer's situation (e.g. starting a job).

A portal platform's Content Management System (CMS) can also be considered a collection of services. These manage all kinds of content, e.g. product descriptions, images, forms, and their meta data, and offer functionality such as administration and authentification for the content's capture, modification, and release for publication in the enterprise portal.

3.2 The Integration Platform Level

The integration platform level provides services for the consolidation of applications, business processes, and business partners as well as for the integration and transformation of data between the various applications. In this sense, an EAI adapter based on J2EE-CA can be considered a service which integrates the functionalities of a core banking system such as Kordoba.

Using this kind of EAI adapter components it is possible to collect additional data for describing and classifying multi-channel sales customers by rapidly and flexibly integrating legacy systems as well as a wide variety of data sources.

The integration platform level differentiates between services for connecting corporate systems and a collaborative integration of business partners ("business-to-business integration"). In the latter, services that are used or offered by the business partner are made available as business processes via web services and their orchestration to enable dynamic integration of business partners and flexible modification of the processes.

3.3 The Application Server Level

The application server - the basis of the enterprise portal platform - must meet strict requirements in terms of execution, monitoring and coordination of global transactions as well as reliability, availability and scalability. In this context, application server level services can also be considered services provided within SOA.

To coordinate and monitor global transactions, application servers implement a transaction management service that coordinates the so-called "resource managers" such as relational databases, file systems, and other back-end services.

Scalability is a middleware service to support constantly growing numbers of users. It ensures that performance, specifically response times, remains unchanged.

This is sometimes realized by running the application processes on a cluster solution, i.e. a number of interconnected computers that may be heterogeneous as well as geographically dispersed, with automatic load distribution services offered by the application server.

Finally, an enterprise portal solution needs to guarantee a high level of reliability and availability. The failure of one or more server processes must not impact the infrastructure's availability. Reliable functioning of the application and execution of orders even with large numbers of user queries and transactions running at the same time must be ensured. The necessary basic services are made available on the application server level.

4 Sample Implementation

To demonstrate the deployment of an enterprise portal platform in multi-channel sales, the following scenario describes an automotive credit bank for financing individuals (retail banking). The bank traditionally offers financial services revolving around the purchase or leasing of a vehicle and is typically associated with an automobile manufacturer. Over the past few years, several of these banks have evolved into full banks offering classical banking services and products for retail customers (cf. [Fran02, p. 710; MuFa03, pp. 8-19]).

4.1 Use Scenario: Automotive Credit Bank (Retail Banking)

Banks connected with car manufacturers usually do not have their own branch offices. Personal consultations take place at the dealerships with a salesman. Other typical channels are call centers, kiosk terminals, and the Internet. Corporate customers are often served by mobile consultants. In this scenario, the typical roles within an enterprise portal platform are, on the customer side, an anonymous prospective customer, an existing customer, and an FSP's or car manufacturer's employee (Figure 6). The bank side includes a call center agent, a car dealer, and a mobile consultant.

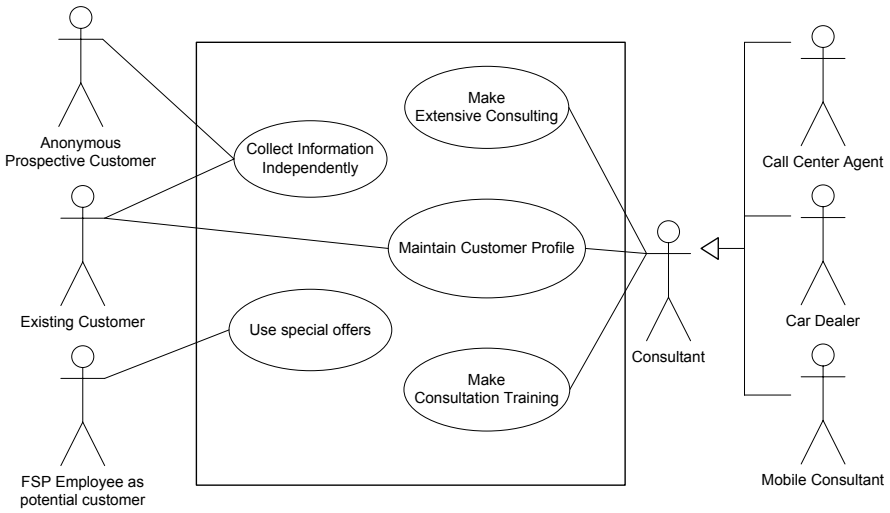

Figure 6: Exemplary roles and use cases of an automotive credit bank

Some exemplary cases of the many tasks automotive credit banks are currently addressing (cf. [Mess04]) are:

- Consulting: To increase the quality of advice given at the dealership, standardized consultation processes are prepared which (using different views) can also be employed for customers' individual research or in call centers. Informational dialogs for existing customers can be personalized by accessing existing customer data, or the dialogs are adapted according to each customer's level of knowledge or preferences.
- Training: The portal platform can also be used for training purposes. Sales staff in the various channels can run through consultation processes, browse current product offerings, or obtain background information.
- Active and event-driven customer contact: Another use case is connected with customer management. Existing customers can be addressed actively on the basis of certain events, e.g. their current life situation. This type of push concept is also conceivable for the FSP's or car manufacturer's staff who are made special offers regarding a private pension plan via the corporate intranet portal.

4.2 Implementation

As part of a project at ibi research at the University of Regensburg, an enterprise portal was to be created whose functionality was in line with the aforementioned scenario. The objective of the project was to study how enterprise portal platforms can be used in multi-channel sales in banks. The BEA-Weblogic Workshop 8.1 SP 2 development environment was used to implement the enterprise portal.

Chapter 3 describes which generic services are already offered by the platform and which levels they are used on. From the banking perspective, the domain-specific services must now be determined and defined to be as coarsely grained as possible or as finely grained as necessary with regard to their reusability and ability to be modified.

In the use cases introduced above, it would seem logical to use reusable and suitably grained services, e.g. for consulting. These services can be used for self-consulting sessions on the Internet, as a supplement for consultation provided in the branch office, in the call center, or by the mobile workforce, and for training sales staff; taking into account the different user groups on the one hand and the representatives' common role as consultants on the other.

To determine a suitable form of financing (drawing credit or consumer credit) this type of consultation service might consist of the following process steps with suitable finely-grained services coupled as necessary:

- Orientation and information regarding forms of financing for retail customers
- Selection of the form of financing (drawing credit or consumer credit)
- Product configurator for consumer credit
- Budget to determine the monthly balance available
- Feasibility check for consumer credit
- Interest and redemption
- Self-disclosure
- Application for consumer credit
- Application for extension of drawing credit
- Saving of session data
- Consultant contact

Services which are reusable in other applications include budgeting, self-disclosure, data storage, and consultant contact. The budgeting service can be reused in consultation services for mortgaging or the financing of private pensions plans. When defining the "Save consultation session" service, however, the developer may discover that the service cannot be generalized since the data obtained during the consultation session are too specific to this type of consultation and would need to be mapped to highly different data structures. This example points out some of the problems that bear watching when properly customizing domain-specific services for an SOA, e.g. in regard to their granularity and reusability.

While this task is a design problem independent of the implemented tools and enterprise portal platform, the quality of the development environment is measured

against the fast and flexible implementation of an SOA and the banking processes. Figure 7 shows an excerpt of the implementation of the consultation process for finding a form of financing, initially as a "page flow" process.

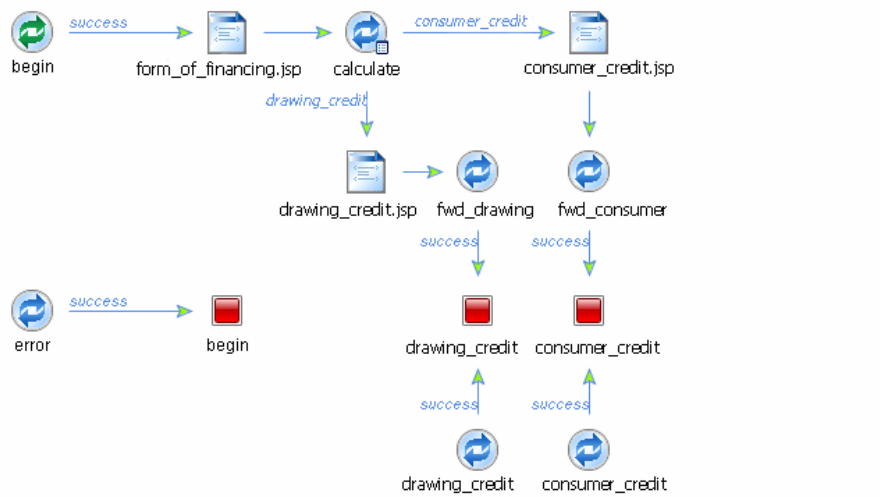

Figure 7: Realization of a consultation process with page flows

The page flow technology provided by the enterprise portal platform is used to integrate the consultation process with a company portal. Page flows can be defined as a sequence of steps or states. Transitions take place either initiated by a user action or by an action's return value. The correct flow can even be triggered by the results of an interaction with other systems that were integrated via EAI, e.g. the results of a credit rating query. From a technical standpoint, a page flow can currently be the graphical representation of a struts controller [Apac04]. JSPs are assigned to it, and java controls can be invoked. Java controls are used, among others, to call enterprise java beans, web services, databases, and much more to the page flow.

During project realization a number of additional features of the development environment were used. They were only mentioned briefly in this paper to help illustrate the implementation of an SOA. Another example is the development of the "active event-driven initiation" process step and its implementation using the service functionality of the campaign management system. This system allows for the definition of events and rule-based activation of actions (campaigns) on a high level of abstraction. This is done flexibly by the appropriate department who do not need a broad knowledge of IT basics. Figure 8 shows an example: an absolute purchase date turns out to be an unsuitable event because every time a customer or service logs in, the campaign should automatically be called or e-mail contact initiated if the customer's car is 3 or more years old.

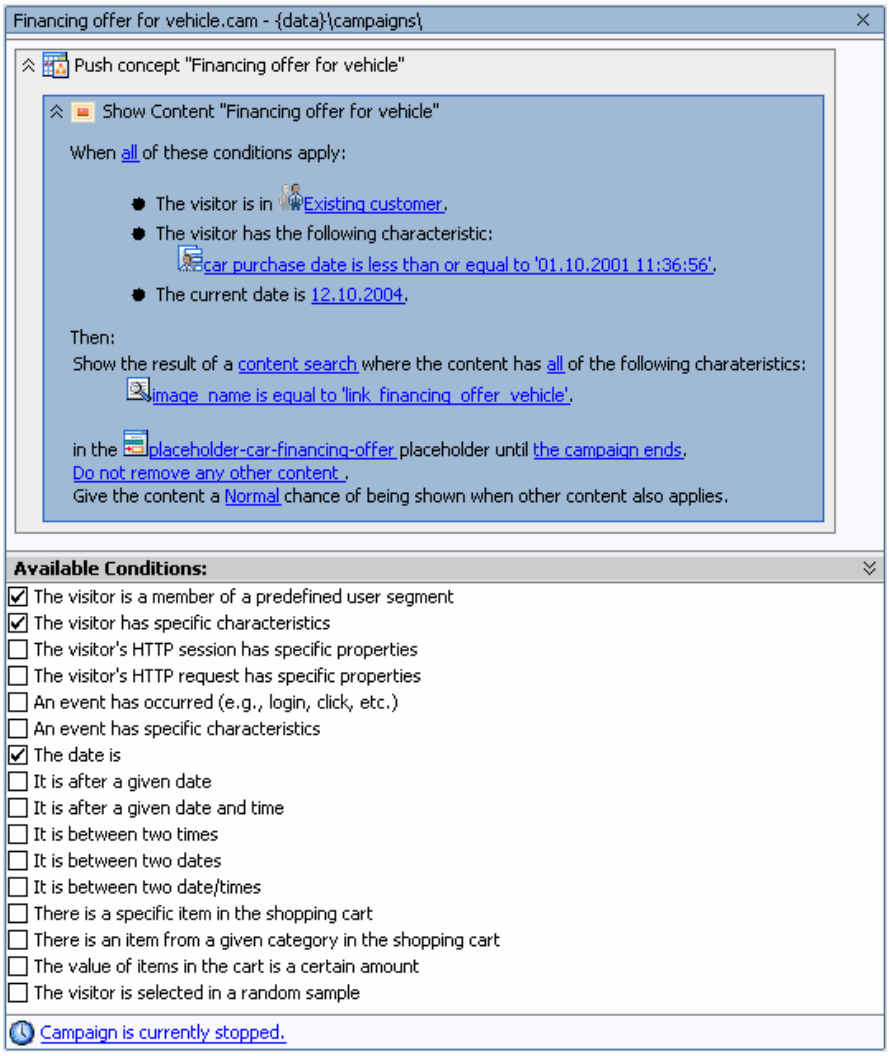

Figure 8: Condition editor for push concepts

5 Outlook on the Future

Initial results of the project lead to the belief that a multi-channel platform for banks can be realized technically on the basis of an SOA using the enterprise portal platforms available today. The corresponding standards are still being improved and constantly expanded. For example, the restrictions of web services

with regard to security and transactionality have just been solved and, as this document goes to press, are already available in platforms like the BEA Weblogic Workshop as Service Pack 3. The Java Specification Request (JSR) 207 which defines the JPD is being reviewed and modified to match the "Organization for the Advancement of Structured Information Standards" (OASIS)'s BPEL. The intention of a reference implementation at ibi research is to apply these new functionalities to specific application domains. An important basis for this project are the results of the aforementioned bank-specific SOA study that is to provide a suitable classification of the corresponding services.

We want to thank Ben Gebauer und Daniel Jobst for support and constructive discussions.

Literature

[abax04] abaXX Technology: abaXX Technology AG. http://www.abaxx.de, 2004, downloaded on 2004-05-01.

[Apac04] Apache Software Foundation: The Apache Struts Web Application Framework. http://jakarta.apache.org/struts, 2004, downloaded on 2004-05-01.

[Bran+04] Brandner, M. et al.: Web services-oriented architecture in production in the finance industry. Informatik Spektrum, Volume 27, No. 2, 2004: pp. 136-145.

[Bart+03] Bartmann, D. et al.: Retail Banking - Status-quo und Entwicklungsperspektiven. HMD - Praxis der Wirtschaftsinformatik, Issue 233 (40), October 2003: pp. 7-20.

[DeSc00] Dewal, S.; Schnichels, L.: Bank 2010: Eine fachliche und technische Vision. Softwaretechnik-Trends, No. 20, Issue 3, 2000.

[Felt03] Felten, G.: Multikanalbanking bei der Postbank. Workshop der Forschergruppe Augsburg-Nürnberg in Kooperation mit der GI-Fachgruppe Informationssysteme in der Finanzwirtschaft, September 16, 2003, Dresden.

[Fran02] Franke, D.: Autobanken: Weiter auf Kurs. Die Bank, No. 10, 2002: pp. 709-711.

[GrVo03] Grimm, S.; Volk, L.: Multichannel-Management im Allfinanzvertrieb. BIT - Banking and Information Technology, Volume 4, No. 4, 2003: pp. 23–36.

[Gron03] Gronover, S. C.: Multi-Channel-Management. Dissertation, Universität St. Gallen, 2003.

[ITVe04] IT Verlag: Integration & Migration Day 2004, Munich, February 19, 2004. http://www.it-verlag.de/veranstaltungen_neu/events.html?Integration, 2004, downloaded on 2004-05-01.

[JCP04] JCP (Java Community Process): JSR 207 - Process Definition for Java. http://www.jcp.org/en/jsr/detail?id=207, 2004, downloaded on 2004-05-01.

[KuWö02] Kuschke, M.; Wölfel, L.: Web Services kompakt. Spektrum Akademischer Verlag: Berlin, 2002.

[LaWe03] Laures G.; Wedekind K.: Serviceorientierte Architektur (SOA) – Wertstiftendes Architekturparadigma. Presentation held at the iX conference in Heidelberg, November 2003.

[Lese+02] Leser, F. et al.: Informationssystemarchitektur für Multi-Channel-Lösungen. Work Report No. BE HSG/CC BN/ 8, Universität St. Gallen (HSG), Institut für Wirtschaftsinformatik, 2002.

[Mehl01] Mehlau, J. I.: IT-Architekturen für Finanzdienstleister – Ein State of the Art Report. BIT - Banking and Information Technology, Volume 2, No. 3, 2001: pp. 41-58.

[Mein04] Meinhold, G.: EAI und SOA: Die Komponenten fallen nicht vom Himmel. Objektspektrum, No. 2, 2004: pp. 33-36.

[Mess04] Messner, W.: Kundenmanagement: Bricht das Wertschöpfungsnetzwerk der Autobanken auf?. Die Bank, No. 4, 2004: pp. 265-269.

[Moor99] Moormann, J.: Umbruch in der Bankinformatik - Status quo und Perspektiven für eine Neugestaltung. In: Moormann, J.; Fischer, T. (Hrsg.). Handbuch Informationstechnologie in Banken. Gabler: Wiesbaden, 1999, pp. 3-20.

[MuFa03] Mummert; F.A.Z.-Institut: Branchenkompass Mobilienfinanzierer. Mummert Consulting und F.A.Z.-Institut für Management-, Markt- und Medieninformationen: Hamburg, Frankfurt, 2003.

[OASI04a] OASIS: Universal Description Discovery Integration (UDDI). http://www.uddi.org/, 2004, downloaded on 2004-05-01.

[OASI04b] OASIS: Web Services Business Process Execution Language TC. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel, 2004, downloaded on 2004-05-01.

[Sun04] Sun Microsystems: J2EE Connector Architecture. http://java.sun.com/j2ee/connector/index.jsp, 2004, downloaded on 2004-05-01.

[W3C04] W3C (World Wide Web Consortium): Web Services Description Language (WSDL). http://www.w3.org/TR/wsdl, 2004, downloaded on 2004-05-01.

[Wood03] Woods, D.: Enterprise Services Architecture. O'Reilly: Gravenstein, 2003.