SOA solutions
White paper
September 2006

Lotus® software

IBM

IBM portal and collaboration products switch on SOA.

Deploying IBM portal and collaboration products for immediate business value—a technical guide

Dr. Barry Devlin
Dublin Software Laboratory
IBM Software Group

Contents

Introduction

This white paper demonstrates how to use IBM portal and collaboration products to rapidly build an initial service-oriented architecture (SOA), integrating and updating a set of legacy applications and in the process significantly improving user productivity through a modern, portal-based interface. We base our discussion around a simplified but realistic customer care scenario.

We start with a brief background to the concepts and drivers for an SOA.

Then we introduce our scenario and describe the customer care environment prior to the introduction of an SOA. Our main viewpoint is through the eyes of a customer service representative (CSR) who used the system every day, but we also look at the evolution and structure of the system from an architect's point of view. Thus, we see the problems the CSR had using the system, as well as the difficulties the IT department encountered when providing this service.

The two main sections of this paper describe the SOA-based solution. The new architecture is outlined as well as the technical considerations when integrating existing applications into the SOA. We briefly examine how existing, or legacy, applications may need to and, indeed, can be decomposed into reusable services and how these services combine in composite applications. We introduce the middleware products chosen—IBM WebSphere® Portal, IBM WebSphere Portlet Factory and IBM Lotus® Sametime® software—as well as some solutions and the rationale for these choices.

Finally, we show the benefits for both the user and IT populations. Users gain improved productivity, reduced learning curves and the opportunity for more innovation in their work. The IT department is relieved of routine maintenance work on existing applications and so can focus on delivering new function rapidly and effectively.

Highlights

An SOA supports service orientation, based on open standards; a service is any well-bounded, defined and repeatable business task that can be invoked in a standard manner.

SOA enables the modeling and design, assembly, deployment and management of flexible, integrated applications from reusable services that are independent of the applications and computing platforms on which they run.

Composite applications are most readily implemented within a portal architecture, with individual portlets broadly corresponding to discrete services.

The background to SOA

A service-oriented architecture is simply an IT architecture that supports service orientation, based on open standards. A service is any well-bounded, defined and repeatable business task that can be invoked in a standard manner. Note that the orientation here is strictly business. The scope of this task can range from very narrow to quite broad. Thus, it may be a simple, one-step task, such as setting a customer's billing address, or a more complex task involving several steps and a number of possible outcomes, such as opening a new account. Services can be nested; that is, one service can call another to perform a subtask or even can determine whether another task is called or not. This type of service integration and the linking of outcomes allow applications to be both flexible and integrated—two key requirements in modern business and the basis for true enterprise agility.

SOA thus enables the modeling and design, assembly, deployment and management of flexible, integrated applications from reusable services that are independent of the applications and computing platforms on which they run. These processes are supported by a governance approach and by best practices derived from experience, as illustrated in Figure 1 (see page 4). Such flexible, integrated applications are commonly known as composite applications and form the foundation for adaptive business processes. In the past, a business process was a largely static set of tasks within one organizational unit, implemented within a single, monolithic, traditional application. Going forward, a business process will be founded on composite applications that consist of flexible business logic between services (usually known as workflow) and independent and easily modified services that span organizational boundaries. Composite applications are most readily implemented within a portal architecture, with individual portlets broadly corresponding to discrete services.

Highlights

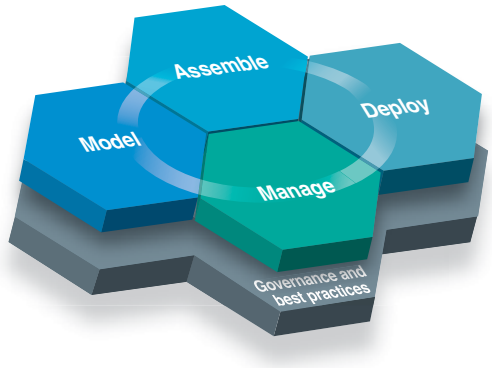

Figure 1. Service-oriented architecture

Within the SOA environment, portlets form the key means of user interface, and task pages provide the basic business functions.

While much attention has been focused on services that can be extracted from traditional, operational, line-of-business applications, other types of services can be just as easily incorporated into an SOA. Among the most important of these are portal and collaborative services that are the basis for user interaction with the workflow and are key to the significant improvement in user productivity required today. Within this environment, portlets form the key means of user interface, and task pages provide the basic business functions. An orchestrated workflow is essentially a business process that involves presentation of task pages and portlets by the portal server to a user, based on an understanding of the particular user's role, and that supports the most appropriate navigation and linkage of these tasks and portlets in the user's current role and circumstances.

Highlights

An SOA enables the creation of business processes that support and encourage teaming through local and remote collaboration.

In combination with portal and collaborative services, an SOA enables the creation of business processes that support and encourage teaming through local and remote collaboration. Such approaches can drive substantial innovation within the business by easing access to the knowledge that resides in users' heads and to information dispersed broadly across multiple data repositories.

For further information on the background of SOA, please see, *Opening the door to a service oriented architecture, IBM Workplace, Portal and Collaboration products for an SOA*, IBM white paper, January 2006, GC28-7727-00, available on ibm.com.

Introducing the customer care scenario

This section introduces a typical, legacy application environment of a fictional company.

This section introduces a typical, legacy application environment from a fictional company called Galactic Resource Provisioning (GReP), a nationwide retailer of automobile customization kits and accessories. The company provides customer care through a central telephone support center that handles inquiries, complaints and technical questions. GReP wanted its customer service representatives to drive new business by cross-selling in the next year. But the CSRs were resistant; they believed their workload was already too high.

The company's Complaints Management System was originally designed to provide every function that a customer service representative would need.

Consider the actions an agent, Maria, had to perform during a customer call. A legacy client-server application called the Complaints Management System (CMS) occupied a large window on Maria's screen and was at the heart of almost everything she did. The original design specification for CMS stated that it should provide every function that a CSR needs. Indeed, at one time, it almost achieved its goal. However, changes in the business and technology had combined to restrict the role of CMS.

Highlights

Changes in customer behavior and the company's IT systems meant that the CSR had to move between the customer database, CMS and SAP system to match a caller to his or her customer number and to verify purchase history and warranty status.

When a call came through, CMS tried to find the phone number in the customer database and display the customer details. Unfortunately, this function had become less effective as customers increasingly began to use cell phones or withhold their numbers for privacy reasons. In many cases, Maria had to ask the caller for his or her name. Maria then made the first of a number of journeys into the SAP system to do a search. On finding the customer number, she typed it into CMS, which could then find the rest of the customer record.

Next, Maria had to check whether the product was actually purchased from GReP and whether it was still under warranty. This necessitated further trips into the SAP system, followed by cutting and pasting the results into CMS. This meant the SAP window was always open on Maria's desktop, and she spent a lot of time switching back and forth and cutting and pasting information between CMS and the SAP system. Depending on the mix of products the customer owned and how he or she bought them, Maria may have needed a number of SAP windows open to figure out the caller's entitlement to support. As Maria often pointed out, this was a waste of time and effort.

To be fair to the IT department, originally CMS handled all data transfers between itself and the customer and product database. However, when GReP acquired an accessories business, it migrated its own homegrown customer database to the SAP system the other company had used. The effort involved in integrating CMS and the SAP system had proven too difficult.

Highlights

The CSR's desktop was crowded with windows for all the systems that needed to be accessed to resolve a customer support issue.

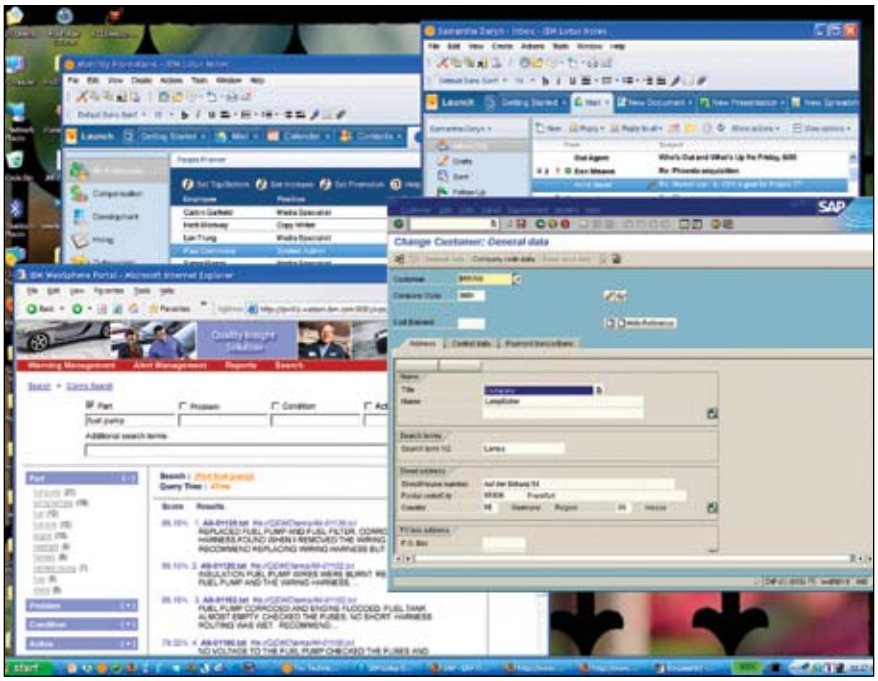

Figure 2. Screen shot of legacy call center system

Finally, Maria was ready to deal with the actual customer inquiry. As she entered details of the inquiry or complaint into CMS, the system automatically identified keywords that could be used in the answers database, a content management system, which appeared as yet another window on Maria's desktop. While keyword identification is a powerful and valuable feature, it had proven impossible in the past to seamlessly integrate CMS and the answers database, so Maria had to cut and paste the keywords. This was yet another interface with which Maria needed to be familiar.

Highlights

CSRs were faced with a number of challenges—including a disorganized desktop, different user interfaces and integration levels, the need to manually cut and paste information between systems and a lack of workflow support—and as a result they were not as productive as they could have been.

If Maria was able to satisfy the customer's inquiry from the existing answers database, she created a cross-reference in CMS and closed the complaint record. If not, she called on second-level support, which lead her into yet another application, the fourth, on her desktop. This was the company's standard IBM Lotus Notes® e-mail application. Maria then cut and pasted problem details from CMS into an e-mail and agreed with the customer to get back to him or her. It was up to Maria to make sure that the response she received from second-level support was conveyed to the correct customer in a timely manner, either by phone or by e-mail.

As can be seen clearly in this scenario, CSRs faced a number of challenges:

- *A disorganized and cluttered desktop made it difficult to keep track of tasks in hand.*
- *Multiple, different user interfaces increased training costs and slowed down user responses.*
- *Cutting and pasting between different application windows was time consuming and error prone.*
- *Different levels of integration between applications caused confusion.*
- *A lack of workflow support meant that agents had to keep track of long-running tasks themselves.*

As a result, CSRs were far less productive than they could have been and were compelled to invent ad hoc processes to manage their work. Consequently, the company faced significant challenges in expanding the CSRs' role to include cross-selling and revenue generation.

Highlights

The Complaints Management System was originally designed so that it fully integrated all the applications the CSRs needed.

As a result of business- and technology-driven change, CMS had lost much of its integration, and automatic exchange of information was replaced by manual transfer by CMS users.

The legacy system architecture

As previously mentioned, the original design point for the Complaints Management System was that it should be a fully integrated application serving all the needs of CSRs. By the nature of their job, agents require access to a number of data sets created and owned by other company departments, such as product data, problem resolution information, customer contact information and orders/purchase data. CSRs also generate information themselves: some product resolution information as well as customer contact logs. CMS had been designed to receive nightly copies of slowly changing information such as product and problem resolution information and to store it in its own server. Customer and orders/purchase data were to be accessed directly through a custom application programming interface (API) to the order entry system. Even e-mail was fully integrated into the original concept of CMS.

As a result of business- and technology-driven change, however, CMS had lost much of its integration. The introduction of Lotus Notes software meant that the built-in CMS e-mail interface fell into disuse as the users received a full-function Lotus Notes environment. As GReP had moved to standard packages—an SAP system for accounts management and IBM WebSphere Content Discovery Server software[1] for access to answer documentation—it had become increasingly difficult to migrate and maintain the file transfer and API interfaces. Automatic exchange of information was replaced by manual transfer by CMS users, and use of full-function SAP and WebSphere Content Discovery Server user interfaces became the norm. This is shown in Figure 3 (see page 10), which depicts the system architecture immediately before the decision to implement an SOA.

While the level of disintegration shown here is somewhat extreme, the vast majority of companies that have developed integrated applications in any area of business have experienced similar problems. The reasons fall into two broad categories—business change and technology evolution.

Highlights

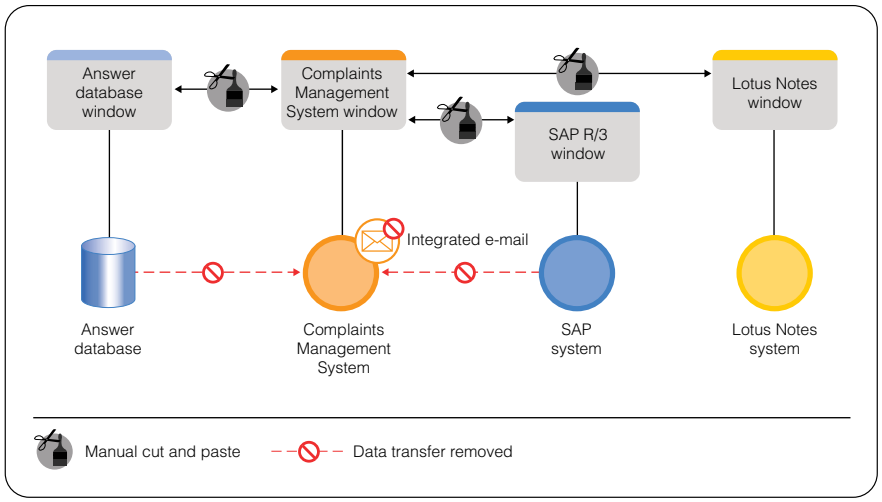

Figure 3. Complaints Management System legacy architecture

Business changes that disrupt integrated applications include the following:

Disrupting business changes can include mergers and acquisitions; new business processes brought about by marketplace needs; outsourcing and partner relationships that require new interfaces; and the fast pace of On Demand Business.

- *Mergers and acquisitions disrupt existing applications and interfaces, often beyond the capacity of the IT department to respond in a timely manner.*
- *Changes in business processes in response to marketplace needs cause new integration points to be required and old ones to be made obsolete.*
- *Outsourcing and partnerships cause functions included in integrated applications to be replaced by interfaces to external IT systems.*
- *On Demand Business, with its rapid rate of change, cannot easily be met by the necessarily slow maintenance cycles of an integrated application.*

Highlights

Technology changes that can cause disruptions include introduction of packaged applications that disrupt interfaces with integrated applications; new hardware platforms, operating systems, databases and middleware; and paradigm shifts such as the change from centralized computing to client-server systems.

Similarly, evolution of technology affects integrated applications:

- *Packaged application implementation, at a minimum, disrupts interfaces to an integrated application and, in the extreme, may replace all or part of it.*
- *Changes in hardware platform, operating system, databases or middleware can all cause significant disruption to integrated applications.*
- *Paradigm shifts such as the change from centralized computing to client-server may shift the fundamental design point of an integrated application.*

As a result of these pressures, CMS, in common with many integrated applications throughout the industry, had become largely dysfunctional. What was once a highly integrated system architecture had become disintegrated and was held together by the end users themselves. This was clearly unsustainable, and it was here that an SOA could help.

Customer care the SOA way

The traditional approach to a disintegrated environment doesn't address how to prevent the system from disintegrating again, because the original hardwired interfaces are replaced by new, equally hardwired ones.

When faced with a disintegrated environment such as the one we've described, the traditional approach is to examine the architecture of the system and recognize that the key issue lies in the lack of automated interfaces between the systems. The IT department then begins designing and building a new set of batch and real-time interfaces. While this may address much of the issue, in reality, it brings us back only to the state of the system before it began to disintegrate. It doesn't address how to prevent the system from disintegrating again, because the original hardwired interfaces are replaced by new, equally hardwired ones.

A service-oriented architecture provides a new way to think about the problem. It forces IT to consider the situation from a business perspective rather than a technological one. That is not to say that the technological viewpoint is unimportant, for, as we shall see later, modern technology offers many

Highlights

opportunities to support the business in new ways. However, starting from a business perspective and a service orientation allows IT to ask the key questions: what are the users actually trying to achieve here, and how can we help them to achieve their goals more productively?

Business services and user roles

Identifying what the users are trying to achieve from a business perspective involves two main areas: determining the services and creating groupings of these services that users perceive are closely related to one another.

Identifying what the users are trying to achieve from a business perspective is an iterative process that comprises two main areas:

- *Determining the services—well-bounded and repeatable business tasks—performed by users*
- *Creating groupings of these services that users perceive are closely related to one another in roles that users play at different times during their day*

This process of identifying services and roles can occur at different levels in a business. It can be done at the enterprise level in order to reengineer an entire business or at a divisional level to reassess the application portfolio and needs of a business area. Such analyses, although valuable, can prove quite time consuming. In our scenario, however, we were operating in the relatively confined boundaries of the customer care department, so we could expect the analysis to be much faster and more painless, leading to the identification of a number of distinct services and user roles. For the purposes of this sample scenario, we have defined two user roles supported by six services.

For the purposes of this scenario, we have defined two user roles supported by six services.

The primary role played by a CSR is contact management. In this largely reactive role, the agent handles incoming calls, identifies customers and their entitlements and answers simple inquiries. The second role is solution research, in which the agent proactively engages in creating new answers, interacts with

Highlights

second-level support and reengages with the customer to validate solutions. On a daily basis, the CSR work occurs mainly in the contact management role. There are two reasons for switching to the second role: when an answer to the inquiry cannot be found and the problem must go to second-level support, and when second-level support comes back with a new solution and the CSR must reconnect with the customer.

When, as GReP intends, CSRs take on a formal cross-selling responsibility, the services associated with that function will be grouped together in a third role.

Each role is composed of a number of services, some specific to a role and others that appear in multiple roles. Services are usually surfaced to the users as portlets in a portal-based environment. The following table shows the services and their corresponding roles.

Each role is composed of a number of services, some specific to a role and others that appear in multiple roles.

Service	Role
Inquiry handling	Contact management, solution research
Answer search	Contact management, solution research
Customer search	Contact management
Product inquiry	Contact management
Collaboration	Solution research
Action item notification	Contact management, solution research

The inquiry handling service is central to both roles. Through this service, customer calls are tracked and documented, acceptable answers are recorded and calls finally are closed.

The answer search service provides keyword-based search of the database of existing solutions.

Highlights

The customer search service accepts a customer name as input. When the customer name is not unique, the service provides a list of matching customer numbers and addresses, allowing the search to be further refined. When a unique customer is known or discovered, the product inquiry service provides a list of products owned by that customer and their warranty statuses.

The collaboration service provides e-mail, instant messaging support and access to contact lists.

Finally, the action item notification service provides the CSR with information about actions that are required. The most common use of this service is to inform the agent that an answer has been devised for a problem previously escalated to second-level support and that the customer should be contacted again.

We must answer the question: how can we help users to achieve their goals more productively?

Composite applications, templates and workflows

The second question we posed earlier was: how can we help users to achieve their goals more productively? At an architectural level, this is achieved by creating templated, composite applications and by enabling workflows within these applications.

At an architectural level, this is achieved by creating templated, composite applications and by enabling workflows within these applications.

Simply put, a composite application aggregates a set of existing components into a single, coherent entity for end users. Clearly, given that it's based on user perception, the boundaries of this single entity can be somewhat arbitrary. In our scenario, we've decided that all of the customer-facing and problem-solving functions performed by a CSR constitute a single composite application, based on the fact that the individual components and their interactions implement a set of interrelated business logic for the user.

Business logic effectively exists at two levels in SOA: in the services themselves and in the linkage between them. Business logic within services is of a more fundamental and unchanging nature than that found between services. For

Highlights

example, the customer search service incorporates the standard and necessary business logic of having a unique identifier for each customer. A service that changes business information often includes substantial business logic. In our case, the inquiry handling service contains logic to create new inquiry references, extract keywords from complaints and so on.

The business logic between services is usually known as workflows. Workflows are more flexible than intraservice business logic. They need to be, because it is here that much business innovation and productivity improvement take place. Flexibility to change workflows is a core SOA value. In our call center application, the workflows are rather simple and standard, the main one being the process by which inquiries are escalated to second-level support, the results are passed back to the CSR and from there are passed on to the customer. To improve customer satisfaction, GReP may decide that for high-priority problems, second-level support should close the problem with the customer directly. Such a change in the workflow can be implemented by business users themselves using modern portal workflow capabilities.

The business logic between services is usually known as workflows.

Flexibility to change workflows is a core SOA value.

When GReP introduces the new cross-selling role for CSRs, this is likely to be a separate composite application, rather than an extension to the workflow in the existing application. This is because this new function is more closely related to other selling functions in the company and can be easily based on what has been previously developed for the sales organization. Templates are the SOA concept that enables such reuse of composite applications. A template describes a composite application in an abstract form, including the services it uses and how they are linked by the business logic. The template is an XML file containing meta information about a class of similar composite applications. Once a template has been defined and stored, it is a relatively simple

Highlights

The new SOA is well structured and layered, separating the business view and the user's view of the business from the supporting IT components.

matter even for a business user to instantiate a new, related composite application by modifying parameters of the services and workflows between them. In many cases, this can remove the bottleneck of IT resources to create new, innovative business functions. Of course, in other cases, new services are needed or a more complex change to the business logic is needed; in such cases, the IT department will need to be involved.

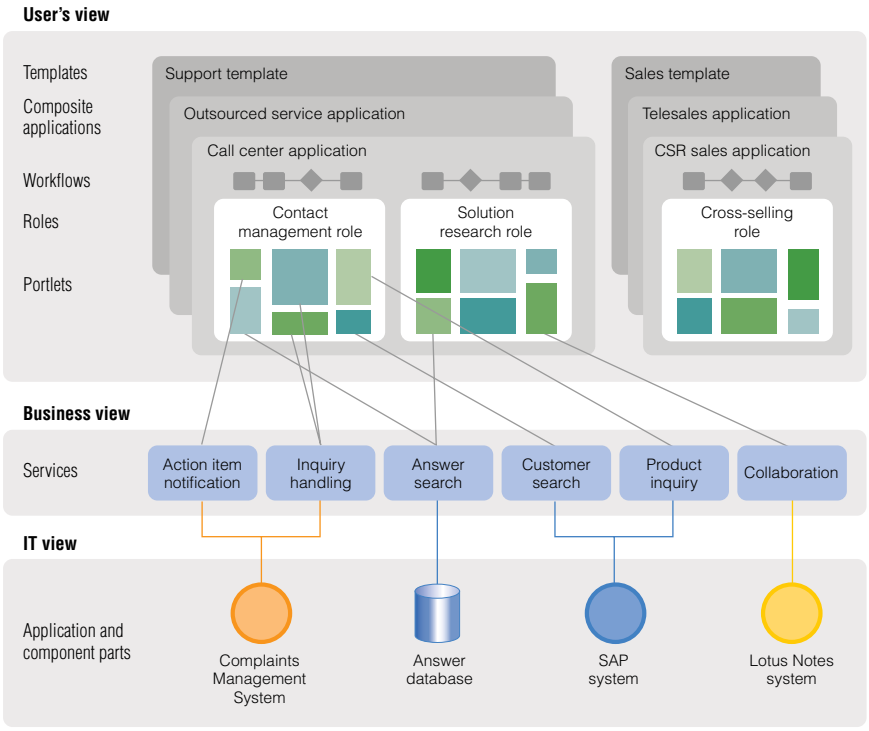

Figure 4. SOA-based call center architecture (not all service linkages are shown)

Highlights

The manual transfer of information between applications is incorporated in the composite applications and their workflows; prior IT architectural approaches hardwired the links between the applications and components at the IT level.

SOA system architecture

Figure 4 shows the new SOA-based view of the call center environment and illustrates the architectural aspects discussed above. Of particular interest is that this new architecture is well structured and layered, separating the business view and the user's view of the business from the supporting IT components. When this is done, the business processes and the flexibility to change them become part of the user's view—enabling enhanced productivity and agility as end users can adapt application behavior and flow to suit their own working patterns. The manual transfer of information between applications (as shown in Figure 3's depiction of the legacy architecture) has been incorporated in the composite applications and their workflows, where it can be managed but remains flexible. In prior IT architectural approaches, and indeed as the original GReP call center application had been designed, these links were usually hardwired between the applications and components at the IT level.

Implementing SOA using the IBM portal and collaboration product set

Having decided on the overall architecture, the next step was to choose the product set and implementation. As previously stated, our approach started from the end user's viewpoint. Therefore, the first question was how the roles and services described above could be presented to the user. IBM WebSphere Portal Version 6.0 software has been designed to be the front end of an SOA architecture and was the clear choice to deliver the above architecture.

A role usually corresponds to a tabbed page or a set of tabbed pages in a portal, and as a general rule of thumb, each identified service can be associated with the portlet.

Using IBM WebSphere Portal software

Figure 5 (see page 18) shows how the services associated with the contact management role have been mapped to the portal screen. A role usually corresponds to a tabbed page or a set of tabbed pages in the portal, and as a general rule of thumb, each identified service can be associated with a portlet. On this screen, we see a portlet associated with action item notification (top left), answer search (bottom left), customer search (top right) and product inquiry (bottom right).

Highlights

For the user, the first thing to notice is the common appearance of all the portlets, although their content and interactions are with three very different applications. The unique, disparate client-server interfaces have been replaced by a common look and feel shared by all the elements of the portal. For the user, there is one common interface to all applications, and for the IT department, the costs of user training and support have been dramatically lowered.

The resulting view is considerably less cluttered and less daunting than the legacy system shown in Figure 2 (see page 7). Each service has its place, and services not being used in the contact management role are hidden. However, the portal approach does allow users the flexibility to change the layout of their screens if they wish. The extent of customization allowed can be set by the administrator of the system, based on the end user's needs or skills. Thus, more experienced agents can choose always to display certain portlets in positions that suit their way of working, while junior agents might have less flexibility, so they can focus on learning to use the applications.

Each service has its place on the screen, and services not being used are hidden.

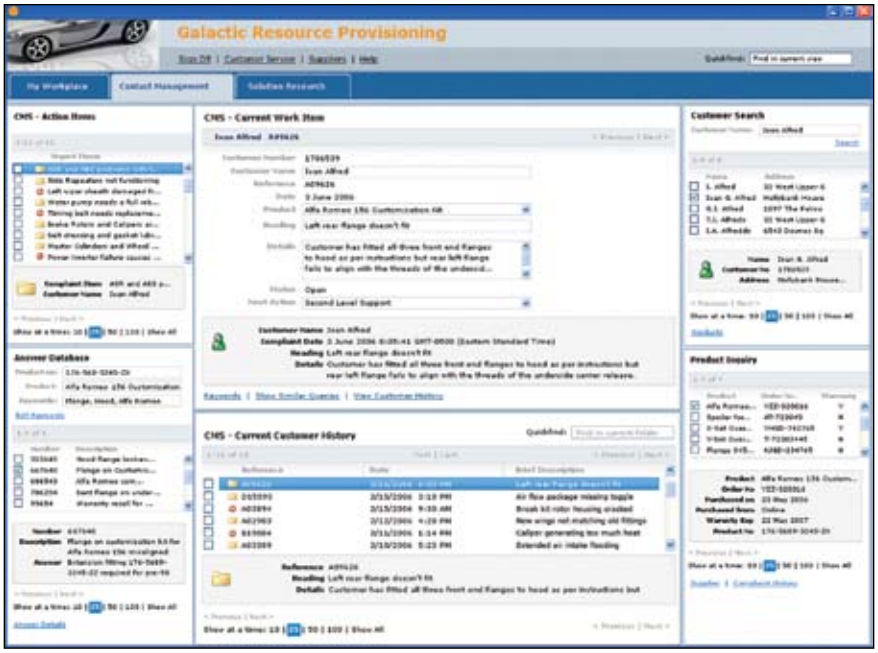

Figure 5. SOA customer care, contact management role

Highlights

"On-the-glass" integration between portlets allows the contents of fields in different portlets to be linked together, so information in one portlet becomes automatically available in another, obviating the need for cutting and pasting on the part of the end user.

The two center portlets are both associated with the inquiry handling service. This service illustrates that a one-to-one correspondence between service and portlet is not always necessary. In the case of this service, the amount of information available from it is too large to comfortably fit into a single portlet, and so the top center portlet displays the key information about the current work item—information that is always required—and the three buttons at the bottom of the portlet allow additional information to be displayed in a second portlet, as the user desires. The bottom center portlet shows the history of complaints from this customer, when the user clicks the history button. Other options that can be shown here are the keywords that the system has identified for the complaint, allowing editing by the user, or a listing of similar (based on the keywords) complaints that have been previously closed.

These three alternative portlets behave as functions that can be called from the current work item portlet. They have been implemented so that they are effectively subservices of the inquiry handling service, but if there was a case for reusing them more broadly or modifying their behaviors, they could be "promoted" to be full-fledged services in this application.

One of the most obvious and important benefits that WebSphere Portal software provides to the user is integration between the portlets. This "on-the-glass" integration allows the contents of fields in different portlets to be linked together, so that information in one portlet becomes automatically available in another, obviating the need for cutting and pasting on the part of the end user. In Figure 5, therefore, we see that customer name and number are synchronized between the CMS current work item, the CMS history and the SAP system-based customer search and warranty inquiry portlets. Similarly, the keywords identified by the CMS application are made available automatically to the answer search portlet.

Highlights

By wiring portlets together, either programmatically or by definition in Web Services Description Language (WSDL), changes in properties in one portlet are automatically communicated to other related portlets.

All of these functions are implemented through the cooperative portlets and portlet wiring introduced in WebSphere Portal Version 5 software. By wiring portlets together, either programmatically or by definition in Web Services Description Language (WSDL), changes in properties in one portlet are automatically communicated to other related portlets. Depending on a user's role and privileges, he or she can change the extent to which information is passed between portlets. However, in this example, such freedom was considered to be beyond the business needs of the end users and was confined to the application developers.

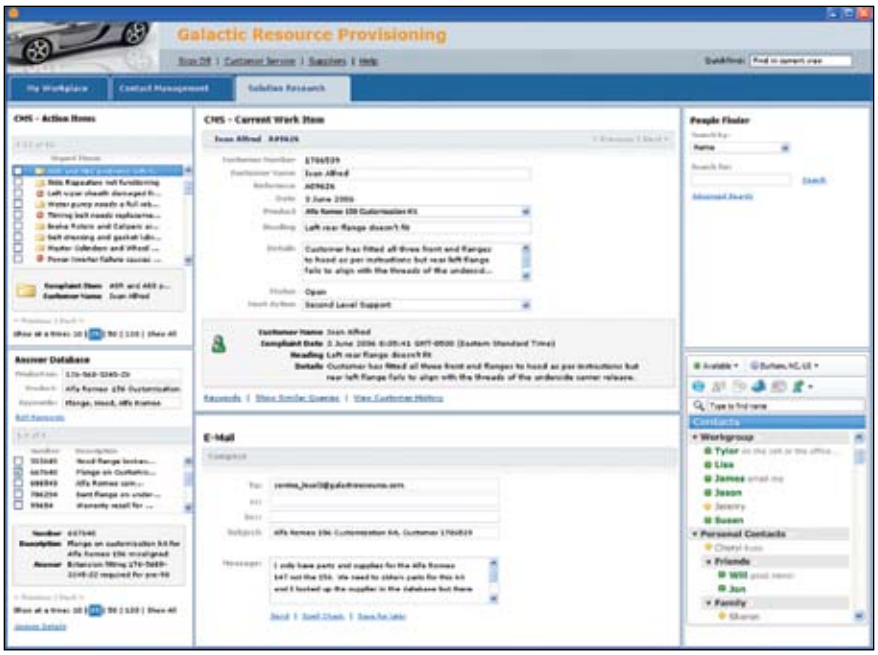

Figure 6. SOA customer care, solution research role

Highlights

A key aspect of SOA is the explicit inclusion of people in the business processes, as implemented in the IT systems; such integration enables new ways for people to interact and removes bottlenecks in the business processes by facilitating direct person-to-person communication.

Turning to the solution research role shown in Figure 6 (see page 20), we can see that the same principles apply. On this screen, three of the portlets are the same as those provided in the contact management role. In addition, the collaboration service supplies three additional portlets: the portlets on the right, which provide a people finder and a contact list with multiple ways to communicate with the required person (via a right-click context menu), and the e-mail viewer/editor portlet, which is used extensively while operating in this role.

A key aspect of SOA is the explicit inclusion of people in the processes of the business as implemented in the IT systems. Such integration enables new ways for people to interact and removes bottlenecks in the business processes by facilitating direct person-to-person communication (for example, using instant messaging). In our scenario, the introduction of IBM Lotus Sametime functionality and its close integration in the portal environment provides benefits in improved collaboration and faster responses. At the heart of these improvements is the contact list portlet at the bottom right of Figure 6, where the online presence of the user's contacts is indicated visually. This presence awareness is carried over into other portlets as well, as seen in the CMS current work item portlet in Figure 5, and allows the CSR to involve available people via instant messaging when a problem is urgent.

A new function delivered in WebSphere Portal Version 6 software is worth noting here, although it is not shown in our scenario. This is the introduction of true workflow function within the user interface. Notice that in the solution research role, there is an implicit workflow in which the CSR hands over a customer's problem to second-level support and, at a later stage, receives an answer back, which then must be communicated to the customer. The workflow function in WebSphere Portal Version 6 software enables business users

Highlights

to formally define this process and its decision points, automate tasks such as sending e-mails and track work items through the defined workflow. This new function will allow GReP to ensure that customer problems are not lost between departments and will improve response times to the customer when the CSR is busy.

Portlets—borrow, buy or build?

The first choice for sourcing a portlet always is to borrow it.

Having chosen WebSphere Portal software as the platform, we now examine the sources of the various portlets. There are three basic choices, and we see all three in this example.

The first choice for sourcing a portlet always is to borrow it—that is, to reuse a portlet or a service that already has been developed internally. In our scenario, this is the case for the answer search portlet. This application had previously been developed as a lightweight customization of WebSphere Content Discovery Server software using a browser-based, portal-ready interface. With only the most minor modifications to enable linking of the fields, this application was reused as a portlet in the new environment.

The next choice usually is to buy a portlet.

The next choice usually is to buy a portlet. IBM, as well as many vendors, sells portlet-ready interfaces to a wide variety of common applications. In our scenario, the collaboration portlets fall into this category. IBM Lotus Domino® and extended products portlets provide a set of ready-to-use collaborative portlets. The portlets include standard Lotus Notes and Domino features such as e-mail, calendar and scheduling, discussion, teamrooms and to-dos, as well as the Lotus Notes View portlet capability, which allows access to documents from any view of any Lotus Notes database. Three portlets are used in the scenario.

Highlights

The Lotus Notes View portlet can give users access to Lotus Notes databases using a Web browser, and the Lotus Sametime Contact List portlet can give users access to Lotus Sametime instant messaging.

Including IBM Lotus collaboration products

The Lotus Notes View portlet (bottom center, Figure 6) gives users access to Lotus Notes databases using a Web browser. This portlet can be configured to provide access to specific types of Lotus Notes databases. In this scenario, Lotus Notes View portlet would be used to display the Lotus Notes mail database for authenticated users in the portal. The people finder portlet (top right, Figure 6) can be used to locate people in the organization and to see detailed information about them, such as name, phone number, job title. Once found, a person is visible as a person link that indicates online presence and displays a menu of instant messaging and other options. The Lotus Sametime Contact List[2] portlet (bottom right, Figure 6) gives users access to Lotus Sametime instant messaging on a Lotus Sametime server with all Lotus Sametime users in the organization. They can engage in informal online business conversations with people, display only online names in the list, sort the list, modify status message or modify groups.

The integration of presence awareness and instant messaging is critical to our fictional business, because these new tools allow and encourage direct, real-time interaction between the first- and second-level support teams.

The integration of presence awareness and instant messaging is critical to the business, because these new tools allow and encourage direct, real-time interaction between the first- and second-level support teams. This reduces response time and increases customer satisfaction. The close coupling of Lotus Sametime functionality and the WebSphere Portal software-based environment allows users to easily move between the e-mail and messaging approaches as needed.

Another important trend in the provision of portlets is that vendors are creating customizable solutions for particular problem areas. By combining a set of portlets into a solution, vendors reduce implementation times and costs for customers. A good example is the IBM® Workplace™ *for SAP Software* offering that provides a full set of role-based, customizable portlets to access SAP

Highlights

functions, measure performance against objectives and link to a variety of collaborative functions that might be used by sales people. Such a solution could provide the basis for the cross-selling function envisaged in the future for CSRs.

The last choice is to build a portlet; this is generally the most expensive choice, but it may be possible for later systems to borrow any new portlets and thus defray the future costs of application development.

The last choice is to build a portlet, as this is generally the most expensive choice. However, one must always consider that the cost need not be borne only by the first project. In the spirit of reuse, it may be possible for later systems to borrow these new portlets and thus defray the future costs of application development.

IBM WebSphere Portlet Factory software

The simplest situation when building a portlet is when the required services are already exposed through an API to a legacy application.

There are different levels of complexity encountered in this process, depending on the nature and source of the services to be accessed. The simplest case is when the required services are already exposed through an API to a legacy application. This is the case with the SAP function required in this example. The SAP system provides Business Application Programming Interfaces (BAPIs) to query customer and product information. IBM WebSphere Portlet Factory software provides the functionality to easily and quickly build portlets and to interface with the SAP BAPIs. As seen in Figure 5, these portlets are specific to the needs of a CSR in accessing the SAP system. Rather than having to learn the entire SAP interface and navigate through it for every request, these very focused portlets enhance user productivity and reduce user learning curves.

The more complex case is when a system was originally designed as a fully integrated application and must be decomposed into independent, stateless and callable components, which can be accessed in the same way as services through portlet-based front ends.

The more complex case in our example is the Complaints Management System. CMS was originally designed as a fully integrated application and thus had to be decomposed into independent, stateless and callable components, which can be accessed in the same way as services through portlet-based front ends. Again, WebSphere Portlet Factory software is the tool of choice to create the portlets easily and rapidly. However, decomposing the old CMS application and mapping its components to the services identified earlier was where the real effort lay.

Highlights

When using legacy components, it is advisable to limit their scope so that they match the characteristics of the new, desired service as closely as possible.

The effort in decomposing an existing application depends on a number of factors, such as the language in which it was coded, how well structured it was and the extent to which different functions are independent of the order in which they are called. In general, newer applications built in more modern languages are easier to break into components. When using existing application functionality as a basis for an SOA service, some trade-offs in the definition of the service may be necessary, because of limitations in the input, function or output capabilities of the legacy component. When using such legacy components, it is advisable to limit their scope so that they match the characteristics of the new, desired service as closely as possible. This is done so that the underlying function can be changed or replaced at a later stage as cleanly as possible. Bear in mind, however, that such a later migration should only be undertaken as business needs and value dictate.

Conclusions

There are clear and well-defined benefits for both the user and IT communities in approaching an SOA implementation in the manner described in this paper. First, let's take a look at the new call center environment through GReP's CSR Maria's eyes.

The CSR's new desktop is considerably more structured and organized, helping to improve efficiency and shorten initial learning curves when new functions are introduced.

Her new desktop, as shown in Figures 5 and 6, is considerably more structured and organized than it was previously. Functions related to a particular set of tasks are grouped together, integrated to share state information and arranged predictably. Only function required at a particular time is exposed at that time. Functionality she never needs, such as the entire SAP interface to which she previously had access, has been removed. Thus Maria's day-to-day efficiency has been improved, and her initial learning curve has been shortened when new function is introduced.

Highlights

The definition of the CSR roles in terms of business services linked together into a simple set of workflows through cooperative portlets provides a significant productivity improvement for Maria. It also reduces the level of error introduced by retyping or cutting and pasting existing information between applications.

The efficiency provided by this contextual and focused interaction with the system has a less measurable but nonetheless real benefit for the business.

The efficiency provided by this contextual and focused interaction with the system has a somewhat less measurable but nonetheless real benefit for the business. Senior and skilled users like Maria now have more time to focus on the customer rather than the IT system, and it is in this improved and extended interaction that the seeds of innovation are found.

The IT benefits are even clearer: the componentization of function allows reuse of existing code and reduction in development costs, and loose coupling between components eases deployment and allows IT to be more responsive to changes demanded by the business.

The IT benefits are even clearer. The componentization of function allows reuse of existing code and reduction in development costs. Loose coupling between components eases deployment and allows IT to be more responsive to changes demanded by the business. In fact, the use of templating enables empowered business users to create and customize applications themselves, as and when the business requires it, reducing the extent to which IT is perceived as a bottleneck to change.

Because we've focused here on integration on the glass rather than deeper process and data integration, the introduction of the new system was relatively fast and reasonably low cost. This is not to suggest that such deeper integration can or should be avoided. Different processes will need different approaches, depending on the tightness of integration required. However, this lighter level of integration can be a very powerful pilot approach even in such tightly integrated cases.

Highlights

Our fictional company has taken the first steps in implementing a service-oriented architecture, has proven the value of an SOA and has demonstrated the way that users can become more productive and innovative in their daily tasks.

Because IT staff have spent less time in back-end integration, they have been able to prioritize new function and middleware that directly benefits the business. Real-time presence indicators and interactions between first- and second-level customer support teams enabled by the Lotus Sametime product and underlying infrastructure mean that new customer problems can be handled more quickly and with improved satisfaction levels. And although we have not described it here, the built-in workflow functions in the WebSphere product set will enable further efficiencies in this collaboration in the future.

By focusing first on the business services required by the call center and on how these services can be easily integrated and delivered to the users through IBM's portal and collaboration products, GReP has taken the first steps in implementing a service-oriented architecture. These first steps are both speedy and beneficial to the business and to IT. They prove the value of an SOA and demonstrate the way that users can become more productive and innovative in their daily tasks. From here, GReP can undertake the larger process of reinventing the way its employees work, sure in the knowledge that its understanding of the process is clear and its choice of middleware enables the business change the company requires. In the words of an old Irish proverb, *Tús maith, leath na hoibre*—A good beginning is half the work.

For more information

To learn more about how an SOA can help benefit *your* business, visit:

ibm.com/soa

To learn more about taking a people-centric approach to SOA, visit:

ibm.com/software/workplace/soa

Lotus Software
IBM Software Group
One Rogers Street
Cambridge, MA 02142
U.S.A.

Produced in the United States of America
09-06

---

1 IBM WebSphere Content Discovery Server software was based on the iPhrase OneStep Platform, acquired by IBM in 2005, and is now also known as IBM WebSphere OmniFind™ Discovery Edition software.

2 Formerly Lotus Instant Messaging Contact List

LOW10770-USEN-00