Enabling Enterprise Applications for Services

Kei Yuasa, Kannan Govindarajan, Harumi Kuno,
Kevin Smathers, Kevin Wilkinson, Umeshwar Dayal
Intelligent Enterprise Technologies Laboratory
HP Laboratories Palo Alto
HPL-2005-130
July 8, 2005*

methodologies for service-oriented applications, programming models for service-oriented applications, service development and maintenance

In this paper, we discuss techniques for supporting enterprise applications for business services. These applications can be quite complex, requiring support for regional variations in business logic, multi-step workflows, and long term transactions. Furthermore, such applications must coordinate the efforts of a mobile workforce that is distributed around the world, and integrate with client-side personal productivity tools, such as Microsoft Excel. In order to address these, and other, requirements, we employ a number of techniques to integrate the personal productivity application infrastructure with the business productivity application infrastructure. In particular, we use personal productivity applications to encapsulate the closure of a computational unit with all of the information needed to execute it, which enables us to carry business logic over to the client side, and then incorporate that encapsulation into the larger context of the business application. We have implemented a prototype, and discuss the application of our work in HP's outsourcing service business.

* Internal Accession Date Only
Approved for External Publication

Enabling Enterprise Applications for Services

Kei Yuasa , Kannan Govindarajan, Harumi Kuno,
Kevin Smathers, Kevin Wilkinson, Umeshwar Dayal

Hewlett-Packard Laboratories
1501 Page Mill Road,
Palo Alto, CA 94304 USA
{kei.yuasa, kannan.govindarajan, harumi.kuno,
kevin.smathers, kevin.wilkinson, umeshwar.dayal}@hp.com

Abstract. In this paper, we discuss techniques for supporting enterprise applications for business services. These applications can be quite complex, requiring support for regional variations in business logic, multi-step workflows, and long term transactions. Furthermore, such applications must coordinate the efforts of a mobile workforce that is distributed around the world, and integrate with client-side personal productivity tools, such as Microsoft Excel. In order to address these, and other, requirements, we employ a number of techniques to integrate the personal productivity application infrastructure with the business productivity application infrastructure. In particular, we use personal productivity applications to encapsulate the closure of a computational unit with all of the information needed to execute it, which enables us to carry business logic over to the client side, and then incorporate that encapsulation into the larger context of the business application. We have implemented a prototype, and discuss the application of our work in HP's outsourcing service business.

1 Introduction

Information technology has had a significant impact on both personal productivity as well as business process productivity in enterprises [1]. However, technologies that support personal productivity have evolved independently of technologies that enable business process productivity [2]. As sketched in Figure 1, such business process productivity applications have matured over the years from a simple two-tier client-server model using proprietary protocols for interaction to web-based applications and services that use industry standard protocols and encodings to communicate amongst various components of the enterprise system. Business productivity tools typically support complex workflows within and across functional organizations within a company, as well as across companies. Personal productivity applications, on the other hand, have evolved from a relatively simple set of tools to support tasks such as document publishing and simple financial analysis, to a framework that is capable of embedding a significant amount of custom logic.

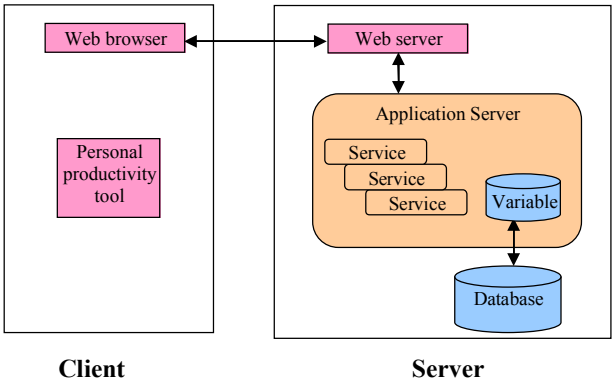

Figure 1 Current Enterprise Application Model

Our motivation for providing support for such applications arises due to our interest in providing infrastructure for enterprise applications for services businesses. We emphasize the distinction between "Web" services and "business" services [3]. Web services are performed by computers and typically implemented as software applications. Business services, on the other hand, are performed by humans and are typically implemented as processes that can span multiple teams and geographies. Because business services are people-intensive and subject to regional variations, personal productivity tools play a key role in their design and delivery. For example, we have found that costing of service components is often performed using tools like Microsoft Excel, a statement of work describing service level agreements is often a Word document, etc. The information used to produce such documents reflects local variations and originates from disparate data sources in the enterprise. Finally, because large long-term service contracts can be quite complex, sales and design teams of service providers have strong mobility requirements for being able to carry out work on the customer's site. In the remainder of this paper, we use the term "service" to refer to "business service."

The challenge is that global customers expect uniform service delivery across regions, and integrating local business processes that depend exclusively on personal productivity tools can be difficult. The traditional solution for centralized coordination is central business productivity tools with central business processes. However, a centralized solution would typically require users to learn a new interface technology with an associated learning curve, as well as a long development and deployment cycle. Furthermore, even if a solution is already deployed, employee turnover and merger/acquisition activity requires new employee training. Use of familiar, personal productivity tools mitigates these issues.

In this paper, we introduce some key mechanisms needed to support enterprise applications for business services, an emerging form of business application that blurs the traditional distinction between personal productivity tools and business productivity tools. We adapt the abstraction of closures from programming languages, and show how business application writers can generate closures on the 'server-side' as well as how these closures can be mapped to existing personal productivity tools on the 'client-side'. In addition, we are interested in specific steps in a workflow that have certain properties. Specifically, these steps must be atomic (indivisible) with respect to the rest of the workflow. I.e., it can be viewed as a black box that takes a set of inputs and produces a set of outputs in a single step (no partial completion). Another property is that it must be isolated, in the sense that any other concurrent step in the workflow cannot affect the closure (the inputs or outputs). Hence, we avoid traditional problems with synchronizing caches. We wanted our techniques to be independent, and span the multiple middleware infrastructures that exist in most enterprises. We accomplish this by naming all the relevant entities and providing a mechanism for associating named entities in a closure with entities in a personal productivity tool or a centralized workflow system.

The rest of the paper is organized as follows. We begin in Section 2 by outlining a problem scenario. In Section 3, we outline traditional approaches and discuss why they don't address some of the issues we need to address on our context. In Section 4, we introduce our notion of closure and discuss a specific implementation in Section 5. Section 6 contains our conclusions and potential directions for further research.

2 Problem Scenario

Let us consider the specific case of a solution architect trying to design the technical solution in support of a large outsourcing deal that involves a global customer outsourcing desktops, network and servers in their IT infrastructure. The solution architect uses MS Excel to estimate the various regional costs, and MS Word to write the statement of work, capturing deliverables. These deliverables will be passed to regional and global delivery teams, who will use it to create and execute setup and ongoing management plans for service delivery. Because of the potential number of regional workflows involved in this process, it is not practical to install any workflow-specific installations on the solution architect's client machine.

The solution architects have to gather a lot of information about many different aspects of the customer's IT infrastructure in order to create a technical solution that meets the expectations of the customer while taking advantage of the core capabilities of the service provider. For example, suppose the customer has 10,000 employees in the Americas, 5,000 in Europe Middle-East and Africa (EMEA), and 5,000 in Asia Pacific. The cost and delivery processes for desktop support may vary significantly by region. It may be standard for the service provider to promise a 5 day order to installation time in the Americas and EMEA, but not in Asia Pacific.

The information used for designing the service solution comes from a number of sources. Inputs to costing may come from information gathering sessions held at the

customer site, whereas the cost formulas might come from the service provider's regional sources. The results from costing activity may flow into the deliverables, and be combined with both global and regional processes. For example, it may cost more for the provider to meet a global service level agreement of a 5-day order-to-installation time because Asia Pacific will require custom processes.

The solution architect may also want to cross-check the assumptions captured in her personal productivity tool with the capabilities of the delivery organization before committing anything to the customer. This may require coordination with an enterprise-wide system.

In summary, the main goals we identify here are:

1. Support for disconnected operational steps in complex workflows
2. Ability to modify regional business logic locally
3. Ability to seamlessly transition between disconnected and online modes of operation without losing any information
4. Have minimal additional installations to support disconnected operation

3 Related Work

We recognize that much prior work focuses on content management *or* mobility solutions *or* enterprise information integration *or* leveraging personal productivity tools in Web applications, but we believe we uniquely address all of these areas at once.

Content Management

Halevy et al [4] observes that there is a large pool of enterprise information contained in schema-less and schema-poor business documents such as Microsoft Word and Excel. Content Management Systems (CMS) have been used to store and exchange such documents, typically in a coarse-grained manner since the content is largely opaque. However, the introduction of XML formats for business documents has made it feasible to extract content from these documents and support collaboration. There has been work in automated metadata extraction and annotation for such documents [5,6] and tools are becoming available to map the fine-grained content of such documents into content management systems (e.g., content management support in DB2 [7]). Our key insight is that for our domain, the important applications to be integrated are personal productivity tools, e.g., Excel and Word. By providing interfaces to those tools, we close the loop and round-trip content from one data source to another, e.g., relational to Word or Excel to relational.

Mobility Solutions

We acknowledge a long history of work in disconnected operating systems [8], disconnected file systems [9], disconnected Web clients [10], and mobile databases [11], and mail systems such as IMAP and POP3. These enable information access by supporting disconnected and weakly connected operation, typically by requiring specialized software to be installed on the client machines and devoting a good deal of attention to cache consistency issues. For example, the Coda disconnected file system extended the Andrew File System to offer client read-write access to shared information by mobile and static users by supporting mechanisms such as server repli-

cation, isolation-only transactions, translucent caching, and hardware surrogates [8]. Similarly, systems supporting disconnected Web clients typically install a proxy Web server on the client and load it with a cache of Web pages and thus only support static content [10].

Commercially available workflow systems also support disconnected operation. However, workflow systems are inflexible as custom programming is typically required to alter the computation of the functions.

Existing mobility solutions don't address the needs of Enterprise Applications for Services. They often require dedicated clients, and do not facilitate the integration of federated data sources or applications. In these systems, the problem of disconnected operation is simpler because the functionality provided by these systems is well-defined and understood. In addition, the problem we face is more complicated as it must provide multiple client applications for a wide variety of user communities.

Enterprise Integration

Enterprise Information Integration (EII) and Enterprise Application Integration (EAI) are complementary techniques that address different facets of the overriding enterprise integration problem. To the best of our knowledge, no existing work supports the disconnected operation or tight integration with personal productivity tools that our users require.

Web services provide some underlying mechanisms for integration, but our requirements are more dynamic. If Web services supported the programmatic generation and integration of mobile, serializable services, then we could have used Web services to implement our requirements. However, because Web services separate data-encoding/transport (XML/SOAP/HTTP) from operations (WSDL), significant programming may be required to achieve the semantic coupling between data and operations that we need.

Integrating Personal and Business Productivity Tools

Microsoft has already integrated their personal productivity tools with their operating system. They provide API's that let applications programmatically invoke Office tool functionality. SAP's Mendocino project leverages Microsoft Smart Client technology to allow SAP applications to access Outlook calendars for a handful of business processes [12]. To date, SAP and Microsoft have not focused on creating general mechanisms for interoperability between arbitrary business applications and personal productivity tools.

ReportingEngine's Formula One e-Spreadsheet Engine provides a Java API that enables toolset J2EE projects and applications to read in Excel spreadsheets, update data values and formulas at the cell level, force recalculations, and export data as Excel workbooks, individual values from cells, XML, or to a database [13]. Similarly, the Microsoft .NET Visual Studio Tools for Office (VSTO) enables .NET applications to access the values of fields in personal productivity documents (e.g., Excel or Word documents). However, neither of these systems offers a framework for addressing data integration and contextualization issues.

4 Our Solution

Our solution had to provide a way for data and logic to that operates on the data to be encapsulated in a simple manner that could be transported. We therefore adapted the notion of closures from programming languages for this purpose. In this section, we outline the general technical principles that we base our solution before describing the specific implementation in the next section.

4.1 Approach

A closure is an abstraction representing a function, plus the lexical environment in which the function was created [14]. Here, we use a set of calculation and related data for a web service for the closure so that it can be extracted from the server and can be executed in the client environment.

Figure 2 shows a conceptual structure. A closure is built with calculation (business logic) and data used in a computational service. This closure is an independent environment for executing the computation. It can be executed both in server and client sides. Once a closure is extracted from the server and executed in the client, the connection to the server is not needed. The user can update data on the closure while disconnected, and later reconnect to the server to restore the data.

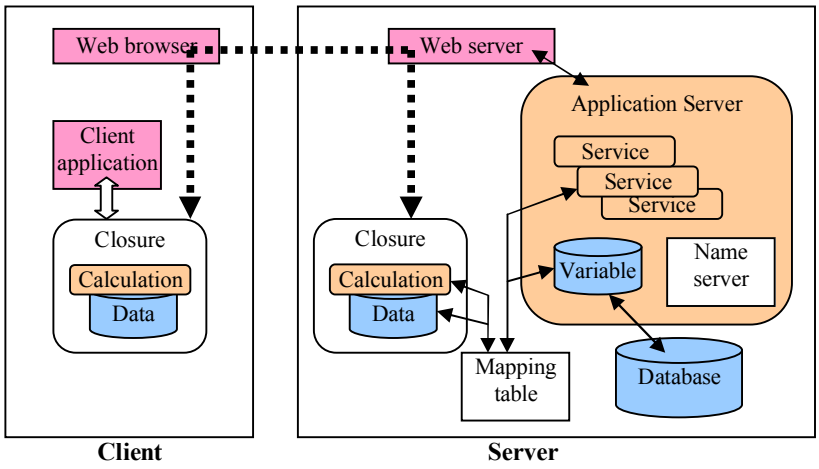

Figure 2. Closure

4.2 Closures for Enterprise Applications for Services

In Web applications, the variables that hold various pieces of relevant information are typically not accessible without explicit business logic support. While there is extensive support for persisting values through notions such as entity beans in J2EE, or

persistence services in .NET, we need a generic mechanism to identify various pieces of information that may potentially need to be transferred from the server side to the client side. We use the notion of 'variables' to denote such entities in the abstract. These variables are mapped both to the 'back-end' of the server system as well as to the data elements in a closure that can potentially be shipped to a client machine.

Figure 3 shows a simple example of cost calculation for an IT outsourcing service. IT services have large structures with many modules. The "Global PC management" service has "Hardware Procurement" and many other modules. Both the service and modules have lots of fields to configure. Each instance has a unique "Name" field. E.g., there is only one service object whose name is "Global PC management" in one name space, which depends on a server name, user name, the customer, and other contexts. Other fields are either input parameters or output parameters. Gray boxes are for inputs and white boxes are for outputs. The user (solution architect) provides values for the input parameters, and executes a calculation to get output values.

For example, in Figure 3, a field "Seat_Count" means the number of users for this service. There are several regions for the customer company so that the total of the seat counts for all of the regions is assigned to "Seat_Count". Each module has its own input and output parameters and uses the parameters of itself and "Global PC Management" service to calculate cost for itself. Finally, the costs for modules are added up to calculate the total cost for the service. The cost per seats can be calculated by dividing the total cost by the number of seats.

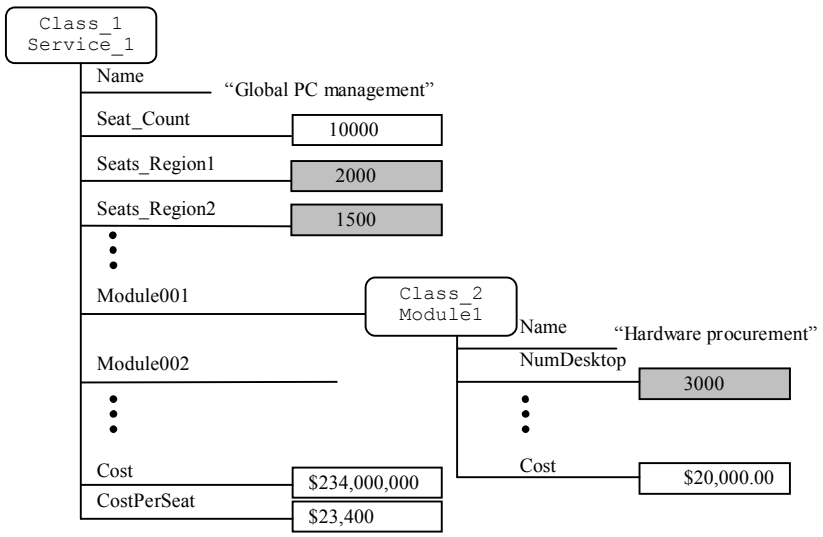

Figure 3. Template, Instance and Field

Our name server utilizes Resource Description Framework (RDF) [5] notation to describe the class fields. An instance of services becomes a subject resource identified by a URI. In this example, the instance of "Global PC Management" has a name:

```
"http://managedservice.hp.com/Service/Global PC Man-
agement"
```

Here, the prefix (`http://managedservice.hp.com`) is a URL of the server. The rest is a combination of class name "Service" and the name of its instance. This pair is registered in the name server when this instance is created.

Fields are identified as RDF properties, and their values are referenced using XPATH-like expressions. E.g., in Figure 3, the value of the property `NumDesktop` for the resource Module001 would be referenced as `Module001/NumDesktop`, and its value would be `3000`. The calculations can be defined as relationships among the fields and sub-fields within this service structure. For example:

```
Seat_Count = Sum(Seats_Region1, Seats_Region2, …)
Cost = Sum(Module001/Cost, Module002/Cost, …)
CostPerSeat = Div(Cost, Seat_Count)
```

The values of fields are stored in the database and used in the calculation by the computational service in the application server. In order to generate a closure from the data in the fields, a name server in Figure 2 manages the subject names and property names. The name server translates XPATH-like expressions to values in the database.

5. Implementation of Closure

We have developed a prototype application, called the Service Configurator, for supporting the design of service solutions [15, 16]. We leveraged the Microsoft Smart Client extensible platform's Offline Application Block [17, 18] to implement a closure mechanism that uses Excel spreadsheets to maintain data and calculations in a single file. Calculations are defined as macros that take input from a set of cells and produce the values on other cells. These macros can be implemented in any .NET supported language.

5.1 Defining Computational Services as Excel File

Here, a business manager tries to define a computational service for calculating a cost for the service described in Figure 3. First, he/she logs in to the administrator page of the Service Configurator and invokes "Define new Service" command on an object "Global PC Management". Then, a template Excel spreadsheet in Figure 4 is downloaded. This file includes "Enumeration", "Parameter" and "Calculation" sheets. The "Enumeration" sheet includes all strings of enumerations used in the object. The cells in this sheet are used as content of dropdown lists in "Parameter" and "Calculation" sheets. The table in the "Parameter" sheet has three columns; "Property name", "Cell name", and "Mode". At this point, all properties are listed up in "Property name" and other columns are blank.

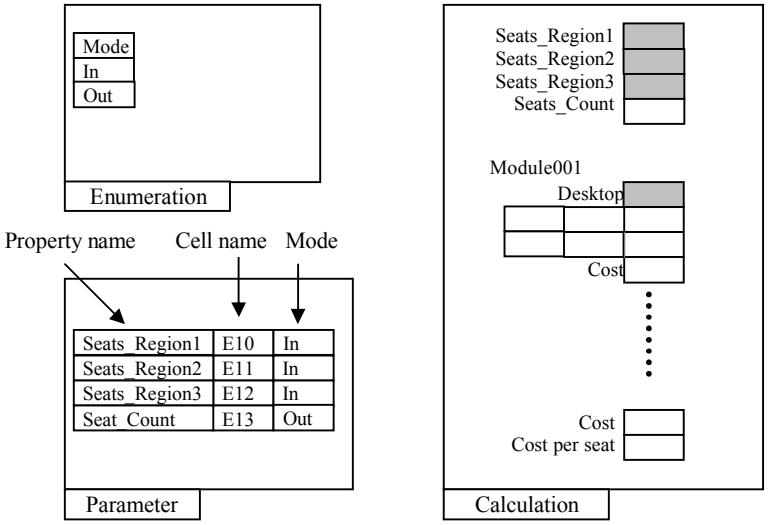

Figure 4. Structure of Excel spreadsheet for service definition

The business manager defines the business logic in "Calculation" sheet. For example, the properties Seats_Region1, Seats_Region2, Seats_Region3 are arranged to cells E10 to E12, and Seat_Count is to E13. As Seat_Count is a total of regional seat counts, a formula "=SUM(E10:E12)" is defined as content of a cell E13. Likewise, the cell for Cost is defined as a summation of cells for cost of modules, and the cells for "Cost per seat" is defined as the cost divided by Seat_Count. Then, the business manager fills in the "Cell name" columns in the "Parameter" table and sets the mode of the cell to "input" or "output". This table will be used as a mapping table later.

When the template spreadsheet is completed, the business manager saves it to a local file, reconnects to the administrator page, and uploads the spreadsheet to the server. This Excel file is examined by the application server for the types of input cells, and input/output modes for cells. If it passes, the Excel file is stored in the server and released as a template Excel file for a new service.

5.2 Use of Excel-Based Closure

Figure 5 describes the mechanism of Excel-based closure. The new computational service can be used in both online and offline modes. In the online mode, the template file is executed in the server side. An end user (solution architect) sets values to the input parameters in the web browser, and invokes the calculation. An Excel application is invoked in the server side. The template file is loaded, and the "Parameter" sheet is read to create a mapping table. This table is used to store values of data to the cells. The calculations defined in the spreadsheet are executed, and the output parameters are read by using the mapping table.

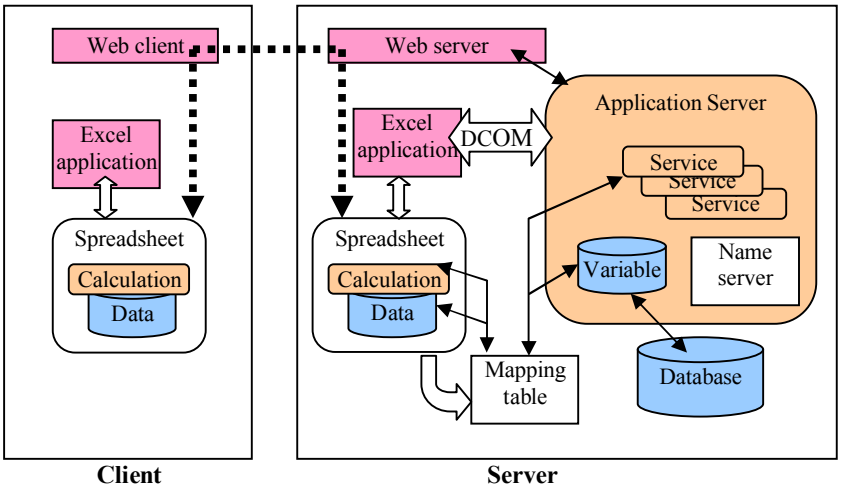

Figure 5. Implementation of closure using Excel spreadsheet

In the offline mode, after the spreadsheet is filled with values in the server, it is saved as an Excel file and downloaded to the client. This includes the service calculations as formulas and current values of data in cells. The user can use local Excel application to open this spreadsheet, change input parameters, and get results in the output cells. This service can be executed in the client environment without the connection to the user. After the data input is completed, the user comes back to the web server and uploads the Excel spreadsheet. On the server, Excel is invoked again, loads the uploaded file, picks the input values, and restores them to the database.

The Excel Application in the service is invoked by the application server using DCOM (Distributed Common Object Model) [19] interface.

5.3 Authorizations

As Excel support security functions, we can use them for user's privilege. In our example, there are three types of users: an administrator who designs the Department section class, a business manager who defines a service on the class, and an end user who uses the service. The administrator can change the data structure. The business manager cannot change the data structure but can change the service on it. The regular user can update only input parameters.

When the business manager defines the service, "Enumeration" sheet in the Excel file is locked and cannot be changed. The property name column of the table in "Enumeration" sheet is locked so that the business manager can only change the cell name and mode columns of the table.

When the spreadsheet is registered as the service and provided to users, all of the spreadsheets except for input parameter cells are locked so that the end user cannot change the service calculations.

6 Conclusions and Future Work

In this paper, we describe how the emerging field of enterprise applications for services motivates the need to integrate personal productivity application infrastructure with business productivity application infrastructure. We considered a use case from the realm of applications supporting business processes in the services business to show the practical relevance of such integration. Some of the desired characteristics of the solution we were aiming for were:

1. Support for offline operational steps in complex workflows
2. Ability to modify regional business logic locally
3. Ability to seamlessly transition between offline and online modes of operation without losing any information
4. Have minimal additional installations to support offline operation

Our solution relies on the notion of closures where we encapsulated a computational unit with all the information required to execute it into one entity. We were then able to move this closure across computational environments. We also outlined how we could implement some of these ideas using commonly available personal productivity tools such as Excel. Spreadsheets provide good structures to define closures because they can hold data and calculation together, and can be saved to files and exchanged between the server and client. Because our end users have Microsoft Excel installed and know how to use it, there's no need to install, learn, and maintain custom client software.

Some of the key directions for future work are the following. Currently, our implementation of closures is static in that the values are loaded once and used multiple times. We may want to consider supporting values that are updated before use to ensure that the data in the closure is current. We are also exploring policies for applying regional logic and providing other mechanisms for making it easy to extend business logic in a decentralized manner while providing the advantages of a centralized system. Our current prototype addresses mainly the design stage of the service lifecycle and we are exploring how these techniques may be extended to cover other stages of the service lifecycle, in particular the implementation and ongoing delivery stages.

In this paper, we have outlined how we have used closures to integrate a single personal productivity application and a single business productivity application. We are also exploring how to relax restrictions (atomicity, isolation, etc.) on the steps in the global workflow that we can support through personal productivity tools. We are also interested in enabling controlled interaction amongst two or more closures on the client side without any additional installations (e.g., Excel and Word documents sharing content). Finally, we are exploring how closures could be used to update not only data but also business logic such as region-specific costing and analysis functions.

Acknowledgements: We are grateful to many people from HP Managed Services for their contributions, input, and feedback regarding this research.

References

1. McKinsey Global Institute Report. “US Productivity Growth 1995-2000, Understanding the Contribution of Information Technology Relative to Other Factors”, October 2001.
2. Maniscalco, J.: “Empowering the User Personal Proces Automation”, eAI Journal, pp. 43-46, Dec 2002
3. Baida, Z., Gordijn, J., Omelayenko, B.: A Shared Service Terminology for Online Service Provisioning. In: ICEC 2004: Proceedings of the 6th International Conference on Electronic Commerce, New York, NY, USA, ACM Press (2004) 1-10
4. Halevy, A., Ashish, N., Bitton, D., Carey, M., Draper, D., Pollock, J., Rosenthal, A., Sikka, V.: Enterprise Information Integration: Successes, Challenges and Controversies, 2005 ACM SIGMOD Conference, Baltimore, MD., June, 2005.
5. Reif, G., Gall, H., Jazayeri, M.: WEESA - Web Engineering for Semantic Web Applications, WWW Conference, May, 2005, Chiba, Japan.
6. Cardinaels, K., Meire, M., Duval, E.: Automated Metadata Generation: the Simple Indexing Interface, WWW Conference, May, 2005, Chiba, Japan.
7. DB2 Content Manager, http://ibm.com/software/data/cm/, http://www.db2mag.com/story/showArticle.jhtml?articleID=51000328
8. Satyanarayanan, M.: The Evolution of CODA. ACM Transactions on Computing Systems 20 (2002) 85-124
9. Huston, L.B., Honeyman, P.: Disconnected Operation for AFS. Technical Report citi-tr-93-3, University of Michigan, CITI (1993)
10. Stanski, P., Giles, S., Zaslavsky, A.: Document Archiving, Replication and Migration Container for Mobile Web Users. In: SAC '98: Proceedings of the 1998 ACM symposium on Applied Computing, New York, NY, USA, ACM Press (1998) 400-404
11. Bernard, G., Ben-Othman, J., Bouganim, L., Canals, G., Chabridon, S., Defude, B., Ferri;, J., Ganarski, S., Guerraoui, R., Molli, P., Pucheral, P., Roncancio, C., Serrano-Alvarado, P., Valduriez, P.: Mobile Databases: a Selection of Open Issues and Research Directions. SIGMOD Record 33 (2004) 78-83
12. Danielsson, M. Microsoft-SAP Mendocino project has opportunities, drawbacks. SearchSAP.com. June 6, 2005.
13. Reporting Engines: http://www.reportingengines.com.
14. Wikipedia: (2005) http://wikipedia.org.
15. Burg, B., Govindarajan, K., Kuno, H., Smathers, K., Yuasa, K.: An enterprise applications platform for the managed services business. Technical Report HPL-2004-119, Hewlett-Packard Laboratories (2004)
16. Kuno, H., Yuasa, K., Govindarajan, K., Smathers, K., Burg, B., Carau, P., Wilkinson, K.: Governing the contract lifecycle: A framework for sequential configuration of loosely-coupled systems. In: DNIS 2005. (2005)
17. Microsoft Corporation, “Smart Client Application Model and the .NET framework 1.1”, http://microsoft.com/netframework/using/building/windows/analystreports/smartclient.aspx
18. Microsoft Corporation, “Smart Client Offline Application Block”, web document http://msdn.microsoft.com/library/en-us/dnpag/html/offline-ch01.asp
19. Horstmann, M., Kirtland, M.: “DCOM Architecture”, web document http://msdn.microsoft.com/library/en-us/dndcom/html/msdn_dcomarch.asp