Extending Applications for New Interface Technologies: An SAP R/3 Case Study

Michael Good, Robert Wenig, and Thuan-Tit Ewe

SAP America, Inc.
950 Tower Lane, 12th Floor
Foster City, CA 94404 USA
Tel: +1-415-287-3226
E-mail: michael.good@sap-ag.de

ABSTRACT

SAP has seen increasing demand for its R/3 system to have a more extensible user interface. We describe an intelligent terminal interface that extends R/3's capabilities to telephone, multimedia kiosk, World Wide Web, and enhanced graphical interfaces. Incremental changes to the existing presentation server add large amounts of user interface capability. Choosing an appropriate programming interface maximizes compatibility with existing software and minimizes learning time for application developers.

Keywords

Extensibility, business software, reengineering, OLE.

INTRODUCTION

SAP is the industry leader in standard enterprise application software [3]. SAP's R/3 product supports reengineering projects in areas such as financial accounting, sales and distribution, material and plant management, and human resources (HR). The R/3 system has a three-tier client-server architecture with presentation, application, and database servers. The SAPGUI graphical user interface runs on all major desktop platforms. R/3 application products, totaling many million lines of code, are developed using SAP's ABAP/4 Development Workbench.

SAP was facing several business problems regarding the presentation-level software. There was no provision for user interfaces other than standard desktop GUIs. The company thus was losing business to competitors who could provide interactive voice response (IVR) telephone interfaces for their products. Similarly, opportunities were being lost for providing other new interfaces such as multimedia kiosks and World Wide Web pages. ABAP/4 programmers find most of their time is spent struggling to make a decent interface, as GUI design support in the workbench has lagged the best PC-based tools. The details change, but similar problems recur for application vendors as user interface technology evolves and improves [4].

More fundamentally, the user interface for many applications was developed for trained specialists. Many reengineering projects restructure work to emphasize both less specialized tasks and greater self-service from customers and vendors [2]. R/3's interfaces could be redesigned to better support non-specialist users, both for desktop and alternative interface technologies.

R/3 already provides several application programming interfaces (APIs) to support application extensibility. For example, a remote function call interface provides similar capabilities to many remote procedure call systems, and is accessible on PCs via OLE Automation [1]. However, the existing interfaces have two principal problems:

1. The business rules in the application software are only fully accessible via direct interaction with the SAPGUI, not through the existing programming interfaces.

2. Thus, programmers usually need to learn ABAP/4 as well, requiring long learning times to become familiar with the R/3 system and development environment.

THE INTELLIGENT TERMINAL

We have developed a new integration technology for the R/3 system by programming directly to the SAPGUI through an „intelligent terminal" interface. The current SAPGUI communicates with the application server through a terminal-like protocol, evolved from the

To be submitted to the ACM's CHI '96 Conference on Human Factors in Computing Systems April 14-18, 1996 Vancouver, BC, Canada

mainframe heritage of the predecessor R/2 product. On Windows systems, the SAPGUI interface currently consists of two executable files. We have converted one of those executables into a 32-bit dynamic link library (DLL) with three entry points. We then developed a C-language API on top of this DLL. The API is similar to other „screen-scraping“ programs that take the contents of the screen and make them accessible through standard data structures and function calls.

We then built two higher-level interfaces on top of this API. The OLE Automation server is used by Windows clients such as multimedia kiosk programs developed using Visual Basic. The serial-line server is used by clients running on other operating systems, such as IVR clients on OS/2.

By programming the SAPGUI directly, we circumvent both the learning curve and business rule problems. To build a new interface to an existing application, one just needs to learn how the application is performed with the existing interface. Instead of learning an entirely new programming environment, one only learns how to use a new API within the existing programming environment. The SAP-specific knowledge required for developing a new interface can thus primarily reside with the users of the system, rather than with the programmers of the new interface.

Intelligent terminal programs work primarily with transaction codes, function keys, and the labels of the controls that appear on the screen. The following Visual Basic program retrieves a telephone number for a specific personnel code. This code fragment assumes we have already created the OLE Automation object called „Sap“ and logged onto an R/3 system. The program goes to the „pa30“ transaction for maintaining HR master data and inputs the personnel code into the text field to the right of its label. In an IVR application, this number could be keyed in over the phone. The program then selects the „Addresses“ radio button to the left of its label, and inputs the „Change“ function key so that the data may later be changed by the user. The telephone number is placed into a variable, which could then be read back to the user over the phone.

```
Dim Ctrl As Integer
Dim MyID, Telephone As String
MyID = "1502" ' Would come from user
Sap.Transaction "pa30"
Ctrl = Sap.FindControlRight("Personnel")
Sap.Controls.Item(Ctrl).Value = MyID
Ctrl = Sap.FindControlLeft("Addresses")
Sap.Controls.Item(Ctrl).Selected = True
Sap.SendKey("Change")
Ctrl = Sap.FindControlRight("Telephone")
Telephone =
    Sap.Controls.Item(Ctrl).Value
```

CURRENT USAGE

Partner companies have developed several new application interfaces with the intelligent terminal. These interfaces were developed in a few days. Prior attempts with existing interfaces to R/3 had failed after several months of work.

We demonstrated kiosk and telephone interfaces for self-service human resources transactions such as address changes at the 1995 Sapphire user conference. A proactive interface for service monitoring is part of Andersen Consulting’s DAVINCI demonstration. The software monitors the plant management system and can call or fax the responsible party when it sees that equipment is failing too frequently. A kiosk application for self-service entry, update, and review of personnel qualifications is currently under development.

Within SAP, we have demonstrated telephone and Web interfaces for monitoring the status of problem reports that one has entered into a vendor’s problem-tracking system. We have implemented a Visual Basic program that can save any R/3 screen as a Visual Basic form, providing a starting point for GUI redesign. The OLE Automation server opens access to R/3 to many Windows applications. As an example, we can now save and replay R/3 interactions to and from an Excel spreadsheet. This was written to prototype an easier way for SAP to maintain the sample company database provided with R/3’s training materials.

FUTURE WORK

We see the intelligent terminal as a first enabling step to move R/3 interface development to the presentation server, leaving the business logic at the ABAP/4 application level. We are working on providing programmers access to the table and field names in the repository, rather than having to rely on surface features of screen appearance. We also plan to add support for heterogeneous network interfaces.

CONCLUSION

The SAP R/3 intelligent terminal is a case study of extending an existing large-scale application to work with new human-computer interaction technologies. Incremental changes to a small part of the current software product allow us to easily add a diversity of interface styles that were not supported in the original product design. Choosing the right level of programmability is also important. It is not sufficient to have „open“ interfaces. They must be open at the level that allows developers to solve real-life business problems quickly and effectively.

ACKNOWLEDGMENTS

We thank our partners at Edify, Co-Development, and Talx for driving the development of the intelligent terminal.

REFERENCES

1. Brockschmidt, K. *Inside OLE.* 2nd Ed. Microsoft Press, Redmond, WA, 1995.
2. Hammer, M. and Champy, J. *Reengineering the Corporation.* HarperBusiness, New York, 1993.
3. Lieber, R. B. Here comes SAP. *Fortune*, Oct. 2, 1995.
4. Telles, M. Updating an older interface. In *Proc. CHI '90 Human Factors in Computing Systems* (Seattle, April 1-5, 1990), ACM Press, pp. 243-247.