In Proceedings of 5th Asia-Pacific Web Conference, September 27-29 2003, Xi'an, China, Springer LNCS 2642, pp. 251-262.

Architecture for a Component-based, Plug-in Micro-payment System

Xiaoling Dai[1] and John Grundy[1, 2]

Department of Computer Science[1] and Department of Electrical and Electronic Engineering[2]
University of Auckland, Private Bag 92019, Auckland, New Zealand
{xdai001, john-g}@cs.auckland.ac.nz

Abstract. Micro-payment systems have the potential to provide non-intrusive, high-volume and low-cost pay-as-you-use services for a wide variety of web-based applications. However, adding micro-payment support to web-sites is usually time-consuming and intrusive, both to the web site's software architecture and its user interface implementation. We describe a plug-in, component model for adding micro-payment support to web applications. We use J2EE software components to encapsulate micro-payment E-coin debiting and redemption and discrete user interface enhancement. A CORBA infrastructure is used to inter-connect J2EE and non-J2EE vendors and micro-payment brokers. We demonstrate the feasibility of our approach with an on-line, pay-as-you-use journal portal example and outline an approach to using web services to further generalize our architecture.

1 Introduction

Many current e-commerce systems adopt a macro-payment model and architecture [16], [17], [18]. A customer makes a small number of purchases which have a reasonably high cost per purchase. In order to pay for purchases, a "heavy weight" interaction between the vendor of the product or service and an authorisation agent (bank, credit-card company etc) system is carried out. This typically involves the customer supplying credit card details or "digital money" certificates, which are communicated to the authorisation system using complex encryption algorithms. Business processing logic and database updates are performed by the authoriser before the purchase is approved. The vendor system waits for approval before providing the customer with goods or services. This approach works well for relatively small numbers of transactions and relatively high purchase price (to offset the cost of authorisation) [3]. In some e-commerce scenarios this approach has a number of fundamental flaws. It requires the authorisation system to always be on-line. High numbers of transactions or low-price purchase items are infeasible, due to bottle- necking or prohibitive cost per-transaction. In addition, with some approaches the customer's identity can not gener-

ally be hidden from the vendor and customers are charged for products, services or information even if they don't use them [3].

We describe the NetPay micro-payment model and a component-based software architecture that we have been developing for NetPay. NetPay provides an off-line micro-payment model using light-weight hashing-based encryption. A customer buys a collection of "E-coins" using a macro-payment from a broker. These coins are cached in an "E-wallet" on the customer's machine. The customer, when buying many small-cost items from a vendor, pays for these transparently by the passing of E-coin information to the vendor. Periodically the vendor redeems the E-coins with the broker for "real" money. E-coins can be transparently exchanged between vendors when the customer moves to another site. In previous work we have described the hard-coding of NetPay support into web-based applications, an approach used by most macro- and micro-payment solutions [4]. Major disadvantages of such an approach is the difficulty and time to add micro-payment support to existing applications, the mis-match of implementation technologies and application services if trying to reuse code, and potential design and software architecture mis-matches.

We describe a new component-based approach for encapsulating micro-payment support for web-based applications - "vendors" of products, services or information to be purchased by micro-payment. We use an example web-based application, an E-journal portal system, that we have developed independently for a teaching project and then have enhanced by adding NetPay components via J2EE Enterprise JavaBean components and Java Server Page proxies. These reusable NetPay components are plugged into the existing journal site to enhance it with micro-payment support with minimal or no code changes. We describe our approach's architecture and design and we illustrate the E-journal interface after plugging in these components. We discuss related work, the advantages and disadvantages of our approach and outline possibilities for further research.

2 Example Application Domain

On-line journals have become popular and are usually paid for on a macro-payment, subscription basis. Using subscription-based payment, a customer first has to subscribe to the journal by supplying payment details (credit card etc) and the journal system would make an electronic debit to pay for their subscription by communicating with an authorisation server. The customer would then normally go to the journal's site where they login with an assigned customer name and password. The journal looks up their details and provides them access to editions of various journals if their subscription is still current. The E-journal site may provide a range of journals for a specific publisher or act as a portal for multiple publishers, paying each on a per-subscription or per-usage basis. If the customer's subscription to a particular journal has run out, they must renew this by authorising a payment from their credit card.

Fig. 1 (a) outlines some of the key interaction use cases for this scenario. Problems with this approach are that there is no anonymity for the customer (the journal system knows exactly who they are and when and what they read), they can not move to other sites without first

subscribing to them too, and they must pay for the whole journal (often very expensive), even if they want just one or two sections or articles. These issues apply to many other information sources on the internet where vendors want to charge for a variety of information content [1], [7].

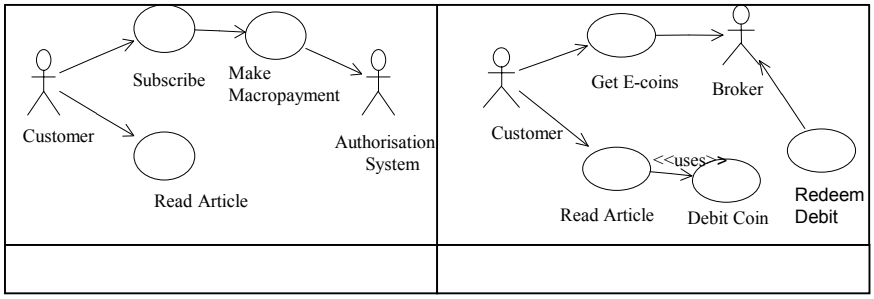

Fig. 1. Two on-line newspaper interaction scenarios

An alternative approach is a micro-payment model. There are several approaches to micro-payment [5], [6], [8], [9], [10]. We outline the basic interactions of our Net-Pay model [2]. Fig. 1 (b) outlines the key interaction use cases for this scenario. The customer first goes to a broker and purchases "E-coins" using a single macro-payment. These are stored in an E-wallet on the customer's machine. The customer can then visit any journal site or newspaper site or music site they wish, their wallet giving the site an E-coin. Each time they view an article (or section or page, depending on the item charged for) or download a song their E-coin is debited. The vendor redeems debits with the broker (for "real" money") periodically e.g. each night/week. The customer can move to another site and unspent money associated with their E-coin is transferred from the first vendor to the second. If E-coins run out, the customer communicates with the broker and authorises another macro-payment debit. The standard macro-payment methods cannot be effectively or efficiently applied for buying inexpensive information goods, like single articles of an on-line journal, because transaction costs are too high. Encryption mechanisms used are slow and each transaction typically "costs" a few cents. Macro-payment suits spending small numbers of large amounts. An Internet micro-payment system allows spending large numbers of small amounts of money at web sites in exchange for various content or services, as in the E-journal scenario above. The design of micro-payment systems is usually quite different from existing macro-payment systems, since micro-payment systems must be very simple, secure, and efficient, with a very low cost per transaction.

The rapid growth of the Internet has led to the appearance of thousands of different web sites, which are created to provide information to millions of people around the world. Producing and maintaining a quality web site takes a great deal of time, dedication and money. To help offset the development costs, web site producers are keen to use their site as a source of income. Therefore, an efficient micro-payment system whose components can be plugged into the existing sites to handle such a trade is needed. Such a micro-payment system would ideally be easily reusable i.e. not require extensive redevelopment to integrate with the web application's architecture; not require any modification to the web application itself; provide effective and efficient

debiting of customer "E-coins" and redemption of these coins via a broker for "real" money; and integrate seamlessly with both the architecture and user interfaces of the web application. In our E-journal example, the journal provider would want to charge small amounts on a per-article basis (perhaps varying amounts) and have this micro-payment support integrated seamlessly and with minimum effort with their existing web application architecture.

3 An Overview of NetPay

A new micro-payment protocol called NetPay allows customers (e.g. journal readers) to purchase information from vendors (e.g. on-line E-journal) using "E-coins" on the WWW [2]. A broker provides a source for E-coin purchase and vendor redemption of E-coins for "real" money. NetPay is intended to provide a secure, cheap, widely available, and debit-based protocol for micro-payment applications. NetPay differs from previous micro-payment protocols in the following ways: NetPay uses "touchstones" signed by the broker and E-coin indexes signed by vendors which are passed from vendor to vendor.

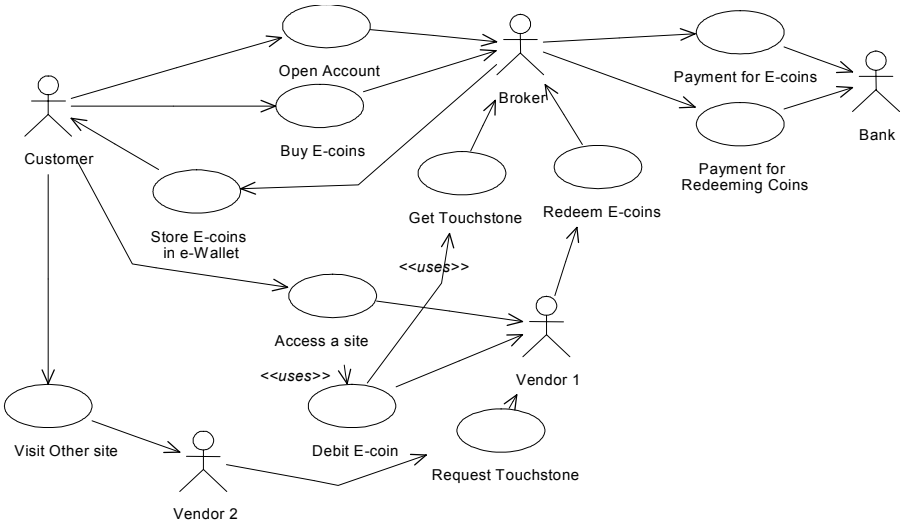

Fig. 2. Basic NetPay component interactions

The signed touchstone is used by a vendor to verify the electronic currency – the "paywords" encoding E-coins, and the signed index is used to prevent double spending by customers and to resolve disputes between vendors. There is no dependency on customer trust required with this approach. Customers have an "E-wallet" that manages their available E-coins: the wallet may reside on the customer's PC or may reside on the broker and vendor machines, being passed from vendor to vendor as the customer accesses information at their web sites. In our NetPay approach we make the

assumption that the broker is honest and is trusted by both the customers and the vendors. The micro-payments only involve customers and
vendors, and the broker is responsible for the registration of customers and for crediting the vendors' account and debiting customers' accounts. Fig. 2 outlines some of the key NetPay system interactions.

Initially a customer accesses the broker's web site to open an account and acquire a number of E-coins from the broker (bought using a single macro-payment). The broker sends an "E-wallet" that includes the E-coin ID and E-coins to the customer and the customer's host caches this information. The customer browses the home page of the journal web site and finds a desired article to read. Each article will have a small cost e.g. 5-10c, and the customer would typically read a number of these. When wishing to read the details of an article, the customer clicks on the article heading and the vendor system debits the customer's E-coins by e.g.. 10c. The vendor verifies that the E-coin provided by the customer's E-wallet is valid by use of a "touchstone" obtained once only from the broker. If the payment is valid (coin is verified and sufficient credit remains), the article is displayed on the screen. The customer may browse other articles, their coins being debited (the index of spent coins incremented) each time an article is read. If E-coins run out, the customer is directed to the broker's site to buy more.

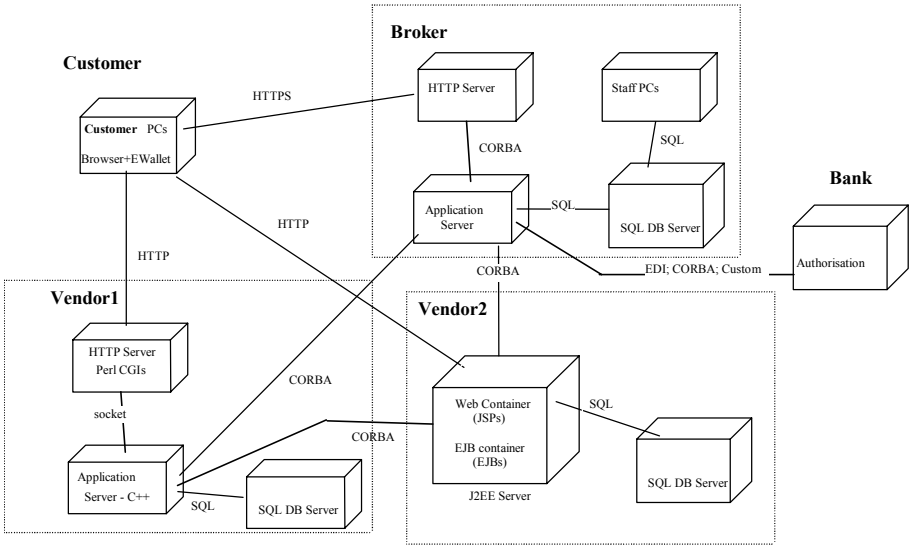

Fig. 3. Basic NetPay software architecture

When the customer changes to another site, the new vendor site first requests the current E-coin touchstone information from previous vendor's site. The new vendor contacts the previous vendor to get the E-coin touchstone and "spent coin" index and then debits coins for to further news articles. At the end of each day, the vendors all send the E-coins to the broker redeeming them for real money.

4 A Component-based NetPay Architecture

We initially developed a software architecture for implementing NetPay-based micro-payment systems for thin-client web applications that used hard-coded vendor facilities for micro-payment [4]. We have extended this work to develop component-based NetPay vendor services, supporting much more easily and seamlessly reused vendor server-side NetPay functionality. NetPay micro-payment transactions involve three key parties: the Broker Server, the Vendor Server, and the Customer browser.

This architecture is illustrated in Fig. 3. The Broker server and the Customer browser are same as the previous NetPay architecture [4]. The **Vendor** web sites provide a web server and possibly a separate application server, depending on the web-based system architecture they use. The Vendor web server pages provide content that needs to be paid for and each access to these pages require one or more E-coins from the customers' E-wallets in payment. In our architecture Vendor application server accesses the Broker application server to obtain touchstone information to verify the E-coins being spent and to redeem spent E-coins. They communicate with other vendor application servers to pass on E-coin indexes and touchstones.

Vendors may use quite different architectures and implementation technology. In the example above, Vendor #1 uses a web server with Perl-implemented CGI scripts, C++-implemented application server and relational database. Vendor #2 uses a J2EE-based architecture with J2EE server providing Java Server Pages (web user interface services) and Enterprise Java Beans (application server services), along with a relational database to hold vendor data.

5 NetPay Component

Our example E-Journal system has a number of customer web browser clients used by customers to access the journal site and read article contents. Another web client is used by staff to manage the redemption of spent E-coins with the NetPay broker server. The vendor J2EE server has a number of web pages e.g. JSPs or servlets and EJBs providing an implementation of the E-journal web system. We add to this a number of NetPay components: EJBs to provide E-wallet management (tracking spending of E-coins by customers; E-coin exchange with the client-side E-wallent application or server-side W-wallet management; touchstone exchange with the NetPay broker or other vendors and E-coin validation). We also provide redemption support for the vendor to communicate with the NetPay broker and redeem customer-spent E-coins for real money.

Fig. 4 shows a high-level view of how these various components interact. The end-user clients access only the session beans. Within the enterprise bean tier, the session beans are clients of the entity beans. On the back end of the system, the entity beans access the database tables that store the entity states. The Session beans access the client-side E-wallet application, broker server and other NetPay-implementing vendor servers via CORBA remote object interfaces.

In our E-journal example system one session bean, `ArticleDBEJB`, represents an interface to the journal article database and is used to select article records. A number of JSPs provide for customer login (if required), journal information and high-level organisation, journal searching and article display.

There are two NetPay session beans that have been added to this E-journal system, `EwalletControllerEJB` and `RedeemControllerEJB`. These session beans provide a client's view of the application's business logic such as MakePayment, ValidateECoins, EwalletReqest and RedeemPayment. Hidden from the clients are the server-side routines that implement the business logic, access databases, manage relationships, and perform error checking.

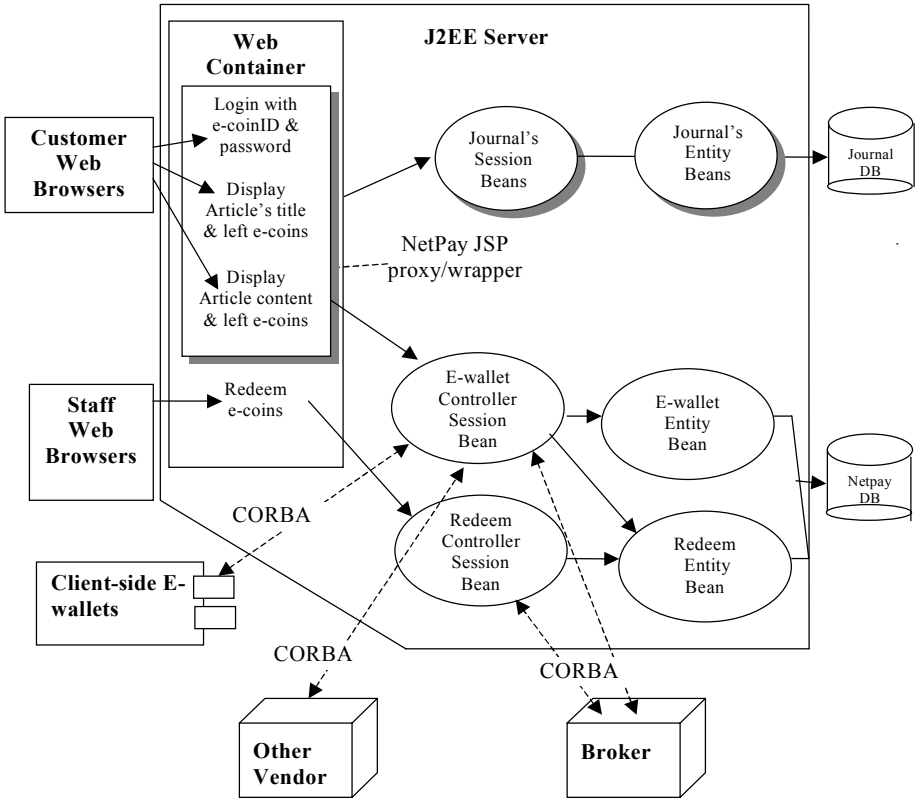

Fig. 4. E-Journal System with NetPay Components

For each NetPay business entity the system has a matching entity bean, in this case an EwalletEJB and RedeeemEJB. These beans provide a component-based interface to various database tables: `EWallet`, `ECoin` and `RedeemCoin`. For each column in a table, the corresponding entity bean has an instance variable. Because they use bean-managed persistence, the entity beans contain the SQL statements that access the tables. For example, the `create` method of the `EwalletEJB` entity bean calls the SQL `INSERT` command. The NetPay database tables that are used by our entity beans

may be added to the existing E-journal database if this is possible, or may be stored in a separate database of their own if required.

In order to add our NetPay micro-payment facility to the E-journal, or to other 3rd party J2EE-based applications, we need to be able to add our EJBs to their J2EE server and to detect when pages are being accessed by customers that need to be paid for. We also need to ensure that if the customer attempting access does not have enough E-coins they are directed to the NetPay broker site to buy some more. If the customer wants a server-side E-wallet managed by the vendor vs running a client-side E-wallet application, we need to have the vendor obtain the customer NetPay user-name/password and obtain the E-wallet from the NetPay broker or the previously-visited NetPay-enabled vendor. In addition the customer usually wants an idea about the cost of an article or other information/service before purchase, and access to their available credit in E-coins.

There are three main ways to integrate the NetPay user interface facilities: (1) modify the existing system web pages to incorporate NetPay information (we have developed some JSP page includes so this can be done easily); (2) generate web pages that display the existing system pages in frames and make appropriate interactions with NetPay EJB components; and (3) generate proxy web pages that interact with NetPay session beans and redirect access to the original web pages. These approaches are illustrated in Fig. 5 below. Each has advantages and disadvantages – the first requiring updates to the existing system web page implementations, the later two requiring re-naming of these pages so the generated pages are passed control at appropriate times. In this paper we show examples of the E-journal extended using the first approach.

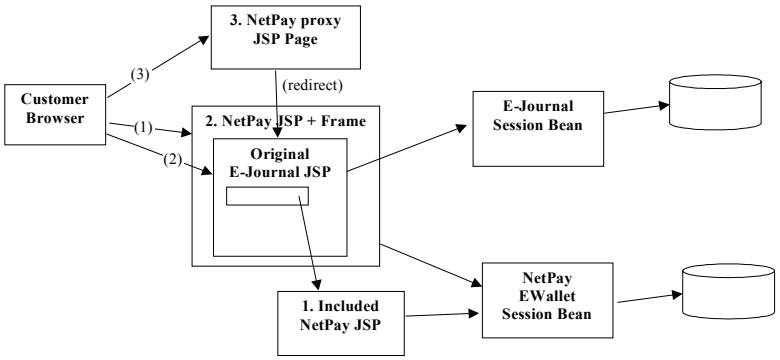

Fig. 5. Ways of integrating NetPay micro-payment functionality with E-journal web pages

A sequence diagram is used in Fig. 6 to show how interactions occur between various components in our NetPay-enabled E-Journal application. For example, when buying an article the customer selects the article for reading e.g. clicking on the URL in a returned article search page or in a journal content page. The web browser requests the article content from the appropriate JSP, and this JSP or its generated proxy requests payment for the content of the article from the NetPay E-wallet session bean. The E-wallet session bean communicates with either the client-side E-wallet or the server-side E-wallet (managed by the E-wallet entity bean) to debit the customer's E-coins to pay for the article. If insufficient coins are available, the customer is directed

to the broker site to buy more. Otherwise, the journal article content is displayed to the customer.

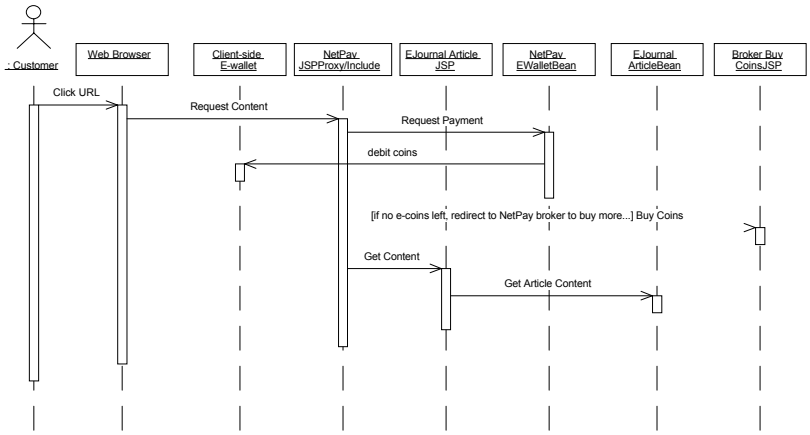

Fig. 6. Click-buy Article and Redeem E-coins Sequence Diagram

6 Implementation

In previous work we implemented a CORBA-based vendor architecture [3]. However, these NetPay components are not optimally reusable and substantial modifications may need to be made to an existing web-based application to incorporate them. We implemented these new NetPay J2EE EJBs and JSP includes to allow for much easier plug-in of NetPay micropayment facilities into existing J2EE applications.

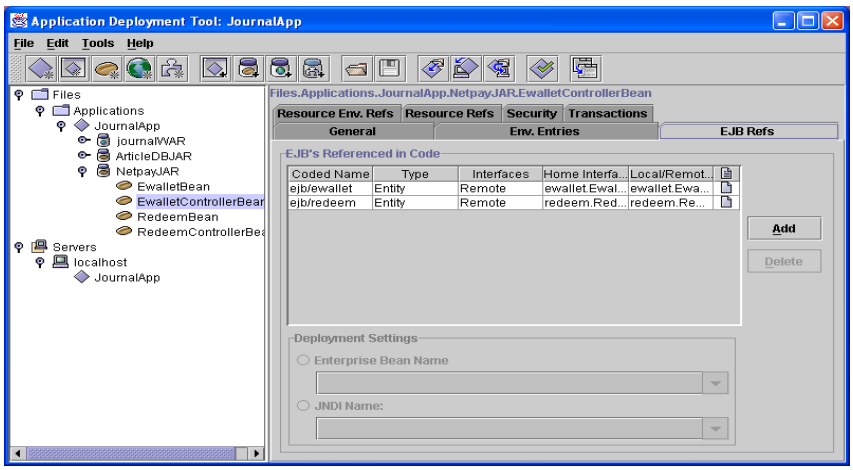

Fig. 7. Plugging in the NetPay vendor-side components with a J2EE deployment tool

The Ewallet and Redeem EJB components are plugged into the existing E-journal system by deploying them into the E-journal system's J2EE server. The Ewallet com-

ponent is used to obtain E-wallet from broker or another vendor, make payments by using the client-side or server-side Ewallet managed E-coins, and to generate redeem request data. The Redeem component is responsible for selecting payments and sending these to the NetPay broker.

EJB deploytool provides interface to define relationships between enterprise beans. That makes it easier to plug-and-play components. Fig. 7 illustrates the interface and the relationship between Ewallet component and Redeem component. There is no relationship among existing journal component (ArticleDBJAR) and NetPay components. The NetPay components are plugged in the existing system very straightforwardly.

7 Discussion

The area of micro-payment on the Internet has attracted much recent research attention. Besides NetPay, there are several micro-payment systems that are based on a payword-based micro-payment protocol. These systems can be classified as credit-based and debit-based. Payword [10] is an off-line credit-based system where a user's account is not debited until some time after purchases. This provides more opportunity for fraud since a large number of purchases can be made against an account with insufficient funds. PayFair [11] is a debit-based micro-payment system that employs some parts of the Payword scheme. A payword chain purchased from the broker will be bound to a specific vendor. Many payword chains can be purchased in advance from the broker and stored in the customer's machine. There is no digital-signature required for witness of the payment promise in the system. NMP [12] is a credit-based protocol that improves the fairness for customers from Payword protocol.

The Payword-based micro-payment systems described above share a key disadvantage - they are all vendor specific. The E-coin (paywords) in the systems are only usable at one vendor and have no value for any another vendor. Some currently available micro-payment protocols are not only specifically designed for selling information goods on the Internet, but they can also be used for wireless communications [13], [14]. This can be an important issue for mobile communications where call charges are still large in comparison with Internet based communications. It also reduces delay and removes the possibility of incomplete payment protocols due to communications failures. Our NetPay server-side E-wallet can be used to provide a similar capability.

The use of component-based approaches to building and extending enterprise and web-based systems has become popular [16], [17], [18], [19], [21]. Many approaches focus on enterprise business logic extension [16], [18], rather than a combination of user interface and logic extensions [19], [20]. Our approach to adding NetPay to an existing E-journal uses a similar technique to that we developed for adding collaborative work components to an existing travel-planning application [20]. We plugged in EJBs into the E-journal's existing application server and annotated the E-journal's JSPs to make appropriate E-coin balance, article cost, credit checks and coin debits to the NetPay EJBs. This allows for minimal or no code impact (none if using proxy JSP

pages) to the existing system's infrastructure. The use of an EJB-based software architecture enables the application developer to work on the business logic aspects of the application without having to be concerned with system-level issues, such as transactions, security, multithreading, and so forth. In addition, the NetPay EJBs are completely separated from the particular domain knowledge of the web application, enabling each enterprise bean to be reused in different EJB-based vendor systems via plug-and-play with the existing vendor components. While our approach to adding NetPay user interface support to existing J2EE-based vendor application JSPs is basic, usability evaluation of this NetPay-enabled application has indicated it provides a good user interface to users.

We are currently designing a portal infrastructure using web services that will allow a NetPay-enabled vendor to act as a purchasing portal to non-NetPay supporting vendors by redirecting page accesses to these vendors and charging the customers E-coins in the process. This approach will allow for dynamic registration of vendors and support cross-vendor product searching. We are investigating approaches to using NetPay for mobile information content micro-payment applications, both with a server-side E-wallet and client-side E-wallet storage by the mobile device. Our other work focuses on the development of tools to allow existing component-based applications to be NetPay-enabled without any manual component programming, deployment and configuration. We will apply these to experimenting with adding NetPay to other 3rd party J2EE web applications.

8 Summary

We have described the design and development of software components to enable NetPay to be seamlessly added to existing J2EE-based web applications. NeyPay functionality is embodied in Enterprise JavaBean software components and JSP includes or proxies, allowing the existing application to be easily micro-payment enabled. Our NetPay EJBs use a CORBA infrastructure to communicate with customers' client-side E-wallet applications, with a broker server, and with other vendor application servers, whether J2EE-based or not. We have successfully added NetPay components to as separately-developed J2EE-implemented E-journal application to demonstrate our approach's feasibility.

References

1. Blankenhorn, D.: Charging for Content, E-commerce Times. http://www.ecommercetimes.com/perl/story/306.html.
2. Dai, X. and Lo, B.: NetPay – An Efficient Protocol for Micropayments on the WWW. Fifth Australian World Wide Web Conference, Australia (1999)
3. Dai, X., Grundy, J. and Lo, B.: Comparing and contrasting micro-payment models for E-commerce systems, International Conferences of Info-tech and Info-net (ICII), China (2001)

4. Dai, X., Grundy, J.: Architecture of a Micro-Payment System for Thin-Client Web Applications. In Proceedings of the 2002 International Conference on Internet Computing, Las Vegas, CSREA Press, June 24-27, 444--450
5. Furche A. and Wrightson, G.: SubScrip – An efficient protocol for pay-per-view payments on the Internet, The 5th Annual International Conference on Computer Communications and Networks, USA (1996)
6. Herzberg, A. and Yochai, H. : Mini-pay: Charging per Click on the Web, 1996 http://www.ibm.net.il/ibm_il/int-lab/mpay
7. Herzberg, A.: Safeguarding Digital Library Contents - Charging for Online Content. D-Lib Magazine (1998), ISSN 1082-9873
8. Hwang, M-S., Lin, I-C. and Li, L-H.: A simple micro-payment scheme. Journal of Systems & Software, 55(3)(2001) 221--229
9. Manasse, M.: The Millicent Protocols for Electronic Commerce. First USENIX Workshop on Electronic Commerce. New York (1995)
10. Rivest, R. and Shamir, A.: PayWord and MicroMint: Two Simple Micropayment Schemes. Proceedings of 1996 International Workshop on Security Protocols, Lecture Notes in Computer Science, Vol. 1189. Springer (1997) 69—87
11. Yen, S-M.: PayFair: a prepaid internet ensuring customer fairness micropayment scheme. IEE Proceedings-E Computers & Digital Techniques, vol.148, no.6, Nov. 2001, pp.207-13. Publisher: IEE, UK.
12. Ji, D-Y. and Wang, Y-M.: A micropayment protocol based on PayWord. Acta Electronica Sinica, 30(2)(2002) 301—303
13. Sangjin, K., Heekuck, O.: An atomic micropayment system for a mobile computing environment. IEICE Transactions on Information & Systems. E84-D(6) (2001), 709—716
14. DongGook, P., Boyd, C., Dawson, E.: Micropayments for wireless communications. Information Security and Cryptology - ICISC 2000. Third International Conference. Proceedings (Lecture Notes in Computer Science Vol.2015). Springer-Verlag, Berlin, Germany (2001), 192—205
15. McGarvey, R.: Micropayments enable teensy content purchases. Econtent, 24(1) (2001) 18—21
16. Allen, P.: Realising E-Business with Components. Addison-Wesley, October 2000.
17. Bichler, M., Segev, A., Zhao, J.L.: Component-based E-Commerce: Assessment of Current Practices and Future Directions. *SIGMOD Record* 27(4)(1998) 7—14
18. Fingar, P.: Component-Based Frameworks for E-Commerce. Communications of the ACM(2000)
19. Chong, N.S.T., Sakauchi, M.: e-CoBrowse: co-navigating the Web with chat-pointers and add-ins - problems and promises. Parallel and Distributed Computing and Systems 2(2000) 803—808
20. Grundy, J.C., Wang, X., Hosking, J.G.: Building Multi-device, Component-based, Thin-client Groupware: Issues and Experiences. In Proceedings of the 3rd Australasian User Interface Conference, Melbourne, Australia (2002) 28—30
21. Ryley, S.: Corporate portal development: a practical approach ensures real business benefits. Business Information Review 18(2)(2001) 28—34