

Providing Application Services for Small Businesses

Carol Lushbough
Department of Computer Science
University of South Dakota, Vermillion, SD 57609

Abstract - Many small businesses require distributed information systems but do not have the resources to develop, host or maintain them. An alternative approach may be to offer the software as a distributed service. Students in the University of South Dakota (USD) Systems Analysis course partner with small businesses to design software services using a model-driven system development approach and the students in the Software Engineering course implement the systems using Sun Microsystems' J2EE architecture. After the design and implementation of the software service has been completed, the system is handed over to a software company who contracts to host and maintain the software. A contract is negotiated after the design specification phase of the system and before detailed design begins.

Keywords: Application Service Provider, Model Driven, J2EE.

1.0 Introduction

In the past, software alternatives for small businesses included purchasing prepackaged software or commissioning custom software. An emerging alternative for small businesses is to contract with an Application Service Provider (ASP) to host and maintain their applications. An ASP manages and delivers application capabilities to multiple entities from a data center across a wide area network.^[1] ASPs were generally thought to be associated with middle-tier and large companies in the context of "outsourcing." In the late 1990's small businesses found outsourcing software and services to an ASP to be cost-prohibitive.^[2]

One factor that contributes to the growth of ASPs is the high cost of customized software. As software becomes more complex, the cost of development grows. Companies with anywhere from 20-100 employees who are feeling the pressure of upgrading their in-house technology to cope with growth and those who need more complex software to run their business are ideal ASP candidates.^[3] Small business application requirements often include an Internet presence, intranet access, persistent storage, security, updates, backup and recovery. Even if the small business is able to find prepackaged software to meet its needs, this software may require in-house expertise to install, operate, and maintain. Many small businesses seem willing to sacrifice the benefits of custom software to save money. These small businesses are lured by the idea of the reduced up front costs of prepackaged software.

It is important to choose a software development platform that will easily support a robust distributed architecture because of the growing complexity of software and the requirements for Internet and intranet access, security, and persistent storage. The Java 2 Enterprise Edition (J2EE) platform is a great choice as it addresses all of these issues.

2.0 ASP Model

An Application Service Provider can give a small business access to "big business" applications at a relatively low cost. ASPs offer businesses access to applications via the Internet or more secure virtual private networks. Virtual private networks provide an encrypted connection from the business to the ASP's network. In addition to hosting custom application software, ASPs can also provide access to prepackaged software such as e-commerce, financial management and office suites.

Generally, the ASP owns and operates a software application. The ASP also owns, operates and maintains the servers that run the application and employs the people needed to maintain the application. The ASP makes the

application available to customers everywhere via the Internet, either in a browser or through a of thin client. The ASP bills for the application either on a per-use basis or on a monthly/annual fee basis.

The ASP model has evolved because it offers significant advantages over traditional approaches. The pay-as-you-go model is often significantly less expensive than the outright purchase, installation and maintenance of a software system. The ASP model, as with any outsourcing arrangement, eliminates the need for technical employees, who can be very expensive and very specialized. The ASP model also eliminates the need for specialized infrastructure for the application as well as the backend services. For example, if the application requires an Oracle or MySQL database, both the application and the database have to be supported. ASPs typically have more bandwidth than small or midsized companies. This is useful when the application harvests data from Internet sources, such as distributed databases.

3.0 J2EE Architecture

3.1 Overview

The Java 2 Enterprise Edition (J2EE) is a state-of-the-art architecture for developing, deploying, and managing reliable enterprise applications in production environments. It was developed by Sun Microsystems and has been available since June, 2000. According to Sun Microsystems, this framework has the ability to “support millions of users, and millions of dollars in financial transactions every day.”^[4] The J2EE architecture provides the ability to develop a multi-tiered distributed system (see Figure 1). The middle-tier is generally responsible for the implementation of the business logic of the system and is developed modularly through the use of Enterprise JavaBeans (EJB). EJB components are Java code that implements an enterprise's business processes and entities. They perform the application's business operations and encapsulate the business logic. “Applications written using the Enterprise JavaBeans architecture are scalable, transactional, and multi-user secure.”^[4] These applications may be written once, and then deployed on any server platform that supports the Enterprise JavaBeans specification.

The J2EE architecture divides the environment into containers. A container provides specific services to components. A component can be thought of as an independent “bean” that communicates with other “beans” through its container. An EJB container provides such services as transaction management, security, multi-threading, distributed programming, and connection pooling.

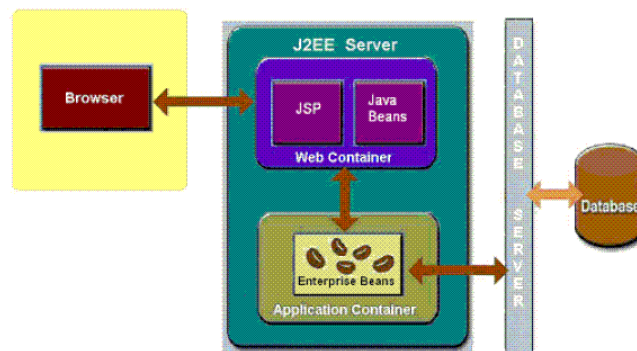


Figure 1 J2EE Architecture Diagram

3.2 Advantages

- *Reusable components* – The J2EE platform defines a set of standards for implementing reliable, scalable distributed applications through the development of enterprise java beans (EJB). An EJB is a reusable component of a J2EE application whose methods may be called by multiple clients. It may also be integrated into multiple applications and deployed to multiple servers.

- *Portable applications* – J2EE applications can be deployed to any machine running a J2EE Application Server. This reduces the risk of vendor dependence and system obsolescence.
- *Distributed system* – The J2EE platform supports multi-tier, distributed application development. Multi-tier applications generally are configured to have a client tier, business logic middle tier, and a database tier. Not only can each of these tiers be deployed to different servers, components within the tiers can be distributed.
- *Application interoperability* – J2EE applications are able to interoperate with other systems (e.g., other J2EE applications or legacy systems), using their services and providing services to them.
- *Focus on implementing business logic* – The J2EE Application Server handles many of the system details including connection pooling, threading, transaction management and security. This allows the developers to focus on writing code to address the business logic and minimizes the time spent on system development.

4.0 Example – Dataware Customer Management System

4.1 Project Description

In 2003, the Dataware Customer Management System was chosen as the University of South Dakota Systems Analysis & Software Engineering project. Dataware is a South Dakota based company that provides its customers with services such as web-hosting, server co-location, Hosted Software Vendor (HSV), off-site storage, Business Contingency Planning (BCP), VPN transit and firewall security patching. Dataware was in need of a system that could manage their customer information and provide a single access point for their trouble ticket and statistical reporting software.

Dataware keeps track of the contracts made with each customer. New customers are entered into the system by Dataware management generally after an agreement has been reached and a contract has been drafted. A contract includes a start and end date, the cost, length of the contract, terms, the company contact that signed the contract and a scanned copy of the signed contract. Resources and services that are included in the contract are entered into the system along with any additional contacts.

Dataware provides a variety of resources to customers such as different types of computer hardware and software. Computer equipment is stored in racks and is measured by U's. Each rack contains 42 U's of storage. The equipment housed in a rack could be owned by Dataware and used by a customer or owned by the customer. The different types of equipment include items such as servers, switches, routers, UPS, monitors, keyboards and mice. Each piece of equipment has a U size to indicate how much space it takes up in the rack. It is important for Dataware to keep track of the server components. They need to know its type of processors, memory, drives, network cards, motherboard, CD, cabling etc. and if the server is rack mount or stand-alone. Some of the attributes of software include: vendor information, product name, version (major and minor), value/cost, license count, description, expiration date, status, notes, support contact, and warranty start and end date.

Dataware also keeps track of the services provided to each customer. Services are purchased through a contract and have a name, type and notes. A service could be IP addresses, domain names, SSL certificates, web-hosting (shared or dedicated), etc. Information required for IP addresses include, address, whether it is external or internal, mask, ip address to which it is tied, device, dns name, and cost. Domain names include the registrar name, contact id, administrative contact, dns servers and expiration date.

The primary problems that were addressed in the design of the new Dataware Customer Management System included:

- There were four separate systems used by Dataware and their customers to provide monitoring and customer/staff communication in the areas of trouble tickets and statistical reporting. Each system required separate interfaces and user login information.
- Dataware kept track of its customers and their resources in a variety of different places and formats including documents and spreadsheets.

- Customer data was not available outside Dataware headquarters.
- Dataware wanted to be notified when a contract was about to expire.
- Dataware requested ad hoc querying and reporting capabilities

4.2 Model-driven System Design

The design of the Dataware Customer Management System was developed using a model-driven system development approach and produced four major deliverables: a requirements description document, a Use Case Model, a Data Model and a User Interface Model. Through an educational grant, Rational Software Corp. provided USD with the Rational Enterprise Edition of Rational Rose. This software provides a process (the Rational Unified Process), and a language (the UML) to successfully visualize, specify, document, and construct a software system.^[6]

The modeling of the application began with the creation of a requirements description document, an overall description of the system to be developed. This document describes those features that are essential for the operation of the business and identifies those users the system will service. The requirements description document was created by interviewing users and analyzing the existing systems.

Based on the description of the system requirements, students developed the Use Case Model which identifies the functions the system will offer the users. This model aids in the documentation of user requirements, helps in the discovery of object classes and helps to identify object responsibilities. The use cases provide an outside view of the system and what it must do, that is, the services, behaviors and responses it must provide.^[7]

Using a use case template, students document specific use case scenarios. The required scenario information include: title, list of actors that are being provided the service, goals of the use case, business and verification rules associated with the use case, pre and post conditions, alternative course of events, related use cases and the steps involved in carrying out the scenario. These use case scenarios are used in the development of the UML sequence diagrams.

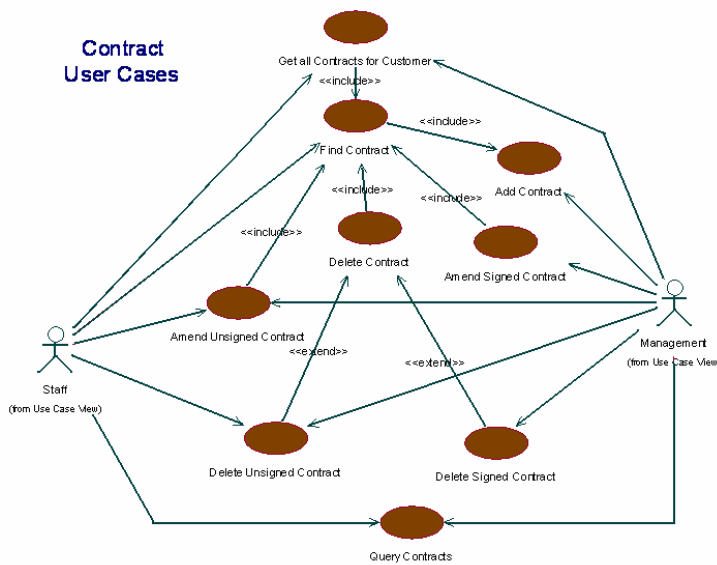


Figure 2 UML Use Case Diagram showing contract required functionality

The *Data Model* was developed through the identification of the objects involved in the use cases. Objects are responsible for providing knowledge and services to the system. CRC (classes, responsibility and collaboration)

cards developed by Beck and Cunningham^[8] are very helpful in identifying the responsibilities and relationships of the objects. The primary deliverable resulting in the development of the Data Model are UML class diagrams.

The *User Interface Model* involves the design of the screens required for the use cases. It is important that the users be closely involved with the screen design. For the Dataware Web-based client tier, students used HTML and Java Script for screen prototyping.

4.3 System Implementation

The Software Engineering class at the University of South Dakota chose the J2EE architecture for the implementation of a three-tiered distributed application service. Because Dataware provides 24x7 services, they needed a system that could be accessed from locations outside of the company headquarters but would not jeopardize the security of the data.

The three-tiered ASP model provides a flexible, distributed service that is available to all the users. The client tier consists of a browser based implementation using the model/view/controller pattern developed using Java, Java Server Pages (JSP) and Java Script. Figure 3 presents a sample screen from the client application. The client tier manages the user interface and the interface to the middle tier. The middle tier relieves the client application of the difficulties of executing complex business rules and making queries to database system. The enterprise beans and the associated system services running in the middle tier handle this complexity.

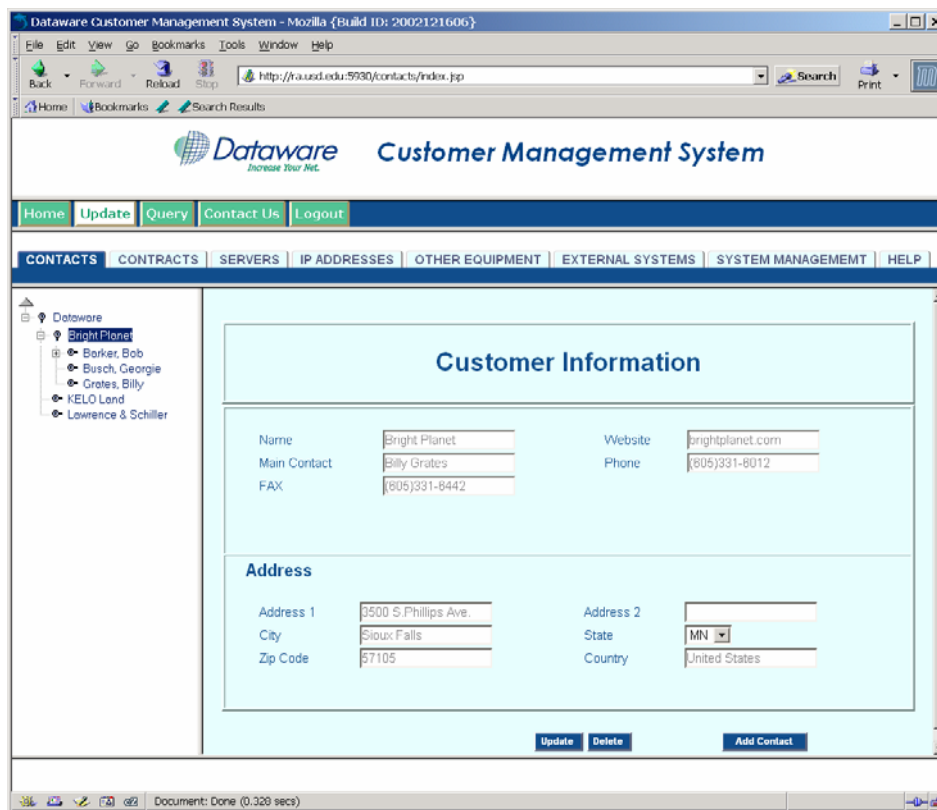


Figure 3 Client Interface Example

The client connects to the middle-tier EJB components deployed to a Sun ONE Application Server through the RMI-IIOP protocol. Using this protocol enables the client application to access enterprise bean references.

The J2EE pattern catalog was used in the design, development and implementation of the EJB tier of the system. Patterns represent expert solutions to recurring problems in a context and thus have been captured at many levels of

abstraction and in numerous domains.^[5] Since there is only one instance of this EJB tier, the updating of this portion of the system simply requires that a new version be deployed to the application server.

The persistent data is stored in a MySQL database. The EJB container accesses the database through the Data Access Objects developed by the students. The database can reside on a server at Dataware or can be hosted by the ASP. Figure 4 provides a simplified diagram of this system.

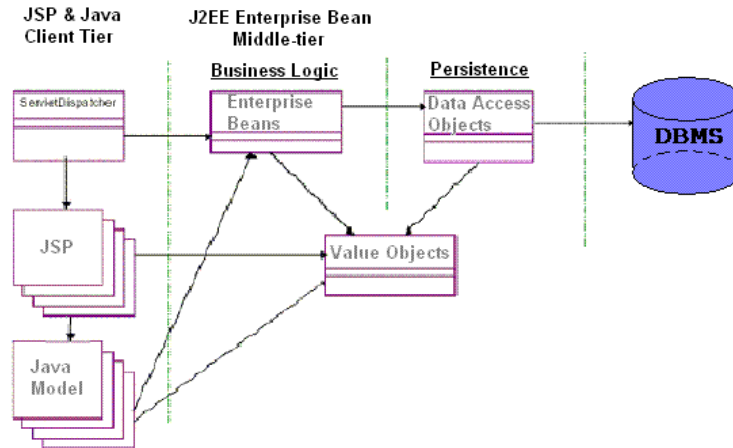


Figure 4 Three-tiered System

4.4 System Security

The J2EE architecture defines security at two levels: authentication and authorization. When a user logs on to a computer and provides a password, the portion of the system that verifies the user's name and password is performing authentication. The verification process is flexible and can be implemented by taking advantage of the underlying security of the operating system and can be customized.

Through the authorization process, users are granted access to "authorized" portions of the application based on who they are. According to Sun, "Each client belongs to a particular role, and each role is permitted to invoke certain methods."^[4] For example, the system administrator role is given permission to access the unsigned contract functionality of the system but is not allowed to modify any information associated with a signed contract. The roles of the users along with the methods they are allowed to access are not defined in the code but at the deployment level.

5.0 ASP Handoff

After the completion of the project, the application is handed off to a local ASP. Dataware will be charged a yearly fee for this service. This fee is less than the purchase and maintenance of competing software and provides many benefits. Users are able to access the system from anywhere they have Web access without compromising data security. Dataware is not responsible for the backup of the data. The ASP will work with Dataware to determine security requirements of the application and data. The ASP will discuss the issues of offsite data storage with Dataware. A tradeoff exists in accessibility and autonomy in the ASP model. Dataware needs to establish a trusting relationship with the ASP in which they determine that the ASP is capable of hosting their data professionally and securely.

6.0 Conclusion

The University of South Dakota's Computer Science Department feels that the development of a "real world" distributed system using a model-driven design approach and the J2EE architecture is a valuable experience for the students. This two-semester sequence is a culmination of many of the skills developed in other courses. Because of

the growing complexity of software, the introduction of the ASP model is appropriate. Feedback from students indicates that this approach helps them find jobs and makes them productive more quickly in their new job.

7. References

- [1] Patnayakuni R., Seth N., *Why License When You Can Rent?*, Proceedings of the 2001 ACM SIGCPR conference on Computer personnel research, San Diego, California, United States, Year of Publication: 2001, pp.: 182 – 188.
- [2] Frank Dzubeck, *Application service providers: An Old Idea Made New*, Network World,” 08/23/99.
- [3] Application Service Providers (ASPs), iStart New Zealand’s e-Business Portal, <http://www.istart.co.nz/index/HM20/IS1/IP21886>.
- [4] Sun ONE Application Framework Overview <http://docs.sun.com/source/817-0447-10/s1afovew.html>
- [5] Alur,D., Crupi, J., Malks, D., *Core J2EE Patterns, Best Practices and Design Strategies*, Sun Microsystems, Inc., Copyright 2001, pp. 9.
- [6] Quatrani, T, *Visual Modeling with Rational Rose 2002 and UML 3/e*, Addison Wesley Professional, Copyright 2003, pp. 26.
- [7] Brown, D, *An Introduction to Object-Oriented Analysis*, Wiley, Copyright 2002, pp. 197
- [8] Cunningham, W. and Beck, K.: *A Diagram for Object-Oriented Programs*, in Proceedings of OOPSLA-86, October 1986