Architect Enterprise Applications with Java EE

Activity Guide

D68136GC10
Edition 1.0
March 2013
D71727

ORACLE®

Author

Tom McGinn, Joe Boulenouar, Joe Greenwald

Technical Contributors and Reviewers

Linda deMichiel, Ian Evans, Roberto Chinnici, William Markos, Shreechar Ganapaphy, Vishal Mehray

This book was published using: Oracle Tutor

Table of Contents

Preface

Profile

Before You Begin This Course

- Thorough knowledge of distributed computing and communication concepts, Java EE technologies, experience with notation languages such as UML
- Working experience with the analysis and design of object-oriented software systems

How This Course Is Organized

This is an instructor-led course featuring lectures and hands-on exercises. Online demonstrations and written practice sessions reinforce the concepts and skills introduced.

Related Publications

Additional Publications

- System release bulletins
- Installation and user’s guides
- *Read-me* files
- International Oracle User’s Group (IOUG) articles
- *Oracle Magazine*

Practices for Lesson 1

Chapter 1

Practices for Lesson 1

Practices Overview

In these practices, you will test your understanding of Enterprise Architecture.

Practice 1-1: Exploring Enterprise Architecture

Fill in the blanks with the correct answers:

1. Using your site to answer this question, what are the three top tasks or responsibilities of the EA? EAA? ____

   ____

   ____

2. Be prepared to give an “elevator speech” defining what Enterprise Architecture is and the need for it.

3. Which of the following are likely Enterprise Architect or Enterprise Application Architect functions or tasks? Why or Why not?
   a. Database migration and tuning
   b. Liaison between system designers and development team
   c. Lead facilitated workshops to derive functional requirements
   d. Model aspects of the application architecture
   e. Propose design and implementation recommendations for hardware and server software
   f. Determine interfaces for distributed services
   g. Choose the ORM solution provider
   h. Ensure IT goals and direction align with the business goals
   i. Participate in social functions with the CIO

Practices for Lesson 2

Chapter 2

Practices for Lesson 2

Practices Overview

In these practices, you will model the initial architecture.

Practice 1-0: Exploring Enterprise Architecture

Business Background:

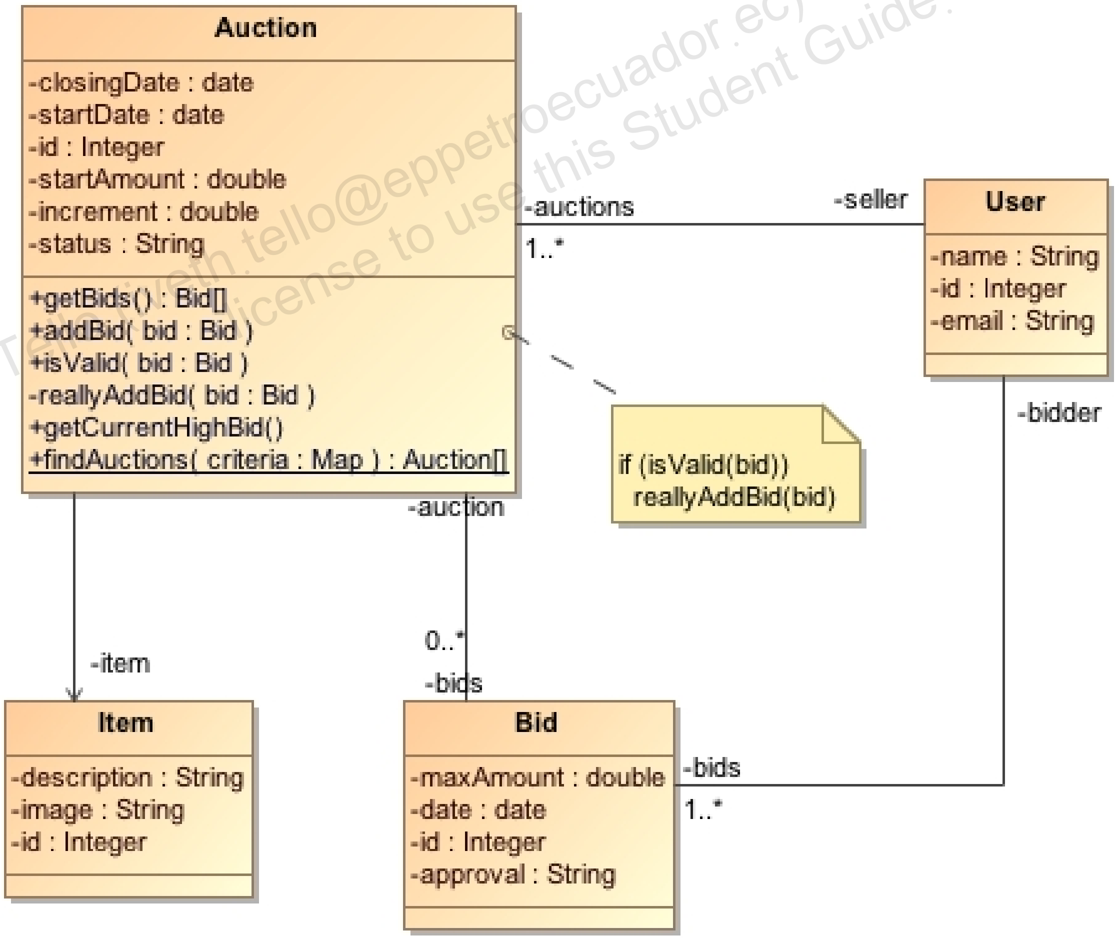
Your company is a leading on-line retailer that has decided to expand into the Auction business. The business analysts have delivered to you the requirements as a starter set of diagrams: a basic Domain model and set of Use Case descriptions (below).

For this exercise, you are to decompose the system into subsystems based on the provided requirements. As this is the first draft, these be done at a high level.

Per your site standards, as the Enterprise Application Architect, you are to draw up an initial set of UML diagrams for the architecture. You are to create a high level Component and Deployment diagram for a Java EE architecture.

For this first draft you should assume that you are deploying the Auction system as an enterprise application on a single Java EE server running on a single JVM implementation.

Domain Model



Use Cases

Login: All users of the Auction system must first log on to the system, before they can invoke other operations. To log on, users must authenticate themselves, for example, they could supply an user ID and a password. This use case is related to the addUser and findUser use cases.

Add User: Adds a new user to the system database. This use case is invoked by the login use case, when a new user accesses the Auction system for the first time. In addition to the user ID and password already supplied, the system asks the user for additional personal information: name, address (both location and email).

Find User: Finds an existing user in the system database. This use case is included by the login use case, when an existing user returns to the Auction system.

Create Auction: Sellers are allowed to create auctions, to auction items off. Each auction holds information, such as the item being auctioned, the initial bid price, the date the auction starts and the date it will end (or how long the auction will last), and a list of all bids placed on it.

Place Bid: Bidders are allowed to bid on auctions. Given an auction, the bidder is allowed to enter a bid on that auction. The Auction system should not allow a bidder to place a bid on an auction that is lower than the current bid price for that auction. The Auction system should also allow the user to find out what the current bid price for the auction is, or even notify the user automatically when the current bid price for the auction changes, or when the bidder is no longer the current high bidder for the auction. The system keeps a list of all auctions a bidder is currently bidding on. The bidder can retrieve and view this list.

Find Auction: Users must be able to provide criteria to be used to retrieve a collection of auctions in which the user might be interested.

You are also given a number of non-functional requirements to keep in mind as you consider how to build this Auction system:

Portability: It has been requested that the Auction system be implemented as a web-based application in which the client side is straight HTML. This maximizes the likelihood that clients can interact with the Auction system, because almost every platform supports rendering of straight HTML.

Scalability: The application's runtime characteristics must remain acceptable as the number of clients, or requests per client, increases.

Performance: Response time when placing bids, or when selecting auctions to bid on, must remain short.

Tasks:

1. Identify the high-level subsystems (also referred to as services or components) that will be needed for a Java EE architecture and model these using a UML Component diagram. Using the Component diagram, draw the Deployment diagram showing where these components will be deployed.

2. Create the diagrams using one of the following approaches:
   a. Use the provided tool: UMLet to create the diagrams. Navigate to the Desktop folder and start the tool by double-clicking the **umlet** executable. Alternatively, open a terminal window and navigate to the Desktop and run the application using Java:

```
cd D:\WINNT\Profiles\Administrator\Desktop\UMLet
java -jar umlet.jar
```

   b. Use Flip Charts in the classroom or draw on the white board. This option is recommended when the classroom supports ample white board space.

Consider the UML models that you produced, and identify the risks that might be associated with that architecture.

Practices for Lesson 3

Chapter 3

Practices for Lesson 3

Practices Overview

In these practices, you will consider architectural choices for Security.

Practice 3-1: Describing Risks in the Auction system

All web-based, Internet-based applications must deal with security risks. Because the application is deployed over the Internet, the server side does not have control over the actual user interacting with the system, nor the application actually running on the client, nor the network that communications will travel on, between the client and server sides.

In this practice, you consider the following:

- The security risks with which an architect should be familiar
- The parts of the model of the Auction system that are vulnerable to these risks.
- The strategies you can employ to address these risks.

In Practice 2-1, you sketched a number of UML diagrams to describe a number of use cases for the Auction system.

Tasks:

1. Annotate your UML Component diagram with the security risks that are applicable to each use case, indicating where in the interaction the security risk could apply. Also, indicate which strategy you would introduce to address each risk, modifying the UML diagrams when needed.

Practices for Lesson 4

Chapter 4

Practices for Lesson 4

Practices Overview

In these practices, you will weigh capacity planning and performance options.

Practice 4-1: Architect for multiple tiers

The requirements for the system have increased:

- The current system, running inside a JVM on a single server, was sufficient for POC and initial startup, but the site has become popular and is no longer able to handle the capacity:
    - The number of concurrent users has risen sharply and is expected to increase at a rate of 10 – 25% per month.
    - Performance is noticeably degraded. For example: pages are taking a long time to display and many users are experiencing long wait times when submitting bids. Queries to search for auctions are taking many seconds to complete. Reports for managers are taking too long to run. Some bids have been lost or not recorded and some auctions have simply "disappeared' for no apparent reason. This has led to legal issues for your company.
    - Frequent downtimes and periods where the system is unavailable have been occurring. This needs to be resolved. Up times greater than 99.99% are needed. The business is experiencing significant revenue loss as a result.
- As popularity of the site increases, the business is realizing it will have to make changes to the application over time. The system should be architected in such a way as to make both application design and system performance and reliability easier to manage and expand over time.

Tasks:

1. Consider the topics discussed in the lesson.
    - What questions would you ask to further refine the business needs and requirements? What could you ask to help prioritize their QoS needs?
    - What options in terms of: hardware, software, design and architecture are available to you to meet and exceed these requirements and issues?
    - What are the constraints on your available choices?
    - How would you prioritize what needs to be done in terms of architecture? What three things need to be addressed first? Plan the rollout of your choices and their impact on systems and people.
        - Hint: Use the matrix (Prioritization of QoS Requirements) presented in the lesson as a structure for organizing your decisions.
2. Prepare a short (3 minute or less) "sales pitch" to management to present your proposed architecture choices. Be sure to identify the benefits as well as potential risks and trade-offs. Also, describe the next steps you would take and why.
3. Update your logical UML diagrams to show the changes to your architecture as a result of your choices above.

Practices for Lesson 5

Chapter 5

Practices for Lesson 5

Practices Overview

In these practices, you will modify your object model to increase flexibility and identify services.

Practice 5-1: Design a Flexible Object Model

The business has given you additional requirements to add to your current architecture.

- It is important to distinguish between sellers and bidders because there are tasks in the Auction system that only one kind of user may request.
- New Use Case: **Pay Bid:** Bidders must be able to pay for their winning bid on an auction, once the action is over. The Auction system asks for credit card information at the time of payment. The system should also notify the seller of that auction, when the winning bidder actually pays for that auction. You will use an external clearinghouse payment processor.
- The user can provide credit card information to be used as a default method of payment. This should be saved so they do not have to reenter the information whenever they win an auction.
- New Use Case: **Add Item:** Sellers are allowed to create items to be auctioned off. The item to be auctioned and the auction for the item are considered to be independent from each other.
- The Auction system should also allow the seller to create multiple auctions for a number of copies of the same item conveniently. This can be achieved by allowing the seller to create a first auction, and then allowing the seller to create another just like it, or by allowing the seller to specify the number of copies of an auction to create, as part of the process of creating each auction.
- There is disagreement as to whether to use a database or LDAP server for user authentication. You should modify your architecture to easily account for either of these solutions later in the development.

Tasks:

1. Modify your Domain model to address these new requirements.
2. Modify your architecture to incorporate your changes. Identify and include new or changed subsystems or components to support this functionality.
3. Make a first cut “guess” at fine grained as well as course grained services that will be needed.

Practices for Lesson 6

Chapter 6

Practices for Lesson 6

Practices Overview

In these practices, you will plan for your transactions.

Practice 6-1: Transaction Planning

Tasks:

1. As a class, walk through documenting one use case's transaction with a Sequence diagram. Use the use case: Adding a New User.
2. In pairs, document one transaction from the list provided below:
   a. Place Bid
   b. Create Auction
   c. Pay for Bid (**Note:** Consider the implication if you choose to offer the second highest bidder a chance to "win" the item if the highest bidder refuses to pay.)
3. As a class, create a "prototype" bid system that does not use transactions. What impact does this have on your architecture? What are the risks associated with this approach? What are the benefits? Are there sufficient reasons to consider this as a viable approach? If so, what are they?
4. As a class, analyze your transaction designs (1-3) with respect to possible service loss. What happens if there is a server crash and a bid is lost? Or an auction is lost? Or an auction cannot complete? Also, consider the effect of pessimistic versus optimistic database locking on your transaction.

Practices for Lesson 7

Chapter 7

Practices for Lesson 7

Practices Overview

In these practices, you will consider how you would modify your existing architecture to take advantage of the new features in Java EE 6.

Practice 7-1: Consider impact of Java EE 6 new features on your architecture

Java EE 6 introduces some significant new architectural features and options. These include (but are not limited to):

- EJBs and servlets can be called asynchronously.
- CDI and the new definition of beans eliminate the need for supporting, helper classes.
- Services can be exposed as JAX-WS or JAX-RS and implemented by POJOS and EJBs.

Tasks:

1. Each group should consider their existing architecture in light of these (and other) new features and be prepared to answer the following questions:
   a) What major and minor changes would you make to your existing architecture in light of these new features?
   b) What impact do these new features and capabilities have on design, application architecture, and enterprise architecture?
2. Assume your management is reluctant to migrate from their existing Java EE 5 platform to Java EE 6. Craft a short (3 minutes) presentation to convince management why they should consider upgrading to Java EE 6. One group should take the position: yes we should upgrade. Another group should take the counter position.
3. (Optional) If there is sufficient time and interest, a group may choose to alter their solution architecture to incorporate changes based on Java EE 6 new features.

Practices for Lesson 8

Chapter 8

Practices for Lesson 8

Practices Overview

In these practices, you will consider Client tier architectural choices.

Practice 8-1: Architecture for the Client Tier

The client has requested that you enhance the user experience of the Find Auction use case in the Auction system. The suggestions that the client wants you to implement include:

- Improve the user interface. Make the user interface more fun to use (that is, use of the Aqua user interface: pulsing buttons, photo-realistic icons, and so on). Make it look "better."
- Make the user interface more interactive. As currently implemented, the user can request that the system find auctions that match the criteria the user specified, but the list of auctions found is static. The client suggested that it would be better if this list were interactive:
  - When new auctions are added to the system, and those new auctions match the criteria specified by a user currently examining the corresponding list of auctions, the list should automatically update to include the new auctions.
  - When information for an auction that's included in the results of a search (such as the current high bid price) is changed by some other user (by placing a greater bid on that auction, for example), the list of auctions should automatically update to present the new high bid price for the auction that changed. This can be achieved currently, but only by asking the system to repeat the current search, which is unacceptably slow and unfriendly to the user.
- The system seems to take too long, between the time the user requests that a search be performed or updated, and the time the screen is updated; or between the time the user requests to scroll to the next page in the resulting list, and the time the screen is updated.
- The user has requested the application be made available on mobile devices like smart phones and tablets.

Changes to address these concerns cannot compromise the application's ability to run on any platform currently supported.

Tasks:

1. Outline potential strategies, both application-level and in terms of technology choices that you could use to address these requirements. For each strategy listed, outline the way in which that choice could be used to address the concerns listed above.

2. For each strategy choice, consider the tradeoffs and risks associated with it. How can you find out and weigh the benefits and costs?

Practices for Lesson 9

Chapter 9

Practices for Lesson 9

Practices Overview

In these practices, you will consider Web tier architectural choices.

Practice 9-1: Architecture for the Web Tier

The preferred designs for the Auction system have revolved around a web-based front-end to the application, for portability. With such designs, all the client requires is a web-browser to render the HTML-based user interface (UI) to the application.

In Practice 8, you incorporated some HTML-related client- or server-side technologies into your design (such as AJAX) to improve its QoS characteristics.

A consequence of this approach is that most of the logic responsible for the front-end to the application actually resides on the server side, in the Web container that is responsible for generating the HTML-based UI on-the-fly, on a per-client, or per-session basis.

You previously sketched a high-level possible description for the Find Auction and Place Bid interactions. However, the actual application-level structure of the Web tier for the application is not really specified. You've only mentioned the Web components responsible for generating HTML at a rather high level. This lab asks you to architect a more detailed structure for the Web tier for the Auction system.

You have been given certain constraints, as far as the characteristics of the web-based application are concerned:

Portability: The application must be deployable to the widest variety of client platforms.

Maintainability: Changes to the flow through the user interface (in terms of the order in which pages must be displayed, or the ability to introduce additional pages into the flow) must require minimal effort.

Duplication of effort should be avoided. For example, when multiple pages incorporate the same fragments, you should not have to code this more than once.

Tasks:

1. Consider the patterns available for use in architecting the Web tier. Which ones would you use and why?
2. Consider some of the changes to the Java EE architecture made possible with Java EE 6. How and where could you incorporate these changes into your architecture? For example, using Java EE 5 you might use a polling model. Using Java EE 6 you might use a push model for notification.
3. Choose a Web framework to use (ex: Struts, JSP-Servlet, JSF, ADF Faces, Spring Web Flow) in your system. Be prepared to give a short (3 minutes) presentation on what you chose and why. Make sure to address risks as well as tradeoffs. You should be considering; cost, ease of use, open standards, availability of developers, training issues, performance, and so on.
4. Add to your existing Component diagram to show the additional detail in the Web tier needed to support your choice for Client tier.

Practices for Lesson 10

Chapter 10

Practices for Lesson 10

Practices Overview

In these practices, you will consider Business tier architectural choices.

Practice 10-1: Architecture for the Business Tier

For this practice you will focus on detailing the Business tier architecture and design needed to support the two use cases: Find Auction and Place Bid.

Tasks:

1. Enumerate Java EE technologies that you could leverage to build the business tier for the Auction system. Indicate how each of the technologies selected could help you achieve any of the functional or non-functional requirements (maintainability, correctness, scalability, performance.)
2. If you were to choose to make some subsystems available as Web services, how might this change your architecture? Consider the impact of SOAP-based vs. Restful web services.
3. For each of the choices made in Task 1, in terms of how you would address the requirements given in your OO model for the auction system, consider whether there is an alternative to address the same requirement. The alternative might be to build application-level functionality or to delegate the requirement to the underlying Java EE platform. The alternative could also be in the choice of Java EE technology provided by the underlying platform with which you choose to leverage your application-level structure. Discuss the trade-offs associated with each of the alternatives you identified.
4. Update your UML models to reflect the changes you have made form the above questions.

Practices for Lesson 11

Chapter 11

Practices for Lesson 11

Practices Overview

In these practices, you will consider Integration and Resource tier architectural choices.

When a bidder wins an auction, you do not want to be responsible for the actual processing of credit card payment requests, or the actual transfers of funds to the sellers' (credit card or bank) accounts. Traditionally, online e-commerce Web applications rely on third-party credit card processing or bank clearinghouses to take care of financial operations. You need to interact with these third-party clearinghouses over the Internet and it might be that these clearinghouses are not Java EE-based services.

There are a number of non-functional requirements to keep in mind, as you consider how to build this part of our Auction system:

Portability: The system needs to be able to interact with third-party clearing houses, regardless of their protocol or implementation.

Maintainability: The changes that are required to switch the system from one third-party clearinghouse to another should be minimal.

Performance: The interaction with the third-party clearinghouse might be expensive or slow, but the Auction system must never block clients for significant lengths of time. Also, it might be that the clearinghouse only accepts financial transactions in batches. But you cannot ask the users to wait until other transactions are ready to issue them all in batch form to the third-party clearinghouse.

Correctness: The system should perform its own operations plus the clearinghouse-based financial processing as a single operation. You should not mark the auction as paid until and unless the clearinghouse processes the payment successfully.

Practice 11-1: Integrate with External Service Providers

Tasks:

1. List the integration technologies (Java EE or otherwise) that you could leverage to build this part of the Auction application, along with the way in which you would use each one to achieve the requirements listed above. For each one, include the risks associated with using it how you could mitigate these risks as well as the tradeoffs associated with choosing it. Risks and tradeoffs should include, but are not limited to: security, performance, interoperability, transactional capability (i.e.: ability to rollback or compensate should a part of the transaction fail or time out), impact on architecture, design and implementation factors, staffing and cultural issues.
2. Describe how you could use Java EE 6 technology to capture an interaction between the Auction system and the third-party clearinghouse that allows the Auction system to initiate a request to the clearinghouse to process a payment without requiring that the Auction system block while waiting for a response from the clearinghouse, but makes the response available to the Auction system as soon as it is available. Look for the most maintainable means to achieve this interaction.
3. Update your UML diagrams to reflect your architecture choices.

Practice 11-2: Using the Auction System as a Service

Tasks:

So far, you have assumed that the Auction system is an interactive Web application to be used by individual buyers and sellers over the Internet to create one item or one auction at a time. However, it might be attractive for you to enter into partnership with other retail sales outlets and enable them to make batch loads of items and auctions.

1. Defining Service APIs.
   a. Define a service API that allows you to offer support for batch loads of items and batch creation of auctions.
   b. Consider how you will allow outside vendors to query and obtain report data on their items and auctions. What types of data would you provide? How would you do so? What impact would this reporting requirement have on your current architecture?
   c. List Java EE technologies that could be leveraged to build the service API to the Auction system, and sketch the way in which the API is implemented.
   d. List trade-offs associated with your approach.
2. Update your UML diagrams to reflect the changes in your architecture for these tasks.

Practice 11-3: Software As a Service Discussion

You've been approached by a large retailer (such as "Buy More, Inc") to allow them to offer auction services on their Web site by relying on your Auction system to provide them with back-end support. The Auction system supplies their system with the business knowledge necessary to run auctions; they provide a front-end to handle auctions in such a way that the user interface integrates with the rest of their retail web site.

Tasks:

1. Consider the requirement as stated. Is it feasible? Is it clear, concise and complete? Is it reasonable? If not, why not and what questions would you ask to clarify and refine the requirement?
2. Be prepared to present and defend your answers. What effect, if any, would supporting this requirement have on your architecture? What are the risks associated with supporting this requirement?

Practices for Lesson 12

Chapter 12

Practices for Lesson 12

Practices Overview: MegaBank Case Study

In these practices, you will evaluate system requirements and make architectural choices, considering trade-offs.

Problem Summary

MegaBank (MB) is a traditional, brick and mortar, international bank. MB has nearly 100000 account holders worldwide. Analysts working for MB have determined that to successfully compete with other banks, MB must provide online banking services to its clients.

In pursuit of this goal, MB recently acquired a smaller, regional bank called Thirteenth National Bank (TNB). TNB has just over 35000 account holders and is well known for its customer loyalty. TNB has been providing online banking services to its customers for a number of years, which is of great interest to MB.

MB has decided to implement the TNB banking services throughout the international corporation. MB currently has a completely internal system with various applications used by its staff. TNB has both internal applications, as well as the online, customer-oriented applications.

MB management wants you to create an initial architecture in the form of a proposal to solve its current problems. MB would like to leverage the existing applications that are owned by MB and TNB and combine them into a single, unified architecture. MB recognizes the need to provide sophisticated online services, as well as the potential for business-to-business (B2B) integration that online services might bring in the future.

Functional Requirements

MB is concerned about the user interface for this online application. Because MB does not expect its users to be highly technical, every effort should be made to eliminate unnecessary or confusing steps. In short, MB wants the interface to be simple, consistent, and easy to use.

Though not a requirement, MB would someday like to have its internal staff use portions of this application. For example, customers should be able to view its past transactions online, and they should also be able to walk into the bank and have a teller give them a list of their past transactions. The same process could be used in both cases.

MB has already identified the following requirements for its online application:

- View account balances
- View pending transactions
- View transaction history
- Transfer funds between user accounts
- Buy stocks
- Sell stocks
- View stock portfolio
- View list of unpaid bills
- Pay a specific vendor
- Configure automatic bill payment rules:
  - Date of payment
  - Amount of payment
  - Account to debit

- Email and telephone support on a 24/7 schedule
- Update user information
- Change password

MB might have another process currently in place to fulfill some of these requirements, such as opening and closing accounts, and the existing TNB solution might fulfill other requirements.

It is expected that MB will use its existing databases to store all account and customer data.

Along with the user-oriented requirements in the previous list, the tellers and managers in the banking staff have identified the following requirements:

- Open new accounts
- Close existing accounts
- Set up online banking for users
- Correct errors with account balances
- Use OLAP tools to analyze customer data

MB also requires that the application provide some form of authentication. All users of this system must log in before accessing any of the services. Online users will only receive access by means of the Web. All interaction with bank data will be mediated by the application itself. Finally, all users should log off when they are finished with the system.

MB also has discussed the possibility of future enhancements to this application. MB would like to ensure that the architecture is capable of handling any of the following possible additions:

- A premium service for some customers, which would provide streaming stock information with the latest prices and market conditions
- Brokerage services for enhanced stock management, which might include automated buying and selling of stock based on user supplied criteria or interaction with actual brokers
- An online loan application service

Note – This is not an all-inclusive list of requirements, and you might discover some additional functional requirements as you analyze the MegaBank scenario.

Non-Functional Requirements

Flexibility

MB has identified flexibility as a paramount concern for this application. MB is willing to use its existing infrastructure (or portions of it) for now, but MB wants to avoid “vendor lock” in the future.

Scalability

With the acquisition of TNB, MB has about 135000 customers. MB would like to grow this user base by at least 20 percent in the next quarter. TNB reports that about half (or about 17500) of its customers are using the online services that TNB currently provides. However, there is no information about the concurrency of these users, or about how often the customers are using the application.

Availability

Because this application will often be the only interface that many users have with MB in the future, MB requires no less than 99 percent availability. MB is willing to invest in infrastructure, if needed, to achieve this goal because MB will save money in the long run by reducing paperwork and staff.

Reliability

MB will not accept more than one failure per every 100000 transactions, a reliability rate of 99.999 percent. MB places a high importance on reliability because of its industry. MB knows that users will not accept any glitches or errors in online monetary transactions. Ultimately, if the customers do not believe that the online application is safe and reliable, they will not use the application and might leave MB altogether.

Performance

MB recognizes that part of the benefit of online banking is the savings in user time. Therefore, MB wants a system that is responsive to user requests. While MB understands that there will be various bandwidths and connections from its users, MB feels that it is reasonable to expect a performance of no more than three seconds per request, as measured at the MB firewall.

Extensibility

One problem with the existing TNB online application is a lack of extensibility. Consequently, MB wants to ensure that this new architecture is extensible. MB already has plans for future enhancements, and MB wants a system that will allow for easy and cost efficient additions.

Security

MB is very concerned about security, and MB knows that its customers will share this concern. MB is mandating the use of the SSL protocol for all access to the online application. MB also wants the system to track and log all failed login attempts, so that after three failed attempts, the user account is “locked”. If an account is locked, the user will have to visit a branch of the bank to correct the problem. MB wants firewalls to protect its sensitive, internal data.

Manageability

MB has a small staff of system administrators, who will manage the online application after it is deployed. To avoid major manageability problems, MB wants to ensure that the architecture is only as complex as it needs to be to meet the functional and non-functional requirements

Existing Infrastructure

MB has the following software and hardware in place:

- An IBM AS/400 server that hosts MB’s existing applications. These applications are written mostly in COBOL, with some C and C++ that has been added along the way to support enhancements.
- An IBM DB2 database that stores MB’s customer and account information.
- An external clearinghouse that serves as an intermediary between banks, businesses, and other entities that need to transfer funds. The clearinghouse mainly handles checking account activity, but also handles any external fund transfers. All external transactions are processed nightly.
- An existing DB2 OLAP application that permits bank staff to analyze data to determine the usefulness of MB’s services. This application is provided by IBM and runs on the existing AS/400 server.

TNB has the following software, services, and hardware in place:

- A recently purchased Sun E10000 server (Starfire™ hardware) that hosts TNB’s existing application with the Solaris™ 8 Operating Environment (Solaris OE) as the operating system.
- A third party company that handles static web content.

- A centralized server that runs on the Sun E10000 server and is housed and maintained by TNB. This server manages all of the dynamic web content, which is CGI-based and written in Perl.
- A component in the application server that provides the rule-based logic for account processing. The business logic is written in COBOL and C, as well as a proprietary language provided by TNB's application server.
- An ORACLE® RDBMS that stores all data.

Assumptions and Constraints

MB has identified the following assumptions and constraints for the online application:

- Java EE technologies should be used. While other technologies could be used, the MB analysts prefer the standard of Java EE technology.
- Payments must not exceed the customer balance. No payment can be made for more than the amount available in the current account balance. Overdraft protection will not be offered in the initial release of the online application.
- The user must have at least one account with MB to be considered a customer.
- The interface should take bandwidth into consideration. Users will be accessing the application with varying bandwidths, and users with higher bandwidth connections could have a more robust interface than those with lesser bandwidth connections.
- The application that MB uses to interface with the external clearinghouse is proprietary. MB would like to continue to use this application. The application supports a CORBA interface.
- Stock market data is currently accessed by way of a proprietary messaging application owned by TNB. This application receives information from a third-party service. MB is interested in having a more "open" application that would enable them to choose different stock monitoring service providers in the future.

Risks

MB has identified the following risks as critical considerations for the online application:

- Though there is a small development group inside TNB that has familiarity with J2EE technology, MB has only COBOL and C/C++ developer.
- To provide J2EE technology services, a compliant application server will be required.
- All transactions made at the bank should be correlated with any online transactions. For example, if a customer goes to the bank to withdraw funds from a checking account, the balance that the customer sees online should reflect the pending withdrawal.
- Some customers will not be comfortable with using an online interface.

Note – While these risks are the most critical for MB stakeholders, the list is incomplete. Expand the list with any additional risks that you identify for this architecture.

Practice 12-1: System Requirements, risks and assumptions.

Tasks:

1. Read the problem summary for MegaBank.
2. List the functional requirements of the system. This might be completed as a class or small-group activity. Consider the following concepts:

- The existing functionality of both banks
- The new functionality that is required for this application

- The integration with existing applications

3. Organize the list of functional requirements into groups of distinct services. Label each service with a clearly defined name, such as Stock Service.
4. Discuss the non-functional requirements with the class or your group, and spend some time discussing their presumed priority. Consider the trade offs that will be required to realize the non-functional requirements.
5. List any other risks and assumptions that you might have discovered beyond what is listed in the problem summary itself.

Practice 12-2: Create use-case diagrams for one group of services

Tasks:

1. Identify and list the actors involved in the MegaBank scenario. Note that a single actor might be involved with several services.
2. Select one service that you wish to work with for the remainder of this lab. From this point forward, you will work with only the service you select.
3. For each functional requirement in your chosen service, document the use-case scenarios. Be sure to include any alternate flows of events.
4. Create a use-case diagram that models all of the use cases and their interactions for your chosen service.

Practice 12-3: Create a class diagram

Tasks:

1. Identify and list the actors involved in the MegaBank scenario. Note that a single actor might be involved with several services.
2. Select one service that you wish to work with for the remainder of this lab. From this point forward, you will work with only the service you select.
3. For each functional requirement in your chosen service, document the use-case scenarios. Be sure to include any alternate flows of events.
4. Create a use-case diagram that models all of the use cases and their interactions for your chosen service.

Practice 12-4: Create a sequence diagram

Tasks:

1. Select one of the use cases from your chosen services and create a sequence diagram.

Practice 12-5: Identify components and Java EE technologies

Tasks:

1. Identify the components that you will need in the system to fulfill your selected use case from Exercise 4. Determine which components you will build, which you will buy, and which are already owned.
2. Analyze the relationships between these components with regard to:
   - Extensibility
   - Containment
   - Dependencies
3. Create a component diagram to capture the key components and their interactions.
4. Determine if there are any components that could benefit from reification. If so, alter your component diagram accordingly.
5. Create a high-level deployment diagram to capture the fundamental applications that are involved, along with their physical location, and the interaction between them.
6. Refine your high-level deployment diagram into a detailed deployment diagram that includes the components