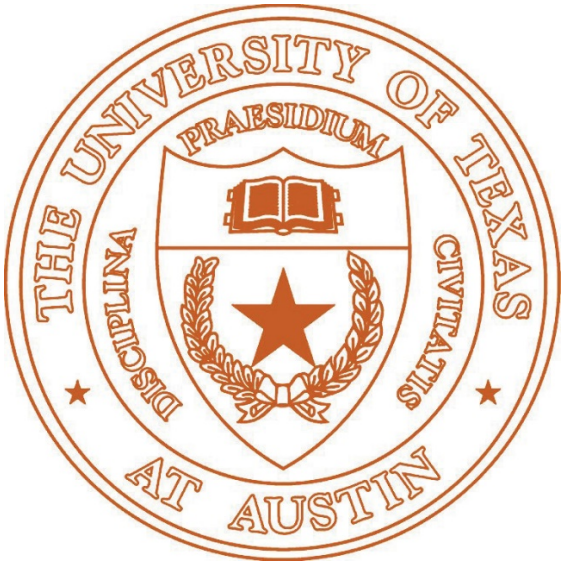
University of Texas at Austin, Cockrell School of Engineering
Software Architecture – EE 382C.7



Assignment # 3
Derivation and Evaluation of Solution and Deployment Blueprints

May 07, 2016

Gabrielson Eapen
EID: EAPENGP

Table of Contents

3. Derivation and Evaluation of Deployment and Solution Blueprints

3.1 Deployment Blueprint #1 (DB1)

Part: 1

Table 1: Satisfaction of domain functions by solutions

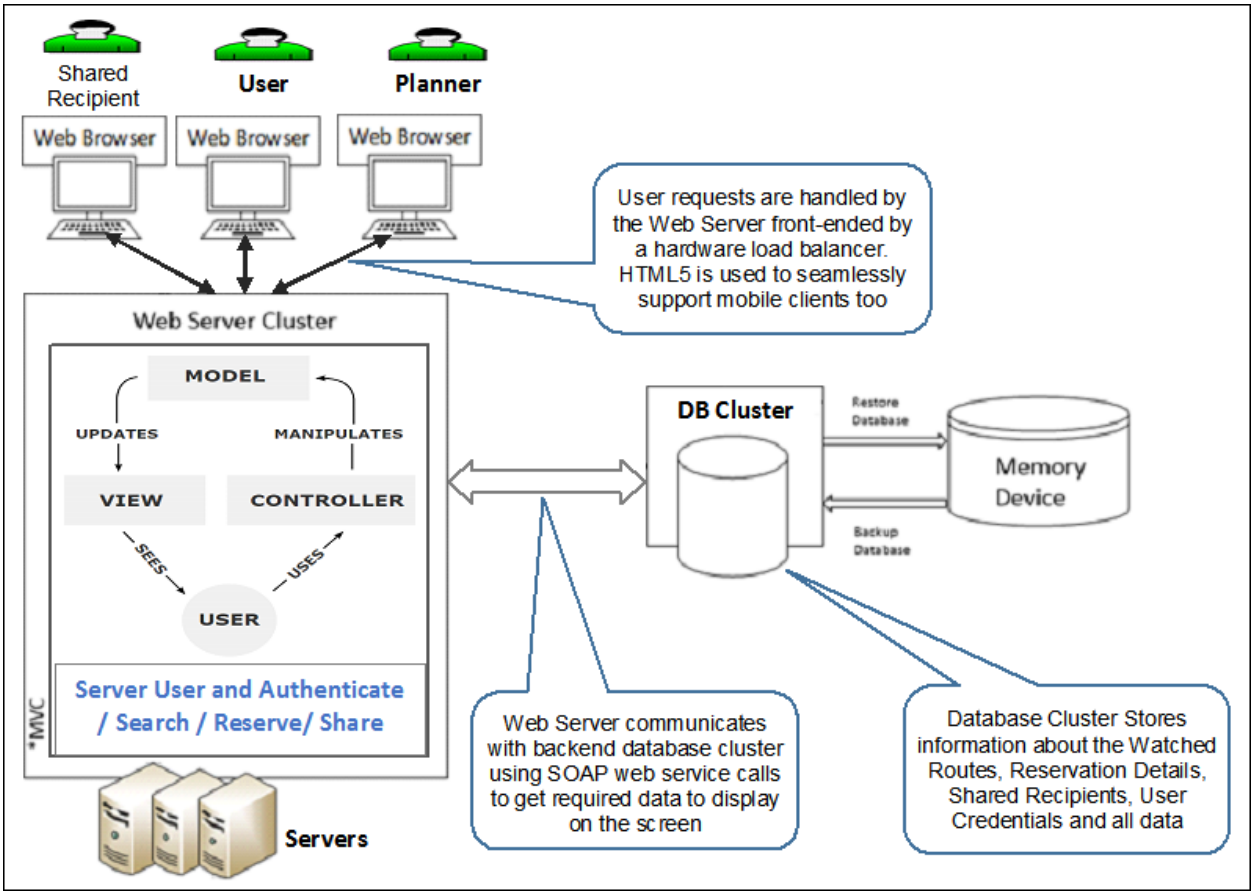
SB Solution Component	BB Functions and Data satisfied
Server User and Authenticate / Search / Reserve / Share	Functions: • Authenticate and Authorize • Search for a flight • Select a seat for a flight • Enter number and types of luggage • Select meal for a flight • Modify an existing reservation • Confirm reservation modifications • Select a flight • Reserve a flight • Sort and filter flight results • Review existing reservation details • Review all reservations • Cancel an existing reservation • Register a new user • View User Profile • Edit User Profile • Update User's Email Address • Update User's Credentials • Add a Travel Planner • List Travel Planner(s) • Remove a Travel Planner • Watch a given route • List watched routes • Remove a watched route • Share Itinerary • Add Recipient to a Shared Itinerary • Pay for a reservation • Process travel credits • Pay for a reservation modification • Add payment details Data: • User Credentials • User Email • Credit Amount • User Reservation List • Trip Reservation Details

	• User ID • User Profile Information • Planner Information • Payment Information • Watched Routes

Table 2: Allocation of Solutions to deployment components

DB Component	SB Components Allocated to DB Component
Web Server Cluster	Server User and Authenticate
Web Server Cluster	Server Search
Web Server Cluster	Server Reservation
Web Server Cluster	Server Share
Database	N/A
Web Browser	N/A

Figure 1: Graphical Depiction of DB1[1]



[1] *MVC: Server User and Authenticate / Search / Reserve / Share modules all use MVC architecture

Rationale

- Satisfaction of Stakeholder qualities (with priorities):
    - Availability: Because of the multiple servers, redundancy exists and there is no single point of failure. An individual server can be down, but the remaining servers keep the application or service online. Even when there is need for planned maintenance, each server can be serviced individually without taking application down.
    - Usability: MVC is a software architectural pattern for implementing user interfaces on computers. Because of separate view, model and controller interconnected components, different views can be created for Shared Recipient, User (Traveller) and Planner with the use of same model and controller and thereby improve usability.
    - Performance: User requests are distributed among servers with the use of a hardware load balancer in web farm. Keeping the ratio of requests per server at lower level, reduces the work each server has to do. Contention is minimized and it does not cause additional slowdown or performance impact.
    - Security: Web service proxy is used to call web service from UI which secures the system. Also, XML encryption is used by SOAP web service to secure user confidential data and payment information. Furthermore, the data is stored on the separate database server, which offers a more control of security than on client machines.
    - Scalability: Load balancer and multiple servers can manage increasing number of user requests. By increasing the scalability will not affect performance of the system. Also if required new server can be added easily and load balancer will start distributing requests on the new server as well to manage scalability of the system.
    - Extensibility: Everything about MVC has been designed with extensibility in mind. Anything in its processing pipeline is replaceable and if needed new services can be injected into the main pipeline. For example, view component can be extended without making changes in model or controller. Similarly new services or features can be added without changing UI or database.
    - Data backup and recovery: Additional memory disk (storage) is used to back up data regularly. In recovery scenarios or if data is accidentally deleted we have ability to (rollback) restore our database.
    - Maintainability: User interface requirements tend to change more rapidly than business rules. Users may prefer different colors, fonts, screen layouts, and levels of support for any new devices or mobile clients. Just as we saw with extensibility, because the model does not depend on the views, adding new types of views to the system generally does not affect the model. As a result, the scope of change is confined to only the view component.
    - Project Schedule: MVC divides application into three interconnected components called view, model and controller. So that makes it easy to estimate time for a task and track it to maintain the project schedule.
    - Project Cost: As we are using web server cluster and database cluster, the onetime installation cost may be high. But it is the least prioritized constraint as we are a new player in this industry segment. It can also be argued that maintenance cost, development time will be less with MVC and this can be used to recoup other costs. In addition most of the applications and software used are open source which incurs less cost.

Design inspired by:

- MVC design pattern: Model view controller separates the application logic from the user interface and the control between the user interface and the application logic. Planner, Shared Recipient and User (Traveler) can have different views from same functionality.
- Object Oriented Architecture: Inheritance is used for similar based functionality or related functions such as select a meal for the flight or select a seat for the flight.
- Web server redundancy: Web server requests are distributed across multiple Servers in cluster. It increases availability, redundancy and performance of the system.
- Web Service Proxy Pattern: Proxying web services secures web service call from the User Interface to secure the application.

References:

- MVC design pattern: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller
- object-oriented architecture: https://www.classes.cs.uchicago.edu/archive/2013/spring/51050-1/lectures/lecture.1/lecture.1.pdf
- Web server redundancy: https://rimuhosting.com/knowledgebase/rimuhosting/load-balancing-and-failover
- Web Service Proxy Pattern: https://docs.mulesoft.com/mule-user-guide/v/3.5/web-service-proxy-pattern

Notes:

- Independent software modules run under the web servers that support Server User and Authenticate / Search / Reserve / Share modules. These modules satisfy domain functions. However, there may be multiple implementation components involved in satisfying a domain function and an "Implementation Task Decomposition" decomposes a domain function into steps.
- There is no way to directly show BB functions in the DB. BB functions are distributed in solution component and showed how it will be fulfilled in the DB

Part: 2

Table 3: Components and functions in business blueprint

BB Component	BB Functions
Authenticator	Functions: • authenticateAndAuthorize • updateUserCredentials • updateUserEmail
Payment Handler	Functions: • payForAReservation • processTravelCredits • payForAReservationModification

Reservation Handler	Functions: • reserveAFlight • reviewAllReservations • reviewExistingReservationDetail • modifyExistingReservation • confirmReservationModification • cancelExistingReservation • selectMealForFlight • selectSeatForFlight • enterLuggageDetails
User Handler	Functions: • addPaymentDetails • registerANewUser • viewUserProfile • editUserprofile • addATravelPlaner • listTravelPlanners • removeATravelPlanner
Search Handler	Functions: • searchFlights • sortAndFilterFlightResults • selectAFlight • watchAGivenRoute • listWatchedRoutes • removeAWatchedRoute
Social Handler	Functions: • shareItinerary • addRecipientToSharedItinerary

Table 4: Technologies and Functions provided by each technology

Technology	Functions provided by each Technology
Web Browser	Functions: • Authenticate and Authorize • Search for a flight • Select a seat for a flight • Enter number and types of luggage • Select meal for a flight • Modify an existing reservation • Confirm reservation modifications • Select a flight

	• Reserve a flight • Sort and filter flight results • Review existing reservation details • Review all reservations • Cancel an existing reservation • Register a new user • View User Profile • Edit User Profile • Update User's Email Address • Update User's Credentials • Add a Travel Planner • List Travel Planner(s) • Remove a Travel Planner • Watch a given route • List watched routes • Remove a watched route • Share Itinerary • Add Recipient to a Shared Itinerary • Pay for a reservation • Process travel credits • Pay for a reservation modification • Add payment details
Web Server	Functions: • Authenticate and Authorize • Search for a flight • Select a seat for a flight • Enter number and types of luggage • Select meal for a flight • Modify an existing reservation • Confirm reservation modifications • Select a flight • Reserve a flight • Sort and filter flight results • Review existing reservation details • Review all reservations • Cancel an existing reservation • Register a new user • View User Profile • Edit User Profile • Update User's Email Address • Update User's Credentials • Add a Travel Planner • List Travel Planner(s) • Remove a Travel Planner • Watch a given route

	• List watched routes • Remove a watched route • Share Itinerary • Add Recipient to a Shared Itinerary • Pay for a reservation • Process travel credits • Pay for a reservation modification • Add payment details
Database	Functions: • Authenticate and Authorize • Search for a flight • Select a seat for a flight • Enter number and types of luggage • Select meal for a flight • Modify an existing reservation • Confirm reservation modifications • Select a flight • Reserve a flight • Sort and filter flight results • Review existing reservation details • Review all reservations • Cancel an existing reservation • Register a new user • View User Profile • Edit User Profile • Update User's Email Address • Update User's Credentials • Add a Travel Planner • List Travel Planner(s) • Remove a Travel Planner • Watch a given route • List watched routes • Remove a watched route • Share Itinerary • Add Recipient to a Shared Itinerary • Pay for a reservation • Process travel credits • Pay for a reservation modification • Add payment details

Table 5: CompFuncCoeff for each component-technology pair

CompFuncCoeff (c,t)		Component(c)					
		Authenticator	Payment Handler	Reservation Handler	User Handler	Search Handler	Social Handler
Technology(t)	Web Browser	3/3	3/3	9/9	7/7	6/6	2/2
	Web Server	3/3	3/3	9/9	7/7	6/6	2/2
	Database	3/3	3/3	9/9	7/7	6/6	2/2

Table 6: Maximum value of CompFuncCoeff for each technology

CompFuncCoeff (c,t)		Component(c)					
		Authenticator	Payment Handler	Reservation Handler	User Handler	Search Handler	Social Handler
Technology(t)	Web Browser	3/3	3/3	9/9	7/7	6/6	2/2
	Web Server	3/3	3/3	9/9	7/7	6/6	2/2
	Database	3/3	3/3	9/9	7/7	6/6	2/2

Table 7: CompFuncBoundaryError for each technology

CompFuncBoundaryError (t)		Degree to which t does not satisfy the best fit components	Degree to which t exceeds the best fit component	Total Error for t
Technology(t)	Web Browser	1 – 9/9	0	0
	Web Server	1 – 9/9	0	0
	Database	1 – 9/9	0	0

Table 8: Components and data in business blueprint

BB Component	BB Data in Component
Authenticator	Data: • User Credentials • User Email
Payment Handler	Data: • Credit Amount
Reservation Handler	Data: • User Reservation List • Trip Reservation Details

User Handler	Data: • User Reservation List • Trip Reservation Details • Planner Information • Payment Information
Search Handler	Data: • Watched Routes
Social Handler	Data: • Shared Reservation List • Reservation Shared Recipients

Table 9: Technologies and data provided by each technology

BB Component	BB Data in Component
Web Browser	Data: • User Credentials • User Email • Credit Amount • User Reservation List • Trip Reservation Details • User Reservation List • Trip Reservation Details • Planner Information • Payment Information • Watched Routes • Shared Reservation List • Reservation Shared Recipients
WebServer	Data: • User Credentials • User Email • Credit Amount • User Reservation List • Trip Reservation Details • User Reservation List • Trip Reservation Details • Planner Information • Payment Information • Watched Routes • Shared Reservation List • Reservation Shared Recipients

Database	Data: • User Credentials • User Email • Credit Amount • User Reservation List • Trip Reservation Details • User Reservation List • Trip Reservation Details • Planner Information • Payment Information • Watched Routes • Shared Reservation List • Reservation Shared Recipients

Table 10: CompDataCoeff for each component-technology pair

CompFuncCoeff (c,t)		**Component(c)**					
		Authenticator	Payment Handler	Reservation Handler	User Handler	Search Handler	Social Handler
Technology(t)	Web Browser	2/2	1/1	2/2	4/4	1/1	2/2
	Web Server	2/2	1/1	2/2	4/4	1/1	2/2
	Database	2/2	1/1	2/2	4/4	1/1	2/2

Notes:

- All of the above mentioned technologies are required to perform all BB functions.
- There is no direct association of the BB functions to these technologies but these technologies are required to execute the code written to perform BB functions.
- So these are the key technology to implement the system.

Part: 3

One possible means of aggregating into an overall quality assessment value is as follows:

- ✓ Determine thresholds for each metric, such that metric results can be normalized to a "metric scale" such as "good" = 3, "acceptable" = 2, and "poor" = 1. For example, maybe a transaction response time under 2 seconds is "good," while under 3 seconds is acceptable and anything greater than 3 seconds is unacceptable.
- ✓ If an objective has more than one metric, aggregate the "metric scale" values by weighting them according to the degree to which they accurately predict/assess whether the respective objective has been met.

- ✓ Similarly, aggregate the resulting objective values into sub-factor values by weighting based on the importance of the objective to the success of the system and the ease by which the objective can be achieved.
- ✓ Finally, aggregate sub-factors and qualities based on the priorities defined for the deployment blueprint

- UTILITY
    - Availability
        - Minimize Hardware failure time
            - *Objective:* Power outage at one server should redirect traffic to another available server in <3 sec (M, H)
                - *Metric:* Record response times for redirecting the traffic to different server while one server machine is switched off. (H)
            - *Objective:* Restart after disk failure in <7 min (M, M)
                - *Metric:* Record response times for redirecting the traffic to different server while one server machine is switched off. (M)
        - Maximize scalability
            - *Objective:* Demonstrate that web server farm/cluster will keep system available by performing function 'Search for a Flight' under progressively larger loads (M, M)
                - *Metric:* Record response times for 'Search for a Flight ' request while executing a test script from 100 machines with 10, 20, 30, 55, 70, 85, and 100 web client processes each. Conduct test with one web server and two web servers and compare response as load increases. (H)
    - Usability
        - Minimize user efforts
            - *Objective:* Perform function 'Search for a Flight' in scenario 'Plan a Trip' using <4 user clicks (H, M)
                - *Metric:* Record number of clicks from starting screen to perform ' Search for a Flight ' request. (H)
        - Minimize learning time
            - *Objective:* Ensure novice takes <5 clicks to perform function 'List Watched Routes' in scenario 'Add a Fare Watch Alert' (M, M)
                - *Metric:* Record number of clicks clicked by novice to perform 'List Watched Routes' request. (L)

- Performance
    - Minimize transaction response time
        - *Objective:* Perform function 'Search for a Flight' in scenario 'Plan a Trip' within 5 seconds under load of 500 concurrent requests (estimated peak based on 10K users searching for flights on Friday Morning) (H,M)
            - *Metric:* Record response times for ' Search for a Flight ' request while executing a test script from 40 machines with 50 web client processes each. (H)
    - Maximize scalability
        - *Objective:* Demonstrate that web server farm/cluster will scale by performing function 'Search for a Flight' under progressively larger loads (M, M)
            - *Metric:* Record response times for 'Search for a Flight' request while executing a test script from 20 machines with 1, 5, 10, 20, 30, 40, and 50 web client processes each. Conduct test with one web server and two web servers and compare response as load increases. (H)
- Security
    - Ensure password confidentiality
        - *Objective:* Perform function 'Authenticate and Authorize' using different username whose password is not known (H, H)
            - *Metric:* Try to use different passwords for the username whose password is unknown. Password should remain secret and user should not be able to login. (H)
    - Increase Privacy
        - *Objective:* Encrypt confidential data (H, M)
            - *Metric:* Call web service to get User Payment Information. The output should be in an encrypted format (M)
- Scalability
    - Maximize number of users
        - *Objective:* Demonstrate that web server farm/cluster will scale by performing function 'Search for a Flight' under progressively larger loads (M, M)
            - *Metric:* Record response times for 'Search for a Flight' request while executing a test script from 20 machines with 1, 5, 10, 20, 30, 40, and 50 web client processes each. Conduct test with one web server and two web servers and compare response as load increases. (H)

- Extensibility
    - Enable pluggable advising service
        - *Objective:* Minimize coupling between BB components (L, M)
            - *Metric:* Measure coupling using "Number of dependencies between components" (M)
        - *Objective:* Maximize inheritance from "Reservation" class (M, M)
            - *Metric:* Number of properties inherited from "Reservation" (M)
            - *Metric:* Number of methods inherited from "Reservation" (H)
- Data Backup and Recovery
    - Minimize data backup time
        - *Objective:* Perform 100GB data backup time in <1.5 hours (M, M)
            - *Metric:* Record time for backing up data of 100GB during application maintenance (M)
    - Minimize recovery time
        - *Objective:* Restore 100GB data to the database in <2 hours (M, M)
            - *Metric:* Record time for restoring data of 100GB (M)
- Maintainability
    - Minimize complexity
        - *Objective:* Minimize coupling between BB components (L, M)
            - *Metric:* Measure coupling using "Number of dependencies between components" (M)
- Project Schedule
    - Minimize time
        - *Objective:* Minimize project time (H, M)
            - *Metric:* Estimate time for each task and Keep track of the task time to compare with estimated time. (H)
- Project Cost
    - Minimize cost
        - *Objective:* Minimize cost of development (H, M)
            - *Metric:* Determine the initial hardware investment. (H)
            - *Metric:* Determine the initial investment in off-the- shelf software, including the database and web server. (H)
            - *Metric:* Estimate the number of person-hours required to code in-house modules. (L)

- *Objective:* Minimize cost of maintenance (L, M)
    - *Metric:* Estimate the number of person-hours required to define a new feature and bug fixes. (L)

- *Objective:* Minimize cost of operation (H, H)
    - *Metric:* Estimate the number of person-hours required to set up the system initially (One time cost). (H)
    - *Metric:* Estimate the maintenance fees on off-the- shelf software. (M)