Effective Testing Principles for the Mobile Data Services Applications

Satish Srirama[1], Rajasekhar Kakumani[2], Akshai Aggarwal[3], Pravin Pawar[4]

[1]*Department of Computer Science, Informatik V, RWTH Aachen University, Germany.*
srirama@i5.informatik.rwth-aachen.de

[2]*Department of Electrical and Computer Engineering, Concordia University, Quebec, Canada.*
krs175@yahoo.com

[3]*Department of Computer Science, University of Windsor, Ontario, Canada.*
akshaia@ uwindsor.ca

[4]*Department of Computer Science, University of Twente, The Netherlands.*
p.pawar@ewi.utwente.nl

***Abstract*— Wireless communication technologies like GPRS, UMTS and WLAN, combined with the availability of high-end, affordable mobile devices enable the development of advanced and innovative mobile services. Devices such as mobile phones and Personal Digital Assistants let the users access a wide range of new offerings whenever and wherever they happen to be. A strategic approach for the quality assurance of these mobile data services should take into account a number of characteristics unique to the mobile paradigm such as the increased complexity of emerging handheld devices, the greater sensitivity to security and load related problems in wireless infrastructure and increased complexities of scale. This paper identifies the major factors influencing the development and testing strategies for these applications and works out effective quality assurance principles to ensure productive and scalable mobile data services.**

***Index Terms*—Quality assurance, Software testing, mobile data services**

I. Introduction

With the proliferation of advanced wireless networks and devices, mobile operators and enterprises can now provide a variety of types of data services to their users. Today's mobile communication platforms such as GSM, GPRS, EDGE, 3GSM and CDMA 1xRTT offer 'always-on', higher capacity, Internet based content and packet based data services. The array of mobile data services encompasses a wide range of applications such as color Internet browsing, e-mail on the move, powerful visual communications, multimedia messaging (MMS), M-Commerce applications, Mobile Office applications like calendar and location based services. These advanced wireless applications are part of a complex structure that spans wireless devices, wireless networks, the Internet and back-end systems that typically reside on enterprise platforms [1]. This paper presents the strategies for effectively assuring the quality of mobile data services. The strategies can be used by the application developers, quality assurance professionals and the mobile service providers to help them create productive and scalable mobile data services and to provide to the service subscribers a branded experience that will provide maximum return, reduce churn, and increase average revenue per user. This paper is organized as follows:

Section 2 of the paper explains the data services model under consideration and provides taxonomy of these applications. Section 3 identifies various factors which are unique to the mobile data services paradigm and which may have to be considered for development of testing and quality assurance strategies. The testing guidelines for the mobile device clients of these applications are given in the Section 4. Section 5 focuses on the scaling issue and provides guidelines for the back-end system performance testing. Section 6 concludes the paper and discusses the future work.

II. Data Services Architecture

A typical networked wireless data service is comprised of the following components:

- A network aware application residing on a wireless device (the client)
- The wireless network and the Internet, and corresponding communication protocols [1]
- The back-end system (generally, at the service provider's site) consisting of the load balancer, web servers, application servers, content servers, other servers such as *LDAP* servers storing client application, and databases. It may also include the server replicas with mesh interconnections to achieve fault tolerance, scalability and better performance. A back-end system

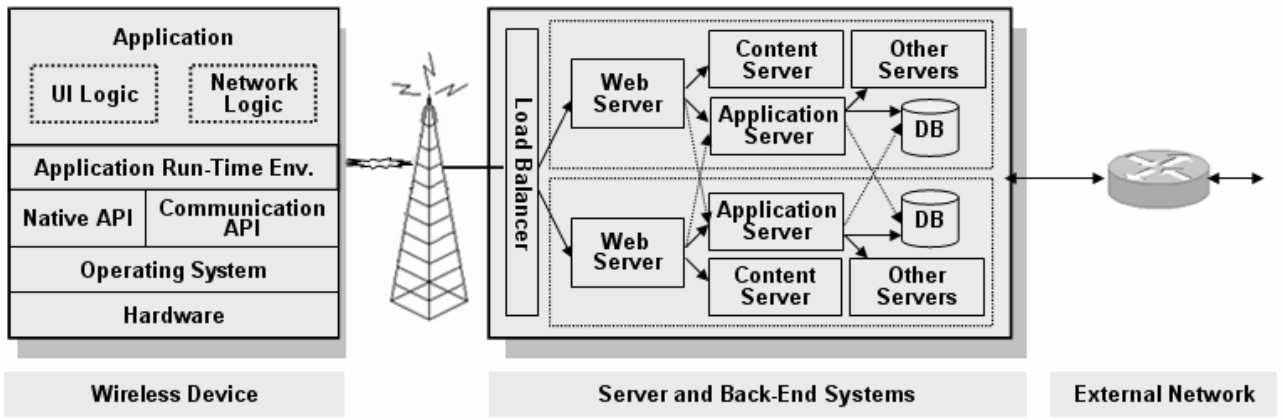

Fig. 1. The Mobile Data Services Model

may have a number of tiers in the client's n-tier architecture [2].

- A gateway to the external network

A mobile host initiates a request for the data services application. On this request, the client application is downloaded to the mobile device. This application executes inside the application runtime environment such as *KVM* (for *J2ME*) or *BREW* depending on the implementation. The client application and the server communicate using *HTTP*. A sample scenario depicting the request for a data service application and its execution is shown in [3]. Herewith, we present taxonomy of the mobile data service applications. Note that a particular application may fall in more than one category described below.

Client only applications: For execution, the client only applications do not have any server side counterpart in the fixed network. However, the application needs to be downloaded from the fixed network. An example of such an application is *taskscheduler* application which lets the user store various appointments. The application later reminds the user of the appointments at the scheduled times.

Content applications: The content applications, on execution, fetch the content requested by the mobile user from the back-end system. The content is regularly updated in the back-end system. These applications include the entertainment and information applications like *news, astrology, and songs*.

Client-Server applications: In these applications, the client sends a request to the server. The server processes this request and sends the response back to the client. *Auctions on the mobile, live games information, and location based services* are some of the examples of this type of applications.

Client-External Server applications: In such applications, the service provider's back-end system routes the request from the client to the external application server via the local application servers. Certain translation of the data in the request may be required, so that external servers can identify the request. Apart from *e-mail* and other office applications, this category also includes *m-commerce* applications such as *mobile banking*.

Applications Using Native API: Some applications invoke the native applications for various purposes. For example, a *video* application opens the native player for playing the video. The *appointment* application sets the alarm on the phone by using the corresponding API. The *meeting* application uses the native API to fetch a local list of friends, which belong to a particular group.

The categorization of the applications presented in this Section is important, as every category may require different testing strategies.

III. Factors Influencing The mobile Data Services paradigm

The development and testing strategies for the mobile data services are influenced by various factors, many of which are different from those in the traditional applications. An application may be developed on one platform and deployed on ones which are vastly different. Every application has to cater to a wide range of devices that implement different versions of the application runtime environment, and may use different proprietary extension APIs [4]. The quality assurance team must be aware of the specific methods, which developers use to take into account the effect of the following factors:

Increased complexity in handheld devices: The handheld computing platform for the mobile data services is not just a firmware, but consists of an operating system, native APIs as well as the application runtime layer. New devices, which support multiple mobile networks and communication technologies, are emerging. This requires that the applications should support multiple interface and rendering standards. Another challenge for the developers and testers of the data services applications is to conform their application to multiple, older version of mobile device software [5].

New security concerns: With the ability to download applications and to execute the code on the device, comes the corresponding risk associated with the compromise of user authentication, enterprise data and transactional security. Viruses and other malicious code can also cause problems. Information written by the data services application may be hacked by unauthorized persons and is susceptible to misuse [6].

Resource poorness of the mobile devices: The mobile handsets have limited processing power, limited memory, improper user interface with a limited set of features, and finite energy source. This severely restricts the size of the application, memory use, processing power, and the number of applications running simultaneously on a mobile device [7].

Inherent limitations of the wireless medium: Wireless connection is a decisive factor in contributing to mobility. However, bandwidth limitations impose several restrictions on the volume of data that can be transferred. The available bandwidth differs depending on the location of the mobile host. Moreover, the network operators prioritize the connection for the voice call as compared to the data call [8]. These factors contribute to degradation of the user experience.

Increased complexities of scale: The mobile data services are enjoying rapid growth in the number of service subscriptions and popularity. Data enabled handhelds are becoming pervasive worldwide, as content providers or enterprises can be reached from many parts of the globe. Performance is one of the critical factors to the success of data services applications. It is required to regularly monitor the performance of the back-end system to check the impact of deployment of new applications on the existing applications [9]. The back-end system must be able to provide reliable data services even during the peak usage and emergency situations without affecting adversely the throughput and latency. Hence, the effective back-end system testing strategy should consider the issue of providing scalable data services.

The testing activities for the data services applications can be classified as the End-User console application testing and back-end systems testing. The former deals with testing the client part of data service applications. In the later, we primarily concentrate on the performance testing aspects arising due to the economy of scale in mobile data services.

IV. Testing Principles For The Console Applications

Software testing is an iterative process and should start from the very beginning of the project. The application developers need to get used to the idea of designing software with testing in mind. When creating applications, developers must consider that the speed of the application should not compromise the use and purpose of the application. This issue must be considered in the very early phases of application design. Depending on the type of application, developers must also remember that end users may use other applications concurrently. Therefore, excessive consumption of the device's processing power and memory should be avoided. Such tests are well supported by the unit testing tools such as *J2MEUnit* and *BREWTestUnit*.

Once the client application is developed, it should be tested for the resource utilization and performance on the mobile device. This can be achieved by inserting the test code in the source and observing the logged results. The use of embedded software testing tools such as *Rational Test RealTime* [10] is quite advantageous. It provides the functionality to profile memory and performance, to analyze code coverage, to obtain runtime tracing, and to analyze the behavior of program execution on the handset. One among its very useful features provides detailed test and runtime analysis reports which are hyperlinked to the relevant source code.

Part of the client application validation process can be executed on the customized emulators which provide the input mechanisms and screen appearance identical to that of the target handset. Designing new emulators for the handsets and integrating them with the development and testing toolkits is relatively an easy task. For the mobile data services applications, the testers must pay more attention to usability and form factors than is the case with traditional desktop applications. General usability testing guidelines for the mobile applications are well elaborated in [1, 2, 6, 11]. Apart from these, the applications should be tested for the use of native APIs. For example, an *Office Assistant* application fetches your schedule from your *Office Outlook* account and stores the meeting information locally. In this case, mobile devices have different constraints on the number of characters to be accepted for the local *scheduler* component. Similarly, the differences in the native implementations of the features such as *address book*, *SMS* or *MMS storage*, and *picture formats* should be considered for the applications like *greetings and Address Book Transfer*, which share these features across multiple handsets.

The application must also be tested for appropriate security mechanisms. If the nature of an application requires saving sensitive data, the data must be encrypted and hidden to retain confidentiality. In addition, the user must be informed clearly that sensitive data is being stored into the device's memory. The application should be checked for the use of secure communication if sensitive information is to be transmitted between two end points or if a user may be charged for the result or consequences of a transaction. Encryption is also needed if sensitive data is transmitted via *Bluetooth* or *Infrared* [6].

One among the most ignored issues by the developers of the mobile data services application is that of error conditions, exceptions handling and proper reporting to the user. It is usual to see the exceptions displayed on the screen for commercially available mobile data services applications. The comprehensive testing strategy should have test cases for handling all the possible error conditions. If the application exceeds the persistent memory quota allotted or if it does not have enough space to store the data, the user should be notified accordingly. In case, an application is updated, it should be tested for the compatibility of the existing data with the newer version. The tests should also include call and SMS interrupt tests, system notification interrupts and user's unnecessary key press events to check that the application pauses and resumes its state once the interrupts are processed [11]. For the *client-server applications*, it is necessary to ensure that the connection timeouts are properly handled, and

the user is notified about them. The network connection should be closed when it is not needed anymore or when the application is closing.

For the client-server applications, creating network logic requires that the developers to be familiar with how HTTP works, which services are available and their location, and how to encode requests and decode responses [1]. The communication between the client and server needs to be verified to assure that no unnecessary data is sent and encoding of the requests and responses is optimal. Not every user input should be sent to the server. Suitable compression techniques should be explored. Mobile clients benefit from compression when the available bandwidth is scarce. But even when resource-constrained devices have better connectivity, the performance loss caused by decompression is almost negligible [12].

A client application should do some preliminary validation checks on the input data prior to sending the request to the back-end system. The back-end system should also check for the error conditions and propagate proper error messages back to the client. For example, if the external bank server is down, the application server of the back-end system should send to the user, a response with the message *'The bank server is down. Please, try after some time.'* instead of the *'connection timeout'* message.

V. Back-End System Performance Testing

The mobile data services are among the most popular mobile applications. For the key mobile services providers, the data services environment consists of around a thousand services with a few million users resulting in more than eighty million hits in a week [3]. However, there are numerous non-technical challenges in ensuring the quality of performance of an application. Often performance testing does not have the same priority as feature delivery and it is often ignored when delivery schedules reach the specified limits. However, ignoring performance testing inevitably leads to disaster [13]. To effectively carry out performance analysis of the back-end systems, two broad approaches of *load testing* and *performance modeling* should be considered.

In the first approach, load testing tools are used to generate artificial workloads on the system so that the performance can be measured under load conditions. Sophisticated load testing tools can emulate hundreds of thousands of *virtual users* that mimic real users interacting with the system. The *performance metrics* such as response time, latency, utilization and throughput measured during the test run can be used to identify and isolate system bottlenecks, to fine-tune application components and to predict the end-to-end system scalability [14]. The following must be considered while creating the successful load tests:

Identifying demand forecast: Considering the data service lifetime, the peak and average usage of the service needs to be estimated. A few applications such as *MMS, ringtones, videos* are among the most popular. The seasonal applications such as *New Year Greetings* receive maximum hits within the span of a few days. The applications providing live summary of a game are expected to receive tremendous hits within the span of a few hours! The behavioral patterns of the service users over a period of days, weeks or months as well as based on sex, region and other attributes need to be considered for identifying demand forecasts. The data mining techniques such as associative rule mining have been successfully used for such purposes [3].

Mimic realistic usage scenarios: The scripts for the load testing of a mobile data service should mimic realistic user distribution. For instance, the scripts for mobile banking should simulate more percentage of the transactions for *the balance checking* activity compared to those for the *funds transfer*. Similarly, each of the *email service* testing scripts should start with the login activity. Depending on the usage pattern, further actions of the script should simulate *email reading* or *writing* activities. Special attention needs to be paid to maintaining the *virtual user sessions* during the script execution.

Elimination of the bottlenecks in the testing process: It must be ensured that the components other than the *System under Test* like links, switches, authentication servers and the testing tool should not become a bottleneck in performance measurements [9].

Performance of the external servers: For the client-external server applications, the external server where the requests terminate is a potential bottleneck.

The results obtained from the load and stress testing prove to be of great use in suggesting the design changes to alleviate the bottlenecks. Interesting back-end system performance testing case studies identifying fatal memory leak, disk I/O bottleneck, scalability bottleneck due to semaphore contention, and database I/O bottleneck due to poor use of caching are presented in [13].

In the *performance modeling* approach, performance models are built and then used to analyze the performance and scalability characteristics of the *System under Study*. These models can be grouped into two categories: *simulation models* and *analytical models*. Simulation models are software programs that mimic the behavior of a system as requests arrive and get processed by various resources. Analytical models are based on mathematical laws and computational algorithms are used to generate performance metrics from model parameters [14]. Analytical performance models frequently employ *Markov Chains* models, specified by using *Queuing Networks* and *Stochastic PetriNets*. The applications of performance modeling for large scale *J2EE* applications and *EJBs* are elaborated in [14, 15]. *J2EE* and *EJBs* are frequently used implementations for large scale mobile data services back-end systems.

The work done in [14] studies a real-world J2EE application of a realistic complexity and shows how to exploit analytical performance models in order to address the problems in the capacity planning. In this work, an *Infinite*

Server queue is used to model the client machine. The queue emulates virtual clients sending requests to the system. *Processor Sharing* queues model the CPUs of the *WebLogic* servers and two CPUs of the *database* server. A *First-Come-First-Served* queue is used to model the disk subsystem of the database server. A queuing network model generated by following this procedure can be fed to the performance evaluation and prediction systems such as PEPSY-QNS [16]. Such systems support various performance measurement methods and calculate performance measures including the throughput, utilization, average service time and average response time for every job class.

The work reported in [15] considers the performance prediction for an *EJB* system based on the modular structure of an application server and the application components. It describes a framework for constructing the layered queuing models and for their inclusion in the server. The queuing models are structured around the software components, based on the templates for *EJBs*, and for their inclusion in the server.

The performance modeling exercises elaborated in [14, 15] should be useful for large back-end systems as shown in Fig. 1. For smaller implementations, the experimental results obtained in the modeling of Apache Web Server [17] should prove helpful. The work done in [17] describes a model of the *Apache* web server, which consists of a processor sharing node with a queue attached to it. The total number of jobs in the system is limited. The arrival process to the server is assumed to be a two-state *Markov Modulated Poisson Process* which represents bursty arrival traffic.

VI. Future Work And Conclusion

This paper has presented the guidelines for testing key aspects of the mobile data services. These aspects determine the behavior of an application on a mobile device and are to be considered for studying the performance issues of back-end systems for large scale mobile systems. We have listed the main factors influencing the testing strategies for mobile applications. We have also presented some important issues, which are likely to be overlooked while designing the testing methodologies. The guidelines, described in this paper, are designed to help the developers and mobile data service providers in ensuring the bug free and scalable applications.

The future work consists of developing the performance models and metrics for the m-health monitoring services [18]. The m-Health monitoring service gathers patient's vital signs collected from the medical sensors attached to the patient's body and delivers this data in a near real time fashion to the healthcare professionals who access this data from the back-end systems. One of the interesting features of this service is that the mobile device acts as a data producer and entities in the fixed network act as the consumer [8]. This inversion of the role is an interesting aspect in modeling the performance of an m-Health monitoring service.

References

[1] SUN White Paper, "The Complexity of Developing Mobile Networked Data Services", whitepapers.zdnet.co.uk, June 2003.

[2] AppLabs White Paper, "AppLabs Wireless Application Testing Whitepaper", AppLabs Technologies, Philadelphia, PA 19103.

[3] P. Pawar, A. K. Aggarwal, "Associative Rule Mining of Mobile Data Services Usage for Preference Analysis, Personalization & Promotion", in *proc. of WSEAS International Conference on Simulation, Modeling and Optimization*, Izmir, Turkey, Sept. 2004.

[4] Q. H. Mahnoud, "Testing Wireless Java Applications", Sun Developer Network, Nov. 2002.

[5] M. Cundy, "Testing Mobile Applications is Different from Testing Traditional Applications", Veritest Tester's Network, Aug. 2001.

[6] "Developer Platforms: Guidelines For Testing J2ME Applications", version 1.2, Forum Nokia, Nov. 2004.

[7] G. Foreman, J. Zahorjan, "The Challenges of Mobile Computing," IEEE Computer, April 1994, pp. 38-47.

[8] N. Dokovsky, A. V. Halteren, I. Widya, "BANip: enabling remote healthcare monitoring with Body Area Networks", *International Workshop on Scientific Engineering of Distributed Java Applications*, Luxemburg, Nov. 2003.

[9] R. Udupa, P. Adiga, "Performance issues and Evaluation techniques for Networking Devices", *3rd Annual International Testing Conference*, India, 2001.

[10] J. Campbell, "Memory profiling for C/C++ with IBM Rational Test RealTime and IBM Rational PurifyPlus RealTime", IBM Developer Works, Apr. 2004.

[11] "Application Developer's TRUE BREW Test Guide: Requirements and Test Cases", Quallcomm Incorporated, Oct. 2001.

[12] M. Tian, T. Voigt, T. Naumowicz, H. Ritter, and J. Schiller, "Performance Considerations for Mobile Web Services", Elsevier Computer Communications Journal, Volume 27, Issue 11, July 2004, pp. 1097-1105.

[13] A. Avritzer, R. Farel et. al., "Performance Analysis in the Age of the Internet: A New Paradigm for a New Era", www.cse.iitb.ac.in/~varsha/allpapers/itc17sig.pdf, (2002?)

[14] S. Kounev, A. Buchmann, "Performance Modeling and Evaluation of Large-Scale J2EE Applications", *29th International Conference of the Computer Measurement Group on Resource Management and Performance Evaluation of Enterprise Computing Systems*, Dallas, Texas, Dec. 2003.

[15] J. Xu, A. Oufimtsev, M. Woodside, L. Murphy, "Performance Modeling and Prediction of Enterprise JavaBeans with Layered Queuing Network Templates", *The fourth workshop on specification and verification of component-based systems (to be held in)*, Lisbon, Portugal, Sept. 2005.

[16] G. Bolch, M. Kirschnick, "The Performance Evaluation and Prediction System for Queueing NetworkS - PEPSY-QNS", Technical Report TR-I4-94-18, University of Erlangen-Nuremberg, Germany, June 1994.

[17] M. Andersson, J. Cao, M. Kihl, C. Nyberg, "Performance Modeling of an Apache Web Server with Bursty Arrival Traffic", *in Proc. of International Conference on Internet Computing*, Las Vegas, Nevada, USA, June 2003.

[18] D. Konstantas, R. Bults, R. Herzog, "MobiHealth: Innovative 2.5/3G Mobile Services and Applications for Healthcare", *11th IST Mobile and Wireless Telecommunications Summit*, Thessaloniki , Greece, June 2002.