UTILIZING SAP’S SYBASE UNWIRED PLATFORM AND MOBILE BUSINESS OBJECTS TO CREATE A MOBILE PLANT MAINTENANCE APPLICATION.

by

ALLEN CARROLL

Advisor
DR. HALA ELAARAG

A senior research paper submitted in partial fulfillment of the requirements for the degree of Bachelor of Arts in the Department of Mathematics and Computer Science in the College of Arts and Science at Stetson University DeLand, Florida

Spring Term
2013
ACKNOWLEDGMENTS

This was made possible due to the software technology provided by System Analysis and Program Development and the server access they so graciously provided to both myself and my advisor. I would also like to personally thank my advisor, Dr. Hala ElAarag, for all the assistance and guidance that she has provided me for the entirety of my time involved in this project. I would also like to thank Dr. Salah Sharieh, Dr. ElAarag’s SAP contact, for all the assistance he has provided, without which I could not have gone as far as I did.
# TABLE OF CONTENTS

ACKNOWLEDGMENTS ........................................................................................................ i

TABLE OF CONTENTS .................................................................................................. ii

LIST OF FIGURES ........................................................................................................ iii

ABSTRACT ...................................................................................................................... 1

1. INTRODUCTION ........................................................................................................ 2

2. Related Work ............................................................................................................... 4

3. Implementation .......................................................................................................... 17

4. Results ...................................................................................................................... 21

5. Conclusion ................................................................................................................ 24

REFERENCES ............................................................................................................... 25
LIST OF FIGURES

Figure 1: The SAP HANA database architecture.................................................................6
Figure 2: Memory alignment and resulting access patterns for row and column stores.........7
Figure 3: Accuracy of the runtime estimation........................................................................... 8
Figure 4: Example of SOA application using Sybase Workspace........................................ 9
Figure 5: Example of a service order application using MBO’s in SUP ............................. 11
Figure 6: Sybase Unwired Platform – Data flow/security.................................................... 12
Figure 7: SAP Afaria – Interface.............................................................................................. 15
Figure 8: Sybase Control Center – Interface............................................................................ 16
Figure 9: SAP GUI - Plant Maintenance Service Order......................................................... 18
Figure 10: SUP – Workflow Package: Layout.......................................................................... 19
Figure 11: SUP – Workflow Package: Get Order.................................................................... 20
Figure 12: SUP - Workflow Package: Editbox.......................................................................... 21
Figure 13: Hybrid-web-container – Demo Flights Application.............................................. 22
Figure 14: SUP – Query Result Checker and our Demo Mobile Application....................... 23
ABSTRACT

The purpose of this project is to use System Analysis and Program Development’s (SAP) Sybase Unwired Platform (SUP) to create a custom web-centric application for the explicit purpose of plant maintenance. Using database information obtained from Stetson’s proprietary access to a SAP server named ‘iVedha’, we have created a mobile application for plant maintenance that can be accessed across a variety of devices. Being a web-centric application, this program is also accessible from other web-enabled devices, and while the exact layout on each web-centric device may differ slightly, they all follow a general pre-structured theme that is relatively uniform across each platform.
1. INTRODUCTION

With the ever growing need and usage of mobile devices in today’s society, the mobile development market has seen a subsequent rise in recent years. Sybase, taking notice of this growing trend, began pursuing development towards implementing a set of tools back in 2005 that were intended for development usage in the mobile market. Some may have thought that this was a sign Sybase was moving away from its already well-established stance in database implementation and design, however as CEO of Sybase John Chen stated [FON12], this was actually a sign of the opposite happening. For Sybase, this was a golden opportunity to not only develop for the mobile market, but for them to bring their own level of database expertise to the mobile market in general.

The Sybase Unwired Platform (SUP) is essentially a series of tools that were created specifically with mobile platform design in mind. The general consensus is that if you have the proper tools that are optimized for the explicit task of mobile platform development, then one can create an optimized application for said device. With tools such as these, creating an Enterprise Resource Planning (ERP) system with a mobile interface is made surprisingly easy when compared to using more basic tools. [TAN07] The core of Sybase’s Unwired Platform’s tool is structured around the open-source Eclipse coding program in such a way that the environment is highly immersive to the point where the need of manual coding is drastically reduced while still maintaining a sense of efficiency.

System Analysis and Program Development (SAP), a software company well known for
its foothold in the information systems of businesses world-wide bought out Sybase in late 2010 for $5.8 billion USD. [WOR10] The deal struck between these two conglomerates was mutually beneficial for both parties in the long run. As a result, each company had viable assets that each of them could mutually benefit from. Sybase was able to gain access to SAP’s own database technology as well as their already well-established large business information system market. [SYB10a] SAP would also gain full access to Sybase’s SUP tools and their expertise in mobile platform development, providing them with a golden opportunity to gain a foothold in the mobile platform development market. [WOR10] With this buyout, SAP would have everything they needed internally to not only place a foothold in the mobile development market, but to add their own database and business structure into the market itself.

In late 2012 [BOR12], SAP executive Sanjay Poonen announced to the world that SAP is planning on getting heavily involved in the mobile application development market, with plans of bringing their software to at least a billion people by 2015. This is a bold statement, but SAP plans to increase development of their products with mobility in mind. Already possessing a mobile application store for Blackberry, Android, and iOS devices, SAP has nearly all the resources they need ‘in-house’ to achieve their goal of putting themselves in the mobile market. [BOR12] Whether or not they will achieve the success they are striving for remains to be seen, although SAP has already seen success in applications such as Recalls Plus [LEV12], on Apple’s App Store, an application for the purpose of alerting parents on up-to-date information pertaining to recalls and safety information.
2. Related Work

2.1 SAP ERP

One of SAP’s most notable developments in the business world is their development of their own Enterprise Resource Planning (ERP) systems. ERP systems are by definition “any software system designed to support and automate the business processes of medium and large businesses.” [VOG02] In the case of SAP, this information system conglomerate makes explicit usage of their database technology to provide a sturdy backbone for their ERP application systems through which business data can be efficiently and reliably transferred both internally and externally in a business. [BÖG11]

One important thing to note about SAP’s ERP systems is that its ultimate performance is largely dependent on the performance of two different pieces of technology, an application server and a database management system. [BÖG11] SAP uses their own application server in their ERP systems, but they do offer the option for the use of several different database management systems. This provides a certain layer of flexibility and further demonstrates the amount of customization that SAP’s ERP offers to its target users.

While SAP’s ERPs are gaining, and have gained, ground in the economic market, it should be noted that just because such a system is implemented in a business’s infrastructure does not mean that said business will gain any improved benefits as a result. If fact, the opposite results can be expected when the system is either improperly implemented or the organization that will be hosting and/or supporting it is either improperly trained or ill-prepared [WIL11]. As with any other information system or technological change that occurs within a company or
organization, successful integration is key, not just with the hardware and software, but with the end users as well. With this in mind, SAP ERP users must make sure that the system is properly integrated in a manner that is both efficient and reasonable [VOG02]. SAP ERP’s can be tricky [WIL11] to use, as depending on the level of authority the intended users will have over the actual database the system runs on, mistakes both trivial and serious may be hard to correct, meaning more effort wasted and profit lost. Luckily, there are alternatives in preparing for SAP ERP integration. Several companies, such as WileyPLUS [WIL11] offer virtualized environments that can replicate the SAP ERP environment for training purposes. What makes this an invaluable tool is that unlike an actual SAP ERP system, resetting database information and application data is relatively easy. It is not an alternative for the actual ERP, but it is a valuable tool for training purposes. One such simulator provided by WileyPLUS is a Java-based [WIL11] platform that is accessed via a Java-enabled web browser. Because of this, it requires no special installation features, except for installing and enabling Java on the machine intended to run the simulator. This also has the benefit of being near-hardware independent, allowing for cross-platform usage.

2.2 SAP HANA DBMS

Of the technology that SAP has pioneered in, database management is one of their most well-known achievements. Most of today’s SAP’s ERP systems run with the intention of using “in-memory” [BOE12] data storage. What this means is that while the standard practice of storing and recalling database information, such as SQL tables for one example, from a physical hard disk drive to memory is still a viable option, SAP does things differently through means of storing the information directly in the active memory rather than to just a physical storage
device. By doing this, SAP takes advantage of the huge latency decrease that comes from not having to read and write to physical media, which is useful as it is generally assumed that most information stored on a database is intended to be kept available on demand in its most up-to-date status. One such database that makes use of this principle that SAP commercially uses is the SAP HANA database management system (DBMS).

![Figure 1: The SAP HANA database architecture](FÄR12)

SAP’s HANA DBMS is unique among the DBMS’s that exist on the market in part due to its significant flexibility that it offers. However, one of its notable features is how it makes efficient use of the hardware [FÄR12] that it runs on, it is able to take advantage of the server’s drive type, number of cores, memory capacity, and memory locations. The DMBS is designed from the bottom up to be geared for heavy business usage and was thus designed with the ability for smooth integration with applications designed by SAP and is capable of using the same application’s servers to allow for shared-data communication [FÄR12] between the DBMS and the application in question. This DBMS is equipped with the following three different types of
query engines, as seen in Figure 1, to meet current and future needs in a business environment that is still in the process of evolving: an SQL-based engine, a text search engine, and a graph engine. The SQL-based [BOE12] engine allows for easy SQL access to structured data and allows for easy transfer of data in and out of the DMBS. The text search engine is standard and is meant to exist as a flexible, generic way to sift through structured, semi-structured, and unstructured data as pure text. The graph engine is a set of algorithms that allow for graphical data to be comprised from business information, an expected feature from software designed for business functions.

2.3 OLAP and OLTP

![Figure 2: Memory alignment and resulting access patterns for row and column stores [RÖS12]](image)

Online Analytical Processing (OLAP) and Online Transactional Processing (OLTP) are two important types of activities that are used in databases that have been structured around a column or row format [RÖS12]. OLAP is generally considered to be geared towards the storage of data in column format. The reasoning behind this is that column-based storage is typically the primary choice for storing data that is not only going to be recalled on the fly [SIK12], but is also not expected to be changed or updated often. As a result, OLAP is optimized for fast recollection of data and is not the most efficient choice for making inserts within databases. On the other hand, OLTP is geared towards row-centric storage and is made with the understanding that data
accessed through this means is going to be updated on a normal basis and thus, while it may not be as fast at recalling data on the fly, it is very accurate. Figure 2 shows how the same set of data can be stored in either row or column format for use with each process.

![Figure 2: Example of data storage](image)

**Figure 3: Accuracy of the runtime estimation** [RÖS12]

OLAP and OLTP are both processes that have had an impact on how future databases are formed. SAP’s HANA DBMS takes advantage of both of these processes with its hybrid-database [RÖS12] structure. HANA is able to partition its own internal memory and store sections of its own data in either a row-based or column-based structure. By doing so, each section of data can be optimized based on its intended usage. Figure 3 illustrates the end results of a test run to determine the effectiveness of an ‘advisor’ application that can be used in conjunction with DBMS’ such as SAP’s HANA DBMS. The figures shown in the data illustrate how effective row-centric or column-centric storage can be for certain categories of data, and that having the correct storage type can greatly affect the time expended overall. For data that isn’t expected to change often, like a list of street addresses, it can be stored in column format,
while data such as a list of momentary exchange rates can be stored in a row format, allowing for levels of optimization that are not possible in a database that can only handle one structure.

2.4 SOA

Service Oriented Architectures (SOA) are by definition [TAN07] systems of frameworks, patterns, and principles of development and design for the purpose of creating applications that are based on different services and designs in a manner that allows for interconnectivity between similar applications. The principle behind this idea is that the functions of the application are exposed to other applications and the users as services rather than just abstract functions. This is very useful for web application development as the services created by the application can be designed to be accessed by multiple clients and because they are designed to be loosely connected to other software and services, the design of the services and sometimes the services themselves can be reused as templates in other projects and applications. Figure 4 illustrates a generic result of what the final result of such loose interconnectivity of differing applications and functions can be if implemented properly.

![Figure 4: Example of SOA application using Sybase Workspace][1]
Application Programming Interfaces (API’s) are a prime example of the capabilities that SOA has to offer in terms of loose compatibility. API’s are the interfaces [BEA08] that are used to bridge the gap between the programmer and the language library used, and it makes sense that having the ability to recycle and reuse structures and designs in the productions of an API would be of benefit in future projects. Enterprise Service-Oriented Architecture (E-SOA) makes use of this in its API design. E-SOA is quite an adaptable platform that is used and maintained by SAP and, as a result, is quite adept at interfacing and coupling with SAP’s own backend databases. As of 2008, there are at least 1,700 different web services provided by SAP’s E-SOA alone that are at disposal to developers for the creation of their own applications. This, combined with the fact that SUP provide integrated tools [TAN07] that support the design of SOA applications, makes SOA a powerful structural design pattern to use.

2.5 Sybase Unwired Platform/Server

Sybase’s Unwired Platform (SUP) is a unique piece of Sybase’s mobile development toolset in that it allows for reliable and secure mobile application deployment over multiple platforms. Essentially, the SUP [SYB11b] is a modified version of the popular coding tool Eclipse. It exists both as a standalone version and as a plug-in add-on for pre-existing Eclipse editions. SUP is however more than just a simple add-on to an already working system. SUP allows for a more mobile-centric development process to be easily adopted as it allows for multiple mobile application development features to be combined into one easy-to-navigate environment, allowing for a more efficient design procedure to take place.

SUP’s main feature is its ability to allow it’s users to set up secure connections to a variety of back-end data resources, such as SAP ERP servers for instance. It also allows for
users, once a secure connection has been established to the back-end server, to utilize the back-end data objects in the form of special objects called Mobile Business Objects (MBO). MBOs [SYB11b] can be thought of as containers on the developer’s work environment that represent and dictate how certain back-end objects behave, both in structure and data flow. MBOs are vital to the SUP mobile developer because of how versatile these features are, allowing for developers to easily create representations of back-end data objects and as a result allow developers the ability to create relationships between one or more of these MBOs. These MBOs themselves, being representative containers, are considered the backbone of every application, regardless of the target platform that may be developed on.

![Diagram of Mobile Application Diagram](image)

**Figure 5: Example of a service order application using MBO’s in SUP**

A key feature of SUP is its Unwired Server, which is essential to virtually all mobile applications developed with web-centric usage in mind. The Unwired Server is located inside the inner firewall of the machine that is to be the host and is where all applications that are created with
the SUP are deployed to. From within the firewall, the Unwired Server connects to a designated relay server within the DMZ region between the inner and outer firewall as seen in Figure 6, which in then facilitates a connection to the device or devices that are attempting to make a secure connection to the Unwired Server.

![Sybase Unwired Platform – Data flow/security](SYB11a)

The data that passes between each individual device and the Unwired Server goes through a series of security features. Depending on the device itself, the first stage of security lies within the mobile devices’ built-in security features, usually in the form of user authentication like device passcodes. Another device security can be found in the device management software known as Afaria [SOL05], a piece of software provided by SAP which allows for remote
permissions and or data control of devices, giving the privileged user or admin the ability to potentially wipe the device in question remotely if it ever becomes compromised. The application itself also has some built-in security measures, which vary depending on the type of application used. Replication-Based Synchronization (RBS) [SYB11a] applications use either UltraLite or UltraLiteJ as the primary databases depending on the type of native language the device runs on. Messaging-Based Synchronization (MBS) applications use either SQLite or UltraLight depending on the device’s native language. Workflow container applications have the most variety in terms of storage and security as it varies greatly depending on the types of devices used. For Workflow containers, a Windows mobile device the application makes use of the system and file registry while iOS devices make use of a SQLite database to store and secure information on the device itself. The local database that each application uses, with the exception of Windows devices, is encrypted to prevent unauthorized access to and/or tampering with data within the device.

The relay server [SYB11a] that the data to and from the device flows through can be thought of as a means of facilitating and authenticating proper connections going to and from both firewalls, meaning the user should never have to manually open up any new ports for connections between the SUP server and the device it is connecting to. Data that passes from the device to the designated relay server between the inner and outer firewalls of the system that is hosting the particular SUP server is encrypted in both directions and the relay server itself has no way of knowing the contents of what information is being passed and cannot modify data that flows through it.

The Unwired Server itself makes use of a role-based access control (RBAC) [SYB11a] system to control and protect incoming and outgoing information. The server itself does not have
its own security system integrated within it, but rather makes use of the preexisting security features on the host system itself. The Unwired Server, for this reason, has several plug-ins available for several well-known security systems, and even then the process to create custom plug-ins for more obscure and/or customized security systems is a possibility. The server has a set of ‘logical roles’ [SYB11a] that represent levels of privileges for either a user or group type which can then be mapped to actual system roles, or ‘physical roles’ on the host system itself. An example of this could be seen in the system admin can map the ‘admin’ logical role of the Unwired Server to the ‘admin’ physical role of the admin group on the system, effectively making every member of the host system’s admin group simultaneously members of the ‘admin’ group on the Server.

2.6 SAP Afaria and Sybase Control Center

SAP’s Afaria and the Sybase Control Center (SCC) are managerial applications that allow for SUP applications, user devices, and the privileges that these both share to be managed, monitored, and controlled. Afaria differs from SCC in that it is primarily a managerial application for the explicit purpose of managing and monitoring users, groups, and their devices that are currently allowed on the SUP’s Unwired Server. One notable difference between the two is that Afaria started off as a Windows application designed to run on Windows Server 2003 for the explicit purpose of managing and monitoring computers and the permissions of those using them and has only introduced mobile support as early as 2005 [SOL05]. However Afaria has made the transition to mobile management very well through its ability to implement incremental patch updates that facilitate compatibility with new mobile devices that may surface between Afaria upgrades. [SOL05]
Through Afaria [SYB10b], an authorized user can see and alter what permissions certain groups or certain users have in the current system, what devices are currently connected and which ones have authorization to connect at all, as seen in Figure 7. This, combined with its ability to monitor information about what servers groups, users, and their devices are allowed to connect to, makes Afaria a valuable security tool for both client-side and server-side purposes. Device status and permissions can be monitored and edited remotely from the Afaria control center, meaning an authorized user can either temporarily lock up permissions of a certain device of a user or even revoke them in their entirety. It is even possible for the device to be remotely wiped of sensitive information through these same methods, giving mobile users an extra level of added security in the event of a compromised device.

SCC, unlike Afaria, is more centered on managing and controlling users and groups in relation to applications deployed on the server already. Devices that have the appropriate credentials to reach the particular SUP Unwired Server can be registered into the system and
afterwards then be assigned certain applications, or workflows as their better known, by an authorized user. SUP stores its list of authorized users in a local LDAP server that the SCC in turn uses to both authenticate and ‘role map’ SUP’s logical roles to the current systems’ hierarchical structure of users and groups.

Like Afaria, the command interface is web-based, meaning that in order to connect to either one, a user would need the proper credentials, a valid internet connection, and a modern internet browser that supports Adobe Flash. Once logged in, a registered user is able to monitor which devices have access to and are currently connected to the Unwired Server as well as monitor and change what permissions each user or device has, as seen in Figure 8.

![Figure 8: Sybase Control Center – Interface](image-url)
3. Implementation

Our plan was to use Sybase Unwired Platform’s utilization of Mobile Business Objects (MBO) to create a mobile application for the explicit purpose of allowing plant workers using SAP’s database services to view and/or place orders for plant maintenance purposes on the fly via remote devices such as tablets and cellular phones. We began by creating a shell application within SUP and populating it with MBO’s, each of which represent a data container on the back-end server we were utilizing. Each of these contained object data that was to be called from the SAP Perth server we originally decided to use. However, we continued to face the same problems that we faced early on in our experimenting with the SUP environment and the back-end SAP server. After switching to another SAP server, we once again found similar results, mainly suggesting that the MBO’s wouldn’t bind to the back-end data due to supposed authorization errors. After even more investigating, we found out that the errors that we were encountering were not due to user permission errors on the back-end database, but rather permissions with the back-end database and the use of MBO’s themselves. The SAP servers that we were originally using turned out to be unsupportive of mobile platform development and thus that was the reason for the errors that we were receiving.

Once we were able to acquire our own dedicated SAP back-end server that was fully supportive of mobile development, we were finally able to have the data bound to the objects that we created in the GUI environment in SUP. We decided to start small and see if we could get any data. An example of information that could potentially be viewed on the original GUI of
a non-mobile platform can be seen in Figure 9, which is what a typical worker/student would see if they were viewing an order.

![Figure 9: SAP GUI - Plant Maintenance Service Order](image)

We spent some time trying to map out the back-end data, which was in the form of several ‘Business API’s (BAPI’s) before we eventually learned of a built-in function within SAP’s non-mobile GUI that has nearly every BAPI to-date mapped and documented out. Through this we were able to learn about the structure of the back-end data in order to know what data was essential to be passed into certain MBO’s in order to get certain order information back from the back-end to the front-end. From this information, we would be able create a simple mobile application that takes in an 12-digit order ‘number’ and return raw order data if the order exists within the server. If the order number does not equal that of a corresponding order within the system or the query cannot be completed, then the application will default to a user defined
preset error screen. This could then be deployed to our Sybase Unwired Server and then assigned to either of our registered devices on the server. Currently, we have several iOS devices and several Android emulators with the proper Hybrid-web-container applications for connecting to and utilizing the applications deployed on our Unwired Server.

![Figure 10: SUP – Workflow Package: Layout](image)

Before we do anything else, we would create a personalization key that will be our link to facilitate passing the user’s input of the order number on the end GUI to the corresponding back-end MBO and set it in as a load argument for our order number variable. We would then deploy the project itself to the server before creating a new workflow package, which is how the structuring of the final GUI and dataflow is orchestrated. Figure 10 illustrated the general flow that this application shell will follow. In this workflow package, we would start by setting up the application to be client initiated, meaning that the user has to log into the application and input information before anything happens.

We would then create a simple start screen and add two menu items, a ‘Get Order’ menu icon and a ‘Cancel’ menu icon. The ‘Get Order’ icon would be used to create a link to our MBO
we want to pass information into on the back-end. We would set the type of action to take place to be an online request before setting the MBO we want into the detail list and subsequently setting the type of query to be a global search before finally generating an error screen to go to in the event something goes wrong later and setting things up so the default success screen to be our MBO return info if our query goes through properly. We would then create a key and bind it with our personalization key from earlier. It is this binding that allows the data the user inputs to make it to the back-end properly.

Figure 11: SUP – Workflow Package: Get Order

Once our ‘Get Order’ menu icon is all set up as seen in Figure 11, the ‘Cancel’ icon is standard and can be set up with default options that come set for cancel buttons. The final part of the application would then be the input area. We would create an editbox and place it in the start window alongside the other menu options. This editbox can be adjusted to allow for a 12-character limit to match the 12 character limit of an actual order number. The input value would then be set to be bound to the key we created earlier. This means that when a user inputs characters into the editbox, the data is passed from the key, to the personalization key, and
finally to the MBO before it is then parsed and evaluated. Figure 12 illustrates what this would look like in the workflow editor.

![Workflow Package: Editbox](image)

**Figure 12: SUP - Workflow Package: Editbox**

4. Results

Currently, we were unable to research very far into the creation of the SUP application that we originally intended to create form the beginning. This was chiefly due to issues that we experienced early on not being resolved into almost half-way into the second semester of our research. We were however able to properly utilize the software required for creating a couple mobile applications that showcase the capability and potential that the SUP has for mobile platform development.

Our first application that we were able to create was a simple demo application that had a dropdown list of airline companies, as seen in Figure 13. Once a company was chosen from the
list, the user would then press a corresponding submit button, which would then send a request to the back-end server and return a list of ‘available flights’ in table format for the user to see. Again, this is simply a demo application that we were able to create from a preexisting tutorial and deploy, however we were unable to do so until we were able to acquire our own back-end SAP server that was compliant with mobile development.

![Image](image1.jpg)

**Figure 13: Hybrid-web-container – Demo Flights Application**

We have a demo application of our own that is meant to take in a 12-digit order number and return raw data about said order, but due to current server problems, we are unable to verify the end result of the application, as are we currently unable to showcase the previously mentioned demo in full as back-end data relationships are not functioning right and as a result both applications are returning a 503 error every time we try to submit data, indicating there is a problem on the back-end either preventing each query from reaching the server or preventing the full execution of our queries.
Based on results obtained through testing environments built into SUP itself, there is little reason to doubt that the application wouldn’t fail. In Figure 14, the results from the submission of order number ‘000000500800’ clearly return valid data regarding the order itself, meaning that the information that gets sent to the back-end database is being assigned to the right value and actual data is being called forth to the front-end as a result. In Figure 14, our demo mobile application can be seen as well, which if the backend data connection was working like it should, would be displaying the same data as well.

Figure 14: SUP – Query Result Checker and our Demo Mobile Application
5. Conclusion

As things currently stand, we have verified that we can definitely create the application that we originally sought out to create, but due to various permission errors coupled with time constraints, we were unable to delve into the technology as much as we would have hoped. For the last quarter of the time we were allotted to work on this project, our backend server started having periods of unreliable downtime, and while it was designated for us to use for our project, it was still hosted off-location and we ourselves were not directly in charge of its maintenance, so a decent amount of time we couldn’t work properly on the project as much as we would have liked. However, after spending as much time as we did with SUP and its internal structure, it is very possible that if we were given a few more weeks and added support on the back-end server, we would be able to create something a little closer to our original goal.

We feel that despite the amount of progress that we made with our final application, we did learn quite an amount about SUP and its inner-workings. The toolset has a serious future in the mobile development market due in part to the sheer versatility it provides coupled with its user-friendly interface.
REFERENCES

[BEA08] Beaton, Jack K.; Myers, Brad A.; Stylos, Jeffrey; Jeong, Sae Young (Sophie); and Xie, Yingyu (Clare). “Usability evaluation for enterprise SOA APIs.” In Proceedings of the 2nd international workshop on Systems development in SOA environments (SDSOA ’08). ACM, New York, NY, USA, 29-34. DOI=10.1145/1370916.1370924 http://doi.acm.org/10.1145/1370916.1370924


[BOR12] Bort J., “This Exec Has An Outrageous Plan To Make SAP Bigger In Mobile Than Apple”, Business Insider, Nov. 4 2012.


[WOR10] Worthen, Ben, and Scheck, Justin. "SAP Strikes Deal for
