The final destination:
Building test bed apps for DIL environments

Arild Bergh
Cyber Systems and Electronic Warfare Division
Norwegian Defence Research Establishment (FFI)
Kjeller, Norway

Recent years have seen a massive growth in the everyday use of smartphones and tablet (here called smart mobile devices or SMDs) and a turn towards app based software to connect to cloud based services. This has had a huge impact on how people access information and communicate in all spheres of life, with always online now being the default mode for most people. These changes also affect the military, albeit at a slower pace. Moving towards a diversified app-architecture for sharing and processing information has clear operational benefits [1]. However, a key differentiator between civilian and military contexts is the fact that in actual operations the military will, at least on the tactical level, operate on disadvantaged, intermittent and/or limited (DIL) networks.

The IST-118 working group is exploring solutions to deal with DIL related issues, and to be able to test these as realistically as possible could be very beneficial. This paper proposes a possible solution, the MLAB App Builder developed at the Norwegian Defence Research Establishment (FFI). MLAB lets non-developers create SMD apps for Android, iOS, Windows Phones, etc. simply by using point and click. This means that polished apps can be created easily, and using built-in facilities for monitoring of user interaction will help to test everything from content filtering to offline use. Apps created in MLAB uses an architecture that allow apps to work on and offline when connections are intermittent and allow for transparent connections to external services or local substitutes.

Keywords: app development, test bed, mobile devices, smartphones, disadvantaged networks, mil-app market

I. INTRODUCTION

We are in the midst of an on-going explosion in the use of smart mobile devices (SMDs) such as smartphones and tablets. Currently 1.75 billion people use a smartphone, with numbers expected to rise to 2.5 billion in 2017 [2], and more than a billion smart phones were shipped per year in 2013 and 2014 [3]. The percentage of people using a smartphone is rapidly increasing in most countries, with more than 50% of the population using them in nine countries [4]. This rapid uptake has also pushed the price below 30 USD for basic smartphones. At the same time there are millions of “apps” (the small, task oriented applications that all SMDs rely on to provide access to information, communication and services) available to tailor these generic devices to one’s own taste and needs. In short, SMDs and apps are currently the default ICT for most people around the world.

Given that these devices have an array of sensors and communication methods they are an interesting proposition for a range of uses, from medicine [5] to emergency situations [6]. This also applies to the military, the idea of using commercial off the shelf (COTS) resources means that there is a number of research projects looking into mobile phone use, both on the end user/app side [7], [8] and the networking side [9], [10].

This is not only an issue of costs, albeit that is a considerable benefit. For the next generation of military recruits SMD and app use is the default way of accessing and sharing information and knowledge. Failing to capitalize on the skills that younger people build up around these devices will likely mean a less effective use of information services in the military.

At the same time “power to the edge” has been a rallying cry in network based warfare/defence for some time [11]. The Norwegian military is also moving from a “need-to-know” way of working, to a “responsibility to share” paradigm. The sharp end of this change will ultimately play out in the tactical domain. The combined benefits of low costs devices with a variety of sensors and communication methods; and the ability to create apps to act as front-end interfaces to different web services makes SMDs ideal as a generic starting point for future gateways to information in a service oriented architecture (SOA).

A key challenge when trying to use COTS SMDs in the tactical domain is connectivity issues in so-called Disadvantaged, Intermittent and/or Limited (DIL) networks [12]. This is not the only problem; the design of devices (size, durability) and battery time are examples of other pertinent issues. However, these can be worked around, whereas connectivity issues are core to the benefits that can be derived from SMDs. To provide reliable access to web services in such environments have been a core issue for many researchers, and in a summary paper from Johnsen et. al. [13] the different issues that the protocol stack needs to handle are summed up.

As we can see in Figure I.1 below, the final destination is “the application” or, in the SMD world, the app. Whereas the other elements in this stack do not have a (end) user interface, the app element will ultimately fail or succeed based on user perception. A failure here, be it through connectivity issues or other problems, can lead to an excellent underlying stack being deemed unusable by the end user.

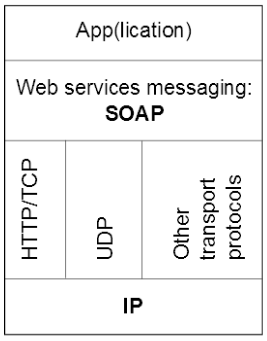

Figure I.1 Adapted from Johnsen et. al. [13, p. 5]

The app(lication) element can therefore benefit greatly from letting real users at the tactical level test likely scenarios. Such testing will also uncover any problems with assumptions made elsewhere about likely data loads, acceptable information update frequencies, etc.

The problem with such testing is that

- apps can take a lot of resources to create, both time and money;
- given the fact that it is a test, getting the right level of polish on the user interface to avoid users being hung up on rough edges in the app which are not actual problems during the test period is problematic;
- it can be difficult without considerable preliminary case studies to be sure that the app fits the way of working that the tactical domain end users expect.

Together these testing issues can skew the data negatively if they are not addressed.

Using the MLAB web based app builder developed by the Sinett 3.0 project at the Norwegian Defence Research Establishment [14] is one way of building test apps that deal with all those issues. MLAB also has an internal architecture that supports DIL environments. This paper will first describe what MLAB is and how it works. Then it will highlight why MLAB can be useful for creating test bed apps for researchers wishing to explore how apps can interact with web services they create and/or infrastructure testing. Finally, it will examine the architecture of MLAB to explain how these benefits can be achieved.

II. MLAB: What is it and how does it work?

A. Background

MLAB is a complete app development environment that “anyone” who have used PowerPoint can use. The key impetus for its development was the emerging requirement to provide teaching material in a format that new recruits to the Norwegian army would be familiar with, i.e. apps [15]. However, MLAB is not restricted to learning apps; it is robust enough to develop any type of app supported by its underlying languages and the target SMD platforms.

Subsequently we identified the following issues that had to be addressed:

- Non-programmers (typically instructors) had to be able to develop these apps without traditional intermediaries such as programmers and designers. This was both a cost issue and a “dialogue overhead” issue; i.e. it can be difficult for a non-programmer to express exactly how they envisage an app should function to a programmer. Enabling them to build apps that are “WYSIWYG” (what you see is what you get) in a web browser by simply pointing and clicking facilitates experimentation and immediate feedback as to how well their choices will work.
- The apps created had to work on different SMD operating systems, such as Android or iOS.
- End users had to be able to explore the apps being built in a way that is familiar to them, i.e. we needed an “app market” similar to the App Store used by iPhones.

B. The MLAB ecosystem

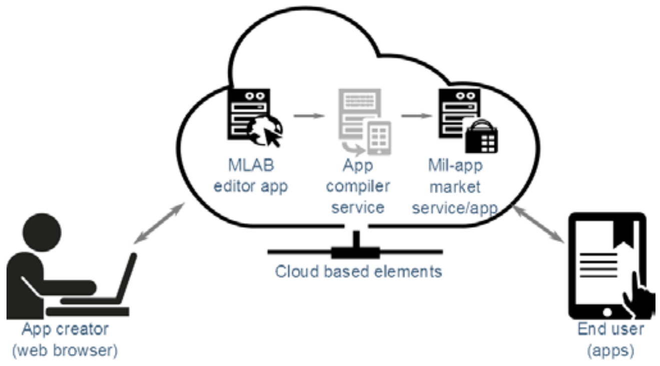

Figure II.1 The MLAB ecosystem: three elements running as web apps and/or web services.

The entry point is the app editor where the app creator, typically an instructor or someone who has intimate knowledge of how a particular task should be performed, builds an app. Each app is built up of one or more pages whose look and feel is based on previously created **templates**. Each page is constructed by adding predefined building blocks. These blocks include **components** that encapsulate functionality such as chat or mapping; **features** which turn on certain app-wide functionality such as usage tracking; and **storage plugins** for storing and retrieving information. Each of these elements can interact with remote web services, but this is not mandatory.

Secondly we have the compiler service which takes pages from the app builder and compiles them to a standalone app. It relies on the Cordova open source platform (see http://cordova.apache.org/) that facilities app development using HTML5, Javascript and CSS3. These are the three core web technologies we encounter in everyday web services such as social media sites or online banking. Cordova gives access to some of the API functions of the different SMD operating systems, such as camera or GPS. In addition it creates a native loading mechanism for the target operating system which, when compiled, loads the first HTML page in the app in the SMDs native web page viewer. Afterwards the HTML/Javascript (and CSS3) code runs on its own, utilizing the latest interactive features of the different languages. This loading functionality is what enables the app's HTML/Javascript code to run on many different SMD platforms without having to be re-written (although it has to be compiled separately for each different operating system).

Finally there is the app market. This is a basic RESTful API service using NodeJS and MySQL to let app creators upload apps, and end users search, browse and download apps. Different front ends can be developed to connect to this API, such as web pages or native apps on the SMDs. It uses a basic database for information such as app name, description, categories, etc.

Together these three elements provide a complete ecosystem that can be installed on basic webservers with PHP/NodeJS and MySQL support. These can be a part of the general Internet, or running on standalone TCP/IP networks. The app editor is currently being finalized and the compiler and app market services have been specified and are soon starting a short development cycle. Once MLAB is completed it is hoped that it will be possible to open source the entire ecosystem; as it is based on open source languages and libraries throughout there are no technical or license problems standing in the way of this goal.

III. Using MLAB to create test bed apps

A. *Potential benefits*

I would suggest that the case for using MLAB to create test bed apps rests on one rather obvious, but less important benefit, and some not immediately obvious, but more important benefits. Clearly the economics are important here, apps can be developed for free by anyone who is given access to the app editor. However, more important is the possibility to enhance the quality of a) test results in the field and b) future research. This can be achieved through five aspects that are particular to MLAB, and one that is common to all SMD based apps. The five aspects all spring from Sinett 3.0's focus on the socio-technical aspects of ICT [16]. In short, a socio-technical focus means that we look at the interaction between people and technical systems and their behaviour around such systems, with an equal interest in both elements.

- Firstly, as novices can use MLAB, a researcher's "clients" in the military can be partners in developing the test bed app. This, I believe, will enhance the quality of the testing in the final part of the protocol stack in Figure I.1. There is a lot of tacit knowledge underlying tasks in the military. Tacit knowledge is knowledge that is so "obvious" to a person that they do not deem it worthy of mentioning to others [17]. However, such hidden knowledge is often the key to successfully completing a task. An example of this can be how threats in the field are prioritized at the tactical edge. By letting the clients develop the apps such knowledge will become a part of the app and is also made visible to the researcher.
- Secondly, there may be different ways of solving optimization issues (as discussed by Johnson et. al. [13, p. 5]), such as content filtering, that the end users can provide input on during tests. Perhaps an aggressive content filtering algorithm which makes the overall test succeed by reducing the network load is actually an overall failure because the end user is getting too little information, too late. MLAB test bed apps can therefore (in)validate test results based on actual tactical domain use.
- Third, MLABs use of preformatted templates and high level components makes it easy to develop polished apps without the rough edges that often characterize test bed apps. This will help researchers by avoiding users getting distracted by problems in an app that is irrelevant to the actual test, but which may still render the test app useless for the tester. Test users may also, for instance, mistake problems in the test app for problems in the network, further complicating the test results.
- Fourth, access to most types of web services, for instance those listed in the *tactical SOA foundation services* by Johnsen et. al. [13] can be encapsulated as components in MLAB. This means that high level experts can use MLAB as a channel for distributing configurable task oriented components such as service discovery or messaging.
- Fifth, this component encapsulation means that additional iterations of a test, even by different teams, can re-use the component that is embodying the service connection and enhance it further, if required.
- The final benefit, not specific to MLAB apps, is the ability to collect, without interrupting the actual use of the app, contextual data from the sensors of the device. For instance, in what positions (and hence, what type of terrain) did users send the most data, when did they request updates, how often was the device in use (i.e. screen on) when data arrived from web services, etc.

The first five benefits will of course vary depending on the scenarios being investigated and the clients one is working with. However, they are all based on the ability to combine the skills, knowledge and user experience of non-technical "final destination" app users in the tactical domain with the technical know-how of researchers. This can be done in ways that are not available when apps are created directly by the researchers or third parties. Furthermore, MLAB created test bed apps allows us to frame particular experiments within a larger app context. By this I mean that issues around access to, and sharing of, information does not happen in isolation, but within a device usage context, and this is "enabled" by using MLAB to create custom test bed apps. In summary, the roles and outcomes when using MLAB to create apps may look something like what is outlined in the table below.

Table III.1 Who does what, and for what purpose, when creating an MLAB test bed app?

Actor:	**Researcher**	**App creator**	**App end user**
Stage:	***Pre app creation***	***App creation time***	***App run time***
Task:	Create a reusable component encapsulating, for instance, web service interaction, including configuration options.	Add component to app being built, set configuration options.	Interact with component, enter/read information to/from device, and hence a web service.
Outcome:	Testable, re-usable code linking to test scenario.	Testing if configuration options make sense, and/or if further settings should be user defined; see what other components are added to work with the component being tested.	"Real-life" feedback, more realistic testing of scenario.

B. *Enabling the potential benefits*

The underlying architecture of MLAB, as discussed briefly in *II.B, The MLAB ecosystem*, is central to unlocking the potential benefits of using MLAB to create test bed apps. It is therefore worth discussing three aspects that will be of particular interest for researchers working in DIL environments.

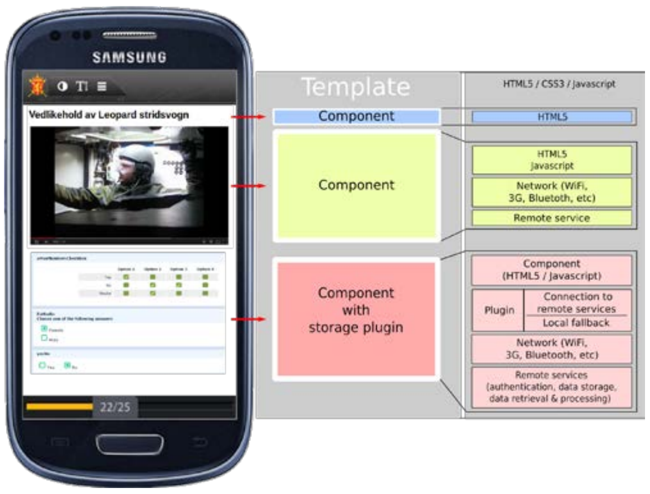
SAMSUNG
Vedlikehold av Leopard stridsvogn
22/25
Template
HTML5 / CSS3 / Javascript
Component
HTML5
Component
HTML5 Javascript
Network (WiFi, 3G, Bluetooth, etc)
Remote service
Component with storage plugin
Component (HTML5 / Javascript)
Plugin
Connection to remote services
Local fallback
Network (WiFi, 3G, Bluetooth, etc)
Remote services (authentication, data storage, data retrieval & processing)

Figure III.1 Outline of the MLAB page architecture

In *Figure III.1 Outline of the MLAB page architecture* we see how a **component** can be anything from a simple heading, or a more complex, composite HTML5 element which uses Javascript for user interaction and *asynchronous JavaScript and XML* (AJAX) to store data remotely [18], [19]. The component is what the app creator interacts with; as they develop an app they select components to add to a page by clicking on icons in a list, just as text boxes, videos and images are added to a PowerPoint presentation. A component can (if required) be written so that it requests configuration parameters from the app creator at design time, e.g. what server URL to use. In short, the app creator always works on the level of components, and need no programming experience. The component developer however, can make the underlying code as simple or complex as is required. The component can be 100% Javascript, or pure HTML5 code, or (most often) a combination of the two, with HTML5 fulfilling the traditional role of displaying information, and Javascript manipulating it.

There are two other aspects of MLAB that are useful for test bed app development. The first is of special interest in a SOA setting, it is the **storage plugin**. These plugins are developed just like a component, except they do not have a visual interface. The idea is that a chat component, for example, can be developed that has all the interaction elements required (display chat, enter new text, send button, view history, start new chat button, etc.). Then the app creator selects a storage plugin to connect the chat component to a server, this could be a simple MySQL database with custom code to read new messages or a complete XMPP server. This gives a lot of flexibility both for researchers and app creators.

The storage plugins use an internal MLAB API that has been developed to support storage facilities. A key aspect of this is that all data is first stored locally, and then pushed to the final destination as and when a connection is available to the storage plugin. This means that by default, basic DIL network handling is a part of MLAB. Furthermore, the plugin itself can add additional handling of problematic network connectivity. It could for instance skip all updates except the last one when connectivity is restored (if this is a realistic method for handling connectivity issues obviously).

The second aspect that will be useful for test bed applications is **features**. A feature is an app-wide element that can be used to add functionality that runs in the background, regardless of which page or component the end user interacts with. This could for example be a position tracking feature which reads the GPS position at regular intervals. Features, which are a type of component, can also be linked to a storage plugin, so in the GPS example positions can be stored, and then shared with other users of the same app on other devices.

Framing all this is the **template** which takes care of the formatting (i.e. look and feel) of *all* components. This means that component developers (who in our case would be a researcher) can focus on the technological side, without the overall user experience suffering from lack of attention.

C. *Example of a component that faces DIL issues*

To wrap up this section it is useful to discuss an example of a component that highlights what sort of issues an app in a DIL environment would have to tackle. This is a simple counter component for military personnel to count the number of certain types of observations or incidents. It has been requested by representatives of the Norwegian military and thus represents a real need, and is not a theoretical use case. It has not been developed yet; here I just raise the issues that we envisage having to tackle, without presenting any solutions.

For simplicity's sake we will assume there is no overlap in the counting, i.e. each counter increment represents a single, unique observation. The train of thought when developing this component would follow these broad outlines:

- In a standalone version where each user uses it locally this component would simply increment an integer and store it on the device using the standard MLAB storage plugin API. This would obviously not be affected by DIL network issues.

- In a networked version where several users use it to count the same type of observations, this is also trivial as long as they all have network connectivity all the time. One would have a server with a function which, when called, incremented a central counter and returned the latest value. There may also be a read-only function to get regular updates of the counter on devices that had not sent any updates for some time.
- In a DIL network environment the counter needs to be stored locally, and then sent up to the server. We would need a different server function which increments the counter with a specified value so the component can make a single update call when multiple counts have been cached locally when the network was unavailable.

However, the above solutions are technical solutions, this is where MLAB test bed apps can expose some user interface issue. The core question for a real life user would be *"how much can I trust this counter?"*

- First one could add a "last updated" message in the component. This could be enhanced with further information of how often other users have updated and their most recent submissions. Such information would help the end user to evaluate if the current count is realistic or is better ignored; frequent counter updates followed by mainly silence could be a bad sign.
- Some mapping information showing where the updates were last updated could further enhance trust in the data, if one's own are has only been silent for a minute, whereas other areas have been silent for an hour it could indicate that one should have less trust in information from that area.
- One could consider adding some form of "network selection" facility, so one could use a *mobile ad hoc network* (MANET) [20] (through Bluetooth or Wi-Fi direct for example) which would let users nearby update their counters.

This is by all means not an exhaustive list of DIL network environment issues, but it does show how test bed apps, and the MLAB architecture, can help researcher better understand what features the elements in the protocol stack should support to enhance the user experience of tactical domain users.

IV. Conclusion

The MLAB ecosystem is currently work in progress, with completion of a usable version estimated mid-2015. This paper has hopefully shown that being able to easily create apps that are user friendly and focus on realistic tasks can enhance the research in ways that extend beyond the purely technical. The inclusion of the user (and user's managers') view points and experiences should help to improve the acceptance of protocols and algorithms that are developed. At the same time, the component aspect will enhance the re-usability and sharing of code between researchers, and between researchers and tactical domain users.

[1] M. Tortonesi, A. Morelli, C. Stefanelli, R. Kohler, N. Suri, and S. Watson, "Enabling the deployment of COTS applications in tactical edge networks," *IEEE Commun. Mag.*, vol. 51, no. 10, pp. 66–73, Oct. 2013.

[2] "Smartphone Users Worldwide Will Total 1.75 Billion in 2014 - eMarketer." [Online]. Available: http://www.emarketer.com/Article/Smartphone-Users-Worldwide-Will-Total-175-Billion-2014/1010536. [Accessed: 12-Jan-2015].

[3] "Gartner Says Sales of Smartphones Grew 20 Percent in Third Quarter of 2014." [Online]. Available: http://www.gartner.com/newsroom/id/2944819. [Accessed: 12-Jan-2015].

[4] "Worldwide Smartphone Usage to Grow 25% in 2014 - eMarketer." [Online]. Available: http://www.emarketer.com/Article/Worldwide-Smartphone-Usage-Grow-25-2014/1010920. [Accessed: 12-Jan-2015].

[5] E. Agu, P. Pedersen, D. Strong, B. Tulu, Q. He, L. Wang, and Y. Li, "The smartphone as a medical device: Assessing enablers, benefits and challenges," in *2013 IEEE International Workshop of Internet-of-Things Networking and Control (IoT-NC)*, 2013, pp. 48–52.

[6] R. Ferrus, O. Sallent, G. Baldini, and L. Goratti, "LTE: the technology driver for future public safety communications," *IEEE Commun. Mag.*, vol. 51, no. 10, pp. 154–161, Oct. 2013.

[7] J. Keller, "DARPA to create app store of military mobile apps that run on rugged smartphones and tablets," *Military & Aerospace Electronics*. [Online]. Available: http://www.militaryaerospace.com/articles/2013/04/DARPA-Transformative-Apps.html. [Accessed: 04-Apr-2014].

[8] V. Kaul, C. Makaya, S. Das, D. Shur, and S. Samtani, *On the Adaptation of Commercial Smartphones to Tactical Environments*. 2011.

[9] "Military Tests 4G LTE Technology During Bold Quest 13.2 > Camp Atterbury Joint Maneuver Training Center (CAJMTC) > Latest News and Multimedia Releases." [Online]. Available: http://www.campatterbury.in.ng.mil/PublicAffairs/LatestNewsandMultimediaReleases/tabid/781/articleType/ArticleView/articleId/1311/Military-Tests-4G-LTE-Technology-During-Bold-Quest-132.aspx. [Accessed: 04-Apr-2014].

[10] "Remote Troops Closer to Having High-Speed Wireless Networks Mounted on UAVs." [Online]. Available: http://www.darpa.mil/NewsEvents/Releases/2014/04/07.aspx. [Accessed: 07-May-2014].

[11] D. S. Alberts and R. E. Hayes, *Power to the edge: command, control in the information age*. Washington, DC: CCRP Publication Series, 2003.

[12] J. Sonnenberg, "Disconnected, Intermittent, Limited (DIL) Communications Management Technical Pattern," 2009.

[13] F. T. Johnsen, T. H. Bloebaum, P.-P. Meiler, I. Owens, C. Barz, and N. Jansen, "IST-118–SOA Recommendations for Disadvantaged Grids in the Tactical Domain," DTIC Document, 2013.

[14] A. Christensen, "Mobil-apper uten programmering for Forsvaret," *forskning.no*, 11-Nov-2014. [Online]. Available: http://forskning.no/teknologi-data-informasjonsteknologi/2014/10/snart-kan-ola-soldat-lage-mobil-app. [Accessed: 12-Jan-2015].

[15] C. Jackbo Gran, "Mobile Bakery," presented at the NORDEFCO ADL Conference 2014, Gol, 14-May-2014.

[16] M. K. Fidjeland and B. K. Reitan, "Web-oriented Architecture-Network-based Defence development made easier," FFI, Kjeller, Norway, 2009/01784, 2009.

[17] H. M. Collins, "What is tacit knowledge," *Pract. Turn Contemp. Theory*, pp. 107–119, 2001.

[18] A. Bergh, "From the death grip of PowerPoint to mobile freedom: The Mobile Learning App Builder (MLAB)," FFI, Kjeller, Norway, 2014/01452, Aug. 2014.

[19] A. S. Hofgaard and A. Bergh, "Komponenter til app-byggeren MLAB," FFI, Kjeller, Norway, FFI-notat 2014/01462, Forthcoming.

[20] G. Hinojos, C. Tade, S. Park, D. Shires, and D. Bruno, "BlueHoc: Bluetooth Ad-Hoc Network Android Distributed Computing."