Challenges of Mobile Application Development

There is no doubt that the application development platforms we see today are getting more powerful everyday, with an ever growing list of funky features, bells and whistles. But there is a downside to this, in that these development environments are themselves getting more complex and more difficult to master.

Hypothetically speaking, given *infinite* time and resources, it becomes possible for very powerful enterprise solutions to be developed and deployed. However that is not the reality and development projects can too often turned to disaster, when a supposedly *productive* tool assumes a monstrous life of its own, and greedily devours valuable and limited project resources. And unfortunately, the matter does not *stop* there.

Further, the **necessary domain expertise** has become vital for the success of these development projects in order to deliver a successful solution expected by the business. Such expertise is very hard to come by though. Project leaders *not only* need highly competent technical teams, capable of driving powerful development platforms, *but they also* need a team with the right domain experience for success.

Generally speaking, necessary domain experience refers to the specific project focus. For instance, financial applications developers need to be conversant with the financial business processes and jargon. This comes back to the need for development teams to be able to communicate and to understand the business requirements and the expectations of their customers.

The latest Java and Microsoft's .Net platforms are very powerful. But they are also *very complex* platforms, while remaining generic in nature. Therefore it becomes impossible for them to wrap a particular domain expertise necessary for projects in specialised fields.

Let's be more specific and talk about the true enterprise mobility requirements. As with similar enterprise systems, there is a very long list of challenges that need to be addressed. They unfortunately cannot be handled with your favourite programming language like Java, C#, or C++ and the platforms they come with. They come with *neither* the mobility specific features *nor* the expertise required.

Mobile applications are inherently complex. They need to be implemented using a three-tiered architecture which creates many challenges for developers in several key areas; occasionally connected mobile application development, providing reliable data exchange, backend middleware infrastructure, data integration mechanisms, scalability and security.

It is obvious that one needs more than just a low level programming tool. What is needed is a Commercial off-the-shelf (COTS) mobility platform. (COTS is a term for systems which are manufactured commercially, and then may be tailored for specific uses).

A brief list of mobility requirements and challenges includes:

- Handling occasionally connected networks over limited bandwidths
- Ability to operate in various modes (offline, occasionally connected, online)
- Robust, industrial-strength data synchronisation engine
- Security
- Scalability
- Mobile device programming
- Flexible mobile forms development environment
- Data integration to and form backend systems
- Seamless application distribution and updates to field devices
- Easy integration into mobile peripherals including printers, barcode scanners, card readers etc.

This list could even be longer. Imagine trying to handle these requirements from scratch using Java or Microsoft .Net platforms. One would end up with many sleepless nights and lots of stress.

Mobile Device Application Development

Creating a standard application on a mobile device itself would requires a huge amount of platform knowledge and a programming experience, not to mention the specific experience in C# or Visual Basic, Windows Forms Programming, database programming etc. for a .Net programmer. The list goes on and on. Similar issues arise with the Java platform.

Another level of complexity is introduced due to the different levels of platform support for different footprint devices. Microsoft .Net has its Compact Framework as the counterpart on the Windows CE based devices. Java SE (Java Standard Edition, J2SE as previously known) has the limited device version of the platform known as Java ME (or J2ME) which comes with many different configurations. Most of limitations of the Java ME platform are being addressed by the device manufacturers by providing additional libraries and frameworks to offer a workable environment for developers. This makes it impossible to have portable solutions across multiple devices with different capabilities.

Data Synchronisation and Communications Complexities

The next challenge is to implement a data synchronisation infrastructure that transmits secure business data between mobile devices and backend systems. Knowledge of communications protocols and experience with occasionally-connected networks and their hidden traps and bottlenecks becomes imperative. However, this skill-set is not a commodity that you can easily buy from your next door IT shop.

Next comes the transaction management over communication channels; what happens if the device gets disconnected while it is transmitting data, etc. Things can get really hairy and unpredictable. Well, OK, some may say: “there are database products available in the market that come with built-in data replication engines or agents. Why can’t we use them?”.
Yes, but they don’t come for *free*. Besides the dependency on a specific database vendor, other questions come to mind, such as whether they are suitable for a mobile environment, whether they can provide a scalable platform that will grow with the business, so on and so forth.

Middleware, Security and Scalability

Writing a client application for a specific device platform is just the tip of the iceberg. This is where things start to go wrong for those who think this is the only thing they need to do.

“Well, I have a very nice couple forms running on my device, isn’t that all I need ? Let’s deploy them to devices in the hands of remote field users”. Not quite!

How about the data synchronisation and communications related complexities mentioned above? What about a scalable backend infrastructure to handle multiple concurrent user connections? And managed access to backend resources, databases, etc? What about security related issues? Surely someone has to worry about and deal with these aspects of a mobile solution.

With the advent of the application servers, things have gotten better for the experienced developers. One does not have to worry about implementing many low level requirements relating to concurrency, resource management, transaction management, and security.

However the amount of expertise in terms of configuration, design and development of any business components using application servers is still very high, and it requires a considerable amount of resources and experience. This is obviously not every business’s cup of tea when deploying solutions.

Challenges of Integration into Business Systems

Just exchanging data between mobile devices and a central repository or database may not be the only requirement of a mobile solution. In most cases data needs to be collected and distributed at the backend, to and from different business systems. This is another challenge when creating mobile solutions. One would need to consider using a purpose-built product rather than a generic programming platform to address the data integration requirements.

Where to go from here?

The moral of the story: Use the right tool for the right job.

In order to manufacture or create any product, a set of suitable tools needs to be identified and used for the job. Creating mobile solutions is no exception to this rule. It is always very attractive from the developers' perspective to get excited about new challenges and to try to solve them, even if it would sometimes mean reinventing the wheel. We do all share the "I have done it" ego which is probably built into our genes. It is sometimes easy to get carried away.

For successful mobility solutions, one would need more than a generic programming language or platform. A more highly abstracted product set is needed to cover the basic requirements of a mobility solution so that the teams can focus on just the business requirements rather than the intricacies of mobile software development.

At Bright Software, this is exactly what we have being working very hard to achieve: A set of tools and products that provide a solid platform, a good starting point from which application developers can start building mobile solutions.

Our aim is to address the infrastructure requirements discussed here and to allow them to concentrate their efforts on the business's real mobility requirements. Most importantly we strive to enable businesses to do so without the need for decades-long software development experience and mobility expertise.