

From Big Iron to Little Iron ...and Beyond

Copyright © 2006 by Apacheta Corporation. All rights reserved.

Peter Clare

Mobile solutions are repeating the computer industry's transition from proprietary computer systems to commoditized hardware. Apacheta's "Reuse-Oriented Architecture" offers a better way of developing enterprise mobile applications to meet this demand.

It reminds me a little of the old days. You know, when Big Iron ruled the earth. We would buy a piece of hardware - a BIG piece of hardware to do what today would be considered a small job. Software was secondary - and minimal. Except for the raw operating system, everything was custom-built in-house.

The minicomputer business changed all this. Sure, smaller and cheaper machines could do bigger and more complex tasks, but this was not the real driver for change. The real driver was the fact that there were so many new hardware manufacturers offering so many new choices but there wasn't any software for these new, cheap, and powerful machines. The new minicomputer hardware manufacturers were, well, hardware manufacturers - they didn't know anything about software. For the most part, they couldn't code their way out of a paper bag. Most of them didn't even know how to build an operating system for their own boxes. They had to go out looking for something, anything, to run on their hardware.

Enter Unix, and Oracle, and a handful of other software technologies whose business successes were built on this desperate need for something,

Hardware has become a commodity...

anything, to run on these minicomputers. Microsoft, of course, was still a gleam in some guy's eye - not even on the radar yet as far as enterprise systems were concerned. And what did all this desperate need accomplish? It commoditized the hardware. Customers started buying software - and they didn't care what box it ran on.

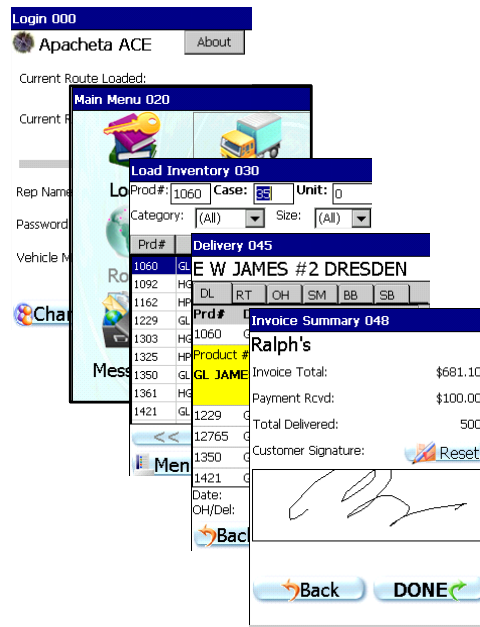
The survival of hardware manufacturers depends on their software solutions...

And the rest is history - except in mobile. Mobile solutions are still dominated by hardware manufacturers. Only now we are buying Little Iron instead of Big Iron. It reminds me a little of the old days. But the handwriting is on the wall. Mobile hardware is getting cheaper and more powerful - and commoditized. And customers are beginning to base their purchasing decisions on what software is available, and they are beginning to shy away from custom operating systems and solutions that lock them into a particular hardware platform. Clearly, hardware margins are plummeting. Hardware manufacturers are beginning to realize that they must provide software solutions - that their very survival depends on it. And these software solutions have to be developed using well-understood and popular technologies.



A problem in mobile is that there aren't that many well-understood or popular software development technologies available. Mobile solutions are for the most part still built like the old days when software solutions were created using tools which were unique to a particular platform. Most mobile solutions today are monolithic applications built using software development tools and techniques which went out-of-favor at least a decade ago. Sure, companies like Microsoft are putting a lot of effort into penetrating the mobile market, and they have provided some very useful tools no doubt.

You can't just dumb down your enterprise system for mobile...



But there are gotchas. Picture this. You start with a large, complex, and very sophisticated enterprise system which has evolved over decades and which embodies your business processes as effectively as your organization can muster. Sometimes, some of your employees need remote access to your enterprise system so you might start by dumbing down some of your critical applications for use on a PC. Sort of works. Enter the truly mobile device: small, sometimes connected, and incredibly difficult

to develop, deploy, and use. You can't just dumb down your software another step for your mobile workforce. The user interface must be re-designed, the data structures must be tailored to this new device's characteristics, the communications strategies employed must be radically different, and even your business process must be altered. This is not your father's Oldsmobile we are talking about here.

A web services model does not work in a mobile, sometimes disconnected, environment...

And if you have any doubts, try using internet access on your mobile phone or PDA. Check out your WAP (What-A-Pain) browser. Only the most desperate amongst us would use this stuff for anything useful. Building a mobile enterprise application based on an on-demand web services model is insane. Let me repeat that - insane. It just doesn't work. You have to re- envision your business process, using the thick client capabilities of the mobile device to provide the instant gratification user experience that makes a good mobile application very very sexy. And you have to re- envision your business process to cater to the sometimes connected (aka, often disconnected) real world environment. Click, I'm in front of my customer and I'd like to get some information from our enterprise CRM system - but I don't have connectivity or connectivity is so slow that I don't know I'm connected. What am I supposed to do, wait... *Sure*, the customer will wait.

Remember all that commoditized hardware? Some of your mobile workforce are using cellphones as mobile application client devices, some are using PDA's. Some are using this hardware or that hardware platform. What are you supposed to do, build a different version of your mobile application for each new scenario as it unfolds each year. And what about software updates. How do you deploy updates to your mobile workforce? Are you going to download the tens of megabytes or more of monolithic code everytime some bug gets fixed or some enhancement is required. Deploy over-the-air, that is, to hundreds or thousands of mobile devices - or does everyone have to bring their device in for the update. *Uh..., right.* And what do you do if the business process or look-and-feel is a little different or very different in various sub-populations of your mobile workforce? How do you control this Beast

Apacheta's "Reuse-Oriented Architecture"

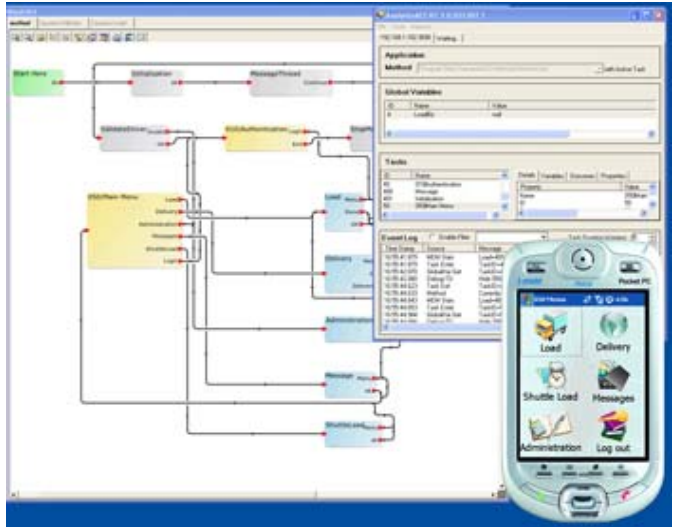
Enter Jim and Steve, our dynamic duo who have been re-born as part of Apacheta Corporation. After nearly twenty years in mobile middleware, Jim and Steve's team (The Band, as we call them) have created a software development platform based on their experiences, and



presumably their failures, in building mobile applications in a wide variety of customer environments. I have been involved in software development for a long long time, and I can say that these guys are good - very good. Of course applications which are built using Apacheta's mobile development platform, VisualACE, get all the Motherhood and Apple Pie of seamlessly sitting as add-on's to the Microsoft C# Visual Studio or the Java Eclipse IDE's. A mobile application developer also gets all the benefits of XML-based data integration, SQL database access, Business Facades, and a Model-View-Controller (MVC) architecture which decouples the data model from the user interface view from the flow control of your application. Motherhood and Apple Pie in today's world.

Apacheta provides Motherhood and Apple Pie...and a lot more.

But this is not all. No no, this is not all (for you Cat-in-the-Hat fans). The very DNA of Apacheta's solution is designed to cater to and enhance the mobile experience. Apacheta's MVC architecture encourages the



The DNA of Apacheta's solution is designed for the mobile environment...

building of simple (and small) software components called "tasks". The flow of control through an application is established in VisualACE via a visual workflow paradigm by dragging-and-dropping and connecting user-defined task icons representing simple business processes like Display Customer Info Screen or Capture Signature. Any one of these simple tasks can be independently developed and deployed over-the-air in real-time, and can be connected or re-connected to any other simple task in the entire business process. These changes can be deployed to any individual mobile device or any set of mobile devices without changing *any* of the code already running in that device.

Deploy your changes safely and efficiently over-the-air...

So let's say you have 5,000 mobile devices deployed out there. You want to roll out some fixes or enhancements. You can choose one or ten or a thousand devices and deploy your changes over-the-air, say 10K bytes transmitted per device - 50Mb for a complete deployment to all devices. Beats sending 5,000 times say 20Mb (is that really 100G bytes?) using traditional techniques, wouldn't you say?

Ok, say you deploy your changes to a 100 or so devices and it is a disaster. My bad, something doesn't work. With Apacheta's solution, you can rollback the changes *without changing any code*. All you do is re-connect the tasks which define your workflow using VisualACE - very easy, drag-and-connect

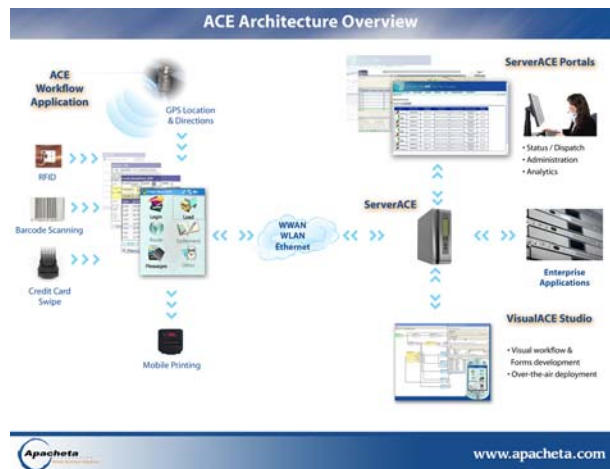
the tasks. Now deploy the XML which defines your new workflow to the affected devices and Apacheta's Virtual Machine takes care of the rest. Fini.

Apacheta uses a transaction model designed explicitly for the mobile environment...

Ok, what about my data? Apacheta's solution uses a transaction model designed explicitly for the mobile environment. Jim and Steve's twenty years in mobile really shine here. Data is bundled up into so-called "manifests" and shipped securely and efficiently using least-cost based routing. Apacheta's transaction model allows data to travel via multiple paths and guarantees data consistency (i.e., no partial data) even if bits and pieces travel via different paths. So for example, your device may start transmitting via WAN but come into range of an 802.11 network before completing a transaction. Apacheta's VM will continue sending your transaction where it left off using the faster (and cheaper) 802.11 network and will assure that you get everything in the correct order even if the later 802.11 data arrives before the slower WAN data.

Now let's say you want to update some data on a mobile device but you aren't really sure which devices have what data. I don't know about you but I have to re-boot my PDA phone every couple of days and who knows what state my application data is in after this.

Apacheta's manifest system will only send the data that is needed. So, for example, you can just tell the manifest system to ship the entire mess and it will figure out what you already have and only ship what is missing. So your application is always guaranteed to have a consistent set of data objects. Pretty handy, huh?



Apacheta uses least-cost routing via multiple asynchronous paths...

I mentioned least-cost routing. You can assign a value to individual pieces of data and a cost to different network transport mechanisms. So data will be sent as soon as an appropriately priced network transport becomes available. This allows you to make choices about what to send based on how valuable you think the data is.

Apacheta's business analytics can give you a powerful inside look into your business process...

A mobile application is by its very nature close to the person using it. In fact, because a mobile application is usually an integral part of your mobile workforce's professional day-to-day life, often carried around on a belt or in a purse, an Apacheta-enabled solution can give you a powerful inside look at your business in real-time. Apacheta's Virtual Machine constantly gathers information about the tasks it is executing and can log or send selected information back to a business analytics server without any programmatic intervention - this ability is built deep into Apacheta's architecture and is available to any Apacheta-enabled solution. For example, you can use this information to determine which business processes are effective and which are not, or figure out how time is spent in various business processes. You can try re-ordering your business processes to see if efficiency and ease-of-use increases or decreases.



If you have a GPS, you can analyze or optimize your business processes based on where they are performed or where they are likely to be performed in the near future. With a local Bluetooth or 802.11 connection, you can access external device information from a mobile device and fold this information into your business analytics stream, determining for example whether a collocated machine is operating properly and/or enabling your corporate resources to assist your mobile workforce in real-time.

Apacheta's solution encourages Best Mobile Software Practices...

Apacheta's solution encourages, and in some cases, enforces Best Mobile Software Practices and greatly facilitates the creation and deployment of your mobile applications. There is really nothing quite like it today and there is absolutely no doubt in my mind that Apacheta is paving the way toward enabling the next wave of serious mobile enterprise applications.

Apacheta is delivering products in selected vertical industries...

As a proof of concept, Apacheta has developed and is selling vertical applications in Direct Store Delivery/Route Accounting, Transportation/Proof-of-Delivery, and a few other selected micro-verticals. We are eating our own dogfood so to speak and the results so far have been nothing short of amazing. This stuff really is disruptive. Apacheta's solution is ideal for extending business processes which are, right now, pushing against the edge of the infrastructure and which you need to push out into a mobile, sometimes connected, landscape.



If you are developing a mobile application, you really need to take a look at Apacheta. 'Nuff said.