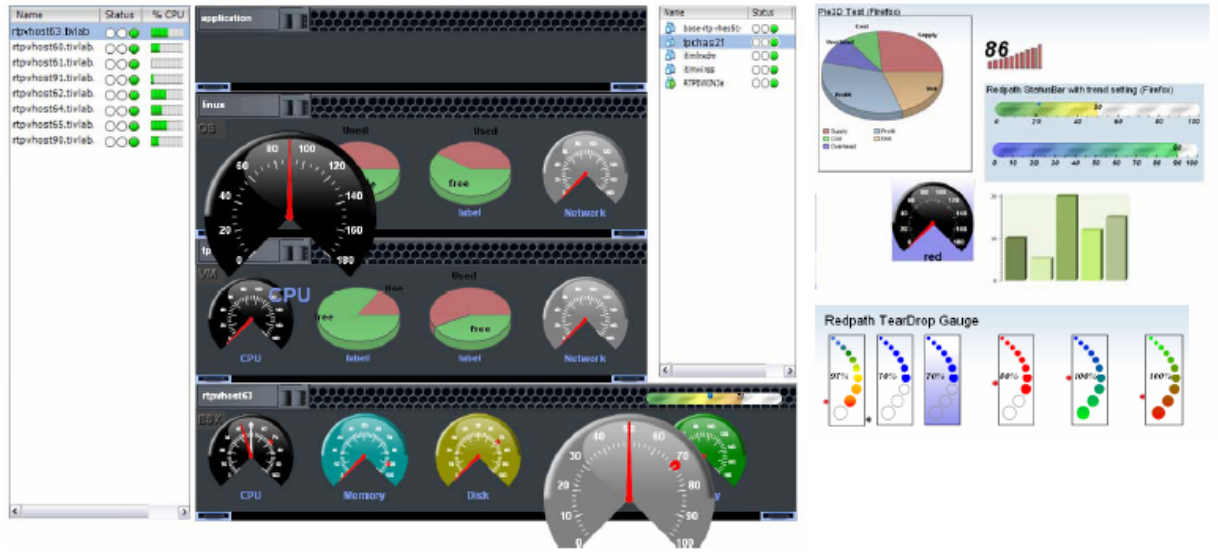
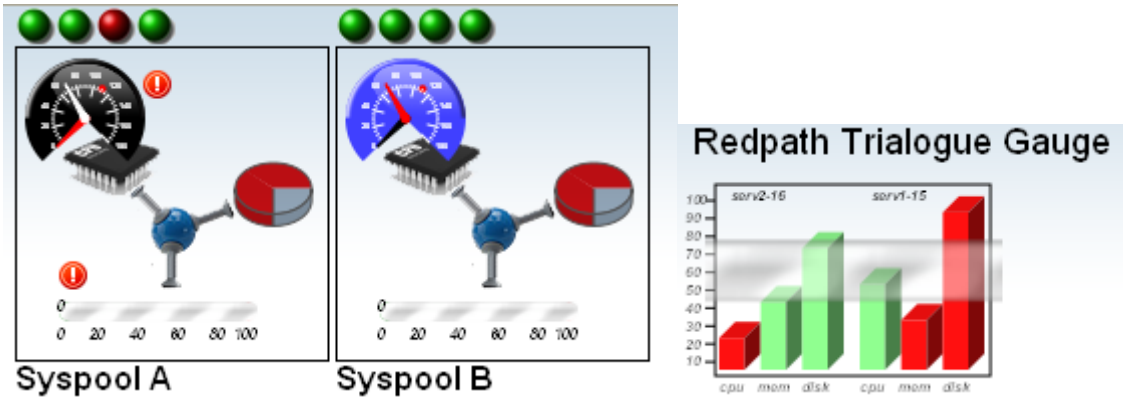
Richard Redpath

email: richardredpath@hotmail.com
website: richardredpath.com
phone: 302-588-4002 (EST text/cell)

Effective Operations

This document presents IBM hands on effective operations (projects) to understand my catalyst position with IBM. Each project will have some captivating visuals and a note for more detail documentation. My role at IBM SWG (Software group) is incubator projects. Previously I was a research staff member of the IBM Watson Research Center and continue to do ready to use product exploration for IBM Software group. All projects are 100% of my work unless noted.





Tivoli DoJo Widgets and BSM Dashboard redesign (2009):

Storyline:

IBM Tivoli is moving to DoJo a JavaScript Platform for its Business Service Management for scaleability of its Tivoli servers enabling the client machine for processing via web browser access. I have been given the directive to redesign Dashboard Business Service Management interfaces as well as prime the widget component infrastructure.

Problem:

Sophisticated well rendered widgets are required for the BSM environment. Our customers are looking for data to leap at them. A high performance client based offloading platform is required for scalability of Tivoli servers.

Results:

Some innovative dashboard widgets have been created which can leap out for attention or closer inspection of values. Patents have been filed. Additionally work was done for concepts in the Virtual Machine Dashboard.

Tivoli iWLE Dev System for iWidgets (2009):

Storyline:

IBM Tivoli is moving to DoJo a JavaScript Platform for its Business Service Management via the Web. Pages are to be created using the new iWidget Lotus Standard. iWidget is a JavaScript based platform which allow the client browser to perform processing.

Problem:

The current development environment is 450mb using a compile package deploy concept with start and stop of a development client server. The current alternative eclipse system is basically the same as the 450mb aforementioned system. Development time and debug is very slow. We have an idea to have a Tomcat based dev system for all Tivoli.

Result:

The Tomcat iWLE dev system is 10mb and installs in 6 seconds verses the 450mb system which installs in at best 20 minutes. There are many advantages in time and debug over the old system. IBM is in a unique position now to expedite a client based BSM system from development for our customers. Additionally iWLE has automatic wiring of iWidgets for events to communicate.

Tivoli TIP iWidgets enablement (2009):

Storyline:

IBM Tivoli Integrated Portal (TIP) is a dashboard server which is Portlet based for deployment of web pages. I am on loan to the TIP team to enable this system to support the Lotus iWidget platform which is JavaScript based.

Problem:

This is a very complicated and convoluted infrastructure that has been edited by many IBM contract development sites world wide. It is quite inefficient with goals for 12 to 14 seconds as a standard to display a page. Design documents do not exist.

Work:

My work on this project was the event processing between all artifacts iWidgets and Portlets as well as GUI construction metaphors used for Portlets and now for iWidgets.

Result:

The IBM Tivoli Integrated Portal now supports the iWidget platform along side with Portlets. My work was the automated wiring and event process of iWidget components and Portlets. Initially the TIP team estimated 2Q 2010 to be available and we were able to deliver an operating environment 4Q 2009. The goal was to enable use of iWidgets or Portlets seamlessly as artifacts on a page. The user will not know the difference. For the first time there are some in depth design documents that can be referenced by the TIP Team which is located in IBM China now for my work.

iFrame security of iWidgets (2009):

Storyline:

IBM Tivoli is moving to iWidget a JavaScript Platform for its Business Service Management via the Web. iWidgets can be hosted on any server and do not have to exist at the Dashboard server location.

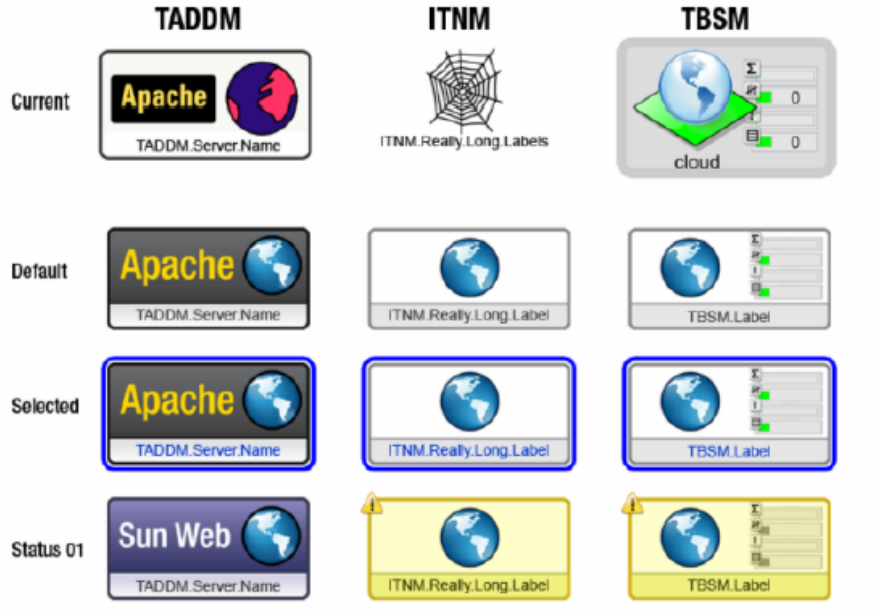
Problem:

There are security issues at hand for JavaScript based loaded pages from a proxy Dashboard server platform as the URL looks like it is from the secure server. Proxy loading is important as it allows the browser to obtain data from different hosts that have been secured through single signed in. The standard is to html iFrame the artifact http source which will prevent the source loaded JavaScript from accessing the DOM of the browser. The short coming for this solution is that events cannot be disseminated between iWidgets for wiring (automatic) hence the iWidget model is broken.

Result:

A tangible exploration was performed and revealed aspects that need to be considered for a viable product and its prereqs. The document is enclosed. The exploration was very important as I found that open source code (OpenAJAX) had some serious short comings as well as not viable for IBM as some customers do not allow open source code in some IBM products. A solution was created as a sandbox that provides the function we needed without open source code but required only one prereq if using the Microsoft browser and that was IE 8.0.

Document: iFrameExploration2.pdf



Tivoli TADDM visual/platform redesign (2009):

Storyline:

The Tivoli Application Dependency Discovery Management (TADDM) product is changing the internal platform design from Struts and Portlets to Javascript by the IBM Poland Team for the new release. The new client processing concept is the edict for Tivoli to scale Tivoli servers. The TADDM product has 1200 visuals that are to be modernized. I was chosen to use my design skills as well as my tech experience to create a universal model for all Business Service Management products of Tivoli and to help TADDM get to the new platform architectute.

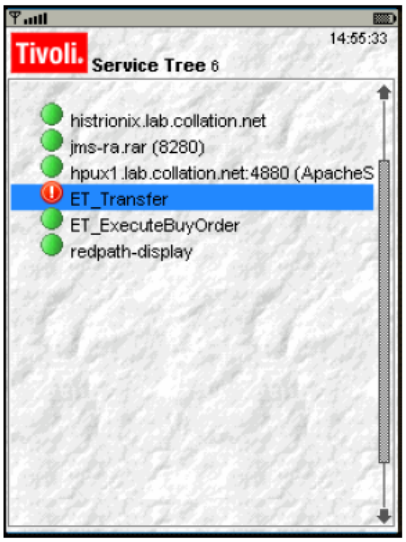
Problem:

This required socializing visual design people to my ideas as well as careful political management. My intent was to change the primitive child metaphors (the visuals are SVG). In addition, this required strategic decisions for the success of the product delivery which politically conflicted with the needs of other IBM software components that are being forced down into products that are not ready. There are technical, visual and political issues at hand.

Result:

An internal application was created which enabled the visual design team to create 1200 images from a standard set of SVG images (100). This deriving permutation process will enable the customer to create new metaphors in the future that are consistent. Decisions were made that did not include some IBM infrastructure being canvassed by others (temporarily) to meet delivery of the TADDM product. The successful delivery of the TADDM 7.1 product has enabled the developers to delay adding infrastructure that has now been realized that is not needed. The product has the opportunity to lead a new direction for best processing on the client and lighter weight as well as performance. The IBM infrastructure that was being canvassed by another group which required political savvy to avoid is now materializing as a deprecated direction for IBM. Our intent was to avoid this competent complicated infrastructure that we thought was a losing proposition (it required clever political steering to avoid).

History: TADDMhistory.pdf



Tivoli TBSM Blackberry status (2008):

Storyline:

The IBM Tivoli TBSM product (Tivoli Business Service Management) was asked by customers to provide a Blackberry SLA (Service Level Agreement) Alert application.

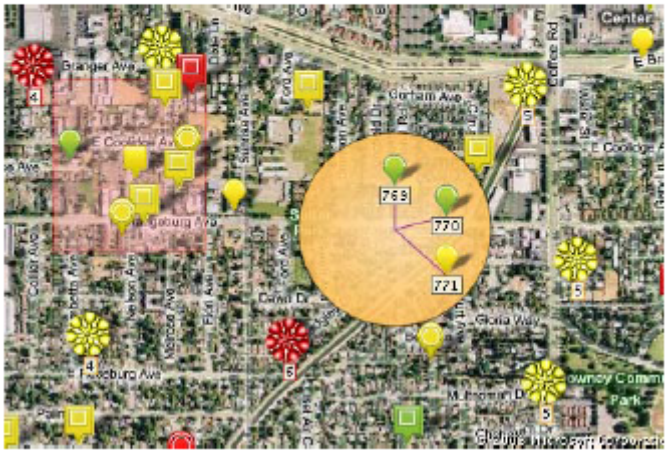
Problem:

IBM development was given the directive to enable the customer to connect to a web site page to install the blackberry app and configure to access the customer's Tivoli management system in a seamless manner.

Result:

The customer can simple connect to a web page and click and install the Blackberry application to configure which resources to manage. The Blackberry application performance was 400 status items in 2 seconds in a periodic check process. This performance was significantly better than a desktop web page which took 12-14 seconds of which I had pointed out in the past the absurd inefficiencies done on the desktop that should be addressed and now had solid evidence. This project required the creation of unique Blackberry widgets.

Document: TBSMblackberryStoryboardV7.pdf



SmartMap (2008-2009):

Storyline:

IBM Tivoli under BSM (Business Service Management) has been given a requirement to provide GIS map visuals for their BSM products.

Problem:

There are IBM legal issues with using Google maps API as well as an alliance with ESRI Corporation for Map services. Additionally there are huge node systems at hand to solve such as 40 millions meters in France to manage and straight forward Google map API usage will not solve the problem. This was a very tedious projects as the ESRI API was rewritten three times since ESRI changed their API during this time. In addition, the Yahoo API for maps was explored but tossed since Yahoo did not want to be in the enterprise business.

Result:

A vender agnostic JavaScript API was created which allows for Google, MS Earth (Bing) and ESRI to be plugged in as a choice by the customer thereby alleviating IBM from legal issues. The scale enables 1 million devices to be addressed at one time as documented as well as enable 40 million. Additionally it was discovered that we can plug in an Open Source Map System for additional customer security for owned map information and no license fee. This project created an additionally IBM Services revenue area.

Work:

The project was completed by myself and one other person in France to leverage their Large scale utilities work.

Document: SmartMapDesign.pdf

Tivoli Micro Install Process (MIP 2008)

Storyline:

IBM Tivoli provides agents to report information for resources that are managed by the IBM Tivoli infrastructure. For example, one bank has 10,000 agents that are deployed. The problem we have is the deployment size at 70mb due to the universal install system. One customer had 200 agents and complained the updating process for agents took a business week to install and verify.

Result:

The idea was to use the pkzip library and create a data image which is then concatenated to an executable that is sent to the deployed machine. The executable would run and unpack itself which would install the agent, additionally the running executable would remove itself in a clever way under the MS system which does not allow this. This executable can be built with selected deployed items using the standard operating command line (copy and cat) under Unix and Microsoft, no special creation program is needed to create the deployment package for artifacts.

Customer:

The anxious 200 agent customer was able to deploy in 30 minutes all agents. The 10,000 agent customer required Tivoli to use this process as the standard.

Document: MIPspec.pdf

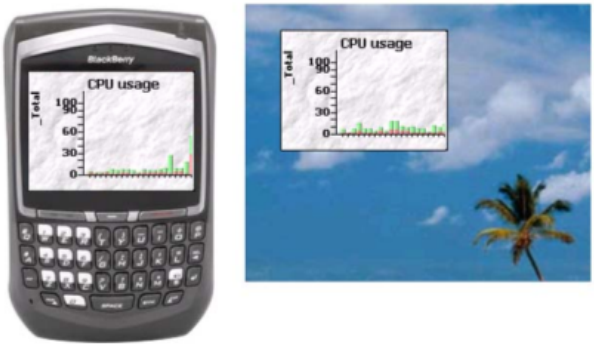
Internal IBM project targets for this incubator:

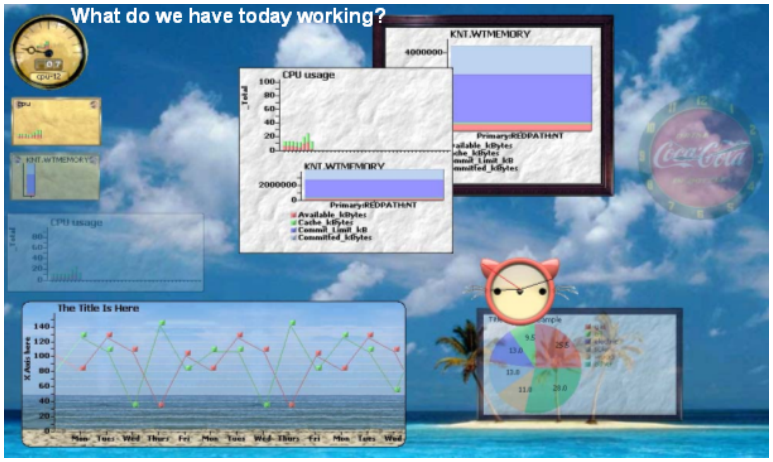
Deutsche Bank micro JVM Install IBM Tivoli

Commerze Bank micro JVM Install IBM Tivoli

WellPoint Corporation micro JVM install IBM Tivoli

The same Widget body on a Blackberry and on the desktop





Mobile Phone Widget Model (Diaphanous 2007):

Storyline:

IBM Tivoli was exploring a means for applications on all mobile phones (Nokia and Blackberry) for disseminating resource and status information. I had an idea (called Diaphanous) whereby a widget could be written that could run on desktop computers as well as most mobile phones without any change. The idea was to use the MIDP platform for mobile phones as the programming model and enable this platform to run on the desktop.

Result:

The IBM J9 Micro Java kernel (JVM) is used by many OEM phone manufacturers. I modified the 700k micro JVM to run under the desktop OS as a shell to run MIDP apps with sophisticated clipping. A complete graphics system was created to enable Tivoli to show sophisticated graphs under the MIDP programming model. Additionally a security model was created in which a simple text properties file could be sent as an attachment in Lotus notes that would materialize a widget on the desktop. This attachment would be no larger than 500 bytes to show a very sophisticated graph of information.

Document: Tivolireadme.pdf

Internal IBM project targets for this incubator:

MIDP for Domino IBM Lotus

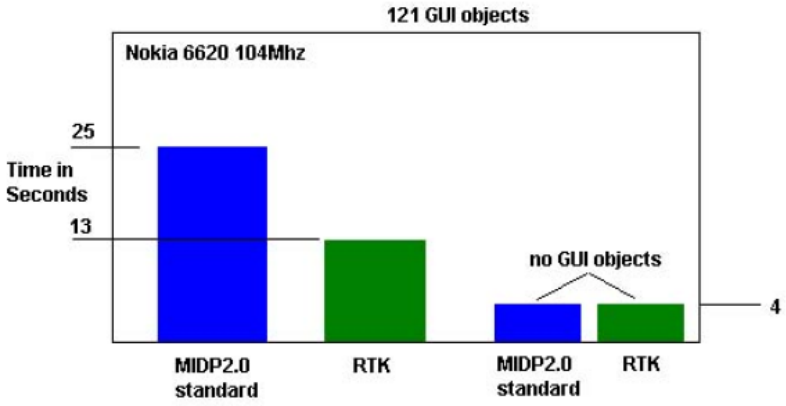
Diaphanous project Santiago Chile IBM UK

Diaphanous project Tivoli TEP product

Diaphanous project Museum IBM Italy/IBM UK

RTK Lotus Traveler project IBM RTP/PHX

Blackberry Tivoli Netcool Product Scorecard



Mobile Phone IBM MIDP toolkit (RTK 2006-2007):

Storyline:

There are a few widget toolkits which run under the standard MIDP platform for mobile phone applications. The MIDP platform is run on 100s of millions of phones. These toolkits produce child like renderings and are proprietary. The performance is not at my standard as well as the size for applications. I convinced my management to let me write a toolkit which will operate on all phones including Blackberry.

Result:

The toolkit RTK was created called Redpath Toolkit but later the initials were stated as Rightsize Toolkit for internal politics. Very well rendered Mobile phone applications could be written starting at 70k in size on phones at 102mhx with great performance.

Document: rtk.ppt.pdf and rtk1.10.pdf

Internal IBM project targets for this incubator:

RTK project SoftBank IBM UK

RTK for JPMorgan project IBM UK

RTK for Sametime Mobile project IBM Austin

RTK for Voice Project

RTK for Restaurant Project

Lotus Machine Translation Server (2005):

Storyline:

My directive was to organize the five IBM Research Labs (Korea, Japan, China, Germany, and USA) to create a single common machine translation architecture and product from 18 years of research for the IBM Lotus Division. These five labs had different approaches and distrust for each other's intellectual property. Additionally, IBM Lotus Division had Lotus Core Services creating a server for this machine translation service.

Problem:

IBM Core Services produced an inferior server which had an 8% failure rate in a closed system with a very low transaction rate. This had to be proven as they were very well versed in arguing the standard when there was nothing to compare to.

Result:

I notified my direct manager (IBM Director) that we did not have a plan B and failure was eminent. This failure would be a bad mark for our consistent success. The directive was to create a plan B and that was to write a competing server in three weeks, we had nothing to lose. The competing server was thoroughly tested with 8 millions transactions. This competing server had a throughput of 42,130 transactions per minute compared to 154 for the Lotus Core Services. A document was created with charts and test cases (lotustest.doc). The IBM Lotus Core Service escalated the issue and we simply presented the evidence to IBM Vice President. The IBM VP chose our server as the product model.

Document: lotustest.pdf