UPGRADE PLAN FOR 3 LEGACY APPLICATIONS AT HEWLETT-PACKARD

Paul Milleson

Table of Contents

Abstract

At Hewlett-Packard, we need to upgrade three highly critical global applications (3-tier applications), which were written in-house to meet our specific needs. These upgrades involve purchasing new hardware and software for both the front-end (user screens) and the back-end (database) systems. When there is this much extensive upgrading to perform on a heavily used, world-wide application (three in our case) there is much concern by the stakeholders (business owners) about potential down-time. The most salient concepts of an upgrade plan that alleviates down-time fears are:

1. Have a completely separate test environment that won't affect production operations in any way.
2. Complete rigorous testing operations by those same users that already use the applications in the most extreme ways and allow them to form their own test plans.
3. Pick tester leads who can verify bona fide errors before submitting them for repair and can verify that an error is completely repaired after a developer has stated that it's repaired.
4. Use a Wiki (a website that stores text for all to see) or some other form of bug tracking software to log errors and repairs, all of which bring visibility to the presence of an error and finality to the close of an error.
5. Make sure that the primary stakeholder for a given region has full confidence in the test lead chosen to represent his region and have him sign a document saying so.
6. Collect a signed document or at least a confirmation email from each stakeholder before attempting to schedule the go-live date.

7. Make sure to have a stated roll-back plan before the go-live date.
8. Make sure that you can get all the test leads to perform a quick go/no-go test just after roll-out to establish that there is basic functionality, before beginning deep level testing.

Besides those 8 fear alleviating concepts, this project plan demonstrates the following:

- Keep management and stakeholders happy by being vigilant to inform them as to what will happen in the project and when.
- Consolidate procurement activities into the early parts of the project.
- Establish testing policies that will ensure a "no surprises" end to the project.
- In order to cut the risk in half when performing a 3-tier application upgrade, always break the project into 2 separate release phases – the database phase (hardware and software) and the application phase (hardware and software).
- When possible, process key project deliverables in parallel to enhance the time-line.

Introduction

At Hewlett-Packard we have three highly critical global applications (3-tier applications), which all run on a single Coldfusion server with a single Microsoft SQL back-end server. One application deals with compiling the code, which is typically in the C language and which is compiled on build servers controlled by a Linux server. Another application performs a similar task, only the destination of the compiled code is to software engineers at licensee companies, world-wide, that are either testing our software or making modifications to it. The third application tracks which engineers (world-wide) are working on which software projects and also needs to be available for manager input at all times.

All three applications were written in-house to meet our specific needs. Since we are both a Hardware and a Software company, the internal distribution of the latest compiled code to our development engineers all over the world is of primary importance. Surprisingly, the bottleneck is not with the build servers or the Linux server, mentioned in the first case, it's with the Coldfusion server and its back-end Database server. This is because those front-facing servers are quite old and so is their software versions. Additionally, all three Apps are running on those same servers.

Each application is written in Coldfusion 5 and hosted (using IIS) to branches of our company worldwide by a 9-year old Windows server. Additionally, they all maintain their data on an equally old Microsoft SQL Server. Coldfusion 5 reached end-of-life (no longer supported) several years back and even the next version, which was 6.1 has reached its end-of-life. Due to its advanced age, the hardware on the SQL Server locks up periodically requiring repeated system reboots. This is unacceptable behavior for these three applications, because the company depends on their continuous availability.

With all this background information, four things should be readily apparent:

1. We are in a dangerous situation as a company because any prolonged down-time by any of these three crucial applications would cost the company significant time and money.
2. Somehow the failing SQL server must be replaced before it stops altogether.
3. Since the software versions that these applications were created with are no longer supported by the companies that are responsible for them, it would be prudent to upgrade to the current versions that are supported, at the same time the hardware is replaced. This

adds increased risk to the whole replacement operation, since it is unknown, what effect, the new versions of software will have on the current running versions of the three applications.

4. There is no luxury to alleviate these circumstances by having significant periods of server downtime. The handling of these issues must be done in such a way, as to have almost no interruption of service, for any of the three applications.

1 Inform and Discuss Proposal with Stakeholders

(1.1 Prepare for introductory Stakeholder meeting)

To begin the project, it is necessary to have a meeting with all the stakeholders (business units) within the company. In this meeting you will introduce all of the reasons why an upgrade to the servers which host Hewlett-Packard's 3 main applications is of utmost importance. The display of the facts will be accomplished using a series of PowerPoint slides. It would be useful (and more convincing) to have a few slides that go over how many users "hit" each of these applications each day. It would also be useful to display where the hits are coming from to support the fact that these applications are truly global. If there is information about how often these servers crash due to their age, this would be important to mention. Make a slide that will show the benefit of the upgrade in time and dollars and in increased reliability. Increased reliability is a big benefit because time lost due to the three main applications being down is very costly. Another time saving benefit to underscore is the fact that the new servers will not only run much faster, but the new versions of the database software and the application software are also supposed to run more efficiently. Of course, it should also be shown that Coldfusion 5 reached end-of-life (no longer supported) several years back and even the next version, which was 6.1 reached end-of-life, and that the

SQL version has also reached its end-of-life. After all of this information is placed on slides, it would be prudent to make a short half-page overview document. Also make a stakeholder sign-off sheet to display during the meeting. Now create a meeting using Microsoft Outlook and take into consideration global time zones and openings in the schedules of the stakeholders. Attach the overview document to this invite and send it off.

(1.2 Stakeholder meeting)

On the day of the meeting, gather up your favorite laptop with the slides contained therein, and proceed to the meeting room. Make sure that this room is a fully-equipped Audio/Video room so that you can interact face to face with all of the stakeholders world-wide. Present your slides in a confident manner because it is very important to fully sell the stakeholders on the need for the upgrade. This is not a "nice to have" upgrade, this is a "must have" upgrade. Be sure to make that point come across clearly. Introduce the concept of world-wide testers and test leads. Make sure to stress how this project must be a collaborative effort between everyone in order to come off successfully. Mention the relationship between the testers, the test leads, and the development team and how their respective responsibilities mesh during the error discovery and repair process. Describe the Wiki and how the leads will use it to track the incidence and the repair of errors, in conjunction with the development team. Stress the importance of each stakeholder getting to know and trust the lead tester for his region and how, in the end, they will be required to sign-off on the completion of a test phase for their region, based on the word of this test lead. Introduce the Stakeholder sign-off sheet at this point, and ask for any questions. After the questions have been answered to the satisfaction of the stakeholders, adjourn the meeting.

(1.3 Stakeholder meeting follow-up)

Mail out a meeting recap along with any action items that were divvied out during the meeting. Make sure to address specific concerns vocalized by certain stakeholders during the meeting in the form of direct emails to them, but cc all the rest of the stakeholders because frequently some of them had the same concerns but just didn't mention them. Continue to pitch the fact that this upgrade is going to benefit everyone, in the form of time savings and reliability.

2 Acquire and Inform Test Participants

(2.1 Prepare for Tester meeting)

The tester meeting is very crucial because it is within this meeting that the actual testers will be addressed and their relationship with the test leads will be described. Before this meeting, it will be necessary to statistically determine who the most frequent or extensive users are on each of the 3 main applications on a per region basis. It is from these statistics that the best tester and lead tester candidates can be determined. It is desirable to ask a person if he would be willing to be a tester, but in some cases, the testers can just be assigned by their managers. The same goes for test leads. The reason to have test leads is because there needs to be a level of verification for errors. Sometimes a tester thinks he has found an error in the application where one doesn't really exist, or he has introduced an error through the improper use of the application. This is why it's a good idea to have a lead person who can substantiate the error, and then if it is a bona fide error, enter it into the error tracking Wiki. This also provides a reliable point of contact for the person in the development team to interact with and to notify regarding error repair status.

Some Wiki's have built in functionality to handle notifications in both directions. This means they notify the repair team when a bug (error) has been entered, and they notify the test lead as error repair status is updated. After a developer has signified that an error has been fixed, the lead tester should verify this, and mark "completed" into the Wiki. When all known errors for an application have been "closed", a "testing complete" note should be entered. When all three applications in a region have been marked "testing complete", the regional stakeholder should be notified. He should then complete a "testing complete" form for his region and send it to the project manager.

Since very few errors are region-specific, (and a bug fix one place, fixes the bug everywhere), this means that testing is slow at first and then speeds up, and then slows down again. If possible, encourage regions to test different areas of the applications than other regions. In this way, when one region gets stuck waiting for a bug fix, the other regions can continue testing until they hit bugs. As this process continues, the development team may get an avalanche of bugs early on since the queue of bugs is being fed from all directions. This is why everyone in the web application development team (the team who built these apps) should be involved in bug fixes, rather than new development, if at all possible. Once the bulk of bugs have been fixed, the rate of the discovery of additional bugs should slow down to a more tolerable level and testing will accelerate.

Prepare PowerPoint slides to illustrate all of the points mentioned above. Also prepare an overview sheet that summarizes a lot of this information. Now create a meeting using Microsoft Outlook and take into consideration global time zones and openings in the

schedules of the potential testers. Attach the overview document to this invite and send it off.

(2.2 Tester Meeting)

On the day of the meeting, gather up your favorite laptop with the slides contained therein, and proceed to the meeting room. Make sure that this room is a fully-equipped Audio/Video room so that you can interact face to face with most of the potential testers world-wide. Present your slides in a confident manner because it is very important to fully sell the testers on the need for the upgrade. This is not a "nice to have" upgrade, this is a "must have" upgrade. Be sure to make that point come across clearly. Introduce the concept of world-wide testers and test leads. Make sure to stress how this project must be a collaborative effort between everyone in order to come off successfully. Mention the relationship between the testers, the test leads, and the development team and how their respective responsibilities mesh during the error discovery and repair process. Describe the Wiki and how the leads will use it to track the incidence and the repair of errors, in conjunction with the development team. Get people to sign up as testers. It is important to have several testers for each region, so they can continue testing if someone calls in sick on a crucial testing day. The test leads need to be very reliable and responsible because they are crucial to the testing process. Introduce the concept of a Stakeholder and mention the Stakeholder sign-off sheet and all that needs to be accomplished before it can be signed. Ask for any questions. After the questions have been answered to the satisfaction of the testers, adjourn the meeting.

(2.3 Tester meeting follow-up)

Create an email for all the meeting attendees as well as the ones that failed to show up. Provide a meeting recap, go over the action items and compile a list of the testers and test leads that volunteered during the meeting. If it was not possible to get a complete list of testers and test leads from all the regions, then mention that and ask for more volunteers. Be sure to mention that this upgrade is going to benefit everyone in the form of increased speed and reliability.

3 Acquire Infrastructure

(3.1 Research Hardware)

There are many considerations to take into account for the 4 new servers that need to be ordered for this 3-tier application and its backup set. The software itself calls for specific hardware minimums. It's always a good idea to exceed those minimums by a significant amount.

In a 3-tier application, the database server does most of the heavy lifting and thus needs to be the most robust server. The application server on the other hand can be significantly less robust because it spends a lot of time waiting for the database server to catch up with database calls that have stacked up. If one must cut costs, do so on this server.

One way to beef up the database server is to not only have at least 4 processors on the motherboard but also to go with 64-bit processors (Quad-Core AMD, Intel, or otherwise). The other obvious thing to do to enhance performance is to provide lots of RAM memory in this system. (64gb is about right)

The application server needs a minimum of (4) 64-bit CPU's on the motherboard and about 64gb of RAM as well.

The fact that we are upgrading the version of the supporting software behind a set of production applications to a higher level means that our old development environment is no longer useful for development. This necessitates the purchase of a duplicate set of servers and all the software that goes onto them, to emulate the production environment. Then and only then can the developers continue to write code that will run in production, and also have the means to duplicate production issues that need to be solved "off-line" as it were.

So, we need to find vendors who can sell us these servers and get price and availability on them. We also need to concern ourselves with what kinds of warranties and maintenance plans are included with these servers.

(3.2 Purchase Hardware)

Once we acquire the best quote on the afore mentioned hardware, we place the order. It now contains: (2) 64-bit servers (application) and (2) 64-bit servers (database) all with 64gb of ram. These machines are from one of the big 3 computer manufacturers, so the lead-time for getting them is short.

(3.3 Research Software)

In order to put these (2) 3-tier applications together, we will need (4) 64-bit O.S. Copies, (2) for Production and (2) for Dev, (2) Microsoft SQL 2012 and (2) ColdFusion 11 copies. It is always a good idea to check to see if one can take advantage of an upgrade offer by a software manufacturer rather than paying full price. (Assuming you have the former version

of the software and can prove it.) It's also prudent to note that some software companies charge a licensing fee for each CPU in your system. If the company is a Microsoft Partner or pays for a Microsoft Subscription Service, there may not be a charge for the operating systems (o.s.) or even the copy of Microsoft SQL 2012. We need to find vendors who can sell us these software packages and get price and availability on them. We also need to concern ourselves with what kinds of warranties and maintenance plans are included with these software packages.

(3.4 Purchase Software)

Once we acquire the best quote on the afore mentioned software, we place the order. It now contains: (4) 64-bit O.S. Copies, (2) for Prod and (2) for Dev, (2) Microsoft SQL 2012 and (2) ColdFusion 11 copies. These software packages are each sole-sourced to their respective software manufacturers, and the lead-time for getting them is short. (They can also be purchased as an online download for slightly less money)

4 Implementation

(4.1 Prepare for a meeting of hardware and developer departments)

As before, prepare some PowerPoint slides to demonstrate what's going on as far as how the hardware and software is going to change. Display the various interactions between the developers and the test leads. Describe on a slide, how the wiki will be used and the role of the stakeholders. For the hardware guys, describe what kinds of hardware will be coming in and what kinds of software will need to be loaded onto it. Show the differences between the phase 1 (database) and phase 2 (application) upgrades. Show how, for a time, the development group will have access to both of the new ColdFusion 11 servers (once they are

build and loaded by the hardware group) and how they can then copy over the production code and see what kinds of issues will crop up when old ColdFusion 5 code runs on ColdFusion 11. Show that this phase 2 kind of testing and repair will be much more extensive than just the phase 1 database upgrade kinds of repairs, so they should try to get a serious jump on the problems – early on. Show that one of the nice things about having the two CF8 servers running (even after it's running in world-wide test mode) will be the fact that teams of developers can work on problems in the code right after they are discovered during WW test, without needing to have the testers stop and wait for a fix. Create an overview document to highlight the whole scenario.

Create a meeting using Microsoft Outlook and take into consideration global time zones and openings in the schedules of the hardware and development teams. Attach the overview document to this invite and send it off.

(4.2 Carry out the hardware and developer department meeting)

Present the PowerPoint slides and summarize all of the upcoming events that these two groups will be heavily involved with. Mention that since the hardware and software is already on order, things should start coming in any day now. The developers should be eager to find out how well the current production code plays in the new ColdFusion 11 environment, because they have been requesting this upgrade for a long time. Bring up a discussion of how the developers anticipate testing will go during each of the phases. Talk about the importance of playing nicely with the testers and test leads even though working with them will be frustrating at times. Reinforce that while the phase 1 testing is going on, the development team will have access to both of the new ColdFusion 11 servers all by

themselves and so this is a very good time to attempt to find as many bugs as possible. Because phase 2 is an application middle-ware upgrade, (ColdFusion is a middle-ware product) the errors will be consistent and predictable. This is because the changes to functions, made in the new version, that are causing errors with the old code, will fail wherever that old code tries to do the same thing elsewhere, throughout the entire application. One might ask why the writers of a new version of something, would code it in such a way as to break functionality? The answer is simple. Sometimes, in an effort to make a function do more things, or to perform in a more standard or logical way, it becomes necessary to change the way the function is used. When a change is made to how a function is used, then everywhere that function is used in the old way, the application will break. Once it's determined how to use that function in the new version without it breaking, it's easy to use search tools to locate this function throughout the application and change the old code to use the function in the new way. Ask for any questions. After the questions have been answered to the satisfaction of the hardware and development groups, adjourn the meeting.

(4.3 Hardware and Development Team follow-up)

Create an email for all the meeting attendees as well as the ones that failed to show up. Provide a meeting recap and go over the action items and display a list of test leads by region so that the development team can start becoming acquainted with them. Be sure to mention that this upgrade is going to be a big team effort and require cooperation between everyone.

(4.4 Receive Hardware - 4 Systems)

As each system arrives, the hardware group will log its arrival

(4.5 Load Hardware - 4 Systems)

The hardware group will then add the RAM and Video Cards to each system. They will then load the appropriate operating system as follows:

- The CF Systems will each receive Windows Server 2012 R2 Enterprise x64 Edition
- The SQL systems will each receive Windows Server 2012 R2 Enterprise x64 Ed

They will then test each machine's operating system for correct RAM and CPU recognition after boot up and then load the appropriate additional software as follows:

- The CF systems will each also receive ColdFusion 11 Standard Edition (Upgrade Version) - (referred to as APP-1 and APP-2)
- The SQL systems will each also receive Microsoft SQL 2012 Enterprise Edition (64-bit) (referred to as DATA-1 and DATA-2)

Then test basic functionality of each database system and then add the server names to the DNS and then copy the production database over to DATA-1 and DATA-2

Test the basic functionality of each ColdFusion system and **c**opy the production ColdFusion 5 code over to APP-1 and APP-2 and also add these server names to the DNS

(4.6 Modify the current Staging Environment for testing of the new database)

Attach developments' ColdFusion 5.0 staging server to DATA-1 via the creation of a new DSN (Data Source Name) in the ColdFusion Administrator

(4.7 Create 2 new test environments for the development team using APP-1, APP-2 and DATA-2)

Attach both of the new ColdFusion 11 servers to DATA-2 via the creation of a new DSN (Data Source Name) in the ColdFusion Administrator on each one. Then configure IIs (Internet Information Server) on APP-1 and APP-2 to render the ColdFusion content folders as APP1 and APP2.

5 Verification – Phase 1

(5.1 Summon the testers via email and provide the link to the newly configured Staging server)

Create the email just mentioned and include a brief paragraph stating that testers should work with the application on the staging server, just as if they are performing actual work in the production versions of the 3 main applications. The nice thing about working on the staging server is that none of the data matters, and so the testers can make up any data they'd like when testing. Also, list the test leads by region, so that each tester knows who his test lead is. Also provide a link to the wiki because some of the people on this email are test leads.

(5.2 The test leads are informed about errors found by the testers)

When an error is found by a tester, he is supposed to demonstrate the error to the test lead. The test lead is then responsible for determining if the error is a bona fide error or not. If it

is, then they insert it into the bug tracking wiki. If we're using a bug tracking wiki like Bugzilla, at this point, the wiki notifies the dev team that an error has been found. (this cycle repeats until there are no more errors) The dev team fixes the error and logs the fix into the wiki and the wiki notifies the test lead that the error is fixed.

(5.3 The test lead verifies that the error is fixed and closes the error in the wiki)
As mentioned, it is the responsibility of the test lead to close the error in the wiki after proving beyond a reasonable doubt that the error is in fact, fixed. This is a safe guard because if an error fix is accomplished by a developer and also closed by the same developer, he has a built in bias to signify that the error is fixed even if it is only partially fixed or not fixed at all. Also, there is a small possibility that the path that he follows through the application, to reproduce the error, may not be the same path that the tester used to find the error, and in fact may hide the error. This is a common misunderstanding in code repair because the way the coder envisions how a template will be used, is not always the actual way the user works with the page.

From the other perspective, if the error fix verification were to be completed by the original tester who found the error, again, this person is not completely impartial, and they may maintain that the error is still occurring even when it's not. They might simply be using the application in an improper way. This scenario is even more likely, if the error never had to be verified by the test lead.

(5.4 The last error is fixed in the wiki, for a given region.)
When the last error is fixed in the wiki, for a given region, the test lead informs the stakeholder for that region that testing is now complete and error free. This is where there

needs to be a lot of trust between the regional stakeholder and the test lead. He's not going to want to sign anything if he thinks there are still errors in any of the three applications because these errors could cause down-time and a serious loss of productivity.

(5.5 The stakeholder for that region signs the approval sheet for this region.)

This is a very crucial and significant point in time. This is the culmination of months of work. Since this is only one of several regional stakeholders, the testing is not completely final, but as far as his region is concerned, the new database server works. At this point he mails (or most likely emails) a signed approval sheet to the project manager.

6 Final Tasks – Phase 1

(6.1 After the project manager has collected the approval sheets from all regions.)

At this point the project manager has received approval sheets from all of the regions. He is elated and beside himself with joy. He proceeds to email out suggested go-live dates to all stakeholders. Once a consensus has been achieved for a go-live date, the date is put on calendar and confirmed with the hardware group and the development group.

(6.2 On the go-live date, the development team copies the current production data to DATA-1)

Because the "big three" applications are global, choosing a go-live date is quite difficult. There are only a few safe times to swap in a new database server. Saturdays are not too bad for this kind of thing (if it's not typical for lots of people to work on Saturday), as well as Sunday mornings. At 4pm on Sunday, Japan begins their 8am Monday, so the database upgrade should be complete before then. The other difficulty is that we plan to have a quick – unanimous test, world-wide, right at the time that the new database is connected. Getting all of the test leads online at precisely the same time will require a few very specific emails.

Another tool that's invaluable on this day is an instant messaging tool such as Yahoo Messenger. The idea here is that test leads, hardware techs, network admins, development engineers and of course the project manager, should all be sending each other text messages as early as one hour before the swap to the new database. Yahoo and others offer the ability to create an instant chat room, which is even more helpful for keeping track of who's ready to test and who's nowhere to be found. When one or more of the test leads isn't showing up in the chat, it's time for someone to call them on the phone. The other test leads are good candidates for that.

Since the new database needs a perfectly fresh copy of the production data (for all three apps) in order to resume operations where the old database left off, it is necessary for there to be a point in time where there is absolutely no activity on any of the 3 main apps. The way this is accomplished is simply to turn off all web access to the apps by stopping the IIs service on the application server. Since everyone, world-wide, has been alerted as to when this shutdown will occur, it should be done right on schedule. Once the web access has been cut-off, it's time to begin backing up the production databases for all 3 apps. This operation usually doesn't take hours, but rather minutes (like 10 – 30 depending on the amount of data in each one.) The backup command can be exercised against each database all at the same time, if desired. The backed-up data is placed into a shared folder on the new database server. Once the data is there, a restore command can be exercised on each backup file, and the new database will acquire a fresh copy of each one.

Once the data is on the new database server, it's time to disconnect the old database server and reconnect the new one. First, go into the ColdFusion administrator and create a new

DSN (Data Source Name) for each restored database. Go into the app_globals.cfm file and change the values of the DSN pointers from the old DSN's to the new ones. Now the application will perform all database operations on the new database server.

The test leads, all test the applications in a designated set of crucial areas (using the Roll-out Critical Test Set) to make sure that they are all working correctly and then they email the project manager when satisfied. (or tell him in the chat room.) If for some reason, the 3 apps or even one of them, starts manifesting errors, the roll-out is scrapped and it becomes time for a roll-back. Go into the app_globals.cfm file and change the values of the DSN pointers from the new DSN's back to the old ones and the 3 apps are back to the database server they had before. Turn on all web access to the apps by starting the IIs service on the application server. Since the roll-out, in this scenario was a failure, and the system is as before, a global email is sent out declaring that the three main apps are back online again and can now be used, but they won't have any improved performance because the roll-out of the new database server was scrapped and rolled back to the old one.

Had the roll-out been error-free, we would have left the values of the DSN variables, pointing to the new databases on the new database server and the email would have declared the upgrade a success, and that the system is now back on-line for all to use.

7 Verification – Phase 2

(7.1 Summon the testers via email and provide the link to APP-1)

While phase 1 was being accomplished, the dev team was performing tests on the APP-1 and the APP-2 servers to start to address the errors that came up when the old CF5 code was run

on a CF8 server. At this point, they've eliminated all the errors that they can find, and are now ready to start the world-wide testing.

Create an email and include a brief paragraph stating that testers should now begin the testing of the application on the APP-1 server. They should use this server in the same manner that they perform actual work in the production versions of the 3 main applications. On the APP-1 server, as with the staging server, none of the data matters, and so the testers can make up any data they'd like when testing. By now, each tester knows who his test lead is. Also provide a link to the wiki because some of the people on this email are test leads.

(7.2 The test leads are informed about errors found by the testers)

When an error is found by a tester, he is supposed to demonstrate the error to the test lead. The test lead is then responsible for determining if the error is a bona fide error or not. If it is, then they insert it into the bug tracking wiki. If we're using a bug tracking wiki like Bugzilla, at this point, the wiki notifies the dev team that an error has been found. This cycle repeats until there are no more errors. The dev team fixes the error and logs the fix into the wiki and the wiki notifies the test lead that the error is fixed.

(7.3 The test lead verifies that the error is fixed and closes the error in the wiki)

As mentioned, it is the responsibility of the test lead to close the error in the wiki after proving beyond a reasonable doubt that the error is in fact, fixed. This is a safe guard, as mentioned in phase 1.

(7.4 The last error is fixed in the wiki, for a given region.)

When the last error is fixed in the wiki, for a given region, the test lead informs the stakeholder for that region that testing is now complete and error free. This is where there

needs to be a lot of trust between the regional stakeholder and the test lead. He's not going to want to sign anything if he thinks there are still errors in any of the three applications because these errors could cause down-time and a serious loss of productivity.

(7.5 The stakeholder for that region signs the approval sheet for this region.)

The conversion to version 8 of ColdFusion was the harder of the two tasks (the other being the conversion to the new database) because ColdFusion is already supposed to be compliant with MS SQL 2012, and furthermore, if the database queries all follow standard ANSI SQL format, they should be compatible with any database. So, it's significant that a region has proven that the development team seems to have found all the incompatibilities between version 5 of ColdFusion and version 11, as applied to the code in the 3 main apps. This is again, the culmination of months of fine work. Since this is only one of several regional stakeholders, the testing is not completely final, but as far as his region is concerned, the new application server works. At this point he mails (or most likely emails) a signed approval sheet to the project manager.

8 Final Tasks – Phase 2

(8.1 After the project manager has collected the approval sheets from all regions.)

At this point the project manager has received approval sheets from all of the regions. He is even more elated than before because this phase was even harder than the previous one and it is now complete. He proceeds to email out suggested go-live dates to all stakeholders.

Once a consensus has been achieved for a go-live date, the date is put on calendar and confirmed with the hardware group and the development group.

(8.2 On the go-live date, the dev team changes the DSN variables on APP-1 to point to DATA-1)

Since DATA-1 has been the actual production back-end database since it was approved at the end of the phase 1 roll-out, it contains completely current data. Switching APP-1 (which has just passed all the testing) to DATA-1 is as simple as changing the values of those 3 DSN variables in the app_global.cfm file and creating those 3 DSN's with the ColdFusion Administrator on APP-1.

As mentioned in phase 1, finding a usable go-live date, that doesn't affect too many people, is tricky, but one has been set. Also mentioned before, tools like Yahoo Chat will be used to get all regions ready for the simultaneous "go/no go" test. We will get the test leads, hardware techs, network admins, development engineers and of course the project manager, all sending each other messages in the chat room as early as one hour before the swap to the new application server. If anyone's missing, we will have a test lead give them a call and get them on board.

Since everyone, world-wide, has been alerted as to when this shutdown will occur, it should be done right on schedule and only the test leads should be using the system during the shutdown. This is really no big deal, because the rest of the world (other than the testers and test leads) has no idea that there is an APP-1 server, nor do they have the URL to get to it. Just to make things obvious, the current application server is taken off-line (taken off the network by pulling its Ethernet cable) at the moment of the published down-time.

When shutdown time occurs, the test leads all begin testing the applications on APP-1 in a designated set of crucial areas (using the Roll-out Critical Test Set) to make sure that they are

all working correctly and then they email the project manager when satisfied. (or tell him in the chat room) Once the project manager knows that all of the regions have completed their tests successfully, it is time to make APP-1 useable with the same URL address. In the DNS (the Domain Name System which controls all server names on a corporate network) have the network administrator change the IP address associated with the former main application server to the IP address of APP-1 and change the name of APP-1 to the same name the former application server had. This will cause the original URL that everyone uses to connect to the three main applications to now connect to this new application server. Next, we need to send out an email that declares the upgrade to be a success, and that the system is now back on-line for all to use.

If for some reason, the 3 apps or even one of them, starts manifesting errors, the roll-out is scrapped and it becomes time for a roll-back. Since the original application server's DSN variables weren't touched, and it is merely disconnecting from the network, bringing it back online, happens instantly when it is plugged back into the network. Since the reassignment of the IP address in the DNS is only performed after all tests were a success, nothing has changed there, and the original URL will still work. Since the roll-out, in this scenario was a failure, and the system is as before, a global email is sent out declaring that the three main apps are back online again and can now be used, but they won't have any improved performance because the roll-out of the new application server was scrapped and rolled back to the old one.

Conclusion

It might be readily obvious after reading all this, that surgically implanting two new servers into critical production environments is not a job for the faint of heart. That being said, it should also be apparent that with careful planning, there are ways to do so with very little risk. In the scenario that I've outlined here, the risk is so small, that even in the event that either of the new systems fail miserably during the roll out, the most time lost would be about an hour. That kind of risk is very tolerable and almost unheard of in the industry. The trick that accomplishes this minor miracle is redundancy. At no time is testing actually performed on the application server in production, except during the final minutes of the roll-out. And that "go/no go" testing is accomplished very quickly, because this is the one place where I tell the testers exactly what to test, with my "Roll-out Critical Test Set". Also, the testers I use at the end are the test leads, because they are the most experienced and familiar with the applications. I hope that this report will someday, be a help to some poor soul who's stuck doing an upgrade as nasty as the one detailed herein.

Roll-out Critical Test Set

1. Test and observe that the main login page works functionally.

2. Test and observe a page that is supposed to contain secure and non-secure hyperlinks (Use the Validation Build page). Enter the page with your secure (test lead) login and verify the presence of secure hyperlinks. Logout and use your non-test lead login to see if the secure hyperlinks disappear.

3. Test a web form and see if it has working validation to prevent submission with improper values. (such as placing letters in number fields) Use the create RFB page for this.

4. Submit this web form and see if it appears to submit successfully and without errors.

5. Verify the presence of this new RFB.

6. Perform a task in which you expect a generated email to come to you and verify that it did indeed come to you. (Use the password reset page for this.)

Infrastructure Drawing