Kenny & Company
Management Consulting

Agile Deployment Strategies for Projects in Productive Systems

Many of our clients are currently engaged in agile transformations of their software development capabilities or in the implementation of agile software development projects. As we assist them in these efforts we are often asked to address their existing paradigm around deployment. Typically we need to achieve three broad deployment objectives in keeping with agile principles.

- Continuous Deployment: More deployments to a Quality Assurance or Test environment (QA) so that users can test continuously.
- Flexibility Between QA and UAT: Ability for a subset of code from QA to move to a User Acceptance Testing environment (UAT).
- Hot Patches: Ability to handle hot patches into the Production Environment (Production) for production issues.

Let's create a simple architecture to discuss our recommended deployment strategy. In this architecture, we have three environments: QA, UAT, and Production, and are using the common Eclipse Software Development Kit (SDK).

Much of the ad-hoc usage and "pre-testing" of the system occurs in the QA environment. The UAT environment contains only the code that is going to be tested prior to deployment to Production. Code that is to be deployed to Production must first pass QA "pre-testing" where it gets integrated with other code before a final round of testing in UAT and a final deployment to Production.

We typically recommend the following deployment strategy to enable our clients to achieve the above goals:

Kenny & Company
Management Consulting

About Us

Kenny & Company is a management consulting firm offering Strategy, Operations and Technology services to our clients.

We exist because we love to do the work. After management consulting for 20+ years at some of the largest consulting companies globally, our partners realized that when it comes to consulting, bigger doesn't always mean better.

Instead, we've created a place where our ideas and opinions are grounded in experience, analysis and facts, leading to real problem solving and real solutions – a truly collaborative experience with our clients making their business our business.

We focus on getting the work done and prefer to let our work speak for itself. When we do speak, we don't talk about ourselves, but rather about what we do for our clients. We're proud of the strong character our entire team brings, the high intensity in which we thrive, and above all, doing great work.

What's Inside

Continuous Deployment ..... p 3

Flexibility between QA and UAT..... p 3

Hot Patches ..... p 3

Validating the Strategy – Real World Scenarios ..... p 3

Automated Functional Tests ..... p 4

Conclusion ..... p 4

About the Author ..... p 4

For a complete list of **Kenny** & **Company** publications and information about us, please visit our website www.michaelskenny.com or email us at info@michaelskenny.com.

Continuous Deployment

To meet the first objective of continuous deployment, we recommend that anything checked into Head will be deployed to the QA environment, once a day or preferably nightly. This allows the users to start testing the system at the beginning of the following day and gives them an opportunity to provide timely feedback. This is in keeping with the agile principles of continuous testing and feedback.

Flexibility between QA and UAT

To provide the required flexibility between QA and UAT, we propose creating a branch called UAT. The process here would be that at the end of a Sprint, whatever code we feel comfortable is complete (it has passed QA testing) is merged into the UAT branch. From there, it gets deployed to the UAT environment for testing and then to Production.

Hot Patches

For dealing with Hot Patches, we propose creating a unique branch called Hot Patch. If a production issue were to occur, the coding fix should be done in the Hot Patch branch. It would then be merged to the UAT branch and then deployed to the UAT environment for testing. Once it has passed UAT, it is deployed to Production. The code is then merged to Head to keep all the branches in synch.

There may already be a testing cycle going on in UAT at the time a Hot Patch needed to be tested. In this case we recommend overwriting the UAT environment so only the Hot Patch code is included. This situation is addressed in more detail below.

Validating the Strategy – Real World Scenarios

We provide three real-world scenarios and use our recommended strategy to illustrate what happens and why it should happen that way.

- Normal Operations
- Errors found in UAT
- Hot Patches

Scenario: Normal Process

This scenario is straightforward. Let's say that we are in a four week sprint and six backlog items get checked in and deployed to QA each week. That means there are 24 items altogether. The users have been testing each of these and providing feedback to the development team which has been making the necessary changes and fixes.

However, at the end of the sprint, four items failed testing or failed to meet the requirements. In this case, the 20 items that did pass would be bundled and deployed to the UAT branch. We will assume that all 20 items pass in UAT since they were already tested in QA. There could be times when something passes in QA but fails in UAT, which we will address later. Once it has passed UAT, it is deployed to Production.

Let's continue using this scenario, but in this case, of the 20 items in UAT, only 18 pass and two fail. We can fix the code or remove the code.

If it's something easily recognizable and easily fixable, it may make sense to just fix it. It would be checked into Head and merged into the UAT branch where it would be redeployed into the UAT environment.

If it's something complex or something you don't have time to look into, then you may have to remove the code. The code would have to be removed from the UAT branch by redeploying from Head minus the item that failed.

In either case, whether you fix the problem or remove the problem, you have to redeploy something to the UAT environment and you will have to test everything again in UAT. This is because this is your last opportunity to get things right before going to Production. You have just changed the code and you may not know all the interactions in all the files and there is the potential that something else has failed. So in this scenario, of the two items that failed, let's say that one item we can fix and one item we are unable to fix. For the one item we can fix we will check the fix into Head. Since the other doesn't have a fix, nothing is changed in Head for now. We now create a .war file from Head that contains 19 Backlog items – the original 18 that passed plus the 19th that is the fix. This bundle gets redeployed to the UAT environment and testing can begin again for those 19 items.

Scenario: Errors Found in UAT

As mentioned previously, in the normal operations scenario, there could be times when something passes in QA but fails in UAT. We will use another scenario to discuss why something might fail in UAT that had already passed in QA.

In this scenario, in order to continuously deploy to QA for testing, we have completed six Backlog items per week. Suppose all six of these pass during that initial testing in QA. Any of the subsequent 18 Backlog items that went to UAT in the following weeks might have done something to

break one of the original six items. Since we didn't necessarily retest everything, we run a risk that something is broken when we move to UAT. At least when we do testing in UAT, we will catch this.

Another possible reason for errors in UAT is that the developer may have failed to identify all the files needed in the deployment and these were not moved to UAT. Maybe those files existed in QA because of some other piece of development that got checked in. Suppose this extra development and the relevant files are not ready for UAT so they are staying in QA, then the first item will fail in UAT due to the missing files.

Scenario: Hot Patches

Let's now give a scenario of the Hot Patch process. Let's continue the same scenario where we now have with 19 items being tested in UAT. However, a production issue has come up and a fix needs to go to production as soon as possible. In this case, the fix is coded in the Hot Patch branch and then deployed to the UAT environment to be tested. Once tested, it gets deployed to Production. It then gets merged with the UAT branch and the code in the UAT branch (that now includes the code for the hot patch) gets redeployed to the UAT environment.

In this scenario, we overwrote the UAT environment so only the Hot Patch code was included. This obviously puts a halt to testing and requires testing in UAT start over once all the code is replaced. The reason to do this is that hot patches are very disruptive to normal operating procedures and we also want to be risk averse when moving code to a production environment.

If we fail to start the UAT testing cycle over again, then we are assuming that whatever code passed UAT prior to the hot patch will still be fine subsequent to the hot patch. Remember, the hot patch was only tested by itself. This is a big assumption and one that can be potentially dangerous given that this is the last chance to test something before it goes into Production.

Automated Functional Tests

Finally it is important address the use of Automated Functional Tests (AFTs) in all these scenarios. AFTs are very helpful in avoiding many potential deployment issues. As we saw in the case of Hot Patches, there are situations in which we will have to start a testing cycle over. Also, as we saw in the scenarios of why something fails in UAT but not in QA, there are situations when newly added code may break the code that came before it. For both of these reasons, AFTs help save time and headaches.

By running AFTs, you are saving time when having to start a test cycle over. You are also buying yourself piece of mind that all the code you have put into your Production system already will work once you add new code. So as you work on new development, new AFTs should be written every time it makes sense to do so. It is best to run these scripts in environments where there is code coming together. In this scenario, it makes sense to run them in QA and UAT. These AFT libraries will grow and will definitely save you time and trouble and will lead to improvement in overall quality of your product.

Conclusion

As you can see, this strategy uses the minimum number of code branches an enables the use of the same branches over and over again each sprint. Instead of just one reusable UAT branch one can create a new one each sprint. This is done to provide some sort of marker for each sprint as required.

About the Author

Michael K. Chau is a Consultant with Kenny & Company, a consulting firm committed to changing the way consulting services are delivered and raising the bar for consultants and clients. Mr. Chau is a Certified Scrum Master with over 13 years of consulting experience delivering complex IT solutions for clients across industries including Pharmaceuticals, Healthcare, High-Tech, and Insurance Services. He has led multiple ERP integrations, Business Process Reengineering initiatives, and Mergers and Acquisitions. Mr. Chau holds a degree in Industrial and Systems Engineering from the University of Florida.

This article was first published on www.michaelskenny.com on September 3rd 2010. The views and opinions expressed in this article are provided by Kenny & Company to provide general business information on a particular topic and do not constitute professional advice with respect to your business.

Kenny & Company
Management Consulting

Kenny & **Company** is a management consulting firm offering Strategy, Operations, and Technology services to our clients.

Who We Are

Partner Led

Our Partners are personally committed to our clients and lead every engagement.

Experience, Perspective and Passion

We average over 20 years in professional services and bring tailored approaches to every client engagement.

Focused, Collaborative, High-Impact

We work side-by-side with our clients in highly focused teams to solve complex business problems.

Client First

Our highest priority is our client's professional and personal success. We believe clients should expect more.

Guarantee Our Work

We guarantee our clients complete satisfaction every engagement every time.

Contact Information

Firm Headquarters
Serving San Francisco, Silicon Valley & Los Angeles
1710 South Amphlett Blvd.
Suite 302
San Mateo, CA 94402

Northwest Office
Serving Portland & Seattle
707 SW Washington St.
Suite 925
Portland, OR 97205

For inquiries: info@michaelskenny.com

To see additional publications and learn more about us, please visit our website at: www.michaelskenny.com.

Also, follow us on: