Deployment Planning and Risk Analysis

How do we deploy Waterfall II safely

These are the notes from a meeting held from 4-8pm on April 26th. The primary purposes of this meeting were to examine the risks of deploying Waterfall II and to come to a consensus on an overall approach to deployment that would best manage those risks. A secondary purpose of the meeting was to examine the general risks and dynamics of deployment as part of an ongoing effort to improve our ability to deploy future products successfully.

These notes are an edited version of what appeared on the whiteboards and flipcharts during the meeting. The raw notes, exactly as transcribed, are attached.

Attendees

- Sharon xxx (facilitator and recorder)
- A.C. xxx
- Will xxx
- Craig xxx
- Neal xxx
- Melora xxx
- Satish xxx
- Lisa xxx
- Dave xxx
- James Bach

Contents

Meeting Overview

The major events of the meeting followed the spirit of the original agenda. Many issues were raised and concerns discussed, while Sharon kept us focused on the objectives of the meeting. We accomplished the stated mission of the meeting in the four allotted hours. Specifically:

1) Sharon began by clarifying the purposes of the meeting and presenting an agenda.
2) We brainstormed a list of risk drivers, which are problems and challenges that contribute to deployment risk.
3) We brainstormed a list of "nightmare scenarios", which are serious problems or patterns of problems that could befall us during and after an attempt to deploy a new version of the product.
4) As a way of preparing to consider alternative deployment plans, we brainstormed list of risk mitigation ideas for two of the nightmare scenarios. We noticed that many of those ideas would apply to the other scenarios, as well.
5) We brainstormed and discussed a list of alternative deployment strategies. It quickly became apparent that two most viable choices are staged deployment and full deployment.
6) We examined the benefits, risks, and implementation issues associated with each deployment strategy.
7) We came to a consensus that the full deployment strategy involves significantly less risk to execute than the staged deployment option. We also see how we can execute a full deployment with less risk than we've experienced in past deployments. Among the improvements we intend to implement is a more detailed and reliable rollback plan.
8) We identified next steps and assigned action items to the team.

Risk Drivers of Deployment

The following factors came from our brainstorm. We did not discuss them in much detail, but no driver appears on this list unless it received general assent from the team. The items have been edited into sentence form and reordered by affinity.

- Many complicated changes have been made to the product.
- . We have no rollout plan to explain changes to our customers.
- . We have no coordinated roll-forward plan for deployment.
- . Only one customer is in beta as of 4/26.
- . External beta is too brief and has too few customers.
-
- . There is no rollback plan.
- The rollback plan is untested.
- We don't know our criteria for deciding to rollback.
- The baselining process is not stable, and it doesn't meet security requirements.
- We have many more customers, now, who could be impacted.
- The product is used in more and diverse ways than it used to be.
- The new system has an unknown impact on workarounds currently used by customers.
- Our requirements and design may be inadequate. (We may not have correctly understood user needs and usage patterns).
- We have little experience with TRX.

- We have little experience with ClearCase and ClearQuest.
- We have no baseline of information about the performance of the current product.
- We don't know the performance and reliability of the new system, under load.
- Our performance goals are not specified.
- Our tools for monitoring the reliability of the production system are inadequate.
- Our tools for determining usage patterns are inadequate.
- We have not done a security review of the system.
- Lab machines are not secure.
- We're relying on third-party testing for some important tests.
- People may get burned out.
- Employee turnover may impact our ability to execute the deployment.
- We don't have enough hardware to do a staged release.
- Critical hardware may fail.
- There may be unidentified or unmanaged single points of failure in the system.
- The data migration process is not optimized.
- Our database deployment process is not documented or automated.
- Our application and web deployment process is immature.
- Training for Customer Value is minimal.
- Training for maintenance staff is minimal.
- Dependencies with other projects could interfere with deployment.
- Critical maintenance items may interfere with deployment.
- The Firefly code freeze.
- Continual crunch mode could make us complacent about risks.
- Deferred items are sometimes forgotten (the "black hole").
- Code reviews started late.
- We have no formal freeze process for requirements.
- Important stakeholders (such as QA) are sometimes left out of requirements process.
- Overall, there is little documentation of key processes.
- We have no failover for beta.
- Business requirements seem to change frequently.

Nightmare Scenarios with Mitigation Ideas

At the beginning of the meeting, it was suggested that deployment risks fell into three categories:

- Time to Market
- Customer Satisfaction
- Technical Integrity

Later, we brainstormed some "nightmare scenarios" that expanded upon those three basic deployment risks. (These have been edited for clarity and redundancy. See the appendix for the raw list):

- We can't get through the deployment in the time required.
- There are a large number of different post-deployment critical escalations.
- There is one critical post-deployment problem so serious that it cripples the system.
- There are problems that make the system seem unusable to most customers.
- There is a data corruption problem or some other problem that is not discovered until it does a lot of damage.
- We lose customers' data.
- The system can't handle the user load.
- TRX does something that corrupts the service.
- We experience a security breach.
- We decide to rollback based on a misunderstanding of the problems in the system.
- Our rollback process fails: it takes too long or it loses customer data.
- It takes us too long to fix critical problems.

For two of these scenarios, we listed some ideas for mitigating the risks:

We can't get through the deployment in the time required.

- Create detailed deployment plan.
- Practice the plan.
- Estimate the deployment time as accurately as possible.
- Use additional machines so we can abort the deployment and bring old machines back online in case deployment is blocked.
- Use site unavailable screen.
- Insert checkpoints with entry and exit criteria into the deployment plan.
- Assure 7x24 staff availability.

There are a large number of different post-deployment critical escalations.

- Determine criteria for determining a critical issue.
- Plan in advance what actions we may take and how decisions will be made:
  how to have a smooth running release status meeting
  who can call for a rollback
  whose makes the final decision to rollback
  who should be in the release status meeting
- Have resources available to fix issues.
- Have resources available starting at 6am.
- Hold periodic status meetings to assess situation w/IT&CV&ENG&QA.

- Coordinate release w/CV to deploy at non-peak period.

Deployment Alternatives

We discussed several deployment alternatives and their variations. Eventually, we decided that all of the options were just variations of full deployment and staged deployment:

Option A: Deploy all at once to production

Benefits

- Minimizes time to market.
- No need to maintain parallel code branches.
- Avoids technology enhancements that would be required for staged release.
- Support, development, and testing focus on only one product at a time.
- Reduced operational effort and cost compared to staged release.
- It's the simpler of the two plans—fewer variables to manage.

Problems

- We could lose all our customers if there's a big blowup.
- Load hits the new system all at once.

Implementation

- . Improve the detailed deployment plan.
- . Practice the deployment plan.
- . Develop a detailed rollback plan.
- . Practice the rollback plan.
- . Conduct load testing and define acceptance criteria
- . Have at least one external beta customer. Three would be better.
- . Investigate what would be involved in giving special attention to selected custoers during the transition. . Review the deployment and rollback plans (use notes from 4/26 risk meeting in that review).

Option B: Deploy in stages to subsets of customers

Benefits

- Minimizes impact to customer base in case of trouble.
- Allows management of load ramp up.

Problems

- Requires concurrent testing and maintenance of old system and new system.
- Potential for longer response time to problems.
- Requires more HW.
- Double deployments.
- We'd have to create a second URL to handle parallel systems, and deal with all the problems associated with that.
- QE98 and ECadmin connectivity would have to be figured out.
- We don't have enough people, presently, to handle all the work.
- Would significantly delay other projects.
- The rollback plan would be more complex.
- We may lose customers because of delay in full rollout.
- We aren't fully meeting customer needs under current operational conditions, let alone the conditions of a staged release.

Implementation

- We'd need to update Onyx to capture which released system each user is on.
- 30 days to exercise system and meet criteria. (what criteria?)
- We'd deploy first to about 10% of the customer base, then to everybody.
- We'd focus first on "small guys who have a lot of credit card activity"
- We'd deploy in mid-month
- If we deploy new system only to new customers, then the first stage will need to be longer.

Next Steps

General

- Find out how other companies manage deployment risks.
- Inform test outsource firm that performance tests are no longer optional to complete by GA.

Deployment Plan

Owner:

- A.C. (will know schedule by end of day Thu)

Helpers:

- . Satish
- Melora
- Lisa
- Neal

Rollback Plan

Owner:

- . A.C. (will start Thu)

Helpers:

- . Rod
- Rob
- Jill
- Steve

Load Test

- Create Use Cases
- Create Scripts (outsourced)
- Review load test results for deployment criteria

Appendix: Raw Notes

Deliverable
-Deployment approach for W2
-Globally... steps for review for major releases
Started by brainstorming risks:
Time to Market
Customer Satisfaction
Technical Integrity
risk drivers:
only 1 customer in beta as of 4/26
lots of complicated changes
untested rollback plan
no rollback plan
unknown or short external beta
not fully stable baseline process
-doesn't meet security requirements
lots more customers that could be impacted
more diverse usage patterns
don't know our rollback criteria
little experience TRX
no rollout plan to customers-communication
no coordinated roll-forward plan for production deployment
lack of performance information baseline
lack of knowledge about performance/reliability under load
use of 3rd party testing
employee turnover
inadequate reliability monitoring tools
lack of security review
insufficient hardware for staging
no optimized data migration process
no documented /automated database deployment process
immature application/web deployment process
minimum CV training
minimal maintenance training
little documentation
dependencies with other projects
resource burnout
Lack of experience with ClearCase/ClearQuest
critical maintenance items/911
Firefly code freeze
risk complacency
deferred items forgotten
unmitigated single points of failure
lack of adequate requirements & design
potential hardware failure
no formal freeze on requirements
important stakeholders left out of process
lab machines are exposed
code reviews started late
inadequate tools to determine usage patterns
unknown impact on current workarounds
unspecified performance goals
no beta failover
frequently changing business requirements

Can't get through the deployment in the time required
Large # of critical escalations
One huge critical issue

something happens that makes the system unusable
delayed discovery of critical problems
Very difficult or time consuming to fix critical problems
TRX does something that corrupts the service
False alarm triggering rollback
data loss
Rollback failure (takes too long, data loss)
security breach
failure of system to handle load

Can't get through the deployment in the time required
-create detailed plan
-practice the plan
-estimate deployment time accurately
-use additional machines so we can sotp and bring up old machines in case deploy
ment is blocked.
-utilize site unavailable screen
-the plan should have checkpoints with entry and exit criteria.
-7/24 resource availability

Large # of critical escalations
-Criteria for determining a critical issue
-plan action to take when decision point is reached.
-how to have a smooth running meeting
-who can call for a rollback
-whose decision is it to go/no go?
-who should be in the meeting?
-Have resources available to fix issues
-Have resources available for 6am
-Periodic status meetings to assess situation w/IT&CV&ENG&QA
-calibrate release w/CV to deploy at non-peak period

mitigation master strategies
-deploy everything to production w/rollback plan w/data loss
-deploy everything to production w/rollback plan w/o data loss
      Benefits
      -time to market
      -minimize double maintenance
      -no new product enhancements
      -focused support, development, testing
      -reduced operational efforts/costs

      Issues
-could lose all customers if there's a big blowup

Implementation
      -Improve detailed deployment plan
      -practice deployment plan
      -develop detailed rollback plan
      -practice rollback plan
      -conduct load testing & define acceptance criteria
      -have at least one external beta customer/goal of 3
      -investigate what would be involved in handling selected customers during the
transition
      -review plans (use the meetings notes from 4/26 risk meeting to help with
this)

"An outage is much less of a problem than corrupted data"

-staged deployment to subset of customers
      -what is the duration? What criteria?
      -if only for new customers, then it needs to be longer

      Benefits
      -minimize to customer base
      -allows management of load ramp up

      Implementation
      -30 days to exercise system and meet criteria
      -est. 10% customer base, then deploy to everybody
      -"small guys who have a lot of credit card activity"
      -deploy in mid-month

      Issues
      -double testing
      -double maintenance
      -potential longer time to respond to issues
      -update Onyx to capture which system users are on
      -more HW required
      -double deployments
      -URL issue-- how to handle
      -don't have enough people, presently
      -delays other projects significantly
      -QE98 and ECadmin connectivity
      -more complex rollback
      -may lose customers because of delay in rollout
      -can't meet customer needs under current conditions, let alone under
      conditions of a staged release

mitigation variables
-extend beta w/more customers
-comprehensiveness of the rollback plan

NEXT STEPS
-find out what other companies do
-Inform outsourcer that performance tests are no longer optional to complete by
GA

Deployment Plan - AC (will know schedule by end of day Thu)
Satish
Melora
Lisa
Neal

Rollback Plan - AC (will start Thurs)
Rod
Rob
Jill
Stev

Load Test

Create Use Cases
Create Scripts
Review Load Test Results for deployment criteria