



*Mat Ellis, Founder, Cloudability*

Traditionally, technology projects have been treated as if all of the features, benefits and costs can all be accurately defined right at the start. Lengthy ROI and business model reviews, combined with the waterfall method of project management have managed to reinforce our belief in this lie. All too often we recognize late in the project life cycle that our technology needs have changed and so we scramble to catch up, resulting in late delivery, lower than forecast ROI, cost overruns and, in the worst cases, abandoned projects. The cost of this inefficiency, both measurable, and in terms of lost opportunity, customers, market share, etc., are significant, and getting higher as we come to rely more and more on technology in every area of business.

The cloud finally provides businesses with a level of agility that's finally coming close to matching their need to fast paced change. This is putting the current approach of 'nothing changes' under such strain that we believe companies will be forced to fundamentally change how they manage their IT investment. Already we are seeing some leading edge companies apply radically different financial control and planning processes in order to apply this newfound agility to their business.

Here at Cloudability, in our meetings with customers and partners, it often feels like the primary limiting factor is that the humans are having trouble keeping up with the new ways of working these improvements in technology have made possible. Finance requires, rightly, a more conservative approach to process than many other departments, and this change will be challenging at first while norms and best practices are defined.

To illustrate this, we will examine a project lifecycle as it appeared to Mark and Jill (the CIO and CFO respectively of the fictional company Acme Corp) and then see how the project might have fared on the cloud.

Jill ran a tight ship, and around the end of every summer, each team was expected to have a good idea of next year's spending plans. This year Mark was under a lot of pressure from

customers, salespeople and even his ops team to fund a big upgrade to one of Acme's main products. The new project was code named Pink. Early estimates indicated Project Pink would take a year and cost almost \$5MM.

The product Project Pink was replacing was nearly seven years old now and showing its age. The ops team had done a great job of keeping it going without a big investment, but nothing is for free. Sales had stagnated, and earlier in the year Acme's best sales person had quit. The CEO suspected she was lured away by the prospect of better commissions at a rival, who had a more modern product.

Acme's customers were loyal, but it was getting harder to keep them happy. The system would regularly slow down, and then the Ops guys would be forced to stay up all night to get things unblocked again. Significant improvements were pretty much impossible thanks to the unpredictable impact they might have on the fragile system.

Mark was pretty confident he could persuade the rest of the exec team to fund Project Pink, but didn't relish the months of planning and haggling ahead of him.

The last few months of that year were spent planning and preparing the business case. ROI models were endlessly refined until they became so complex they began to resemble the technical documents. Another project manager was hired to share the load, and a really detailed implementation plan was completed, albeit a little rushed to make the deadline. Hiring requisitions were readied, and vendors put on alert.

Finally, Mark got his budget approved, but slightly reduced. This would mean cutting a few corners but it didn't seem anything too serious at the time to cut back on load testing. When everyone came back to the office after the winter break his team got stuck in. Vendors were picked, equipment ordered, contracts signed and code developed.

By the early summer the project was a few weeks behind schedule. It had taken longer than expected to get all the hardware installed and the bugs worked out at the new data center they had selected. Customers had to look at static screenshots of the new system instead of getting to play with early prototypes.

By the time the preview versions of the new system were in the hands of the customers they were two months behind schedule. While early focus groups had loved the screenshots now they were actually using the new system there were all kinds of real-world workflow issues that urgently needed changes. Another few weeks lost from the schedule.

All these delays ate into the load testing time. The beta test group of customers had waited for a long time to get access, and then waited a bit more while the workflow problems were sorted. This meant that the load tests could only happen at night and over weekends. The big changes the beta group demanded really ought to have been load tested again, but there was the budget nor the time, and so Mark made the decision to take a chance and skip them.

Finally, the big day came to start migrating customers. At first things went smoothly, but pretty soon it became clear that there were some serious scale issues. The new system was a big improvement over the old one, and nobody had accounted for the massive increase in the time customers spent in the system, and the much larger and more complex reports they built as a result.

The migration had to be paused while they figured out what to do. The developers worked overtime trying to optimize the system, but the real changes they needed were going to take months to get right. Ops were run very thin while they supported two high maintenance systems: the old one still had the bulk of Acme's customers on it, but the new one also had a set of very engaged users. Both had many problems to fix.

By the end of the year, the developers had a set of changes that in testing seemed to show a big improvement in performance. It was hard to tell for sure as nobody wanted to experiment on the migrated users, they had been having enough problems these past few months. Mark desperately wanted to add some more hardware to some parts of the system just to be sure, but there just wasn't the budget.

The migration was restarted, and the performance improvements really made a difference. The system was much quicker and after a few months of debugging with the early adopters the rest of the customers had a smoother initial experience.

Finally, by Easter the migration was over. The system was slowing down a little, but it was manageable, and at last the old system could be decommissioned. The overruns were mounting: he had to hire some expensive contractors to temporarily beef out the Ops team, and these guys weren't cheap after 9 months. The data center housing the old system had annual contracts, which Mark was forced to renew when they missed the deadline for decommissioning the old product.

The total cash cost of these overruns was considerable, but paled in comparison to the opportunity costs. The entire technology organization had done nothing else but the migration for 18 months just to get on the same level as their main competitor. In that time their competitor had continued to improve their product, and now the market them as the leader. Some key staff and customers had been lost too.

Now let's rewind, and see how Project Pink might have fared if it had been deployed by a company experienced in using the cloud.

There is still have a significant amount of planning before the project started but the much lower capex requirements reduce the risk significantly, along with the planning requirements.

A commitment to millions of dollars of hardware and software is replaced by a commitment to 6 months of labor and external vendor costs. During this time a much more accurate ROI model can be developed, based on actual data instead of projections.

Mark also doesn't have to wait for contracts to be agreed with hardware, data center and network vendors. His developers can get to work almost straight away and Mark can try several different approaches to find out which combination is better.

In this scenario, the beta group of customers can be given access to the very early mockups almost as soon as the developers have created them. This leads to a much earlier, and importantly, a much more interactive feedback loop with the customers. The workflow problems would have been caught much sooner, way before the rest of the system was built saving development time.

A full load test is rarely possible in a traditional scenario, requiring several times the hardware that you have on hand. But on the cloud the ops team is able to generate much higher and more realistic loads. This not only identifies bottlenecks that before we detected much later but also allows them to plan a more reliable deployment strategy and practice with copies of live customer data.

As a result, the Ops team find the bottlenecks before a single user is migrated, and the developers can much more quickly test their performance upgrades. It's not inconceivable to see the production migration starting in half the time when compared to the traditional approach, and with a much higher chance of success thanks to all the testing Ops did in the load tests.

One thing the extra load testing is unlikely to catch is the changes to user behavior driven by the new system, which resulted in a customer using many more cycles of computer time than before. Again, Mark and Jill's life is much easier on the cloud. Extra hardware can quickly and easily be applied to the problem while Acme staff study the problem and decide what to do, likely a combination of charging extra for some of the new features, suck down some of the extra costs and lean on developers to make the system more efficient.

So far we've focused on the some of the benefits inherent in building Project Pink on the cloud, but what are the new risks and costs?

The first risk is of over spending. If a developer fires up half a dozen large sized 'instances' (a common measure in the cloud, roughly analogous to a virtual server) at a cost of \$2/hr each on January 2<sup>nd</sup>, Jill's team won't get the bill until February 15<sup>th</sup> and will likely not get anyone's focused attention until early March. By this time the developer has spent almost \$20,000 without anyone knowing or approving it.

Overruns can be made much worse through security compromises. IT Security is something teams usually focus later in the project cycle, prior to deployment. Hackers can spend an incredible amount in a very short time once they get hold of a compromised set of credentials or a set of servers that have not been locked down. It's not uncommon to see bandwidth charges of \$25,000/day thanks to illegal content being hosted on an insecure server.

Finance teams must put into place new controls to prevent overspending. Changes to control

systems should address:

1. Discovery of cloud buyers
2. Access to cost data for these buyers
3. Frequent or automated review of costs, to detect overruns

These basic financial controls address over spending but a second set of controls is required to avoid waste. A simple example would be a large and expensive computer system that is running at 5% of capacity, which should clearly be replaced by a smaller (and cheaper) implementation.

This process can easily be worked into the regular financial reviews most companies already conduct, usually on a quarterly or annual basis, and goes to the root of a CFO asking executives “do you really need this?” Only now the finance team itself can verify the answers they receive by looking at hard data.

At this point, with engineers purchasing only what they need and the necessary protections against accidental overspending in place, you may think things will return to normal. You would be wrong in thinking so.

The next issue is one of budgeting. The cloud has a major impact in two areas here.

First, before the cloud, if a system was suffering from capacity problems, there were usually only very expensive and unaffordable quick solutions. But now the solution is readily at hand: you can just deploy more equipment!

The flip side of this coin is where costs increase simply because sales have increased. Eventually, you will have to move to viewing costs as unit based versus the traditional method of looking at IT costs as “we spend \$x per year on IT”. If a service is costing \$10/month and is yielding revenues at \$100/month per user, it doesn’t really matter if IT costs this year are over “budget” as sales are too.

By linking IT costs to some business action finance teams not only improve budget accuracy. They regain some leverage in discussions with inefficient IT teams. “We can’t switch servers off without affecting performance” can now be countered with “We can’t leave them on without losing money”.

As most cloud services are account based (i.e. you login with a user ID and provision services), it’s possible to make those align with internal product or project codes. IT managers should be given access to the same data as finance staff so they can focus on unit costs as part of their operational responsibilities.

The last part affecting finance is in strategic planning. The current method allows a company to devise and enforce a strategy for IT investment. A typical system goes like this:

- We’re going to spend x% of sales on IT, which is \$yM
- \$zM has to be spent on mandatory projects (regulatory, legal, operational, etc.)
- Rank discretionary projects by ROI and we’ll spend the rest on those until our total spending reaches \$yM

The cloud makes this kind of top down strategic financial planning much harder: imagine your company builds a hit product like Instagram, with millions of users signing up in almost no time at all. Pre-cloud systems simply couldn’t grow at that kind of speed. Post-cloud your strategic decisions on how much IT to invest in division A or division B can quickly be rendered irrelevant.

It is in this area that a forward thinking CFO can play a very important leadership role in a cloud based organization. Once technology teams are held accountable to unit costs instead of overall budgets they quickly adapt their behavior to make the most efficient use of their budgets. We are already seeing leading cloud users do this, and even to break down unit costs into costs-per-feature, so that high use/high profit features are prioritized over low use/low profit ones.