Test Automation: A Project Management Perspective

Amith Pulla

amith.pulla@intel.com

Abstract

For most QA leads or managers, it's always difficult to get the project manager (PM) to invest in test automation. The term project manager refers to the person who manages the project funding and expenses. Sometimes program manager or project sponsor owns the function of managing the funding and resources. Project management as a function exists in both traditional and agile environments, as project cost management is not in scope for most agile development practices including Scrum. It's important to learn how project managers and stakeholders view test automation and what the impediments in investing in test automation are. Test automation costs are mainly associated with resources, training and tools. For example, a senior program manager once said, "Test automation is great thing to have, but not a great thing to invest in". This paper looks into how project managers view test automation and try to measure its value or ROI (Return on Investment). This paper also offers guidance to QA and Testing teams on how to show the value of test automation in the terms that project managers can easily understand.

Project managers view the project from the classic triple constraint model that mainly includes Scope, Cost (Resources + dollars), Schedule, and Quality management. Project managers don't see a direct correlation between these four project variables with investing in test automation and are usually reluctant to invest in test automation. Investing in test automation has a measureable value on at least 3 out of 4 project attributes and individual metrics can be created for each attribute to show that overall ROI can be in positive territory over time. With the right metrics PMs can be more confident in investing in test automation.

To make a case for investing in the test automation, the QA lead needs to translate investments and ongoing costs for test automation into positive impact on project attributes: Cost, Schedule, Quality and Risk. Once the decision is made to invest in test automation, the QA lead needs to provide an individual metric for each of the four projects' attributes that shows tangible and measurable value to project managers in the terms they most care about.

This paper tries to offer methods, examples and metrics to QA and test leads to show the value of test automation in both traditional and agile environments, with a focus to get the project and program managers to invest and trust in automation efforts.

Biography

Amith Pulla is a QA Manager at Intel Corp, currently working at the Intel site in Hillsboro, Oregon. Over the last decade, Amith has been involved in software testing strategies and processes for applications mainly focused on sales and marketing. Amith has worked extensively on projects involving multiple platforms and complex architectures. Amith also worked on developing test methodologies and techniques to meet the business needs and timelines. As part of his QA lead role, Amith focused on improving and refining QA processes and standards for efficient software testing in agile environment. Amith also worked as a project manager and ScrumMaster in short assignments.

Amith has an M.S. in CIS from the New Jersey Institute of Technology; Amith got his CSTE and PMP certifications in 2006. Amith got his CSM certification in 2012.

1. Introduction

Over the years, with the emergence of test automation tools and growing complexity of software projects, many experts agree that test automation does bring value in terms of better quality and faster delivery, but only small percentage of software development projects actually use test automation today. Even in agile projects, not all projects use test automation as much as the team wants to. The need for test automation and the level of test coverage varies for each application and product, but most QA leads agree that using test automation can cut testing time significantly leading to faster delivery and higher confidence in quality.

Most QA leads and test managers agree that they should use some level of test automation in their project, but lack of funding and priority are some of the biggest obstacles. Implementing test automation can be expensive, there are costs related to additional resources with automation skills, automation tools, test machines, servers and training. In addition, there are ongoing costs of maintaining the automation framework and scripts as long as the application and product is used.

The QA Leads need to work with project managers and stakeholders to make a case for investing in test automation efforts. They need to show the value of test automation to project in the terms that are easily understandable. In the book Experiences of Automation, the authors say "Make your benefits visible to managers. You need to continually sell the benefits of automation." (Dorothy Graham & Mark Fewster, 2012) In this paper we'll discuss how test automation can bring value to the project and some of the ways QA or Test leads can demonstrate the value using metrics and examples.

2. Project Management Constraints

Most project managers, manage the project using the triple constraint model. Triple constraint is the balance of the project's scope, schedule (time) and cost. The traditional triple constraint model now evolved into five variables of project management. Let's look at how test automation impacts each of the project's variables.

2.1 Scope: No impact, we all agree that regardless of amount of testing or test automation of test cases has no impact to product scope or business value delivered. The product scope, features or capabilities are defined by the business or product owners, test automation may impact quality but not scope. Test automation is a task(s) on the project, but not a story or feature that will be delivered to users. Test automation is tool used by testing teams to improve quality and to complete testing faster; it's not visible to end users.

2.2 Cost: Investing in test automation can have an initial spike in project costs. Costs go up with investments in automation resources, training, automation tools and setting up a suitable framework to maintain automation. The initial investment and ongoing resource costs need to be offset over a period of time. The QA lead needs to come up with a cost per each manual regression cycle and automated regression cycle (factoring in cost/investments in test automation), depending on number of regression cycles the project needs, the ROI should move to positive territory at some point. We can calculate the inflection point so that the PM can see the ROI depending the duration of overall project including support phase.

2.3 Schedule: We can clearly show positive impact to this attribute. Regression test cycles will shrink from weeks to days or hours enabling the teams to go to production faster. Again, the QA lead needs to calculate estimated time for manual vs. automated regression cycles per fixed test coverage. The project's release schedule should change as the test automation is built out, clearly showing the value to PM.

2.4 Quality: Testing or test automation won't by itself change the product/application and improve the quality, test automation is a tool to execute functional or system test faster, test-fix-test cycles can be run faster, allowing the project team to find and fix defects faster, leading to improved quality of the application or product given the same schedule. Defects found per each regression cycle can also be a good indicator of the efficiency of test automation. The test automation can enable teams to have multiple regression cycles given the same project schedule, this allows the team to improve overall quality and increase the users' confidence. Test automation can be used with Continuous Integration (CI) framework to run build verification tests after each code check-in.

2.5 Risk: Test automation increases the speed and efficiency of testing, with good test automation coverage, teams can execute more test cases compared to manual testing in the same time. More test coverage means better understanding of application behaviors and defects. By increased test coverage, teams can manage risk effectively.

3. Test Automation and Quality

3.1 Regression Testing

Regression testing, the type of testing that is performed to make sure that previously working functionality is not altered or impacted, is usually done by re-executing existing functional test cases or test suites. The goal of regression testing is to uncover software defects or regressions in existing functional areas of the application or product after new changes are introduced. The new code changes can be related to adding new functionality or features, enhancements, patches or configuration changes.

A typical regression testing cycle will have all or a subset of existing test cases that need to be run depending on the application or platform, the existing test case numbers may range from hundreds to thousands and running them manually can take days or weeks to complete. Sometimes due to the scheduled constraints, teams may take a risk-based approach to regression testing, executing only a portion of test cases, but if the application is large with hundreds of functional, integration and system test cases, running a subset of test cases can take days or weeks.

3.2 Test Automation

Regression testing, running the same test cases or test suites manually in multiple cycles can be laborious and time consuming, depending the on desired quality, this may significantly impact the project schedule. Automating these test cases brings down the time taken to run the regression testing cycles. Manual testing that could have taken weeks or days can be now run in hours.

Once the tests cases or test suites have been automated by the right automation tool and framework, they can be run on-demand and repeatedly faster than manual testing. Running regression testing using test automation is a cost effective method especially if the test cases don't change often and frequent regression cycles are necessary. Frequent maintenance activities like upgrades or patches over the lifetime of the application can cause working features to regress; this means a regression cycles need to be run for any change that goes into the environment.

In the article Cost Benefits Analysis of Test Automation, Douglas Hoffman says that "Test automation is not always necessary, appropriate, or cost effective. In cases where we are making decisions based upon an expected return on investment, analysis can direct us to where test automation can benefit us" (Douglas Hoffman, 1999).

3.3 Test-Fix-Test Cycles

For any software application, quality is achieved not just by better testing or better test coverage, but by going through series of Test-Fix-Test cycles until defect counts are reduced to below thresholds. Test-Fix-Test Cycle is basically the testing team running a set of functional tests on the application and logging defects, then the development team fixes the defects and deploys the code changes to the test environment, the testing team re-runs the same test cases and verifies the fixes, but may find other defects. Testing in itself doesn't change or improve the quality of the product or application, the goal of testing is to find defects or deviation from expected behavior so that the development teams can fix the defects or issues reported. Depending on the number of defects found per regression cycle, the project team needs to run several regression cycles to ensure the defect numbers are below acceptable levels.

4. Metrics and Measures

Metrics help the project managers, project team members and stakeholders visualize the value of test automation. ROI on test automation can be calculated in many ways, but we will discuss few examples like test coverage, time gain and defects found. Good test automation metrics can quickly be related to key project attributes like cost, quality, risk and schedule. Below we will outline good test automation metrics with some examples that project teams can use.

4.1 Test Automation Coverage

Every application or product will have test cases in the regression test suite that need to be run every time there is a change. As teams are actively developing new functionality and features, the regression test suite grows.

Test automation coverage metrics tells the team how many test cases are automated vs. total number of test cases that can be automatable. Some teams may call this metric as Max Automation Coverage. Better test coverage means finding defects faster, allowing the team to fix them sooner leading to improved quality and reduced risk.

Other metric that is popular is Automation % = # of Automated Test Cases / Total Test Cases [manual + automated]. This metric tells the team how many of the test cases are automated in the total test suite or test bed.

4.1.1. Example Graph

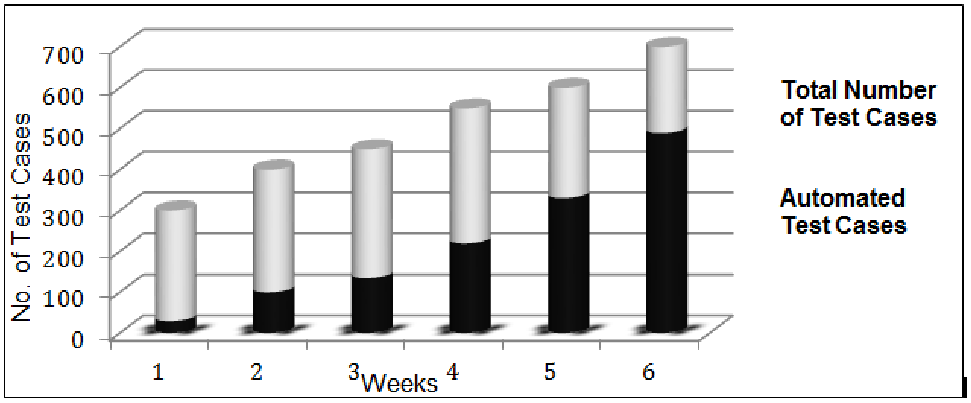
Test automation coverage is typically expressed in percentage (%); it is the ratio of number of test cases automated to total number of automatable test cases.

$$TAC = \frac{\text{No. of Automated Test Cases}}{\text{No. of Test Cases that can be automated}}\ x\ 100$$

For instance, if the project team has 1200 test cases that can be automatable and only 500 test cases are automated and ready to run on demand:

$$TAC = \frac{500}{1,200}\ x\ 100$$

$$TAC = 41.6\%$$



4.2 Time Gain per Regressing Cycle

Regression cycles need to be run frequently and probably daily if team has Continuous Integration. Running regressing cycles by executing test cases manually can take days or weeks depending on the number of test cases in the regression suite. Test automation can significantly cut that time. With automation, regression test cycles can be run faster, allowing the teams to run them multiple times in a shorter time reducing the overall project schedule and cost.

The time gain and effort reduced for each regression cycle can be converted into cost saving considering that the project needs fewer testers as number of test cases to execute manually declines, but the team needs to consider the initial cost of automation (effort and tools required to automate the test cases). The initial cost of automating a set of test cases is always high, but with each regression cycle there is a certain cost saving, after a certain number of regression cycles the cost saving for each regression cycle will add up to the initial cost of automation. After this point, the ROI on test automation efforts moves into positive territory. The project team needs to find that inflection point based on the number of regression cycles, sometimes the initial cost of automation may be offset after 10 regression cycles or it may take 50 regression cycles to get to that point.

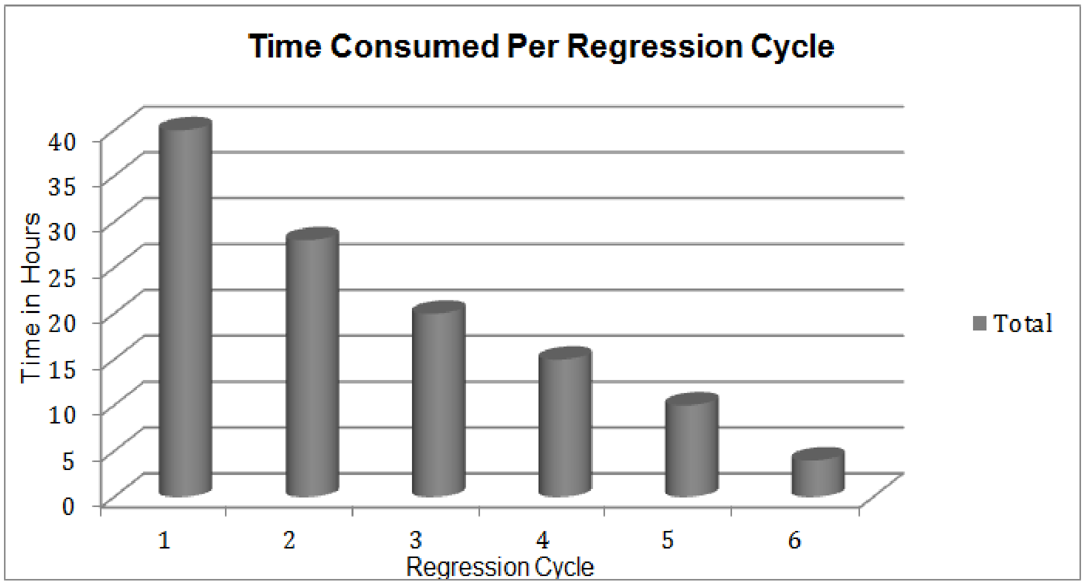
The project team needs to look at the overall life of the application or product; a typical life of a software application can fall between 2 to 6 years. Depending on the number of regression cycles needed throughout the life of the application or product, the team can make a decision on if investing in test automation will have positive or negative ROI.

4.2.1. Example and Graph

Time gain per regression cycle is comparing the time it takes to run the regression cycle manually vs. running it using automation tool. For instance, if it takes the testing team 40 hours (5 working days) to execute 200 test cases manually and same 200 test cases can be executed by the automation tool in 4 hours, the time gain is 8 (hours) x 5 (days) – 4 (hours) = 36 hours

This 36 hours savings can be converted into cost savings my multiplying it by number of testers on the team and hourly rate. But again we need to factor in the initial cost of test automation.

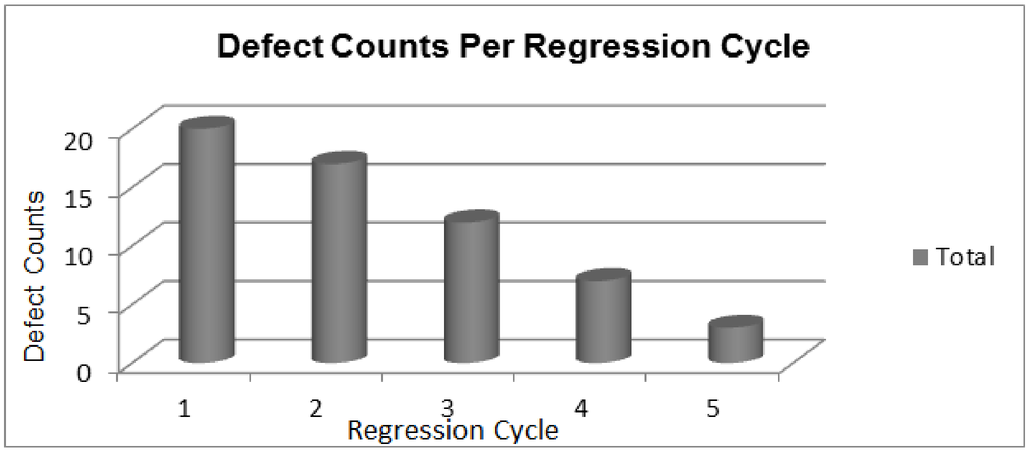
For example, if it takes $50,000 to automate 100 test cases and there is a cost saving of $3,000 for each regression cycle with the use of automated test cases, it will take 17 regression cycles to offset the initial cost, after the 17th regression cycle the ROI on test automation is positive.



4.3 Defects Found per Regression Cycle

With every regression cycle, the number of defects found can tell the team about the quality of the application and effectiveness of the test automation. More defects found indicate poor quality and less means better quality. However, if the test automation is finding more defects, it says your test automation is working and effective. Defects found per regression cycle are a good indicator of quality and as they trend downwards over a series of regression cycles, it builds the confidence of the stakeholders.

4.3.1. Example Graph



4.4 Cost Savings (ROI) and Challenges

Cost-benefit analysis can be applied to test automation; implementing test automation for a project can add additional costs, mainly for resources with automation expertise, training, tools and hardware. Depending on the number of regression cycles needed or projected for the life time of the application, the ROI on test automation can move into positive territory.

As discussed in section 4.2, ROI on test automation can be simply calculated by the below formula, but not many quality experts agree that this is the right approach.

ROI = (cost of manual testing– cost of automation) / cost of automation

Teams need to consider the ongoing costs for automation, mainly related to maintenance. Every application goes through changes throughout its life, all these changes impact test automation scripts. The automation scripts need to be tweaked or sometimes re-scripted depending on the change to the application. Defect tracking costs may also vary depending on the manual effort required as number of defects found can fluctuate for each regression cycle.

According to Michael Kelly, the real return on investment for automation is based on different types of automation done by the team and the value it adds to the overall testing effort (Michael Kelly, 2004). Automation does offer tangible and intangible benefits. Tangible benefits may be related to hours saved and time gained, intangible benefits can include faster feedback from users, finding defects sooner in the development cycle.

5. Test Automation with Continuous Integration (CI)

Continuous integration (CI) is a software development practice of merging all developer workspaces into a common codebase once or several times a day. Continuous integration originates from extreme programming (XP). Though CI was originally intended to be used with automated unit tests written through the test-driven development, later running automated functional test cases has become a standard.

Most agile teams today have some level of continuous integration (CI) built in. Continuous integration is not just actions of source control integration and automated builds, but as a complete process including integrating code to baseline, automated compilation and build, running unit tests and automated regression tests (if available) and most importantly if the tests fail, all the activities that follow to make the build clean and stable again.

6. Automation Tools

Test automation is basically using a software product or tool that is outside of the application under test to execute tests and report failures. Test automation tools can automate repetitive tasks or steps that are originally documented by a test analyst. The automation tools can offer additional benefits by expanding the assortment of testing that is difficult or time-consuming to execute manually.

Choosing the right test automation tool is a key decision that project team needs to make, the decision can be influenced by cost of the product, the technology used in developing the application under test, skillset needed to create and maintain automation, scalability, body of knowledge, compatibility etc.

6.1 Open Source vs. Vendor Tools

Open Source tools are freely available and can provide lot of flexibility, but they need some upfront work to create a framework. If the team has programming and development skills, open source tools may be ideal. On the other hand, vendor tools usually come with a framework, are user friendly and designed to hit the ground running. Vendor tools or vendors also provide support and training if needed.

There are many open source UI based functional test automation tools available in the market today like Selenium, Watir. On the other hand QTP, Ranorex, TestComplete, Test Studio (Telerik), eggplant, Certify are popular vendor based tools that used by many teams.

6.2 ATDD Frameworks

ATDD (Acceptance Test Driven Development) is an agile engineering practice that is built on test automation framework. ATDD is a complete paradigm shift from other agile software development practices where developers build the application code based on user stories and acceptance tests; an ATDD framework like Cucumber, FitNesse is used to integrate automated tests to application under test. ATDD frameworks have a Wiki mechanism to document test cases in plain language and the tests are typically defined by users or product owners. The tests are run by the product owners on a day-to-day basis and it creates a constant feedback loop between development team and business.

ATDD integrates developers, testers, product owners and users into development effort, a kind of forced collaboration as the product is being developed. ATDD brings the concept to quality is everyone’s responsibility into practice.

6.3 Cost Benefit Analysis on Tools:

Cost benefit analysis is the important aspect of test automation tool selection. There are many factors that influence the decision; here are few things that teams need to consider.

1. Initial investment, licensing and ongoing costs for the automation tool
2. Flexibility and scalability of the automation tool
3. Initial framework setup necessary for the automation tool
4. Skillset and training required to use the tool
5. Compatibility with application under test

7. Test Automation in Agile

Test automation is becoming a foundational practice in agile development and it should be the way agile teams need to build software. Agile teams see the value of test automation in terms of Velocity and Quality. The value of Test Automation is clearly visible within the CI model; it's obvious that teams can deliver better quality and business value faster. Test automation in itself cannot deliver much value if not used daily within the CI model. Test automation has limited value if it's run every few weeks and just before the release. The true value of test automation is only realized when team can run it every day as part of CI, it allows the teams to find defects faster when there is enough time fix them.

7.1 Test Automation as an Engineering Practice

For most agile teams, test automation with CI is a standard engineering practice, a job zero (must do) to keep up with the velocity, release cadence and quality expectations. Test automation scripts need to be run daily in an integrated environment to find and fix defects more frequently. If this doesn't happen, teams are not using automation to its full potential to deliver best ROI. The true ROI of test automation is realized when Scrum teams can use automation every day in the development sprints, find and fix defects faster during the sprints.

In agile, test automation certainly increases quality, improves time-to-market, but not necessarily decrease costs because of the iterative nature of changes to applications. Many projects use test automation to increase velocity, quality though there is significant increase to the cost. Test automation enables teams to release more frequently as testing can be completed faster, this is more important for some projects than cost savings. If it's an agile project with a working CI model, test automation should be fully integrated with the software development process,

8. Continuous Deployment with Test Automation

In the past couple of years, Continuous Deployment has revolutionized software delivery. Continuous Deployment takes the CI and test automation to new level of agility and delivery. With Continuous Deployment agile teams can transform its software delivery and can deliver features and fixes as they become ready without having to wait for set release date.

8.1 Automated Deployments and Infrastructure Automation

Infrastructure automation or automated deployments have been emerging in the past couple of years and are seen by some as standard engineering practices for agile teams. Infrastructure automation is a key aspect of DevOps, where development and operations team work together. Test automation and Continuous Integration (CI) have been around for some time now but if the agile teams cannot push working features, patches to production (to users) quickly and efficiently without pulling in all the development and operations teams, then they may not be as agile as they think. There are many tools, some open-source that teams can use to reduce the time consumed in doing a production release by automating deployment tasks. Automated deployment tools can deploy code to production in minutes, saving time for development teams and reducing production downtimes.

8.2 Industry Examples

LinkedIn uses Continuous Deployment to transform its software delivery, they significantly improved the frequency and speed of their production deployments, and with this they are able to deliver features and fixes as they become ready. Other companies seeing value of Continuous Deployment are Facebook and Etsy.

LinkedIn, the largest online professional networking site, adopted Continuous Deployment as the software development methodology. Under this model, a developer writes code for a new feature and checks it into the main line of software that is shared between all developers, a line known as "trunk" within the software version control system. The newly-added code or feature is subjected to large number of automated tests that will identify any defects. Once the tests are completed and no issues are found it is merged into trunk and classified as ready for deployment, this will allow product managers to select and push new features into production as needed.

Etsy, an online marketplace for handmade goods, vintage items and craft supplies, has embraced Continuous Deployment practices in last couple of years. Etsy's development team develops small change sets and deploys them frequently (Ross Snyder, 2013). This process allowed Etsy to keep up with user demands and stay profitable. Etsy uses engineer driven QA, test automation and solid unit testing meet the quality standards, all these practices were integral to their development process.

Facebook, the world's largest and most successful social media site has adopted Continuous Deployment to safely update the site with hundreds of changes including bug fixes, new features, and product improvements (Facebook Tech Talk, 2011). Facebook has a test centric engineering culture and uses a test automation framework build with Selenium and Watir. The automated tests run with every change and revision, this helps Facebook engineers to find failures and fix them faster.

9. Conclusion

Project managers or stakeholders sometimes may not see the value of test automation and benefits it offers. This could be an impediment to investing in and prioritizing test automation efforts. The project team or the test lead can influence the decision to invest in automation by showing the value in terms of positive impact to key project management attributes.

If project team or the test lead can show the value of test automation in terms of positive effect on project's schedule, cost and quality, the aspects that project managers and stakeholders most care about, it will be much easier to make a case for test automation.

References

Douglas Hoffman. 1999. Cost Benefits Analysis of Test Automation

Dorothy Graham and Mark Fewster. 2012. Experiences of Automation. Pearson.

Michael Kelly. 2004. The ROI of Test Automation

Rayn Tate. 2013. The Software Revolution Behind LinkedIn's Gushing Profits
http://www.wired.com/business/2013/04/linkedin-software-revolution/

Ross Snyder. 2013. Continuous Deployment at Etsy: A Tale of Two Approaches
http://www.slideshare.net/beamrider9/continuous-deployment-at-etsy-a-tale-of-two-approaches

Chuck Rossi. 2011. Pushing Millions of Lines of Code. Facebook Tech Talk.
https://www.facebook.com/video/video.php?v=10100259101684977