iQ transit

How to Get More From Your Developers

Achieving Better Vendor Performance Though Communication and Transparency

A White Paper | June 2017

7 Actions to Success

Introduction

Perhaps you've seen this before: A vendor or freelancer puts their best foot forward during the procurement process and seems outstanding. But once the relationship is established and work has commenced, the vendor becomes complacent. The developer's performance drops in number of issues resolved, code quality or milestones achieved. The client perceives that the value they are receiving from the developer is worth less than the amount they are paying.

What can clients do to improve the performance of their software developers?

Although clients can't change impossible situations or make low-performing developers better, we've identified seven steps they can take to achieve better performance from their developers. These solutions focus on communication and transparency, and can put companies in a stronger position to find replacement developers, if necessary.

At iQtransit our general approach is to create transparency by making the work that the developer is doing more visible. Therefore, we propose setting up a knowledge management system for documentation, switching to well-established technologies and performing code inspections so other developers can review their work. We also encourage effective communication. Therefore, we recommend clients become familiar with common communication tools that software developers use - and use them. And we encourage clients to reduce the startup time for new developers beginning work on their projects. These steps help create the competition necessary to keep performance at the same level as when they started.

Disclosure: Our advice comes from 20 years of experience working with clients on web and mobile development projects and from analyzing that data. Because iQtransit believes this process works, we recommend and implement these steps with our own clients.

Action 1

Develop a Collaborative Knowledge Base

A simple and effective way to improve developer performance is to create a simple and secure collaborative space (or knowledge base) to store relevant documentation about your project.

A knowledge base can improve developer performance in a few ways. First, since knowledge is more readily available to others, it becomes possible for another developer to work on the project without extensive training. This means the developer will have "competition" from other developers who can more easily take over if performance is lagging. We've seen this have a positive effect on developer performance.

"Create competition with other developers to keep performance strong.

A knowledge base can also help identify "hoarders" - members of the development team who are unwilling to share the internals or secrets of a project (sometimes this is done so that they can't be replaced). A developer who is not willing to share the secrets of the company should be asked to improve their knowledge sharing for the benefit of the company.

The knowledge base also improves the collective "memory" of the corporation, codifying the obscure aspects of the project. This can help prevent any loss of assets were the vendor to stop working on the project. Knowing that all knowledge is protected and documented can force the developer to rely on their performance instead of being de-facto holders of the company's knowledge.

Our Recommendation

Use a software package like Wikimedia, Atlassian Confluence or Basecamp to set up a knowledge base to capture project documentation and digital assets. Encourage (and pay) your developer to capture project details there.

Action 2

Perform a Formal Code Inspection by Outside Developers

A formal code inspection can identify as much as 60% of defects in the reviewed software code.[1]

In our experience, having an alternative, external group of developers perform the inspection provides direct improvements to productivity of developers. A 20% increase in developer productivity due to code inspections has also been shown in research studies.[2]

The code inspection should not be a threat or criticism of the developer, but rather be a positive way to identify errors within the code and open the hidden secrets of the projects to others. Clients can explain that opening the project for others is a process that should be streamlined for other purposes as well, such as a disaster continuity plan.

The inspection keeps the developer on their toes. The introduction of an external party always brings up questions for the existing vendor-developer, such as whether the developers performing the inspection will eventually replace them. Although this is not the intent of the inspection, the possibility of a poor review can motivate non-performing developers to produce better work.

The inspection can also improve code quality. A formal inspection has one of the highest defect-catching rates of software quality techniques performed today.[3]

Finally, the inspection can identify knowledge gaps between management and the developer, found when the 3rd party asks questions and reviews existing documentation. These gaps in knowledge can be captured in a knowledge base as they are identified. If you decide to switch vendors, having that documentation will make the transition easier. Clients can also use the documentation gathered as a requirement for disaster recovery or PCI compliance.

Our Recommendation

Hire a competent, external group of developers to review the work product of your vendor-developers. At the minimum, poor or sloppy programming practices will be identified, and at the best, the vendor will change for the better and improve performance.

"Having an external group of developers perform the inspection provides direct improvements to productivity of developers.

[1] McConnell, Code Complete, Second Edition. Page 485
[2] Fagan 1976, Humphrey 1989, Gilb and Graham 1993, Wiegers 2002.
[3] McConnell, Code Complete, Second Edition. Page 485

Action 3

Manage Source Code

In our work, we've found that it is not uncommon for vendor-developers to threaten to withhold a client's source code for negotiating leverage. It most often happens during a period of transition, such as when replacing the vendor. To avoid this, we strongly recommend that companies directly control access to their source code. Less technical companies, who may not understand exactly what work is being done, can still benefit by holding the keys to the source.

Managing source code repositories can improve communication between companies and their vendor-developers. Today, revision control tools like Git, Beanstalk App and Bitbucket include powerful communication features (like "blame") which allow users to see every change that developers make. All work is visible via commit logs, and can be seen with a clickable web link. Deep communication around that work, such as comments, pull requests, and issue tracking also enhance the visibility and transparency of the project. If developers are not contributing to the project, it becomes obvious and clients can act to improve performance.

Companies can also share the repository with other vendors. Unlike a zip or tar file, the revision control repository includes the complete revision history of the project. This can be used as a tool for analysis or forensics were a project to go bad.

Companies should be involved in the work their developers do. Communication helps to keep them motivated and shows your developers that you care about the work they do. To better manage their developers, clients should manage the source code repository for their projects. We recommend that clients own and hold all the master keys to software source code. Further, by using standard cloud-based tools, you can ensure that you are notified with every change that occurs.

Our Recommendation

Sign up for an online distributed revision control system such as GitHub, BeanstalkApp or BitBucket. When you start a project with a vendor, setup a repository for all source code and be sure the vendor uses it. This prevents any extortion of your digital assets and keeps you in control of the asset.

"To better manage their developers, clients should manage the source code repository for their projects.

Action 4

Choose Well-Established Technology

Developers always want to try out the latest technology. However, as a client, you should acknowledge that the developer's goals may be different from yours when choosing technology platforms. The developer may want to enhance their skills for the next project, while companies want to protect your investments and operate at a profit. We recommend clients to be especially wary of overengineered systems which don't match requirements. Most clients are not Google, and should not be using tools designed for the world's largest companies.

Choosing an obscure technology can make it very difficult to find developers for your project in the future. Clients should reduce risk by staying away from trendy or obscure technologies - unless they are proven or truly needed for their unique capabilities. If your chosen technology takes a pathway which makes it less widely adopted (such as being eclipsed by a competing product), your choices for developers in the future may be either very limited - or very costly.

You can "buy time" with modern technology by waiting until it has been proven and heavily used before migrating or adopting. Be deliberate about your location on the technology adoption curve[4], and be willing to stay back when you don't truly need bleeding-edge. (Bleeding edge technology has hidden costs because it brings unnecessary complexity.)

Our Recommendation

Use well-established technology for non-unique requirements. Be aware of where you are on the technology curve to avoid unexpected technology detours. Main-stream technology means more choices for developers and can mean better performance from your existing developers because they can be replaced.

[4] *https://en.wikipedia.org/wiki/Technology_adoption_life_cycle*

Action 5

Reduce Development Setup Time

When clients reduce the friction of handing a project to another developer, they increase their leverage, which can create improvements in developer performance. Reduced start-up time means that a developer can re-instate a working copy of the development environment quickly and repeatedly.

For example, iQtransit's projects can be reinstated with these 2 simple commands:

```
% git clone http://github.com/project/
% vagrant up
```

Within minutes the development environment is running on the development machine. The site or app is in its native environment, ready to accept changes, and it works the same whether the developer uses Mac, Windows or Linux.

The environment can be useful for training purposes. Because of the "disposable" nature of virtual environments, the project can be modified to see how it is effected by experimental changes, while being completely isolated from other developers. This enables other developers to be trained on the system, which can improve vendor's performance by creating competition and improving skills.

"Reduced start-up time means that a developer can re-instate a working copy of the development environment quickly and repeatedly.

With fast startup, clients can hand off small parts of projects to more specialized, or less expensive vendors or developers. For example, if a minor part of a large project requires HTML development, you can invite an HTML specialist to work on that small aspect of the project. Without the fast startup virtualization offers, the initialization time would be too great to make the HTML portion of the project worthwhile to hand off, so the less specialized (and more expensive) developer would complete the task.

Environment management tools such as Vagrant or Docker help to keep environments uniform among vendors which may have more divergent backgrounds. Because the environment will match the production environment closely, problems with specific developer setups become less important, reducing bugs.

Fast developer startup can give you leverage to negotiate the development fees you pay. After all, if you can hand off a project to another developer in a matter of minutes, there is less barrier to overcome when finding a competitive developer or vendor. This can be effectively used in negotiations with your developer, providing gains in value.

Our Recommendation

Use tools like Vagrant, VirtualBox and Docker to reconstruct programming environments quickly. This makes it fast and efficient for other developers to work on the project.

Action 6

Improve Access Control and Security

To efficiently hand over a project to another developer, you need to be able to setup, share and revoke credentials quickly and easily. You need to be sure that all the passwords and digital keys are not only stored safely, but also that they are accessible to those authorized to use them.

The most sophisticated tools provide features such as 2-factor authentication, sharding (storing only portions of keys in different locations to make exposure more difficult), strong encryption and ability to revoke permissions as vendors change. However, simpler tools can suffice.

Our Recommendation

Consider using a cloud-based access control tool such as LastPass or HashiCorp Vault to manage secrets organizationally.

“To efficiently hand over a project to another developer, you need to be able to setup, share and revoke credentials quickly and easily.

Action 7

Continuous Integration and Review

Clients should ask their developer for a password protected stage/preview site (for web sites) which is continually synched with a revision control system or code repository. This allows clients to review their developer's progress on the site, app or software on a continual basis, and prevents developers from periodic slowdown, hidden within the project's schedule.

Such systems are often called "continuous integration" systems and considered a best practice.

Continuous integration results in improved communication and transparency and can help "break down barriers between customers and development". [5]

> **"To maximize performance, developers should be using continuous integration tools.**

Continuous integration is the opposite of "dog and pony show" development, which often occurs with outsourced development teams. This occurs when the developer assures the client that everything will be ready in two weeks, but no feedback or communication occurs until "demo day". Due to the developer's other projects, little work gets done until the last few days. Then, a large amount of work spent is dedicated to the presentation of the work, meaning very little actual work is getting done.

Our Recommendation

At the minimum, developers should offer a continuously updated (minimum once per day), password-protected staging site where clients can review changes at any time. To maximize performance, developers should be using continuous integration tools like Jenkins, Go, or Travis CI.

[5] *https://www.martinfowler.com/articles/continuousIntegration.html*

Next Steps

Now That Every Company is a Technology Company, Clients Need To Get The Most Out Of Their Vendors.

Clients can't make their developers better, but they can change how they manage them. We recommend that you pay close attention to what your vendors are doing, by creating collaborative systems and engaging the communications pathways that developers already use. Create openness and transparency so anyone can contribute. Understand, communicate and learn as much as you can about the work your vendors are doing. We believe you will be rewarded with strong, sustained performance from your development team.

"Clients can't make their developers better, but they can change how they manage them.

Charlie Dalsass
Founder and CTO

e: charlie@iqtransit.com
s: live:charlie_1644

Charlie is the CTO and co-founder of iQtransit. He was previous co-founder of Neptune Web, and Lead Developer at Banta Integrated Media (now R.R. Donnelly).

iQtransit, Inc.

110 Canal St. | 3rd Floor | Lowell, MA 01852

e: inquiries@iqtransit.com | p: 1-978-886-0798 | f: 1-617-507-5959