The Cost Benefits of Genuitec's Secure Delivery Center

A White Paper that explores the concerns, issues and business value of utilizing Genuitec's Secure Delivery Center - a service that assists enterprises in managing and distributing internal software tool stacks, freeing developers to do what they do best - write code.

September, 2011

"There is nothing so useless as doing efficiently that which should not be done at all."
-Peter Drucker

genuitec

The Cost Benefits of Genuitec's Secure Delivery Center

EXECUTIVE SUMMARY

The goal of this white paper is to explore the business value of utilizing Genuitec's Secure Delivery Center (GSDC) and how a company with 500 developers can save up to $650,000 annually in time cost and realize full ROI in weeks. The paper will explore common issues faced by today's enterprises that are currently experiencing a variety of pain points when standardizing, distributing and managing the roll-out of their software tool stacks to development teams. The paper will also encapsulate several areas where corporations are experiencing developer time loss and how GSDC can free time and money from the tooling logistics process.

…a company with 500 developers can save up to $650,000 annually in time cost…

WHY DO WE NEED BETTER DELIVERY SOLUTIONS?

Deciding on the types of tools ideal for a company, and then setting up the tool's preferences and add-ons are complicated enough tasks, often requiring the time and energy of a central tooling group or dedicated "toolsmith." However, once the tools are decided upon, additional concerns arise that require teams to manage licenses, updates, roll-out timing, standardization and more.

…teams can be locked into "diseconomies of scale" whereby new developers are added and trained, but development time is wasted on communicating the necessary tools and processes needed to complete the project…

These common tasks are considered non-value-added work; meaning no code is actually written and projects are no longer moving forward. Teams can be locked into "diseconomies of scale" whereby new developers are added and trained but development time is wasted on communicating the necessary tools and processes needed to complete the project.

When you factor in multiple groups updating tooling separately and managing their tools independently, these common corporate problems can be increased by an order of magnitude. The task of simply "picking the right tool" suddenly seems much easier than the actual implementation of that decision.

...the cost to corporations can rise to $62,000 per year for license management alone…

LICENSING CONCERNS

Let's start with something seemingly simple: licenses for software use. For example, a corporation currently utilizing MyEclipse Blue Edition as a development standard may download the offline installers directly from the MyEclipse Web portal, and then place them on an internal server for developers to consume. Seems easy enough - but how are these end users keeping their licenses up to date? Often, the onus is on the individual end-user to key-in their license code. But what happens if they don't? What if some developers have issues with their license keys? Many corporations find license management a problem that can consume time away from their developers' core competencies in this diseconomy of scale. But what does it really cost?

Studies by leading salary analysts have indicated that a Jr. Java Developer in a high-tech corridor of the United States will cost a company - between salary and overhead - approximately $63 per hour. This means that if a low-level developer spends even 2 hours per year on license maintenance or problem resolution, it will cost an enterprise about $125 in developer time. If you factor the same for a team of 500 developers, the cost to the corporation rises to about $62,000 per year for license inconveniences alone.

Lost money to handle licensing management:
$62,000 USD/year for 500 developers

…the responsibility for maintaining the tool versions and each team's installation guides often falls to a higher-up Sr. Developer or Team Lead…

MULTIPLE TOOL VERSION CONTROL

Another common issue is having to manage disparate teams' tool stacks simultaneously. Not everyone in the company will be working with the exact same tooling or version number. It is the nature of the beast that corporations often have multiple projects under simultaneous development and use different tools for each project. Teams will often standardize on a tool stack and preferences for a specific project and not change or update until the task is complete. This creates a natural fragmentation between teams, despite the fact they may be using the same tool brands or plug-ins.

The responsibility for maintaining the tool versions and each team's installation guides often falls to a higher-up Sr. Developer or Team Lead who will have to deal with any "broken build" issues. A common scenario involves a team member neglecting to update their tools, or updating too soon. The time cost to discover the issue and then the corresponding roll-back or update can be measured in hours, or even days.

...if you have a team of 500 developers, half of whom have tool stack compatibility problems in a year, the enterprise will lose about $88,000 per year just spent fixing multiple-version issues...

Often, the Team Lead will be called after the developer has spent a couple of hours trying to fix the problem with no success. The Team Lead then intervenes and spends a couple of hours troubleshooting and recreating the problems to find solutions. Conservatively, this can take 2-3 hours of developer time and 1-2 hours of Team Lead time per occurrence. At average hourly rates of $63 and $78 per hour (the average for Team Lead roles), every occurrence of tool breakage can cost the enterprise about $350 in lost costs. If you have a team of 500 developers, half of whom have tool stack compatibility issues in a year, the enterprise will lose about $88,000 per year just spent fixing multiple-version issues.

Lost money to fix tool stack versioning: $88,000 USD/year for 500 developers

CLEAR COST OF UPDATES

Obviously, enterprises want developers updating tools at the appropriate times, and not haphazardly. But even planned updates and roll-outs have cost concerns.

A common scenario involves a central tooling team or toolsmith sending out an internal communication that an update is available and required. Often, an update has been made to a master "installation wiki" or other document that details the update and how to get it.

The problems begin to arise as once again the onus is on the end-user/developer to get the update, install it correctly (and do so in a timely manner). In many cases, the software is obtained directly from a vendor portal (such as MyEclipseIDE.com or Eclipse.org), putting the responsibility on the developer to download packages that can be up to 1GB in size (much larger packages may be downloaded by developers, like IBM Rational at 13GB). Meanwhile, overall team productivity is impacted when a forgetful developer misses a required update or applies the change incorrectly causing a breaking source code change impacting the product.

Looking at Genuitec's Pulse statistics (Pulse is software provisioning tool with 1.5 million developer-users), we can see that average Eclipse tooling-based developers update their tools on average 2.2 times per year - we'll say twice a year to keep the math easy. A team of 500 developers downloading 1gb of software from a vendor at 1mb/second takes about 17 minutes. Installing the new software and setting up the required preferences, as well as updating workspaces can easily take another 30-45 minutes.

…since an update is usually required to move on to the next project, it can easily take a full business day to get all developers on the same page to start the project…

If we factor in a conservative 5 percent (25 developers of the 500) download or setup failure rate, we can easily reach an hour of time - if all goes well for the majority of developers. Since an update is usually required to move on to the next project, it can easily take a full business day to get all developers on the same page to start the project. So, if we say 8 hours twice a year to roll out updates, even lower-level developers can cost the company over $510,000 in lost time on updates alone. Overly complicated update scenarios or software rollback instances will increase costs exponentially.

Lost money to rolling out updates:

$510,000 USD/year for 500 developers

GETTING THE "NEW GUY" RUNNING

Getting a new developer up and running with the right tools and ready to contribute on a project can often be measured in days rather than hours. The work flow takes the route of downloading, installing, updating and general setup of a workbench - all aspects considered time consuming but necessary tasks. Once the new software developer is set up, you now have to hope that he followed the multiple-page "installation wiki" perfectly; otherwise more time will be lost fixing workbench issues.

Naturally there are questions and concerns the new team member

…new hires ranked second in the major factors that contributed to failed projects…

will have, requiring both the time and energy of the team leads and senior-level developers. Between the download time, setup of environments and troubleshooting, new developers are lost without the necessary skills and/or expertise to actually join the project or get it started.

This type of wasted time on new hires ranked second in the major factors that contributed to failed projects according to KPMG. From a respected KMPG-sponsored survey, 38 percent of respondents identified the change in essential personnel as a risk, and they ranked this factor as number four for wasted time on development projects.

Furthermore, from a Standish Group survey on why software projects both succeed and fail; IT managers responded that incomplete requirements and specifications ranked number two with 12.3 percent respondents stating this is a serious project challenge.

And, from the "Failed *IT* Projects (The Human Factor)" by Sheila Wilson (University of Maryland Bowie State University, 16 May 1998) the author states through extensive research that:

> *"Retaining team members throughout the project plays an important part as to whether the project and the team will be a success. If the team size, skill level, and cooperation exist, changing the same team structure may cause many problems. First, retraining new team members is very likely to affect scheduling and budgeting. Members already on the projects have to take time out of their already too busy schedule to bring the new members up to date on the project status. This decreases productive time spent on the project, and it increases the budget, schedule, chances for more mistakes, etc."*

As the research demonstrates, bringing new developers into a project will negatively impact the time spent on building and maintaining a project, so much so that projects could actually fail with enough employee changeover or inter-project movement. Now, mix-in incomplete requirements and specifications and the prospects for project success appear dire. By eliminating the responsibility of the development teams to invent, maintain and execute development tooling installation procedures, corporations can avoid innumerable risks.

ADVANTAGES OF GENUITEC'S SECURE DELIVERY CENTER

GSDC provides corporations with a cost-effective way to mitigate concerns around tooling updates and management. For example, license compliance and management is handled directly through a centralized dashboard that is managed and maintained by a toolsmith, who in turn works under the "lead" or chief project coordinator. This license management method takes the onus off developers to find, apply and manage their workbench licenses and instead frees them to actually code their project. It puts tool responsibility back into the hands of those employees hired for that role, the toolsmith with the approval of the lead.

…this license management method takes the onus off developers to find, apply and manage their workbench licenses and instead frees them to actually code their project…

Further, GSDC allows toolsmiths to conveniently set up vast developer tool stacks ONCE, ahead of time, and provide a centralized location for updates and version control. By also enabling push capabilities, the struggles of making sure everyone applied the update are a thing of the past. Developers pull down their required updates directly from the company intranet where GSDC is hosted allowing toolsmiths control of the entire update and correct version number process.

GSDC allows companies to manage versions of tools between teams easily. By giving toolsmiths visibility into the tool stacks installed on each developer machine it is easy to have employee X and employee Y on different tools for different aspects of the projects. Teams can even have multiple, easily-managed instances of tool versions (such as MyEclipse 9 and 10, for example) on the same machine, but provisioned correctly to avoid the broken build commonly found when instances of older and newer software are used at once.

Conveniently, GSDC does not require companies to "reinvent the wheel" when it comes to setup. GSDC integrates with existing systems and intranet services. It provides customizable, sample intranet portals that companies can utilized (if desired) and ability to deploy software through existing SMS infrastructures. The supplied management and end-user interfaces give the enterprise a quick proof point to rapidly test GSDC, as well as a customizable interface they can brand and incorporate within their existing infrastructure.

…GSDC does not require companies to 'reinvent the wheel'…

In short, GSDC allows your developers and the team leads to do what they do best - ship quality software. GSDC easily removes the requirement for developers to be toolsmiths, but empowers toolsmiths to visualize and control the tool-use for each project.

YOU'RE FEELING SECURE

Security and integrity of your chosen tooling setup is a key concern that is surfacing more frequently. The ability to "lock down" development environments - and thereby eliminating the ability to change or update them mid-project - has become a go-to strategy for many IT departments. However, the lock-down is often hard to enforce, as it has historically relied more on an honor system (i.e., "please don't change anything") or network access control to limit the ability to see/discover update sites.

…minor inconveniences and troubleshooting of tooling problems cost the company nearly $700,000 per year…

The fallacy of these strategies is clear: they are impossible to enforce. Developers can find update site mirrors. Someone pushes the update button without thinking. These scenarios are all-too-common, and the legacy solutions tend to decrease risk, rather than eliminate it.

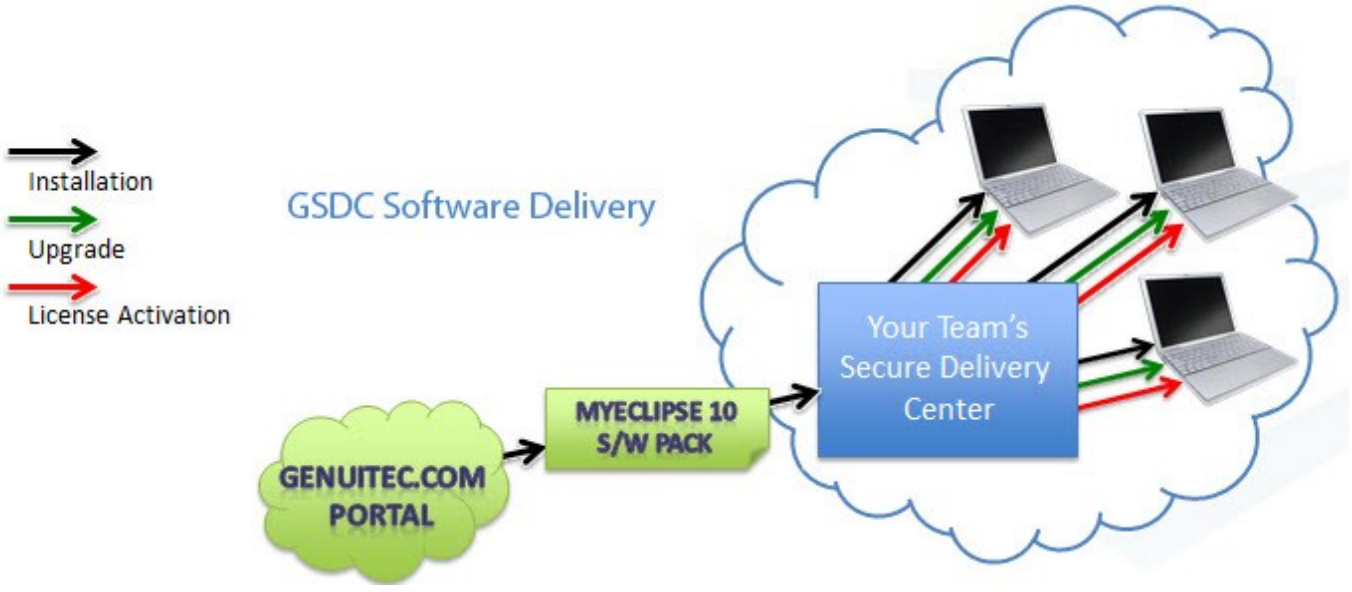
GDSC eliminates these concerns by bringing your distribution internal and completely behind the firewall, and allowing you restrict access only to internal sites if you so choose. This also allows you to time update roll-outs more effectively, as the update sites only become available when you authorize them.

COST CONSIDERATIONS

…GSDC presents a savings of 93% in developer time costs annually…

In the sample scenarios described above for a team of 500 developers, minor inconveniences and troubleshooting of tooling problems cost the company nearly $700,000 per year. That's about $1,400 per developer in additional time costs - and that's in ideal, everyday scenarios.

In contrast, GSDC costs only $50-$100 per developer with an annual maintenance fee of $1,500 - it presents a savings of nearly 93 percent in developer time costs lost to the above described



…full initial ROI can be realized within 2 months or less, with subsequent calendar year ROI-gain in about one month…

issues. Additionally, time overhead for the team leads and others on maintaining tool wikis, installation instructions and distributing the software criteria are eliminated, resulting in further soft cost savings.

The initial cost in setup and implementation of GSDC will vary by company, but must be considered to calculate ROI. Even estimating a man-month at a Sr. level to focus solely on GSDC setup would result in a one-time, soft cost of $10-13,000. At this metric, full setup ROI could be realized within two months or less, with subsequent calendar year ROI-gain in about one month.

CONCLUSION

The quantifiable costs associated with non-development tasks that software developers are asked to perform are very high - with the "intangible" work related aspects even higher. License concerns, tooling updates, multiple version control, environmental lock-down and new employee setup are just a few of the key concerns that are costing today's IT departments a bundle of cash and are threatening the success of software projects.

Utilizing emerging technologies like Genuitec's Secure Delivery Center can remove external responsibilities from developers and create centralized management for multiple tooling groups. This can easily and consistently result in up to $650,000 in annual savings and realization of ROI in mere weeks.

The Cost Benefits of Genuitec's Secure Delivery Center

September, 2011

Genuitec, LLC
2221 Justin Road #119-340
Flower Mound, TX 75028
USA

http://www.genuitec.com/sdc

Copyright ©2011, Genuitec, LLC. All rights reserved.

This document is provided for informational purposes only and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document.

Genuitec, MyEclipse and MyEclipse Blue Edition are trademarks of Genuitec, LLC. All other brand or product names may be trademarks or registered trademarks of their respective companies and should be treated as such. JRE, Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc./Oracle in the United States and other countries.