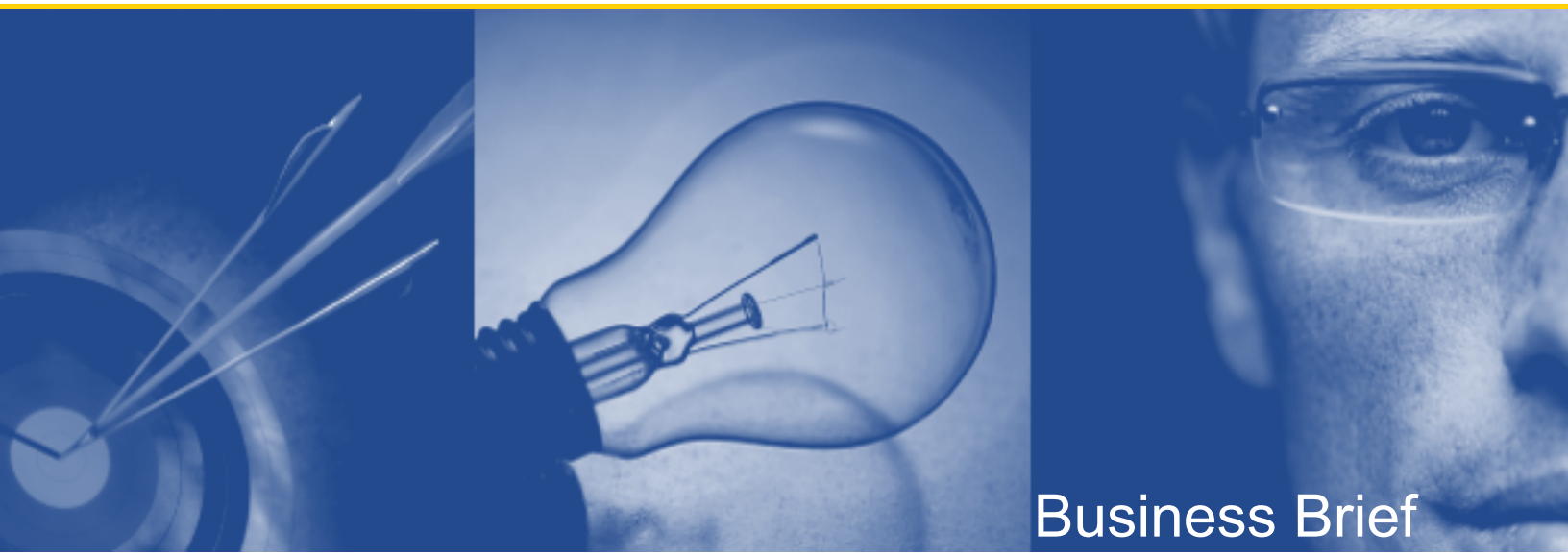
QUEST SOFTWARE®

The Hidden Costs of Freeware

A Critical Look at the Costly Use of "Free" Software

Written by
John Weathington,
CEO of Excellent Management Systems
Quest Software, Inc.



WARRANTY

The information contained in this document is subject to change without notice. Quest Software makes no warranty of any kind with respect to this information. QUEST SOFTWARE SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTY OF THE MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Quest Software shall not be liable for any direct, indirect, incidental, consequential, or other damage alleged in connection with the furnishing or use of this information.

TRADEMARKS

World Headquarters:
5 Polaris Way
Aliso Viejo, CA 92656
e-mail: info@quest.com

Please refer to our Web site (www.quest.com) for regional and international office information.

Updated—April, 2009

CONTENTS

INTRODUCTION

The song that the freeware siren sings is an alluring one; however, it may be leading your team to shipwreck on a rocky coast. As an efficiency expert, I'm always on the lookout for inconspicuous drains on your precious resources, and if you're using freeware—or are considering it—there are a few things you should understand.

I'm not saying that freeware is bad. It's actually a very good thing. Without it, we would never have great products such as Apache, Linux or Java. In fact, there's nothing inherently wrong with freeware, just like there's nothing inherently wrong with subprime lending. In both cases, however you could be setting yourself up for trouble if you don't fully understand the implications of what you're getting into.

Obviously, the lure of the "free" in freeware is where the trap lies. Sure, you can download and use the software without remitting any money. But if you really believe there's no cost involved, I'm afraid you're sadly mistaken. I'd like to share with you **three key hidden costs of freeware**.

HIDDEN COST # 1: PRODUCTIVITY LOSS

I recently ran a $3 million project with about 15 people. Of that $3 million, no more than $50,000—just over two percent—was spent on hardware, software licenses and other non-human resources. The other 97.5 percent of the money was spent on labor. With 15 people on my team, my average HR burn rate was about $1,500 per hour. That's why my number one concern was team productivity, not how much I spent on hardware or software.

To make sure my team members remained productive, I made sure they had the best development environment I could afford. I calculated that each day we lost due to a network failure or a system crash was a $12,000 productivity loss. To mitigate this, I insisted on a dedicated, high-powered server and equipped all of my developers with commercial software. I took no chances with freeware in this case.

Freeware tends to have an insidious way of slowly sapping productivity. It's analogous to a slow leak in a tire. I remember driving back to California from Chicago one year. On my first pit stop, someone kindly notified me that my rear tire looked a little low. When I checked the tire pressure, it was at 20 psi, half of what it should be. Of course, I filled it back up to 40 psi, only to realize 500 miles later that it was already down to 30 psi. I continued my somewhat paranoid trek back to California, constantly checking and refilling my rear tire.

In the same way, freeware—which is typically far less functional than commercial software—can gently drain your productivity and you may not even notice. It's not just the developers' time that you need to worry about. On the project mentioned above, I had only six developers, but they were at the core of the project's productivity. We developed the system in a highly iterative and evolutionary manner, so the developers needed to crank out code quickly in order to conduct user acceptance testing. If the developers were slow to deliver due to the limited functionality of freeware, the whole team would be weighed down.

Of course your mileage will vary depending on the freeware under consideration, but in my experience there will be a drag (we'll get into why later) on productivity. Some of the better freeware offerings may only drag your team down 10 - 20 percent; however, the wrong freeware can really devastate your productivity by 300-500 percent or even more. It doesn't take a formal experiment to conclude that a team of Oracle database developers using Quest Software's Toad® for Oracle will be many more times as productive as the same team using Oracle's "free" SQL*Plus. Even with the best freeware, the math is easy—10 percent of a $3 million project is $300,000, which is several times the amount I could ever spend on software licensing.

HIDDEN COST # 2: HIGHER RISK

Risk is also difficult to quantify, especially if you're not diligent in risk management on your projects and programs. Freeware is clearly a risky proposition, which translates to additional costs. For example, the same can be said of foregoing health insurance. My monthly premium is sky high, and it truly pains me to pay it. But I take great care to make sure it never lapses. Why? Because I don't need the expense and paperwork involved with having a serious accident while I'm not covered by health insurance.

Using freeware is like walking around with no health insurance. Commercial software—which often comes with strong warranties and support—is your protection against developmental catastrophe. In a nutshell, risk is simply uncertainty that is usually tied to an unfortunate event (loss). For instance, the serious accident I just talked about is not a certainty (fortunately for me), but it might happen, so I consider it a risk.

To frame risk, let's leverage some concepts from Six Sigma's FMEA (Failure Modes and Effects Analysis). While constructing an FMEA, you'll analyze three primary characteristics of risk: detection, probability and impact. In general, you want to strive for high detection (your ability to detect a bad situation), low probability (the chances that the bad situation arises) and impact (the effect the bad situation will have on your project or program).

What tends to happen with freeware when compared with commercial software is that probability and impact increase, which translates to a higher contingency reserve. When assessing risk for your project or program, contingency reserve is calculated by totaling the product of probability and impact for each risk. For instance, let's say I have three risks — A, B, and C — that have a probability of 30 percent, 20 percent and 50 percent respectively. If there's an impact of $500, $300, and $900respectively, my contingency reserve would be calculated as (.30 x $500) + (.20 x $300) + (.50 x $900) = $150 + $60 + $450 = $660. The table below will make this calculation clearer.

RISK	PROBABILITY	IMPACT	CONTINGENCY RESERVE
A	30%	$500	$500 x .30 = $150
B	20%	$300	$300 x .20 = $60
C	50%	$900	$900 x .50 = $450
Total Contingency Reserve			$660

As you can see, the lower you can keep your probability and impact, the less your contingency reserve costs will be.

Of course, this is a simple example. On a large project you may have hundreds of risks, bringing your contingency reserve in the hundreds of thousands—or even millions—of dollars. Every time I've done an analysis this way (comparing risks of freeware to risks of commercial software), the difference in the contingency reserve that's required for freeware always makes commercial software the obvious choice.

HIDDEN COST #3: LOWER QUALITY

In some cases, freeware is simply lower-quality software. Low-quality software has far-reaching effects, not only on your project but also on your organization. It can directly impact your project costs and schedule, and it can also cause ongoing problems for your organization once your software system is put into production.

A case in point: I remember working on a web application for a product management team, where we decided to leverage some freeware. During system testing, our application would crash at random points. We struggled to isolate the cause and trace the exact nature of the crash. However, we did narrow it down to the low-quality freeware that was just not performing properly. We were never able to understand why the code was failing. And we lost weeks of development time in first trying first to diagnose the problem, then in writing layers of code to compensate for the shortcomings of the freeware.

The costs can rise exponentially when undetected bugs from low-quality software are bundled with the ultimate solution you deploy in the organization. When a bug is uncovered in development, you might lose a few development cycles. But if a group of end users uncover a bug, it needs to go through an entire corporate maintenance and improvement cycle. This is a very costly and recurring problem that can come with freeware.

Don't get me wrong. Some commercial software is of low quality. However, as a rule, you'll find more problems with freeware than you would with commercial software. Also, your chances of recovering from maintenance issues are much better with commercial software, especially if you build a good relationship with your vendors.

Also, you have a better chance of producing higher-quality code with commercial software development tools. Freeware is often deployed with only basic functionality. Typically, the features in a software development tool that allow you to produce higher quality software (like advanced formatting, bug proofing, code tuning and suggestions for best practices) are found in the higher-end, fee-based software.

YOU GET WHAT YOU PAY FOR IN FREEWARE

Obviously, I've made a lot of assertions in this document that put freeware in an unfavorable light. As I mentioned earlier, freeware is not a bad thing. But if you're trying to run a large project or a program, you must be aware of the hidden costs of productivity, risk, and quality that that exist in most freeware.

I've been a database and web application developer for more than 20 years, managing development efforts for most of that time. I've certainly benefited from freeware and the communities that support its development. However, I've also incurred costs that are often very difficult to detect.

Whenever I work with management on driving change, one of the first things I look at is how the workforce is being compensated. That's because people are fundamentally motivated by what they're paid to do. The same holds true for software development. When software is created for something other than a commercial purpose, it becomes at best a secondary concern.

There are a couple of different ways freeware comes into the marketplace. The most common way is when an organic group of developers band together on their free time to build software for the greater good of the software community. This is a popular mentality among software developers and it has brought us some of the best freeware available. You have a reasonable chance of achieving quality software over time using this method, but there's really no guarantee. Since it is all volunteer work, there are generally no barriers to entry, so the quality of developers joining the effort is quite random. Quality control is pretty much at the discretion of the development team, so you could very possibly be inheriting buggy software.

Support is the biggest concern however, as there's no way to structure a service level agreement. This means that when your developers need help understanding something, or worse yet, dealing with bugs in the freeware, they will be referred to an online forum of volunteer developers with questionable credentials and no real obligation to respond. Slow response times will impact your team's productivity and the quality of support will present a higher risk for the development effort.

The other way we see freeware enter the marketplace is through commercial companies that either offer "free" software as a loss leader, bundle "free" software into a larger commercial software license or offer a limited version of their software for free with the hope that you will upgrade to a commercial license. In all cases, quality and functionality can suffer.

Having consulted at companies that employ all three strategies, I can tell you with great confidence that they will direct their key resources to their money-making products, not their freeware offerings. Of course, this is not what the marketing brochure says, but it's the reality of software development.

Also, the lower levels of functionality in the freeware offerings will certainly hamper productivity. I remember one project where we tried to use the free version of Toad® for our development; however, there was a restriction on the free version that only allowed a handful of developers to be online at the same time. Obviously, this was not practical, so we upgraded to a commercial license.

CONCLUSION: COMMERCIAL SOFTWARE IS THE CLEAR CHOICE

I hope this document opened your eyes to a few things to take into consideration when using or considering freeware. Focusing on the big picture will prevent you from falling victim to the common traps associated with free tools. If your goal is to produce solutions at the lowest overall cost within the shortest timeframe, then the cost of a software license is not your biggest concern; it's getting control of productivity, quality, and risk.

These are somewhat elusive costs, but the astute manager will take my advice and ignore the siren's song by investing in the best possible commercial software licenses—a very small price to pay for avoiding a development shipwreck.

ABOUT THE AUTHOR

John Weathington is President and CEO of Excellent Management Systems, Inc., a management consultancy that helps companies dramatically improve efficiency, and avoid penalties and fines. He has over 15 years experience in business intelligence and data warehousing, is an accomplished project management professional (PMP) and Six Sigma Black Belt, and has consulted to Fortune 500 companies such as Cisco, Silicon Graphics, eBay, and Sun Microsystems. Recently he helped a large technology firm fortify a $100 million government contract. For more information, please visit http://www.excellentmanagementsystems.com.

ABOUT QUEST SOFTWARE, INC.

Quest Software, Inc., a leading enterprise systems management vendor, delivers innovative products that help organizations get more performance and productivity from their applications, databases, Windows infrastructure and virtual environments. Quest also provides customers with client management through its ScriptLogic subsidiary and server virtualization management through its Vizioncore subsidiary. Through a deep expertise in IT operations and a continued focus on what works best, Quest helps more than 100,000 customers worldwide meet higher expectations for enterprise IT. Visit www.quest.com for more information.

Contacting Quest Software

Phone:	949.754.8000 (United States and Canada)
Email:	info@quest.com
Mail:	Quest Software, Inc. World Headquarters 5 Polaris Way Aliso Viejo, CA 92656 USA
Web site:	www.quest.com

Please refer to our Web site for regional and international office information.

Contacting Quest Support

Quest Support is available to customers who have a trial version of a Quest product or who have purchased a commercial version and have a valid maintenance contract. Quest Support provides around the clock coverage with SupportLink, our web self-service. Visit SupportLink at http://support.quest.com

From SupportLink, you can do the following:

- Quickly find thousands of solutions (Knowledgebase articles/documents).
- Download patches and upgrades.
- Seek help from a Support engineer.
- Log and update your case, and check its status.

View the ***Global Support Guide*** for a detailed explanation of support programs, online services, contact information, and policy and procedures. The guide is available at: http://support.quest.com/pdfs/Global Support Guide.pdf