10X Development

The Effectiveness and Economics of High Performance Software Teams

Kenneth Ledeen,
Peter Jaffe,
Edward Nesson

Executive Summary

A High Performance Software Team is remarkably similar to a winning sports team. It needs talent, leadership, structure, and spirit. Just like a winning sports team, it doesn't need to be filled with superstars, but it can't have any laggards either. The 1983 Celtics had Larry Bird and Robert Parish, but it took more than two stars to win. (Van Riper, 2008) Talent is a necessary, but not sufficient condition for success. Like sports teams, High Performance Software Teams also need leadership, strategy, discipline, and coordination. They need to know how to work together as a cohesive unit. Consequently, it often takes time for an HPT to mature, to find the winning groove.

High Performance Software Teams deliver - they deliver better code, faster, at lower cost, with fewer bugs, and dramatically lower maintenance cost and complexity. Most importantly, they deliver predictable results, on time and on budget, a rare combination in the world of application software development.

High Performance Software Teams combine developers with unique capabilities and experience, advanced software engineering methods, and sophisticated project management practices to produce dramatically better results than its conventional counterparts. The productivity increase is often **more than 10 times** that of conventional software teams.

There is a sweet spot for High Performance Teams. They are not well suited for ongoing software maintenance and support and provide little advantage during much of the application's life. Consequently, they are only one part of a total approach to the application software life cycle.

The cost of creating and supporting such a team, the relatively short and bounded development cycle, and the episodic nature of the need for the team make outsourcing this function economically advantageous both for established firms for whom software is an adjunct to their core business, as well as for companies seeking to develop new software solutions.

This paper explore both the benefits and costs of High Performance Teams. It presents some of the research on their behavior, and addresses how be to create and, manage them, as well as considering the economics of outsourcing new application development to an organize that specializes in that particular function.

Background

For more than 40 years it has been generally known that great programmers dramatically outperform average ones. The findings from the first study in 1968 have been tested, revised, and repeatedly substantiated. The differences are enormous. Great programmers aren't 50% better than average ones, they are routinely 1000% better. They not only get their work done sooner, but their code is easier to maintain, and often runs significantly faster, and uses less computing resources. Googling the topic turns up plenty of references (Martin) (Veksler) (Spolsky) (Glass, 2002) (Haack, 2007).

Bill Gates has been widely quoted as saying:

> *"A great lathe operator commands several times the wage of an average lathe operator, but a great writer of software code is worth 10,000 times the price of an average software writer."*

The original study considered coding, debugging, program size, and run time in a controlled test group for two different test problems. The results below show the ratio between the best and worst developers (Sackman, 1968).

	Problem 1	**Problem 2**
Coding Time	16:1	25:1
Debug Time	28:1	26:1
Size of Program	6:1	5:1
CPU Time	8:1	11:1
Run Time	5:1	13:1

In an article titled "Hitting the High Notes," Joel Spolsky describes a study done at Yale that measured the performance of students in an advanced programming class. Even looking at just the top 25% of the students in what is by definition already a highly selective group, the variations were still 10X (Spolsky).

The performance and productivity advantages of individuals is mirrored in the performance of teams as well. High Performance Teams dramatically outperform conventional ones. Steve McConnell, the author of seven highly respected books on software development, discussed team productivity in his blog titled "10X Software Development". Among many other references, he describes a comparison of the Microsoft and Lotus development teams:

> *Excel took 50 staff years to produce 649,000 lines of code. Lotus 123 took 260 staff years to produce 400,000 lines of code. Excel's team produced about 13,000 lines of code per staff year. Lotus's team produced 1,500 lines of code per staff year. The difference in productivity between the two teams was more than a factor of 8, which supports the general claim of order-of-magnitude differences not just between different individuals but also between different project teams (McConnell).*

The implication seems obvious – hire great programmers, build great teams, get great results. It turns out

that things aren't quite that simple. For starters, a high performance team filled with A-level developers is not always the right solution. In many cases it isn't. And, to make matters worse, hiring great developers, retaining those developers, and building great teams is not all that simple. If it were, everyone would do it. This paper attempts to dissect the process, looking at what it takes to build a High Performance Team, along with when you should use one and when you should not.

What is a "High Performance Software Team?"

High Performance Teams require six critical components. Leave out any one of the six and the performance of the team will suffer. They are:

1. Great developers – this is a necessary but not sufficient component. Not everyone has to be the next Bill Gates. Everyone does, however, have to share the core characteristics of a deep understanding of software principles, passion, strong work ethic, and solid communication skills.
2. A collegial working environment, one that encourages consultation, shared problem solving, and collaboration. This is VERY important. For starters, a collegial environment raises the skills and productivity of the less capable and experienced staff. Software problems are very hard and complex. Even the most talented developers benefit enormously from the ability to confer with colleagues as a way to identify alternatives that might not otherwise have been considered. You can't do this if the overall atmosphere is wrong.
3. "Appropriate" project processes. Bad process can kill productivity fast. Good process fosters communication, empowers the team, frees management, and brings clarity. Processes can be complex, stretching from requirements gathering through continuous integration and release management.
4. Challenging problems. Having a continuing stream of challenging problems is important for two reasons. First, because great developers want to solve interesting problems, just like top athletes like to be on winning teams. And second, because the productivity advantages of the HPT are greater the more complex and challenging the task.
5. Teamwork - the HPT excels because the whole is greater than than sum of its parts. Teamwork comes from the intersection of the right players, the right leadership, and the right culture.
6. Knowledgeable Management, at multiple levels. A High Performance Team is like a race car. It can go fast, but in the wrong hands it can end up in a wreck in no time.

The Sabremetrics Analogy

For most of the modern baseball era, the prevailing thinking was that the way to have a winning team was to find (or in the base of the Yankees, to buy) the very best players. The theory was that top talent wins games. There is obviously some truth to that. You couldn't field a team with a bunch of mediocre players and hope to win.

The thinking changed, though, as more thorough analysis of performance suggested an alternative approach. As described in Michael Lewis' *Moneyball*, Billy Beane of the Oakland Athletics assembled a winning team without acquiring top talent by applying these principles, combining solid, balanced team, with process and strategy.

Theo Epstein, manager of the Boston Redsox, hired Bill James, a sabremetrician, to do the same, and reversed 86 World Series drought.

What does this have to do with High Performance Software Teams? The guiding principles are the same: find the best talent you can, but not only super stars, then add leadership, strategy, careful management, experience, and process.

The Complexity Spectrum

A High Performance Team is not universally better than more conventional ones. It depends a lot on the task at hand. This list considers a range of activities with estimates, based on both the literature and our own experience, of the relative gain of a High Performance Team.

Activity	Gain (time)	Comment
Complex Debugging	100:1	For complex problems, it is not uncommon for average developers to be stymied and be literally unable to find the cause and the solution.
Architecture / Design	20:1	It isn't merely a question of who gets done faster, but more the quality of the work that is completed. This is an aspect of application development where very small incremental costs yield enormous returns. If the design is done right, much of the coding can be done with less talent developers.
Framework Development	15:1	Much of the success of new application development hinges on the implementation of the patterns and templates that will form much of the basis for the application. The application "framework" addresses detailed issues like persistence, event and error handling, as well as UI implementation pattern. This is the implementation component of Architecture and Design.
New Application Development	10:1	This is the phase of a project where the patterns and templates that were implemented in the Framework are used to build out the initial set of features. The relative advantage of the HPT derives from the need to evolve and refine the design patterns.
Major Application Extensions	3:1	Simpler than starting from scratch, adding major extensions to an application often requires understanding the consequences and implications of changes, or, alternatively, envisioning how existing capabilities can be abstracted and reused.
New Feature Development	2:1	These are areas where innovation is less significant and existing patterns can be applied.
Feature Extensions	1:1	Incremental enhancements mostly involve following existing patterns. It is worth noting, though, that the HPT often produces code that is cleaner, more consistent with the design patterns, and hence easier to maintain.
Minor bug fixing	1:1	An inappropriate use for an HPT.
Configuration / Implementation	1:1	The ratio is less than one here because most HPT members simply don't want to do this sort of work

Not Everyone Needs Their Own High Performance Team

And few organizations need them all the time!

If great developers outperform average ones by 10X and command salaries that are only 20% more than their less productive colleagues, then why wouldn't every organization strive to hire only the very best. The answer is simple. First, there aren't enough great developers to go around. Like any activity, only 10% of the developers are in the top 10%. There is far to much work to be done to relegate it all to this small group. Second, the majority of software development tasks not only don't require the talents of an HPT, but many projects are a poor match to the skills, pace, and general makeup of the HPT. A Formula 1 race car is great for winning races, and for proving new technologies, but not for going shopping or taking the kids to soccer practice.

Most product organizations only need the HPT for the early releases of their product or when a major restructuring or extension is undertaken. The ongoing maintenance and support, feature extensions and additions can all be handled more effectively by a conventional team. The analysis presented in Appendix A explores the differences with quantitative detail. The HPT that did the initial development may be refocused on other projects, but often their talents and energies are dissipated if the staff continue on in an organization that can't effectively utilize them.

Similarly, most corporate IT environments only realize the advantages of an HPT when major new applications are to be developed or substantial additions to functionality are required. For large organizations, new application development is often substantially less than 10% of the total IT budget, dwarfed by operational costs and the ongoing support and maintenance of existing applications.

The obvious conclusion is that organizations should make sure that you really NEED an HPT before you try to build and sustain one, particularly given the time and cost involved.

Make Versus Buy

Several factors influence the make versus buy decision.

Time to establish the team	In the best of cases, it will take approximately six months to put a team in place if the critical mass, the nucleus that includes some top-notch talent, is not already present. If you are lucky, you might hire someone who can bring along some former colleagues, but it's not prudent to plan on that. If you are not hiring people you know, then the time frame gets even longer because we all make hiring mistakes.
Cost to establish the team	Estimates range from $100,000 to $300,000 for a fixed hiring cost per developer. These costs include the level of effort required by senior staff in the recruiting and interviewing process, possible fees or hiring bonuses, the time that new hires take before they are fully productive, drain on your existing technical staff to integrate and mentor new hires, and a factor that accounts for the inevitable bad hires.
Long term needs	If you only have episodic requirements for major new development; if the bulk of what your department does is maintain systems that are in production, add minor features, or customize capabilities for new clients; then it is likely that an internal HPT will dissipate its energy, lose interest, and not be sustained. If, conversely, you

	expect that there will be a constant stream of new and varied projects, then by all means, build such a capability internally.
Corporate Culture	This is by far the most subtle aspect. Some organizations are simply not well suited to hosting an HPT internally.

The Economics of High Performance Teams

For this analysis we have assumed a blended external rate of $175/hour, and an internal burdened cost of $78/hr[1]. Appendix A considers two scenarios – new application development and ongoing enhancement, maintenance, and implementation.

Despite a cost premium of more than 2:1, the outsourced HPT is likely to cost just slightly more than 1/4 what it would cost for an internal team that does not have comparable skills and experience. More importantly, it will require almost eight times the effort. Appendix A explores the details of this analysis in the context of both new application development and ongoing maintenance and support.

For ongoing development, the pattern is largely reversed. The HPT costs significantly more for a project of comparable scale, largely because it has no real advantage once the design patterns are well established and stability has been achieved.

The most common fallacy we encounter is the assumption that there are savings to be had simply by switching from external to internal resources. That is true, but only if it is done at the right point in the project, and for the right tasks. Otherwise, the entire success of the project is likely to be jeopardized and costs will almost certainly spiral out of control.

The ideal solution is a hybrid, combining internal resources with an external HPT. The internal resources participate in all aspects of the initial design and development so that they can establish a deep understanding of how the new system works and can assume responsibility at the earliest possible appropriate time.

Conclusions

The secret ingredient in successful application development projects is engaging a High Performance Team. The economics are often counter-intutive. An outsourced HPT billing at rates 4X the internal staffing cost can routinely deliver better results, more predictably, with code that is easier to maintain, in less time, for lower total cost. Go figure.

[1] Calculated as follows: Assumes average salary of $100,000, 35% overhead including benefits, 3 weeks of vacation, allowance for professional development and other corporate activities of 10 days/year, 10 days of holidays.

Appendix A - Cost Analysis

Comparison of Outsourced HPT and Internal Conventional Teams

The tables below present an analysis of the relative cost of an outsourced High Performance Team and an internal conventional team. In many instances organizations will combine elements of both teams, using the HPT for the tasks for which they are best suited, and including members of the conventional team as an integral part of the development effort from the outset, so that a deep understanding of the project architecture and design is preserved.

The projects differ in the amount of new architecture, design, and conceptual innovation that constitute them. The first, labeled "New Development," is typical of a project that is starting from scratch, either for a new application, or possibly for a new implementation of an existing one.

The second project, labeled "On-going Development and Implementation" is typical of what occurs in subsequent years with both internally used applications and software products. In this instance the technology framework is in place, the design patterns are set, and the relative advantage of the HPT is substantially less.

Both projects were scaled at 4,000 units of HPT effort. The Conventional Team efforts were derived using the relative performance ratios. HPT cost per hour of $175 is typical for the industry. The internal team assumes a fully burdened cost of $78 per hour based on a median salary of $90,000, standard benefits, vacation, and 10% time directly related to the project.

Column	Contents
A	Relative advantage of a high performance team over a conventional one
B	Percentage of a "typical" project that falls into this category
C	Number of hours required for a HPT to complete
D	Cost for HPT
E	Number of hours for a conventional team
F	Cost for a conventional team

New Development

	A	B	C	D	E	F
Architecture / Design	20.00	10%	400	70,000	8,000	624,000
Framework Development	15.00	15%	600	105,000	9,000	702,000
New Application Development	10.00	30%	1,200	210,000	12,000	936,000
Major Application Extensions	3.00	15%	600	105,000	1,800	140,400
New Feature Development	2.00	10%	400	70,000	800	62,400
Feature Extensions	1.00	6%	240	42,000	240	18,720
Minor bug fixing	1.00	6%	240	42,000	240	18,720
Configuration / Implementation	1.00	8%	320	56,000	320	24,960
Totals		100%	4,000	700,000	32,400	2,527,200

The total cost for the project done entirely by a High Performance Team is $ 700,000. Despite dramatically lower cost per hour for their staff, a Conventional Team total cost is $2,537,200, more than 3 times the HPT cost.

The difference arises from the concentration of architecture, design, and framework development activities. In practice we strongly recommend that the development effort\include internal staff not because the costs are lowered but rather because the transfer of information to the team who will have long term responsibility is far more effective if they are involved from the outset.

Ongoing Development, Implementation

	A	B	C	D	E	F
Architecture / Design	20.00	0%	0	0	0	0
Framework Development	15.00	0%	0	0	0	0
New Application Development	10.00	0%	0	0	0	0
Major Application Extensions	3.00	10%	400	70,000	1,200	93,600
New Feature Development	2.00	20%	800	140,000	1,600	124,800
Feature Extensions	1.00	20%	800	140,000	800	62,400
Minor bug fixing	1.00	15%	600	105,000	600	46,800
Configuration / Implementation	1.00	35%	1,400	245,000	1,400	109,200
Totals		100%	4,000	700,000	5,600	436,800

The total cost for the project done entirely by a High Performance Team is $ 700,000. Despite dramatically lower cost per hour for their staff, a Conventional Team total cost is $436,800, significantly lower than the cost of the HPT.

Bibliography

Glass, R. L. (2002). *Facts and Fallacies of Software Engineering.* Addison-Wesley.

Haack, P. J. (2007, June 25). *10 Developers for the Price of one.* Retrieved from haacked.com: http://haacked.com/archive/2007/06/25/understanding-productivity-differences-between-developers.aspx

Lant, M. (n.d.). *How to Build a High Performance Agile Team.* Retrieved from michaellant.com: michaellant.com/2010/08/26/how-to-build-a-high-performance-agile-team/

Martin, T. (n.d.). *The Tenfinity Factor.* Retrieved from DevTopics: http://www.devtopics.com/programmer-productivity-the-tenfinity-factor/

McConnell, S. (n.d.). *10X Software Development.* Retrieved from Construx.com: http://forums.construx.com/blogs/stevemcc/archive/2008/03/27/productivity-variations-among-software-developers-and-teams-the-origin-of-quot-10x-quot.aspx

Sackman, H. W. (1968). xploratory Experimental Studies Comparing Online and Offline Programming Performance. *Communications of the ACM , Vol 1 no. 1*, 3-11.

Spolsky, J. (n.d.). *Hitting the High Notes*. Retrieved from joelOnSoftware.com: http://www.joelonsoftware.com/articles/HighNotes.html

Van Riper, T. (2008, 1 29). *Greatest Sports Teams.* Retrieved from forbes.com: http://www.forbes.com/2008/01/29/greatest-sports-teams-biz-sports-champions08-cx_tvr_0129greatestteams.html

Veksler, D. (n.d.). *Some Lesser Known Truths about programming.* Retrieved from Dot Mac: http://dotmac.rationalmind.net/2010/08/some-lesser-known-truths-about-programming/

About the Authors

Kenneth Ledeen	Is the founder and Chief Executive Officer of Nevo Technologies, Inc,. In addition to his work there, he is the co-author, with Harry Lewis and Hal Abelson, of *Blown to Bits: Your Life, Liberty and Happiness After the Digital Explosion,* a detailed exploration of the pervasive impact of digital technologies on modern life. Ken has been developing sofware and leading software organizations since the mid 1960s. He graduated from Harvard College in 1967.
Peter Jaffe	Is the Chief Operating Officer at Nevo Technologies. Pete has direct responsibility for building Nevo's High Performance Team and for putting in place the processes that enable it to function so effectively. He joined Nevo in 1998 after completing both undergraduate and graduate work at MIT.
Edward (Ted) Nesson	Is a Principal at Nevo Techologies where he has had leadership responsibilities for a broad range of projects. Before joining Nevo, Ted was the Senior Vice President for Engineering at Gemstar/TV Guide, where he led their efforts to develop the digital products and services that have replaced their magazine. Ted received his AB, Masters and PhD degrees from Harvard University

About Nevo

Founded in 1997, Nevo provides software development services to a broad range of clients in a diverse array of industries. Our work has spanned higher education, manufacturing, energy conservation, financial services health care and more.

Nevo is the embodiment of the principles outlined in this paper. We have built a High Performance Team by combining three key elements:

- Expertise - we hire the very best people we can find, and then immerse them in an environment that is carefully crafted to maximize their ability to function effectively.
- Experience - nothing quite matches the advantage that comes from doing sometihing a second, third, or fifteenth time. New clients benefit from the knowledge gained throughout Nevo's fifteen year history
- Process - Delivering predicable and repeatable results is only possible if a set of project principles, tools, practices, and procedures. We often adapt to the specific needs of individual clients, but always within the context of a process that has been proven to deliver.