



## Perfect Coaching: How Companies Are Maximizing Software Delivery ROI through Just-In-Time Training

Software development teams are being asked to be more productive and efficient—to do more with less. Today's environment mandates that companies control costs and keep business-critical applications aligned with ever-changing business strategies to meet competitive threats. As companies begin to exhaust the more obvious cost-cutting options they must shift their focus toward the goal of creating sustainable productivity gains in their software development efforts. Companies must consistently upgrade the skills of their staff as an integral part of their long-term success strategy.

The next phase of optimizing and integrating software development staff skills and processes has begun. As companies reflect on past software development efforts and attempt to realign IT within the overall organization, they are realizing that the quality of their software development teams greatly impacts immediate and long-term costs and directly effects revenue growth. Companies recognize that having average or good software developers is not enough. In short, companies are realizing that great software developers are critical to the overall health of an organization.

Great software developers are more productive. In fact, studies have shown that they are 10 – 20 times more productive than average developers in delivering software features quickly and efficiently. The higher quality code they produce makes a distinct difference in the long-term costs of maintaining a software application. With long-term maintenance representing the largest portion of the cost of developing software, poorly designed code increases that expense by making it difficult for developers to modify the code as business needs change and new requests are made. This dramatic productivity difference between average and great developers translates to a higher return on the investment in creating and retaining great developers.

---

But great software developers are not developed by accident. In order to become great, software developers must be exposed to frequent combinations of practical training in the fundamentals and experiential training by directly applying what they've learned in lab or classroom settings.

To date, most Companies' attempts to achieve the elusive goals of developer training, retention and increased software development ROI have fallen short. Most managers have relied on traditional means of training, such as e-learning and classroom training or outsourcing and knowledge transfer. These methods of training and delivery are important, but have some fundamental flaws and disadvantages. Based on these attempts, and by leveraging new methods for integrating learning and delivery, a better way has emerged.

A new model of "developer coaching" is fast becoming the most effective method for creating great software developers. Coaching is a training technique focused on teaching developers what they need and when they need it, while delivering working code. By incorporating the basic principles of learning at the side of a software development expert, coaching overcomes many of the obstacles presented by traditional approaches to training. The quality of learning is higher and the process is much more cost-effective.

The information included in this paper will help you to understand how you can use coaching at your organization to create great developers while delivering needed software features to the organization. It discusses coaching techniques, as well as the advantages of using a coach versus a traditional training or delivery approach. Finally, this document will compare the return on investment of both traditional training and delivery approaches with a coaching approach and discuss some specifics of a typical coaching model.

## **The need for software development training**

Simply cutting training from the IT budget is a reflection of companies questioning the real value of traditional training but it's not a permanent solution. Instead, companies need to focus on ways to use training budgets with more precision.

There are a number of risks associated with merely dropping training from the budget rather than redirecting the investment to optimize knowledge gain and software delivery.

- Reducing training opportunities degrades the overall skill level of development staff.
- Reduced learning opportunities create lower job satisfaction, as most developers feel less challenged.
- Lower skill levels negatively impact the quality of software.
- Lower software quality increases the long-term maintenance costs of software applications.

- 
- Lower skill levels negatively impact the productivity of software developers.
  - Lower productivity slows the pace of delivering software features to the organization.
  - The inability to consistently deliver quality software and high turnover ultimately increases the dependency on outsourcing.
  - Outsourcing and limited opportunities for improvement create lower morale that results in higher developer turnover.
  - Higher turnover results not only in a loss of internal technical knowledge but also in valuable business domain knowledge.

While it's obvious that reducing training opportunities and increasing outsourcing are risky and at best, a short-term solution to organizational desires to become more self-reliant and cost-effective, companies are justifiably reacting to unsatisfactory returns on investments in traditional methods of training and delivery that have proven to be inefficient and ineffective.

## What's wrong with traditional approaches?

Two approaches have traditionally been used to deliver and maintain software within companies. The first is to send development staff to classroom or lab-based training either in person or via a self-directed approach such as distance learning or computer-based training. Students are then expected to use their newly acquired knowledge to complete development projects or tasks at the organization. The second approach is to outsource a project to a firm possessing the necessary skills to deliver what's needed and then attempt to transfer knowledge about the delivered software to internal development staff. Both approaches have significant flaws and even when combinations of both approaches are attempted, the results are seldom satisfactory.

While classroom or lab-based training tends to cover core fundamentals that developers need, this approach suffers from a variety of shortcomings whether it's instructor led or self-directed.

*First, we look at the classroom approach:*

### **1. When you send developers to school they solve lab problems. When you teach them something new on the job, they solve real business problems.**

Lab exercises do not effectively apply to the real-world because they are lab-based. The examples used for skill practice are overly simplistic and not grounded in the often complicated problems of real-world development projects. Worse, they have no application to the developers' current (often behind schedule) project, which they will return to face as soon as class ends. The class material is so out-of-context that it is difficult to effectively apply the lessons learned at the workplace.

---

## **2. The value of classroom training received depreciates as the time-until-use increases.**

Delays in the application of knowledge received also leads to tremendous inefficiencies. Developers may attend a class or complete an on-line tutorial with good intentions of applying new knowledge when they return to work but more often than not, other job demands begin to erode any chances of retaining what was just learned. Studies have shown that months or even weeks later, as much as 80% of new knowledge evaporates because it is not practiced and applied immediately.

## **3. Feedback, Feedback, Feedback**

Even in circumstances where developers are able to immediately apply what they've learned on a real-world project, other challenges will present themselves. Without immediate validation or feedback from an expert as questions arise or road blocks occur, developer progress will often stall or worse yet, they may attempt to implement software features "as best they can" creating lower quality code, higher maintenance costs and as a result, lower self-confidence.

## **4. One size does not fit all**

Traditional training cannot tailor the learning style to fit every participant. Every learner comes to conclusions and understands technology at a different pace and style, whether it's kinesthetic, visual, or auditory. Most curricula are designed for the lowest common denominator, which means that it is geared toward novice participants who learn in an audio format. Faster learners, and learners who prefer a visual or kinesthetic learning experience, lose out, leading to wasted training dollars.

*The second approach is to outsource a project:*

Outsourcing a project is the second method routinely used by companies to deliver and maintain software. The motivation is clear: outside firms typically possess the necessary skills needed to complete a project.

### **1. The Big Bang Approach**

This "big bang" approach tends to leave people and process behind by leaving internal staff ill-equipped to move forward with on-going development and support. The oft-touted knowledge transfer commonly attempted at the end of the project is insufficient regardless of any training attended by internal staff in advance of the transfer. Once the experts leave and internal staff officially inherits what's been delivered, developers are left with same problems as if they had attempted to build the system themselves.

---

## 2. Outsourcing Costs More

Dollar-for-dollar and person-for-person, outsourcing costs more than having internal staff complete projects. If internal developers had the skills and, of course, the time, projects would not need to be outsourced, at least not as fully or as frequently.

By themselves, neither classroom training or outsourcing with knowledge transfer deliver satisfactory results, but there is an approach that offers an optimal blend of both approaches at a lower cost.

### Coaching: Maximizing software delivery ROI

Coaching is a learning and delivery model that utilizes frequent and direct personal interaction to facilitate learning in the context of a real-life software project. In this approach, a software development expert, “The Coach”, works directly with one or more developers to complete a software project or a number of tasks. The coach works side-by-side with developers to gather requirements, design architecture, write lines of code and deliver the software features to end-users.

Coaching is a blended training and delivery approach offering an optimal combination of traditional classroom training and provides a number of benefits.

- Coaching integrates the three keys to learning for software developers
  - ♦ Classroom approach – concepts and fundamentals
  - ♦ Experiential learning – real-world application of concepts
  - ♦ Self-development – individual responsibility and motivation for learning
- Learn alongside an expert

An expert is always available to provide feedback and deliver knowledge that is relevant to the task at hand. Depending on the nature of work, other experts can participate at various points in the project.
- Learn using your data in your technical environment while implementing features designed for your company

Developers will solve real-world problems with newly acquired skills working on features requested by end-users.
- Learn what you need, when you need it

Coaching enables a just-in-time or on-demand approach. Learn what you need, when you need it and when you’re ready to apply it.

- Coaching delivers highly-practical knowledge and working code  
Learning takes place in the precise context of the current task or project goal. The organization benefits from modernized development skills, motivated developers and actual working, high-quality features that the organization needs.
- Team environment  
The collaborative approach required by coaching helps foster a team environment, as everyone is responsible for the success of the project.
- Job satisfaction  
Immediate and personalized feedback allows developers to take risks using the safety net of the coach. As a result, developers build greater confidence and job satisfaction, which reduces burnout and turnover.

These benefits translate into an immediate and long-term positive ROI. The below examples show how coaching delivers more—at a lower cost—than other approaches to learning and delivery.

### Example #1: Classroom Training

A company decides to send three developers from its staff to a four-day training class to acquire skills needed for a new software application they're expected to work on.

Four-day classroom training fees for three developers	$\$1600 \times 3$	=	\$4,800
Time off for three developers for four days @ an hourly rate of \$50/hr, including benefits and taxes	4 days (8 hrs) x \$50/hr x 3 developers	=	\$4,800
Value lost from the three developers away from work equals 50% of hourly rate	50% of [4 days (8 hrs) x \$50/hr x 3 developers]	=	\$2,400
Cost of three developers working on a two-month, full-time project	320 hrs x \$50/hr x 3 developers	=	\$48,000
Cost of re-work, throw-away effort and long-term maintenance of sub-par code for the three developers	50% of development cost		\$24,000
	<b>Total cost</b>		<b>\$84,000</b>

### Example #2: Outsource and Knowledge Transfer

A company decides to hire an outside firm to develop the new software application and perform a knowledge transfer to three developers from its internal staff.

Cost to hire an outside firm and use two of their developers for the same two-month project from Example #1	320 hrs x \$100/hr x 2 outside developers	=	\$64,000
Cost of re-work, throw-away effort and long-term maintenance by the three internal staff developers inheriting the outsourced application	50% of development cost	=	\$32,000
<b>Total cost</b>			<b>\$96,000</b>

### Example #3: Coaching Engagement

A company decides to hire a coach to assist and train three developers from its internal staff to develop the new software application.

Cost to hire one coach to work with three developers for two months on a project	\$125/h x 320 hrs	=	\$40,000
Cost of three developers working on a two-month, full-time project	320 hrs x \$50/hr x 3 developers	=	\$48,000
Reduced cost of re-work, throw-away effort and long-term maintenance by the three internal staff developers as opposed to the approaches in Examples #1 and #2	50% of three internal developers' efforts	=	(\$24,000)
<b>Total cost</b>			<b>\$64,000</b>

In the final analysis, coaching yields a superior ROI when compared with traditional training and delivery options. In fact, a typical coaching engagement costs 30%-50% less than a traditional classroom learning experience or outsourcing project. It also delivers far more value in terms of long-term turnover and development costs. Why is this the case?

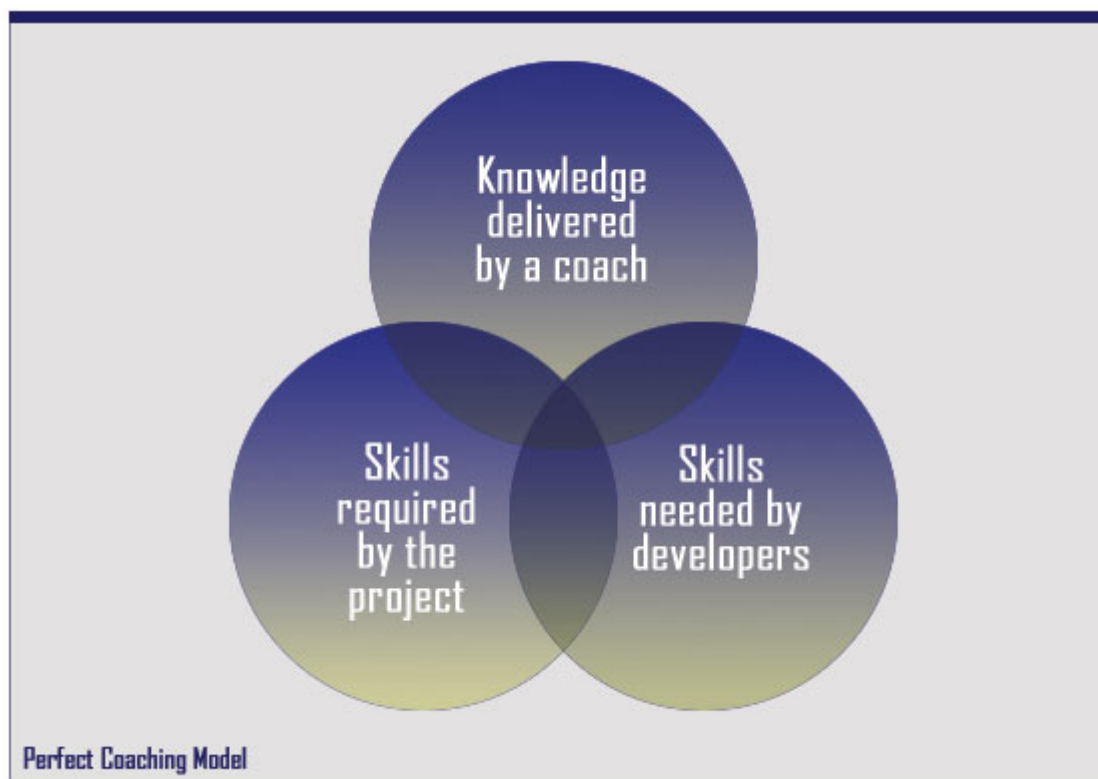
Coaches help offset the cost of lost productivity caused by traditional classroom training. With a coach, the developers do not need to leave their work site or take time away from tasks for self-directed study.

---

More importantly, the expensive cost of experimentation and error is greatly reduced and turned into learning opportunities, as developers immediately become productive members of the team.

### How does coaching work?

Coaching success is measured by the modernization of skills, process improvements and the completion of tasks such as new software features or modification requests. Knowledge transfer is continuous, not an afterthought. The diagram below illustrates the systematic focus on continuously aligning the skills required by the project with the skills needed by developers and the necessary knowledge the coach will deliver.



A coaching engagement can last a few weeks, a number of months or longer depending on variables such as the skill level of the staff and the project goals. A coach works with developers on a daily basis and becomes a member of the team, attending team meetings and helping complete task assignments. Work is typically organized within one or two-week iterations to allow time for reflection and planning before and after each iteration concludes.

---

There are a number of repeatable steps and processes critical to the success of coaching that are invoked by the mantra: Educate, Act, Reflect, Review, Repeat.

- Define tasks or projects that offer the optimal mix of learning and overall value to the organization
  - ♦ The coach meets with the IT or development manager to discuss project goals and each individual team member's skills and experiences.
  - ♦ The coach meets with each individual on the team to discuss skills, goals and experience.
  - ♦ Development tasks are prioritized based on a mix of value to the organization and learning opportunities. Tasks are selected so that they stretch people but do not derail them.
  - ♦ Team members are assigned to tasks and effort is organized across a number of one or two-week iterations.
- Before the first iteration begins, fundamentals are delivered via courseware content in a structured classroom environment. Content will be specific to the upcoming development iterations with some overview of technical concepts relating to the broader project goals.
- At the beginning of each iteration, a combination of learning methods can be used including after-action reviews.
  - ♦ Formal fundamentals are delivered in a classroom setting to establish the context for the development about to be attempted in the upcoming iteration.
  - ♦ Developers reinforce and increase learning by preparing for and presenting fundamentals or "lessons learned" pertaining to features they are working on. The coach and other team members reinforce the learning through group discussion and code reviews.
- During each iteration
  - ♦ At certain times, the coach and a developer tackle tasks by working side-by-side at the same computer. This allows the coach to get a sense of the developer's understanding and progress and provides the developer with critical reinforcement and feedback.
  - ♦ Progress is tracked, reported and discussed with each team member and manager involved in the effort. Frequent evaluation and feedback helps everyone chart the progress and value of the effort.
- To place focus on aspects of a developer's experience that they might not otherwise think about, the following questions are commonly asked.

- 
- ♦ What are you struggling with and what do you feel you accomplished?
  - ♦ What did you expect to happen in a particular situation and what actually happened?
  - ♦ What worked well and what didn't? Why?
- Throughout the entire process and across all iterations
    - ♦ Attempt to identify team members who could ultimately assume the role of coach.
    - ♦ Teach people self-development. Being "students of the game" and knowing how to learn on their own is also part of the bargain.
    - ♦ Create a blog or wiki to record and discuss challenges, successes or anything else worth noting throughout the process. This serves as informal documentation and memorializes tribal knowledge that often goes uncaptured.
    - ♦ Follow a development process or methodology as a part of coaching to instill best practices for estimating, planning and steering development efforts.

### **Optimal learning experiences lie somewhere between boredom and anxiety**

Developers need to go through an experience, think about what happened, theorize cause-and-effect connections, and use their mistakes and successes to apply their knowledge to new situations. Developers need to see the value of what they're learning. If they don't value it, they won't learn it. Coaching provides the environment that facilitates experiential learning.

Most developers don't have time to adequately process their job experiences and integrate new and familiar knowledge or they feel detached from learning mostly because the formal classroom process is so disconnected from real life. Most companies allow the richest opportunities for learning to go largely untapped. Coaching cures and integrates these classical ailments and turns learning into a deliverable.

Using a coaching approach, the development staff shows clear progress in increasing and improving their skills while delivering working code to end-users. In other words, coaching success means a completed project or tasks plus skills modernization plus process improvements.

Whether the organization has not implemented any new technology projects, has attempted and failed at such projects, has had marginal success, or has had moderate success at adopting new technology, adopting a coaching model will ensure that people and process are part and parcel of successful project implementations.

When companies fail to develop their best people through experiential learning, they miss the chance to build up the bench strength required to stay ahead of the competition.

---

Training and empowering internal staff boosts morale, increases employee retention and preserves vital business domain knowledge. Over the long haul it's more economical to have internal staff deliver software features.

Coaching has emerged as an optimal blend of traditional approaches to software development learning and delivery. Coaching costs 30%-50% less, and has a higher rate of effectiveness—leading to a far greater return on investment. Coaching creates an action learning environment where developers can safely test their skills while still completing needed software features and development tasks.

## About Visionpace

Visionpace is a software development company based in Kansas City, Missouri. Since 1992, the company has provided nationwide coaching, consulting and training services using Microsoft technologies. Their consultants provide expertise in the following areas:

- .NET
- Agile Methodologies
- B2B and E-Commerce
- BizTalk
- SharePoint
- SQL Server
- Test-Driven Development
- Visual FoxPro
- Web Design
- XML and Web Services

Visionpace has also developed an exclusive line of coaching services, called Perfect Coaching™. This coaching service is based on the research findings of this paper, as well as on extensive feedback from clients.

Visionpace coaches are dedicated to assisting every level of software developer to move forward and complete their assigned projects through the use of coaching. Visionpace coaches and developers are recognized as national experts through their publications in numerous technical journals and their speaking engagements at Microsoft and other public events.

## Let Us Design Your Perfect Coaching Engagement

Perfect Coaching™ from Visionpace is a software development approach that blends the best of classroom learning with practical on-site development. This approach strengthens your entire software development staff and turns them into a high-performance team. A Perfect Coaching engagement can improve the skills of your existing staff, enhance your

---

software development processes and deliver the tools your staff needs to be more effective in the context of their day-to-day work.

Our software development coaches are on top of the ever-changing world of software development. Perfect Coaching offers a unique way to train your staff in specific technologies and methodologies while they develop real-world, on-site applications. Our coaches work alongside your developers to deliver working software features that are critical to your business plans. Perfect Coaching is Expert Training with Immediate Application.

The best way to find out if this unique approach is for you is to complete a Coaching Review. For more information contact Visionpace at [coaching@visionpace.com](mailto:coaching@visionpace.com) or call 816-350-7900 or 888-904-7900 and ask to speak with someone about Perfect Coaching.



17501 East Hwy 40, Suite 218  
Independence, MO 64055  
Ph (816) 350-7900  
Fax (816) 373-3020  
[www.visionpace.com](http://www.visionpace.com)

© 2005 by Visionpace Inc.

All rights reserved. No part of this publication may be reproduced, photocopied, stored on a retrieval system, or transmitted without the prior written consent of Visionpace Incorporated. Visionpace, the Visionpace logo, and Perfect Coaching are trademarks or registered trademarks of Visionpace Incorporated. Information contained in this document is subject to change without notice.