

# Perfect Coaching: Maximizing ROI on Software Developer Training

White Paper  
April 28, 2005

**Visionpace**  
Software Development  
In Pace With Your Vision

Visionpace, Inc.  
17501 East Hwy 40, Suite 281  
Independence, MO 64055  
Ph (816) 350-7900  
Fax (816) 373-3020  
[www.visionpace.com](http://www.visionpace.com)

# TABLE OF CONTENTS

<b>INTRODUCTION: CREATING GREAT SOFTWARE DEVELOPERS.....</b>	<b>3</b>
<b>TRAINING SOFTWARE DEVELOPERS .....</b>	<b>4</b>
WHY TRADITIONAL TRAINING DOES NOT WORK .....	5
<b>COACHING: THE NEW MODEL .....</b>	<b>6</b>
<b>THE MAXIMUM RETURN ON INVESTMENT.....</b>	<b>7</b>
CASE STUDY #1: CLASSROOM TRAINING.....	7
CASE STUDY #2: COACHING ENGAGEMENT.....	7
THE FINAL ANALYSIS: COACHING V. TRADITIONAL TRAINING .....	8
<b>HOW DOES COACHING WORK?.....</b>	<b>9</b>
BEST-PRACTICE COACHING TECHNIQUES .....	10
<b>CONCLUSION .....</b>	<b>11</b>
<b>ABOUT VISIONPACE.....</b>	<b>11</b>
<b>BIBLIOGRAPHY .....</b>	<b>12</b>

© 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.

# INTRODUCTION: CREATING GREAT SOFTWARE DEVELOPERS

Great software developers are critical to any healthy IT organization. They earn billable revenue for us; they create excellent software programs; and they maintain our mission-critical systems with new functionality and updates. But great software developers are not an accident. They are not born that way. In order to become great, software developers must be trained.

Great software developers in an IT organization have important advantages. The first is that they are more productive. In fact, studies have shown that good developers are 10 – 15 times more productive than average developers in writing new code. In other words, good developers write code more quickly and efficiently. This is a dramatic difference in job functionality between average and great developers.

Great developers also write higher quality code that is easier to maintain. This higher quality of code makes a distinct difference in the long-term costs of maintaining a program. In fact, many IT managers do not realize that maintenance is actually the biggest cost of developing a new software program. At least 50% of the cost of development happens after the initial coding is completed. Messy, irregular code increases that expense by making it difficult for the succeeding developers to take over and make adjustments to the code, as business needs change. Good, clean code reduces that maintenance expense. And great software developers write great code.

To create great software developers, and reduce maintenance expenses, most managers rely on traditional means of training, such as e-learning and classroom training. These methods of training are important, but have some important flaws and disadvantages in their techniques. E-learning and classroom training do not bring the highest return for the dollar in terms of quality or efficiencies gained by the developer.

In recent years, the new model of coaching has become the most effective tool in creating great software developers. Coaching is a new training technique with an ancient twist. By incorporating the basic principles of learning at the side of a software expert, coaching overcomes many of the obstacles presented by traditional training. The quality of learning is higher, and the process is much more cost-effective.

The information included in this white paper will help you to understand how you can use coaching at your organization to create great developers. It discusses coaching techniques, as well as the advantages of using a coach versus a traditional training program. Finally, this document will compare the return on investment of both training and coaching.

## TRAINING SOFTWARE DEVELOPERS

Training is critical to every IT organization, because it creates great software developers. Every great developer has had training, somewhere along the way. He or she gets trained in language vocabulary, coding skills, architecture, and customer requirements analysis. All of this begins as a simple exchange of knowledge and skills, followed by repeated practice—and testing—of those knowledge and skills.

There are many advantages to regularly training your software developers. Highly trained developers write cleaner code, solve problems more quickly, and are more productive overall. Their programs require less maintenance later, which costs the company less money over time.

Moreover, regularly trained software developers report higher satisfaction with their work lives. They feel more challenged because they are learning new skills frequently. Well-trained developers are less likely to quit their jobs for this reason. This lower turnover is critical in today's highly mobile workplace.

There are many important reasons to invest in the training of your software development staff. The problem with this investment is that the traditional *methods* of training are expensive, inefficient, and ineffective.

## WHY TRADITIONAL TRAINING DOES NOT WORK

Two approaches have traditionally been used to teach skills and knowledge to their software developers. Most training happens through a classroom experience, as in traditional classroom training. Other training happens through an independent learning experience, which is today dominated by e-learning, such as CBT's, web tutorials, and interactive distance learning classes.

The biggest problem with both e-learning and classroom training is that they do not effectively apply to the real-world because they are lab-based. In other words, 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 developer's current (often behind-schedule) project, which he or she will return to face as soon as the learning experience is over. The class material is so out-of-context that it is difficult to effectively apply the lessons learned at the workplace.

Poor timing of the knowledge received also leads to tremendous inefficiencies. A developer may attend a class or tutorial before, during, or after a significant project milestone is completed. He or she may learn many valuable things during the class. However, if those lessons are not applied immediately to the project at hand, then they are usually forgotten. Six months to a year later, as much as 80% of the learning has evaporated because the knowledge was not practiced and applied immediately. This is an extremely inefficient way to learn.

The lack of personalized, immediate feedback in both e-learning and large classrooms also limits the opportunity for the developer to grow both personally and intellectually. Because there are so many participants competing for attention in the classroom, no one participant gets the feedback he or she needs to correct mistakes on the spot. With most e-learning, the feedback comes from a computer program, which means it is immediate—but not personal. (The computer cannot, for example, then answer a question about why the developer got a certain result.) The lack of immediate and personal feedback breeds sloppy coding habits which are then brought back to the workplace, leading to poor quality code and high maintenance.

Finally, no traditional training program can tailor the learning style to fit every single participant. Every learner comes to conclusions and learns the technology at a different pace and style, whether it's kinesthetic, visual, or audio. Most curricula is tailored to the lowest common denominator, which means that it is geared toward the slowest participants who learn in an audio format. Faster learners, and learners who prefer a visual or kinesthetic learning experience, lose out, leading to wasted dollars invested in training once again.

## COACHING: THE NEW MODEL

We all understand that training is necessary to create great programmers. However, the old models of training no longer work in our dynamic, fast-paced workplaces, where software systems are developed in a matter of days, instead of months and years. A new model of training, called “coaching,” addresses all of these challenges. Coaching is a more efficient, effective, and personalized way to create great programmers in your IT department.

Coaching is a new model of learning that utilizes direct personal interaction to facilitate learning, using a real-life, current software project. Similar to an expert blacksmith working with an apprentice to shoe a customer’s horse, a software expert works directly with a single developer or a team of developers to help them complete an immediate software project. They work side by side with the developer to write lines of codes, design architecture, and perform maintenance to the software system.

The goal of the coach is to deliver exactly the information and skills needed by the developer to complete their project—no more, and no less. The developers can be beginners, intermediates, or advanced. The developer shares the coding skills, architecture skills, design skills, and vocabulary needed to solve each problem as it arises, thereby helping the developer to complete their tasks and deliver a high-quality product on time.

When a developer tests a new skill to solve a problem, the coach gives him or her personal feedback—on the spot. This intimate learning environment allows developers to see their mistakes immediately, and prevents the bad habits that result in sloppy code. This level of personalized feedback is virtually impossible in a classroom of fifteen or twenty participants.

This personalized learning environment also gives the developer the chance to build their confidence. Confidence in problem-solving is the true hallmark of a great developer. One-on-one feedback from the coach ensures that the developer can take risks to solve problems in a “live” project that they never would have taken before. He or she gets the opportunity to fail, with direct supervision and support from the coach. Failing in a structured learning situation (that can easily be corrected by the coach, if necessary) builds confidence in the developer, while at the same time assuring that the final code in the project will be the highest quality possible. This kind of real-world, time-sensitive learning experience is virtually impossible to replicate in a classroom environment.

## THE MAXIMUM RETURN ON INVESTMENT

Coaching offers the best return on investment for training dollars spent on software developers. It is a well-known fact that IT budgets are under pressure. Software programs must be developed with fewer people, fewer dollars, and in fewer days. There is less money available for training or consulting. Let's look at an example of how coaching delivers higher quality training—for less money.

### CASE STUDY #1: CLASSROOM TRAINING

You decide to send 3 of your intermediate software developers to a four-day classroom training to learn some new skills. Your total costs might look like this:

4-day classroom training fees for 3 developers	$\$1600 \times 3$	=	\$4,800
Time off for 3 developers for 4 days @ hourly rate of \$50/hr, including benefits & taxes	$4 \text{ d} \times \$50/\text{h} \times 3$	=	\$4,800
Value lost from the 3 developers away from work on their current project, at 50% of hourly rate	$50\% \times \$50/\text{h} \times 3$	=	\$2,400
Cost of mistakes and errors of 3 developers attempting to apply knowledge after returning to work on a 2 month, full-time project	$320 \text{ h} \times \$50 \times 3$	=	\$48,000
<hr/>			
<b>Total cost of classroom training</b>			<b>\$60,000</b>

### CASE STUDY #2: COACHING ENGAGEMENT

You decide to hire a coach to work with your 3 developers instead. The coach is an expert in her field, and she charges \$125/hour. Your total costs might look like this:

Hiring 1 coach to work with all 3 developers for 6 weeks to complete a project on-time	$\$125/\text{h} \times 40 \text{ h} \times 4 \text{ w}$	=	\$20,000
<hr/>			
<b>Total cost of coaching engagement</b>			<b>\$20,000</b>

## **THE FINAL ANALYSIS: COACHING V. TRADITIONAL TRAINING**

In the final analysis, coaching costs less than traditional training, and delivers more value. In fact, a typical coaching engagement costs 30%-50% less than a traditional classroom learning experience. Why is this true?

Coaches cancel the cost of lost productivity caused by traditional classroom training. With a coach, the three developers do not need to leave their work site, like they do for a class. They can stay at their desks, and work directly with the coach. Deadlines are not sacrificed. This eliminates the cost of lost work time, and lost billable time as well.

More importantly, the expensive cost of experimentation and error from traditional training is erased. Developers do not learn sloppy habits because mistakes are corrected on the spot by the coach. The coach can even step in and write code in place of the developer if necessary. He or she is a functioning part of the team.

At the same time, coaching delivers the training in a more realistic, time-sensitive, and personal way. Coaching is overall more effective than traditional classroom training—at a lower cost.

## HOW DOES COACHING WORK?

Coaches come in all shapes and sizes. Some coaches work independently, as one-person contractors. Others work as part of a team at a software development firm. Most coaches specialize in teaching a specific language or skill set.

Coaches can work one-on-one with one particular software developer, or they can work with an entire team of developers. The coach works side-by-side with the software developers, at their office or cube location. They show up daily, and work the same shifts as the developers they are coaching. In fact, a coach often acts as a functioning member of the team, attending team meetings, and even completing task assignments if necessary. A coaching engagement can last for as little as one week, or as long as six months—depending on the skill level of the staff and the complexity of the current development project.

There are five basic steps to a successful coaching engagement:

1. **Analyze requirements.** A good coach will spend the time upfront to assess the skill level of your development staff. They will also take the time to learn the requirements and deadline of the current project, before making a commitment.
2. **Design a custom training plan.** The coach should next design a custom training plan, based on the results of the assessment. This plan will detail the approach and techniques that he or she will use to train the current deficits in skills of the staff, in order to meet the current project deliverables.
3. **Deliver coaching.** Coaches then work with the individual or the team of developers to help them meet their current project deliverables. They may use a mixture of training techniques to accomplish this goal: small-group training, mentoring, pair programming, or modeling.
4. **Track progress.** A good coach will not only train the individual, he or she will also measure the progress of that individual. Regular evaluations, feedback, and reporting to the individual and to the manager help everyone to chart the progress and value of the engagement.
5. **Improve processes.** The best coaches not only show up to train the developers, but also leave behind a good set of improved daily processes. These processes ensure that clean, high-quality code is developed long after the coach has completed the engagement.

## BEST-PRACTICE COACHING TECHNIQUES

The best coaches take a blended learning approach, using a wide variety of techniques to train the individual or team of developers. No matter what techniques they use, the goal is always the same: the staff should gain enough skills and vocabulary during the coaching engagement to complete the project at a top-quality level. Moreover, the staff will then be able to complete other projects at the same top-quality level after the coach leaves. In short, the goal is to create great programmers.

All good coaching engagements utilize the fundamental principles of action learning: the coach delivers only the skills and knowledge needed by the software developer at that exact moment. Unlike the typical classroom scenario, where the developer walks away with hundreds of topics he cannot immediately use—and quickly forgets—the coach provides the tools needed to solve the immediate problem at hand, a real-world development project.

A good coach is not unlike a master craftsman, or even a high school track coach. Imagine the blacksmith of 150 years ago, supervising an apprentice in a small shop. After teaching some general knowledge and skills, the master craftsman works side-by-side with the apprentice to make sure that the iron is wrought in precisely the correct way, while still delivering a high-quality horse shoe.

Like the master craftsman, the good coach uses a combination of the following best-practice techniques:

- **Fundamentals training**—beginner developers learn the fundamentals of a select few topics, such as programming language vocabulary
- **Modeling**—the coach demonstrates the correct code-writing procedure, while the developer “looks over their shoulder” to see how it should work
- **Pair programming**—the coach and the developer sit at one computer, and one person writes code, while the other watches; they switch off frequently, giving the developer the chance to see how an expert programmer works and then get feedback on his or her own coding skills and choices
- **Code review**—continuous, personalized feedback on the code written by the developer, giving him the chance to correct mistakes immediately; this technique solves the bad habits of sloppy code development

All of these best-practice techniques serve to accelerate the learning experience for the individual developer.

## CONCLUSION

Coaching has emerged as a far superior method for creating great software developers. Coaching costs 30%-50% less, and has a higher rate of effectiveness—leading to a greater return on investment. Coaching creates an action learning environment where developers can safely test their skills, while still completing their project on time. And coaching gives developers the feedback they need to write top-quality code, preventing costly errors later, during routine software maintenance. Dollar for dollar, coaching is a better way to create great developers than traditional classroom training or e-learning.

## ABOUT VISIONPACE

Visionpace, Inc. 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
- Access
- ASP Classic
- B2B and E-Commerce
- BizTalk
- Crystal Reports
- SharePoint
- SQL Server
- Visual Basic
- Visual FoxPro
- Web Design and Services
- XML

Visionpace has also developed an exclusive line of coaching services, called Perfect Coaching™. This service includes three comprehensive levels of coaching: Agile Development, Pair Development, and Co-Development. These coaching services are 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 pair programming, code review, modeling, and personalized feedback. They spend extra time learning the latest tools, techniques, and methodologies required to mentor beginner, intermediate, and advanced developers. Many of the 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.

## BIBLIOGRAPHY

- Bliss, Doug, and Russ Swall. "Software Development in Pace with Your Vision."  
Presentation. Visionpace, 2005.
- "Instructional Design: Blended Training." Entelechy. [www.unlockit.com](http://www.unlockit.com). April, 2005.
- Jantsch, John. "Perfect Coaching." Kansas City, Missouri, 2005.
- Johnson, Lauren Keller. "Real-Time Learning: How the Best Companies and Leaders Make  
It Happen." *Harvard Management Update*. January 2005.
- Mortimer, Mark. "Interactive and Experiential Learning to Increase Business Performance."  
*The HRM Guide*. UK Human Resources. April 18, 2005.
- Patton, Susannah. "Do It Yourself Development." *CIO Magazine*. January 15, 2005.
- Singh, Harvi, and Chris Reed. "Achieving Success with Blended Learning." *2001 ASTD  
State of the Industry Report*. American Society for Training & Development and Centra  
Software. March 2001.
- Ward, Susan. "Choosing a Business Training Solution." Excerpt from *Your Guide to Small  
Business*, published by About, Inc. 2005.
- West, William V. "Value-on-Investment and the Future of E-Learning in the Training  
Market." *Educational Technology*. September/October 2004.
- Wilson, Jack, and Brian Beatty. "Blended Learning: Combining Instructional Methods for  
Maximum Training Impact." Option Six, Inc. November 20, 2001.