Distributed Development: Insights, Challenges, and Solutions

Paul E. McMahon
PEM Systems

Today, many organizations are facing difficulties competing for new work due to a critical shortage of engineering skills. Employing the power of distributed development can increase an organization's opportunities to win new work by opening up a broader skill and product knowledge base, coupled with a deeper pool of personnel to potentially employ. By distributed development, we mean development efforts that span multiple organizations and/or multiple physical locations. This article provides an overview of key issues and challenges managers and project engineers are facing today on distributed development efforts. Insights into root causes of difficulties and recommended solutions are provided. Within this article the terms distributed and virtual are used synonymously. Information presented in this article has been excerpted and condensed with permission from the newly published Virtual Project Management: Software Solutions for Today and the Future [1].

Other authors using the terms *virtual collaboration*, *virtual development*, and *distributed development* have addressed the subject of this article. Regardless of the name used, this subject is about the issues involved when multiple organizations and/or multiple physical locations join forces on an advanced technology software-intensive effort.

It is important to note that I am not referring to traditional subcontract relationships. Rather, this article focuses on the use of *virtual teams* operating more as a single *integrated* team employing some degree (to be discussed) of common processes, support services, and technical strategies driven through a streamlined management chain.

While just a few short years ago such a project would have seemed inconceivable, modern technologies like e-mail, the World Wide Web, NetMeeting, teleconferencing and videoconferencing are today providing new possibilities for distributed teams to work in a more integrated and productive manner.

Why Care About Distributed Development?

To understand the critical importance of succeeding in a virtual collaborative environment, one only needs to examine the changes taking place inside today's workforce. According to a recent study conducted by the U.S. Bureau of Labor Statistics, approximately 25 percent of all workers age 16 or over have been with their current employer 12 months or less. This same study indicates that the average worker is now expected to change jobs every five years [2].

These statistics paint a picture of an increasingly mobile work force. The 20-year employee holding a wealth of corporate knowledge inside his head may well be a corporate asset of the past. At the same time, corporations are experiencing an increasing demand for software-intensive solutions produced from a combination of existing products and new developments. This, in turn, is driving a greater demand for personnel with increasingly specialized software skills; that is, skills geared toward specific software products. Compounding this demand for key people is the seemingly never-ending shortening of cycle times to enter and succeed in new markets.

> "Within this demanding environment, even the largest mega-corporations are finding they can no longer maintain inside their own corporate walls all the critical skills necessary to compete in many new markets."

Within this demanding environment, even the largest mega-corporations are finding they can no longer maintain inside their own corporate walls all the critical skills necessary to compete in many new markets. As a result, more and more companies today are reaching out in a cooperating manner to organizations previously viewed only as competitors. While the rationale for embracing virtual operations is evident, many corporations today are struggling with implementation-related challenges.

Distributed Development Challenge

Research conducted by Booz-Allen [3] has identified four characteristics common to many of today's collaborative failures:

- Cultural incompatibility.
- Leadership struggles.
- Lack of trust.
- Inbred notions of competition.

Recently, a three-year collaborative development study was documented in *Virtual Project Management: Software Solutions for Today and the Future* [1]. The results indicate that beneath these symptoms lie a number of key relationships that include both technical and non-technical factors. On the positive side, this study also indicates that, once these key relationships are understood, practical and affordable actions can be taken to aid organizations in achieving successful virtual operations. The referenced study is based on experiences derived from real distributed projects that occurred between 1994 and 2000.

The Eight-Step Plan

In this article we employ an eight-step plan (see Figure 1) as a framework to assist our investigation of distributed project challenges. This framework can be used as an aid in setting up a new distributed project, or in instituting improvements to an ongoing one.

While the eight steps identified may appear traditional and relevant to any project, our focus in this article is on specific issues related to distributed operations. It should be noted that the use of the term *steps* is not intended to imply that virtual project success can be achieved through a *cookbook* approach. Nor do we mean to imply that these steps are easily achieved. Many of the challenges faced on

virtual projects are closely coupled to communication. Evidence of this fact can be seen throughout the eight steps starting with Step 1.

Step 1: Selecting Team Leaders

It should not be a surprise to anyone who has worked on a collaborative endeavor that one of the most important decisions to project success is the choice of team leaders. While strong conflict management skills, and a willingness to consider alternative approaches are desirable traits for leaders on any project, these leadership characteristics are particularly critical to distributed project success.

Unlike most traditional collocated projects, virtual projects face the added challenge of teammates with differing backgrounds, experiences, and technical expectations. This situation can give rise to frequent and often intense conflict. While most project conflicts faced are not insolvable, all too often timely and effective resolution falls short due to a breakdown in communication.

Conflict and Communication

Conflict is not unique to distributed projects, but it is not uncommon for traditional, collocated approaches to conflict resolution to fail in a distributed environment. To understand why, we must look closer at communication in the organization.

In the early 1980's, Alan Cox conducted a survey in which he found that more than 66 percent of middle managers believed that more than half of the communication in their organizations occurred informally [4]. Experience on distributed projects indicates this is not only true, but some of the most critical communication with respect to conflict resolution occurs in this manner. Unfortunately distance, differing experiences, and internal team competition often hinder informal communication on distributed efforts.

It should be noted that the term *informal* in this article means unplanned and undocumented.

A Partial Solution to Virtual Project Conflict

While distributed development provides the potential power of rapidly accessible personnel with key skills and key product knowledge, it often does so at a cost of interpersonal team bonds built over time through shared experiences. Although there does not exist a simple cure-all for the loss of shared experiences and traditional informal communication on distributed projects, a number of partial solutions do exist.

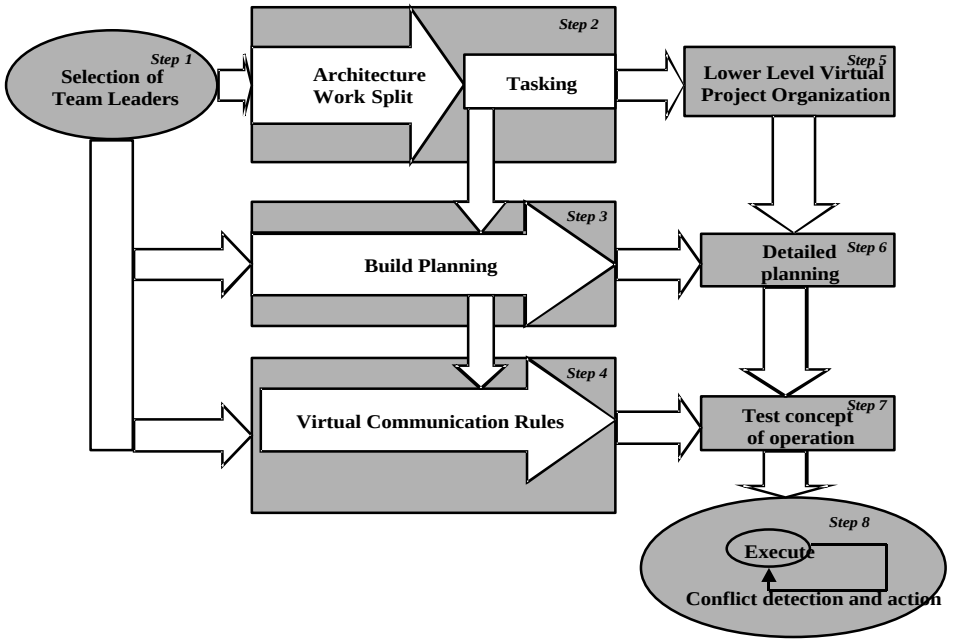

Figure 1: *Eight-Step Plan*

For example, experience indicates that collocating a small team of senior systems designers during the critical creative design stage of a distributed project can be effective. Studies have shown that when team members must interact frequently and for short durations, collocation offers the best opportunity for success [5]. This does not mean that full-time collocation is required for the life of the program. Cases where repeating cycles of intense collocated work followed by periods apart have worked well, especially during the early critical creative design stage.

It is also important to note that early collocation is multi-purpose. This activity supports the shared development of a project's common technical vision, but equally important it also starts the process of building interpersonal bonds among key teammates across project organizations/sites. Other recommendations in support of a shared leadership vision are discussed in Step 7.

Step 2: Architecture, Work Split, and Tasking

While collocation of key personnel early can go a long way to getting a distributed effort off on the right foot, a significant number of managers remain skeptical about the viability of distributed development. When surveyed on this subject, many managers have expressed fear in not knowing if a remote team member is "doing the right thing [6]."

When one considers how much a new engineer traditionally has learned about task expectations through informal means, this fear is understandable. Managers who have known only collocated operations often take informal communication for granted, but they intuitively know how much they rely upon it every day.

Even in organizations where task assignments are formally written, there is usually a significant reliance on informal communication to clarify and guide the new engineer. With respect to designing or coding a solution, this informal guidance often takes the form of an experienced engineer relaying examples of solutions patterns that he/she knows will fit that particular organization's accepted technical architecture. Although the process described is commonplace, the relationship being described among architecture, work split, and tasking has not always been well understood.

Definitions

When the term *architecture* is used, it means the "components" of a system and the rules defining how the components are connected, along with any constraints. When used, the term work split means the allocation of responsibilities across physically separated sites or organizations. Work split can also be thought of as *site-level tasking*.

Architecture as a Management Tool

Traditionally, many think of architecture as a technical issue, and work split as a separate and distinct management issue. But in practice, work split decisions can fracture a sound architecture. Furthermore, a

sound technical architecture can actually provide one of the best task communication and coordination techniques.

For example, think about how a senior technical mentor guides a new engineer. The most effective mentors guide by listening first and then providing feedback that ensures the approaches chosen fit within the range of an organization's acceptable solutions. This is another way of saying that effective mentors guide through the vision of a technical architecture.

When used appropriately, a sound technical architecture can go a long way to addressing a manager's concerns about whether a remote team member is in fact "doing the right thing." Often it is through informal architecture-centric discussions that teammates gain the real insight needed to accurately meet task expectations within a specific organization. But for architecture to be effective as a task communication aid, the work split definition across physically remote sites must follow the architecture definition, not the reverse.

Architecture First

Too often we see work split decisions made without due consideration to the technical architecture. When work split decisions are forced prior to architecture definition, we often find distributed projects suffering from fuzzy task responsibilities and technical leadership struggles. Without a well defined architecture, remote groups often find themselves heading down inconsistent paths leading to project conflict and control struggles.

For example, the choice of computer hardware platform has been a topic of intense inter-site battles on many distributed efforts. By documenting agreed to platform constraints early, unnecessary project conflict during a critical project stage can often be avoided.

There are many sources available that can provide more in depth information on the key role of architecture to project success [1, 7, 8, and 9].

Do Not Delay the Work Split

There is another side to this coin that must also be adequately considered. Delaying the definition of work split too long can have equally devastating effects on a distributed project. The right answer to the problem of fuzzy task responsibilities is not always simply delaying the work split definition until the architecture is defined. When work split decisions are delayed too long, internal teams' mistrust quickly builds. Be aware that the consequences of defining a work split that leaves tasking grey in certain areas has proven to be a poor solution to this challenging area. While your architecture needs to be in place when you define your work split, you should also know that architecture is an evolutionary product. Never wait for the architecture to be 100 percent complete, or you'll never get your work split defined.

> "Studies have shown that when team members must interact frequently and for short durations that collocation offers the best opportunity for success."

Step 3: Planning the Builds and Site-Specific Infrastructure

While a sound architecture can aid work split definition and task management, it does not convey when product functionality is available. This is the purpose of a build plan.

Today, many companies are moving toward incremental build approaches especially on distributed projects because a build approach reduces integration risk. You can think of a build as a set of hardware and software that meets a subset of the functionality of the final deliverable product. *Planning and coordinating the builds across distributed sites may be the single greatest challenge faced on virtual projects.*

One of the keys to effective build coordination on distributed projects is found in the technical infrastructure (hardware computer platforms and software tools). The criticality of the technical infrastructure to effective build planning can best be conveyed through two competing technical infrastructure visions often found on distributed efforts. These two visions are referred to as the *maximize use of company-owned assets* vision and the *seamless* vision.

Maximize Use of Company-Owned Assets

Maximized use of company-owned assets means the use of existing company-owned organizational assets (computer hardware and software tools) to the maximum extent possible to meet project needs. Those who demand that the project's direct infrastructure costs be kept to a minimum drive this vision. While driving hard toward this vision does reduce the up-front project expenditures, it can also increase the project's integration risk and the overall project cost since not all existing hardware and software will be the same.

Seamless Vision

On the other hand, those driving the seamless vision believe that an engineer should be able to log in and do 100 percent of his engineering work using identical tools and processes from any workstation at any remote project location. While the advantages of the seamless vision are evident, the cost of common hardware, common software tools, and software licenses can quickly become prohibitive for a single project. It is also important to keep in mind that the choice you make with respect to infrastructure (hardware, software tools) cannot be made independent from your project's process decision unless you keep your process definition high. But keep in mind that if the process definition is too high, it is more likely to lead to miscommunication. In the following section we discuss a common distributed project process pitfall.

Do Not Drive Process Commonality Too Deep

One of the most common pitfalls witnessed on distributed projects is referred to as the "let's use the most mature process available" pitfall. This pitfall usually starts with the project leadership's decision to mandate that all project sites use a common process. While at the appropriate level a common process makes sense, the pitfall is tied to what often happens next. Rather than define the common process at the appropriate level and allow individual sites the appropriate freedom to leverage site-specific procedures, oftentimes a mandate is sent across the distributed sites to drive procedure commonality (different from process commonality) as well. The common set of procedures chosen is usually supplied by the highest software maturity-rated organization on the project.

It is natural to look to your teammate with the highest process maturity for software guidance. However, procedures represent only a small part of the complete process maturity picture. They are often too site-specific to make sense for application across multiple organizations (each with their own culture and history) in conducting development activities.

When attempts are made to drive commonality too deep into a virtual

organization the lack of an enabling organization, supporting infrastructure, and supporting culture at each of the remote sites is almost certain to lead this initiative to failure.

Solution to the Common Process Initiative

A process freedom line is defined to be the point in the process where a site/organization is free to make process-related decisions. For example, a project level procedure may call for a design document to be produced with specific design artifacts, but may not require a specific document format. I recommend that virtual projects define process freedom lines at the point where products and people must come together across divergent sites/organizations. It is not recommended that the project attempt to dictate how specific sites/organizations accomplish their tasks internally.

The freedom line definition essentially tells each site where they are free to leverage site-specific procedures (that can include site-specific support organizations and company-owned assets) in implementing solutions. This strategy has proven effective at balancing the management of the project's integration risk, while at the same time leveraging the strengths of individual sites/organizations.

Step 4: Virtual Communication Rules

Architecture definition and planning are critical on all projects, collocated or virtual. Virtual communication, on the other hand, presents distributed projects with new challenges not previously faced in traditional collocated environments.

Today, virtual communication is in its infancy. We are just now starting to comprehend the implications of first generation lessons on using the World Wide Web, teleconferencing and videoconferencing, NetMeeting, and e-mail. We recommend that virtual project communication lessons drive written virtual project rules (guidelines) to aid engineers in the effective application of new tools.

For example, in the early stages of a large virtual project it is not uncommon for engineering personnel to receive 50-60 e-mails per day! Think about it. If you take just four minutes to process each e-mail, at this rate you could spend half of your day just handling e-mail correspondence. E-mail flooding is the result of personnel being given a new tool and insufficient training in its use. E-mail, voice mail, teleconferencing, videoconferencing and NetMeeting all require training in more than just the mechanics of their use.

I recommend that each project create its own guidelines as it moves forward. And don't ignore rules and lessons that may seem obvious. I challenge those who have been involved in distributed efforts with the following questions:

- Does your distributed project have rules for the use of e-mail and teleconferencing?
- Have your people been trained and are they following the guidance provided?

> "On virtual projects the building of trust requires a more proactive management stance."

It has been our experience that while few disagree with the concept of guidelines, most distributed projects in operation today are not doing the best job of deploying effectively virtual communication technologies, and the unfortunate part is that it could be costing you plenty in human resources. Furthermore, this recommendation is simply not that expensive to deploy!

For more examples of first-generation virtual communication lessons, see [1].

Step 5: Lower Level Virtual Project Organization

The recommendation for the lower level structure of a distributed project organization may not be a popular one. At the top end of the organization where a breath of issues must be addressed, the Integrated Product Team (IPT) structure tends to function well, and I recommend it. This is the level where *heads-up* activities exist. By heads-up activities I mean work that must look across the multiple sites and organizations of the project. But I have also observed – and many clients concur – that a strict IPT structure is weak when it comes to producing products that include detailed design, code, and test cases.

Where the *real* engineering, or what we refer to as heads-down work, occurs, I have found that on distributed efforts a *hybrid* of IPTs and functional groups is often more effective. When I use the term heads-down work I mean the engineering work associated with building and testing actual executable code.

An example of why we recommend this structure can be seen in the need for an infrastructure implementation group that provides common services that multiple product development teams may need across remote sites. Too often, when virtual projects try to drive a strict IPT structure deep into the organization, responsibility for critical common services is lost. This is because when a strict IPT structure is employed at lower levels of the organization, you often find that each of those lower level IPTs focus almost exclusively on their own specific product. As a result, each solves their own specific problem in their own specific way.

On the other hand, when organizations recognize the need for an infrastructure implementation group that is not focused on any single specific product, commonality across multiple sub-products can be more effectively achieved.

Step 6: Detailed Planning

Detailed planning is important on any project, but on a virtual project its critical relationship to work split is often misunderstood.

Often on virtual projects work split decisions get delayed. This can occur for a multitude of reasons – most are not technical. But, all too often, great pressure continues to be brought on the engineering team to complete the detailed project plan despite the uncertainties of where work will actually get done. What is too often misunderstood in these situations is the extent to which detailed planning directly depends on work location.

While some project planning can be done independent of location, think about the real issues an engineering manager faces when it comes to developing a really detailed plan that is actually executable. Here are just a few of the critical questions to be asked:

- Is the development hardware available?
- Have the software tools and licenses been procured and installed?
- Have we identified the engineering personnel that will do this job?
- Have the identified personnel been trained on the chosen platform, language, tools?

To develop a detailed plan that is executable, managers must make assumptions with regard to each of these issues. These are the real issues that truly impact project performance.

Now think about how the answers to these questions are affected when work is moved to a different location. Based on my experience, if you are doing detailed

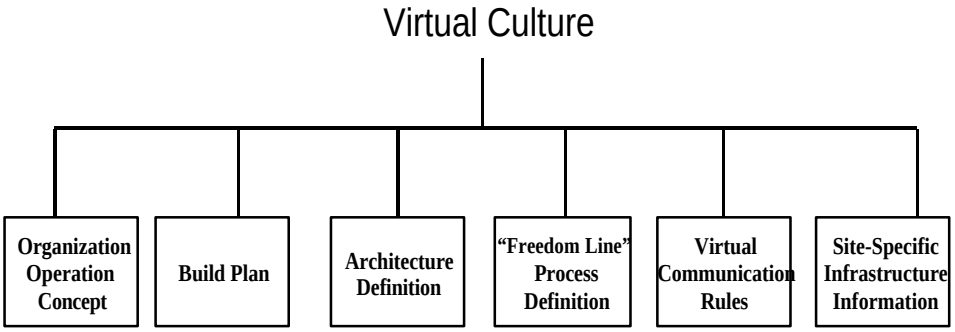

Figure 2: *Sample Virtual Culture*

planning, and the work location is still fuzzy, you can start planning right now on doing your detailed plan all over again!

Step 7: Test the Operation Concept of the Virtual Organization

It is unreasonable to expect new virtual project organizations to instantly operate as effectively as strongly cultured time-tested collocated operations. Effective organizations – collocated or virtual – do not just happen.

We recommend that newly established virtual project organizations set aside the time for project leaders to walk through key organizational scenarios that are most likely to cause leadership friction. When leaders take the time to discuss openly their visions of the virtual organization, potential problem areas can often be uncovered and resolved quickly.

Often, at the start of a new project, leaders are uncertain where the most likely trouble spots might be in the operation of the distributed organization. For example, experience has shown that when multiple organizations are involved, task management of remote personnel is a critical area. It is recommended that you walk through your task management model and your risk management model, and be sure to do it in a face-to-face setting with all your project leaders present. For more information on recommended organizational scenarios, see the referenced book [1].

Step 8: Execute

As discussed in Step 1, increased conflict is to be expected in a distributed development environment due to the lack of shared experiences and interpersonal bonds. Recognizing this fact, a key to effective process deployment on virtual projects is ensuring that leaders are aware of the warning signs of unhealthy team conflict. An example of an unhealthy conflict warning sign is what we refer to as the repeating issue warning sign. This is the case where:

- A valid issue is raised by a virtual team member.
- The issue is worked through by the team.
- A consensus is reached and the issue is put to bed.
- One week, or one month later, the same issue returns.

Does this sound familiar? Have you ever sat in a meeting and thought you were sitting through a rerun of an old movie? If you detect the repeating issue warning sign on your distributed project be sure to deal with it rapidly before it does permanent harm to your team. See reference for more examples of unhealthy team conflict warning signs, and recommended actions [1].

Deploy A Virtual Culture

In this article I have emphasized challenges being faced today on many distributed projects. I have stressed the impact of the loss of traditional informal communication in distributed environments, and have provided related recommendations. I also recommend the deployment of what is referred to as a virtual culture.

A virtual culture [1] is a simple, yet powerful concept that brings an information-age perspective to the notion of culture. Think of a virtual culture as a physical framework that supports effective communication across distributed project sites.

The virtual culture – unlike traditional collocated engineering cultures – is product oriented. It is not intended to replace past traditional collocated cultures. In fact, I don't think you should try to replace strong local cultures. Rather, my recommendations are based on leveraging the strengths of your teammates within their proven environments.

The virtual culture complements the existing site-specific cultures providing the critical information needed to coordinate and communicate key tasking information across distributed sites. This approach reduces the risk of rework when remotely developed products are integrated together. A sample structure of a virtual culture is provided in Figure 2. Virtual cultures can be implemented through a Web site or through a shared directory system.

It is worth noting that a key difference between a virtual culture and a traditional culture is found in its formality. Experience indicates that an effective virtual culture cannot be informal. In other words, it must be written down. We recognize that in today's world this emphasis on the written word may not be popular.

However, recommendations with respect to a more formal virtual culture should not be interpreted as a step backwards to the days of voluminous documentation. The virtual culture is not intended to include historical milestone-type documentation, but rather it focuses on those critical pieces of information that must be coordinated and communicated across distributed sites. Experience indicates that when you go virtual and utilize remote operations that more things do need to be written down in support of effective remote communication.

Conclusion

The potential gains of virtual operations are great. Nevertheless, implementation issues cut deep inside present-day engineering organizations. Inside traditional engineering environments, common cultures, common site infrastructures, and common experiences provide key ingredients supporting team trust.

In collocated environments, trust appears to just happen through little more than the passage of time. In reality, in these traditional environments there have always been numerous informal factors hard at work building trust on a daily basis. In the past these informal activities may not have received the attention they deserve. This is because in strongly cultured collocated environments the benefits of informality came to us essentially for free. In the virtual world this is no longer the case.

On virtual projects, the building of trust requires a more proactive management stance. It requires leaders who are willing to listen to alternative approaches put forth by team members who may

have very different backgrounds and experiences from their own. But listening is only the first step.

When alternative ideas are accepted they must also be effectively communicated to the full team. And on virtual projects, we now know we cannot rely on traditional collocated informal mechanisms to achieve this communication. Therefore, in the virtual world, the written word takes on new and increased importance.

Think about the information that is today conveyed through unplanned meetings in hallways, at lunch, casually in cubicles and over the tops of cubicles. While experience has shown that e-mail, teleconferencing, videoconferencing and NetMeeting are all powerful distributed development communication tools, they cannot replace what collocated organizations have taken years to mature. Consider deploying the virtual culture concept on your distributed project to aid communication to all your team members. It is not that expensive to implement, but the potential cost of not implementing one is.◆

References

1. McMahon, Paul E. Virtual Project Management: Software Solutions for Today and the Future. Boca Raton: St. Lucie Press, An Imprint of CRC Press LLC, 2001.
2. Gannett News Service, "Job Hopping by Young Workers Increasingly Common," 29 Aug. 1999.
3. Norton, Bob, and Cathy Smith, eds. Understanding the Virtual Organization. Hauppauge, NY: Barron's Educational Series, 1997. 68.
4. Cox, Alan. The Cox Report on the American Corporation. New York: Delacorte Press, 1982. 112-114.
5. Gindele, Mark E., and Richard Rumpf, eds. "Effects of Collocating Integrated Product Teams," Program Manager, July-Aug. 1998: 38.
6. Haywood, Martha. Managing Virtual Teams. Boston: Artech House, 1998.
7. Software Engineering Institute, <www.sei.cmu.edu/architecture/definitions.html>.
8. Shaw, Mary, and David Garlan, eds. Software Architecture: Perspectives on an Emerging Discipline. Englewood Cliffs, NJ: Prentice Hall, 1996.
9. Bass, Len, and Paul Clements, eds. Software Architecture in Practice. Reading, MA: Addison-Wesley, 1998.

Additional Reading

1. Karolak, Dale Walter. Global Software Development. Los Alamitos, CA: IEEE Computer Society, 1998: 35-46.
2. Mayer, Margery. The Virtual Edge. Newtown Square, PA: Project Management Institute Headquarters, 1998.
3. Lipnack, Jessica, and Jeffrey Stamps, eds. Virtual Teams: Reaching Across Space, Time and Organizations with Technology. New York: John Wiley & Sons, Inc., 1997.
4. Reifer, Don. Practical Software Reuse. New York: John Wiley & Sons, 1997.
5. Royce, Walker. Software Project Management. Reading, MA: Addison-Wesley, 1998.
6. Highsmith, James A. Adaptive Software Development: A Collaborative Approach To Managing Complex Systems, New York: Dorsett House Publishing, 1999.
7. Deeprose, Donna. The Team Coach. New York: American Management Assoc., 1995.

About the Author

Paul E. McMahon is an independent contractor providing technical and management leadership services to large and small engineering organizations. McMahon began his career in the early 1970's as a flight simulation programmer. Before initiating independent work at PEM Systems in 1997, he held senior technical and management positions at Hughes and Lockheed Martin. Today McMahon employs his 27 years of experience in helping organizations deploy high quality software processes integrated with systems engineering and project management. He has taught software engineering as an adjunct at Binghamton University, N.Y., and published more than a dozen articles and a book on virtual project management.

118 Matthews Street
Binghamton, NY 13905
Phone: (607) 798-7740
E-mail: pemcmahon@acm.org