Distributed Software Project Development

Anup Hande*, Sunita Suralkar, P.M.Chawan*****
Department of Computer Technology, Veermata Jijabai Technological Institute

Abstract

Now a days in Software development life cycle it is not possible that all the team members are present at the same location. This lead to the development of new concept of Distributed Software Project Development. Distributed Software Development (DSD) allows team members to be located in various remote sites during the software lifecycle. Managing these network of distant sub-teams is an issue which needs to be handled by organizations. In this paper we have discussed about the drivers, challenges and tools for distributed project development. This also gives the quick overview of Distributed Project Management.

***Keywords*— Distributed Projects, Distributed Software Project Management.**

I. INTRODUCTION

A distributed development project is a research & development project that is done across many business worksites or locations. It is a form of R&D where the project members may not see each other face to face, but they are all working collaboratively toward the outcome of the project. Often this is done through email, the Internet and other forms of quick long-distance communication. It is different from outsourcing because all of the organizations are working together on an equal level, instead of one organization subcontracting the work to another. It also is similar to, but different from, a virtual team because there is a research element.

The trend to distributed development teams is fuelled by economics, desire for competitive advantage and evolving business models. Companies seeking to reduce costs and tap into global talent pools are outsourcing business processes or increasing their spend offshore. Companies looking for supply chain efficiency are integrating their business processes with those of their customers, partners, and suppliers. Manufacturers of consumer electronics, appliances and other products are building partner ecosystems to differentiate their products through software functionality.

These trends place new demands on software development teams and their application life-cycle management tools. Teams today must collaborate effectively across different time zones, countries and company affiliations, and cope with differences in language, culture and process. Improving the ability of these geographically distributed teams to create high-quality software — and accelerate time to market — requires new capabilities in life-cycle tooling.

Geographically distributed teams must rely on the support of automated life-cycle solutions to a far greater extent than collocated teams to coordinate their work, share information and manage their projects effectively.

II. DRIVERS FOR DISTRIBUTED DEVELOPMENT

Just a few years ago, the typical development team could meet over sandwiches in the conference room to discuss requirements, look at code design and review project status. Today, it’s the rare team that can come together in one physical location face-to-face. Though development teams have included remote team members for some time due to the rise of telecommuting, today’s teams routinely include team members who live and work in different time zones, different geographies, even different companies.

The major drivers behind this long-term trend to geographically distributed development are:

Optimizing IT (and Other) Resources:

The need to manage costs and the need to leverage resources "in place," "as needed," "where needed" — without moving resources around are important drivers behind the trend to distributed development teams. Given flat, or only slightly increased IT budgets, IT organizations must maximize the resources they already have in place and better leverage the expertise that exists throughout the company. Because inaccurate or inadequate requirements and specifications are still a leading cause of failure for new applications, teams today involve more direct participation from domain experts and other business-side stakeholders, as well as project and program management, to ensure that projects stay on scope, on time and on budget.

The need to improve participation by non-developer team members results in a more diverse and distributed team. They need the support of strong life-cycle tools to share information and project status, and to foster their sense of community and cohesiveness.

Ecosystems Collaborating with customers, Partners and Suppliers:

Software is becoming increasingly pervasive in today’s consumer electronics and appliances. From cell phones, to automobiles, to home entertainment systems and even refrigerators, as hardware becomes commoditized, software-enabled features and functions are what differentiate products and create competitive advantage. This is a key driver behind the rise of geographically distributed development teams. As consumer products emerge as platforms for new software applications, large manufacturers must compete to foster effective partner ecosystems and leverage third-party software and ISVs to build out solutions on their platforms, shorten time-to market for new product generations, and grab market share. Ecosystem "owners" must be able to securely share source code, APIs, design documents and specifications, and other proprietary information with partners in order to accelerate the pace of innovation. Application life-cycle management solutions for these ecosystems must provide secure, Web-based access to critical information and ensure effective coordination and communication among multiple, autonomous (and potentially competitive) partners.

Global Enterprise Application Development:

Another, related driver for distributed development is the need for companies doing business around the world to rationalize and standardize their enterprise applications for a more global view of their business. These projects typically require IT organizations in multiple countries to collaborate. IDC research shows that companies are making significant investments in large projects for enterprise application consolidation, legacy application replacement, and application modernization.

Global development projects require robust, integrated life-cycle tools to help teams coordinate their work and collaborate effectively across time, geographical barriers, and changes in the team to ensure the quality, performance, and scalability of their applications. Also, because IT organizations undertaking large projects often need to incorporate outsourcers and consultants to augment their existing resources, or to bring in specific domain expertise or best practices, life-cycle tools must flexibly support external users while ensuring security and control over intellectual property.

Managing Costs and Securing Talent Rise of Offshoring:

Companies are leveraging a variety of sourcing strategies to optimize their IT spend and provide greater value to the business given IT budget constraints. While in the past, companies relied primarily on local services providers to meet their variable staffing needs, today they are increasingly leveraging offshore resources to manage costs and access broader talent pools. This is a highly dynamic trend: companies will continue to refine their sourcing strategies as business needs dictate. For example, while many companies are turning to large offshore outsourcers, others have established "captive offshore" operations in India, China or Eastern Europe to leverage skilled labor in these geographies, sometimes as part of their entry strategy into these markets. Whether outsourced or captive, offshoring application development creates significant new risks for IT organizations around project visibility and status, control of intellectual property and security, and team communication and coordination.

Real-Time, Connected Enterprise Follow - the Sun, 24 x 7 Support:

Another important driver behind today’s increasingly distributed development teams is the need to support 24 x 7 high-availability mission-critical applications. The Internet has given rise to explosive growth in "edge of the enterprise applications" —transactional systems such as online banking, online brokerage, retail ecommerce and extended supply chains that connect customers, partners and suppliers directly to back-office systems through Web interfaces and (increasingly) Web services APIs. Failures in these customer- and partner-facing transaction systems — whether due to bugs, performance problems or outages — translate directly into lost revenue, loss of market share and lost profits. The need to provide 24 x 7 support for these business critical systems helps drive the trend to geographically distributed development as companies employ "follow-the-sun" strategies for round-the-clock maintenance and support to accelerate cycle time and shorten the time to problem resolution.

Mergers and Acquisitions:

Another driver for distributed development teams is merger and acquisition (M&A) activity. Mergers typically result in multiple IT organizations that must collaborate over time and distance to integrate existing systems and consolidate operations. M&A activity is ticking up past a trillion dollars per year, and economists expect M&A activity to continue to strengthen as the economy recovers further.

III. CHALLENGES

Communication and collaboration. Distance and time zone differences make direct human interaction problematic: distributed teams may have very little "shift overlap" when communication can take place in "real time." Cultural and language differences compound the problem. Team members must communicate electronically to share knowledge, brainstorm and make decisions and to create the sense of community and visibility within the team that is so important to morale.

Management of change throughout the life cycle. Distributed teams face significantly greater challenges coordinating their work — from managing tasks and communicating status to locating the right information and people. Distributed teams need to rely more on check-in/check-out facilities to help them coordinate changes to software and on automated workflows to help them manage tasks. Companies must also be able to capture and manage important contextual information such as emails, enhancement requests, supporting documentation, etc., so that key information is not lost as team members change over time.

Project management and visibility. Without the opportunity for in-person status meetings, projects can easily fall behind schedule. Distributed teams need the support of automated systems to help them monitor progress and identify risks so they can keep their projects on track.

Accommodating external *team members*. Distributed teams are often more dynamic and diverse than collocated teams, and frequently include consultants, outsourcers, customers, suppliers, or partners. Enabling these external team members to participate effectively in the software development process requires automated systems that facilitate access while securing sensitive information.

IV. SUCCESS CRITERA

There are three main success factors for a distributed development project:

1. Select and/or recruit good, strong, highly skilled people.
2. Spend some money for face-to-face meetings, especially at the beginning of each major project.
3. Build an organizational design that supports working in a distributed development, including the right incentive systems.

By doing these three actions, one may obtain advantages beyond pure outsourcing or offshoring, namely much higher motivated employees in all parts of the distributed network, higher retention and certainly one gains from the diversity of the network.

To be successful in a global market, a company should manage the risks of global software development, but can use the positive aspects as input to shape the development process in detail and the culture in general.

V. TOOLS

The rise of the Internet as a ubiquitous connection between distributed locations and the quickly maturing market place of collaborative tools are essential ingredients for complex project success. Software vendors already have released Web-enabled versions of many familiar project management tools, enabling specialized tasks such as tracking requirements, schedules, and budgets to be distributed to multiple sites and scaled for multiple users.

Software engineering tool suites are beginning to follow suit. Similarly, more organizations are employing Web-based repositories, such as project Web sites, portals, and workspaces, for intelligently sharing and storing files both

within and across corporate firewalls. Those structured collaboration tools, often enhanced with workflow functionality, are instrumental for enabling the project management hard skills previously mentioned—such as budgeting, scheduling, and tracking requirements—on complex projects. However, they fall short for enabling the increasingly critical soft skills such as defining the business value, clarifying the vision, determining requirements, providing direction, building teams, resolving issues, and mitigating risk. Research on virtual teams treats this lack of support as a pre-existing constraint, recommending face-to face meetings as often as possible and at critical points in the project to augment email and telephone communications. Organizations are now beginning to leverage real-time collaboration tools to bridge the soft skills gap for distributed teams. Tools such as instant messaging, Web conferencing, whiteboards, and desktop videoconferencing provide substantially different communication possibilities than the familiar telephone, email, and face-to-face options. Tools for unstructured collaboration can enhance communication by enabling more frequent collaboration between distant coworkers. In contrast to early incarnations of unstructured tools (for example, expensive, room-based videoconferencing systems), these inexpensive desktop tools are designed for frequent, ad hoc use. Telework Consortium pilots indicate that these characteristics can lead to increased communications and trust, thereby facilitating quick decisions and enhanced team cohesiveness.

Looking across the IT industry, organizations with multiple locations and trading partners are rolling out integrated digital environments for secure sharing of files and databases, leveraging technologies from Electronic Data Interchange to Web Services.

Workflow functionality is enabling great leaps forward in productivity by minimizing lag time between tasks. Communication and collaboration tools are maturing rapidly, and despite the lack of interoperability, single-vendor applications are functional and stable enough to support distributed work in standardized environments. The emerging move toward contextual collaboration promises to bring these communication tools to our fingertips by linking them within the familiar applications we “live in.” Security concerns and limited network capacity still limit the use of advanced tools such as desktop videoconferencing in some work environments, but progress is accelerating on these fronts as well.

The Application life-cycle management solution should provide the following capabilities to be effective:

Full Life-Cycle Support:

Distributed teams require scalable, reliable software configuration management tools to ensure that all software development artifacts — including source code, requirements and specifications, design documents, screen shots, architectural diagrams, test plans, etc. — are reliably captured, versioned and shared. Distributed teams also need to rely on robust check-in/check-out facilities to help them coordinate their changes to software assets. Finally, distributed teams need integrated task management to help orchestrate their work and keep them on track.

Built-in Collaborative Tools:

Research shows that most teams rely today on ad hoc meetings, email and, to some extent, groupware to collaborate — that is, team communication today takes place, for the most part, outside of the SCM environment. With information dispersed across two or more separate systems, information is difficult to locate. It may be impossible to reconstruct the history around a particular design decision. There is no easy way to search for information. Worse, because it is scattered across multiple systems, information can easily be lost.

This is a major problem for geographically distributed teams, because so little of the context around the source code is captured in a usable form. Distributed teams need integrated collaborative tools so that the human interactions and knowledge exchange that inform the development process and shape the final product are captured automatically in the project repository in a structured way.

Collaborative development environments that incorporate threaded discussions, announcements, alerts and messages in the context of the software development process are the solution for distributed teams. Collaborative development environments automate the dissemination of information, foster a sense of community, and create a resource-rich virtual space in which team members can interact and share ideas.

Project Management Facilities:

Automated life-cycle solutions for distributed teams must provide strong project management capabilities to help them keep their projects on track. Project dashboards that provide visibility into project and task status, identify risks and reveal trends help ensure that problems are addressed before they can cause project delays. Solutions should support the team’s choice of software development methodology — whether waterfall, RUP, Agile or other method — according to the project’s unique requirements.

Security:

Because distributed teams often include external team members, security and the protection of intellectual property is a particularly important concern. Geographically distributed teams need to be able to limit access to project information on a "need to know" basis, by role — to particular projects or to portions of a project — without inhibiting collaboration with external team members.

Wide Area Network Support:

In order to facilitate teamwork across geographic and corporate boundaries, tools for distributed teams must be designed for operation over a wide area network typically the Internet. LAN-based products incur substantial performance penalties when they are run over a wide area network, due to the much greater network latency of the WAN. Code check-outs that take a few minutes over a LAN can take hours over a WAN, rendering response times unacceptable and adversely impacting development team productivity.

Replication may be used with LAN-based products to create local copies of the repository for each of the remote sites; the local copies are accessed on the remote LAN using existing "thick clients." In this scenario, WAN traffic is limited to the replication process itself, and LAN access to the local copy is very fast.

There are several drawbacks, however, to the replication approach. First, it creates administrative overhead and expense. A local copy of the SCM repository must be created for each geographical location, and care must be taken to ensure that the replication processes are set up correctly and performed reliably. Additional investments in hardware, software, and network infrastructure are required to support the local copies and replication processes. Redundant administrative resources may also be required. Because replication must done at the repository level, local administrators must have full privileges (i.e., unrestricted access to the entire repository). This can create a security hole for teams that include outsourcers or other development partners, as the partner’s administrator will have access to the full repository. Finally, replication creates data consistency/currency issues. Between replications, the various copies of the repository are out of sync. This disables the concept of atomic commits, and makes integration with issue/problem-tracking systems problematic. It also puts the onus on administrators to craft a replication strategy that strikes the optimum balance between the need for current information and the costs of more frequent replication.

The solution for distributed teams is a "thin client" WAN-based system that provides browser-based access to a single, centralized repository. This avoids the network latency problem (assuming the WAN is available, with adequate response time), eliminates the need for replication, eliminates the need to install servers and "thick clients" or coordinate updates, and provides secure, managed access to all repository assets.

Hosted Solutions for Geographically Distributed Teams:

Considering the myriad challenges that geographically distributed teams face given their dynamic, diverse constituencies — a hosted collaborative development environment may be an ideal solution, particularly when teams span corporate entities. Outsourcing selected business processes to hosted services providers makes sense for companies seeking to devote more of their resources to their core competencies and leverage their overall resources more effectively.

An interesting aspect of the growing adoption of hosted services is the changing attitude towards security. Up until fairly recently, companies were somewhat reluctant to externally host sensitive data or proprietary information. Today, however, most companies recognize that externally managed solutions can in fact offer greater protection than self-hosted solutions, for the simple fact that security is by necessity a core competency for the managed services provider. IT organizations chartered with ensuring the protection of software development assets and artifacts may find that a managed service approach affords them greater protection than an internally hosted solution.

VI. DISTRIBUTED PROJECT MANAGEMENT

Geographically distributed projects let managers compress schedules by employing larger workforces than could fit in a single location (collaboration on any shore), using time zone differences to increase the number of productive work hours in a day (around the-clock operations), and securing scarce resources such as knowledge experts and other specialized resources no matter where they reside (zero geography staffing).

However, these benefits come with increased risks because of the lack of face-to-face communication, in particular, the potential loss of trust, collaboration, and communication richness. Teams of software engineers need at least a minimum amount of face-to-face meetings to be effective. The agile and Extreme Programming movements suggest ways to increase communication such as pair programming, in which programmers share desks so they can see each other to efficiently understand the subtleties of design and debugging. Research on managers engaging in complex information processing that requires rich information and frequent feedback indicates that the more complex the organizational phenomena, the richer the communication must be for the manager to process it effectively. The established trend toward customers placing more risk with the developer means failures at this level will inevitably impact the developers' bottom line and eventually the company's long-term survivability.

To successfully manage complex projects, project management practices must evolve to work in a distributed world, focusing simultaneously on people, processes, and technology.

Managing global software development is not easy and risks lowering overall productivity. Still, the positive impacts should not be forgotten. A major positive effect is innovation. Engineers with all types of cultural backgrounds actively participate to continuously improve the product, innovate new products, and make processes more effective. Achievements are substantial if engineers of entirely different educations and cultures try to solve problems. Best practices can be shared, and sometimes small changes within the global development community can have big positive effects. Here are some of those best practices, which we identified over the past few years as clearly supporting global software development:

- Agree and communicate at project start the respective project targets, such as quality, milestones, content, or resource allocation.
- Ensure that commitments exist in written and controlled form.
- Have one project leader who is fully responsible for achieving project targets, and assign her a project management team that represents the major cultures within the project.
- Within each project, follow up continuously on the top 10 risks, which in a global project are typically less technical than organizational.
- Define at a project's beginning which teams are involved and what they will do in each location.
- Set up a project homepage that summarizes project content, progress metrics, planning information, and team-specific information.
- Provide an interactive process model based on accepted best practices that allows tailoring processes for the specific needs of a project or even team.
- Rigorously enforce within a product line using the agreed standard process that relates to a CMM Level 3 organization pattern.
- Provide sufficient communication means, such as videoconferencing or shared workspaces and global software libraries.
- Rigorously enforce CM and build management rules (such as branching, merging, and synchronization frequency) and provide the necessary tool support.
- Rotate management across locations and cultures to create the necessary awareness for cultural diversity and how to cope with it.
- Set up mixed teams from different countries to integrate individual cultural background toward a corporate and projectoriented spirit.
- Make teams fully accountable and responsible for their results.

Global software development is not the target per se, but rather the result of a conscious business-oriented trade-off. The guiding principles are to optimize the cost of engineering, while still achieving the best feasible integration of all R&D centers worldwide.

These needs must be carefully balanced with additional costs that might occur only at a later point. This includes staff turnover rates, which in other countries might be higher than in Europe; cost overheads related to traveling, relocation, communication, or redundant development and test equipment; unavailability of dedicated tools that allow for globally distributed tools and work environments; impacts on the learning curve, which slows down with more locations involved; cultural differences that can impact work climate; insufficient language skills; different legal constraints related to work time, organization, or participation of unions; and building up redundant skills and resource buffers to be prepared for collocated teams and unforeseen maintenance activities.

VII. CONCLUSION

Distributed development teams need to rely on automated life-cycle management tools to a greater extent than collocated teams, which can communicate and collaborate informally through ad hoc meetings and direct interaction. Life-cycle tools for distributed teams must therefore automate and support all of the tasks involved in the software development process, including source code check-in/check-out and version management; the capture and organization of other software development artifacts, such as requirements, test plans, etc.; effective project management and reporting; and especially collaboration.

Distributed teams need their automated tools to help them bridge the cultural, time and geographical barriers they face. They need intuitive, easy-to-use tools that accelerate their work, help foster a sense of community and teamwork, and facilitate the sharing of knowledge so they can be as effective as if they were all in one place. With the right tools, these teams should be able to reap benefits in increased innovation, higher product quality and faster time to market.

Effective systems will incorporate not only technological advances, but also the complementary efforts required to evolve processes and culture for success in a distributed environment. By taking this action, project managers managers will enhance enterprise performance by reducing risks and increasing the velocity by which effective decisions can be made. As the processes, tools, and technologies to support distributed work mature, more organizations are applying them to support their increasingly complex systems and software development projects.

REFERENCES

[1] Webster M., "The Requirements for Managing the Geographically Distributed Development Organization and the CollabNet Solution", White Paper, IDC, 2005.
[2] Jimenez M., Piattini M., Vizcaino A., "Challenges and Improvements in Distributed Software Development: A Systematic Review", Advances in Software Engineering, Volume 2009, Article ID 710971.
[3] Nidiffer K., Dolan D., "Evolving Distributed Project Management", IEEE Software, September/October 2005
[4] Ebert C., Neve P., "Suviving global Software Development", IEEE Software, March/April 2001.