Global software development and delivery: Trends and challenges

Kathryn Fryer
Mats Gothe
January 15, 2008

from The Rational Edge: Read about the trends surrounding global development and delivery in the marketplace and their underlying motivations, the benefits to be found in a globalized delivery model, and the challenges that such a model can present.

This is the first in a series of articles exploring various aspects of global development and delivery (GDD) as it pertains to software development. GDD is the coordination of development activities across sites to produce software applications, as well as the management across distributed repositories of the assets that contribute to those applications. “Distribution” is a broad term that can apply to one or more dimensions, including the people, artifacts, platforms, and ownership or decision rights.

In this article, we examine the trends in the marketplace and their underlying motivations, the benefits to be found in a globalized delivery model, and the challenges that such a model can present. Finally, we identify some key factors in achieving success with a global organization. These observations are drawn from our experiences speaking and working with our customers in the GDD space and from input from other IBM® representatives in development, marketing, enablement, services, and sales.

This article sets the stage for upcoming articles on the critical attributes of technology solutions to support globally distributed teams and for discussion of how IBM Rational® products fulfill those needs and contribute to the success of distributed teams. We will also propose a reference architecture that can be applied to distributed organizations to help formulate the appropriate technology infrastructure and implementation to enable those teams.
Background
At the risk of stating the obvious, globalization has quickly become a common and significant practice across a multitude of industries. Examples abound in manufacturing, automotive, financial, retail, and other sectors. High-speed telecommunications, accessible travel, global opportunities, and a host of other technological and social advancements have enabled companies to expand their reach and holdings worldwide.

Not surprisingly, technology and software development companies also figure prominently in this movement. Software teams are increasingly distributed around the world, collaborating both internally across the company and externally with partner companies, subsidiaries, and outsourcing service providers.

While companies have been establishing distributed organizations for some time, their motivations for doing so, as well as the way they distribute work, continue to evolve. Even in the short time we've been working on writing this article, some large companies have made significant changes in how they manage distributed teams. Here we'll explore the trends we've observed in globalization over the past year.

As we'll discuss, "going global" offers many benefits. However, distributed organizations face more challenges than collocated teams. We will describe those challenges and provide some suggestions for how to address them. Finally, we will review what we consider to be critical factors to ensure the success of a globally distributed project.

Market trends
Companies continue to expand globally, distributing their teams around the world through a variety of means, including offshoring, acquiring, partnering, and outsourcing. As globalization becomes more prevalent, many companies are evolving their approach and practices, perhaps demonstrating a maturation of the distributed model. Figure 1 highlights some of the changes in global sourcing, which we will discuss in more detail.¹

1. Global Sourcing is Here to Stay and is Evolving

<table>
<thead>
<tr>
<th>Evolving From...</th>
<th>... Evolving To</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offshore labor primarily in India</td>
<td>Labor in multiple geographies around the globe</td>
</tr>
<tr>
<td>Offshoring provides cheaper labor</td>
<td>Offshoring provides efficient access to a larger talent pool and leading edge technologies</td>
</tr>
<tr>
<td>Numerous tactical vendors</td>
<td>3-5 strategic outsourcing vendors</td>
</tr>
<tr>
<td>Predominantly maintenance</td>
<td>New development lifecycle; design and test</td>
</tr>
<tr>
<td>Predominantly technical programming</td>
<td>Consulting, business process outsourcing, SOA, and infrastructure</td>
</tr>
<tr>
<td>Global delivery is a specialty</td>
<td>Global delivery (chain) is the standard</td>
</tr>
</tbody>
</table>

Source: Future of IT Services, Bernardi Research, May 23, 2006; Offshoring is Outsourcing, Gartner, December 14, 2006

Global delivery will continue to grow and evolve — away from a simple cost play for lower value services.
Evolution from "It's cheaper in India" to "Talent is everywhere"

Many initial moves into the global market involved transferring work to a single site, frequently in India, and now often in China or Eastern Europe. Cost was typically the primary motivator -- decisions to transfer work were driven by the availability of skilled labor at rates much cheaper than in the United States and Western Europe.

Increasingly, enterprises are establishing development centers in multiple locations worldwide. While cost savings remains a factor, with many companies maintaining lower-cost centers in Asia and Central Europe, other motivators have also come into play:

- With a larger pool of available resources, companies can staff projects more efficiently and draw on talents from many sites.
- With teams in different time zones, companies can realize a longer workday with what is often called "follow the sun" development.
- Eastern teams handoff work at the end of their day to teams further west who are just beginning their day.
- Presence in a local market can assist in localization and internationalization of applications, as well as in marketing.

The move to strategic outsourcing

As a number of growing outsourcing providers can attest, outsourcing has been popular for some time. Initially, companies often outsourced projects tactically, selecting whatever vendor made sense at the time. Companies may have also considered whether the project itself was a good candidate for outsourcing, or they may have based selection purely on whatever project came up short of resources or funding.

Outsourcing has become more strategic: Pilot projects have led to enterprise-level initiatives, and ad-hoc efforts have led to governed processes. Outsourcing now gives way to "rightsourcing" -- assigning business processes and tasks to the right people with the right skills, regardless of where those people are located geographically or organizationally, internally or externally. Companies now examine their operations and selectively invest internally in their core strengths; then they outsource other services to one or more long-term partners and build relationships with a few key vendors.

Outsourcing is no longer just for maintenance

Similarly, the type of software development work that is outsourced is also changing. Maintenance of existing applications was once the dominant outsourcing practice in software development. Application testing is now growing rapidly, followed by innovation of new application technologies or components. Some companies that consider application development a supporting service outsource all but requirements and acceptance testing. The influence of services-oriented architecture (SOA) has led to the identification and distribution of specific services, whether technical (code) or business.
Other areas of growth outside of software development include business-process outsourcing (BPO). For example, a large pharmaceutical company may outsource the management and administration of drug trials. IT infrastructure management and consulting are also growing. An excellent example of this is Tata Consulting Services (TCS) in India, which indicated in a recent InformationWeek article that over 50 percent of their third-quarter 2007 revenue came from consulting, BPO, and infrastructure services; first-time application development made up less than half of their revenue. (Note that in the same article, TCS noted the company itself now outsources work from India to other countries where competition for talent is less fierce.)

The new normal

If global development and delivery was initially the domain of a few pioneers willing to test the waters to acquire new capabilities or cheap labor, it seems safe to say the tide has turned and GDD has become a widely-adopted practice. At some estimates, 80 percent of software projects are now global. GDD has become the standard way of doing business for most companies that depend on software.

Evolution in organizational distribution

Company organizations are changing to reflect these new approaches to product delivery. Many global organizations initially assigned product or project ownership by locality, making each location or branch responsible for delivering its own application or project, and sites worked independently.

As companies expand their presence around the world, multiple sites become a "team of teams" contributing to a global delivery chain. Each team may own a module or component they deliver upstream for integration with components from other locations or companies, culminating in a final application or product. Teams may belong to the same organization, division, or company, or to different ones.

We are beginning to see full, global organizational distribution, where multiple sites collaborate on a single component to be delivered in the chain. Sites may be very large or, as a result of workforce mobility, as small as a single individual.

In many cases, distribution is due in large part to mergers and acquisitions. In industries such as telecommunications, technology, and financial services, companies acquire market share, technologies, or market entry by acquisition, rather than by internal development. The diverse components of the new, merged business must then integrate into an effective organization and supply chain.

Infrastructure implications

These changes in business practices and organizational distribution have a definite impact on the tools and repository infrastructure required to support software delivery.

When companies began to establish several main offices or branches around the world, the typical infrastructure choice was a corporate enterprise replication backbone: Development servers were
located at each site and connected via a regular replication schedule, with local client access to the local replica usually over a LAN.

With locations multiplying and teams often becoming smaller, the costs of hardware/software and administration of replicas at every site grew significantly. At the same time, network bandwidth and reliability improved and became less expensive. The "hub-and-spoke" model became more attractive, with a few replicated servers (hubs) accessed by multiple regional sites via LAN, WAN, or Web.

Increasingly, companies are moving toward a centralized infrastructure to further reduce costs and administration overhead. This topology involves one or several centralized server environments accessed via WAN or Web clients, and may be implemented internally or provided by an external company as a hosting service -- yet another example of outsourcing.

The continually expanding availability of broadband connectivity across the globe, including the spread of wireless networks, is also making it possible to project the benefits of computing power to wherever it is needed, on demand. Another emerging direction is a "community-style" infrastructure, based on an open-source access model, where distributed team members come together on a shared team server using WAN or Web clients. The supporting corporate configuration management backbone becomes the vehicle for a software delivery chain, staging deliverables into solution integration, test, and delivery.

And of course, with the many mergers, acquisitions, and outsourcing partnerships, heterogeneous development environments have become the norm, combining different platforms, tools, and network topologies, and a mix of both commercial and open source offerings.

**Benefits and challenges**

As we’ve discussed above, companies now adopt GDD with the intent of capitalizing on the many potential benefits, including:

- Ability to gain market share, technologies, or a foothold in a new area of business or geographical market via merger with or acquisition of another company.
- Greater flexibility and a variable staffing model, afforded both by outsourcing practices (that can be increased or decreased as needed) and by differing labor rates that allow you to find the right skills at the right price.
- Lower-cost labor that can offset budget cuts and cost reductions, as well as supplement existing teams so they can expand their focus.
- Access to a broader set of skilled workers.
- Ability to leverage outsourcing providers with specialized skill and experience to address new requirements or innovations in specific areas.
- The possibility of establishing a presence in geographies that may become new markets for products.
- In general, the ability to gain a competitive edge -- to increase speed and decrease costs -- by leveraging skilled but lower-cost resources, experienced outsourcers, and overlapping time
zones. And, as more companies embrace global delivery, not doing so becomes a competitive disadvantage.

Given all the potential benefits, it seems a no-brainer for all companies that develop software to choose to establish teams across the world. However, as noted in Figure 2, which lists the benefits and challenges of GDD, a Gartner Group survey found that half of those clients queried who outsourced projects offshore or nearshore did not expect to realize the anticipated cost benefits.

**Business Drivers and Global Delivery Challenges**

![Diagram showing business drivers and customer pain points for global development and delivery](image)

**Figure 2: Benefits and challenges of GDD**

In fact, global development can be fraught with a number of issues and pain points:

- Misunderstood processes or mismatched processes between teams can lead to mistakes in work transfer, increased rework, and decreased productivity. By some estimates, productivity in a GDD project can drop up to 50 percent, with rework two to five times greater than for a collocated project.
- Communication issues can lead to misunderstandings, omissions, errors, and rework.
- Cultural issues, such as language barriers and differences in work customs or communication styles, can cause delays and affect working relationships.
- Coordinating work across multiple sites and time zones is more time-consuming and costly than for a collocated project.
- Visibility into and control of the development activities at all sites can be challenging, especially when collaborating with other companies or with teams in different time zones.
- Project metrics may be inconsistent or difficult to obtain from heterogeneous infrastructures, different processes, or company security boundaries, making it difficult to measure success.
- Political issues both within the company (organizations that fear losing work or resent the overhead of remote sites) and externally in the country or region, can lead to hidden agendas and conflicting goals.
• Organizations may not share the same objectives, especially when reporting through different management chains or different companies.
• Concerns with security and IP protection, especially in outsourcing situations in countries where IP laws are more lax, can restrict infrastructure and organizational decisions.
• Infrastructures and development tools may vary widely due to mergers, acquisitions, and outsourcing. Even internally, many smaller teams are adopting lightweight tools, frequently from the open source domain and often to support new processes, such as agile development.

These pain points increase with the number of dimensions (e.g., physical, temporal, organizational, and company) and the degree of distribution within the teams.

Critical success factors for GDD

What can a company do to meet these challenges and realize the benefits of global development and delivery? As highlighted in Figure 3, we suggest five critical factors are needed for success:

Success Factors for Global Development and Delivery

![GDD success factors](image)

**Figure 3: GDD success factors**

1. **Ensure coordination and oversight for all sites.**
   
   It's important to "think global" from the start by planning how your organization will support the delivery model you want. Establish how roles and responsibilities will be distributed across the sites, including where the management and decision-making will reside. Identify the tools the teams at each site will use and determine the infrastructure required to enable their use and any integrations with tools at other sites.

   Visibility into distant teams, especially outsourced teams, can be a particular challenge -- how can you be sure of your current actual status, with a holistic view across the entire project and deliverables? How can you reliably measure and compare between components or projects? Compare apples to apples, using consistent measures (as much as possible) for every team.
Identify up front what you will measure to track progress and status, and work with the teams to ensure they can produce the needed metrics.
Select discrete units you can measure objectively. For example, avoid measuring artifacts as "50 percent complete"; rather, indicate they are either complete or incomplete. Where possible, collect data directly from your development tools, so you can eliminate human error and subjective bias. (In some cultures, team members may be more reluctant to voice bad news.)

2. **Establish well-defined and consistent workflows.**
To avoid misunderstandings about how work should be done, define, document, and educate the team on consistent processes that will be followed across all sites.
It is probably unrealistic to expect *all* teams will follow the same process; different tools and technologies, different business needs, and inherited process and culture from acquisitions lead to diverse working styles and methodologies. An organization may have teams following waterfall, iterative, and agile methods, depending on their circumstances. Ideally, all teams in a single project would follow the same process, but in a global supply chain scenario, even those teams may be decoupled, contributing components with some level of autonomy. From the perspective of the overall software organization, it becomes most important to define the bounds or parameters that all processes must meet so that consistent touch points and deliverables across and between teams will be established. Define the artifacts to be produced, the metrics to be tracked, the measures of success, and the mechanisms for providing oversight and governance.
Within a given team, define, document, and educate all members on the processes that will be followed. If exceptions are necessary (e.g., due to access restrictions or tool differences), clearly document where they apply.
Clearly define authority and responsibilities: who does what, what is delivered to whom, and what standards are to be met. You may want to use the concept of a "RACI" matrix to determine who is Responsible, Accountable, Consulted, or Informed for each deliverable or process step.
Handoffs between teams or sites often pose problems. Using terms as concrete as possible, processes should clearly specify the deliverables required at each transition and the expected inputs and outputs. For example, help ensure clarity by providing templates for required artifacts and defining control gates with objective criteria for entrance and exit. Clearly specifying handoff deliverables and content can define a common ground and ensure both parties get what is needed regardless of steps in between. This can be especially useful when working with outsourcing providers who may follow their own internal processes.
Consider how to ensure change tracking, traceability, and accountability across the lifecycle, so you can validate if and when a particular change was implemented and what impact it had, or will have, on other tasks or deliverables.
Where possible, automate your processes and artifact workflows and enforce them in the tools themselves. Tools should make it easier to follow the process rather than to deviate from it, acting as the proverbial "carrot" to encourage compliance, rather than as the "stick."
By reducing manual steps, you both reduce the opportunity for errors and save your team members time and effort. Automation can also help overcome delays due to time zone differences. For example, having an automated build-and-build verification process can enable a team in India to complete and test a defect fix, without waiting for the build team in
the United States to come back to work. Using the tools to enact process can also ease tool and process adoption and help ensure process improvement as teams learn how to work more effectively both within their team and across teams.

3. **Manage your inventory and information.**
    Manage your assets and artifacts. Determine who needs access to which artifacts and ensure they have the required access. This includes considerations for network access and availability of the assets over WAN or Web, security permissions to perform the appropriate actions, and the simple ability to locate the artifacts (and the correct version of the artifacts). Provide traceability between related artifacts where possible, so those further "downstream" in the process are able to understand the context of the artifacts they are using. For example, enabling traceability between defects and test cases to requirements and to use case models can help those executing the tests understand what results they should be seeing and why, and whether tests adequately cover the desired functionality.
    Use a versioning mechanism for your artifacts, tracking and tracing changes, and ensuring traceability between the correct versions of different artifacts. Take a snapshot or "baseline" of your assets at defined intervals, such as the end of each iteration or milestone. Each baseline includes a complete set of associated versioned artifacts that can be treated as a unit or single asset, shared between teams, and used as a reference point. Consider an asset management tool as a way to store and manage these aggregations of related and versioned artifacts and make them available for reuse.

4. **Communicate, communicate, communicate.**
    Clear and effective communication could be the most important of the success factors; it is certainly key to building any solid team. Successful projects encourage collaboration and promote team building and social networking. In a collocated team, collaboration can happen informally in the hallway or "at the water cooler," as well as in face-to-face meetings. With distributed teams, you must work harder to keep everyone in the loop.
    Consider a communications plan that defines how you'll keep all team members informed and working toward common, shared goals. Communicate frequently and openly and work to build trust between team members.
    Take advantage of any and all available collaboration mechanisms. Many of the standard mechanisms are well-known and widely used: e-mail, teleconferences, and various notes or documentation facilities built into development tools. Most of us are also now familiar with chat and instant messaging, Web conferences, and screen sharing. Wikis and team blogs are other mechanisms whose adoption has been spreading recently. Establish a "virtual water cooler" by way of instant messaging, team rooms, or other collaboration mechanisms.
    Trust can be hard to establish in a distributed team, where interpersonal contact may be limited. Create online profiles of team members in a central location, including pictures and even audio clips, so others can connect faces and other personal information to the name and voice. If budget allows, periodic face-to-face meetings provide opportunity to establish relationships with the benefit of nonverbal communication (although note that cultural differences can also affect these nonverbal modes of communication).
    It goes without saying that you should schedule team meetings with consideration for the different time zones of the teams' locations. In cases where no time will conveniently work for all teams, schedule meetings at alternating times, allowing teams to take turns attending at convenient or inconvenient hours. When scheduling meetings and milestones, you must
also consider the holiday schedules that vary across the different countries and cultures. Scheduling a major milestone when one region has a month of vacation or a significant political or religious occasion is likely to cause schedule delays and potential team conflict. Where possible, collaborate in the context of the object or artifact under discussion. Use URL references to share content via e-mail or chat. Annotate or make comments directly on the artifact, and preserve comments with the artifact for future reference. Keep clear records of communications and decisions, so those who were not involved know what was decided and can understand the rationale. For team members in other time zones, this can also save a lot of time in round-trip queries, where an entire workday may elapse before a question on a particular design decision or rationale can be answered by someone on the other side of the globe.

Textual records can also help to address language barriers. Those team members who are challenged by the spoken language can read (and reread if necessary) the written documents in order to better understand decisions and context. Similarly, it may be helpful to provide text backups to vocal communications; for example, shared charts, text captioning, and chat "back-channel" in a teleconference. (Note that such backups also help those with hearing difficulties.)

Be aware of the diversity across your organization and the differences in culture and work habits across sites. For example, team members in some cultures may hesitate to question superiors; others may be more confrontational and challenge authority. Understand these differences and ensure all team members understand them as well, so you can adapt and mesh working styles between teams. Some companies offer resources in diversity or cultural training; in other cases, you may need to learn from experience or ask others who have worked with similar teams.

5. **Establish a flexible, adaptable IT infrastructure and architecture.**

Your IT infrastructure, including your networks and development tools, should enable you to support a variety of distributed organization scenarios. Ensure the tools and platforms you select provide the capability and flexibility you need to support the topologies you require, whether they are replicated, hub-and-spoke, centralized, or some combination. Specific considerations include:

- Client access to servers via LAN, WAN, or Web
- Security and the ability to restrict access to artifacts as needed
- Ability to interact and integrate with other tools and platforms in a heterogeneous operating environment
- Support for team collaboration
- Ability to support necessary languages and code pages

**Summary**

Organizational distribution and global development and delivery are here to stay. Outsourcing and rightsourcing are practices that will continue to grow, evolve, and mature. For many companies, a global team is no longer an *option*, but an *enterprise strategy* and a way of doing business.

The benefits of a global software team can be significant, including cost savings, flexibility, access to key skills and experience, and quicker time-to-market. However, there remain many challenges...
that can adversely affect the ability to realize those benefits, including cultural, communication, and process issues.

Delivering on the critical success factors of:

- Global coordination and oversight
- Well-defined processes and workflow
- Inventory and information management
- Clear and accessible communication
- Flexible and adaptable development infrastructure

can help you overcome the challenges and pitfalls of global software development and realize the benefits of successful global delivery.

The next article in this series will describe the key attributes of software solutions for GDD and expand on how to build the flexible, adaptable infrastructure you need for your teams to be successful.

Notes


About this series

Solution Architects in the Rational development organization, Kathryn Fryer and Mats Gothe develop customer-based scenarios to document best practices with Rational tools and identify opportunities for future solution enhancements. This series of articles is based on their recent work targeting GDD approaches and Rational support for distributed development.