# Stride Methodology – Lean Agile Development in a Dual-Shore Environment

**Yash Talreja**

The Technology Gurus

Dual-shore development introduces new challenges to any process. Especially when the offshore team is a contracted organization, issues of accountability, utilization, and long-term planning come to the fore. Popular agile frameworks do not address these issues, leading many companies to continue using very traditional and heavyweight approaches. The lack of a workable alternative was the motivation for the Stride methodology, which was specifically designed as an agile, iterative, and incremental software development process for dual-shore settings.

Stride is inspired by the widely used Scrum methodology, but has several differences that make it better suited to a dual-shore environment. It has been used successfully in large-scale, long-term, dual-shore projects, combining the advantages of agile development with the structure necessary to manage remote teams.

## Existing Software Development Processes and their challenges

For most of software development's history, a company's approach would fall somewhere on the spectrum between "ad hoc" and heavyweight. Ad hoc projects have little or no documentation to support the development and testing process. This approach allows for easy changes to requirements, but usually introduces tremendous inefficiencies in terms of misunderstood requirements, design conflicts, and difficulties in maintenance. Ad hoc processes quickly become less efficient as teams grow larger.

Heavyweight processes, by contrast, create a surfeit of documentation. IEEE 12207, entitled "Standard for Information Technology – Software Life Cycle Processes," represents the pinnacle of these ideals. It describes thirty specific documents that an organization should create and maintain for each project, including documents that describe the process necessary to revise any of the documents. These processes are so burdensome that they add tremendous cost and staff requirements to a project.

The choice between the chaos of ad hoc approach and the burdensome regimentation of a heavyweight process was never an appealing one, as both had their own costs. In the mid-1990s, some development groups tried approaches based on assuming that requirements will change, and therefore focusing on short-term planning. These processes stressed short development iterations that deliver integrated and working code. They thus avoided the common problem points where projects slip repeatedly because code is written but not tested, or working in isolation but not integrated. A group of proponents of these methods coined the term "agile methods" in 2001.

One widely publicized agile method is Extreme Programming or XP for short. XP started at Chrysler, as an effort to reorganize a struggling project to create a new system. XP dictated several precepts. Two controversial ones were pair programming and test-first design. Pair programming dictates that programmers always work in pairs, constantly reviewing each other's work. Many case studies have described this approach failing when many of the programmers had specialized skills, or varied widely in experience level.

Test-first design says that a programmer must create an automated unit test before performing any implementation work. The programmer then creates the simplest implementation that satisfies the unit test. In practice, this approach breaks down in many situations. For example, it is often difficult to create automated unit tests for modules that manage user interface elements or interact with other systems. Some requirements, such as system performance requirements, can't even be expressed as automated unit tests. XP's success has been mixed at best.

Another well-publicized method, SCRUM, started in the mid-1990s and gained popularity after it was described in a 2001 book. SCRUM calls for teams to self-organize, with managers merely providing the environment. It eschews any formal project management or resource management, instead dealing with any issues at daily team-wide meetings. This can work well for small, homogenous teams, but quickly breaks down with larger teams.

A common management criticism of SCRUM is that the lack of longer term planning makes it impossible to have any visibility into how long resources would be needed or when a truly releasable product which meets market requirements might be ready. Many developers complain that the egalitarian ideals fall victim to strong personalities who kept all the best tasks for themselves. Worst, the shared team responsibility and limited involvement of managers makes it difficult to hold anyone accountable if things go wrong.

## Overview of the Stride Methodology

Stride encourages longer-term planning than most agile methods. It focuses primarily on two to four week "strides" of development, each of which represents a complete iteration with integration and testing. However, Stride also allows planning at the "leg" level, consisting of several strides, and the "journey" level, equivalent to a project or product release version. Typically, the team will review and revise the leg and journey plans every few months. This balances the agile assumption of constantly changing requirements with the practical reality of needing to plan for an offshore team's utilization.

A dual-shore methodology must adapt to the difficulty of communications across distant time zones. For example, Scrum calls for daily project-wide meetings, which Stride replaces with meetings twice a week: one on Monday to discuss plans for the upcoming week, and one on Friday to review progress. For the larger projects common to dual-shore environments, only the Group Leader from each group attends, easing the burden when there is little or no overlap in the teams' working hours. The groups hold their own individual daily meetings to address issues related to their particular tasks.

When an external supplier runs one of the locations, accountability, responsibility, and productivity become important concerns. Many agile processes follow an idealistically egalitarian and democratic approach, in which developers self-assign tasks, and no one assigns responsibility for problems. This approach makes remotely measuring performance and addressing subpar work very difficult. Stride instead allows leaders to assign tasks to particular groups and developers, making it easier for the client to detect and address any issues.

## Roles

There are four key leadership roles in a Stride project:

**StrideMaster:** The StrideMaster serves as a facilitator, making sure that problems don't interfere with the project team reaching its goal. The StrideMaster is not a traditional project leader, but instead enforces the Stride rules to ensure the process works as intended. The StrideMaster also keeps the team focused, serving as a buffer between the project team and any distracting influences.

**Product Owner:** The Product Owner represents the voice of the customer, ensuring that the Stride Team is working on the correct things from a business perspective. The Product Owner is responsible for defining the work as a list of features or "stories," and then prioritizing those features.

**Group Leads:** Each Group Lead heads a small group of two to five people. Each Lead represents his or her group in meetings, works with the group members to come up with effort estimates for tasks, and assigns tasks to group members when appropriate.

**On-shore/Client Manager:** The On-shore Manager can be one of the Group Leads from the onshore (client) teams, or can be an independent person. He/she is maintaining the relationship with the offshore provider, making sure that the offshore team lead(s) are signing up for sufficient work and are delivering high quality work in the promised timeframe and taking corrective action when there are problems.

Usually, the Product Owner, StrideMaster, and the Manager will all be at the "home" or client location. The Group Leads reside wherever their groups are, and will be present in all locations. Note that Manager has a much more active role than in many agile approaches, because they track the performance of the team members and hold them accountable.

## Project Planning

Documentation forms an essential part of all processes. Due to the additional complexity of dual-shore development, quality documentation is an essential part of Stride. Since the teams may have little overlap in working hours, issues that might resolve with quick communication in a single location can cause the waste of a day in a dual-shore setting. All material should be available online from both locations, typically in a project Wiki or other content management system.

For high-level project planning, Stride calls for three key documents:

**Project Vision Statement**: This describes the goal of the project in a couple of sentences. While brief, it serves the essential function of reminding the team members what they're working towards.

**Project-Level Product Backlog:** This collects all of the possible project features, whether required, "nice to have," or "wish list." This Backlog assembles these features into a list, typically in a spreadsheet or in a table on a project Wiki page. The more likely features have more detailed documentation so that Group Leads can assemble rough work estimates, usually with a granularity of weeks. The work estimates allow planning of each Stride and Leg, as well as helping the Product Owner to prioritize borderline features.

**Leg Definition:** This is a breakdown of the Project-Level Product Backlog into various Legs. Each leg can be two to four months long. The Product Owner, the StrideMaster, and the Group Leads

cooperatively decide the dates and number of Legs for the project. Descriptions of the legs for the next several months will include lists of desired features from the Product Backlog. Legs further in the future may simply list themes describing the areas of work to be done, knowing that the exact work items will likely change.

For all legs, the actual feature list will change based on changes in priority, resources, and variance between estimated and actual implementation times. While the Leg Definition is a living document, it plays a valuable role for planning purposes. It allows rough estimation of project length and resource needs, both of which are important when working with an offshore team.

## Leg Planning

At the beginning of each leg, the Product Owner, the StrideMaster, and the Group Leads meet to agree on the features from the Product Backlog to implement. The Product Owner verifies the features priorities. The Group Leads identify any dependencies between features. The participants then choose a set of features so that the total estimated work effort is consistent with the capacity of the team. The team then allocates each of the features to one of the Strides (iterations) within the leg.

For each leg, there are two major planning documents:

**Leg-Level Product Backlog:** This need not be a separate document, but can exist as a section of the Project-Level Product Backlog. The team now refines the previous rough effort estimates to an intermediate level of detail. At this point, there has been neither design work nor detailing of the tasks necessary for each feature. As a result, the estimates typically have accuracy in terms of one to five days.

**Stride Definition:** This is a breakdown of the Leg level product backlog into a series of Strides. The Product Owner, the StrideMaster, and the Group Leads work together to assign the leg's backlog items to particular Strides, and set the dates for each Stride.

The Stride Definition can change as the leg progresses. The actual time necessary for implementation will usually vary from the estimates, and shifting priorities and resource levels may necessitate changes.

## Stride (Iteration) Planning

In keeping with the agile assumption that requirements will change, there is no investment in design activities until it is nearly certain that the project will contain a particular feature. It is at the start of each stride that detailed design and implementation planning occur.

For each stride, the team holds a one-day detailed planning session. Ideally, the Product Owner will have already added enough detail to the backlog items to allow team members to decompose them into tasks. Any necessary clarifications are addressed during the planning session. With the detailed task breakdown, the team can now create estimates that are much more accurate.

At this point, the capacity of the individual groups is well known, any dependencies have already been addressed, and the team can plan with a higher degree of confidence. The Group Leads assign specific tasks to each team member, and the team commits to completing the listed tasks. By doing the planning right before the actual work, the team can have a high degree of certainty of actually delivering the expected features.

The two planning documents for each stride are:

**Stride-Level Product Backlog:** This need not be a separate spreadsheet, but can be a section of the Leg-Level Product Backlog. The backlog items are now backed by detailed descriptions. Effort calculations are extremely detailed based on the component tasks necessary to complete each feature.

**Task Backlog:** This is a breakdown of the individual tasks necessary to complete the backlog items for the stride, the effort estimate for each, and their assignment to specific team members.

## Regular Meetings

Regularly communication is essential to success both in agile processes and in dual-shore development. When combining the two, the importance only increases. While some communication will take place through email, posts to a project Wiki, and instant messaging, there is no substitute for real-time meetings. It is essential to have quality teleconferencing equipment to ensure that all team members can hear each other.

Stride has meetings at prescribed times:

**Daily Group Meetings:** Each group meets every day in a meeting run by the Group Leader. Group members review their plans, check progress, and raise any issues that they need the Group Leader to resolve or communicate to the team at large.

**Weekly Meetings:** The team holds meetings twice a week. For a smaller team, this would include all team members. For a larger team, only the Group Leads and StrideMaster attend. Because this meeting includes members from all locations, the team must schedule it at a time convenient to everyone.

Each Monday, the team coordinates plans for the coming week, agreeing to specific tasks and identifying any potential issues that need attention. Each Friday, the team reviews the week's progress and raises any critical issues. Based on the information gathered at the Friday meeting, the StrideMaster updates "burn down" charts that show the amount of work remaining in the current stride and leg. These charts are public, so that everyone can see at a glance the team's progress towards its near-term goals.

**Stride Review Meetings:** At the end of each of the iterations, the team demonstrates the newly completed features to the project's sponsors, such as executives or customers. The team shows only completed work; Stride rules forbid carefully orchestrated demonstrations that make a feature appear complete. Any planned features that the team didn't complete return to the backlog, and may be reattempted in the next (or a later) stride.

## Accountability

When two or more locations share a code base with continuous integration, it is far too easy for sloppy work, or even sloppy communication, to waste large amounts of time. Due to dependencies, one underperforming team member can have a ripple effect. When an external supplier runs one of the locations, executive-level negotiations may be necessary to resolve problems.

The Client Manager reviews weekly progress to determine whether there may be a problem with the offshore site's performance. Each group's commitment to complete tasks on a weekly basis provides rapid visibility of performance problems. To identify more chronic or systemic issues, after each Stride

Review Meeting, the Client Manager meets with the Product Owner and the StrideMaster. Both the Product Owner and the StrideMaster are there purely to provide information, not to assign blame.

The Manager reviews the execution of the stride, and look for anything that exceeds the normal level of variance expected in a software project. For example, if an offshore group is repeatedly failing to deliver on their commitments, it may require communication with the operator of the offshore location. Other possible concerns might include team members not properly following the feature descriptions, deviation from the features descriptions agreed to by the Product Owner, or groups not proactively communicating issues that will affect the rest of the team.

Manager involvement allows the person performing the StrideMaster role to avoid acting like a project leader. By enforcing accountability from outside the team, they let the team members work together cooperatively, yet ensure that chronic problems are addressed.

## Conclusion

Companies adopt both dual-shore approaches and agile methodologies with the intent to reduce development cost and project durations. Combining the two can indeed have a synergistic effect. However, this is still very much an emerging field. Companies have been attempting dual-shore agile projects for only a few years.

When dual-shore development first became popular, many companies experienced problems due to process breakdowns. The usual response was to hew more closely to the recommendations of IEEE 12207 or similar traditional methodologies. The result was a severe trade-off: successful cost reduction for a painful loss of flexibility. Many companies found that when the inevitable changes in requirements occurred, any cost savings quickly disappeared.

The Stride methodology allows companies to maintain the responsiveness characteristic to agile processes without sacrificing the accountability necessary to manage an offshore location. It has worked successfully on multi-year projects with several dozen team members. By adopting a proven methodology, companies can greatly increase their odds of success.

## About Us

The Technology Gurus is a boutique consulting company that assists companies with their interim CTO, software development and outsourcing management, and M&A-related technology due diligence needs. To find out more about our services, visit us at www.The-Technology-Gurus.com or call us at 650-339-6240.

Yash Talreja is the principal consultant at The Technology Gurus. Yash has more than 17 years of senior/ executive management level experience, at both large companies such as Oracle, PeopleSoft and Amazon.com and startups, such as Televoke and MessageGate. Yash is also the founder of the Lean Agile Group, an organization of more than 4,500 practitioners of Lean Agile software development methodologies. Yash can be reached at yash@the-tech-gurus.com.