no. 2-0027

TUCK SCHOOL OF BUSINESS AT DARTMOUTH
WILLIAM F. ACHTMEYER
CENTER FOR
GLOBAL LEADERSHIP

Improving Productivity at Infosys[1]

Introduction

At the height of monsoon season in 2000, Nandan Nilekani, chief operating officer of Infosys, reviewed a report from a just-completed custom software development project for a promising new client. The report indicated that Infosys invested more time than planned on the project and, consequently, had lost money. The client was satisfied with the project, but Mr. Nilekani was unhappy with the resultant loss.

As Infosys's rapid growth continued, the importance of reinforcing the company's process discipline spiraled up. Project teams that invented new ways to get their work done, rather than following the efficient paths of prior teams that solved similar problems, most often would fall short. To prevent this, Mr. Nilekani planned to redouble Infosys's efforts to strengthen its infrastructure for documenting and promulgating internal best practices.

One step Mr. Nilekani took was hiring Satyendra Kumar, a career quality expert. Mr. Kumar elaborated on the need for a robust process documentation system:

> It is absolutely imperative to document everything in a 50,000-person organization. You can't just count on people keeping in touch with each other. With the explosive growth we have experienced, it becomes difficult simply to find people who have worked on projects similar to your own. Our objective is to establish a well-defined process for everything that is repeated.

By 2006, Mr. Nilekani, by then CEO, and Mr. Kumar had dedicated tremendous effort to expanding the capacity and capabilities of Mr. Kumar's quality assurance (QA) department. The group standardized, measured, analyzed, and documented work process company-wide and set goals for improvement.

Establishing Productivity Baselines

At the foundation of Infosys's approach to generating productivity improvements was an online system for documenting processes, built by the QA team and the internal information systems group. Infosys dubbed the system PRidE: Process Repository @ Infosys for Driving Excellence. Within PRidE, any employee anywhere in the world could view all of the company's standard work processes, which spanned the gamut from core

[1] In writing this case, the author assumed familiarity with the content of a related case, "Infosys: Maintaining an Edge."

work processes for custom software development to processes within support functions, such as finance and human resources.

PRidE was loaded with procedures, checklists, protocols for interviewing clients, organizational models, instructions for dividing work between the client site and offshore, and much more. Mr. Kumar had benchmarked the robustness of the company's processes against standards set by several certification bodies and, by 2006, felt comfortable that Infosys was among the very best in the world.

Critically, Infosys documented not just *how* to get the work done but standards for *how long* each step should take. In fact, the QA department employed a small army of statisticians to analyze data from each project and ensure the time and budget expectations associated with each documented process were up-to-date.

Newly promoted project managers at Infosys went through a rigorous certification program and learned how to use a software tool for integrated project management. Combining the tool with the data available within PRidE, project managers developed reliably predictable budgets, timelines, and even expectations for defect rates to which they would be held accountable. Many project managers believed one of the strongest advantages Infosys held over its competition was its process discipline. Clients valued predictability.

During the planning phase, representatives from the QA department were assigned to support project leaders by verifying plans, timelines, and budgets. In a project's early phases, project leaders interacted with their QA representatives almost every day. Because the QA department had exposure to every project at Infosys, the best QA representatives alerted project leaders to newly discovered practices. Once a project was in progress, the QA representative intervened less frequently, generating interim reports periodically and helping identify root causes and corrective actions when projects fell behind. QA representatives also ensured the project teams operated in accordance with quality assurance directives. Once projects were completed, project managers discussed performance and identified lessons learned with their teams. QA representatives supported end-of-project reviews by analyzing results and offering feedback. They also updated the PRidE system with any improved practices.

When Infosys launched new services, the QA team allowed a period of discovery, careful not to lock into any one process too soon. After any acquisition, QA teams compared documented processes in both companies, analyzed which was better, and updated PRidE.

Improving Productivity by Improving Process

The PRidE system was intended to ensure that each new project started with a plan at least as good as Infosys's prior best efforts on similar projects. Clearly, if process discipline became *too* rigid and *no* departures from established routines were ever allowed, then the potential for improvement was limited (and meeting unique customer requests would be difficult). Therefore, *once project managers acknowledged and understood prior best practice*, Infosys *encouraged* departure from it, allowing up to (a qualitatively assessed) 25 percent change in process per project.

In fact, Infosys's QA team routinely demanded that project managers *exceed* existing productivity baselines. There were many factors that shaped *how much* in excess was expected. The company considered how new technologies might save time for project teams, for example, and, to the extent possible, gathered data from external sources to benchmark Infosys's performance against competitors. In aggregate, Infosys sought productivity improvements of 10 percent per year, a figure the senior management team reckoned was high enough to motivate innovation but not so high that teams would routinely falter under excessive pressure. Business units that fell behind in one year were expected to catch up the next year and received extra attention from the QA group to get there.

Ideas for improvement arose throughout projects. In fact, to ensure all employees had a forum for making suggestions, each business unit maintained formal mechanisms for collecting ideas. Project teams typically identified the highest-impact insights into productivity improvement during the planning phase. Opportunity varied from project to project. Some projects were straightforward repetitions of past work, while others offered promising openings for improvement.

Under pressure, project planners could be tempted to simply adopt the existing approach outlined in PRidE. Mr. Kumar commented,

> We allow up to 25 percent variation, but project leads do not always take advantage of it because it is easier to cut and paste from the previous project.

Infosys kept a simple approval process for departures from documented best practice. Departures from PRidE needed only to be agreed to by the project manager, the delivery manager, and the QA representative, who might object in particular to experiments that other teams had attempted and failed. (Experiments that required additional expenditures might also need approval from Infosys's procurement group.)

To encourage experimentation, Infosys also established project-level recognition programs. In 2006, nearly 200 project teams applied for recognition for process improvements. Nonetheless, Mr. Kumar was unsatisfied and felt the company needed to redouble its effort to inspire and catalyze productivity improvements on the front lines. Other groups were involved in the effort. SETLabs, for example, was working on ways to embed pathways for process innovation into project plans.

Improving Productivity by Driving Even More Work Offshore

Project evaluations, crucial to advancement for project managers, were based on a number of metrics, including quality, customer satisfaction, employee satisfaction, timeliness, and, critically, margins. All project managers were aware that a powerful way to improve margins was to shift more work offshore.

"Percent of work completed offshore" was a closely watched metric on every project. Infosys did not evaluate managers on this metric directly nor tie compensation to it. It was watched closely because it was tied so directly to costs. There was no specific target for the measure, and the average varied dramatically across business units. Infrastructure management was heavily offshored; enterprise system implementation was not.

The fraction of work offshored also varied by project stage. The heaviest opportunity to offshore was in the middle of most projects, with more on-site client interaction required at the beginning and the end. For example, an early step in custom software development projects was to map "as is" work processes within the client organization, and that had to be done on site.

Infosys broke down the "percent of work completed offshore" measure by project stage and sought improvement in each stage. Each project manager worked with both the QA team and clients to figure out how to shift more work offshore. According to one Infosys project manager, in 2001, most high-level design work was conducted on site, but by 2006, nearly all of it was done offshore. Even most testing was conducted offshore by 2006.

The growth of the Internet and corporate networks stimulated third-party software developers to build more software tools that enabled hardware and software maintenance and management to be conducted remotely. By 2006, many employees in large corporations were familiar with the experience of turning over control of their PC to an IT employee in a different location; they could even watch the cursor move around the screen, controlled by the invisible hand of the IT employee doing repair work and troubleshooting. At a business unit level rather than a project level, through the routine annual planning process, Infosys made frequent investments in similar tools for servers and mainframes and implemented them in order to shift more work to India. The IMS enterprise capability unit (ECU), for example, invested in expensive monitoring systems. Their operations centers were full of expansive wall-mounted screens to display various activities within the systems they managed for clients.

When Infosys launched new services and new business units, it was not necessarily clear how much of the work could be handled offshore. Nonetheless, the basic principles were the same: break down the work into specific tasks and try to identify those that required real-time, face-to-face collaboration and those that did not. When Infosys launched the enterprise solutions (ES) business unit, it did not appear that much work could be offshored. Implementing enterprise systems did not involve much custom software development. Instead, projects required customization of software packages developed by other companies, such as SAP and Oracle. That intuitively felt like a process that required client collaboration in almost every step.

That intuition proved untrue. ES was able to move 40 percent of the work offshore within its first year of operation and pushed the offshore fraction to 60 percent within a few years. The rate at which ES increased the fraction of work offshored was limited mostly by client comfort with the idea.

Improving Productivity by Increasing Reuse

While PRidE helped project managers quickly formulate an effective organizational approach to tackling large-scale projects, process knowledge was only one of several forms of knowledge that Infosys endeavored to capture and reuse. All other forms of knowledge—from modules of software code to papers to website reviews and book reviews—accumulated in the knowledge management (KM) system. By 2006, there were thousands of documents in the system, plus a "people knowledge map" that helped employees identify in-house experts in various topics.

Dr. J.K. Suresh, who managed the system, commented on its importance:

> Today's corporations look dramatically different from the way they looked in the early 20th century. They look different because they are structured to mobilize knowledge. Knowledge management is not just about protecting intellectual property. It is about capturing solutions to problems and then helping others who are confronting similar problems find those solutions. Until the mid-1990s, Infosys could do this informally because everyone knew everyone else. That is not the case anymore. We employ tens of thousands, spread over the globe.

As part of the process of wrapping up each project, teams considered what contributions they could make to the KM system, although any employee could contribute at any time. There was, in fact, no process for formally reviewing submissions to the KM system. New submissions were sent to subject-matter experts for review, but these experts reviewed only 20–30 percent of the submissions because they were busy. Nonetheless, any time anyone read a document in the KM system, that individual could rate it, regardless of his or her qualifications. The KM system calculated an aggregate rating for each document, weighing ratings from employees with greater tenure or greater subject-matter expertise more heavily.

Any employee could access the KM system by either using a keyword search (the KM system identified keyword tags automatically; users could also create their own) or navigating through a taxonomy Infosys had created. In presenting search results, the search engine prioritized documents based on their rating and how frequently and recently the documents had been used.

To motivate contributions to and use of the KM system, Infosys created a reward system based on "knowledge currency units" (KCUs). When employees wrote a document and contributed it to KM, they received KCUs based on ratings from others who had benefited. For a few years, KCUs could be converted to money, but Infosys later discontinued the practice so that employees were not conditioned to believe they deserved a "biscuit" every time they did something good. KCUs remained a psychic reward and looked good on performance reviews. Senior executives also reinforced the importance of KM. Mr. Murthy reviewed 12–15 projects per year, and among his most important inquiries were "How much did you contribute to the KM system?" and "How much could you have saved had you leveraged the KM system more fully?"

Infosys relied on project teams to report savings achieved through the use of KM to evaluate its effectiveness. One internal study comparing pairs of similar projects demonstrated that teams making heavy use of KM cut costs by 2–6 percent.

As of 2006, Dr. Suresh believed motivating Infosys's experts to spend more time reviewing and rating submissions was an important avenue to an even more effective KM system. Another avenue was improving documentation of software submissions to clarify the context of their use. Often, a software module that worked well in, say, telecom needed substantial modifications before it could work in banking.

From Small Steps to Giant Leaps

Despite all the energy dedicated to routinely improving processes and productivity, Infosys's senior management team remained hungry to identify more powerful mechanisms for improving processes. They wanted to do this through investments in major projects that would yield major leaps in performance, not through many small steps, project-by-project. In addition to establishing SETLabs, the company created centers of excellence (COEs) within each business unit (some business units had more than one) to fund investments in projects of greater scale than could be funded within a single project. Mr. Kumar reinforced the importance of these mechanisms:

> We demand that every project team perform at a level several percentage points higher than documented best practice. But it is only when you ask for even bigger leaps, say, 25 percent or more, that people take a step back and consider completely new ways of doing business.

The COEs, with budgets of roughly 3 percent of revenues, built and collected tools, methodologies, software modules, reference architectures, and more to make future projects more efficient. They invested in projects that had carefully developed business plans with clearly defined business impact, and they formally reviewed the projects twice per year. Some COEs also developed training programs for Infosys employees.

The number of people working in a COE varied, depending on size of the practice to which the center belonged. In one case, a practice of 1,800 people had 35 employees assigned full-time to the COE. Each COE was headed by an anchor but was managed "like open-source software" so that any employee within a practice could contribute to works in progress. The COEs also maintained connections to outside experts and thought leaders in relevant fields.

To make progress on projects, COEs relied on the full-time COE staff plus "the bench"—employees who were in between billable projects. Historically, Infosys maintained an employee utilization rate of about 75 percent, but that level fluctuated with client demands. COE anchors selected people from the bench to get involved with certain projects. Although staffing through the bench meant COE projects faced staffing disruptions, it also meant COE project teams included people with recent and relevant frontline exposure.

Since Infosys's bread and butter was software development, many of the projects the company funded in hopes of achieving a radical leap in productivity were software tools for its own project teams to use—software for software developers, that is. Two of the most interesting stories follow. The first project was led by SETLabs, the second by a COE.

InFlux: Getting Custom Software Development Off to a Good Start

Much of the work Infosys did was precise, scientific, and technical in nature, yet each project began with a phase rich in more “human” pitfalls. The objective of this phase was to write a crisp and clear definition of *exactly* what the software application to be developed must accomplish, a task simple to describe but devilishly difficult to do.

Companies hired Infosys for custom software development because they wanted to operate more efficiently. Infosys partnered with clients to simplify and standardize work processes, eliminate paperwork, automate information flows from one work step to the next, and make existing computer systems interoperable.

To design the client’s future work processes, Infosys needed to understand the client’s existing work processes and the software applications used to automate them. Even with extensive interviewing, that could be difficult, for several reasons. Each company had a unique jargon. Plus, some IT departments were better than others at cataloging and documenting their applications. Large corporations had hundreds of applications to keep track of. Furthermore, many employees inside client organizations understood only their own particular tasks and maybe the tasks that immediately preceded and followed. Finally, in many cases, there simply was no standard process. Rapidly growing companies were often too busy to make the effort to formalize work processes, so local work teams created their own ad hoc versions of “the way we do things.” Companies that grew through acquisition had a similar problem, as each acquired company was certain to follow its own unique script.

The volume of information collected during this phase was staggering, as it extended to a fine level of detail, right down to descriptions of fields in databases and instructions for messages between applications. For example, Infosys might document the specific format an insurance client used to send information between a customer database and a claims-processing database.

Conversations at this fine level of detail were comfortable for IT specialists but not for most businesspeople. As one Infosys project manager put it, “If you ask a software developer to describe an insurance claims process, within a few seconds you’re talking about database design or screen design.” The level of detail obscured the big picture even for IT specialists. As a result, major opportunities to improve a system could be easily overlooked.

Many Infosys project managers created high-level diagrams (showing task sequences, decision points, information flows, etc.) to facilitate conversations among businesspeople and IT specialists from both the client organization and Infosys. These conversations proved pivotal in transitioning from mapping existing processes to designing improved ones. There was no commonly accepted way of creating these diagrams. Each project manager developed his or her own style.

Once Infosys and the client had agreed on a high-level process design, Infosys could proceed to draft specifications for the software, forging agreement with process leaders in the client organization each step of the way. Inevitably, Infosys had to navigate conflicts among groups in the client organization regarding the best possible process design or groups that felt their processes needed to remain unique rather than become standardized.

With specifications complete, generally several weeks into a project, Infosys's software developers could move on to the next phase of their work—writing code. If there had been good, clear communication throughout the design phase, the probability of a successful project outcome was very high. But good, clear communication—between technology specialists and business generalists, between clients and consultants, and between client groups with differing priorities—was difficult to achieve. In fact, delays of 50 percent beyond the plan for writing software requirements were common as project managers worked to ensure all parties were on the same page before starting to write code.

Standardizing the Process of Writing Software Specifications

In late 2002, Infosys launched a new software tool, known as "InFlux workbench," for its project teams. The senior management team led a company-wide communications effort as part of the launch. The two chief architects of InFlux, N.S. Nagaraj and Srinivas Thonse, both near their 10-year anniversary with Infosys, were pleased. It was a major milestone in their effort to develop a standard, repeatable methodology for writing software requirements. Since 1998, they had been working on the methodology they eventually dubbed InFlux and had been training Infosys project teams to use it. Now that they were able to offer a software toolkit that made InFlux even more powerful, they anticipated far more teams would adopt the approach.

Prior to their work on InFlux, Mr. Nagaraj and Mr. Thonse worked on advanced software architecture projects for Nortel, one of Infosys's largest customers at the time. In 1998, the business unit the pair worked for established a special projects group, known as the COM Factory. It included Mr. Nagaraj, Mr. Thonse, and six other technology experts. Mr. Nagaraj recalled, "We were people who had a flair for writing and researching the latest developments in technology." In addition, the group spent a great deal of time providing technical help and mentorship to project teams in the field.

At the time, Infosys did not have an R&D function at the corporate level, but the COM Factory in some ways became the blueprint for it. When Infosys formed SETLabs in 2000, the COM Factory became part of it.

Years of experience had taught both Mr. Nagaraj and Mr. Thonse the early stage pitfalls in a custom software development project. Through study of outside publications, they also had begun to understand an emerging paradigm in computer science, known as business process management (BPM), which showed how companies could more easily optimize and adapt their work processes as business conditions changed.

Applying concepts from BPM to their own projects, Mr. Nagaraj and Mr. Thonse began to see the power of a top-down approach to business process design. Rather than simply automating client processes as they existed, from the bottom up, they were improving process designs. They were broadening the scope of their work beyond information automation. They were solving *business* problems, starting with the customer value proposition and working back to the most efficient way to deliver that value. They were gaining new insights that they would have overlooked in the past. They had seen the power of modeling and communicating business process designs with simple, graphical diagrams and the wisdom of not getting drawn into the details too quickly.

The two men began to consider what it would take to develop a standard, repeatable approach to modeling processes and writing software specifications. They visualized a methodology that would generate outputs (diagrams, flowcharts, organizational charts, etc.) that people from a wide range of backgrounds and perspectives could analyze easily and then translate into software requirements. After developing their approach a bit, Mr. Nagaraj and Mr. Thonse shared it with a customer, and they received encouraging feedback.

As they further refined the methodology through projects they worked on directly, the men gained confidence. They were highly respected within their business unit, and they had the ears of several project managers and engineers. More and more project teams started using the methodology.

By creating and giving training programs that used past projects as case studies, Mr. Nagaraj and Mr. Thonse hoped to expand the use of InFlux beyond their established power base within the company. As they did so, however, the reaction was less enthusiastic. Some co-workers immediately saw the value of the new approach, while others felt the methodology did not improve their own method of writing software requirements. Not all clients immediately saw the value either, particularly clients from IT departments that had traditional training in automating processes from the bottom up.

In early 2000, Mr. Nagaraj and Mr. Thonse adopted a new corporate-level platform from which to promote the adoption of InFlux: SETLabs. Most of the original SETLabs employees came from the COM Factory, and Mr. Nagaraj and Mr. Thonse reported directly to the head of SETLabs. Soon thereafter, InFlux's popularity grew through word of mouth. Project managers placed numerous calls to SETLabs to get help with the methodology and request training. Fulfilling these requests demanded almost all of Mr. Nagaraj's and Mr. Thonse's time for several months. In short order, InFlux had become SETLabs' flagship contribution to Infosys.

In May 2000, Mr. Nagaraj and Mr. Thonse presented the InFlux methodology to the senior management team and asked for advice. The team encouraged them to continue developing and disseminating their methodology and even assigned quantitative targets for adopting InFlux within project teams. The senior management team also suggested Mr. Nagaraj and Mr. Thonse trademark their innovation, so they did. The two men even hired an external communications firm and formally branded their methodology "InFlux." Along with improvements to their methodology, they marketed new "releases" of InFlux with version numbers, mimicking common practice within the software industry.

The task of developing and disseminating InFlux was growing beyond what Mr. Nagaraj and Mr. Thonse could handle on their own. So the pair wrote a business plan with a budget for a bigger staff, detailing the program's activities, the number of people required and their areas of expertise, and the percentage of effort to be spent on further developing

InFlux versus training and consulting internally with project teams that requested it. Thereafter, Mr. Nagaraj and Mr. Thonse updated the plan annually, and it became one piece of the SETLabs budget that was approved by the senior management team.

Mr. Nagaraj and Mr. Thonse began building a team dedicated to “evangelizing” InFlux by offering consulting and education to Infosys project teams. They added an employee from within SETLabs plus several others from business units that were deeply familiar with the pitfalls of writing software specifications. They dubbed the team members “InFlux champions” and sent them to teams requesting help. Typically, InFlux champions worked with a project team for the first four to six weeks of a software development assignment, often in a ratio as low as one InFlux champion for every three team members.

Despite the additional assistance, by the end of 2000, Mr. Nagaraj and Mr. Thonse still spent most of their time pushing the use of InFlux and too little of their time further developing the methodology. They decided to try hiring graduates from MBA programs as InFlux champions, estimating that an MBA with a “consulting flair” could be an effective advocate for InFlux, able to communicate the benefits of InFlux to project leaders and to train project teams.

Mayank Gupta, hired in December 2000, was the first MBA graduate to join the InFlux team. He recalled the nature of the challenge:

> The biggest barrier was people who had followed existing practice for many years. People did not want to change. In contrast, it was very easy to influence a person who had little experience and was not ingrained in the existing practice.

Unfortunately, it was the experienced people, those most comfortable with existing practice, who most needed to be influenced. Experienced team leaders were very active in the design phase of projects because that phase was well known to be critical to the overall project outcome. They were accustomed to thinking first in terms of software design, not business problem-solving.

In some cases, barriers to using InFlux came from some clients who felt that writing software specifications should be their own primary responsibility or that writing specifications was not Infosys’s core competence or that their method for writing specifications was better. One InFlux champion felt strongly that the clients with whom he had worked were overlooking major opportunities that InFlux would have made plain and visible.

To encourage adoption, InFlux champions developed a few case studies that described in detail how project teams benefited from using the methodology. The case studies emphasized the value of engaging clients in business, not IT, terms and how doing so helped project managers build their ability to advise clients on a broader range of issues.

Gradually, the InFlux champions’ fieldwork began to pay off. As the number of teams that had completed projects using InFlux rose (typically, projects were several months long), the chorus of positive feedback from clients and project leaders grew louder. Mr. Thonse elaborated:

> The conversion rate of InFlux was high—that is, once people used it, they tended to use it again. The hard part was getting people to use it for the first time.

Still, they had a long way to go, and the company was growing so quickly that Mr. Nagaraj and Mr. Thonse hungered for a more efficient way to spread the use of InFlux within the company. Sending InFlux champions into the field was a labor-intensive approach. Making the job even bigger, the project managers who had "converted" to InFlux were getting promoted out of project management roles.

Mr. Nagaraj and Mr. Thonse decided to convert the methodology into a branded-software application. At a nuts-and-bolts level, the software application they visualized would accelerate the process of rendering diagrams of process designs. Until then, this was a manually intensive process in Microsoft PowerPoint or some similar broad-based application. To the extent possible, the software would automate the writing of specific software requirements from those diagrams; conceptually, the application would capture and disseminate the intelligence of the InFlux champions. In their choice to build a software application, Mr. Nagaraj and Mr. Thonse were encouraged by a new SETLabs initiative to develop identifiable pieces of intellectual property and the fact that turning InFlux into a software toolkit would facilitate patenting it.

Infosys tapped five-year Infosys veteran Naveen Bakshi to head development of the InFlux toolkit in October 2001. Mr. Bakshi developed a prototype within just three months, and the InFlux champions piloted the new toolkit with field teams. There was some feedback on the usability and interface of the toolkit and many ideas for new functionality. Critical feedback focused on the methodology itself; some still questioned its value. The toolkit team continued their refinements until the late 2002 company-wide launch of the application, dubbed InFlux workbench. (See Exhibit 1 for sample screens from InFlux workbench application, showing how the toolkit could be used to model the processes within Infosys's internal travel-booking service.)

In conjunction with the launch, Mr. Nagaraj and Mr. Thonse restructured the team's activities, limiting the practice of embedding InFlux champions in project teams for several weeks at a time. Instead, they conducted road shows—visiting all Infosys's software development centers, providing routine support for users of the tool, and reviewing project results.

The publicity and top management visibility at the time of the launch generated a spike in interest. There was a surge in the number of InFlux workbench downloads from the company's intranet. The downloads, however, did not immediately convert to full use by product teams. It took time for someone unfamiliar with InFlux workbench to become comfortable using the software in front of a client. Users could get started with InFlux simply by reading the user manual and attending one to three days of training, but without hands-on field experience, they lacked confidence. Project managers often felt that the first few weeks of a project were very sensitive, especially with a new client. You did not want to stumble or fall behind, and it was an awkward time to schedule training.

Furthermore, InFlux workbench was not comprehensive. Designing large corporate information systems was complex, and the software could not conceivably address all possible issues. One InFlux champion elaborated:

> There were many issues that we knew were not covered by InFlux. As InFlux champions, we know when you need to just work around the software. But early users might not understand the need to work around the software. They might just move ahead by following the instructions. Other users might shy away from InFlux completely when work-arounds were required.

Despite these issues, the InFlux team continued to see a rise in the number of service requests it received from project teams. Absence of requests from the field had brought some SETLabs projects to a halt, but there was more than sufficient pull for InFlux to keep its resource base growing. The InFlux software development team added more programmers and improved InFlux workbench following feedback from users. The feedback was rich, and InFlux workbench over time became more user friendly, more functional, and more internally consistent. The team even launched a networked multiuser version of the software.

Infosys's top managers increased their effort to accelerate the adoption of InFlux in September 2003. One of Infosys's original seven founders, K. Dinesh, explained:

> As usual, an innovation is not just about the product; it is also about the change that you need to create within an organization.

As one of the activities under the initiative, the InFlux team worked with Infosys's QA department to research the feasibility of making InFlux a must-use tool in a project's design phase. Some project teams strongly opposed the idea because they felt either that InFlux was not directly applicable to their projects or that their own homegrown methodology was better. Even some InFlux team members opposed compulsory use. Mr. Bakshi elaborated:

> We believed people would use it if there was value. If you forced people to use it, it would backfire. People would find a lot of reasons to resist, and they would find a way to meet the requirement—to check off a box on a checklist—without any real intent to derive value from the tool.

Officially, InFlux remained "recommended" but it was not "required" for project teams. Nonetheless, project leaders could expect to be asked during project reviews whether they used the tool, and some perceived one needed a very good explanation if one did not use it.

Infosys's management team sought to ground all decisions in hard data, but solid quantitative proof of InFlux's value proved elusive. The process of collaborating with clients to write software requirements was, perhaps, the least measurable and least manageable aspect of Infosys's work, so quantifying the improvement in productivity was difficult. With the QA department's help, the InFlux team made an assessment by comparing the case study of an InFlux project to a case study of a non-InFlux project.

Infosys also attempted to measure the adoption rate of InFlux by internal teams. Even that was tricky because InFlux workbench did not apply to every project and there was no black-and-white clarity about projects to which it did apply. The company could measure the number of employees aware of InFlux, the number of people trained, the number of project proposals in which InFlux was a deciding factor for the client, and the number of requests for help received by the InFlux team. All were trending positively. By 2006, the InFlux team had grown to 30 to be able to fulfill requests for help. There were also lots of qualitative success stories from project teams.

Some field teams remained skeptical, which was not unusual for new tools introduced to the Infosys workforce. Nonetheless, evidence convinced Infosys's top management team of InFlux's value. Managers also felt it was an important symbol to clients of Infosys's ability to engage at a business, as well as a technology, level. Mr. Bakshi described the support from the top:

> Clients spoke about InFlux as one of the company's differentiators frequently, both inside and outside company.

The InFlux team continued to develop new marketing approaches, including developing an internal certification program. The team also started trying to extend its reach outside the company. For example, the group partnered with Singapore Management University to launch an InFlux elective and team members published papers and presented at conferences. They even gave copies of InFlux workbench to some clients, some of whom began using it. The InFlux team also filed its first patent on the methodology in 2005.

Reflecting on what by 2006 was still a "big success but still much work to be done" story, Mr. Nagaraj and Mr. Thonse reflected that collecting success stories, hiring graduates from MBA programs, branding the idea, and converting the methodology to software were all important turning points. "We really had to learn the art of marketing," commented Mr. Thonse. Mr. Gupta added,

> If I were to do it differently, I might have built a bigger team of champions sooner, possibly embedding InFlux marketing teams permanently within business units.

Mr. Nagaraj elaborated:

> The effort to evangelize InFlux took us away from fully developing the methodology and the software. We should have separated the R&D team and actually discouraged it from evangelizing. That would have saved us a lot of time.

Rapid Development Toolkit: Automating Programming

The process of collaborating with clients to write specific software requirements was subject to errors and misunderstandings, but so was the process of writing code. In 2003, in one of Infosys's COEs, Satadal Bandyopadhyay endeavored to reduce the latter source of error by automating portions of the code-writing process.

Mr. Bandyopadhyay was a technical architect working in the enterprise solutions ECU. When clients hired Infosys to help customize and install packaged software applications from companies such as SAP and Oracle, the enterprise solutions ECU took a leadership role in the engagement. Although software vendors encouraged their clients to customize the software *only* within the finite set of parameters they had built into the software in advance, clients often believed they had some unique needs that the software vendors did not anticipate. Thus, Infosys's projects to implement packaged software generally involved some measure of writing new software modules and integrating them with the vendors' software.

While SAP and Oracle were the largest and best-known vendors of software for corporations, Infosys supported over 20 different vendors, including Yantra, a company that provided a powerful warehouse management system (WMS). A WMS was a key part of supply chain operations for any company that held significant inventory, and it provided a comprehensive view of inventory in any number of locations. Real-time inventory information enabled efficient decisions about ordering, restocking, shipping, and handling materials.

Of the 450 employees within Infosys's supply chain group, 200 worked on Yantra implementations. The basic steps of a packaged software implementation were similar to those of a custom software development effort. In the design phase, the team collaborated with the client, clarified business objectives, and wrote software specifications while mindful of the capabilities of the selected packaged software application. In the implementation phase, software programmers wrote custom software modules and installed them alongside the packaged software.

Mr. Bandyopadhyay, who had worked almost exclusively on WMS implementations during his tenure at Infosys, saw problems similar to those faced by the InFlux designers—misunderstandings between Infosys and the client about software specifications, as well as within the Infosys team between people with a business orientation and people with a technology orientation.

However, because Mr. Bandyopadhyay's focus was limited to a specific class of business problems—the implementation of Yantra's WMS—he saw the possibility of building a tool that went beyond InFlux's functionality. He had seen a great deal of logic and software code duplicated from one client to the next and he believed Infosys could actually automate the production (and reuse) of substantial chunks of software code. He visualized a tool that could translate flowcharts of the client's warehouse supply chain into at least a skeleton software code, with some portions fully programmed. Such a tool, he felt, could markedly reduce both error rates and the time required to complete a project. It could also help Infosys demonstrate an increasingly important capability to meet a client need: rapid reconfiguration of software as business conditions changed.

Mr. Bandyopadhyay's role as head of Yantra's COE positioned him to push the idea forward. He was assisted by a group of project managers, technical architects, and other senior members of the Yantra practice. Because Yantra was a relatively small practice, its COE had no full-time employees. Everyone involved in it carried client responsibilities as well; they contributed to the COE project in their spare time and between projects. Clients always came first. Although the Yantra COE had a backlog of dozens of promising ideas, it could sustain only two or three projects at a time, with four or five people on each project team.

Mr. Bandyopadhyay conferred with software architects from within the Yantra practice about the concept. Over a cup of coffee, he also shared his idea with the project manager he reported to, Chandradeep Bandyopadhyay (not related), who immediately saw the technical feasibility of the idea and encouraged him to pursue it.

With renewed confidence rooted in this validation from other experts, Mr. S. Bandyopadhyay spent about six weeks working with a colleague to develop a proof of concept—a few components of the toolkit he visualized. He subsequently tested it during a live Yantra implementation, and the results were promising. Just as he was ready to take the next step and build the full toolkit, however, emerging client work took priority. The project was shelved for six months. Despite the disruption, he was convinced of the toolkit's benefits and planned to resume development as soon as possible.

In January 2004, while briefly between projects, Mr. C. Bandyopadhyay began reconsidering the toolkit, even as Mr. S. Bandyopadhyay remained fully engaged in his own client work. Even though Mr. C. Bandyopadhyay was reenergized by the possibilities, he was once again quickly immersed in a new client project, this time managing a team initially staffed at 35 on a year-long project. The new project was scheduled tightly, and Mr. C. Bandyopadhyay reflected that it was exactly the type of situation in which the toolkit would be valuable.

Unwilling to shelve the project any longer, Mr. C. Bandyopadhyay decided to take a risk. Estimating that he could use the client team to build the toolkit and *still* meet project expectations, he decided to divert five developers to toolkit development. As project manager, Mr. C. Bandyopadhyay was authorized to take such a risk but was also fully accountable for the outcome. If the toolkit effort failed, the team would have to revert to writing code from scratch. In that scenario, Mr. C. Bandyopadhyay would need to overstaff the project to stay on schedule, would have little chance of finishing on budget, and would have to explain why so little custom code had been written for the client early in the project. Nonetheless, the idea looked like a winner, and the client relationship was longstanding and strong. Mr. C. Bandyopadhyay was not worried.

The toolkit development team started building the tool one piece at a time, asking business-oriented project colleagues to test each piece as they went and incorporating their feedback. The team completed the first full prototype of the toolkit in just one month.

The benefit of using the toolkit was apparent soon after. Like InFlux, the toolkit proved an able communications tool—and more. Nontechnical consultants could draw flowcharts, and some of the programming was done automatically from there. There were tremendous productivity gains. In initial plans, Mr. C. Bandyopadhyay projected he would need 65 people at peak staffing for the project. Because the toolkit reduced the amount of code that needed to be written from scratch, the project never required more than 40 people, including those working on the toolkit.

The toolkit, in fact, paid for itself during its first project. Interestingly, however, because it was a project that billed time and materials rather than a fixed fee, it was the client who reaped the early benefits of the innovation. Thus, the toolkit, while improving productivity, *reduced* Infosys's revenues from the project. Nonetheless, Mr. C. Bandyopadhyay believed it was wise to take a longer-term view, one that incorporated the value of error reduction.

> If we had 65 people working on that project writing original code, we may have had to support the project for several months, at no charge to the client, to eliminate bugs. It is better to have 40 people working on the project using the toolkit and be done with it.

Mr. C. Bandyopadhyay was pleased with the toolkit's capability and wanted to deploy it across all Yantra projects. Before a full deployment, however, he wanted to ensure the toolkit was thoroughly reviewed, tested, and freed of bugs. In spring 2005, he decided to tap into the COE's resources and spoke with Mr. S. Bandyopadhyay, who subsequently developed a test plan for the toolkit and allocated bench resources—several programmers—to the process.

As the testing got under way, Mr. C. Bandyopadhyay was anxious about disruptions. He did not want to see testers pulled away for billable work.

> The challenge with these initiatives is that a guy can be on the bench today but tomorrow might be needed for a project. So whenever you plan something like this, you need to keep in mind that at any point you could lose your people.

Fortunately, the people who were assigned to test the toolkit did not get pulled away from the project. They completed testing the toolkit within roughly one month.

The Yantra COE formally released the toolkit, dubbed the Rapid Design Toolkit (RDT), in June 2005. Coincidentally, the launch of the RDT coincided with a best-idea competition hosted inside the enterprise solutions ECU. The competition was initiated and run by a group called InVEST, which, like the Yantra COE, was a virtual organization run by a part-time anchor. By 2004, InVEST's second year, the contest was drawing submissions from nearly 100 teams, with an average size of five people. (The enterprise solutions ECU employed about 4,000 that year.)

InVEST actually hosted four contests each year, seeking innovative ideas in growth, innovation, solutions, and thought leadership; competition for each track was held at different times of the year. Anyone in ES could form a team and submit ideas for the competition. Winners were selected by a panel of experienced leaders in ES and subsequently presented their ideas to the Infosys Management Committee, the highest operational body within Infosys, chaired by COO Kris Gopalakrishnan. The committee awarded funding to the best ideas.

One of InVEST's objectives in running these contests was to protect Infosys's intellectual property through patenting winning ideas. Another was to help the ECU identify young employees with high potential. The winners garnered recognition and attention to their project, which could jump-start the development of their ideas.

Mr. C. Bandyopadhyay and Mr. S. Bandyopadhyay submitted the RDT to InVEST under the "innovation" track. With an idea that had fully blossomed into a valuable toolkit, they handily beat their competition, taking home T-shirts and certificates. More importantly, the competition helped establish the RDT's credibility and spread the word about the toolkit throughout the enterprise solutions ECU.

By then, the RDT was already being deployed by three or four project teams, out of 8 to 10 Yantra projects that were in progress at any one time. Following recognition from InVEST, Mr. C. Bandyopadhyay and Mr. S. Bandyopadhyay pushed further use of the RDT. They incorporated it into the standard training programs and discussed using it at each project kickoff. Naturally, there was some resistance. Objections were not rooted in complexity; the instructions for the RDT were only 15 pages long. In Mr. C. Bandyopadhyay's view, resistance stemmed mostly from "not invented here" attitudes. Nonetheless, once people used the RDT, they saw how much easier it was than programming in JavaScript.

Mr. C. Bandyopadhyay also had a role in evaluating and auditing Yantra projects. In periodic project-review sessions, he had the opportunity to probe the project team's methodology and question the way the RDT was used or why it was not used. He also ensured that questions about the RDT were included in audit checklists. And any time he heard of a project that was not using the RDT, he organized conference calls to discuss why. As he explained,

> When we heard that they wouldn't be using the RDT, we had a call organized. We did not bring in the technical architect for the call. The technical guys sometimes resisted input from other technical architects because they felt they had a vested interest in pushing their own technology. So we talked with project managers. I explained the benefit of using the toolkit in my own projects. When we did that, people began to open to the idea. We asked all the consultants in that team to start using it, to try out the stuff to see how it works.

Mr. C. Bandyopadhyay also persuaded the Yantra practice head, perhaps the only person with more Yantra experience than he had, to join the evangelization effort. He explained,

> Once the people with the most experience were convinced, well, it was difficult to counter that.

To continue maintaining and improving the RDT, the Yantra leaders assigned one of the original developers to be the RDT "anchor." The anchor, with the help of a (part-time) team of nearly 20 others with defined project roles, kept track of reported problems about the RDT, pulled people from the bench to fix them, added new features, and maintained a long-term product road map.

The RDT was stored in an open-source code platform—SourceForge, a software development website that provided free web hosting to projects. Infosys employees could use and contribute software code to projects such as the RDT through the website. In fact, all Infosys employees were expected to contribute to the company's intellectual property development, and this was one possible avenue for doing so. Project anchors then reviewed submissions and selected the ones to incorporate. Anybody could download the RDT and subsequently modify it, but only designated developers who were very familiar with the toolkit were allowed to incorporate changes to the official code base.

Mr. S. Bandyopadhyay felt the open-source environment was powerful. It allowed Yantra consultants around the world to contribute to the maintenance and enhancement of the RDT. One open-source contribution was language enhancement for a Japanese client.

By 2006, Mr. C. Bandyopadhyay was focused on documenting the use of the RDT in PRidE and was working with Infosys's intellectual property lawyers to investigate the possibility of patenting the RDT. (It was tricky because Infosys could not infringe on Yantra's intellectual property rights.) By then, the RDT had become very popular within the supply chain practice. According to Mr. C. Bandyopadhyay,

> The RDT has become so widespread now that people on any new projects will ask for it. It is like a way of life. Many people can only design using the toolkit.

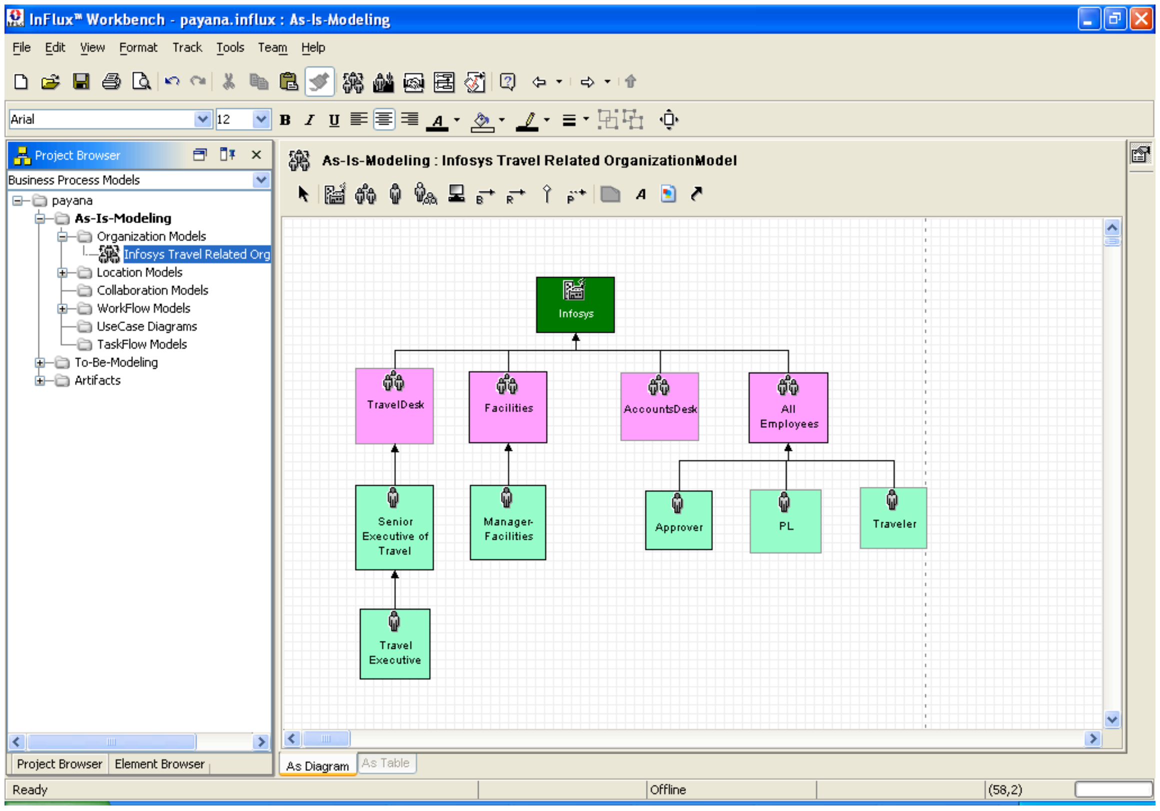
There was not 100 percent adoption, however. Mr. S. Bandyopadhyay observed that it took two to three projects before teams figured out how to maximize use of the RDT, and Yantra projects were lengthy. So, even in 2006, it was early in the adoption cycle. He said,

> People are still learning and trying. We should not force people into it. If you do that, new innovation will not happen.

Mr. C. Bandyopadhyay was pleased in particular that with the RDT in his back pocket, he could make more competitive bids for new business. He could price aggressively and he knew his team could deliver. The RDT not only was improving productivity, it was driving new growth as well.

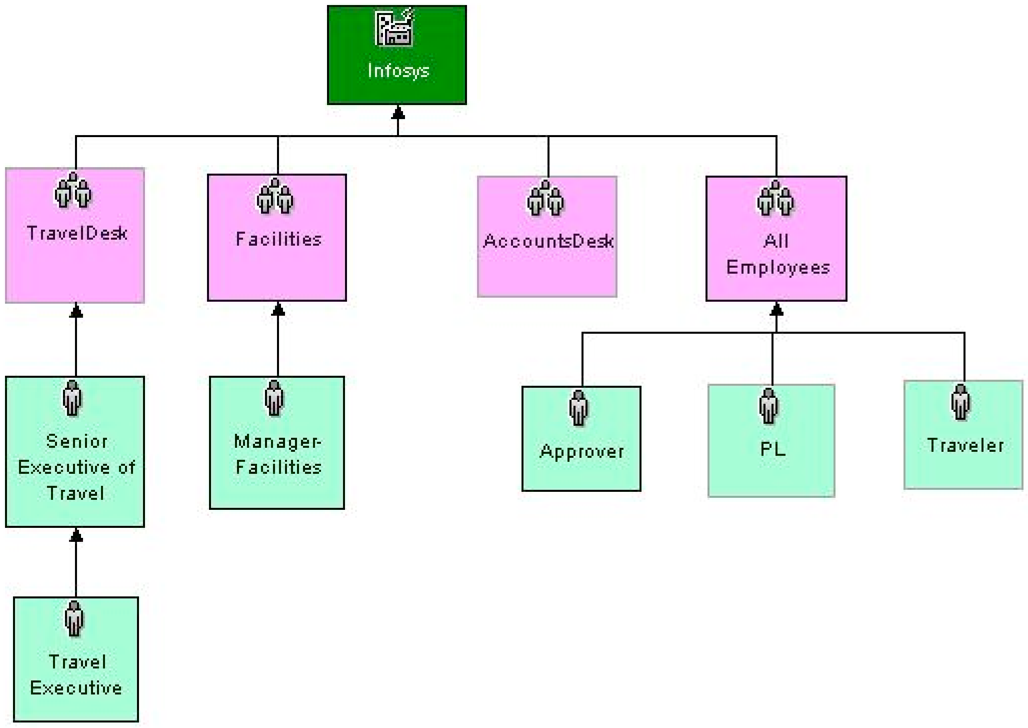
Exhibit 1: Screens from *InFlux Workbench*

Overall View of Application



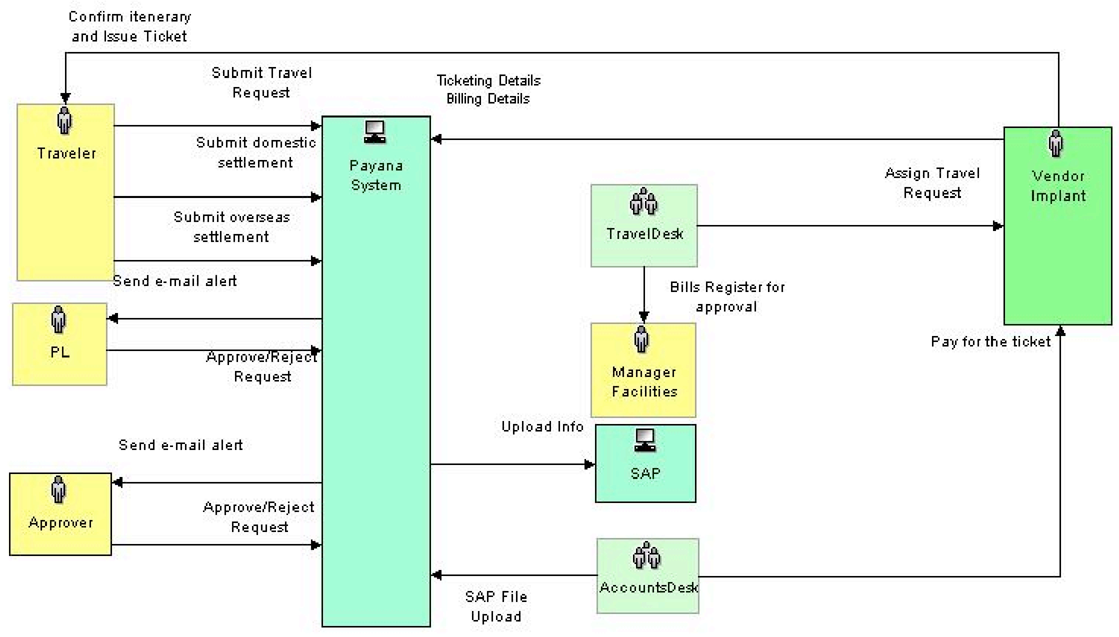
Sample Organization Model

The figure below illustrates the organization model of Infosys's travel-booking service. It shows the organization of the department and people, as well as the reporting structure.



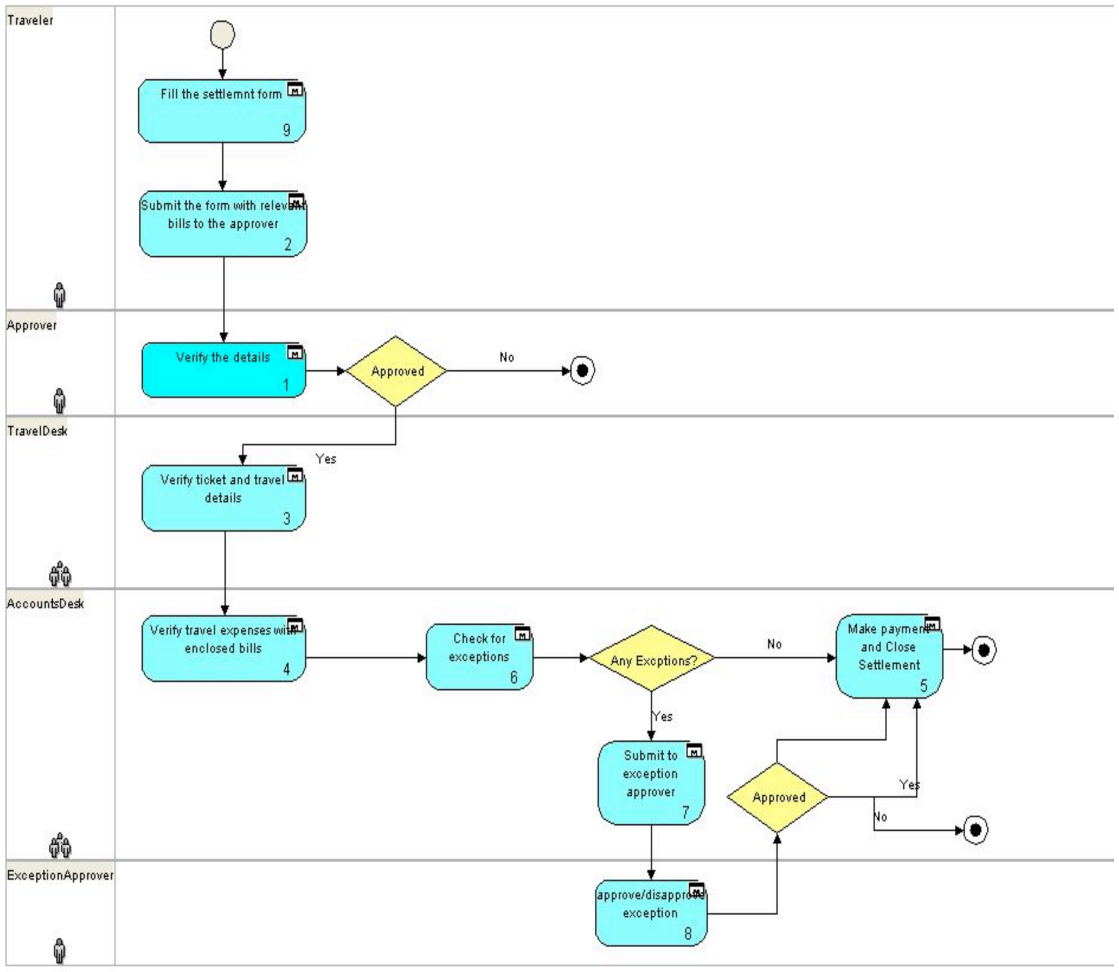
Sample Collaboration Model

The following diagram shows the business collaboration model for Infosys's travel-booking service.



Sample of Workflow Model

Following is a workflow diagram of Infosys's travel-booking service.


Traveler
Fill the settlemnt form 9
Submit the form with relevant bills to the approver 2
Approver
Verify the details 1
Approved
No
Yes
TravelDesk
Verify ticket and travel details 3
AccountsDesk
Verify travel expenses with enclosed bills 4
Check for exceptions 6
Any Exceptions?
No
Make payment and Close Settlement 5
Yes
Submit to exception approver 7
Approved
Yes
No
ExceptionApprover
approve/disapprove exception 8