An Execution Approach for Large-Scale SOA Technology Migration

A pragmatic planning and execution model can effectively modernize integration technology across the enterprise.

Executive Summary
Large-scale migration programs are among the toughest to plan and execute for any enterprise, especially when the migration is only about technology and no new functionality is delivered to the business that is funding the program. One such case is that of technology migration to a service-oriented architecture, or SOA. Such programs have a simple vision — retirement of a legacy integration technology through controlled migration of services to the new standard before support ends for the existing technology.

In any such large-scale technology migration program, one of the most critical elements of design is the unit of migration. Determining how these migration units are designed and sequenced for execution during a long-running transformation program is of paramount importance from both the portfolio management and enterprise architecture points of view.

The approach presented in this white paper is based on the principles of staged lifecycle, iterative delivery, multicriteria decision-making and retrospection. Although the principles, techniques and tactics recommended are applicable to all legacy technology migration programs, we have chosen a real-life migration case in order to provide a realistic viewpoint. The context is that of a large program to migrate the integration endpoints of core banking services from an end-of-life integration technology, CORBA (Common Object Request Broker Architecture), to currently mainstream technology (SOAP/HTTP Web services).

After introducing the context elements, we describe the key elements of a program execution model that is based on the concept of stages and that leverages a factory approach to migration during certain stages. The migration unit optimization approach is based on a concept of migration cycles and multiple influencing factors, such as application dependency and budget availability, organizational priority and others. The recommended approach is applicable to many migration programs that are executed across today’s large enterprises.

Migration Context
IT organizations at large enterprises the world over have continually needed to retire costly legacy technology infrastructures. In such situations, IT organizations typically select among mainstream and future-oriented technology to which they can migrate. To achieve this type of migration, a centrally managed program is often designed with funding support from both appli-
cation development and CTO organizations. Such programs impact multiple stakeholders, and therefore, the coordination effort is typically quite large. In addition, these programs have a lifetime of four to five years; this extended time horizon presents problems such as a lack of long-term visibility, risk of strategy change midway, change in organization priorities (and thus funding), etc.

Given this situation, enterprises need a holistic, top-down approach to program management and planning. Technology typically is the least of the migration pains, because if the project is scoped properly, software vendor partners provide necessary automated tools to facilitate the technology upgrade.

We propose an approach to migration program execution and its internal optimization, based on our experience with migration of the core banking services of a global bank from an end-of-life integration technology (CORBA) to current mainstream technology (SOAP/HTTP Web services). The previous landscape consisted of core banking systems mostly implemented on mainframe systems and front-end channel applications implemented on contemporary technologies – Java and .NET. From an SOA perspective, more than 80% of service providers offer mainframe-based applications, while the rest are delivered via Java technology.

These services were provided to consumers via common middleware built on a service bus topology (see Figure 1). The bank had already established mechanisms for SOA governance and middleware integration that were expected to be applied during the migration program. In terms of program scope, there were more than 1,000 providers that offered 2,500 business services, consumed by about 400 consumer applications through the common middleware mentioned earlier.

Migration Elements

Before outlining the approach for migration, it is important to understand the key elements involved in migrating integration technology. Each of these elements will be part of the migration units (defined in the next section) that are executed during the program. The following fundamental elements must be properly considered while planning the migration of integration technologies:

- **Service and service interface:** A service in SOA is the logical, self-contained business function offered by the provider of the service for the consumer of the service and is described by a well-defined functional/data delivery contract between these two parties. The service interface is the primary manifestation of the service, describing both functional (service signature) and quality of service (execution and invocation policies) aspects of the offered service. While in the case of CORBA the interface is described in the form of IDL and NS configurations, the Web services technology makes use of WSDL, XSD and associated WS-artifacts to describe the same.

- **Service provider (provider application/component):** These applications are the ones that host the actual functionality/data being served through the service interface provided. The service interface lifecycle is primarily in

---

**Integration Technology Migration Context**

![Diagram](image-url)
control of these applications, and the interfaces are evolved based on demand from consumers or due to a change in underlying functions or technology infrastructure.

- **Service consumer (consumer application/component):** These applications host the modules that invoke and consume the services of the third-party provider applications. Such applications are the clients of the providers and thus have a strong influence on the evolution of the service interfaces. As reuse is one of the fundamental reasons why a service orientation is applied, there is generally a many-to-many relationship between the consumer and provider.

- **Service implementation point:** An implementation point is that part of the provider application that is bound to the service endpoint when invoked by the consumer applications. The implementation program/component/class is typically the entry point to the functional/data access logic that the service provides. From a migration point of view, all such implementation points must be considered individually, because post-migration, the endpoints in the new technology also need to be bound to the implementation.

- **Service call point:** A call point is that part of the code in the consumer application from where the provider service interface is invoked. A consumer application may have multiple call points for a given interface. For the purpose of migration, each call point is considered as a separate element to be migrated.

- **Service repository:** As the name signifies, the repository is essentially the hosted catalog of all services offered by the provider applications and all associated information therein. The users of the repository search for services in the catalogs, and if consumers use these provided services, they register themselves as consumers at this central location. Please note that the repository here doesn’t provide runtime lookup and resolution of services, which is the job of typical service registries.

As the first executable step in the migration program, the services that were so far available in CORBA need to be modeled as per the standards of Web service technology. Subsequently, the code artifacts of newly created Web services need to be generated or implemented (as appropriate) for integration with the provider and consumer applications. Once the integration is complete – and both providers and consumers have been tested to their satisfaction – the existing services (on legacy technology) need to be decommissioned as per the retirement process defined under the SOA governance standards of the organization. Figure 2 depicts these steps as a continuous cycle of workstreams in a migration program.

The service repository has a critical role to play in this model, as it is the only place where the old and new services must be modeled and managed by the service designers of provider applications. All existing and new consumer applications refer only to the repository for services they wish to consume. In IT organizations with mature SOA practices and governance, the service repository also acts as the design time and change time governance platform. Therefore, the modeling, planning and lifecycle management tools (critical in migration projects) are developed in close proximity to the repository itself. The lifecycle and its actual elaboration as a migration execution model are described later in this paper.

**Migration Units**

In the context of SOA technology transformation, a migration unit is defined as a logical unit of work composed of the migration elements that must be migrated as a group for dependency,

**Iterative Workstream Model for SOA Migration**

![Figure 2](image-url)
efficiency and organizational reasons. These units are designed and sequenced after taking multiple factors into account. Each migration cycle (described later) executes a set of migration units planned for the duration of the cycle. For simplicity and easier reference, we will use a shorter name — LUM (logical unit of migration) — throughout the rest of this paper.

An example of such a migration unit would be a LUM containing one consumer application with 15 call points that are consuming 15 service interfaces provided by six different service provider applications. Such a configuration would typically be called a consumer-driven migration unit, which is described later.

Migration Lifecycle Model
As legacy technology migration programs have a typical span of four to five years in large enterprise IT landscapes, it is critical to adopt an approach based on an agile philosophy of delivering in multiple iterations instead of a big-bang, four-year waterfall planning approach. That said, such a long timespan requires provisions for defined milestones that can be achieved in an incremental manner. The model that follows is based on this consideration.

Overall Execution Model
The execution model recommended for the migration program is based on the notion of stages, where each stage has different requirements in terms of milestone delivery and thus requires its own delivery model.

The three stages depicted in the model represent the progression in the lifecycle of a service interface to be migrated. While Stage 1 lends itself to a one-time delivery style, Stage 2 is more suited for following a factory-like approach to migrate the interfaces in migration cycles. Stage 3, on the other hand, involves on-demand delivery of interface decommissioning. Stage 1 is a prerequisite for starting the actual migration, and Stage 2 is where the actual migration of applications to the new interface is carried out. As soon as all applications dependent on migrated interfaces move over and accept the new interface, the old one is retired in Stage 3.

Migration Lifecycle Model Guiding the Execution Approach

---

Figure 3
In addition to the stages that mandate specific delivery models of their own, the entire process of technology transformation goes through four workstreams (in line with what is depicted in Figure 2):

- Migration planning.
- Service interface remodeling (design).
- Provider integration and consumer migration.
- Decommissioning (legacy retirement).

Figure 3 outlines the steps that are followed in each of these workstreams and the constraints that apply while the workstream is being delivered. At the end of each workstream, a governance mechanism in the form of a quality gate is recommended. This quality gate should certify completion, as per the objectives and acceptance criteria specified for the deliverables of the workstream.

Three foundational elements are critical to all large migration programs: tracking and reporting tools, governance and operating mechanisms and, most importantly, a robust and comprehensive testing facility. Such a facility must act as the quality signoff authority throughout the migration program. A detailed approach for establishing and operating these foundations is beyond the scope of this paper and, therefore, is not described here.

**Migration Cycles**

Properly defined planning and governance units, as well as the criteria leading to design and sequencing of the LUM, are two of the key factors that drive success of the entire migration program. For the design and sequencing of the migration units, an important prerequisite is to establish the set of criteria that guides this activity. It is highly recommended that this criteria is established during the early part of the migration planning phase. During assessment, a migration sequencing index is computed using these criteria. The units are then sequenced accordingly.

As the duration of the entire program is typically too large to apply this mechanism, we recommend assessing and planning the migration in cycles. Similar to an agile notion of iterations, a migration cycle is a time-bound iteration, and the entire migration program is a set of cycles. In each migration cycle, a set of LUMs is planned and migrated. The application of cycles is an internal operating mechanism for the migration program, and it does not interfere with the concept of a logical grouping of applications in LUMs; instead, it serves as a complement.

In order to balance duration and size constraints, we recommend a cycle duration similar to the budgeting and planning cycles of the organization if the program scope and participants are fairly stable and known (which is true in most legacy migration cases).

### Illustrative Migration Cycles for Program Execution

![Illustrative Migration Cycles for Program Execution](image-url)

*Figure 4*
During migration planning, the LUMs are designed by applying the approach recommended in this paper and evaluating applications for the defined criteria. There are two options in terms of the scope for which the design activity is carried out:

- **Design LUMs for the entire scope** (difficult due to time horizon; not recommended).
- **Allocate LUMs to cycles** (after evaluation and discussion with relevant stakeholders) and then sequence LUMs on a cycle-by-cycle basis.

The latter approach is more sensitive to changes that might need to be incorporated during the program lifecycle based on the changing business and IT scenarios. Figure 4 details this approach with a scenario of a financial institution that has a yearly budgeting cycle. Considering the migration scope presented earlier and the size of a sample migration unit, a cycle duration of 18 months can be proposed in which the following three execution phases will be carried out:

- **Migration planning and budgeting of migration cycle** (about six months). This can be performed in parallel with the execution of the previous cycle.
- **Migration execution for the identified and planned LUMs** (12 months).
- **Migration retrospection** (two to four weeks toward the end of the migration cycle).

The reason for proposing a 12-month execution phase is based on the possibility of around 80 LUMs to be migrated in a cycle, with each LUM migrating a combination of one consumer and up to five provider applications (consumer-driven LUM). Given the scope of migration, approximately five such migration cycles will be executed during the entire program.

The migration cycles are not sequential in nature. During the last six months of the migration cycle, the migration planning of the next cycle can be started so that all the necessary preparation, planning, dependency resolution, syndication and coordination is taken care of before the next migration execution starts. This migration cycle planning (about six months) is a recommended activity before the actual migration within the cycle (expected to span a 12-month period). Before the migration of the defined cycle has been completed, a retrospective is also recommended to assess execution and apply necessary adaptations to the execution approach of the next cycle being planned.

### Design and Sequencing Optimization

The design of the migration units – and the order in which these are planned to be released – is an area that is impacted by a number of factors. While on the one hand we have the migration elements and their interdependencies, on the other we have to take care of the contextual elements, such as budget, resources and organizational priorities, among others. What follows is one approach for optimally designing and ordering (or “sequencing”) the logical units of a migration program.

### Migration Unit Design

Due to a preference for demand-led models, the organizational intent is most often to design LUMs driven by a single consumer application. While we agree that this approach is suitable considering a demand-driven model of resource allocation and execution, certain LUMs can also be designed in a provider-driven way. Generally, there is also a good probability that the LUMs might be composite in nature and of manageable size and complexity, meaning we may have to split the larger LUMs into smaller ones.

A key metric in LUM design that must be examined is inter-application dependency in the form of coupling. Both afferent and efferent coupling needs to be considered.

- **Afferent coupling (Ca):** Applicable to provider applications, this metric indicates the number of consumer applications that depend upon interfaces of the provider application. It is an

### Parallel Migration Cycle Planning and Execution

![Parallel Migration Cycle Planning and Execution](image)

Figure 5
indicator of the responsibility of the provider application.

• **Efferent coupling (Ce):** Applicable to consumer applications, this metric indicates the number of provider applications upon which the call points of the consumer application are dependent. This is an indicator of the independence of the consumer application.

If a provider has a high value of Ca, a provider-driven LUM will be a preferred approach. A lower Ca indicates a provider with fewer consumers dependent on it and thus can be considered for a migration unit driven by one of these consumers. On the other hand, if a consumer has a higher Ce, it indicates that the consumer is dependent on many provider applications, and thus a consumer-driven LUM model should be followed. If the Ce value is low for the application, it might be efficient to migrate it as part of a provider-driven LUM.

It is not feasible to cover large applications in one LUM due to the sheer size of call points and interfaces offered, respectively. In such cases, it’s recommended to use a hybrid approach to LUM creation.

• In the case of a consumer-driven LUM, if the Ce is too large to manage (see below), separate LUMs must be designed by splitting the main LUM into sub-LUMs, each of which is driven by one of the modular units of the consumer (e.g., an accounts module in an online banking application) and all providers on which the client list component is dependent.

• In the case of a provider-driven LUM, if the Ca is too large to manage (see below), separate LUMs must be designed by splitting the main LUM into sub-LUMs, each of which is driven by one consumer and contains the interfaces of the provider on which the consumer is dependent.

The optimal size that should be considered for the LUMs is a single driving application with less than five dependencies. This means that for a consumer-driven LUM, there should be no more than five provider applications. When the number is higher for a LUM, splitting should be considered (as recommended above). That said, the complexity of these interfaces should also be looked into as an important factor while deciding such a split.

**Migration Unit Sequencing**

As demonstrated in the previous section, the dependency metrics Ca and Ce are important indicators of whether a consumer- or provider-driven approach should be adopted. That said, this is not the only parameter that will determine the final decision on LUM design. Other aspects, such as business criticality, budget availability, lifecycle alignment, resource availability, etc., need to be considered to arrive at a final LUM design decision.

In order to evaluate all such parameters, a simple weighted rating method (described below) can be applied. Another option is to use a formal technique such as an analytic hierarchy process (AHP), but that would require availability of more data and information about migration candidates, which might be difficult. However, if such data is available, the AHP technique can help inform an optimal decision about design and sequencing.

The following is a comprehensive list of parameters that can typically be considered in evaluating such cases of integration interface migrations. As this list is fairly large, the two dimensions of importance and ease of data availability can generally be applied to identify the effective set of criteria. In this example, we have rated the criteria based on the assumption that the migration needs to be carried out with a global services provider for the landscape described earlier in the paper.

As depicted in Figure 6, and based on our analysis and rating in the context of global migration programs, we recommend that IT organizations include the first five parameters in the initial criteria for design and sequencing. All of these are important, and the data to use these criteria should be available and able to be extracted with minimum effort. This means the first set of criteria for the migration sequencing index entails:

• **Application dependencies:** Available from software analysis and application repository tools.

• **Budget and resource availability:** Available from program management groups.

• **Business criticality of applications and interfaces:** Available from program management groups.

• **Availability of the application and interfaces in the development and test environments:** Available from infrastructure teams.

• **Availability of required test data:** Available from infrastructure and testing teams.

In some cases, sequencing cannot be decided upon even after applying these criteria. When this
## Representative Migration Evaluation Criteria

<table>
<thead>
<tr>
<th>Number</th>
<th>Criteria</th>
<th>Applicability</th>
<th>Description</th>
<th>Importance</th>
<th>Ease of Data Availability</th>
<th>Parameter Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Dependencies</td>
<td>Applications</td>
<td>See Ca &amp; Ce discussion, page 7 (implicitly depends on number of interfaces and call points)</td>
<td>Y</td>
<td>Y</td>
<td>YY</td>
</tr>
<tr>
<td>2</td>
<td>Budget and Resource</td>
<td>Applications</td>
<td>Availability of budget and IT resources to migrate</td>
<td>Y</td>
<td>Y</td>
<td>YY</td>
</tr>
<tr>
<td>3</td>
<td>Business Criticality</td>
<td>Applications and interfaces</td>
<td>The assigned business criticality as per the organization</td>
<td>Y</td>
<td>Y</td>
<td>YY</td>
</tr>
<tr>
<td>4</td>
<td>Availability in Development and Test Environments</td>
<td>Applications and interfaces</td>
<td>Whether the elements in LUM are available in development and test environment for migration</td>
<td>Y</td>
<td>Y</td>
<td>YY</td>
</tr>
<tr>
<td>5</td>
<td>Test Data Availability</td>
<td>Applications and interfaces</td>
<td>Availability of the test data in the target development and test environment for the interfaces being migrated</td>
<td>Y</td>
<td>Y</td>
<td>YY</td>
</tr>
<tr>
<td>6</td>
<td>Release Lifecycle</td>
<td>Applications</td>
<td>Suitability (for migration) of the release lifecycle of applications</td>
<td>N</td>
<td>Y</td>
<td>NY</td>
</tr>
<tr>
<td>7</td>
<td>SLA Sensitivity of Interfaces</td>
<td>Interfaces</td>
<td>SLA sensitivity, coupled with the current distance of the actual performance from expected SLA limits</td>
<td>N</td>
<td>N</td>
<td>NN</td>
</tr>
<tr>
<td>8</td>
<td>Interface Complexity</td>
<td>Interfaces</td>
<td>Complexity of interface contract and data types used in the contract</td>
<td>Y</td>
<td>N</td>
<td>YN</td>
</tr>
<tr>
<td>9</td>
<td>Implementation Coupling</td>
<td>All (call points, interfaces, applications)</td>
<td>How much coupling exists between the integration endpoints and functional logic of the applications/service implementation or invocation unit</td>
<td>N</td>
<td>N</td>
<td>NN</td>
</tr>
<tr>
<td>10</td>
<td>Reengineering Effort</td>
<td>Applications</td>
<td>Existence of reengineering efforts within the provider or consumer applications</td>
<td>N</td>
<td>N</td>
<td>NN</td>
</tr>
<tr>
<td>11</td>
<td>Special Needs</td>
<td>Applications</td>
<td>Special needs in terms of hosting, testing, technology or functions</td>
<td>N</td>
<td>N</td>
<td>NN</td>
</tr>
<tr>
<td>12</td>
<td>Technology Platform Release Alignment</td>
<td>Applications</td>
<td>Impact of the technology platform release on the applications to be migrated</td>
<td>Y</td>
<td>N</td>
<td>YN</td>
</tr>
<tr>
<td>13</td>
<td>Migration State of Dependencies</td>
<td>Applications</td>
<td>Migration state of the participating applications—especially useful in plan adaptations</td>
<td>N</td>
<td>N</td>
<td>NN</td>
</tr>
<tr>
<td>14</td>
<td>Domain Spread</td>
<td>Applications</td>
<td>Domain spread of the consumer and provider applications</td>
<td>N</td>
<td>Y</td>
<td>NY</td>
</tr>
<tr>
<td>15</td>
<td>Change Cycle Suitability</td>
<td>Applications</td>
<td>Change cycle of the applications. Those with rapid change cycles are suitable to early migration if other factors such as business criticality are suitable</td>
<td>N</td>
<td>N</td>
<td>NN</td>
</tr>
<tr>
<td>16</td>
<td>Outstanding Defects</td>
<td>Applications</td>
<td>Although the migration is technical and no functional changes are needed, the long defect list adds to the risk</td>
<td>N</td>
<td>N</td>
<td>NN</td>
</tr>
<tr>
<td>17</td>
<td>Documentation Quality</td>
<td>Applications</td>
<td>Quality of application documentation, as it will impact the efficiency of the integration</td>
<td>N</td>
<td>N</td>
<td>NN</td>
</tr>
</tbody>
</table>

Figure 6
happens, other parameters can be utilized from the list based on data availability and importance in the context of the LUM being considered. For the first level of decision-making, the five parameters referenced above should be sufficient.

For a given duration (cycle/program), the sequencing of the LUMs needs to be carried out along with the design of the LUM. For this, the same criteria-based decision-making can also be applied to arrive at a score, called a migration sequencing index. The index represents the relative rating of the LUMs with respect to the risks and suitability of the LUM for migration. Figure 7 represents the calculation model for the migration sequencing index in detail.

As depicted, it is a simple weighted rating method for determining the suitability of a unit for migration. The model is based on the maturity analysis model that we use widely for application migration planning. The basic model has been adapted to make it more suitable in the context of SOA integration technology migration. Two critical elements of the model are the criteria in each category (and the previously referenced indicative list) and the weight that should be applied based on the priority of the IT organization planning the migration. This calculation is carried out for all the LUMs in the scope of planning, and the result is an ordered list of LUMs to be developed for the migration program.

**Key Recommendations**

In addition to the approach referenced above, the following are among the key recommendations for migration design and sequencing that would be useful for large SOA technology transformation programs such as the one described earlier.

- **Medium risk levels** are good candidates for grouping in initial set for migration. Risks contributing to medium maturity are capable of being mitigated through various fine-tuning activities. Can have longer knowledge harvesting time than high-maturity applications.

- **Lowest risk levels** have the highest potential to be grouped in the first set for migration. Application risk characteristics are low complexity, low business criticality, higher level of documentation, etc.

Figure 7 represents the calculation model for the migration sequencing index in detail.

---

**Determining Planned Migration Sequencing of LUMs**

- **Provider Applications**
  - High Maturity Units
  - Medium Maturity Units
  - Low Maturity Units

- **Consumer Applications**
  - High Maturity Units
  - Medium Maturity Units
  - Low Maturity Units

- **Interfaces and Call Points**
  - High Maturity Units
  - Medium Maturity Units
  - Low Maturity Units

- **Organizational Elements**
  - High Maturity Units
  - Medium Maturity Units
  - Low Maturity Units

* This maturity profile is an indicator of risk involved, but the sequencing decision is a combination of various factors, including business priority, maintainability, etc.
one hand, it helps bring early confidence and stability to the core set of services. On the other, it paves the way for a comparatively simpler migration unit design because these services are used across many consumer applications.

- It is critical to align the migration cycles with the technical platform releases planned by the organization’s technology infrastructure teams. This alignment can help achieve faster time to market and better quality of deliverables.

- During migration analysis, indirect dependencies on the technology to be migrated should also be identified jointly with technical platform or framework provider organizations. In some cases, the technical platforms offer components that are consuming the services being migrated and thus, indirectly, affect the applications consuming those platform components.

There might be cases where an application must be considered to be part of more than one LUM due to dependencies on or of other applications in those LUMs. In such cases, this application should be migrated as part of the first LUM being moved in that set. This will mean that during migration of the remaining LUMs, the application can simply be consumed, as it is already migrated.

**Conclusion**

As SOA technology evolves and product vendors start to phase out older technologies and middleware, enterprises will be forced to carry out such migration cycles every eight to 10 years (in the case of very mature technologies). Additionally, with the growth in adoption of SOA-style architectures in the enterprise IT space, there will always be a substantial number of technology services being retired. These two aspects mandate a managed approach to migration of SOA integration technology. We believe that the ideas presented in this paper provide the necessary guidance to carry out these migrations in a manner that maximizes service coverage and execution efficiency. It also helps minimize the risks associated with long duration planning and a tightly coupled application landscape.

While we have presented the model and evaluation framework for migration planning and execution, the criteria and parameters that are applied can very well vary across organizations. The maturity of an organization’s IT landscape, IT processes, SOA adoption and governance controls are some of the environmental factors that must be carefully evaluated in the context of the migration program being carried out. As mentioned from the start, although the context in the paper is that of CORBA to Web services migration programs, the principles, techniques and tactics can be leveraged for all legacy technology migrations.

**About the Author**

*Dinkar Gupta* is an Associate Director and Principal Architect in Cognizant’s Banking and Financial Services (BFS) Technology Consulting Group, where he heads the strategy, architecture and technology team assisting a global BFS client. He has a post-graduate degree in computer science and applications and has 14 years of software development experience across multiple industry segments. He is also an IBM-certified solution designer for Rational Software Architect and is actively engaged in architecture transformation consulting and solution delivery for BFS customers. Dinkar can be reached at Dinkar.Gupta@cognizant.com.

**Acknowledgments**

The author would like to acknowledge the contributions of Madhura Kulkarni, Senior Manager of Projects; Partha Basu, Manager of Projects; Diptendu Mittra, Senior Manager of Projects; Tushar Wagh, Architect; and Rishish Kumar, Manager of Projects within Cognizant’s BFS Business Unit.
About Cognizant

Cognizant (NASDAQ: CTSH) is a leading provider of information technology, consulting, and business process outsourcing services, dedicated to helping the world’s leading companies build stronger businesses. Headquartered in Teaneck, New Jersey (U.S.), Cognizant combines a passion for client satisfaction, technology innovation, deep industry and business process expertise, and a global, collaborative workforce that embodies the future of work. With over 50 delivery centers worldwide and approximately 145,200 employees as of June 30, 2012, Cognizant is a member of the NASDAQ-100, the S&P 500, the Forbes Global 2000, and the Fortune 500 and is ranked among the top performing and fastest growing companies in the world. Visit us online at www.cognizant.com or follow us on Twitter: Cognizant.