



# Application Modernization



**Ranganaathan V.V.**  
Advanced Architect  
Enterprise Architecture and Integration

## Table of Contents

<b>1. Summary</b>	<b>3</b>
<b>2. Legacy Architecture</b>	<b>3</b>
<b>3. Need for Application Transformation</b>	<b>4</b>
<b>4. Transformation Strategies</b>	<b>4</b>
<b>5. Application replacement (Total rewrite with web based internet technologies)</b>	<b>4</b>
<b>6. Application Modernization Strategies</b>	<b>5</b>
<b>7. MphasiS Offerings</b>	<b>6</b>
<b>8. MphasiS Value Proposition</b>	<b>6</b>
<b>9. Case Study</b>	<b>7</b>
<b>10. Steps</b>	<b>7</b>
<b>11. Approach Options</b>	<b>7</b>
<b>12. Technology Stack</b>	<b>8</b>
<b>13. Approach 1 – Using code conversion tool</b>	<b>8</b>
<b>14. Approach 2 - Using SOA, custom framework, code conversion tool</b>	<b>8</b>
<b>15. Approach 3 – Rewrite the Entire Application</b>	<b>9</b>
<b>16. Benefits Comparison</b>	<b>10</b>
<b>17. Conclusion</b>	<b>10</b>

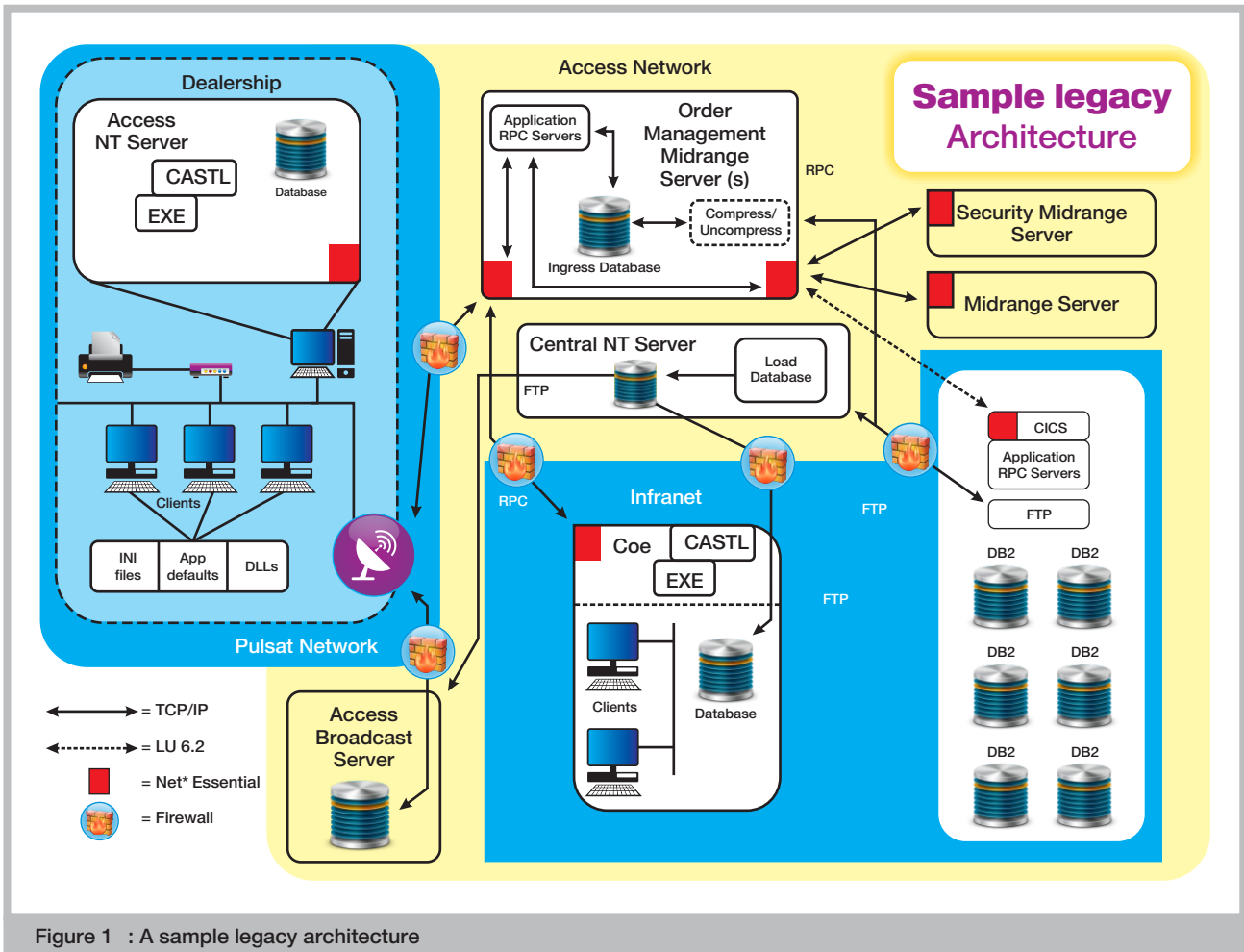
### List of Figures

<b>1. Figure 1: A sample legacy architecture</b>	<b>3</b>
<b>2. Figure 2: Legacy architecture – layered view</b>	<b>5</b>
<b>3. Figure 3: Target architecture using SOA</b>	<b>5</b>
<b>4. Figure 4: Custom framework flow</b>	<b>9</b>

# Summary

Reengineering of software is described as the examination and alteration of a system to reconstitute in a new form. The approach is to renovate and extend the current application into a new technology to best support the needs of the current business. Application modernization should be achieved by leveraging the existing investment in application infrastructure and reposition the product advantageously for the future. The challenge on hand is to convert legacy applications to web applications by reengineering legacy components to re-usable components. The resulting web application can be easily integrated with web technologies.

# Legacy Architecture



A typical legacy suite of applications will be based on three tier architecture: User Interface (UI) layer, business layer, and database layer. Often the older applications are mainframe-resident and written in third-generation languages (3GLs) such as COBOL, PL/I and RPG or any one of a long list of obsolete 4GLs. The suite might comprise of client server applications, mainframe applications, C or C++ programs, Pro\*C routines as part of midrange servers and one or more 3rd party applications typically used for mainframe integration with the application and for database interaction. The database might be Ingres, DB2 or any other equivalent. The application UI will be either in Visual Basic or Visual C++ or Oracle forms or in any other 4GL. The UI layer is restricted to have only cosmetic validation code. They are also sewn with calls to aged data and transaction systems such as DL/I, Datacom/DB, VSAM and CICS.

The business layer is made of C, C++, Pro\*C routines. All business validations will reside in this layer. A database connection will be established in this layer. All the data access will happen with the help of a gateway application; either custom written or through third party provided tools. All SQL queries will be embedded as part of the business layer. The business layer will interact with the mainframe business components through which it will have access to a DB2 database.

# Need for Application Transformation

Applications are the lifeblood of any enterprise and are at the heart of delivering innovation for businesses and governments. The task of finding young skilled engineers to maintain legacy code bases is becoming increasingly tough. The experienced engineers who understand how such systems work in a business environment are reaching their retirement age, forcing the companies to upgrade or continue without vital system maintenance capabilities. Also today, the ideal requirement for any application is to be mobile, connected, interactive and immediate. Any application with the absence of this feature will negatively impact the ability to deliver new services that drive competitive advantage. The decision is no longer whether to transform aging applications and infrastructure or not, but rather how to realize such a transformation.

## Transformation Strategies

Application transformation can follow either of the following strategies:

- Application replacement or total rewrite
- Application modernization

## Application Replacement (Total rewrite with web-based Internet technologies)

Migration of legacy systems for web enabling using web technologies such as Java EE or Microsoft .NET is a growing need of the business. An increasing number of legacy systems can now be migrated by taking advantage of Java EE scalability, speed, reliability, security and cross-platform capabilities along with SOA enablement using web services and Web 2.0 technologies using AJAX capabilities.

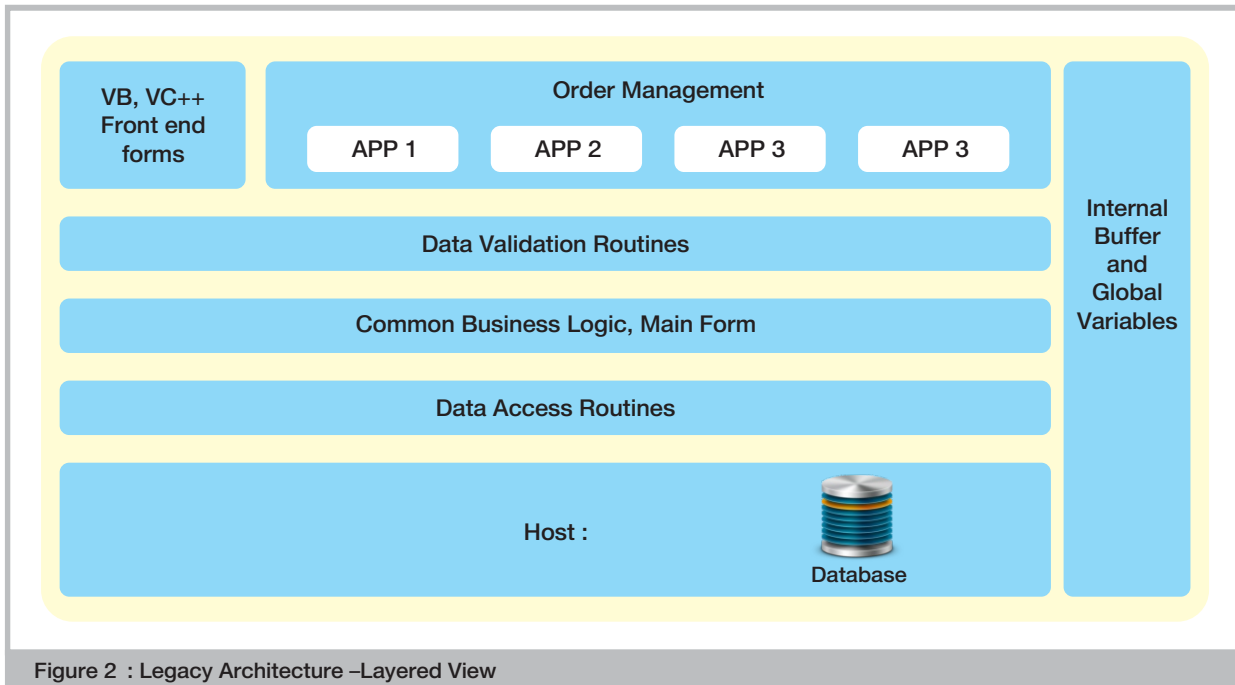
Following are the major business or technology drivers for migration and reengineering of legacy applications to the Java EE platform:

- **Platform Independence:** Java platform applications can run on a variety of hardware and operating systems including Windows, UNIX, and mainframe systems.
- **Java EE Platform Enterprise Application Support:** Java EE standards provide support, standards, tools, and a framework for user interfacing, data binding, communication enabling, web enabling, application configuration, application packaging and ensuring security which constitute the critical requirements for any enterprise application architecture technology.
- **Speed in Application Development:** The component based model enables enterprise software to be developed more rapidly with high reliability
- **Simplifying Application Deployment:** Application servers such as WebSphere, WebLogic, and JBoss provide inbuilt application deployment support. Improved IDE's such as Eclipse, Sun Studio Builder along with open source scripting technologies such as Ant provide improved application deployment support.
- **Enterprise Application Integration -** Prior to the advent of web services, Enterprise Application Integration was very difficult due to differences in programming languages and middleware used within organizations. The interoperability was cumbersome and painful. Today, any application can be integrated easily with web services and SOA technologies.

In reality, many organizations are finding it difficult to replace their antiquated but critical applications because of the huge transformation cost. Since applications are so critical, their wholesale replacement would be either too risky or too expensive and maybe both. Any organization that has plans to do legacy transformation has to do a review of spending priorities and an assessment analysis. If the assessment points to the criticality, wholesale replacement will be too risky for the business. Also, if the cost involved for transformation is going to be too high, organization might find it difficult to fund for the modern equivalent. But if application replacement is a lost cause, there is only one alternative called application modernization. The use of code conversion tools along with a custom framework will ease the cost, pain, and risk of leaving behind the old code base or platform.

# Application Modernization Strategies

A legacy application modernization strategy for small, medium, and large enterprises depends on two different approaches. Many companies are looking at a Service Oriented Architecture (SOA) approach to create distributed applications to help them modernize both their legacy application and to make their composite application more stable giving their business greater agility. Some companies however are looking to simply get rid of legacy applications on mainframe, i5/OS and OS/400, and UNIX servers and avoid legacy problems by moving to web based Internet technologies.



The problem of inaccessibility of the legacy application over the web can be addressed through a web application. A web application provides secure and platform-independent browser access to the legacy application.

The transformed web enabled legacy application with GUI screens can be attached into existing web infrastructures, such as LDAP security. The continued value of the legacy application is further leveraged. The new Java development can be seamlessly integrated with the legacy application, so all add-on capability for the legacy application could be created in the Java environment. The approach of achieving legacy application in stages, by first web enabling them, followed by incremental transition to Java, is the one that greatly reduces the cost before benefit gap and the incremental transition will start yielding ROI<sup>2</sup>.

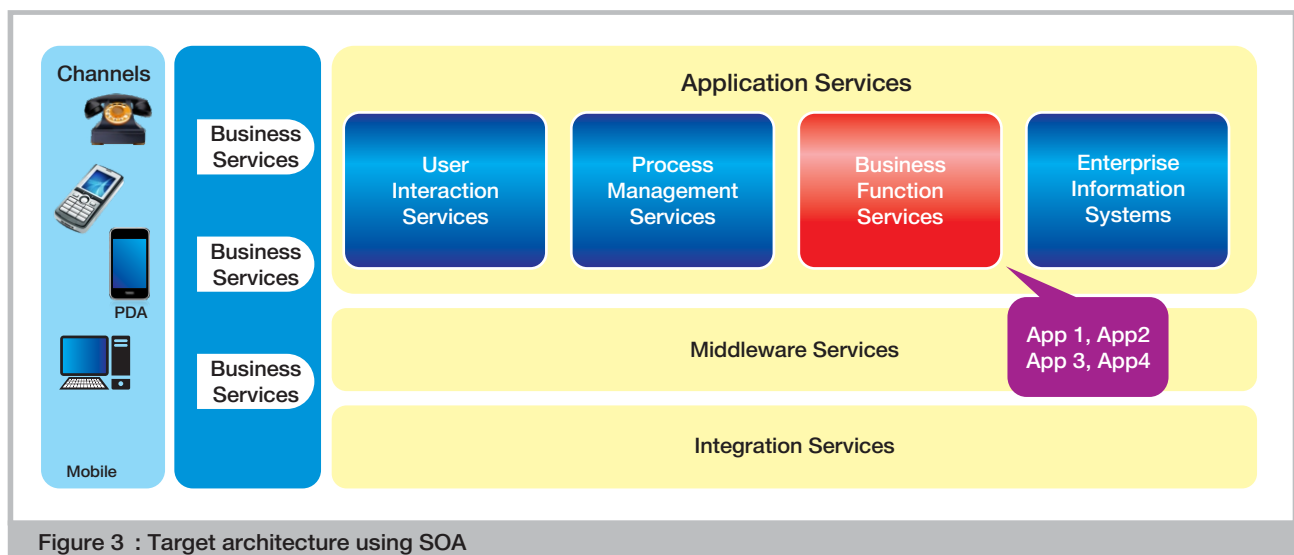


Figure 3 : Target architecture using SOA

Modernization involves either automatic code conversion or migration and SOA adoption. Automatic code conversion does not perform end-to-end conversion of the application; it requires manual intervention from engineers with a clear understanding of how an application and its related data flow work. An external organization often lacks the necessary experience of the application itself and how it relates to the business, prohibiting the outsourcing of either conversion of code or maintenance of code. One solution to the above problem is by adopting the SOA approach, by wrapping the key parts of the application using web services, leaving their logic intact and opening up links to modern front ends and to other applications. The newly created component is then given a WSDL interface, and a SOAP framework is used to build the component into an XML schema.

## MphasiS Offerings

MphasiS migration offerings can be broadly classified under Consulting, Governance, Execution and Sustenance. MphasiS service offerings can be listed as follows:

- Migration Assessment – Assessment of existing application which includes business criticality and cost involved, defining migration strategy, and migration roadmap identification
- Application Upgrade – Upgrading the OS, language, and database versions
- Application Reengineering – Re-architecture, Re-development, web enabling, implementation, and application sustenance
- Application Portability – Transferring the application to a new operating system and/or database. Also porting applications from old application server platform to latest application server platform
- Technology Migration – Legacy applications (Visual Basic, Power Builder, C++/VC++ running in mainframe environment) to multi-tier applications based on J2EE or .Net. This includes OS and database migration as well
- Data Migration – Schema translation, data migration, and cleansing

## MphasiS Value Proposition

- Proven and well defined methodology in migration/reengineering
- Usage of custom framework reduces application development time
- Strong resource base with migration tools and process expertise
- Expertise in O/S, database migration to higher versions or different O/S, and database
- Usage of prior migration experience and learning in migration/reengineering
- Decreased effort coming from a tool based approach reduces the cost of migration
- A detailed rollout strategy reduces application downtime
- Increased predictability due to a repeatable process

# Case Study

This case study is about Web-based Internet technology migration using tools, custom framework, SOA approach, and re-writing from scratch.

## Steps

### Analysis

Analyze the existing application, learn the relevant business process and functionality, understand the technology requirement and risk involved, select the reengineering tools, identify candidates for SOA adoption, and select a suitable framework. Apply a step-by-step implementation process to smoothen the transition to the new system through constant monitoring and project management.

### Aim

- Migrate a legacy application to a web environment using an optimized approach
- Retain the existing business functionality 'as-is'
- Explore usage of any tool for migrating the client side VC++ code to Java/J2EE or identify business services be wrapped using web services
- The source O/S is Solaris 6.x and target O/S will be Solaris 8.x

In this case study, we assume that the legacy database is to be retained as this database is already shared across various legacy applications. Modernizing the legacy database will be taken as the last exercise when all legacy applications are migrated to the web environment.

### Modernization Strategy

- Reengineering of legacy systems into reusable components, which can be easily integrated with web technologies
- Accessing legacy transactions from outside the mainframe and integrating them with online web based solutions
- Wrapping legacy business functions with web services
- Understanding the technology requirements and risks
- Employing messaging and integration brokers to link legacy and web-based architectures
- Accessing legacy databases for use in web-based applications

## Approach Options

Approach	Description
Approach 1	Look for a complete code conversion tool to migrate from source to target application
Approach 2	Using custom built framework based on Java/J2EE and client side validation functions from common repository, SOA adoption, and partial usage of code conversion tool to migrate from source to target application
Approach 3	Total code rewrite from scratch

# Technology Stack

Old stack: VC++, ANSI C, Sybase middleware, Db2

New stack: JDK 1.5.0\_12, Oracle WebLogic server 10.x, JSF 1.2, J2EE 1.4, ICEFaces 1.8.2, Db2

## Approach 1 - Using Code Conversion Tool

### Aim

- To involve an application SME of the source application to understand the business
- To identify a code converter tool and use the trial version to do sample conversion

The tool used is C++ to Java Converter. The approach is to use off-the-shelf migration tools.

### Observation

- The current C++ code does not have a specific pattern
- Tool does not have features to map C++ code to Java framework and develop HTML screen
- The resultant is one single Java program (Servlet) which has UI, validation, and data access layer in a single program
- It is observed that the business validation and data access layer were more reusable (60%) than other layers

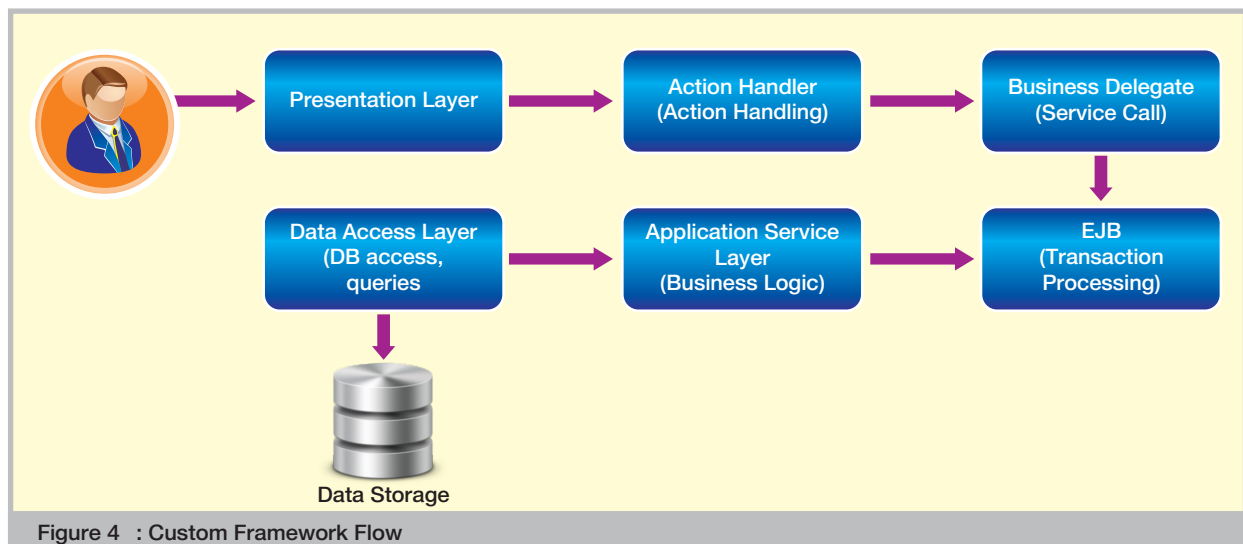
### Result

- Complete automation using a tool is not advisable. Even business validation and data access conversion is only 60% achievable
- Inputs from various forums suggest possible automation is less than 30%. It is not uniform, reliable, and requires manual intervention
- Effort to fill the missing portion will almost be equal to full code re-write

## Approach 2 - Using SOA, Custom Framework, Code Conversion Tool

### Aim

- To involve an application SME of the source application to understand the business
- To leverage the custom Java/J2EE framework for developing the target application using the client side validation components from the reusable library
- To retain the existing legacy business functions and wrapping them using web services using the SOA architecture and by retaining the legacy database.
- To explore a third party JSF implementation to simulate the thick client feature of the source environment.



### Observation

- Availability of framework components helps move the business code quickly. Reuse of client side validations from repository
- HTML screens to be developed manually and converted to JSF
- Using SOA architecture and wrap the business functions and data access validations using web services and exposing them to be utilized by external applications

### Result

- There is a gain of approximately 25% in effort hours when compared to total re-write.

## Approach 3 – Rewrite the Entire Application

### Aim

- Involve an application SME of the source application to understand the business and translate the business understanding to use case, high level design
- Arrive at a custom framework to suit the design based on the business understanding
- Leverage the client side validation components from reusable library and retain the existing legacy database
- Understand and rewrite the custom 'C' routines and Java functions

### Observation

- Framework to be developed/customized to suit the business requirement is utilized for development
- Reuse client side validations from repository and HTML screens to be developed manually and converted to JSF

### Result

No dependency on tool, framework components. It will be a total rewrite based on a new architecture, new framework, exploring reusable legacy components to be wrapped as web service, consumes more time and effort compared to the service wrapping approach (approach 2, above). Also, additional time has to be spent on requirement gathering and for analysis phase.

# Benefits Comparison

Below chart is the benefit comparison chart for the three approaches discussed for various features:

Features	Approach 1 Using Tool	Approach 2 Custom Framework + Tool	Approach 3 Total re-write
Leverage Framework	No	Yes	Partial
Re-usable libraries	No	Yes	Yes
Tool based conversion	Yes	Partial	No
ScalableTransaction	No	Yes	Yes
Security	No	Yes	Yes
Interoperable	No	Yes	Yes
Framework compliance	No	Yes	Yes
Development time (Base 100%)	>100%	approximately 75%	100%

The phased approach to the SOA-based architecture coupled with usage of code conversion and custom framework allows for easier mitigation of risk associated with changes to an enterprise IT infrastructure. Approach 2 has more advantages when compared with other approaches as it uses the best custom framework, web service wrapper, and code conversion tool. It reuses the database and related objects which saves time and cost. Client side validation functions from the repository to be leveraged and Windows screen to HTML screen conversion to be done by web developer. Above all, the end user will not see any difference in user experience.

Well defined methodologies integrate the use of tools to automate re-engineering tasks such as transformation and testing, and manual process for migrating legacy business modules to web technologies which leads to faster delivery schedules and greater accuracy. MphasiS can help reincarnate legacy applications by either integrating legacy systems with new technologies or by porting applications to more stable Java Enterprise Edition (Java EE), which provides platform independence, flexibility, scalability, and performance.

## Conclusion

Migration will be an ever-existent part of the business process. Frameworks are increasing in size and complexity, and new functionality is being implemented on existing systems. Therefore, migration is an important factor to address the overall improvement and efficiency of a business. Often, a company realizes that supporting an old infrastructure is too expensive; however, drastic changes are risky. Splitting the enterprise into domains or application into manageable modules and introducing the concept of custom framework, web service wrappers mitigates the risk and allows migration to a newer infrastructure in well controlled phases and also help business in yielding quick ROI.

## **Author Profile**

### **Ranganaathan V.V.**

#### **Advanced Architect**

#### **Enterprise Architecture and Integration**

Ranganaathan V.V. is an experienced professional with more than 15 years of service in application development, product development, and solution provider. Working with MphasiS since May 2005, his primary responsibilities include architecting solution and providing response to large RFPs, support delivery teams with architecture and design solutions, evaluation of tools, architecture assessment, and submitting white papers and POC on latest technologies.

Prior to working with MphasiS, Ranganaathan has also worked with other application service providers and product development companies in Chennai Area. His experience includes developing and designing applications in several 3GL, 4GL, open systems, architecting Enterprise applications using Java, JEE, frameworks and Oracle database.



# ABOUT MPHASIS.

Mphasis is a \$1 billion global service provider, delivering technology based solutions to clients across the world. With currently over 41,000 people, Mphasis services clients in Banking and Capital Markets, Insurance, Manufacturing, Communications, Media & Entertainment, Healthcare & Life Sciences, Transportation & Logistics, Retail & Consumer Packaged Goods, Energy & Utilities, and Governments around the world. Our competency lies in our ability to offer integrated service offerings in Applications, Infrastructure Services, and Business Process Outsourcing. To know more about Mphasis, log on to [www.mphasis.com](http://www.mphasis.com)

For more information, contact: [sales@mphasis.com](mailto:sales@mphasis.com).

USA: 460 Park Avenue South, Suite #1101, New York, NY 10016, USA  
Tel.: +1 212 686 6655, Fax: +1 212 686 2422

UK: 88 Wood Street, London EC2V 7RS, UK  
Tel.: +44 20 85281000, Fax: +44 20 85281001

AUSTRALIA: 9 Norberry Terrace, 177-199 Pacific Hwy, North Sydney, 2060, Australia  
Tel.: +61 2 99542222, Fax: +61 2 99558112

INDIA: Bagmane Technology Park, Byrasandra Village, C.V. Raman Nagar, Bangalore 560 093, India  
Tel.: +91 80 4004 0404, Fax: +91 80 4004 9999



0711