.NET LEGACY SYSTEM
MODERNIZATION ROADMAP

The Exit of Legacy Systems

Suyati℠
dream. dare. do

INDEX

Suyati℠
dream. dare. do

WHITEPAPER

Introduction

An exit strategy of **legacy systems** is to be devised in order to **maximize** the possibilities with emerging technology.

IT has seen tremendous growth over the years, but many enterprises are unable to reap the full benefits of such advances, trapped as they are in inflexible legacy systems on which they have made huge physical and intellectual investments. In fact, Microsoft estimates that many organizations spend about 75% of their resources simply to maintain their existing outdated legacy systems. About 59% of IT leaders place modernization as their top software issue.

Most systems, including .NET, release updates that co-opt new technology and requirements up to a point. However, with the passage of time, such updates deliver diminishing returns and become inadequate to resolve the growing list of issues related to the system.

Some problems legacy systems face are:

- Complex architecture, which is difficult to deploy and maintain.
- Obsolete technology. The recent advancements of cloud, mobility and big data make most legacy applications seem even more obsolete.
- Difficulty to scale up to meet the demands of a business that has grown over the years.
- Inability to integrate with other modern apps and systems.

Most legacy systems face the bane of extensive manual interventions, mostly expensive and error prone interventions to handle exceptions.

Organizations would do well to consider modernizing their legacy systems to do away with such issues, and thereby reap the twin benefits of improved efficiency and better customer satisfaction.

The convenience of a .NET Framework

The .NET Framework offers a first class development experience for the new needs applications, but moving existing applications forward, to co-opt latest developments such as multi-screen experiences and cloud services can be challenging.

Most legacy systems using the .NET Framework have based their code on Windows servers such as Microsoft IIS, SQL Server, BizTalk, and others, and on other proprietary Microsoft technologies such as .NET remoting, ASP.NET, Windows Forms, Windows Presentation Foundation (WPF), Silverlight,and Windows Communication Foundation (WCF). Modernization entails migrating them to a cloud platform and developing new web-based apps based on HTML5, CSS, JavaScript, HTTP REST and more. Organizations would especially look to migrate their legacy .NET applications to the Azure platform, the feature-rich PaaS and IaaS platform.

It is relatively easy to convert legacy code. However, modernizing a legacy .NET system goes much beyond such code conversions. The best way to go about is in a structured and systematic way.Failure to do so may result in chaos which may subvert established systems altogether, rendering the potential gains from migration to a modern system a non-starter.

Step 1: Understand the Existing System

The first step towards modernizing the .NET legacy system is to gain thorough understanding of the existing environment, to get a complete picture of how people use the applications, the data they use, the functional interdependencies among various assets, and more. Documenting how workloads and application serves the business needs and customers, and the extent of shortcomings in such fronts offer a realistic view of how the existing legacy application serves business needs and what it lacks.

This lays down the basis for estimating the key requirements for the modernization plan, both in terms of technical inadequacies that may prevent the organization from performing optimally, and glitches or shortcomings in serving the customer better. It helps in mapping out the future environment effectively.

A side consideration at this stage is to commit to the required investment and resources upfront. Failure to do so may result in stalled modernization, where the organization would be even worse off.

*Migrating legacy **.NET** applications to the cloud requires understanding dependencies or services such as MySQL, email services or messaging, that the legacy application uses or calls.*

Azure provides a number of services in the cloud, but the dependencies existing in the legacy application may not line up with the ones provided by Azure. Migration is easy when there is an Azure equivalent service available for the required dependency. But when such a service is not available, the work through is to either re-architect the application to do away with the need for such existing services, or more realistically, set up an Azure Worker Role configured to run and manage the unaligned services, and then make the same available in the PaaS deployment.

Migrating legacy .NET applications to the cloud may also require refactoring the code. All cloud platforms, including Azure run a virtualized and shared platform, and hosting in such a platform requires standardized code base among all applications. If the legacy system is not written in a supportive code base, it will not work in Azure and would require a refactoring process.

Step 2: Plan

A good plan stems from having a clear understanding of the objectives and laying down a detailed blueprint, including the procedure and resource requirements to achieve such project goals.

It is not enough to list out detailed milestones for application source code and data conversion. A good plan also requires:

- Mapping all operational areas to address required standards and procedures for operating and administering the new target environment.
- Designing an infrastructure solution that addresses accessibility, reliability, and growth requirements.
- Deciding on the right mix of automated migration processes and adaptation to address differences between the legacy environment and the target platform.
- Devising strategies for controlled migrations, so that the business stays open for customers even as the new way of working gets implemented.

A plan to migrate the legacy .NET application to the Azure cloud platform first needs understanding on:

- Applications in the system fit for the Azure cloud platform without any changes.

- Applications that require changes to the current application code. If the existing .NET application is stateful, then migrating to the cloud becomes that much difficult. Stateful applications makes calls to the underlying file system when running, and stores the data it creates on the node it runs. Since the app holds data on one node, it will not be able to update to the new node, and as such scaling the application for the cloud becomes extremely difficult.

It is after this exercise that a detailed timeline with milestones for making changes in the application code, to enable full integration and optimization with Microsoft Azure, becomes viable. The exact timeline depends on how much code needs to be re-hosted versus transformed, and the extent of resources.

Step 3:
Implementation

In this phase, the changes that are required are effected, culminating in the installation and configuration of the new infrastructure. This phase also includes administrating training to users on how to handle the modernized systems.

The pre-requisites before starting the actual transformation are to:

- Backing-up all existing application code and data.
- Downloading and installing the required software, which may include Windows Azure SDK and Windows Azure Tools for Visual Studio 2010 if the project involves modernizing a legacy .NET to the Azure platform.
- Subscribing to the required services, such as a Windows Azure services account.

Success of the actual implementation would depend largely on:

- Tracking progress of milestones, and taking decisions as per the situation.
- Identifying and monitoring open issues and risks.
- Establishing reliable communications among system developers, legacy system operators, customers and end users.

For the most part, modifying the existing ASP.NET application to run on the Windows Azure platform entails a three-step process of

- Creating a new Windows Azure project to migrate code from the existing web applications
- Migrating the code to the new project and alter the same to work in the cloud rather than on a 'local' web server. Specifically, this entails moving configuration settings from web.config to the Azure RoleEnvironment, setting the number of instances of

the site to be running in the cloud and options for handling session state in the cloud. All this is made possible through Azure AppFabric.

- Deploying the newly cloud-enabled application to the cloud.

Third party automated solutions such as WebMAP make the process easy and simple. It especially helps in churning out new lines of native code without the developer having to rewrite for the new platform. WebMAP, for instance, separates and divides application code optimally between web clients and servers, making the resulting application multi-tier, multi-tenant, and web-enabled, ideal for mobile device clients and software as service models. Developers may also customize the tooling to fit preferred architecture, coding standards and custom component mappings. However, the effectiveness of such automated solutions is limited, and in any case it does not work to add new features that may be the whole purpose behind the modernization.

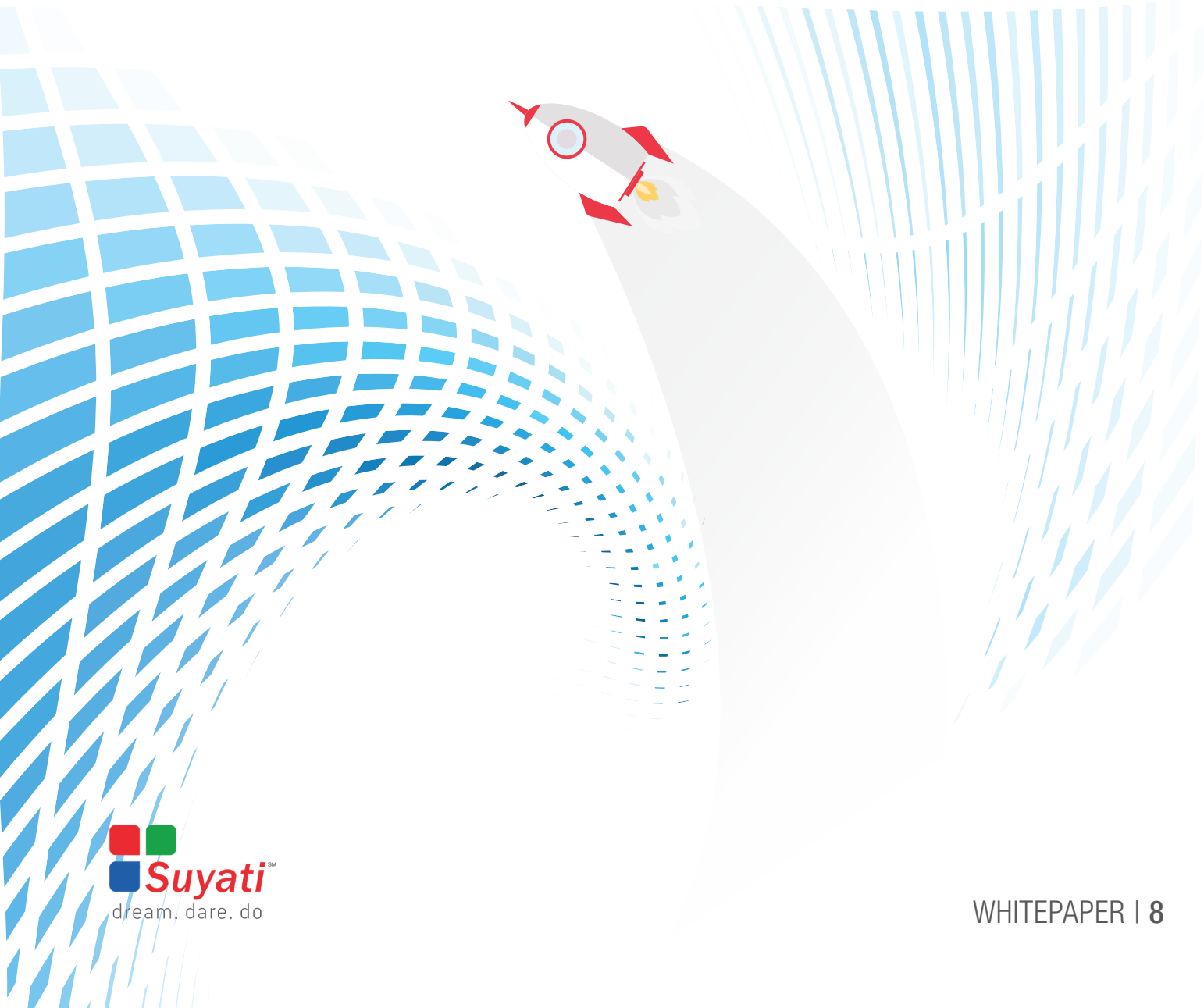
Many enterprises rush into this actual transformation, paying scant attention to understanding the legacy system first and planning effectively for the change. It is only when the above two considerations are done right that transformation becomes easy and chaos-free. Failure to have all resources in place, for instance, could mean a lengthy down-time for the business.

Step 4:
Post-Installation

It is important to test the changed system for accuracy and quality assurance. The parameters of testing may include factors such as performance, security, scalability and reliability.

Many enterprises skip this step. This is a mistake, for without this phase, any glitches may lie undiscovered, and could derail the legacy modernization effort at a later stage.

Legacy modernization is a complex affair and requires commitment. Since the specific requirements vary from enterprise to enterprise and depends on the nature of the legacy system, there is no one-size-fits-all solution. Modernization is more a journey than a project, but when done right, the results are worth the effort. It enables efficient, cost effective, responsive and competitive IT systems that power the enterprise to new heights.



SUYATI TECHNOLOGIES

Suyati is a young, upwardly mobile company focused on delivering niche IT services to support myriad Digital Engagement strategies. Our expertise also includes integration and delivery of CRM, CMS and Ecommerce solutions.

We're well versed in persona management, targeted content, and Ektron's Digital Experience Hub (DXH). In fact, we've built a series of marketing connectors that Ektron offers along with their world class CMS. Do get in touch with us to know more.

www.suyati.com services@suyati.com

Reference:

http://fedr8.com/blog/migrating-legacy-applications-microsoft-azure-3-key-considerations/
http://www.cio.com/article/2426296/enterprise-software/legacy-modernization-101.html
http://www.cio.com/article/2597926/application-management/anatomy-of-an-application-modernization-project.html
http://www.cio.com/article/2600187/application-management/expert-advice-are-you-ready-for-application-re-architecture.html
http://www.cio.com/blog/application-modernization-transform-for-the-future/
http://www.fcsltd.com/appdev/technologies/net_migrate_to_dotnet.asp
http://www.fujitsu.com/downloads/US/GND/case_Stanislaus-Fujitsu.pdf
http://www.pega.com/system/files/resources/Proven-Approaches-to-Legacy-Systems-Modernization-Dr-Khoshafian.pdf
http://www.developerfusion.com/article/119960/upgrade-your-aspnet-site-to-the-cloud/
http://www.mobilize.net/press/bid/196863/Mobilize-Net-Accelerates-App-Modernization-to-the-Web-via-WebMAP
http://channel9.msdn.com/Events/TechEd/NorthAmerica/2013/DEV-B218#fbid=?utm_source=linkedin&utm_medium=social&utm_content=354077
http://www.drdobbs.com/web-development/legacy-app-modernization-via-webmap/240142278
http://devproconnections.com/net-framework/modernize-net-software-development-environment
http://www.iterative-consulting.com/page/legacy_migration_white_paper.html

Suyati℠
dream. dare. do