

# Win in the flat world

## Next Generation Software Factories, using Microsoft Blueprints

– Mohammed Farrukh Nizami, Archana Sachin Ghag

*Today, markets are in a state of flux and business dynamics are changing rapidly as never before. Customers are keen for faster time to market and maximum ROI with minimum investment. For System Integrators, this means reduction in software development duration by increasing productivity and predictability, implementing best practices, as well as faster testing. Microsoft has been championing Software Factories to address the challenges of software development. This paper focuses on Blueprints, latest contextual guidance technology from Microsoft and how it helps build Software Factories.*



## Overview

### Introduction

Is the software industry mature enough to be called an industry in the first place?<sup>1</sup> Does it follow the same trends that are being witnessed in other industries? Are we able to build new products, with minor variations, by assembling and configuring existing components? Do we have structured and systematic reuse practices in place? An honest answer to all these questions is “No”.

If you take a serious look into the way software products are developed today, you will accept the fact that most of the highly talented developers end up writing repetitive code or doing menial tasks. Essentially, we are wasting the talent of highly skilled developers. We could, rather, embed their domain expertise and technical knowledge inside a set of reusable assets that can be stored in a repository. The concept of reusing various assets is not new, it has been there for more than three decades, but mostly it was ad-hoc and opportunistic. We should strive to implement a systematic and predictable reuse practice in place. When one says “reuse”, the first thing that comes to mind is “code” reuse and our answer is most likely based on this perception. To create objectivity, while explaining the concept of reuse, we should not think only in terms of tangible assets, but also in terms of intangible assets like knowledge, intelligence, domain or technical expertise, etc. We can bundle all of our intellect and expertise in the form of guidance, which sits in the realm of our development environment itself. This guidance can be in the form of documents, code generation tools, meta-models, domain specific languages, etc. In a nutshell, you can build guidance in any form that can be easily understood by the developer’s communities and guides them with a step-by-step procedure for implementing a particular process during software development lifecycle or develop a particular feature of a specific type of application. For example, one can build guidance on implementing agile methodology or test-driven development, applying the MVC pattern to a web application, or implementing Single Sign-On for enterprise applications. These reusable assets and contextual guidance form the core of the software factory approach. Jack Greenfield, Senior Director for the Platform Architecture Team in Microsoft’s Developer and Platform Evangelism Division defines a software factory<sup>2</sup> as:

*“... a software product line that configures extensible tools, processes, and content, using a [package of reusable assets], based on a [model of the solution] to automate the development and maintenance of variants of an archetypical product by adapting, assembling, and configuring framework-based components.”*

Microsoft as a software provider is committed to providing a platform for architects, developers and business users to help them author guidance and deploy it within the Microsoft Visual Studio IDE. In the following sections, we will talk about “Microsoft Blueprints” and see how it helps solve some of the problems discussed in the preceding paragraphs.

---

<sup>1</sup> <http://www.softwarefactories.com/>

<sup>2</sup> <http://msdn.microsoft.com/en-us/library/aa480032.aspx>

## Introducing Blueprints

Recently, Microsoft announced a new technology for authoring guidance called Blueprints<sup>3</sup>. Infosys has started assimilating this exciting and promising technology. Guidance Automation Toolkit (GAT)<sup>4</sup>, predecessor to Blueprints technology, was used to build guidance packages and make them available within Visual Studio.

The term Blueprint conjures up the image of an architect pulling out a big rolled up sheet of paper, having some criss-cross lines and curves drawn across it. Blueprint is a term often associated with the construction industry, which contains a diagram of the architectural design of an entire building or a part of it. But, in a broader sense, a Blueprint is nothing but a detailed plan. If you follow the individually well-defined steps listed in a blueprint, you can get to the finished product at the end. A software Blueprint focuses on a particular aspect of a software application. For example, you could have Blueprints that provide step-by-step guidance and code generation capability for Data Access, Application Security, Smart Client, etc. This essentially is what the new MS Blueprints platform is all about.

At the core of the Blueprints platform is the Blueprints Manager. Blueprints Manager is responsible for managing and updating all the Blueprints available on the system. It also manages the Blueprints Repository, which is a per user store for all of the versions of every Blueprint installed on a machine. Blueprints Manager also provides support for various runtime activities - like executing templates and workflows, interfacing with visual studio, etc.

In the following sections, we will explore the new features that are available with the Blueprints technology

## Blueprints - Advantages over GAT

### Support for Versioning

Versioning has always been a problem with the GAT-based software factories. Whenever there is a new version of a guidance package available, you need to uninstall the previous version to install the new version. However, with the blueprints you have an advantage here. One can update the installed blueprints through RSS feeds from within the Visual Studio IDE itself. So delivery of newer versions of software factories becomes as easy as downloading an RSS feed. Blueprints Manager checks for the updated versions of a particular blueprint on the internet, notifies you about the latest updates, and can additionally download and update the underlying Blueprint. Information about all the versions is kept in a repository which allows you to retrieve and access a specific version at any given point of time and update the Blueprints Manager. It is a very important feature considering the dynamics of the Software Development Lifecycle (SDLC). Any software application undergoes several iterations of changes and enhancements during its lifecycle. An application built using a particular blueprint may not be compatible with the future versions of that blueprint. So, any enhancement to the application in future using the latest release of the blueprint at that point of time will not be possible. But this feature allows you to access the earlier version of the blueprint that is compatible with this application and enhance the

---

<sup>3</sup> <http://msdn.microsoft.com/en-us/architecture/aa699360.aspx>

<sup>4</sup> <http://msdn.microsoft.com/en-us/library/bb880038.aspx>

application. The repository-based approach of deploying Blueprints becomes very important if your current version of the Blueprint is incompatible with this old application.

### Tight Integration with Windows Workflow Foundation<sup>5</sup>

Another big leap is the workflow-based guidance. In the current GAT-based software factories, guidance was provided in the form of documents. In guidance explorer, you can see a list of scripts or recipes that need to be invoked to build an application, using a software factory. But, this sort of guidance does not stop a user from executing a recipe out of order. For example, you could run a recipe to generate business entities from a database before even executing the recipe for creating a database connection and you end up getting an error message without even knowing what went wrong with the recipe. For an inexperienced developer, it can become really difficult to understand why a particular recipe is not working.

With the Blueprints technology, guidance can be built on top of the Windows Workflow Foundation, by creating a workflow for an entire development process. Individual activities might reflect the actual steps in the SDLC. For example, you can define a workflow that starts with capturing requirements. Unless you upload the business requirements document in the repository, the workflow won't allow you to move to the next step. You can build further intelligence, so that the workflow can even check if all the important sections in the document have been filled or not. Once this is done, the current activity will be marked as completed and you can move on to the next activity and so on and so forth. You can mark some of the workflow activities optional. This workflow can also be displayed to the user, with each activity showing a red light, a green light, or a check mark, based on its current state.

### Software Factory Authoring Made Easy

With GAT/GAX based software factories, it has always been a pain to use them within an existing solution. To integrate a guidance package with existing projects and solutions you need to do some manual settings in the project files which is way too tedious for a developer to achieve. Unless you know the internal details of these configuration settings, you will not be able to it. One great thing about blueprints is that it alleviates this limitation. You can unfold a blueprint in any existing solution. This brings up a very interesting proposition. You can have a Reference Implementation (RI) built using some latest technologies and best practices & patterns. Create a blueprint using this reference implementation as a template and start using it within this RI itself from which you have harvested these best practices. Essentially you can bootstrap your development process.

While using GAT/ GAX, you have to deal with a lot of xml to show wizards, to store data and to do any actions. But now, Blueprints has made it simpler by introducing the concept of commands to achieve the same result. This makes writing rich recipes a lot easier. Earlier, you had to implement a method called Execute, with no parameters and a Boolean return value, to execute a custom action from within a recipe. It resulted in the use of lots of global variables and hardcoded values inside the Execute method, which made code reuse very difficult. Now with Blueprints, writing custom activities (Commands) is as simple as writing a standard C# (or VB) method.

---

<sup>5</sup> <http://netfx3.com/content/WFHome.aspx>

Writing code generation logic in C# (or VB), rather than in xml, has additional benefits - like reducing the development and testing effort drastically.

## Other Advantages of Blueprints

Apart from some of the major advantages that we discussed above, some other significant advantages of using Blueprints are:

- Blueprints brings the concept of nested blueprints where you can group multiple related blueprints together to build a parent blueprint which can be used to provide guidance and code generation for an end to end solution. This encourages blueprints reuse as you can create a repository of simple and lightweight blueprints and bundle related ones together to build richer and more complex blueprints according to your architectural needs.
- Delivering and Consuming Blueprints is as easy as exposing an RSS feed on the internet and downloading updates from it. In the case of Infosys, where we have a large developer base spread geographically, a RSS-based deployment will be ideal to manage the distribution of blueprints
- Support for automatic builds, using Team Foundation Server and Team Build Server
- Can act as a startup kit for a new development process. It reduces developer training time for advanced concepts & simplifies pattern implementation

## Infosys Blueprints

At Infosys, we have been experimenting with Blueprints. Our primary focus has been toward exploring and evaluating it in terms of its value proposition vis-à-vis GAT. To aid us in our work, we created a Silverlight<sup>6</sup> Software Factory demo, which internally contains three additional blueprints, viz. Silverlight Blueprint, WCF<sup>7</sup> Blueprint and LINQ<sup>8</sup> Blueprint

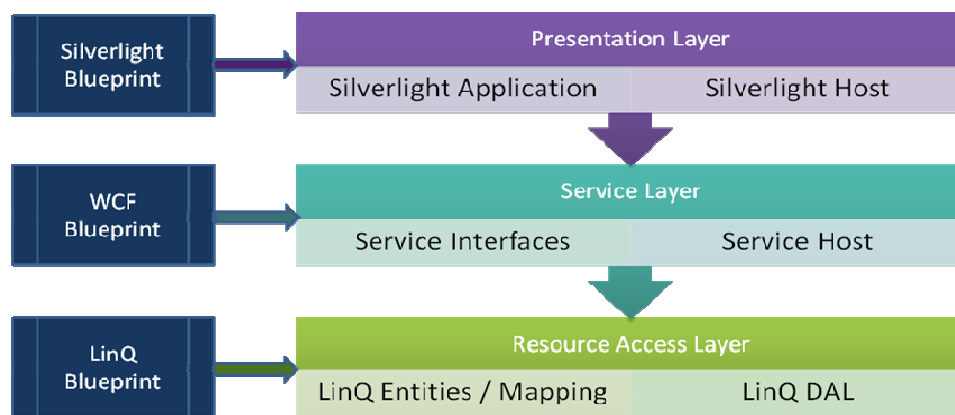


Figure 1: Logical Diagram - Silverlight Software Factory generated application

<sup>6</sup> <http://www.microsoft.com/Silverlight/>

<sup>7</sup> <http://netfx3.com/content/WCFHome.aspx>

<sup>8</sup> <http://msdn.microsoft.com/en-us/netframework/aa904594.aspx>

## Silverlight Software Factory

This is a demo software factory built to showcase the real power of Blueprints and how individual Blueprints can be made to work together to build rich software factories. Each Blueprint within this software factory provides guidance to implement a specific logical layer of a Silverlight application, using various technologies.

## LINQ Blueprint

LINQ Blueprint provides step-by-step guidance to a developer on how to build a data access layer on top of SQL Server using LINQ

### Features:

- Unfolds Data Access Layer in the current solution:
  - Creates data entities container project
  - Creates data access project
- Generate Data Access Layer (DAL) for selected entities using LINQ
  - Generates DAL classes with interface-based approach to support fetching, adding and updating data using LINQ
- Generates data entities from selected database tables and takes care of updating LINQ Mapping XML

### LINQ Blueprint – Workflow

Following is the workflow for implementing Data Access Layer in any application, using LINQ. Each workflow activity takes you through a wizard, which accepts certain inputs to generate the necessary code and configuration artifacts.

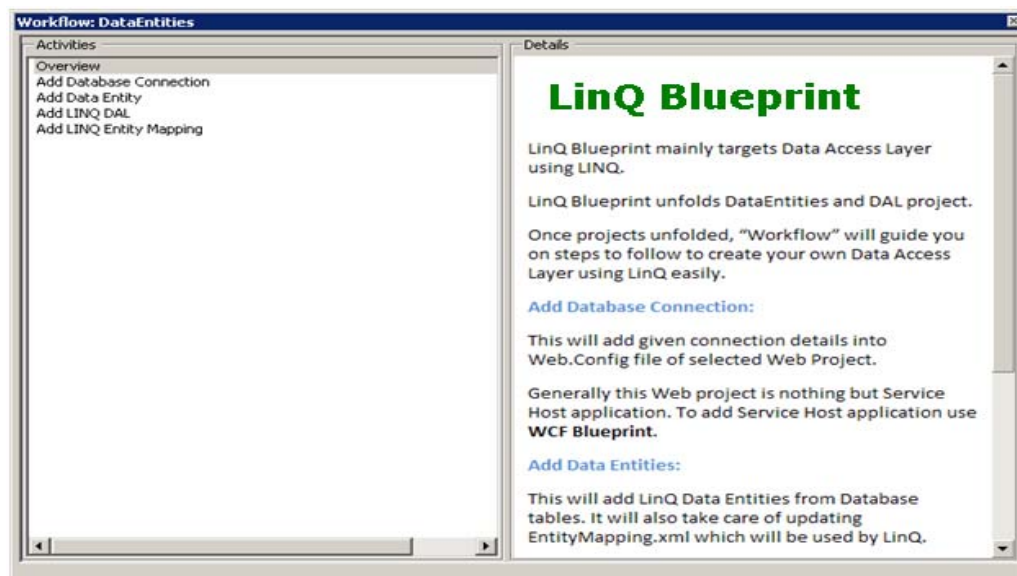


Figure 2: LINQ Blueprint Workflow

## WCF Blueprint

WCF Blueprint provides step-by-step guidance to a developer on how to implement a service interface layer using WCF and finally host it on IIS.

### Features:

- Unfolds service interface layer in the current solution. It creates the following projects in the solution:
  - Data Contracts
  - Message Contracts
  - Fault Contracts
  - Service Interface
  - Service Implementation
  - Service Host (Http Host)
- Generates WCF Service Interface Layer and Service Host
- Generates WCF wrapper artifacts (Data Contracts, Message Contracts, Service Contracts, Service Implementation), accepting DAL interface as an input
- Automates hosting of WCF services in web project and deploying the same on IIS

### WCF Blueprint – Workflow

Following is the workflow for implementing a Service Interface Layer in any application, using WCF. Each workflow activity takes you through a Wizard, which accepts certain inputs to generate the necessary code and configuration artifacts.

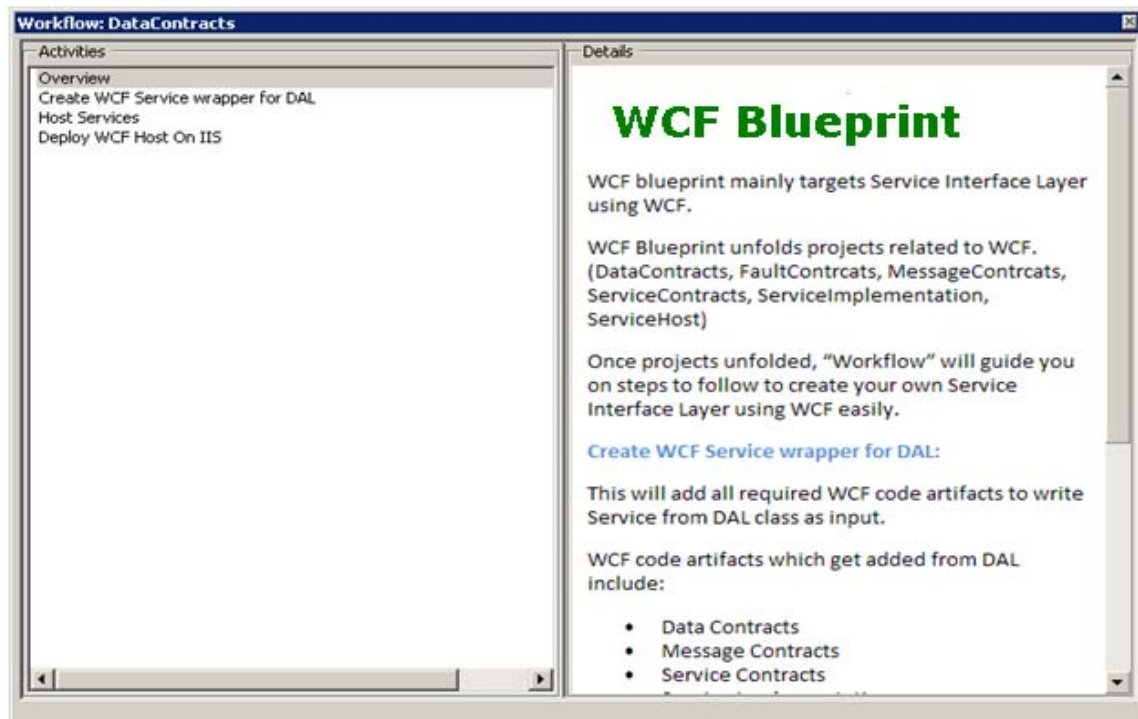


Figure 3: WCF Blueprint

## Silverlight Blueprint

Silverlight Blueprint provides end-to-end guidance to a developer on how to implement a presentation layer, using Silverlight and how to finally host it on IIS.

### Features:

- Unfolds presentation layer in the current solution
  - Creates Silverlight application project
- Generates Silverlight screens from service entities
- Generates Silverlight container application using ASP.NET
- Deploys Silverlight application on IIS

### Silverlight Blueprint – Workflow

Following is the workflow for implementing Presentation Layer, using Silverlight. Each Workflow item takes you through a Wizard, which accepts certain inputs to generate the necessary code and configuration artifacts. Generated code will be high quality, implementing best practices for Silverlight development.

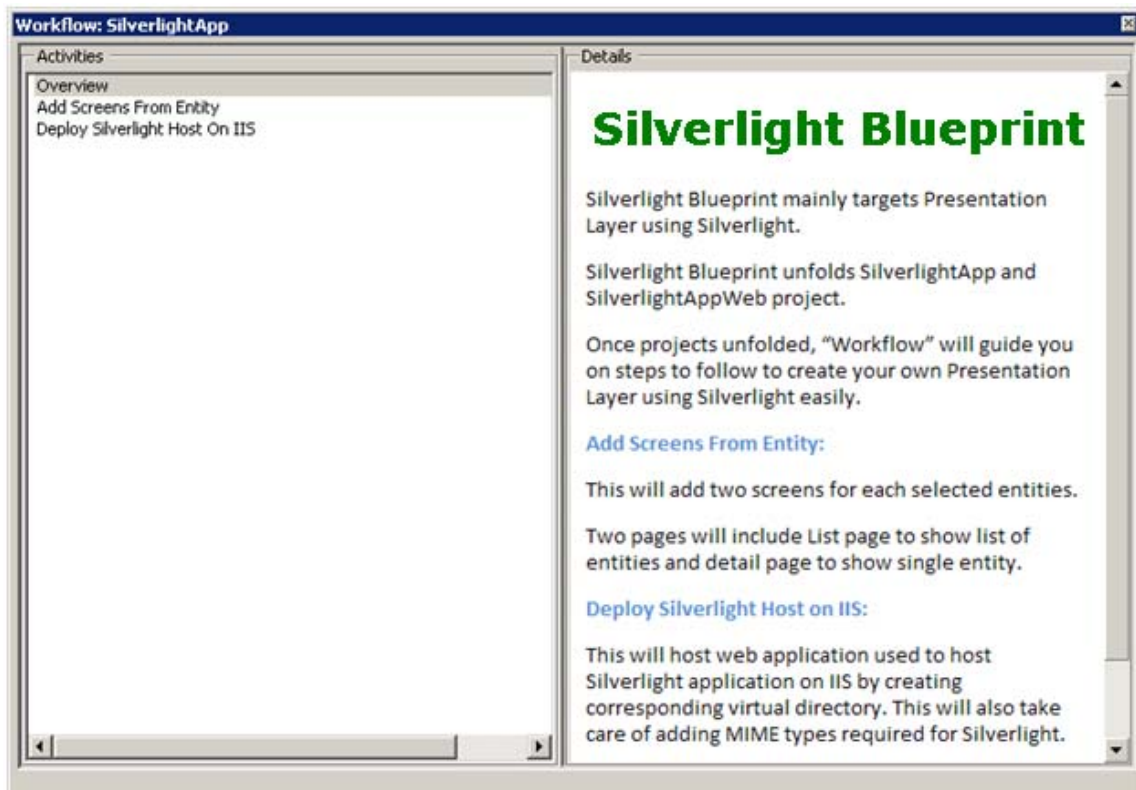


Figure 4: Silverlight Blueprint

## Overall Benefits of Infosys Blueprints

- Provides guidance
  - To build a web application using Silverlight, WCF and LINQ
  - To build a loosely coupled application, using Unity Application Block as DI Container
- Provides jump start to developers working on new technologies - like Silverlight, LINQ and WCF
- Makes it easy to understand architecture by providing actual implementation where theoretical documents are not appropriate
- Workflow - like guidance ensures developers follow guidelines as set by Architects to implement design
- Productivity improvement by generating solution architecture and repetitive code

## Summary

Infosys Blueprints demonstrates that Blueprints technology from Microsoft can be used to provide richer contextual guidance to developers. Blueprints technology is a lot simpler and richer than GAT/ GAX technologies and overcomes many limitations. Infosys Blueprints showcases some of the capabilities of Microsoft Blueprints and provides guidance for building an N-Layered Silverlight Application. Many Business and Architectural problems are much more complex and multifaceted than building a simple Silverlight Application and require non trivial effort, and richer blueprints can be developed to address their demands. Nevertheless, the example presented in this paper gives a fairly good look at Blueprints and how it can provide contextual guidance.

We believe the Blueprints technology advances the contextual guidance toolset by leaps and bounds. Although it is in beta, it holds a lot of promise. The software developer community is encouraged to use this technology to reap huge productivity benefits.

## About the Authors

**Farrukh Nizami** is a Senior Technical Architect with Infosys' Microsoft Technology Center (MTC). He leads a group on Component Based Development and Software Factories. He has 12+ years of experience in software design and development and has been working on Microsoft technologies and platforms, particularly on VB, C#, VB.NET, COM, MTS, MSMQ, SQL Server, etc. His expertise includes Object-Oriented Analysis and Designing techniques, Design Patterns, Enterprise Architecture & Patterns, and Aspect-Oriented Programming.

**Archana Ghag** is a Technical Architect with Infosys' Microsoft Technology Center (MTC). She is an expert on .NET3.0, 3.5 and GAT/ GAX technologies with current focus on building Software Factory, targeting ASP.NET applications. She has 8+ years of experience in software design and development on Microsoft Platform. She has worked on various technologies, including VB, C#, COM, MTS, SQL Server, etc.

## Acknowledgements

We would like to thank Jack Greenfield (Senior Director, Platform Architecture Team, Microsoft), for providing us a chance to participate in Software Factory Advisory Board Calls and connecting us to Microsoft Blueprints team.

We would like to thank Michael Lehman (Technical Evangelist, Platform Architecture Team, Microsoft), for providing us valuable guidance and support on Blueprints Technology, and for making himself available to enhance the tool and remove bugs in the shortest possible time.

We would like to thank Naveen Kumar and Atul Gupta (Principal Architects, MTC, Infosys) for their support in getting this paper quickly out in its current form.

© 2008 Infosys Technologies Limited.

ALL RIGHTS RESERVED

Copyright in whole and in part of this document "**Next Generation Software Factories – using Microsoft Blueprints**" belongs to Infosys Technologies Limited. This work may not be used, sold, transferred, adapted, abridged, copied or reproduced in whole or in part in any manner or form or in any media without the prior written consent of Infosys Technologies Limited.