The Anatomy of Software Frameworks

Sameer Paradkar

What are Frameworks?

Frameworks are not a new kid on the block, and tons of POC/White Papers have been written and placed in the public domain. This Article outlines historical data around frameworks and contains a real work VISUAL STUDIO feature to design application frameworks. So what is it about frameworks that makes them frameworks? Frameworks are skeleton applications, and they drive the workflow to a major extent, in contrast to application libraries, which are components that are passive in nature. On a given day, you still have the choice to build or not to build your application with a certain library feature, but with framework the go/no go has already been decided by the selection of the application framework. Elaborate and specific business features are then designed and built around this core framework. In earlier days, there were Code Generators, like the CodeDom/CodeSmith, that facilitated generation of factored code. A framework favors the recurring pattern once you analyze it, and if there is an element of commonality existing in the software that needs to be factored into a common software element, that element can be associated with the main framework skeleton by means of techniques that will be explained later.

Of the well-known frameworks that exist, MFC and Struts should be mentioned. Although from different worlds, they are very popular for developing applications in their core areas. Frameworks increase application productivity by bundling the common elements together, which then form the guidelines for the team members to follow. The terms intellisense, code generation, aspect driven programming are synonymous and imply similar things in the framework development world. Frameworks speed development, improve the quality of applications, and bring tremendous value to a business.

Frameworks Under the Hood

At times, it feels like an inter-galactic journey when I go through these public domain artifacts, dealing with certain topics of interest. I will try to keep things simple in this Article. All these names – SDK, Guidance ToolKits, ToolKits – cater in some way or other to applications of Frameworks. Their architects avoid always having to re-invent the wheel by housing the commonality into the frameworks. There are several specializations of frameworks, and their applicability depends on the context in which the framework will ultimately run. The mechanisms used to hook these pieces into the core skeleton can be one of the following: ***inheritance, composition, hook methods, reflections***.

Frameworks require the designer to do certain things in certain ways. A couple of years ago, when I was working with a team writing real-time software for a medical product, they had a rather crude mechanism (compared with our current times' mechanisms of reflections) devised by using a service configuration file that contained a listing of the extension components (DLLs) that would get plugged into the main application architecture at runtime through the DLL entry point functions. These DLLs would typically be post-processing applications that would run algorithms on archived data and generate results. In addition to the DLL entry point, there used to be a couple more parameters that would get loaded into memory from the service configuration file, such as the functions pointers and event handlers, among others, for asynchronous/synchronous communication with these applications. There surely had to be methods better than the entry point hook method.

In the remainder of this Article, I will explain standard approaches to framework development. An application based on a framework may also contain design patterns and components along with the core framework skeleton. A classic definition of frameworks reads, *"Frameworks are not simply collections of classes. Rather, frameworks come with rich functionality and strong wired-in interconnections between object classes that provide an infrastructure for the developer."* Frameworks are more than randomized bunches of components, but, as with components, one does not need to delve into framework internals. Designing and building a framework is like

driving a well-engineered automobile, such as a Porsche or a Jaguar--when you've completed the journey. you feel the exhilaration of having been involved with a challenging assignment.

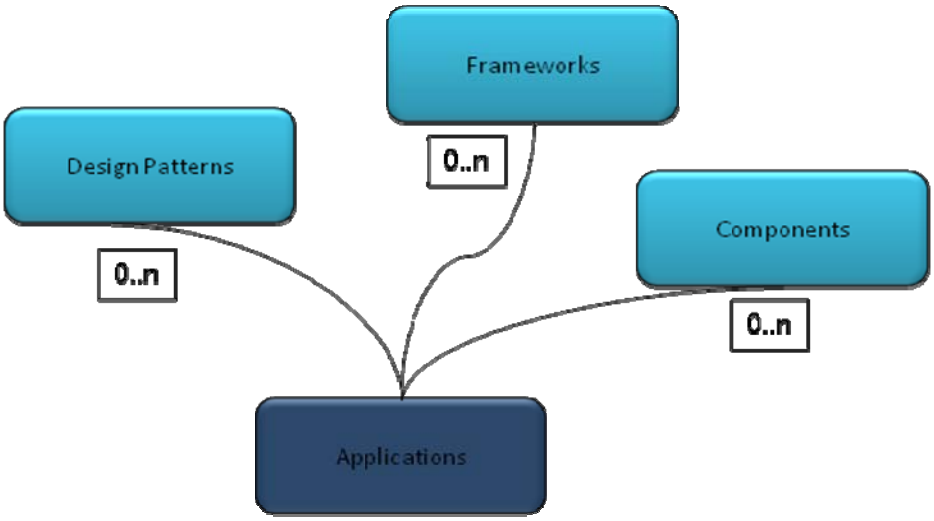

Figure 1. Application Frameworks Dependencies

Real World Frameworks

One of the key question during a decision making stage is what's the benefit in selecting a certain framework. The answer is that GUI frameworks don't have business logic, in contrast with application frameworks that would contain more than 70% of core functionality as part of the framework. Past history reveals that patterns could account for 36% of framework design. Lists of frameworks very popular in the software development world are as follows:

- Microsoft Foundation Classes - MFC
- Java's Abstract Window Tool-kit – AWT
- ACE, an OO framework
- Reusable Objects (ORO) is an open-source framework.
- Webridge Private Exchange is a horizontal framework designed for building B2B applications.
- Medical Business Object framework is a vertical framework designed for medical domain.

Framework Classifications

Frameworks are composed of numerous reusable assets, which include knowledge, design, captured requirements, and software components. As a coherent collection of reusable assets, a framework can be an important investment for an organization. Based on the context of its usage, the framework is classified into the following categories. The white-box category of frameworks is typically expanded by sub-classing existing abstract classes, whereas the black-box frameworks are more like components because they can be accessed only through their interfaces. The answer to the question, "How am I going to consume this to build my customized application?" will guide you with pointers about the classification of this framework.

- White Box Frameworks – Emphasis on Inheritance
- Black Box Frameworks – Emphasis on Composition
- Grey Box Frameworks – Framework is built using both the above approaches.

Framework Development Techniques

A few have been mentioned in this article earlier, but below is the complete listing of the standardized framework offerings.

- Common Spots – Concrete Classes
- Hotspots – Place Holders or Abstractions
  - Composition
  - Inheritance Approaches
    - Hook Methods
    - Template Methods

These hooks vary in purpose. One is a windows message hook; another is hook methods. Hook functions are typically achieved through virtualization. The windows hooks are message hooks that sneak peek into the windows message pumps for certain messages before the final destinations window consumes the messages. This is one of the advanced topics from the Win SDK.

One More Framework Classification

Based on how broad the Framework scope is we have another classification:

- Application Frameworks are horizontal frameworks applicability across domains.
- Domain Frameworks are vertical frameworks with specific applicability to a particular domain.
- Support Frameworks: Basic system level frameworks on which other frameworks or application would be built.

Framework Development Lifecycle

The framework lifecycle goes through the known software lifecycle methodology ***Analyze->Design->Built->Deploy***. Building universally applicable frameworks would require applying refactoring to domain and/or to an existing set of applications, and is not going to be an overnight process. The frameworks are not conventional software applications and require designer expertise to leverage mileage out of application frameworks. Customers of the frameworks are AP -> Application Programmers. One has to write a specialized application that would build functionality specific to the target domain to burn this so-called application framework. This is a highly iterative process, and only after several revisions would the final and stable product be ready for deployment and distribution.

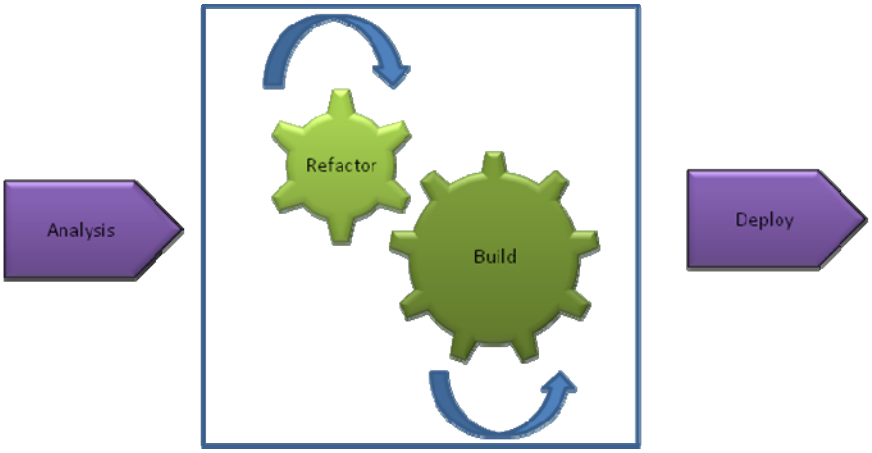

Figure 2. Application Frameworks Design and Build Process

Visual Studio Perspective: Enterprise Templates

There are several variations to this template guide using VISUAL STUDIO. The earlier visual studio nomenclature was VSTemplate, and now it has been renamed Enterprise template. The variations are item and project templates. At the core of Enterprise templates is the VSTemplate file that contains the listing of the entire file bundled with the template. This file is located inside a predefined folder for it to be made available for consumptions. This folder is located under ***"..\my Home Directory\Visual Studio 2XXX\Templates\ProjectTemplates."*** This would ensure that it appears under the New Project -> Visual C# -> My Templates for it to generate the framework skeleton into the new project workspace.

Well Enabled Services ToolKit – Framework

The following block diagram gives the context in which the ToolKit runs. It sits in between the IPTV and Connected Services Framework and is the mechanism to plug the VAS into the CSF. The WES framework provides partial implementation for provisioning, usage, and health. The provisioning interfaces are used to set up the resource (VAS End) for consumption. The usage interfaces are used within CSF for tracking and metering of VAS Consumption. Health interface is provided for getting the health status of VAS.

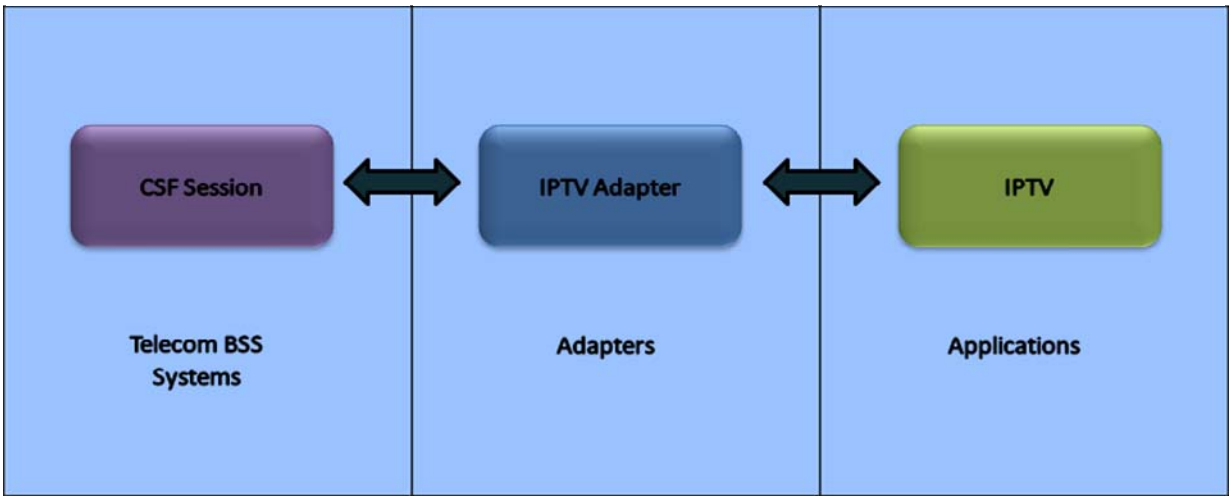

Figure 3. Adapter Architecture in Real World

Methodology of WES ToolKit

The WES Toolkit based service runs into a context that is a mediator between Value Added Services (LCS, IPTV, Microsoft Exchange, and Custom Web Services) and Connected Service Framework. Connected Services Frameworks is a Microsoft Product to provide value added services on wireless and wire line telecommunication networks of service providers. It bundles the complexity of connecting with multiple systems within the telecom enterprise into a simplified, easy use of interfaces so that providing value add services onto telecommunication lines is as simple as writing your specialized service and plugging it into CSF, and you are all set, So *what category of template is this WES ToolKit ? It falls into the Grey Box category, part inheritance and part composition, 50-50 types.* So when you plan to build a framework for your project the laundry list of things you may deliver includes reusability, simplicity, extensibility, flexibility, completeness, and consistency.

———

Author

Sameer S. Paradkar is an Enterprise Architect with Wipro Consulting Services. Sameer has 14 years of experience in IT services and consulting organizations and has worked across different industry verticals. Sameer can be reached at sameer.paradkar@wipro.com or +91-9850093199.

Disclaimer: The views expressed in this article / presentation are mine, and Wipro does not subscribe to the substance, veracity, or truthfulness of the said opinions.

References

- Refactoring: Improving the Design of Existing Code, Martin Fowler, Addison-Wesley.
- Design Patterns: Elements of Reusable Object-Oriented Software, Gamma, Helms, Johnson, Vlissides, Addison-Wesley.
- Connected Services Frameworks: http://www.microsoft.com/serviceproviders/solutions/connectedservicesframework.mspx

Domain Specific Language Tools: http://msdn2.microsoft.com/en-us/vstudio/aa718368.aspx

BPTrends Linkedin Discussion Group

We recently created a BPTrends Discussion Group on Linkedin to allow our members, readers and friends to freely exchange ideas on a wide variety of BPM related topics. We encourage you to initiate a new discussion on this publication or on other BPM related topics of interest to you, or to contribute to existing discussions. Go to Linkedin and join the **BPTrends Discussion Group.**