Towards Framework Selection Criteria and Suitability for an Application Framework

Sheikh I. Ahamed, Alex Pezewski and Al Pezewski
Marquette University
Milwaukee, WI 53201, USA
iq@mscs.mu.edu

Abstract

In recent years, frameworks are very popular as they provide reusable solutions to very difficult problems. However, the number of frameworks for a given domain makes it difficult to select the most appropriate framework. Selecting the wrong framework or selecting a framework that has not yet matured can be disastrous for a project. The choice of an application framework can make or break a project. Unfortunately, the lack of formal criteria and metrics to objectively compare/contrast frameworks, assess the framework's capabilities, and determine the framework's suitability to a particular project, make framework selection extremely difficult. Very little research has been conducted on determining the suitability of a framework. This paper defines criteria that can be used at a high-level to understand a framework's overall usefulness and at a low-level to understand the framework's suitability to a particular domain. The defined criteria of our approach are then used to determine the suitability of using an example framework.

Keywords: Framework selection, Framework suitability, Application framework

1.0 Introduction

Since the late 1990's, software engineers have gained an appreciation for the role that application frameworks can play in writing sophisticated applications. Regardless of domain, frameworks should be used (or at least seriously considered) for writing complex applications [1]. A framework is a reusable design of all or part of a system that is represented by a set of abstract classes and interfaces. The framework describes the way the objects interact [2]. A framework is not simply a collection of API's or a software library. Nor is a framework a code generator. Rather, a framework uses software libraries in conjunction with applying patterns to create the infrastructure for building an application. Through the use of interfaces and abstract classes, a framework imposes order and structure to an application. This order and structure allows the developer to concentrate on solving the mission critical aspects of the project rather than worrying about the "glue" that holds it all together.

Frameworks are critical for fully exploiting the benefits of object-oriented programming. By collecting related functionality into a single coherent whole, a framework provides a package of ready-to-use objects with clear procedures on their use, extension, and integration with an application. Frameworks aren't just collections of objects like many early class libraries; each framework has a common, unifying design philosophy and does as much work as possible for the developer [7].

Most people would agree on the value of using application frameworks. However, due to the overwhelming number of available frameworks, it can be a daunting task to determine which framework to use or if it is more appropriate to write a new framework. Selecting the wrong framework can be disastrous as software engineers struggle to make the project fit the inappropriate framework and write additional code to make up for the inappropriate framework's deficiencies and to satisfy the framework's interfaces, interfaces that may not have anything to do with the project's requirements. There are also situations where it might appear that a framework should be used but it is probably better to not use a framework. Given how critical it is to select the appropriate framework for a given application and to understand when using a framework is inadvisable, it was surprising to find that very little research has been conducted in the area of framework suitability. To address the above problems, we proposed an approach for selecting framework suitable framework with a detailed example framework.

In section 2, we describe the Wafer project and the criteria it uses for reviewing web domain frameworks. In section 3, criteria are provided that can be used to assess a framework's strengths and weaknesses. This information, along with an understanding of a project's requirements, can also be used to determine the applicability of a particular framework to a project. Details are provided for each defined item and the criteria are applied to assess the suitability of the jPOS[14] ISO 8583 framework for use in a transaction

Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04)
0-7695-2108-8/04 $ 20.00 © 2004 IEEE

processing system..

2.0 Related Works

Given the popularity and the number of available frameworks, it was unexpected to find only two approaches for various frameworks (specifically, web frameworks) and only one open source community project, Wafer (Web Application Framework Research) [5]., comparing features of various web frameworks. The two sites, Wafer and Frameworks-Builder.org, both support the Wafer project on SourceForge[9].

Wafer is a research project, which compares many open source Java web application frameworks. For this comparison, Wafer uses each framework to implement a common web log application. This allows frameworks to be objectively compared and allows the focus of the comparison to be the merits of each framework and not the application [6].By using each framework to implement a defined web log application each framework can be compared with a common reference point [7].The criteria chosen by Wafer to compare/contrast frameworks is focused on open source, Java-based, web application frameworks. .

Frameworks-Builder.org [9] is a work in progress providing fairly high-level information about web frameworks. A link is provided to a table summarizing information about J2EE-compliant, PHP, and Python web frameworks. The table presents level of activity (measured loosely in number of downloads and CVS commits in "recent" months), licensing, release date, links to reviews, papers, and books, what makes the framework unique, compatible IDE's and plug-ins, author and user comments.

The above projects has the limitations of : Incomplete criteria., Specific to web framework domain., Specific to Java language., Criteria mostly limited to a high-level listing of information., Lack of formal metrics. and Only superficially addresses suitability of framework question

3. 0 Proposed Solution

In order to judge an application framework and determine its suitability to a problem domain, many more questions need to be answered than the previous solutions provide and more formalized software engineering metrics must be defined.

The following areas have to be considered when selecting a framework. This list is made up of intentionally general criteria that can be applied to just about any application framework regardless of intended domain.

The Framework's Intended Domain: There are frameworks available for a variety of domains including web application frameworks, network frameworks, and media frameworks, to mention a few. When creating a new framework, a domain analysis is first performed to better understand the requirements and concepts the framework needs to address. Existing applications, if any, are reviewed and domain experts are consulted. In addition to this, existing standards for the domain are studied. The result of the activity is a domain analysis model, containing therequirements of the domain, the domain concepts, and the relationships between those concepts [8].

Flexibility in Extending the Framework: By providing well-defined interfaces (hook methods), frameworks are able to adapt to a variety of needs. Well-defined interfaces decouple the application domain's stable interfaces and behaviors from the variations found in different settings. Framework extensibility allows the requirements of different settings to be readily handled, allowing for timely customization of new application services and features [2].

Provisions for internationalization (I18N): Framework support for internationalization (I18N) can save a considerable amount of development time and effort in handling items like character encoding and addressing locale-specific requirements. A common misconception is to think that I18N is only important for web frameworks. However, it can have a bearing on where your applications will be accepted.

Security: The framework selected should have support for your organization's security infrastructure. The framework might also provide additional security functionality that can be very useful to the organization.

Error Handling: The framework might favor or accommodate certain ways of handling errors than other error-handling methods. For example, it might be more appropriate to pass exceptions up the hierarchy if the framework uses declarative programming for handling errors rather than trying to handle the exception where it took place.

Third-Party Product Integration: Development tools such as IDE's are tremendous time-savers for developers. Similarly, organizations tend to adopt and have the infrastructure in place for a particular version control systems (example, PVCS). A certain amount of testing of the framework and these tools during a trial period might be necessary to make sure that they can be integrated.

Coupling: The terms, tightly coupled and loosely coupled, refer to the level of dependence the components of a software system have with each other. A tightly coupled system is highly prone to undesired side effects as changes are made. For example, an application might have a GUI that is tightly coupled to the application's database. Should the GUI change or the

database change, a high level of care must be taken or the application might not function correctly. This can result in complete software failure, data not being displayed properly, or bad information persisted in the database. A loosely coupled system is highly desired and would allow the GUI or database to be changed with minimal impact to the overall functionality of the application..

Design Patterns: The design patterns used by the framework should be scrutinized. When selecting a framework, the organization's experienced software architects and engineers are in a position to understand the real problem that is trying to be solved. These individuals should think about the design patterns that they would select if they were designing a framework to handle the requirements.

.**Language Requirements:** Frameworks tend to target a specific programming language. If your organization has settled on a particular programming language, the language and platform will be two of the major criteria in determining what frameworks are available to you.

Platform Requirements: Generally, the targeted platform will restrict the framework selection by the programming languages that are supported. However, it needs to also be confirmed that the platform's operating system and/or operating system version are compatible with the considered framework. This is particularly true of closed or proprietary platforms where choice of framework and version might be dictated by the platform vendor.

Configuration Files: As part of making the framework applicable to as wide an audience as possible for a given domain, as well as to reduce or eliminate code changes, frameworks place setup, configuration, environmental, and application flow-of-control information in external files. These files might be in a proprietary format but frameworks have generally adopted eXtensible Markup Language (XML) to describe this information.

Framework Complexity: There is a fine balancing act between creating a framework that is too flexible, too broad in scope, and one that is too limited and too small in scope. More flexible frameworks tend to be more complex than more limited frameworks. The former are better suited to experienced software engineers, while the latter might be more suited for more junior staff or limited use. Similarly, a flexible framework will provide room to grow, while a simpler framework might need to be replaced after a period of time depending on the changing needs of the organization. [2].

Framework Inefficiencies: Frameworks enhance extensibility by employing additional levels of indirection (example, through the use of dynamic binding). The resulting generality and flexibility often reduce efficiency. For instance, in languages like C++ and Java, the use of dynamic binding makes it impractical to support Concrete Data Types (CDTs), which are often required for time-critical software. The lack of CDTs yields (1) an increase in storage layout (example, embedded pointers in virtual tables), (2) performance degradation (example, additional overhead of invoking a dynamically bound method and the inability to inline small methods), and (3) a lack of flexibility (example, inability to place objects in shared memory) [2].

Framework Maturity, Reliability, and Stability: Framework stability is a good indicator of general framework maturity. To assess the general maturity of a framework, in terms of structural and behavioral stability, available documentation, framework vendor support, and so on, metrics can be used to provide an objective evaluation. Metrics based on the framework code, leads to a more objective evaluation than other aspects of maturity including version information and time on the market [10].

Interoperability: In today's computing environment, it is very unusual for any reasonably complex application to be entirely self-sufficient. Applications need to access databases, the security infrastructure, and provide information to other applications, example, middleware, to name a few. When considering the suitability of a framework, it needs to be determined whether the framework can interoperate with the intended environment

Implementation: The above criteria are defined to be intentionally general so that they can be widely applied. This will be demonstrated as the criteria are applied to the jPOS framework. Among other potential uses, the jPOS framework can be used to implement financial interchanges, protocol converters, payment gateways, and credit card verification clients/servers [13].

The Framework's Intended Domain: The jPOS framework is intended for use with interchanging popular financial transaction message formats such as ISO 8583, ANSI X9.2, and VISA-1. It can also be extended to implement any particular message interchange [jPOS 2003]. jPOS provides a ready-made infrastructure for parsing and formatting (building) transaction messages. To make the framework highly usable, the way a message is formatted must be entirely at the discretion of the developer using the framework. As a simple example, the framework should not require a particular field in a message. This is logic that is implementation-specific. By including requirements such as whether a field is optional, conditional, or mandatory, the framework's applicability and suitability to a particular domain is greatly diminished.

Flexibility in Extending the Framework: Any framework that is responsible for transaction message parsing and formatting must provide a means to support messages and protocols other than the ones the original authors envisioned. jPOS provides this flexibility by

using an abstract component that can be extended to create a particular type of field. This field can than be packaged in a message container. The jPOS framework, allows for implementation-specific requirements, for example, unique headers and trailers. In fact, the jPOS framework even allows an entire message to be a field of another message.

Internationalization (I18N): Financial messages tend to be encoded using either the ASCII or EBCDIC character encoding standards, which consist of single bytes to depict a character. As a result, lack of I18N support for transaction interchange by the jPOS framework is generally not an issue. However, the jPOS utilities do not have support for dumping messages containing accents and the like. If special characters are used, for example, in an address or merchant name of an ISOMsg, the ISOMSG.dumpString() logic needs to be updated.Security

The jPOS framework provides security support as it pertains to interchanging messages. Some examples include an interface for key stores, an interface for communicating with a security device, a class for transporting keys, encrypted PIN block handling, Derived Unique Key Per Transaction (DUKPT) and Data Encryption Standard (DES) single, double, and triple length keys support. The jPOS framework uses the Java Cryptographic Extensions (JCE).

Error Handling: The jPOS framework includes a number of exceptions that are thrown when the framework encounters an issue. Unfortunately, the jPOS JavaDoc does not provide much information describing why an exception exists.

Third-Party Product Integration: Information is provided at the jPOS web site for integrating the jPOS framework with the Eclipse IDE. However, any IDE, code editor, and version control system with support for Java, should be able to be used for editing and managing code created for use with the jPOS framework.

Coupling: The jPOS framework relies on several industry standard solutions that are free to use. This usage of off-the-shelf software promotes using well-supported solutions and limits dependencies to well-known points. Some examples of off-the-shelf software use include activation.jar and mail.jar for the Operator Log Listener, comm.jar for communications, jsse.jar for SSL support, jce.jar for cryptography, and jetty.jar for Jetty HTTP and servlet container integration.

Design Patterns: The jPOS framework uses a number of well-established and well-chosen patterns. For example, the composite and leaf patterns are used to define a message and its fields, respectively. The delegate pattern is used to define the logging event manager. When we considered examples for this paper, we considered a point-of-sale user interface as well as a message formatter/parser

Language Requirements: As its name suggests, the jPOS framework is a Java framework requiring Java 1.3 or later. There is a C version available for a fee.

Platform Requirements: Any platform with support for Java 1.3 and sufficient resources and appropriate connectivity can be used with the jPOS framework. The jPOS frequently asked questions indicates successful installations on AIX, Linux, Novell (NLM on server), Windows 9x/NT, and OS/400. It is also possible to run jPOS in a 1.2 Java Runtime Environment, and instructions are provided to make the necessary compatibility adjustments.

Configuration Files: Unlike frameworks such as Struts, where XML configuration files are an essential element of the framework, jPOS' configuration is very modest with the majority of configuration-related items dealing with third-party libraries (see Coupling for examples).

Framework Complexity: To utilize the jPOS framework correctly, the developer should have at least an intermediary knowledge of Java and the patterns used by the framework (see Patterns). The developer should also have a strong understanding of transaction processing, the interchange rules, thread handling, and encryption requirements. A rudimentary understanding of ANT is also necessary for generating the JavaDoc, building the examples, and for ongoing development. Experience with using Concurrent Versions System (CVS) and SourceForge is helpful for contributing to the open source project as well as receiving updates prior to their formal release. The framework is written in such a way that the work can be readily divided into functional areas and spread across development teams.

Framework Inefficiencies: jPOS is a well-designed framework fulfilling a need of every transaction processor and payment system. The implemented patterns require a low-level of overhead. Transaction processing requires the ability to acquire and issue transactions asynchronously. The thread model used by the jPOS framework reduces potential bottlenecks by wrapping the communications channel with a MUX class. By separating the multithreaded handling from the message (package) and the communications (channel), the implementation is easier to scale and easier to support.

Framework Maturity, Reliability, and Stability: jPOS is presently at version 1.4.7. jPOS was created by Alejandro Pablo Revilla and first made publicly available on April 23, 1999. According to the jPOS home page, there have been over 15,000 downloads and jPOS is actively used by over 500 personal and corporate users in more than 56 countries.jPOS makes strong use of off-the-shelf products.

There are a number of critical areas not covered by the existing methods but are included in the proposed approach. In addition to addressing the incomplete

criteria used by the existing methods, the proposed solution is applicable to any framework domain, regardless of language, the proposed solution recommends metrics that can be used to assess framework stability, and the proposed solution's criteria are comprehensive, balanced and consist of substantive data

4.0 Conclusion

In considering the questions associated with determining the applicability of a framework for an intended domain, it was unexpected to find that this area has received very little study. This paper attempts to fill the void by defining general yet meaningful criteria that can be applied to virtually all frameworks, regardless of intended domain. The criteria, while extensive, are still manageable. Our criteria is described with an example framework. This list will help in defining the merits of a framework while eliminating details that might detract from that analysis. Organizations might find it helpful to provide a relative weight to each item so that competing frameworks can be more objectively scored. When applying weights, it is important to reduce the chance for personal bias. Because of the general nature of the criteria, this goal can be achieved

Further research is needed to define metrics to further qualify a framework's extensibility, efficiency, and scalability. While a metric was described to predict framework stability, reliability, and maturity, that metric can be extended to also examine the potential for code refactoring be achieved.

References

[1] Cavaness, Chuck. *Programming Jakarta Struts*, Sebastopol, CA: O'Reilly & Associates, 2003.

[2] Fayad, Mohamed E., Douglas C. Schmidt and Ralph E. Johnson, *Building Application Frameworks*, New York, NY: John Wiley & Sons, 1999.

[3] Thomas, Dave. "Building Adaptable Systems – A Conversation with Andy Hunt and Dave Thomas, Part V", Bill Venners (moderator). URL: http://www.artima.com/intv/adapt4.html, March 31, 2003.

[4] Hunt, Andy. "Building Adaptable Systems – A Conversation with Andy Hunt and Dave Thomas, Part V", Bill Venners (moderator). URL: http://www.artima.com/intv/adapt4.html, March 31, 2003.

[5] Eden, Anthony. "Wafer" (Web Application Framework Research) project. URL: http://www.waferproject.org/index.html

[6] Eden, Anthony. "Wafer: Feature Matrix Explained". URL: http://www.waferproject.org/feature-matrix.html (explanation)

[7] Double, Chris. "Chris Double's Radio Weblog", URL: http://radio.weblogs.com/0102385/2003/09/27.html. September 27, 2003.

[8] Schäfer, W., R. Prieto-Diaz and M. Matsumoto. *Software Reusability*. Ellis-Horwood Ltd., 1994.

[9] Thompson, Kris. "Frameworks-Builders.org". URL: http://www.frameworks-boulder.org/

[10] Mattsson, Michael and Jan Bosch. "Characterizing Stability in Evolving Frameworks", *IEEE Technology of Object-Oriented Languages and Systems*, 1999. Proceedings of, Meeting Date: 06/07/1999-06/10/1999, Jul 1999 Pages:118-130

[11] J. Bansiya. "Evaluating Application Framework Architecture Structural and Functional Stability", accepted for publication in *Object-Oriented Application Frameworks: Problems & Perspectives*, M. E. Fayad, D. C. Schmidt, R. E. Johnson (eds.), Wiley & Sons, URL: http://indus.cs.uah.edu/research-papers.htm, February 1998

[12] J. Bansiya. Assessment of Application Framework Maturity Using Design Metrics, *ACM Computing Surveys - Symposia on Object-Oriented Application Frameworks*, February 1998, URL: http://indus.cs.uah.edu/research-papers.htm

[13] Revilla, Alejandro P. "jPOS Programmers' Guide", jPOS.org. 2002.

[14] jPOS.org. About *jPOS.org*. URL: http://www.jpos.org

[15] Rosenthal, Arnon and Eric Hughes, "What is an Application Framework?" The MITRE Corporation. Http://www.mitre.org/tech/facelift/reports/Fmwk-Def-subm.html.

[16] Eden, Anthony (project admin). SourceForge.NET Wafer Project. http://sourceforge.net/projects/wafer/

[17] Clementson, Bill, "Bill Clementson's Blog". http://home.comcast.net/~bc19191/2003_09_21_bill-clementson_archive.html. September 27, 2003.

[18] Zhu, Feng; Tsai, Wei-Tek, "Framework-Oriented Analysis" *IEEE Computer Software and Applications Conference*, COMPSAC '98 The Twenty-Second Annual International, 19-21 August 1998. Pages: 324-329.