

.Net Foundation Framework

Architecture and Code Review Assessment

DRAFT - Findings & Recommendations Report

June 26, 2009

Prepared By:



Contact Information

For inquiries regarding this document contact:

Headquarters - Philadelphia, Pennsylvania

David Consulting Group

1770 E. Lancaster Avenue, Suite 15

Paoli, PA 19301

Web Site: www.davidconsultinggroup.com

DRAFT

Table of Contents

1. Executive Summary.....	3
1.1 Performance.....	3
1.2 Maintenance.....	3
1.3 Reusability.....	4
1.4 Key Findings.....	4
1.5 Key Recommendations.....	4
2. Foundation Framework Technical Overview.....	6
2.1 Technologies Used.....	6
2.2 Layout and Statistics.....	6
3. Assessment Methodology.....	7
3.1 Automated Analysis - CAST Application Intelligence Platform (AIP).....	7
3.2 Architectural Analysis.....	8
4. Assessment Findings.....	10
4.1 Summary of CAST AIP Metrics.....	Error! Bookmark not defined.
4.2 Cyclomatic Complexity.....	Error! Bookmark not defined.
4.3 SEI Maintainability.....	Error! Bookmark not defined.
4.4 Technical Quality Index.....	Error! Bookmark not defined.
4.5 Changeability.....	Error! Bookmark not defined.
5. Architectural Considerations.....	Error! Bookmark not defined.
5.1 Service Oriented Architecture (SOA).....	Error! Bookmark not defined.
5.2 Typical Framework Concerns.....	Error! Bookmark not defined.
6. Framework Comparison Alternatives.....	Error! Bookmark not defined.

1. Executive Summary

Overall the framework is well designed and is based upon .Net 2.0/3.0 and the Microsoft Patterns and Practice team product Enterprise Library. Designing a common framework has a normal goal to help keep maintenance of applications lower while providing consistency for areas like data access, exception handling, caching, etc. Making the framework flexible, agile, and loosely coupled sometimes runs counter to maximum performance and scalability though. The overall benefit of a common framework should be judged on performance, maintenance, and reusability

1.1 Performance

This framework should be stress tested to flush out any areas not meeting expected performance guidelines. Some design areas of the framework should be reviewed and thoroughly tested to look for improvements and design updates. Some of these areas include:

- Custom AJAX implementation
- Version of Enterprise Library
- Membership, role, and profile providers
- Use of reflection
- Caching

1.2 Maintenance

Custom implementations of functionality in a framework similar to existing industry best practices will always result in higher maintenance costs over the long term. For instance on the AJAX technology, Microsoft is constantly improving this functionality to push the envelope on richer web applications. Many improvements were made from .Net 2.0 to .Net 3.5, with many more improvements in .Net 4.0 in the near future. Adding custom functionality like this to the framework will add to software maintenance lifecycle costs.

The Framework.Web.UI.Controls (~45% of the lines of code in the framework) is another example of functionality extended from Microsoft controls. These controls are constantly being improved along with the AJAX control toolkits and even Silverlight. Building this functionality into the framework does help with consistency, but will raise maintenance costs to keep these controls up to date while not breaking older applications. Many times core frameworks like this one will not include UI functionality for this reason. Industry best practice would be to utilize Microsoft or third party controls. Master pages and application themes with styles can control consistency of web sites without the need to roll custom controls in the framework.

The dependency injection framework also competes with industry best practices. Currently Unity in the latest version of Enterprise Library or other alternatives like Spring.Net are great design choices.

1.3 Reusability

The framework is not well documented with code comments in the underlying classes. The comments in many cases are simple generic statements that is already known information and not descriptive enough to really help a developer.

The complexity of the framework also will raise the learning curve of new developers. The framework is fairly complex and without good sample quick start applications, this framework will be hard to learn.

1.4 Key Findings

The libraries are not all built against the latest version of the CLR. This is preventing the use of the latest version of the Enterprise Library that is the cornerstone of the solution.

The use of the ObjectBuilder for creational patterns is causing unnecessary instantiation steps while adding maintenance complexity via required configuration settings. In the current design there is seemingly no need for dependency injection.

The framework is built for the stated purpose of reuse. However, the framework is tightly coupled to every layer of the solution. A framework addressing data access, security, caching and other such enterprise level functionality would serve its purpose of reusability much better supporting a properly designed Service Oriented Architecture.

Common functionality, such as authentication and authorization, are not provided by transparently located services. While there are virtual layers of abstraction between layers of functionality the solution is tightly coupled. This monolithic, client server approach to reuse is antiquated and limiting.

The solution is completely dependant on the Microsoft platform and lacks the use of industry standards for communication, security, messaging, and invocation. This oversight limits the solution's ability to adapt and take advantage of current and future technologies. This is very likely to become an obstacle to lowering the cost of ownership and will increase the cost of maintenance.

1.5 Key Recommendations

The framework should leverage the most recent version of the Microsoft Enterprise Library as this version has matured considerably.

The solution would benefit greatly by introducing sound SOA practices. By encapsulating common functionality in services that are accessible using industry standards rather than referenced libraries, the goal of reuse is achieved with

greater certainty. This also ensures a longer solution lifetime while reducing the risks of rewriting, compiling and distributing code.

Goals of the framework should be restated, evaluated and measured against business metrics to ensure those goals have been met.

A new caching strategy should be addressed. A caching system that is not forced to reside on the same server running the framework will reduce resource usage and increase performance. There are Open Source solutions that offer much better performance and the ability to distribute caching capabilities.

DRAFT

2. Foundation Framework Technical Overview

2.1 Technologies Used

Foundation is a .Net 2.0 technology framework, using some of the .Net 3.0 Windows Workflow Foundation. Foundation is written in C#, using Visual Studio 2005 and the WF Extensions for Visual Studio 2005.

Foundation references or makes use of the following external libraries:

- Microsoft Enterprise Library v2.0 (January 2006)
- Microsoft Object Builder 1.0
- Microsoft Report Viewer v8.0
- Oracle Data Provider 2.102.2
- Crystal Reports 11.5
- FxCop v1.35

Additionally, the Foundation framework uses the following .Net framework features:

- .Net 2.0 Membership & Profile
- P/Invoke

2.2 Layout and Statistics

The Foundation framework is composed of 8 separate projects, listed in the table below.

Project	Lines of Code	Number of Files	Number of Classes
Framework	6,160	89	86
Framework.Security	2,658	16	17
Framework.Utility	2,528	30	37
Framework.Web	40,693	275	353
Framework.Web.Reports	6,426	10	126
Framework.Web.Resources	287	2	2
Framework.Web.Workflow	1,509	7	17
Framework.Workflow	725	7	12
Total Foundation Framework	60,986	436	650

A ninth project, Framework.Build, is used as part of an automated build process.

3. Assessment Methodology

This assessment is the first part of an effort to determine if the applications built using the Foundation framework will meet the Company's objectives for performance and maintainability. This assessment will determine whether the Foundation framework is constructed according to industry best practices and with the high quality level necessary for success. For the Foundation framework, DCG also considers the need for the framework to be accessible and useable by multiple development partners across multiple Company organizations. This intended level of reuse requires a higher than normal level of documentation, explanations about design choices, and defensive programming to ensure that future users and maintainers are able to reap the full value of the framework.

In this assessment, DCG is focusing on the analysis of the source code of the framework, not on the performance of a running application. As with all frameworks, the implementation of the specific application is critical to the perceived success of the framework. This will be addressed in the following two portions of the overall assessment, with a review of a specific application (WebAIM) and a performance test of a running application.

DCG uses the Best-of-Breed automated analysis tool, CAST AIP, to automatically scan the entire code base as well as having expert .Net architects review the architecture, design, and code against current industry practices and standard approaches.

3.1 Automated Analysis - CAST Application Intelligence Platform (AIP)

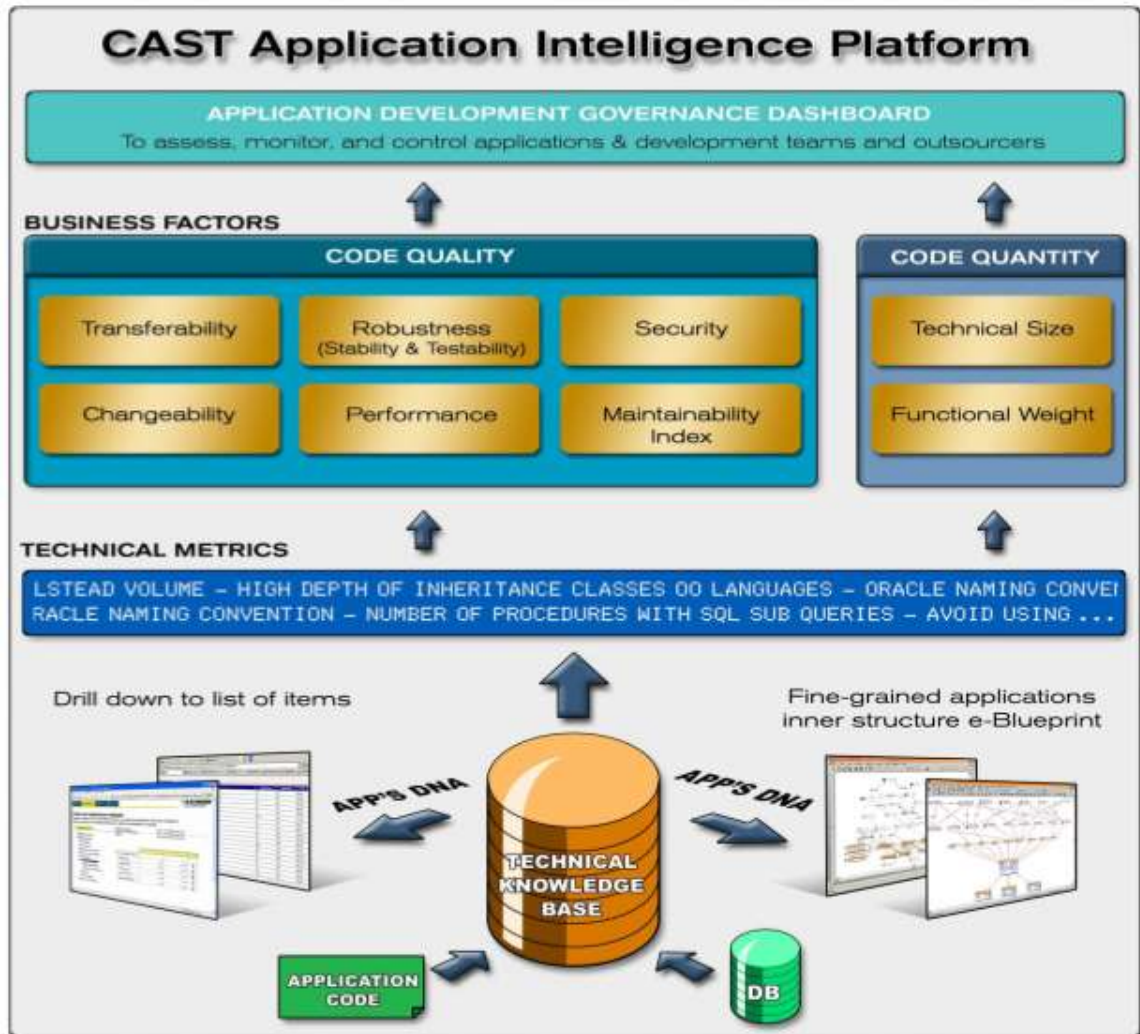
The CAST AI Platform is the industry leading automated code analysis platform, with coverage of all major development tools and languages. CAST AIP automatically scans and analyzes all of the source code and database elements that are part of an Enterprise system. CAST AIP applies over 800 metrics based on standards and measurements developed by the Software Engineering Institute (SEI), International Standards Organization (ISO), Center for Software Engineering, and Institute of Electrical and Electronics Engineers (IEEE). These metrics objectively measure software for the quality and quantity of work.

CAST AIP provides Application Analysts the ability to examine and drill down on critical application characteristics and attributes. The primary Application Health Factors that are addressed are:

- Transferability
- Changeability
- Robustness
- Performance
- Security
- Maintainability
- Technical Size

- Functional Weight

These factors are used as an objective measure of application development to bring visibility in to what teams are delivering, and the health of your applications and provide a roadmap for improving applications.



3.2 Architectural Analysis

DCG consultants review the overall design and architecture of the application along with the specific implementation choices embedded in the application code. These experienced .Net Architects apply real world understanding and knowledge of current and prior technology patterns and best practices to evaluate the quality and long term viability of a given application. The review considers factors such as the ability of developers to understand and maintain the application code, the flexibility and expandability of the system, the reuse of existing libraries, and the use of standard techniques. While reviewing the code, the architects apply their hard-earned wisdom to identify common performance, security, and maintenance problems.

By considering alternative technologies and techniques available at the time of original development and at the time of assessment, and incorporating an understanding of the goals for the application, DCG is able to provide a balanced assessment of whether the application under review is suitable as a strategic platform and the level of change necessary to achieve the enterprise goals.

In this Foundation framework assessment, DCG is particularly focused on questions of whether the framework as implemented can support the high performance web applications needed within the COMPANY organization, and whether the framework follows industry standard Service Oriented Architecture (SOA) principles.

DRAFT

4. Assessment Findings

Want to know more....

Contact DCG at info@davidconsultinggroup.com

Tony Timbol
Vice President, Sales and Marketing
<http://www.davidconsultinggroup.com/>
1770 E. Lancaster Ave, Suite 15
Paoli, PA, 19301
+1.904.287.0294 Phone
+1.904.287.0544 Fax
+1.904.614.0931 Mobile
+1.610.644.2856 ext 28 Corp Offices
Measure. Improve. Deliver.

DRAFT