Taxonomy of Java Web Application Frameworks

Tony C Shan
Wachovia Bank
tonycshan@gmail.com

Winnie W Hua
CTS Inc.
winniehua@yahoo.com

Abstract

This paper describes various web application frameworks and related emerging technologies pertinent to the Java EE model from a technical perspective. A definition of "web application framework" is specified, as this terminology has been widely used and implies drastically different meanings in different contexts. The value proposition of a web application framework is presented to illustrate how a framework can improve the application development productivity and quality. The design philosophy of web application frameworks is articulated. A comprehensive taxonomic scheme is defined to classify various software frameworks and web application frameworks into appropriate categories. Among dozens of web application frameworks available as commercial and open source solutions, the predominant products are investigated, followed by the selection guidelines and recommendations. A reference card is constructed to summarize the key aspects of web application frameworks. Relevant technologies and future trends are also discussed.

1. Introduction

In information systems environment, a framework is a defined support structure in which other software applications can be organized and developed. A framework may include support programs, code libraries, a scripting language, common services, interfaces, or other software packages/utilities to help develop and glue together the different components of a software application. A software framework is a reusable design and building blocks for a software system and/or subsystem.

A software framework can be geared toward constructing applications in different domains, such as financial modeling applications [1] or decision support systems [2].

A software framework consists of frozen spots and hot spots [3]. The frozen spots define the overall architecture of a software system – its basic components and the relationships between them. These remain unchanged (frozen) in any instantiation of the application framework. On the other hand, hot spots represent those parts of the software framework that are specific to individual software systems. Hot spots are designed to be generic. In other words, they can be adapted to the needs of the application under development.

Software frameworks define the places in the architecture where adaptations for specific functionality should be made - the hot spots. In an object-oriented environment, a framework consists of abstract and concrete classes. Instantiation of such a framework consists of composing and subclassing the existing classes [4].

When developing a concrete software system with a software framework, the hot spots are specialized according to the specific needs and requirements of the system. Software frameworks rely on the Hollywood Principle: "Don't call us, we'll call you." [5]. This means that the user-defined classes such as new subclasses receive messages from the predefined framework classes. These are usually handled by implementing superclass abstract methods, in a way similar to the use of the Template design pattern.

2. Software frameworks

2.1. Software framework types

In general, there are seven types of software frameworks in the information systems space:

- Conceptual Framework – overarching architectural model such as the Zachman Framework;
- Application Framework – skeletal structure for an application solution such as WebWork;
- Domain Framework – tailored to specific business sectors such as IBM Information Framework (IFW);
- Platform Framework – programming model and runtime environment such as .Net and Java EE framework;

- Component Framework – building blocks for an application such as Hibernate, iBatis and Cayenne for object-relational mapping;
- Service Framework – business and technical services model for service-oriented computing such as Semantic Web Services Framework;
- Development Framework – a construction foundation to build a rich-client development tool, typically for IDEs, such as Eclipse, Netbeans, and OSGi.

2.2. Definition of Web Application Framework

A Web Application Framework (WAF) is a reusable, skeletal, semi-complete modular platform that can be specialized to produce custom web applications, which commonly serve the web browsers via the Http(s) protocol. It includes building blocks of services and components that are essential for constructing sophisticated feature-rich business service and collaboration systems. WAF usually implements the Model-View-Controller (MVC) design pattern, typically in the Model 2 architecture to develop request-response web-based applications on the Java EE and .Net models. It also integrates services such as search, versioning, and permissions into the basic business objects, enabling applications to leverage framework services with little or no extra work. WAF in this context is a type of Application Framework as defined in the preceding section, specifically for Http(s)-based communications serving HTML/XML.

The Web Application Framework domain layer usually models basic concepts such as users, groups, and permissions. WAF may also include other relevant parts such as a user interface (UI) framework, a UI component library designed for the rapid development and reuse of web user interfaces, and a powerful object-relational persistence engine/utility.

2.3. Why use Web Application Frameworks?

2.3.1. Open standard architecture

The software frameworks significantly reduce the amount of time, effort, and resources required to develop and maintain web applications. Moreover, a framework is an open architecture based on commonly accepted standards (e.g., Java, .Net, XML, XSLT, JAAS, Servlet, JSP, JDBC, ADO.Net) and technologies (e.g., JUnit, XUnit, Ant, Log4j, JDom, Xalan, Xerces, Lucene), enabling any experienced developer to rapidly develop and support the system without a steep learning curve.

This best-of-breed approach to adopting and integrating technologies enables application designers to focus on solving their business problems. Thus, adopting Web Application Frameworks as the standard development infrastructure for web applications is the best way to ensure that development is not locked into any proprietary, dead-end architecture. This approach dramatically reduces technology churn and risk since the industry-standard open source frameworks are actively maintained and enhanced by the highly skilled professionals in the world. These developers take the responsibility of identifying the appropriate technologies, integrating the software, testing these technologies, and providing migration paths for existing users to the latest technology.

A Web Application Framework is typically deployed in an n-tier architecture and uses proven, standard technology. By using standard technologies, a framework can be easily deployed within existing enterprise infrastructures, leveraging existing hardware, software, processes, and people.

2.3.2. Relevant domain services

Virtually all web applications have a common set of basic requirements, such as user management (e.g., secure user login, password recovery), group management, and access authorization. A Web Application Framework usually includes all these functionalities, refined through hundreds of production deployments, freeing developers to focus on the needs of their specific application.

In addition to a basic set of services, web applications typically have two other important similarities: they store important data in a relational database and they interact with users via a web-based user interface. A sophisticated object-relational persistence layer automatically manages how model objects are stored in the database. The persistence layer, by generating optimized SQL from metadata, drastically reduces the amount of effort required to model and refactor database schemas or support additional database architectures.

A Web Application Framework may also include a component-based presentation rendering framework, that enables developers to extend existing UI components or build new components that can be reused throughout an application, e.g. XSL and tag libraries. A breadth of domain services such as data validation, versioning, categorization, print, page navigation and full-text search may be provided. Any application written on top of a Web Application Framework can transparently and immediately take advantage of these basic services.

3. Web Application Frameworks solutions

3.1. Design philosophy

The key design principles that are applied to develop a web application framework are as follows.

- Simplicity – less and simpler code should be written to use a framework. Avoid overuse of XML configuration files. Take advantage of the POJO-centric design.
- Consistency - components, containers and conventions should be consistent
- Efficiency - applications should perform well and scale with support of clustering through sticky sessions preferred.
- Integration – a framework should not compete with good existing solutions, but should foster seamless integration.
- Reusability – constructs in a framework should be fully reusable and are easy to distribute/deploy.
- Non-intrusive – HTML or other markup should not be polluted with programming semantics, with compatibility with ordinary HMTL editors and easy manipulation fro graphics designer to recognize and avoid framework tagging.
- Diagnosis - when things go wrong, the framework should not get in the way; in fact, it should provide useful diagnostics and debugging information.
- Development tools – maximum tool support with minimum dependency on special tools.

3.2. Web Application Framework types

Almost all Java Web application frameworks are based on the MVC pattern. Generally speaking, there are currently five major schools of web application frameworks: **Request-based**, **Component-based**, **Hybrid**, **Meta**, and **RIA-based Framework**.

A *Request-based Framework* is very close to the original CGI specification. It uses controllers and actions that directly handle incoming requests. Each request is essentially stateless. With the introduction of server-side sessions, a certain degree of statefulness has been achieved. Various frameworks basically differentiate themselves by the way they map the logic to URLs and how data is structured and provided to the business handlers.

A *Component-based Framework* abstracts the internals of the request handling and encapsulates the logic into reusable components, often independent from the web medium. The state is automatically handled by the framework, based on the data that is present in each component instance. Together with some form of event handling, this development model is very similar to the features offered by desktop GUI toolkits. Various frameworks basically differentiate themselves by the provided component API and how components are integrated together.

A *Hybrid Framework* combines both request-based and component-based frameworks by taking control of the entire data and logic flow in a request-based model. Developers remain close to the architecture of CGI applications and have full control over URLs, forms, parameters, cookies and pathinfos. However, instead of mapping actions and controllers directly to the request, a hybrid framework provides a component object model that behaves identically in many different situations such as individual pages, intercepted requests, portal-like page fragments and integratable widgets. Components can be wired together and be packaged as groups that are components in their own right. They can be distributed separately and be seamlessly integrated into other projects. This combines the form of reusability in component-based frameworks with the raw control of a request-based approach.

A *Meta Framework* has a set of core interfaces for common services and a highly extensible backbone for integrating components and services. The structure typically implements the Inversion of Control pattern for separation of concerns to flexibly incorporate other frameworks and components. A Meta Framework is sometimes considered as a framework of frameworks.

Rich Internet Application (RIA) refers to a web page-based application running in a browser with rich user interface features that are common in “fat” client, such as drag and drop, tree controls, and tabbed panels. A *RIA-based Framework* uses a client-side container model, which minimizes the amount of server communication – instead of loading an entire HTML page each time the user clicks, the framework either handle the click locally (without involving the server) or request data from the server in an XML format. This means that there truly is a client-side application with state and a user interaction model – the client is far more than a web page generated on a server.

3.3. Major products

3.3.1. Request-based framework

<u>WebWork</u>

WebWork [6] provides robust support for building reusable UI templates, such as form controls, UI themes, internationalization, dynamic form parameter mapping to JavaBeans, and robust client and server side validation.

WebWork was originally developed by Rickard Oberg in 2001, and released as an open source project on SourceForge in March 2002. WebWork joined the OpenSymphony project in the summer of 2002. As of November 2005, WebWork was merged into Struts to become part of Struts Action 2.0 framework.

<u>Struts</u>

Struts [7] uses and extends the Java Servlet API to adopt the "Model 2" approach, a variation of the

IEEE COMPUTER SOCIETY

classic Model-View-Controller (MVC) design pattern. Under Model 2, a Servlet (or equivalent) manages business logic execution, and presentation logic resides mainly in server pages.

Struts was originally developed by Craig McClanahan and donated to the Apache Foundation in May 2000. Struts has been a de facto framework with a strong and vibrant user community. The Apache Struts project is now comprised of two distinct frameworks. The two frameworks are the Struts Action Framework and the Struts Shale Framework. In addition, The Simple Web Framework (SWF) is an event-based framework targeting Struts developers who want to build rich Web applications but do not want to migrate to JSF.

Beehive

Beehive [8] (Apache 2006) is an extensible Java application framework with an integrated metadata-driven programming model for web services, web applications, and resource access. The framework leverages the latest innovations in the Java 5, particularly JSR 175 metadata annotations. The key components are NetUI Page Flow, Controls, and Web Service Metadata.

Beehive evolved from a part of BEA Weblogic Workshop product to an Apache project in May 2004.

Stripes

Stripes [9] is a robust yet lightweight presentation framework with zero external configuration per page/action, binding engine that builds complex object webs out of the request parameters, built-in support for multiple events per form, transparent file upload capabilities, and wizard forms.

Stripes was initially released in September 2005.

3.3.2. Component-based framework

JSF

JavaServer Faces (JSF) [10, 11] is a server-side user interface component framework for Java-based Web applications. JSF contains an API for representing UI components and managing their state; handling events, server-side validation, and data conversion; defining page navigation; supporting internationalization and accessibility; and providing extensibility for all these features. It also contains two JSP (JavaServer Pages) custom tag libraries for expressing UI components within a JSP page and for wiring components to server-side objects.

The specification of JSF 1.0 (JSR-127) was initially released in March 2004. JSF 1.2 Specification (JSR-252) is the next generation of JSF, which has a final draft released in August 2005.

Tapestry

Tapestry [12] complements and builds upon the standard Java Servlet API, and divides a web application into a set of pages, each constructed from components. This provides a consistent structure, allowing the Tapestry framework to assume responsibility for key concerns such as URL construction and dispatch, persistent state storage on the client or on the server, user input validation, localization/internationalization, and exception reporting. Developing Tapestry applications involves creating HTML templates using plain HTML, and combining the templates with small amounts of Java code using (optional) XML descriptor files. In Tapestry, an application is created in terms of objects, and the methods and properties of those objects -- and specifically not in terms of URLs and query parameters. Tapestry brings true object-oriented development to Java web applications.

Tapestry was originally created by Howard Lewis Ship, and the project was moved to the Apache Foundation around early 2004.

Wicket

Wicket [13] is a framework that takes simplicity, separation of concerns and ease of development to a new level. Wicket pages can be mocked up, previewed and later revised using standard WYSIWYG HTML design tools. Dynamic content processing and form handling is all handled in Java code using a component model backed by POJO data beans that can easily be persisted using different technology. Wicket has a transparent state management with no XML configuration files.

The first release of Wicket 1.0 went public in June 2005.

3.3.3. Hybrid framework

RIFE

RIFE [14] is a full-stack web application framework with tools and APIs to implement most common web features. Each of its toolkits is usable by itself and together they offer powerful integrated features that boost your productivity. RIFE ensures that every declaration and definition is handled in one place in the code. This simplifies the developer's task by reducing code replication, enforcing consistency, and easing maintenance.

RIFE 1.0 was released in September 2005.

3.3.4. Meta framework

Keel

Keel [15] is ready made server side infrastructure. Keel incorporates multiple open source projects to provide a best of breed framework that works right out of the box. What differentiates Keel is its component design, allowing new implementations to be added and old implementations replaced without requiring extensive rewrites of the application code.

The first pre-release of Keel was out in January 2003, though the initial group was formed in February 2002 and Keel was named in March of the same year.

Spring

Spring [16] is a layered Java EE application framework, which includes a lightweight container for automated configuration and wiring of application objects via inversion of control (aka dependency injection), an abstraction layer for transaction management, a JDBC abstraction layer, AOP functionality, and integration with O-R mappers.

The origins of Spring can be traced back to a book by Rod Johnson [17], who presented his interface 21 framework that was later released into the open source world, this framework formed the foundation of the Spring framework. The first official 1.0 release was available in March 2004.

3.3.5. RIA-based framework

DWR

Direct Web Remoting [18] is a framework of calling Java code on the server directly from JavaScript in the browser. DWR consists of two main parts: JavaScript running in the user's browser to communicate with the server and dynamically update the webpage, and a Java Servlet running on the server that processes requests and sends responses back to the browser. DWR takes a novel approach to Ajax by dynamically generating JavaScript code-based Java classes. Consequently the web developer can use Java code from JavaScript as if it were local to the web-browser; whereas in reality the Java code runs in the web-server and has full access to web-server resources

The first release of DWR went public in 2005.

Echo2

Echo2 [19] is the next-generation of the Echo Web Framework, a platform for developing web-based applications that approach the capabilities of rich clients. The 2.0 version holds true to the core concepts of Echo while providing dramatic performance, capability, and user-experience enhancements made possible by its new Ajax-based rendering engine.

Echo2 removes the developer from having to think in terms of "page-based" applications and enables him or her to develop applications using the conventional object-oriented and event-driven paradigm for user interface development. Knowledge of HTML, HTTP, and JavaScript is not required. Applications may be hosted using any Java Servlet container. Echo2, like its predecessor, is open-source software distributed under the terms of the Mozilla Public License or the GNU LGPL License, if preferred.

Echo2 initial alpha release became available in March 2005.

JSON-RPC-Java

JSON-RPC-Java [20] is a dynamic JSON-RPC implementation in Java. It allows you to transparently call server-side Java code from JavaScript with an included lightweight JSON-RPC JavaScript client. It is designed to run in a Servlet container such as Tomcat and can be used with JBoss and other J2EE Application servers to allow calling of plain Java or EJB methods from within a JavaScript DHTML web application.

Minimal or zero changes are necessary to existing server-side Java code to allow calling from JavaScript (such as the marshalling and unmarshalling of special types) as JSON-RPC-Java dynamically maps JavaScript objects to and from Java objects using Java reflection. JSON-RPC-Java allows simple exporting of Java objects by reflection on their method signatures (a single line of code is required to provide access to all public methods of a Java object).

The first major release 1.0 was made available in March 2006.

3.4. Selection Guidelines

Selection of one of these frameworks is dependent on the overall architectural decisions as well as the specific functionality of the application. Frameworks are not always required but can significantly reduce the time and effort needed to develop an application from scratch. Other architectural considerations may drive the use of specific frameworks or the migration from an old framework to a mainstream one in the design process.

Struts is supported in IBM WebSphere Application Server and WebSphere Studio Application Developer (WSAD) IDE as well as the Rational Application Developer, which replaces the WSAD.

Similarly, JSF is supported in IBM WebSphere Application Server and WebSphere Studio Application Developer IDE as well as the new Rational Application Developer. Sun Microsystems provides a JSF Reference Implementation, which it is free to redistribute in binary form. In addition, Apache MyFaces project offers an open source JSF implementation of JSR-127.

MyEclipse Enterprise Workbench (Version 4.0) supports Sun JSF Reference Implementation 1.1.01, and MyFaces 1.0.9. It also provides Tapestry integration.

Use of Beehive requires JDK 1.5. The Beehive product supports JSR-175 metadata annotations. Beehive's NetUI Page Flow is based on Struts. Pollinate is an Eclipse technology project slated to build an Eclipse-based IDE and toolset that leverages the open source Apache Beehive application framework.

The mainstay of the Struts framework is the controller components, which can be used with any Java presentation technology. JSF and JavaServer Standard Tag Library (JSTL) are complementary to Struts, and extensions are available.

Spring provides an AOP framework, a unique transaction management abstraction, and a unique data access abstraction. Spring Modules is an associated project, which extends the reach of the Spring platform to areas that are not necessarily integral to the Spring core. Spring IDE is a graphical user interface for the configuration files used by the Spring Framework. It is built as a set of plugins for the Eclipse platform.

As a general implementation guideline in web application designs, Spring may be considered first as a preferred baseline, and subsequently an appropriate front controller implementation is justified – Struts, JSF, Wicket, and RIFE. Flexible rendering mechanisms should be evaluated. Beehive can be used in combination with Struts. The Page Flow in Beehive is more mature than the WebFlow in Spring. Stripes is a good candidate if an application needs wizard forms. Tapestry may be used in niche areas at discretion. DWR and Echo2 are viable options to develop RIA systems.

A reference card is constructed to summarize the key aspects in web application frameworks, as illustrated in Table 1.

4. Other frameworks

4.1. Java

The following table lists the major Java web application frameworks that are not addressed in the foregoing sections.

Cocoon	Millstone	OXF	SOFIA	Canyamo
Maverick	JPublish	JATO	Folium	Jucas
Bishop	Niggle	Barracuda	Shocks	Jeenius
Verge	wingS	Expresso	Melati	Xoplon
jZonic	TeaServlet	JBanana	Chiba	Jacquard
OpenEmcee	Bento	Genie	JWarp	Macaw
Baritus	Nacho	Turbine	Dovetail	Scope
JWAA	Smile	Jaffa	Warfare	JFormular
Dovetail	Japple	Cameleon	Cassandra	Helma

4.2. Relevant Technologies

In the .NET world, the current release of Microsoft Enterprise Library contains seven application blocks. These application blocks are components and aspect-oriented APIs for .NET applications. The Enterprise Library is not considered as a web application framework in this context.

Another application block by Microsoft, called User Interface Process (UIP), is not included in the current release of Enterprise Library. UIP Version 2.0 implements the MVC pattern, and provides reusable code components for web session state management, navigation and layout management, and wizard support. UIP can be categorized as a web application framework.

The Spring framework has been ported to .Net The Spring Rich Client Project (RCP) is a sub-project to provide an elegant way to build highly-configurable, GUI-standards-following rich-client applications more rapidly by leveraging the Spring Framework, and a rich library of UI factories and support classes.

Additionally, DotNetNuke is an open source web application framework, which is an evolution of Microsoft's IBuySpy Portal Solution Kit. DotNetNuke has a basic core that can be extended using pluggable modules and providers with additional functionality. The look and feel of individual sites can be customized using skins, which provides a clear separation between design and content, enabling a web designer to develop skins without requiring any specialist knowledge of development in ASP.NET: only knowledge of HTML and an understanding of how to prepare and package the skins themselves are required.

OpenLaszlo is an open source platform for creating zero-install web applications with the user interface capabilities of desktop client software. OpenLaszlo programs are written in XML and JavaScript and transparently compiled to Flash. DHTML support was recently added.

6. Future Trends

The need for a web application framework is compelling as there have been dozens of frameworks built in the Java space over the years. Among these competing offerings, a few products have stood out to become the predominant contenders for various reasons such as strong user community, standardization, good documentation, tool integration, vendor endorsement, and easy to use. Some examples are Struts/Shale, JSF/MyFaces, Spring, Tapestry, Beehive, and DWR.

Table 1. Web Application Frameworks Reference Card

WAF Type	Primary Attributes	Product	Key Features	Guidelines/Trends
Request-based	• Controllers and actions to handle incoming requests • Stateless • Logic mapped to URLs	Struts	▪ Extend Java Servlet API ▪ Adopt the "Model 2" approach ▪ Action and ActionForm	- Decomposed to two frameworks: Struts Action Framework (merger with WebWork) and Struts Shale Framework (incorporation of JSF) - Predominant front controller implementation
		Beehive	▪ Metadata-driven programming model ▪ NetUI Page Flow ▪ Controls	- Required JDK 1.5 - NetUI Page Flow is based on Struts - More mature than WebFlow in Spring - Pollinate is an Eclipse-based IDE
		Stripes	▪ Lightweight presentation framework ▪ Built-in support for multiple events per form ▪ Transparent file upload	- Good for building wizard forms in an application - Indexed property support - Reuse ActionBeans as view helpers
Component-based	• Logic encapsulated in components • State handled in component instance • Similar to GUI development	JSF	▪ User interface components ▪ API for handling states, events, server-side evaluation ▪ 2 JSP custom tag libraries	- MyFaces is an open source implementation - Reference implementation offered by Sun Microsystems - MyEclipse Enterprise Workbench support
		Tapestry	▪ Component-based structure to construct pages ▪ Templates ▪ Special HTML attribute to denote components	- Applicable in niche areas - Easy editing of templates with ordinary HTML editors - MyEclipse Enterprise Workbench support
		Wicket	▪ Mock-up page via WYSIWYG HTML tools ▪ POJO data beans ▪ Transparent state management with no XML configurations	- Combination of the component model in Echo with the Special HTML attribute to denote components in Tapestry - Use POJO to automate the server-side state management
Hybrid	• Data and logic flow in a request-based model • Component object model to handle actions and controllers • Combination of raw control and reusability in two types of frameworks	RIFE	▪ Logic-less HTML templates ▪ Uniform component model ▪ Metaprogramming ▪ Multi-dimensional conversational state management with scoping ▪ Language-independent template engine ▪ Persistence layer with content management integration and versioning	- Work with domain-specific API and language - Automatic generation of CRUD - Rich dynamic metadata APIs for JavaBeans instances and their properties - Integration with DWR
Meta	• Core interfaces for common services • Extensible backbone for integrating components and services • Inversion of Control pattern	Keel	▪ Multi-layer structure: security, database abstraction, messaging, business logic, and user interface layer ▪ Strict separation of roles and interfaces ▪ Auto-configuration	- Service-based architecture - Extensible structure - Open platform - Anti-patterns collected - Leverage Avalon framework
		Spring	▪ Light-weight container for automated configuration and wiring of application objects ▪ Abstraction layer for transaction management ▪ AOP functionality	- A preferred baseline - Incorporate an appropriate front controller implementation in design - Spring IDE is a graphical user interface for the configuration files used by the Spring Framework
RIA-based	• Rich Internet Application • Client-side container model • Stateful user interaction	DWR	▪ Invoking Java objects at the server side from JavaScript in the browser ▪ Dynamically generating JavaScript code-based Java classes	- Support of DWR in request- and component-based frameworks - Reverse Ajax - Pageable and sortable lists
		Echo2	▪ Ajax-based rendering engine ▪ Object-oriented and event-driven paradigm for user interface development ▪ Server push technology	- Event-oriented design - Support modal dialogs - EchoStudio2 is an Eclipse plug-in for visual development: Form Editor and StyleSheet Editor
		JSON-RPC-Java	▪ JSON-RPC implementation in Java ▪ Light-weight RPC JavaScript client ▪ Basic ORB (Object Request Broker) functionality	- Lightweight JSON format instead of XML for speed - Remote procedure call protocol - Transparent marshalling/unmarshalling of primitive types

Consolidation has occurred and tends to go stronger, as demonstrated in the recent merger of WebWork and Struts (Action Framework), and the incorporation of JSF into Struts (Shale) and Spring (JSF-Spring).

A steady stream of new frameworks continues to proliferate, particularly in the evolving RIA field, which is wide open. Virtually all of them are in the format of open source, even for the vendor-backed products like Echo2. Apparently open source is the direction to go for the future framework development and enhancement.

Use of Meta Frameworks is growing rapidly, as seen in the adoption of Spring. It is expected that more integration and convergence will occur based on the Meta Frameworks, which provide a good foundation to plug and play various components and services, such as Freemaker. The structure tends to become more flexible and loose-coupled, allowing easy swapping of competing technologies to construct best-of-breed solutions tailored to specific domains.

Simplicity will be a critical factor for a framework to sustain in this crowded space. Light-weight technologies like Ruby on Rails and Laszlo pose strong challenges to the "traditional" programming frameworks in the enterprise computing environment. Other related products and emerging technologies will influence the future outlook of web application frameworks, such as XUL, General Interface, XAML, Ideaburst, and Kanemea.

7. Conclusions

A web application framework is a critical design aspect in the distributed system development. This paper describes various web application frameworks and related emerging technologies pertinent to the Java EE model from a technical perspective. A definition of "web application framework" is specified, as this terminology has been widely used and implies drastically different meanings in different contexts. The value proposition of a web application framework is presented to illustrate how a framework can improve the application development productivity and quality. The design philosophy of web application frameworks is articulated. A comprehensive taxonomy is defined to classify various software frameworks and web application frameworks into categories. Among dozens of web application frameworks available as commercial and open source solutions, the predominant products are investigated, followed by the selection guidelines and recommendations. A reference card is constructed to summarize the key aspects of web application frameworks. Relevant technologies and future trends are also discussed.

8. References

[1] Birrer, A., & Eggenschwiler, T., Frameworks in the financial engineering domain: an experience report. in (eds), *Proceedings of the European conference on object-oriented programming*, Springer-Verlag, Kaiserslautern, Germany, 1993, pp 21-35.

[2] Gachet, A., "Software Frameworks for Developing Decision Support Systems - A New Component in the Classification of DSS Development Tools", *Journal of Decision Systems*, Lavoisier, France, 2003, 12(3/4): 271-281.

[3] Pree, W., Meta patterns - a means for capturing the essentials of reusable object-oriented design. in M. Tokoro and R. Pareschi (eds), *Proceedings of the ECOOP*, Springer-Verlag, Bologna, Italy, 1994, pp 150-162.

[4] Buschmann, F., *Pattern-oriented software architecture: a system of patterns*, Wiley, Chichester; New York, 1996.

[5] Larman, C., *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process*, Prentice Hall PTR, Upper Saddle River, NJ, 2002.

[6] OpenSymphony, WebWork Framework, http://www.opensymphony.com/webwork, 2006.

[7] Struts, Struts Framework, http://struts.apache.org, 2006.

[8] Apache, Beehive Framework, http://beehive.apache.org, 2006.

[9] Stripes, Stripes Framework, http://www.mc4j.org/confluence/display/stripes/Home, 2006.

[10] JSR-127, JaveServer Faces, http://www.jcp.org/en/jsr/detail?id=127, 2004.

[11] JSR-252, JaveServer Faces 1.2, http://www.jcp.org/en/jsr/detail?id=252, 2006.

[12] Jakarta, Tapestry Framework, http://jakarta.apache.org/tapestry, 2006.

[13] Wicket, Wicket Framework, http://wicket.sourceforge.net, 2006.

[14] RIFE, RIFE Framework, http://rifers.org, 2006.

[15] Keel, Keel Framework, http://www.keelframework.org, 2006.

[16] Spring, Spring Framework, http://www.springframework.org, 2006.

[17] Johnson, R., *Expert One-to-One J2EE Design and Development*. Wrox, UK, 2002.

[18] DWR, DWR Framework, http://getahead.ltd.uk/dwr, 2006.

[19] Echo, Echo2 Framework, http://www.nextapp.com/platform/echo2/echo, 2005.

[20] JSON, JSON-RPC-Java Framework, http://oss.metaparadigm.com/jsonrpc, 2006.

IEEE COMPUTER SOCIETY