

A Survey of Design Pattern Based Web Applications

Sridaran R, Campus Head, ICFAI National College, Vellore-632 006 India.

Padmavathi G, Professor and Head, Department of Computer Science, Avinashilingam University for Women, Coimbatore-641 043, India.

Iyakutti K, Senior Professor, School of Physics, Madurai Kamaraj University, Madurai-625 019, India.

Abstract

Pattern-based web applications have become popular since they promote reusability and consistency. In few cases, patterns do not produce the desired effect because of lack of experience in applying them. This situation forces one to think of a suitable re-engineering solution for such applications. The objectives of the paper are three fold. It provides a survey of different pattern-based web applications that will be useful for the application designers. It highlights some of the web applications where patterns have been inappropriately handled. A few re-engineering initiatives for such cases are also analyzed.

Key Words: Patterns, Web Applications, Re-engineering, Hypermedia, Semantic web, Partitioning.

1 INTRODUCTION

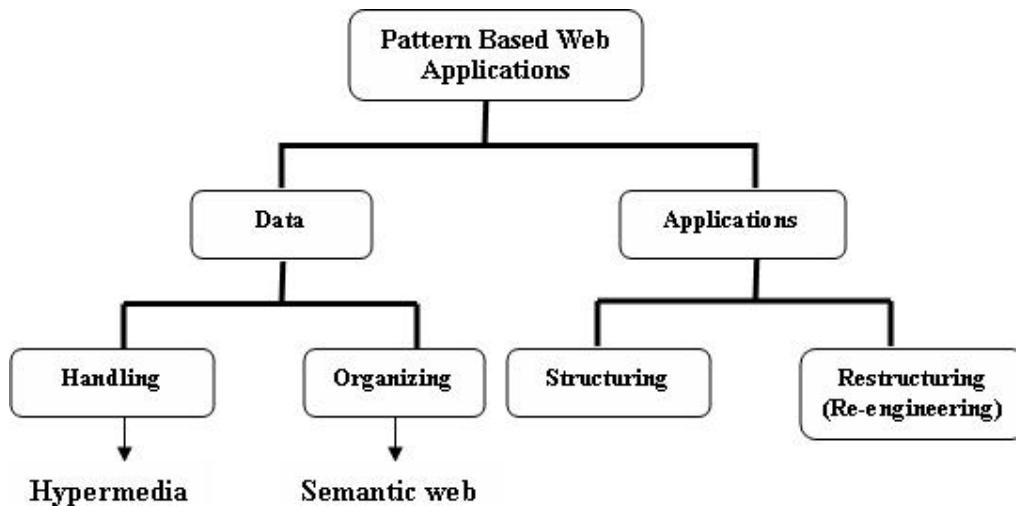
Web application designers need to address many challenges during development in order to comply the quality of service requirements including speed, scalability and security. In recent years numerous non-web based applications have been re-written as web based because of the ever growing business needs. These migration activities are much more complicated and time consuming than a new software development. Most of these challenges are with data handling, organizing, or structuring of the web applications. Design Patterns [Gamma95] [Cooper00] or simply patterns are extensions of object-oriented technologies that are chosen as building blocks by architects and developers. This has given birth to a new class of web applications known as pattern-based web applications. The pattern-based web applications promote architectural reusability and consistency but in certain cases they may also produce adverse effects if not handled properly.

Pattern-based web applications are convenient ways of reusing object-oriented code among projects. In such applications, each pattern participates in the form of classes and/or objects with defined responsibilities. The captured common participants are documented as experiences for future usage. The ‘good’ ones are known as patterns and ‘bad’ ones as ‘anti-patterns’. These documented patterns are reused at suitable contexts in the future and hence they promote architectural reusability.

There exist a few threats in pattern-based web applications.

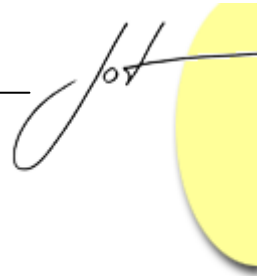
- Since patterns exist in large numbers, choosing the right ones for a context in a web application becomes extremely difficult.
- Every code review should take into consideration the different constraints associated with the design patterns; otherwise changes to the system could damage the purpose of the existence of the patterns.
- Some design patterns specifically address the needs of experts, while others are targeted at novice; few are suitable for both. Hence, pattern advantages could be reaped only by experienced users.
- It is difficult to apply patterns for a frequently changing web application.

This paper presents a survey of different pattern based web applications. It also analyzes some of the liabilities caused due to inappropriate handling of patterns and a few re-engineering efforts being taken up for improvisation of pattern-based applications.



Fig(1) Pattern Based Web Applications

The organization of the paper is depicted in the schematic diagram in Fig(1). Section 2 discusses the pattern-based web applications in the areas of hypermedia and semantic web as cases for data handling and indexing respectively. Handling of structuring issues of web applications using architectural patterns are also discussed under this section. Section Cooper00 introduces some liabilities due to improper usage of patterns for the web applications. Section 4 analyses various re-engineering methods followed to improve a pattern-based web applications.



2 PATTERN-BASED WEB APPLICATIONS

A web application can be classified into pattern-based and non-pattern based. Proper application of patterns in a web application ensures improvements in architectural reusability and consistency. This is achieved by separating the abstraction from implementation. The next session analyses cases where patterns are used in some of the web applications. A summary of most common patterns that are used in web applications are provided in Appendix A.

Hypermedia

A Hyper Control Document[Chang98] is a multimedia document that might be used to activate an existing stand-alone application with the necessary request received from the users. The request may also be accompanied with some parameters. This phenomenon of the stand-alone application being controlled by a hyper control is known as a Hyper-Controllable Application System[Chang98].

In systems like medical diagnostics, medical experts have to carry out a lot of image processing operations during the diagnosis process. These operations will be carried out by invoking many programs with or without using certain parameters. After successful diagnosis, they would like to retrace the diagnosis steps for similar cases in the same manner. Since, a large number of invocations are involved, it becomes difficult to remember these commands, their sequences and associated parameters for other cases. As a solution to this problem, it is necessary to think of an alternate mechanism of storing the invocations and the parameters used in an external document and one such form is 'hyper controls'. These hyper controls may become a Protocol if they are proven to be consistent across many cases. When these protocols are registered in a convenient form, they promote reusability, accuracy and speed of operations. Hyper-control is one such mechanism of providing an external control for an application without much code changes. When many features of hypermedia applications are included in an existing system, the application is said to have the "Hypermedia Functionality" [Vlissides98]. A hypermedia system has an added advantage of integrating the formal and informal knowledge representations [Schwabe95].

The CAIBCO System [Aluen99] is a web-based object-oriented multimedia medical system that combines web and medical systems for a better interaction by the students and medical staff spread over wide geographical areas.

Gustavo Rossi et al [Rossi97] describe a few patterns that may be useful for the reuse of design in a hypermedia application. An approach has also been described for the development of a pattern language for hypermedia. The pattern systems illustrated here are well suited to work with the hypermedia systems.

DISCOVER [Chang98] is a distributed interactive visualization system that aims at providing an external control to an existing application. Five different DPs are used for the implementation of the application.

Daniel Schwabe et al., [Schwabe95] have suggested the Object Oriented Hypermedia Design Model (OODHM) as a specification for complex navigation patterns and transformation of interfaces. The model provides a systematic methodology for creating hypermedia.

Gustavo Rossi et al., [Rossi97] have introduced two patterns namely, NavigationObserver and NavigationStrategy to address the structuring requirements and controlling the navigation flow of a web application.

Semantic Web

Semantic web has got promising areas for the application of design patterns. Many patterns are employed in construction of semantic web.

Aldo Gangemi [Gangemi05] has suggested a 'database schema' like ontology design pattern to facilitate or improve the technologies used during ontology life cycle. His work highlights many features including building tools that assists development, discussion, retrieval and interchange of conceptual ontology design patterns over the semantic web. Similarly, Nitin Arora et al. [Arora06] have aimed at introducing the information object design pattern for modeling the domain specific knowledge. This will be of a greater importance for any web based application that uses the domain specific knowledge to drive a particular application and to formally describe the content that is being exchanged between any two service-oriented applications.

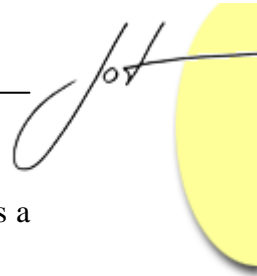
Dietrich J. et al. [Dietrich05] have developed a software design ontology for detecting and documenting DPs found in a software. This work also introduces an effective pattern scanner to detect patterns from a high level language.

In order to become a part of forthcoming semantic web, many attempts are being made to combine semantic web technologies with existing web applications. One such work has been demonstrated by Rován L et al., [Rovan08]. The work provides way of preserving role separation and maintenance of semantic web applications.

The patterns discussed so far are taken up to address specific issues of a web application. Large scale applications essentially have to be started with dividing the entire application into various manageable layers based on some structuring principle. It is essential to facilitate the maintenance and evolution of systems according to the independence of the present classes in each layer. The architectural patterns are proven solutions to address these issues. Two of the architectural patterns with their web applications are given under section 3.

Architectural Solutions for Web applications

Selfa D.M. et al., [Selfa06] have suggested a UML schema for introducing different cases of analysis, design and implementation of a database and web application. This work



does not employ patterns as architectural solutions. However, this could be extended as a pattern-based application by applying suitable patterns.

The architectural patterns are useful in constructing architectures based on a common principle [Buschmann01]. The patterns Presentation Abstraction Control (PAC)[Buschmann01] and Model View Controller (MVC) [Cooper00] are popularly used to provide architectural solutions.

A work by Zdun U [Zdun02] deals with reengineering issues with the help of agents. In this work, PAC has been employed for addressing the structural requirements of the application. The recurring components in reengineering projects are identified and classified to create a reference architecture that can be reused by an application. PAC suffers with more implementation overheads when compared with MVC, discussed below.

The MVC has a wider range of web applications when compared with PAC because of its flexibility of handling.

Bodhuin T et al.,[Bodhuin02] present a strategy using MVC for migrating a legacy COBOL system into a web enabled architecture. The needed information from the COBOL source code is extracted and then wrapper classes are applied to convert them into Java Server Pages.

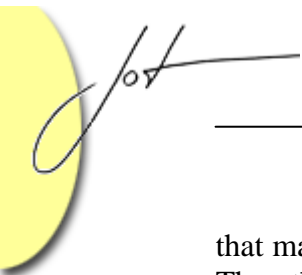
Yu Ping et al.[Ping03] have provided a methodology by which the database functionalities are extracted from the source programs of a legacy web application to form JavaBean objects. The legacy presentation components are translated into JSP pages that are made to refer the JavaBean objects and the linkage informations are extracted and web site is made as controller-centric. This work is being extended to incorporate the language independent feature in the source program of the translation process.

Hao Han et al., [Han08] have showcased a recent revolution in the web application scenario for the integration of web services. This concept has been handled in different names such as mash ups [Cetin07], mix-up or service oriented architecture. They have further extended the phenomenon of integration of websites that even do not provide web services by generating the virtual web service functions at client side.

3 LIABILITIES DUE TO IMPROPER USAGE OF PATTERNS FOR WEB APPLICATIONS

As discussed above, applying patterns without proper experience may introduce several overheads in an application. This session deals with some of them caused due to improper usage in web applications.

In the DISCOVER application [Chang98], the patterns Command Processor, Memento, Document-View, Visitor and Singleton are all designed to be inter-dependent. In other words, it improves tight coupling between the classes defined by the patterns, which is not desirable in pattern-oriented programming. Similarly, the command processor is duplicated for independent handling of dialog codes and computation codes



that may result in inconsistency because of the increased generation of command objects. The other side effect may be due to excessive number of command objects generated by the pattern that may result in efficiency loss. Similarly, Visitor pattern employed in the application is less helpful during growth stage since every new class getting added in the Visitor's list has to update entries in the abstract Visitor class and concrete Visitor class [Cooper00]. Hence, Visitor is inflexible when frequent changes are possible in an application. It is believed that, Memento may not be suitable for image processing applications as the amount of data that a Memento must save might be quite large and taking up a fair amount of storage space [Cooper00].

Even after the application of patterns, some applications may suffer due to the problem of insufficiency. For instance, the work illustrated by Gutuva Rossi[Rossi97] does not take into account all the problems related to navigations and hence not considered as a complete solution. Similarly Aldo Gangemi's work [Gangemi05] is insufficient without the categorization of patterns and a suitable matching algorithm.

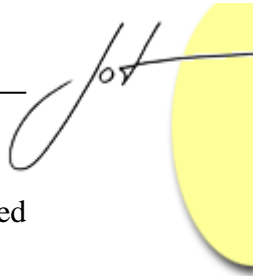
The third category of problems is due to web partitioning issues. This problem arises when web applications are fitted in MVC architecture. The View part suggested by the MVC pattern is generally displayed on the client side, since it deals with the rendering of output. The model and the controller can be partitioned in many ways between the client and the server. This partitioning has to be done at the design phase itself. In contrast to this, MVC, by design is meant to work in single address space and hence it is partition-independent. Hence, implementing a web application in a client-server environment raises the issue of partitioning, which is against the nature of the MVC design pattern.

This issue has been addressed by Avraham Leff et al[Leff01], by introducing dual MVC (dmvc) flavor of flexible web application partitioning (fwap). Here, the model and the controller components of a web application are made to reside at both client and server. In other words, the model and controller components are duplicated at server and client sides. This approach lacks security as it is not desirable to keep the model at the client machine. Moreover, this also suffers from the problem of inconsistency as more than one instance of a model or controller exists at any point of time.

4. RE-ENGINEERING WEB APPLICATIONS

Wendorff P.[Wendorff01] has shared his experience in addressing the problem of maintenance caused due to the usage of uncontrolled number of patterns in an application. In his re-engineering process, many inappropriately applied patterns are identified and when an attempt is made to remove them, it is observed to be not viable due to the fact that many of them are found to be tightly coupled with the other parts of the application.

The re-engineering using cloned pattern analysis has been suggested by De Lucia.A et al.,[Lucia04] to discover similar patterns of code as well as the navigational patterns of a web application. This is proposed as a re-engineering solution for web applications that have not been developed in a systematic way.



Zdun U[Zdun02] locates recurring components of a re-engineering project and tried to integrate them conceptually to a reference architecture.

5. CONCLUSION AND FUTURE WORK

The survey presented in this paper has analyzed different pattern-based web applications for data handling, organizing and structuring issues. The paper has also analyzed the effects on web applications due to improper usage of patterns and outlined a few re-engineering solutions. The future work of the authors involve presenting an architecture that will be useful for web applications to be ported into MVC framework without becoming entangled into the partition decision issues.

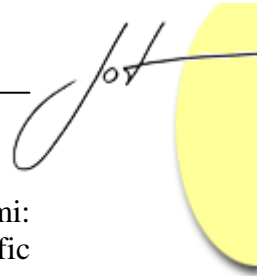
APPENDIX A.

<i>S.No</i>	<i>Pattern Name</i>	<i>Applicability</i>	<i>Consequences</i>	<i>Remark</i>
1	Command Processor	<ul style="list-style-type: none">• Separates request for a service from its execution.• Ideal for the development of hyper-controllable applications.	<ul style="list-style-type: none">• This pattern provides flexibility in handling requests and their functionalities.• It also allows commands to be executed in separate threads of control.• Implementation of indirections costs storage and time thereby leading to efficiency loss.	A reengineering strategy with Memento, Observer, Visitor and Singleton may replace the functionality of Command Processor.
2.	Document View	<ul style="list-style-type: none">• Document holds the core functionality and data, View combines the 'View' and 'Presentation above'	<ul style="list-style-type: none">• Suited for 2-Tier Client-Server architecture	---
3.	Document-View-Presentation	<ul style="list-style-type: none">• The Document holds the core functionality and the data; view to manage the display (render + accept service request) and Presentation deals with output and user input	<ul style="list-style-type: none">• Reuse the rendering output• Pluggable presentation component• Thin user interface• Ideal for multiple window based applications• Increased Complexity	---
4.	Model/View/Controller	<ul style="list-style-type: none">• Dividing an application into functionality(model), display(view) and user input(controller)	<ul style="list-style-type: none">• Easy maintenance of multiple views of the same model.• Synchronized and 'pluggable' views..• Uncontrollable number of updates.• Close coupling between views and controllers	The model can be made to skip unnecessary updates.
5.	Navigation Strategy	<ul style="list-style-type: none">• Ideal to apply when there is a need to establish a relation between two or more objects	<ul style="list-style-type: none">• The pattern encourages dynamic creation and linking of nodes in an	The Prototype pattern can be used to overcome the barriers of the

S.No	Pattern Name	Applicability	Consequences	Remark
		<ul style="list-style-type: none"> at different times. Used in situations where objects stored in a database are to be retrieved whenever an associated object raises a demand. 	<ul style="list-style-type: none"> active hypermedia environment. Allows one to define different kinds of links as well as end points. Used to improve memory requirements by deferring the retrieval of the target code only when needed. Increased number of objects and communication overhead between the classes involved. 	pattern.
6.	NavigationObserver	<ul style="list-style-type: none"> Maintenance of navigation history Maintenance of different viewers for the history Enabling the backtracking in the navigational path. 	<ul style="list-style-type: none"> Decouples navigation from its history and history from the display of it. Provides application independent functionality for the style of viewing the history. Causes overhead when attempts are made to filter certain types of nodes in the history. 	A reengineering effort made by introducing an alternate architecture involving singleton or mediator.
7	Presentation/ Abstraction / Control	<ul style="list-style-type: none"> Defines a structure in the form of levels of cooperative agents. Each agent is divided in to Presentation, Abstraction and Control components 	<ul style="list-style-type: none"> New agents can easily be added / dropped at any time. Easy implementation of multi tasking Increased system complexity 	Provides a maintainable and extensible structure with clear separation of concepts between different system tasks.

REFERENCES

- [Buschmann01] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal:*Pattern-Oriented Software Architecture*, Volume 1, John Wiley & Sons (Asia) Pte Ltd, 2001.
- [Cooper00] James W. Cooper:*Java Design Patterns*, First Edition, Pearson Education, New Delhi.
- [Gamma95] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides: *Design Patterns:Elements of Reusable Object-Oriented Software*, Second Edition, Pearson Education, New Delhi.
- [Vlissidess98] John M. Vlissidess, James O. Coplien, and Norman L. Kerth:*Pattern Languages of Program Design*, Addison-Wesley, 4th printing, August 1998.
- [Aluen99] Miguel Aluen, Hector Arrechendera, Alfredo Matteo, and Christiane Metzner: “Developing a Web-based Object-Oriented Multimedia Medical System”, *32nd Hawaii International Conference on System Sciences*, 1999, pp 7026.



-
- [Arora06] Nitin Arora, Rupert Westenthaler, Wernher Behrendt, and Aldo Gangemi: “Information Object Design Pattern for Modeling Domain Specific Knowledge”, *1st ECOOP Workshop on Domain-Specific Program Development (DSPD) in conjunction with ECOOP, 2006*.
- [Bodhuin02] T. Bodhuin, E. Guardabascio, and M. Tortorella: “Migrating COBOL systems to the Web by using the MVC design pattern”, *Ninth Working Conference on Reverse Engineering, 2002*, pp 329-338.
- [Cetin07] S. Cetin, N. Ilker Altintas, H. Oguztuzun, A.H. Dogru, O. Tufekci, and S. Suloglu: “Legacy Migration to Service-Oriented Computing with Mashups”, *International Conference on Software Engineering Advances, ICSEA 2007*, pp 21-21.
- [Chang98] Ku-Yaw Chang, and Lih-Shyang Chen: “Using Design Patterns to Develop a Hyper-Controllable Medical Image Application”, *PLoP, 1998*.
- [Dietrich05] J.Dietrich, and C.Elgar: “A formal description of design patterns using OWL”, *Australian Software Engineering Conference, 2005*, pp 243- 250.
- [Gangemi05] Aldo Gangemi : “Ontology Design Patterns for Semantic Web Content”, SpringerLink, pp 262-276, 2005
- [Han08] Hao Han, and T.Tokuda: “A Method for Integration of Web Applications Based on Information Extraction”, *Eighth International Conference on Web Engineering, ICWE apos08, 2008*, pp 189-195.
- [Leff01] Avraham Leff and James T. Rayfield: “Web-Application Development Using the ModelNiewlController Design Pattern”, *Fifth IEEE International Conference on Enterprise Distributed Object Computing, 2001*, pp 118-127.
- [Lucia04] A. De Lucia, R. Francese, G. Scanniello, and G.Tortora: “Reengineering Web applications based on cloned pattern analysis”, *12th IEEE International Workshop on Program Comprehension, 2004*, pp 132-141.
- [Ping03] Yu Ping, Kostas Kontogiannis, and terrence C. Lau: “Transforming Legacy Web Applications to the MVC Architecture”, *Eleventh Annual International Workshop on Software Technology and Engineering Practice, IEEE, 2003*, pp 133-142.
- [Rossi97] Gustava Rossi, Daniel Schwabe, and Alejandra Garrido: “Design Reuse in Hypermedia Applications Development”, *Proceedings of the eight ACM conference on HYPERTEXT , 1997*.
- [Rovan08] L.Rovan, and I.Nizetic: “Extending Web Applications for Semantic Web”, *30th International Conference on Information Technology Interfaces, 2008*.
- [Schwabe95] Daniel Schwabe, and Gustavo Rossi: “The object-oriented hypermedia design model”, *Communications of the ACM, v.38 n.8, 1995*, pp.45-46

- [Selfa06] D.M.Selfa, M.Carrillo, and M. Del Rocio Boone: “A Database and Web Application Based on MVC Architecture”, *16th International Conference on Electronics, Communications and Computers, IEEE Computer Society*, 2006, pp 48.
- [Wendorff01] Wendorff.P: “Assessment of design patterns during software reengineering: lessons learned from a large commercial project”, *Fifth European Conference on Software Maintenance and Reengineering*, 2001, pp 77-84.
- [Zdun02] Zdun. U: “Reengineering to the Web: a reference architecture”, *Sixth European Conference on Software Maintenance and Reengineering*, 2002, pp 164-173.

About the authors



Sridaran Rajagopal is the Campus Head of ICFAI National College, Vellore, India. His research interests are Design Patterns and Software Engineering. Contact him at sridaran.rajagopal@gmail.com.



Padmavathi Ganapathi is the Professor and Head, Department of Computer Science of Avinashilingam University for Women, Coimbatore, India. She has 50 publications at National and International level. Her research interests are Computer Networks and Genetic Algorithms. Contact her at ganapathi.padmavathi@gmail.com.



Iyakutti Kombiah is a Senior Professor of School of Physics of Madurai Kamaraj University, Madurai, India. His research interests are Computational Physics and Software Engineering. Contact him at iyakutti@yahoo.co.in