Influencing Web Development Practice: Understanding Through Clarity

David McCurdy

Media Design School
Auckland, New Zealand

davidm@mediadesign.school.nz

ABSTRACT

It is often difficult for web developers to understand the role of systems analysis tools in the development of web-based projects. Sometimes there is a misunderstanding that web-development is different. Sometimes this means that the web-developer ignores the tried and tested systems analysis tools of old and uses an in-house set of tools, at best. Often, the infinite code fix method is used. Many tried and tested systems analysis and design tools can be used to aid web-development. The purpose of this paper is to highlight the usefulness of several systems analysis and design tools from both Object Orientated and Structured Paradigms, within the framework of the n-tier architecture.

Keywords: Object Orientation, Structured Analysis, Web Development, n-Tier Architecture

1. INTRODUCTION

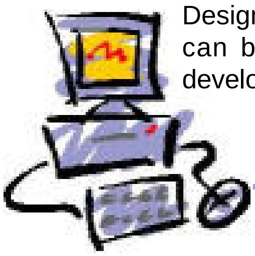
The purpose of this paper is to show that OOAD (Object Orientated Analysis & Design) and SSAD (Structured Systems Analysis & Design) are useful for web development, and can be used in unison for speedy web-development.

The reason the author has decided to write this paper is that many students, teachers and practitioners have asked many questions:

1. What has the DFD (Data Flow Diagram) got to do with web development?
2. How do I code my web-pages from the class diagrams?
3. Coding objects from a relational database, what? How does that work?
4. Shouldn't you just use tools from one paradigm?
5. Why do many web projects get caught in the code freeze, even before coding has begun?
6. Why are you teaching VBScript classes when the objects are not persistent?
7. Should an n-tier based architecture be the foundation of all web-applications?
8. Should development be driven by the look and feel of the final application or from the database?

These questions lead to other questions and generate many lengthy and passionate debates. Some answers to these questions will be provided in this paper. For clarity, the author will define the terminology used; so that ambiguity can be avoided.

This paper draws on the usefulness of many systems analysis tools and highlights their potential role in web-development. In addition, the relationships that exist amongst the tools will be described.



The n-tier model will be described and the usefulness of each systems analysis and design tool will be identified.

1.1 THESIS AND ANITHESIS: THE SYNTHESIS

The thesis of this paper is: web development practice can be made simpler, easy to manage, through greater clarity and understanding of the role of the analysis and design tools in analysis, design and development. Using tools from both paradigms (OO and Structured). The role and relationship of the tools within the n-tier architecture provides the framework to express the thesis.

The antithesis of this paper is: mixing paradigms is inappropriate, deviates from standard practice, deviates from doctrine. Does the n-tier model provide an appropriate framework for discussion.

The synthesis of this paper is: development and analysis are real tasks of the web-developer; if anything this can make development easier and simpler, then it must be good. General opinion of most web-developers is that the n-tier model of web-development is the best model; albeit that it is not used much in industry; however, an n-tier model is downward compatible to its three and two tier cousins; upwards compatibility is not possible; that is why that framework was selected.

1.2 OBJECT ORIENTATED ANALYSIS AND DESIGN

Classes are the key to any Object Orientation Development. Although other models do make the development of projects using OO. Object Orientation has the benefits of greater code reuse, greater scalability, and greater maintainability; if it is measured against its procedural or functional counterpart. The class diagram is very useful for web-development especially if developed hand-in hand with the ERD (Entity Relationship Diagram). Developing both the ERD and Class Diagram hand-in-hand, can make web-development easy.

1.3 STRUCTURE SYSTEMS ANALYSIS AND DESIGN

The corner stone of many transactional-based web-applications is the database behind the scenes. These databases are predominantly relational databases. The Entity Relationship Diagram is useful for the design of the database. Dataflow diagrams are being used less and less in industry; however, the usefulness of the DFD will be transparent later.

1.4 TIERED ARCHITECTURES

Often web development projects are developed using tiered systems. The typical web-developer has used two-tier architecture. This is a client/server-based architecture in which only the client and server are involved in the transactions over the Internet. This usually manifests itself as Web Server to HTML Pages. The three-tier model extends this model by adding applications and their associated databases to the model, these will usually supply none mark-up language information to the Web-Server on request. In the n-tier model, more separation is made: Database Layer, Data Access Layer, Business Logic Layer, Presentation Logic Layer, and Presentation Layer. This interpretation of the n-tier model is what this paper refers to as the n-tier model.

1.5 DATABASE LAYER, SIMPLY IS THE DATABASE SCHEMA AND STORED PROCEDURES (IF ANY)

Data Access layer, is simply the connections to the database and execution orders of the database commands or stored procedures.

Business logic layer, is the abstraction of these database objects, connections and execution orders into business units, for example the customer object is composed of its properties and methods that have a layer of abstraction over the data.

Presentation logic layer, is the production of what will be displayed on the screen for the user to interact with the system. The presentation logic is the what not how of presentation.

Presentation layer, finally, the presentation layer involves the how of presentation: graphics, graphic art work, sound, etc.

The problem with some web-development projects is that the web-designer (with minimal education is database design and computing) does some of those tasks: this can result in a poor quality system functionally. On the other hand, the web-developer (with little or no creative flair or graphic art knowledge) does some of those tasks: this can also result in a low grade graphical front-end system.

1.6 TEMPORAL PARADIGM

It is often said that one should select a paradigm and do everything in that paradigm. That is often correct and it is sometimes correct in web-development. Most transactional-based web applications use a relational database and the data-access tier often uses procedural or functional based coding; however, when the data-access layer is written using OO technology when com-objects are used or the development technology is Java. Mixing these paradigms, as is often the case in the web-development environment, means that the tools of either paradigm in isolation do not provide enough information to aid the developer.

Most web-development projects use a relational database as the backend data-store; the tool of choice for most relational database designers and developers is the Entity Model.

2. SYSTEMS DEVELOPMENT LIFECYCLE

2.1 EVOLUTIONARY PROTOTYPING

Many systems development lifecycles exist and many have been applied to web-development. One common approach in industry is to use evolutionary prototyping. Evolutionary delivery has many pitfalls in web-development, McConnell Describes these as: unrealistic schedule and budget expectations; inefficient use of prototyping time, unrealistic performance expectations, poor design and poor maintainability. However, the evolutionary prototyping has some major benefits, that is: greater progress visibility from greater customer and user feedback. Evolutionary prototyping is fraught with danger, other SDLC tools can be used to reduce this risk. Evolutionary Prototyping was the preferred SDLC of 1/18 junior developers studied.

One way of improving the Evolutionary Prototyping model is to use the Evolutionary Delivery lifecycle using a throwaway prototype. Many problems exist when web-development is driven using evolutionary prototyping. The prototype can eventually become the final product; this often leads to a software solution with much feature creep, lacks maintainability, and is not extensible. Integrating an evolved prototype into an n-tier model is inefficient in a number of different ways. This causes numerous overly complex problems for the web-developer.

It is often thought that prototyping leads to a quality solution; in some cases this is correct. Its highlighted by McConnell (1996), that prototyping does not guarantee high-quality end-user or customer feedback. End users and customers don't always know what they are looking at when they are presented with a prototype. Customers and users can become so overwhelmed by the live software demonstration that they do not look beyond the glitz and glamour to understand the material that is presented to them. It is no wonder that many web-development projects deliver the wrong solution.

The product developed using the evolving prototyping model suffers from a number of problems:

1. No consideration given to the performance in the product design;
2. Keeping poorly structured code that was developed quickly for the prototype;
3. Keeping inefficient code that was developed quickly for he prototype; and
4. Keeping a prototype that was developed as a throwaway.

Actions can be taken that remove these problems. The greatest solution is to use Evolutionary Prototyping with other systems development lifecycles, like Evolutionary Delivery or Staged Delivery.

2.2 STAGED DELIVERY

Staged Delivery involves the development of a project in stages. A good example of this would be a typical database driven website, that requires content, navigation and form to be derived from a database. For both users and administrators of the websites. In staged delivery the administration of the site would be implemented first and presented to the users that would fall into that category. The 'web' browsing portion for standard users would be developed in the second instance. Many sub-systems can exist in web-projects, these for smaller sub-projects. The problems recognised in this lifecycle model is feature creep. Many benefits exist for developing web-projects using the staged delivery model. It was the preferred SDLC method for 16/18 junior web-developers in 2001.

2.3 EVOLUTIONARY DELIVERY

Evolutionary Delivery is a variation of the Evolutionary Prototyping and Staged Delivery as described above. The product is developed in versions; the user is presented with the current version, feedback is acquired, the feedback is used to further develop the project and the cycle continues until the customer is happy. McConnell describes several pitfalls of Evolutionary Delivery: it promotes feature creep, project control diminishes, inefficient use of developers time, and promotes unrealistic schedule and budget expectations.

2.4 CODE-AND-FIX

Often, the preferred web-development strategy is the infinite Code-and-Fix lifecycle. This was the preferred SDLC of 1/18 junior developers studied. Described by McConnell: is seldom useful. The fact is, however, that many web-developers use the code-fix SDLC. Code-and-fix can be used to good effect in a small way, that is for developing throwaway prototypes. Code-and-fix has no overhead, time is not expended in planning, documentation and quality assurance. This is excellent for throwaway prototypes or proof-of-concept projects.

3. PROPOSED SDLC

The proposed SDLC for web-development requires explanation from several alternative perspectives.

The proposed SDLC for web-development is the staged delivery model as described earlier in this paper, with some slight alteration.

The systems analysis artefacts produced at each stage are:

- ♦ Software Concept [Project Terms of Reference, Interviews]
- ♦ Requirements Analysis [Entity Relationship Model, Use-Case Model (users)]
- ♦ Architectural Design [Class Diagrams]
- ♦ Stage 1: Detailed Design, develop, debug, code, test and deliver
- ♦ Stage 2: Detailed Design, develop, debug, code, test and deliver
- ♦ Stage 3: Detailed Design, develop, debug, code, test and deliver
- ♦ [Class Diagram, ERD, Use-Case Model (users), Use-Case Model (developers)]

The systems analysis artefacts are related to and drive the development in the following tiers of the n-tier architecture described in this paper:

Presentation:	None (assumed to require graphic artist)
Presentation Logic:	Site Map Use-Case Model (user driven)
Business Logic:	Class Diagram
Data Access:	Class Diagram
Data Base Layer (Stored Procedures):	Use-Case Model (developer driven)
Data Base Layer (Schema)	Entity Relationship Model

There are many benefits developing the systems from the reverse direction:

1. Database Layer (Schema/Stored Procedures)
2. Data Access
3. Business Logic
4. Presentation Logic
5. Presentation.

Those benefits include:

1. Strong functional flexibility (for users, if requirements alter)
2. Good data storage and retrieval flexibility
3. Excellent code reuse
4. Excellent maintainability
5. Good extensibility.

4. CONCLUSION

Object Orientated Systems Analysis and Design is a useful tool for web development, however, Structured Systems Analysis and Design tools can be used to compliment and add clarity. The Staged Delivery coupled with throwaway prototypes are useful. The n-tier development architecture leads to greater code reuse and greater control over the web-development project.

REFERENCES

Chobe, P. (1998). Web Based Development Using N-tier Technologies. Procedings of the 4th AIS Conference, Baltimore: MY, August 14-16, 1998.

Clark, S. et al (1999). VBScript Programmers Reference. Wrox: Birmingham.

McConnell, S. (1996). Rapid Development. Microsoft Press: Washington.

Philip, G. (1999). Software Design Issues in Web-based Development. Procedings of the 5th AIS (AMCIS) Conference, Milwaukee: WI, August 13-15, 1999.

Schneider G. & Perry, J. (2000). Electronic Commerce. Course-Technology: Cambridge.

Sutherland, J. (2002). The Object Technology Architecture:Business Objects for Corporate Information Systems. Website: URL: http://jeffsutherland.com/papers/boa_pap.html Accessed May 2002

Trilogy Development: Website: N-Tier Architecture URL: http://www.trilogycomputing.com/ntier.htm Accessed May 2002.