# TEACHING OBJECT ORIENTED SYSTEMS ANALYSIS AND DESIGN: A COURSE MODEL

**Dr. Roy A. Boggs, Florida Gulf Coast University, RBoggs@fgcu.edu**

## ABSTRACT

*Information Systems pedagogy has been marked more by change than by constancy. This is no truer than in presenting the differences between the traditional linear Systems Development Life Cycle and a new Object Oriented Systems Development Life Cycle. There are stark differences between the two; and the question quickly emerges of how to present these differences in a simple and easy to understand manner. One possibility in the form of a model is given with reference to an actual application.*

**Keywords**: Systems Development Life Cycle, SDLC, Object Oriented Systems Analysis and Design, OO_SAD, Systems Analysis and Design

## INTRODUCTION

### Structured Projects

The many current versions of the Systems Development Life Cycle (SDLC) are in some form a reworking of the traditional waterfall diagram. Beginning with the structured waterfall model and proceeding through the Incremental, the Spiral, the Next-Generation, the Win-Win, the V- and W-models, and the WEIT model a common thread of increasing cost/performance management and user involvement can be traced [1, 2, 7, 11]. To these can be added the many proprietary SDLCs, [12] as well as partial SDLCs such as Six Sigma [3]. Each of the new SDLCs has given the systems analyst an improved road map for developing and managing information systems projects. In many instances these projects are quite challenging and resource intensive. Indeed, there have been well-documented, notable failures, but overall, a great many demanding projects have been and continue to be successfully implemented.

When making a presentation or comparison of these SDLC models, each individual model, however labeled, can be reduced to four basic steps: Discover, Design, Develop, and Deliver. For project application, the individual road maps are much the same: 1) Find out for sure where you have to go, 2) Know where you are, and then the challenge is 3) How to get from here to there. This process has been tested by experience and works extremely well for highly structured projects. One might call them ‘engineering’ projects, like building a parking lot from scratch: clear the land before laying drainage, etc. The final configuration is known and approved before actual work begins. Just follow the steps along the SDLC road map.

While current systems textbooks quite often present the course SDLC model without serious discussion, sometimes no more than several paragraphs, an examination of chapter sequencing will reveal the traditional incremental, linear SDLC model. For newer (Internet) and less easily definable projects, such structures can be misleading. For example, output design usually falls at about chapter seven, even though the user might need to experience the output and provide

feedback much earlier in the process. This is the reason why there are so many different (linear) SDLC models from which to choose, and why more serious discussions of the SDLC are needed. To this, is now added an object oriented SDLC.

**Unstructured Projects**

However, there are information systems projects, which are not so linear or so highly structured. These are unstructured ('fuzzy') projects. The journey begins before the final destination can be fully defined. One only knows *about* where one needs to go. One might call these 'social' projects. The project suggests entities, but individual occurrences all appear with different, yet to be defined characteristics. One begins the journey, defines a first instance with its characteristics, and then adds characteristics as they evolve. Only at the end of a defined stage does the project have a discernable (if still fuzzy) form. This process is new, from the bottom up. This is the point where one finds a new SDLC format - Object Oriented Systems Analysis and Design (OO_SAD) – that cannot only handle structured but also unstructured projects.

Like its structured counter parts, OO_SAD has four basic steps and follows a road map. The basic steps are not new: Discover (Inception), Design (Elaboration), Develop (Construction), and Deliver (Transition). These steps can be divided and renamed as desired. For pedagogical purposes it is here useful to begin with common terms and then delineate differences, as once upon a time with 'rows' and 'tuples'. There is in reality little that is new here,

However, the basic road map itself is now quite different. It is not linear and it cannot always be progressed with traditional measurement tools. In fact, there is not yet an acceptable description of what an OO_SAD road map would look like and how it might best be depicted. They are no really useful suggestions how a road map might be best presented, especially to students who have already experienced traditional SDLCs. This is the task of the following.

## OBJECT ORIENTED SYSTEM ANALYSIS AND DESIGN: DEVELOPMENT OF A MODEL

For anyone who has lived in a Compile, Link, and Go (CLG) applications environment, object oriented programming is not new. Link libraries and similar structures have been from the beginning part of application languages such as Fortran and Cobol. However, unstructured projects have long been and continue to be a problem; and OO_SAD offers something new for systems analysis and design. But first it must be understood. For classroom purposes, the OO_SAD process must not only be easy to understand, it also needs to be visible. Pedagogical theory holds this as a tenet. All SDLCs present a basic, visible model.

Exactly how does a systems analyst go about preparing for a fuzzy project and how does one present this process? The answer is *from the bottom up*; and in presenting the process, it is first helpful just to walk though the process.

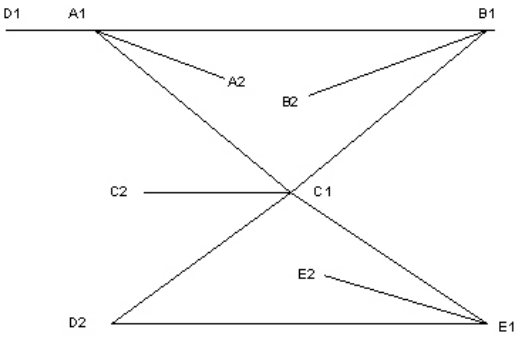
**Structured vs. Unstructured Applications**

First, two examples will serve to differentiate structured from unstructured projects. The examples are well documented, both were quite large, and both were undertaken in the same

house [4]. The first was well structured and was very successful. Extremely minute data for all of a state's highways, cubic centimeter by cubic centimeter, were to be entered into a common database and tracking algorithms developed, with appropriate management reports. As new data were collected, daily operational routines were to be executed. For this project, the traditional linear SDLC proved effective. In the beginning, the final destination was quite clear.

The second project was unstructured and thrashed its way from failure to failure. The project was to create a statewide tracking system for dependent families and children, supporting all state and federal guidelines. Because the state was so large, there was no one family and child entity, and area demographics were so diffuse that is was impossible even to know what an expected entity might look like, let alone from where it might come. Objects oriented *programming* processes were used, and an end to the journey was envisioned but the road map itself was weak. There was no effective SDLC.

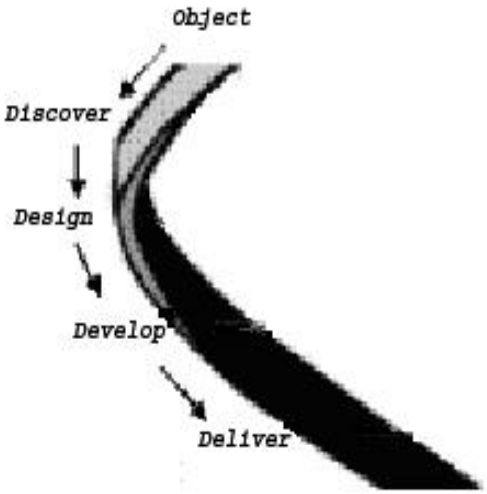
**The OO_SAD Model: A Walk-Through**

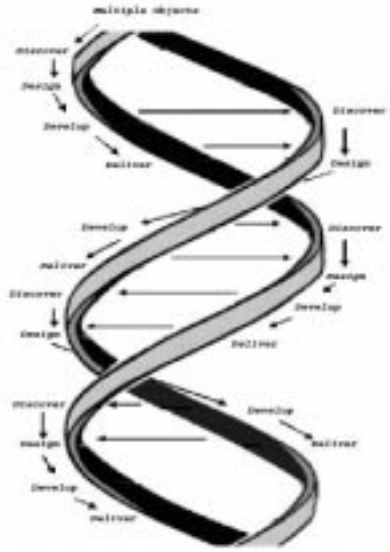



Using the family and child project described above, instead of attempting the traditional, linear SDLC, one might divide the project into steps and conquer one unit at a time. The state is divided into separate units, each based upon unique social, economic, and demographic factors. The organization of the project will begin at A1 and progress in successive steps, each step under cost and management control. While the project has an overall, estimated time and cost chart, these will be modified as each unit is completed.

A1 – professional / academic
A2 – academic
B1 – professional / industrial
B2 – industrial
C1 – professional / academic
C2 – professional

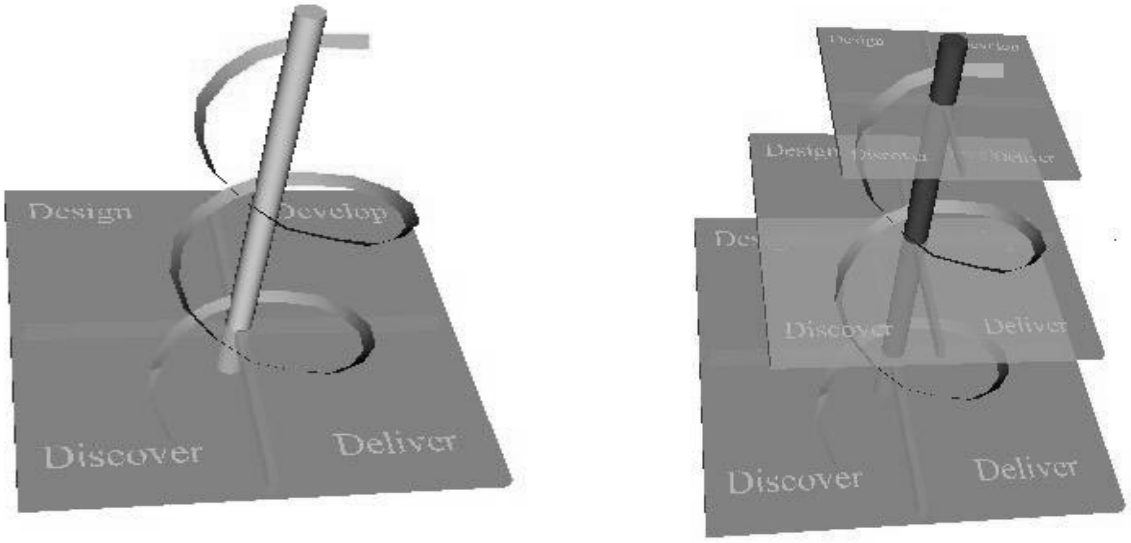
D1 – tourism
D2 – tourism
E1 – tourism / professional / international
E2 – agricultural / international



The project begins with a function(s) (use case) and its users (actors/agents) for A1, and develops classes (functionalities). To this is then added, as successful development permits, additional functions (use cases), first for professional, and then for academic cases. During this process shared functionalities and relationships will become apparent.

The image on the left shows the first case. The image on the right shows how cases share objects as functionalities are developed. The number of strands in the helix will grow as cases are added; but each new case will share objects that have already been developed.



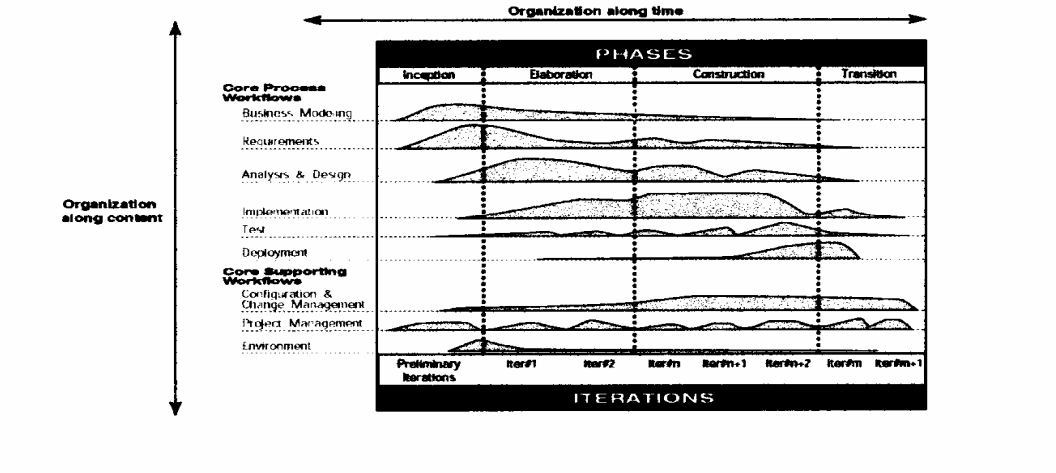

The interrelated cases grow from a platform of Discovery, Design, Develop, and Deliver. As they expand, new and shared functionalities permit the project to be assembled, piece-by-piece, under carefully controlled conditions. If the cases are well developed, it will be possible to understand the various functionalities and relationships. Each platform represents the progress from case to case. The image on the left can represent one case or one stage. It can represent the first case or A1 (above). The image on the right can represent the first set of cases or A1, A2, B1 etc.

From A1 (professional / academic) it will then be possible to migrate with some modifications to the more academic cases to A2 (academic) and to the more professional B1 (professional) with few changes, etc. Then each platform represents, rather than cases, stages in the project; and the project can be carefully managed. In the same manner, the model can be used to develop other aspects such as virtual data objects and peripheral designation.

The second, unstructured project referenced above, may well have had its major flaw in that an overall solution was attempted right from the beginning. This led to large purchases of hardware that were never used and extremely poor user feedback. An OO_SAD, based upon a model similar to the one here might have literally saved the state millions of dollars. It is the beauty of OO_SAD for very fuzzy projects that the project can progress, with cost and time considerations tightly controlled. Something that was lacking in this project and something that made for some rather distressing statewide media attention [9].

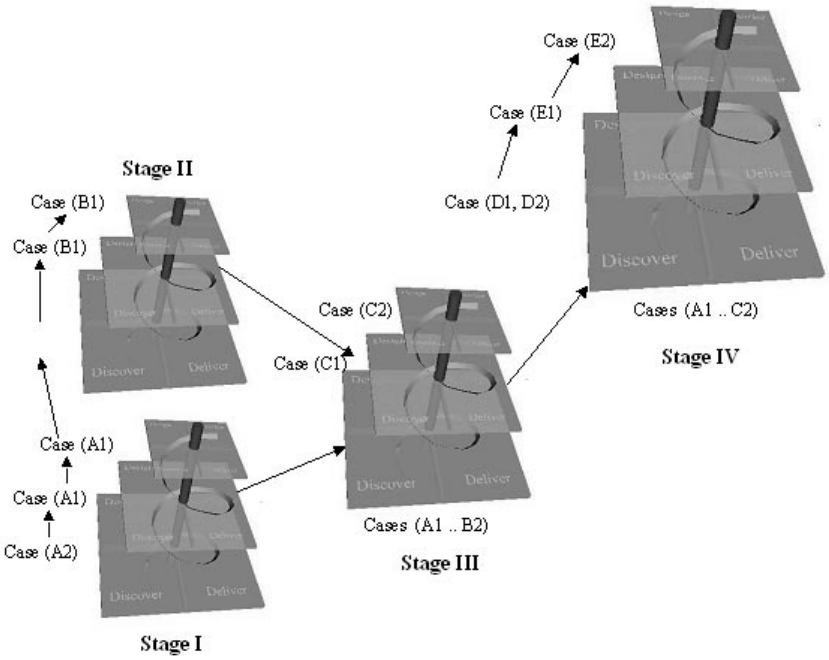
## THE OO_SAD MODEL: THE NEW vs. THE OLD

The common OO_SAD model that is almost always referenced in the literature is not as clear for making an initial pedagogical presentation [5, 8, 10]. While the pieces are all there, it does not serve as a model for explaining object oriented analysis and design.




It is not that the model is false or misleading, the question is – what can one do with this model while making an initial presentation. It is important that the audience understand correctly in the beginning how a project is analyzed and designed from the bottom up, case by case. Object oriented programming does not automatically lead to object oriented systems analysis and design. There needs to be a road map with a useful model.

A clearer view of the entire project can be seen in the following image. There is a clear step-by-step process, from the bottom up, from case to case, from functionality to functionality, and from stage to stage. The model is simple, it is visual, and can serve to illustrate all the various levels. From the very beginning students gain an understanding of OO_SAD; and discussions of cost management and user involvement is clear.




## CONCLUSION

Whatever has been said here, traditional SDLCs cannot be abandoned. It is not the purpose of the above to lessen the importance of understanding the linear SDLCs. To do this, would be to deny reality. Students need to know and understand them – they are what is out there. But for a complete presentation, OO_SAD cannot be left out and its importance will grow very quickly.

## REFERENCES

1. Boehm, B. W., Egyed, A. Kwan, J. Port, D. Shah, A. & Madachy, R. J. (1998). Using the WinWin Spiral Model: A Case Study. *IEEE Computer*, 31(7), 33-44.
2. Boehm, B. W., & Bose, P. (1994). "A Collaborative Spiral Software Process Model Based on Theory W." *Proceedings, 3rd International Conference on the Software Process, Applying the Software Process* (1994).
3. Chowdhury, S. (2002). *Design for Six Sigma. The Revolutionary Process for Achieving Extraordinary Profits*. Trade Publishing.
4. Department of Children and Families. *HomeSafenet Information Technology Audit*. By William O. Monroe, General Auditor (2002).
5. Jacobson , I., Booch, G. & Rumbaugh, J. (1999) *The Unified Software Development Process.* Addison-Wesley.
6. Object Management Group (OMG). *UML Summary*. (September 2001). http://www.dsi.uniroma1.it/~parisi/01-09-72.pdf

7. Projekt WEIT. *Weiterentwicklung des Entwicklungsstandards fur IT-Systeme des Bundes auf Basis des V-Modell-97*. (2003
8. Rational Software White Paper TP026B, Rev 11/01. *Rational Unified Process: Best Practices for Software Development Teams.* Rational Software Corporation, 1998.
9. "Records Project Makes Agency Look Foolish." *The Fort Myers News Press*. 29 Nov. 2001. 8b.
10. The Rational Unified Process: A well-documented, complete yet complex methodology. By Thomas Meloche. Menlo Institute. Pg 1
11. The Federal Government Co-ordination and Advisory Agency for IT in the Federal Administration (KBSt). *Das V-Modell - Chance zum maschinellen Workflow-Management*. (2003)
12. Thome. R. & A. Hufgard. *Was is eine Softwarebibliothek?: Continuous System Engineering*. Germany: Vogel, 1996.