**Systems Design and Analysis**

Design and Analysis (also known as Software Engineering) is the discipline that addresses the question of how to develop software in the real world.  In the real world, software is developed by a team that is part of department that is developing software for a different group of people: the customers.

As a software professional, everything — even the most seemingly-pure technical tasks — you do is impacted by the fact that you are working with one set of people, that the systems will be used by a different group of people, and that the two groups — the builders and the customers — are profoundly, immensely, different.

As a result, being successful as a professional requires human and organization knowledge and skills, as well as technical knowledge and skills.  These two sets of knowledge and skills and their various interactions are the focus of the course work in this area.

Everyone who has studied computer science is familiar with the fact that in analyzing and working with a software system of any complexity, it is critical to be precise and systematic, and to have a good framework for representing the facts about the system. There are innumerable details to keep track of and many decisions to make as a designer or programmer. Forgetting or mis-handling any of them can lead to serious problems in the code.  Intuition and hand-waving are not sufficient, which is why the fields of analysis and design, whether function- or object-oriented, exist.  The same is true of analyzing and working with customers and their organizations: there are innumerable details, any of which can lead to serious problems or complete failure of the system or the project.  A significant portion of the course work in this area is devoted to expanding the students’ abilities to work with, understand, and describe client organizations and work, so that they can approach problems of working with clients with the same rigorous, systematic approach that is applied to the design and construction of software.

Examples of specific concepts and topics addressed in depth in this area include:

* Describing the client organization — not just the data processes and objects — completely and in detail, including all the organizational processes, from the top down to specific processes in which the system is used and the organizational priorities and principles that govern the execution and choices of all those processes.
* Getting good information from customers, who quite often have little idea what they want a system to do and may have only the most ill-formed picture of what they want from a system.
* Working effectively with clients, including sophisticated techniques for stakeholder analysis and meeting design specifications in software projects.
* Designing systems that will actually accomplish the agreed-on requirements.
* Good and bad designs, and why they are good or bad.
* Managing software projects and software teams, including estimation and scheduling and, more importantly, dealing with the near-universal fact of software schedule slips.
* Software process improvement efforts and software process groups.
* Effective software testing.

In summary, the most technically elegant software system will be a waste of time and resources to an organization if the system does not contribute to the organizations needs and goals.  There are far too many stories of multimillion-dollar systems that are developed and discarded because they did not fit the actual needs of the organization.

The field of design and analysis (or software engineering) is a systematic approach designed to avoid these disasters and thus is an invaluable part of the software developer’s education.

Currently, the Department of Computer Science offers the following two courses in the area of Design and Analysis:

* **CSCI 567 – Software Engineering**   
  Phases of the systems development life cycle and the tools used by the analyst in planning, specifying, and implementing a complex computer-based system. Related topics include documentation standards, interaction with users, and design of interfaces. Object-oriented design principles are included. Assignments include at least one major group project.
* **CSCI 663 – Systems Design and Analysis**   
  Design, implementation, and control of computer-based Computer Science systems. Topics include system and program specifications, language selections, forms and file design, maintenance and documentation requirements, and interface with computer users. The topic of working with customers to develop requirements receives particular emphasis.

***A History Of Structured Systems Analysis & Design Methodologies***

***Thomas Kwasa***

***Fall 2011***

***IS6840***

***Dr. Vicki Sauter***

***What is SSADM ?***

Sometimes referred to as SSADM is “a set of standards developed in the early 1980s for systems analysis and application design. “ SSADM uses a combination of text and diagrams throughout the whole life cycle of a [system](http://www.webopedia.com/TERM/S/system.html), from the initial design idea to the actual physical design of the application.” (1) . In general, large-scale system computing for the management of accounting, billing and human resource functions emerged in the mid 1970s. At the time, there were no standardized development methodologies to follow. Most requirements analysis was performed on an ad-hoc basis and mainly depended on the impressions of small groups of people. Often this resulted in systems that did not meet the requirements of the business units.

***Why SSADM ?***

SSADM was sought to help reduce the lifecycle development costs through improved and enhanced design and development. SSADM was also believed to improve the quality of the systems it delivered. It was also thought to improve project management planning and control as well as provide more effective control of inexperienced staff. SSADM suggested an improvement in global understanding and communication. SSADM, by its emphasis on thorough documentation was supposed to assure project stakeholders that system documentation would take a more prominent role. (2)

***1970s pre SSADM – First Generation (Classical) Methodologies***

In the 1970s the “first generation methodologies” of structured analysis attempted to mainly create a computerized model of the administrative functions of the typical office. “Computerized systems were seen as having the same structure and functions as the manual systems and consequently, the underlying idea was simply to model the existing system and then design a computerized version it. “ (3) These “semi formal” design strategies and methodologies of the 1970s focused on mimicking the logical flows of the business. Systems designers acknowledged an incompleteness with this approach. Kimble describes it as having “an anti-realist ontology and rationalist epistemology” (3). This refers to design strategies that rationalized and approximated logical flows believing that “through the application of reason we can begin to make reality more explicable. “(3). The anti-realist ontology refers to the shortcomings of these first generation methodologies to reach outside of the logical for design ideas and guidance. The first generation methodologies required documentation that spelled out the document flows and the functional requirements in detailed terms. Edward Yourdon published his first criticisms of traditionalist approaches in his 1979 book “Structured Design”.

***Criticisms of Classical Systems Analysis Approaches:***

***Classical Methods Were Monolithic:***

Functional specifications were a large and often cumbersome required document for the early design methods. The specifications dealt with the entire system and therefore required that the developers had to understand them in their entirety. “If you didn’t read the last page, you had little idea of how the story would end.” (8) This shortcoming hindered emerging design approaches that sought a more modular approach to design where user may have sought an understanding of one of the building blocks of the system without necessarily needing to understand the others (6).

***Classical Methods Contained Redundancies* :**

Information was often repeated several times in different sections of the functional requirement document. Consider that when the user requirements changed, the adjustments would need to be made in several different areas of the functional specifications document. In the early 1960s updates and revisions could prove to be a challenge in the absence of the word processing capabilities of today (6).

***Classical Methods Contained Inconsistencies:***

Redundancy and the need to revise several areas of the functional document often created the related problems of inconsistency. Several revisions that needed to follow through to several different sections of the functional document often meant occasional errors. Sections that had not been updated correctly caused inconsistencies.

***Classical Methods Were Ambiguous*:**

Detailed user requirements were sometimes interpreted differently by users.

***Classically Based Systems Were Difficult To Maintain:***Functional specifications were often obsolete by the end of the systems development process. Since the design methodology did not consider modular operations, it became more complicated to maintain and update without the guidance of the original documents which, would also by then no longer represent the major functions adequately.

***1980s – SASD and SSADM in Evolution:***

The shortcomings of the classical approaches were in debate in the early 1980s and studies that suggested new ideas were beginning to take center stage. These new ideas espoused the need for more modular design methods that allowed programmers insight into sections of the process rather than the process as a whole. This new approach was generally referred to as SASD for systems analysis and systems design. The evolution generally emerged at the time of the migration from the legacy systems to the newer object oriented (OOP) approaches. Unfortunately these changes led to a new complexity where several different information professionals too often understood only pieces rather than the system as a whole. A new emphasis on a more wholesome graphic breakdown of the system with partitioned sections that reduced redundancy emerged. DeMarco in his 1978 book “Structured Analysis” argued against the use of several classical approaches citing the complexity and volume of functional requirements.

DeMarco claimed (to some debate) that developers were not fully engaged in reviewing the functional requirements which was causing final product to lack in adherence to the original requirements (5). Edward Nash Yourdon in his 1989 book – Modern Structured Analysis, fully claimed to be replacing the “Top Down” approach of classical systems analysis with a more event-driven process illustrated in the graphic of the *Data Flow Diagram*. Modern structured analysis uses functional decomposition, data flow diagrams, process descriptions and a data dictionary to model information systems. (9). Yourdon’s approach detailed out steps as follows A *physical model of the existing system*was created using data flow diagrams and supporting documentation. Then, the physical details were removed from the diagrams and supporting documentation to produce a *logical model of the existing system*. Finally, changes were made and functions were added to the logical model of the existing system.

***Criticisms of SASD Methodologies:***

***SASD was Time Consuming:***

The weaknesses of the new SASD methodologies began to emerge. Developers considered the process of modeling the existing physical system and then stripping away those very same physical components prior to creating a logical model and then modifying the logical model as too time consuming.

***SASD Retained Unnecessary Ties To The Old System:***

There was also the retention of many unnecessary artifacts from the existing system which made the old system the foundation for the newer system. This was fine if the older system had not been originally flawed or ill performing.

***Data Flow Diagramming Methods Differed*:**

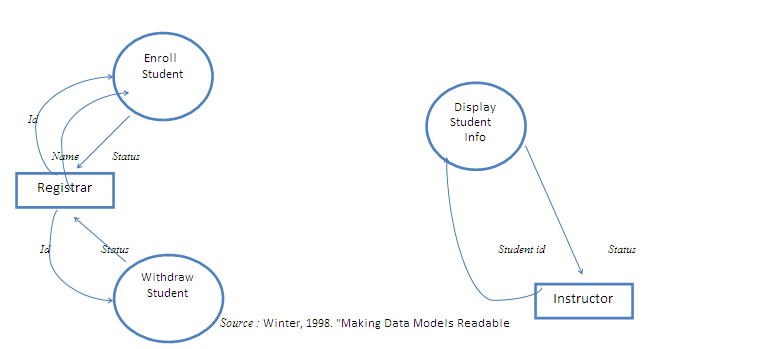
Data flow diagrams were useful but lacked a guideline or standard for their creation. Different developers and analysts working on the same system came up with very different ways of developing them and therefore confusing impressions of the requirements of the new system. These competing views of the system and its interactions caused costly reworking efforts to bring different teams on the same page.

***SSADM Techniques:***

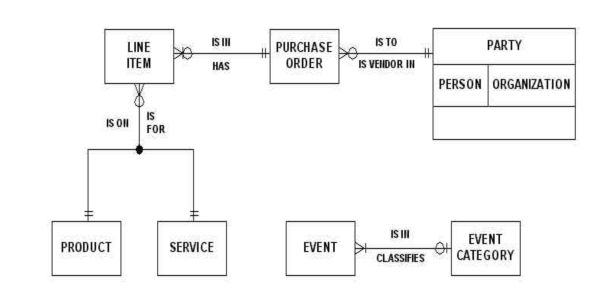
The main techniques encompassed by SSADM include ***Logical Data Modeling*** where the process data requirements are identified, modeled and documented. (4). The system’s data are then separated into entities (system players and actors) and relationships (the associations between the entities). There is also ***Data Flow Modeling*** where the data movements around the information system are identified, modeled and documented. Data Flow Modeling examines the processes; activities that transform data from one form to another; the data stores and the external entities which are the external sources of data for the system. It also looks at the data flows and directions in which the data move. This was adopted from the prior models set forth by Yourdon and DeMarco. Finally there is ***Entity Behavior Modeling*** which is the process of identifying, modeling and documenting the events that affect each entity and the sequence in which these events occur (4).

Here we look at two of the graphical development tools supported by SSADM. The Event List outlines the interactions between different events in a system and their relationships with one another. Edward Yourdon and Tom DeMarco are proponents of the use of several graphical tools to define clear event lists to direct the development of systems. The Entity Relationship diagram addresses the relationships between the different actors within a system. “Information Engineering” was a book published by Finkelstein and Martin that also considered data modeling and graphical tools of utmost importance when creating a system (7). They considered them critical to achieving the goals of SSADM. Finklestein’s and Martin’s approaches to graphical representations are widely in use in the United States. Finklestein defines entities in the designer’s sense of representing "data to be stored for later reference" (8) while Martin refers to entities “as something (real or abstract) about which we store data." (9).

***Simple Yourdon/DeMarco Event Lists***



***Finkelstein’s Entity Relationship Diagram***



***1980s UK Government’s Commissioning of SSADM***

In the 1980s the Central Computer Technology Agency CCTA (UK) was the first agency to explore and evaluate analysis and design methods. At the time, Yourdon’s approaches were widely in use and represented the standard for structured analysis. In spite of the use of a preferred standard, there was a widespread perception of failure on the part of several information technology projects. The large scale technology projects that various government agencies were involved in had suffered from this perception. Cost overruns were common and gaps between the requirements and functionality had exposed the need for a standard. To reduce the costly waste which was occurring at the time, the CCTA sought out a more representative methodology.

The CCTA which oversees various UK government projects, commissioned a proposal from the Learmonth and Burchett Management Systems group (LBMS) as a basis for the design of SSADM. LBMS won out over four other management firms and began to flesh out a wholesome design methodology based on their prior work. The final product was released in 1981 as the first version of SSADM. The CCTA commission had required specific adherences for their model (Appendix A) which LBMS had appropriately taken into consideration. In 1983 SSADM was made mandatory for all government projects in the UK.

***Structuring Of SSADM***

SSADM has some common phases that facilitate the movement from concept to prototype to finished product. The ***feasibility study*** is a high level overview of the project and its benefits. What the project hopes to accomplish and the problems it proposes to resolve. Cost benefit analysis of the project usually takes place at this stage. Once the project has been agreed upon and the ***requirements analysis*** phase begins where an investigation of the technological and business options is undertaken. The purpose of the requirements phase is to examine the objectives of the project and propose long term and cost effective solutions. At this time, the systems developers familiarize themselves with the technical jargon of the business areas while reviewing the data requirements and system’s needs. Users are heavily involved in this phase of the development process. There is often a clear reliance on the comparison of the functionality of the current system to the new system being proposed. ***Specifications and requirements*** write up is the next stage in the creation of the new system. At this point, detailed graphical documents that explore and document the relationships between the players and the system are created. The choice is made for which development option best meets the needs of the users. The ***logical system specification*** design begins and involves a cost benefit analysis of the hardware options. The logical design of the new system also begins to take place. The databases, update and delete functions and the methods of handling system inquiries are all placed into model. Finally the ***physical design*** stage is reached where both the data and the processing options are converted into a design that will run on the targeted system. (9).

***1990s – SSADM vs DSDM and other 4th Generation Tools***

The 1990s saw rapid emergence of new software development methods that questioned the necessity of the “slow and thorough” approaches of the eighties . SSADM is considered a thorough method of creating a system from the ground up but some of the newer development methods are bypassing several steps for the sake of expediency (11). Agile and RAD (Rapid Application Development) make use of some of the facets of SSADM but transition much faster and more directly into the development phase. Keith Richards of KRS consultants cites the DSDM (Dynamic Software Development Methods) as being the alternative to SSADM techniques. SSADM is considered a methodology borne out of waterfall while DSDM is a rapid development technique. Pat Phelan seems to agree with this and he argues that

“SSADM (Structured Systems Analysis and Design Methodology) is based on the traditional Structured Programming techniques. It uses a formal design process that is a direct descendant of the "waterfall" methodologies. This process tends to be relatively slow, but because the process tends to be exhaustive in both finding and debating every reasonable need, the results tend to be large and cumbersome, but thorough.” Pat also does acknowledge a major shortcoming of rapid application design which is that the product often does not hold up well over time. (12).

***Sources:***

1.        “SSADM.” Webopedia. internet.com. 2011. Octoberr 5, 2011 <http://www.webopedia.com/TERM/S/SSADM.html>

2.        Goodland, Mike and Riha, Carol. “SSADM – An Introduction”. 1/20/1999 [http://www.dcs.bbk.ac.uk/~steve/1/](http://www.dcs.bbk.ac.uk/%7Esteve/1/)

3.        Kimble, Chris. System Design Methodologies (SDM) Semi Formal Methodologies: The First Generation Methodologies. chris-kimble.com. November 5, 2011. <http://www.chris-kimble.com/Courses/sdm/Session_5.html>

4.        Watkins, Glynn. Lecture 8 - The Stages Of SSADM. Bristol University Of West England. October 5, 2011.

[www.cems.uwe.ac.uk/~gwatkins/.../lec8the\_stages\_of\_ssadm\_v2.doc](http://www.cems.uwe.ac.uk/%7Egwatkins/.../lec8the_stages_of_ssadm_v2.doc)

5.       DeMarco,Tom. Structured Analysis and System Specification. New York: YOURDON Press, 1978

6.        [Edward Yourdon](http://www.yourdon.com/) and Larry L. Constantine. Structured Design: Fundamentals of a Discipline of Computer Program and System Design , 1979.

7.        Martin, James and Clive Finkelstein. Nov 1981. "Information Engineering", *Technical Report*. Lancs, UK : Savant Institute, Carnforth.

8.        Clive Finkelstein. Nov 1992. Strategic Systems Development*.* Sydney: Addison-Wesley

9.        Martin, James, and Carma McClure. 1985. Diagramming techniques for Analysts and Programmers. Englewood Cliffs, NJ: Prentice Hall.

10.     Winter, 1998. "Making Data Models Readable". Information Systems Management.15, Boca Raton, FL: CRC Press.

11.     Brown, Robert G.. 1993. "Data Modeling Methodologies – Contrasts in Style". *Handbook of Data Management.* Boston: Auerbach.

12.     University Of Aberdeen. Structured Systems Analysis and Design Methods. CS5540. Scotland. October 15, 2011. [www.csd.abdn.ac.uk/~jmasthof/teaching/CS5540/.../lecture9.ppt](http://www.csd.abdn.ac.uk/%7Ejmasthof/teaching/CS5540/.../lecture9.ppt)

13.     Richards, Keith. DSDM and XP On Reflection: KRC Consultants. 2011 <http://www.keithrichardsconsulting.co.uk/site/about/latest/dsdm_and_xp_article.html>

14.     Phelan, Pat. Search Oracle. Using SSADM and DSDM for rapid application development. November

<http://searchoracle.techtarget.com/answer/Using-SSADM-and-DSDM-for-rapid-application-development>

15.   The use of SSADM (Structured Systems Analysis and Design Methodology) as a standard methodology on Information Systems Projects. GRIN. <http://www.grin.com/en/e-book/106034/the-use-of-ssadm-structured-systems-analysis-and-design-methodology-as>

