9 Systems Development Methodologies

9.1 SSADM

Structured Systems Analysis and Design Method (SSADM) is a structured development method that was developed initially in the 1980's as a public domain standard development method (Beynon-Davies, 2009). SSADM focuses on the feasibility, analysis and design aspects of the systems development life cycle. It provides fewer guidelines on the changeover and maintenance aspects of an IS project. Describing SSADM in some detail highlights the methodical approach required for large-scale projects which some may refer to as bureaucratic. It also illustrates the contrast with alternative techniques such as RAD. SSADM has a five-module framework within which are seven stages. The five modules are now discussed.

9.1.1 Feasibility Study

The project will already have been through a planning or initiation stage, so it is necessary at this point to determine whether it is technically and economically feasible. The feasibility study is broken down into four steps:

- *prepare for feasibility study* by assessing the scope of the project;
- *define the problem* (what should the new system do that the present one does not);
- *select the best feasibility option* from those available (typically up to five business options and a similar number of technical options);
- *assemble the feasibility report*, including a rationale for the selected option.

The output from this stage, the feasibility report, now provides the input for the next module; requirements analysis.

9.1.2 Requirements Analysis

This stage is critically important because it is used to gain a full understanding of what is required of the new system. Any errors or omissions made at this stage will be reflected in the rest of the systems development process. The following steps are taken:

- *Establish analysis framework*. The scope of the project is reassessed and then planned accordingly.
- *Investigate and define requirements*. Broad requirements will have been defined at the feasibility stage: these are now expanded into a detailed catalogue of systems requirements.
- *Investigate current processing*. The feasibility study will have created an initial data flow diagram which is now expanded to embrace all the existing processes.

- *Investigate current data*. A logical data model is developed so that the organisation can obtain a clear picture of which attributes the data entities contain and how they relate to each other.

- *Derive logical view of current services*. This involves the revision of the logical data model so that it reflects the business logic of the system under consideration rather than its current physical implementation.

- *Assemble investigation results*. This is the last step in the analysis of the current system environment. The analysts will check for consistency and completeness before proceeding to the next stage.

A number of possible systems solutions for the perceived business requirements are formulated and the impacts and benefits of each will be evaluated. The solution selected will be the one that most closely matches the requirements of the business. The two steps are:

- *Define business systems options*. Activities here will include the establishment of minimum systems requirements, the development in skeleton form of alternatives, the production of a short list of options, and finally a full evaluation of each alternative short-listed option, including a cost–benefit analysis, impact analysis and system development and integration plan for each.

- *Select business system option*. The precise way in which this is done will vary between organisations. The objective is the same, however: for appropriate user managers to select the business system option from the evidence presented by the analysis team.

9.1.3 Requirements specification

This module has one stage which in turn is split into eight discrete steps.

- *Define required system processing*. Here, the features of the existing system that are to remain a part of the new system are added to the details contained in the requirements catalogue.

- *Develop required data model*. Redundant elements from the data model of the existing system are removed (if any exist) and additional required elements are added. In addition, the relationships between old and new entities are reviewed.

- *Derive system functions*. Here, the processes that will have been identified and incorporated in the data flow diagrams are identified more precisely and properly documented.

- *Enhance required data model*. The required data model developed earlier is now enhanced by carrying out relational data analysis and normalisation; the result should be a set of tables which can be implemented using a relational database management system.

- *Develop specification prototypes*. This involves the creation of prototypes for selected parts of the specification so that precise requirements can be validated with the intended end-users; such elements as menus, sample data entry screens and reports may be constructed.

- *Develop processing specifications*. The analyst at this stage is concerned with illustrating the effect of time on data subjected to various actions (i.e. creation, reading, updating and deleting); two tools that are used here are entity life history analysis and effect correspondence diagrams. These are tools used by the professional systems analyst and it is beyond the scope of this book to deal with them in detail.

- *Confirm system objectives*. The penultimate task is to carry out a formal review of the system requirements to ensure that the final requirements specification which follows is complete and fully understood by users and developers alike.

- *Assemble requirements specification*. Finally, the various components (including the required system logical data model, function definitions, requirements catalogue and other items) are assembled into the final requirements specification document, which then provides the input into the next module and stage.

9.1.4 Logical system specification

Here, any constraints on the choice of technical environments are established (e.g. security, performance, ease of upgrade). The appropriate technical option is selected; it must conform with the required strategic and operational criteria which have already been established. The process of developing the systems specification is continued, with the outcome being a set of implementable components. The individual steps are as follows:

- *Define user dialogues*. This is concerned with defining the ways in which the user will interact with the system (e.g. menus and systems navigation).

- *Define update processes*. Here, the definition of transactions which will change data are established (entity life histories are used to support this step).

- *Define enquiry processes*. In addition to navigation and updating, users will wish to perform enquiries on the data held in the system.

- *Assemble logical design*. This is essentially a consistency and completeness check. Once the logical design is complete and has been 'signed off', the final stage can be tackled.

9.1.5 Physical design

This stage is concerned with the delivery of the final blueprint from which the system can be developed and implemented. There are seven steps to be completed:

- *Prepare for physical design*. The implementation environment is studied, applications development standards drawn up and a physical design strategy agreed.

- *Create physical data design*. The required logical data model (LDM) is used as a base for this and the business-specific data design is produced.

- *Create function component implementation map*. The components of each systems function are drawn up. This includes their relationship with the physical function components (the actual business activities) which they support.

- *Optimise physical data design*. The physical data design is tested against the required performance objectives and optimised if necessary.

- *Complete function specification design*. This will be for any function components that required programming.

- *Consolidate process data interface*. The process data interface is located between the physical database design and the process design. This helps the mapping of the database to the processing requirements (especially important when the database has been altered or the processing requirements have been modified).

- *Assemble physical design*. This stage and the whole SSADM lifecycle are completed with this step. A number of products are delivered, including the function definitions, the optimised physical data design, the requirements catalogue and space and timing estimates.

9.2 Rapid applications development (RAD)

The evidence from project failures for projects in the 1980s and 1990s implies that traditional structured methodologies have a tendency to deliver systems that arrive too late and therefore no longer meet their original requirements. Traditional methods can fail in a number of ways:

- *A gap of understanding between users and developers*. Users tend to know less about what is possible and practical from a technology perspective, while developers may be less aware of the underlying business decision-making issues which lie behind the systems development requirement.

- *Tendency of developers to isolate themselves from users*. Historically, systems developers have been able to hide behind a wall of jargon, thus rendering the user community at an immediate disadvantage when discussing IS/IT issues. While some jargon may be necessary if points are to be made succinctly, it is often used to obscure poor progress with a particular development project. The tendency for isolation is enhanced by physical separation of some computer staff in their own air-conditioned computer rooms. Developers might argue in their defence that users also have their own domain-specific jargon which adds to the problem of deciphering requirements.

- *Quality measured by closeness of product to specification*. This is a fundamental difficulty – the observation that 'the system does exactly what the specification said it would do' hides the fact that the system may still not deliver the information that the users need for decision-making purposes. The real focus should be on a comparison of the deliverables with the requirements, rather than of deliverables with a specification that was a reflection of a perceived need at a particular point in time.

- *Long development times*. A glance back at the previous section on SSADM and the waterfall model will reveal that the processes of analysis and design can be very laborious and time consuming. Development times are not helped by the fact that an organisation may be facing rapidly changing business conditions and requirements may similarly be changing. There is a real risk of the 'moving goal-posts' syndrome causing havoc with a traditional approach to systems development.

- *Business needs change during the development process*. This is alluded to above. A method is needed where successive iterations in the development process are possible so that the latest requirements can be incorporated.

- *What users get isn't necessarily what they want*. The first a user may see of a new information system is at the testing or training stage. At this point, it will be seen whether the system as delivered by the IS/IT professionals is what the user actually needs. An appropriate analogy here is the purchase of a house or car simply on the basis of discussions with an estate agent or a garage, rather than by actually visiting the house or driving the car. It is unlikely that something purchased in this way will result in a satisfied customer and there is no reason to suppose that information systems developed in a similar way will be any more successful.

Not only is there pressure from end-user management for faster systems development, IS/IT departments themselves increasingly recognise the need to make more effective use of limited human resources within their departments while at the same time quickly delivering systems that confer business benefits. All this is in a climate of rapid business change and, therefore, rapidly changing information needs. Rapid applications development (RAD) is a possible solution to these problems and pressures. This uses prototyping to involve users and increase development speed. Rapid applications development (RAD) is a method of developing information systems which uses prototyping to achieve user involvement and faster development compared to traditional methodologies such as SSADM. Prototyping produces a preliminary version of part or a framework of all of an information system which can be reviewed by end-users. Prototyping is an iterative process where users suggest modifications before further prototypes and the final information system are built.

9.3 The spiral model

The spiral model is an iterative systems development model developed by Boehm (1988) which incorporates risk assessment. The spiral model was developed in recognition of the fact that systems development projects tend to repeat the stages of analysis, design and code as part of the prototyping process. Each spiral consists of four main activities which are:

- *Planning*. Setting project objectives, defining alternatives.
- *Risk analysis*. Analysis of alternatives and the identification and solution of risks.
- *Engineering*. Equivalent to the build phase of the SDLC with coding and testing.
- *Customer evaluation*. Testing of the product by customers.

The model is closely related to RAD, since it implies iterative development with a review possible after each iteration or spiral, which corresponds to the production of one prototype or incremental version. Before the first spiral starts the requirements plan is produced, so it can be seen that the spiral model does not detail the initiation and analysis phase of the SDLC, focusing on design and build. Although the spiral model has not been applied widely in industry, proponents of this model argue that it includes the best features of both the classic SDLC and the prototyping approach. It also adds validation of requirements and design, together with risk analysis, which is often overlooked in RAD projects.

9.4 The Capability Maturity Model

Another influential model for best practice in the development of BIS is the Capability Maturity Model for Software. This model, which has been revised throughout the 1990s and into the new millennium, challenges organisations to review their process of systems development. It provides a framework for managers to assess the current sophistication of their process for systems development. There are five stages to the model. These are described by the institute as:

- *Initial*. The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics.
- *Repeatable*. Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.
- *Defined*. The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software.

- *Managed*. Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.

- *Optimizing*. Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.