Structured systems analysis and design method (SSADM)

CAROLINE M ASHWORTH

Abstract: The structured systems analysis and design method (SSADM) is the standard structured method used for computer projects in UK government departments. It is also being adopted as a standard by various other bodies. Responsibility for SSADM belongs to the Central Computer and Telecommunications Agency (CCTA), HM Treasury although support and training may be acquired through commercial organizations.

SSADM has certain underlying principles and consists of six stages which are broken down into a number of steps and tasks. Within this framework, a number of structured techniques are used and documents produced.

SSADM may be compared with methods that use some of the same techniques. Two other methods (Yourdon and Arthur Young's Information Engineering) are briefly described and some comparisons drawn with SSADM.

Keywords: systems analysis, methodologies, information systems, JSP, quality assurance.

In 1980 the UK Government initiated a lengthy procedure to select a structured method to be the standard throughout all computer projects in UK government departments. Most of the better-known structured methods were considered, but the method selected was put together specifically for the purpose by UK consultancy, Learmonth and Burchett Management Systems (LBMS). This method was seen to integrate several relatively mature structured techniques (and a newer technique) into a clear procedural framework leading from the analysis of the current system through to the physical design of the new system. After an initial hand-over period, the Central Computer and Telecommunications Agency (CCTA), HM Treasury is now the design authority. It has recently applied to register SSADM as a certification trademark.

Since its introduction in 1981, the use of SSADM has grown to the extent that in 1987 more than 600 government projects are estimated to have used or are using SSADM. SSADM has also been adopted as a standard by public utilities, local government, health authorities, foreign governments and several large private sector organizations. SSADM is now widely available outside the Government. The National Computing Centre (NCC) has a collaborative agreement with the CCTA for the development and administration of SSADM and publishes the official reference manual[1]. The method is also described in a recently published book by Downs, Clare and Coe[2].

The experience from the many government projects has been channelled back into the development of the method through several mechanisms including:

- SSADM user group
- SSADM consultants from the CCTA who support projects
- private sector organizations
- NCC

The current version in use is the third since the introduction of the method. As a result of the experience in use, together with the mechanisms for using this experience in developing and enhancing the method, SSADM can claim to be one of the most mature methods in use in the UK.

SSADM was initially designed to be used in conjunction with two other UK government standards, the Prompt project management and control methodology[3] and structured design method (SDM), a version of Jackson structured programming[4]. The method also works in the context of fourth generation technology and it is now used extensively with a variety of application generators.

BASIC PRINCIPLES

The basic principles of SSADM are shared, to a varying degree, by many of the modern structured methods of systems analysis and design. These principles underpin the whole development life cycle and should be referred to when proposing to tailor the method for specific project circumstances.

Data-driven

All application systems have an underlying, generic data structure which changes little over time, although processing requirements may change. Within SSADM, it is a central principle that this underlying data structure is developed from an early stage, checked

Scicon Ltd, Wavendon Tower, Wavendon, Milton Keynes, Bucks MK17 8LX, UK

0950-5849/88/030153-11$03.00 © 1988 Butterworth & Co (Publishers) Ltd.

against the processing and reporting requirements and finally built into the system's architecture.

Differentiation between logical and physical

SSADM separates logical design from physical design. A hardware/software independent logical design is produced which can be translated into an initial physical design. This helps the developers to address one problem at a time and prevents unnecessary constraints being added at too early a stage in development. This also helps communication with users who may not be computer literate but are able to validate a logical specification or design of their system.

Different views of the system

Three different views of the system are developed in analysis. These views are closely related to one another and are cross-checked for consistency and completeness. The equal weight given to these three techniques and the prescriptive procedures for checking them against one another is a strength of the SSADM approach. The views are:

- underlying structure of the system's data (the logical data structure),
- how data flows into and out of the system and is transformed within the system (data flow diagrams),
- how the system data is changed by events over time (entity life histories).

Top-down and bottom-up

SSADM contains elements of both top-down and bottom-up approaches. In the early stages of a project, top-down techniques such as data flow diagramming and logical data structuring are used. In the logical design stage bottom-up techniques such as relational data analysis are used to provide more of the detail and then reconciled with the top-down views to produce a validated logical design.

User involvement

It is considered important that end users have involvement in, and commitment to, the development of the system from an early stage. By ensuring that the specification and design match the user's requirements at each stage, the risk of producing the 'wrong' system is reduced and the possible problems can be solved before they become unmanageable.

User involvement is encouraged by the use of easily understood, non-technical diagrammatic techniques supported by short, simple narrative descriptions. Users participate in formal quality assurance reviews and informal 'walkthroughs' and should 'sign off' each stage before the developers progress to the next.

As the techniques of SSADM do not require skill in computer systems, it has been found that an ideal situation is one in which a user representative works full-time within the development team. This provides a constant supply of knowledge about the system and provides a bridge between the developers and users.

Quality assurance

The use of informal quality assurance reviews and walkthroughs is encouraged throughout the method. Formal quality assurance reviews are held at the end of each SSADM stage. The end products for the stage are scrutinized for quality, completeness, consistency and applicability by users, developers and experienced systems staff external to the project. Each stage can therefore be signed off to act as a baseline for the subsequent stage.

Self documenting

The products of each SSADM step form the project documentation and are used in subsequent steps. It becomes important that the documentation is completed at the relevant time within the project instead of being left until the project is complete, as often happens when timescales are short. This ensures that the documentation is up-to-date at all times.

OVERVIEW OF SSADM

The structured techniques fit into a framework of steps and stages, each with defined inputs and outputs. Also, there are a number of forms and documents that are specified which add information to that held within the diagrams. Thus, SSADM consists of three features of equal importance:

- structure of the method,
- structured techniques and their interrelationship,
- documents and forms produced.

Structure of the method

Figure 1 shows the stages of an SSADM project. Each stage is broken down into a number of steps which define inputs, outputs and tasks to be performed. The products of each step and the interfaces between steps are clearly defined in the SSADM documentation[1].

The structure of the method illustrates several features of the SSADM approach. First, the current system, in its current implementation, is studied to gain an understanding of the environment of the new system. This view of the current system is used to build the specification of the required system. However, the required system is not constrained by the way in which the current system is implemented. The specification of requirements is detailed to the extent that detailed technical options can be formulated. The detailed design is completed at the logical level before implementation issues are addressed. Finally, the logical design is converted into physical design by the

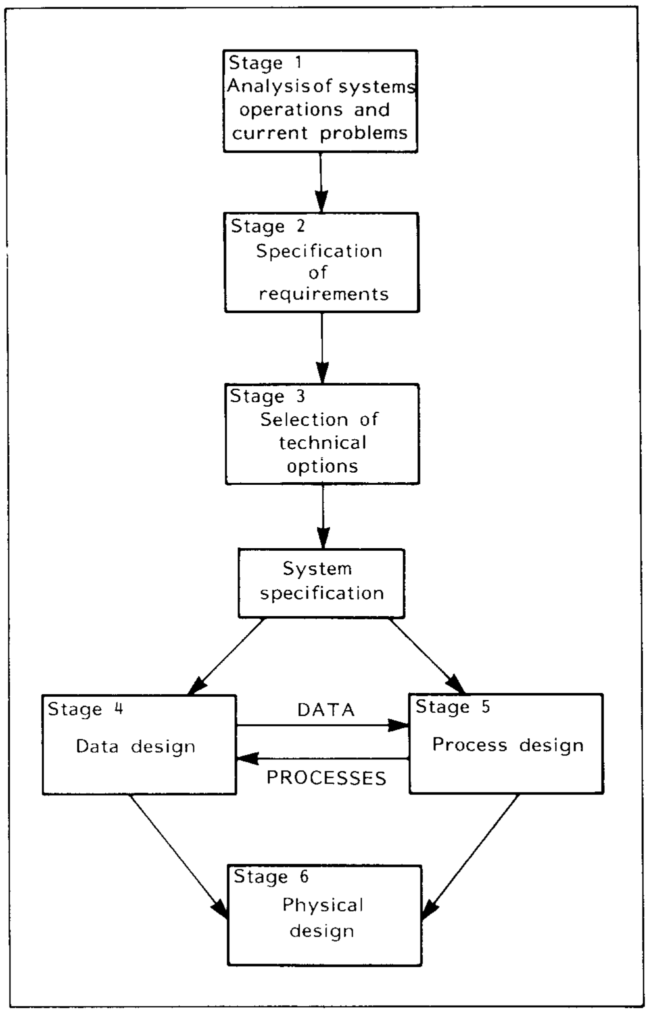


Figure 1. Stages of SSADM

application of simple (first cut) rules. The resulting design is tuned using the technique of physical design control before implementation. The breakdown of each stage into constituent steps is shown at annex A at the end of this paper.

Stage one: Analysis system operation and current problems

The current system is investigated for several reasons, for example, the analysts learn the terminology and function of the users' environment. The data required by the system can be investigated. The current system provides the users with a good introduction to the techniques and the boundaries of the investigation can be clearly set.

The second reason illustrates one of the principles of SSADM that the underlying structure of the data of a system will not change much over time. Even though the introduction of a new computer system may change the functions (a computer system can increase what can be tackled by users), the underlying data required to perform the functions will not change much. If there is no current system, for example where there is a new law that requires support, this stage consists of initiating the project and beginning to document the new requirements.

Stage two: Specification of requirements

In order that the new system will not be constrained by the current implementation, there are a number of steps within this stage to gradually lead the analysts away from the current system towards a fresh view of the requirements.

First, the current system view built up in stage one is redrawn to extract what the system does without any indication of how this is achieved. The resulting picture is the logical view of the current system. This allows the analyst to concentrate on what functions are performed in the current system and to make decisions about what must be included in the new system.

The current system is surpassed by the business system options (BSOs) which are completed next. The BSOs express the requirements in a number of different ways to reflect the different ways in which the system might be organized. These are not implementation decisions, although they may constrain the way the system is implemented. Instead, this is a way of taking a fresh view of what the system is required to do and how the business can be organized to make the best use of the system. Based upon the selected business system option, a detailed specification of the required system is built up and checked extensively.

Stage three: Selection of technical options

At this stage, if the purchase of new computer equipment is required, the development team have enough information to compile the different implementation options for the system. Each option costed out and the benefits weighed against the costs to help the user choose the final solution. This might form the basis for competitive tendering for the final system hardware.

Stage four: Logical data design

This stage builds up the logical data design so that all the required data will be included. It applies a relational analysis technique to groups of data items in the system to act as a cross-check on the data definition built up in stage two. The final data design is checked against the logical processes, developed in stage five, to ensure that all the data needed by the processes is present in the data design.

Stage five: Logical process design

The definition developed in stage two is expanded to a low level of detail so that the implementor can be given the detail necessary to build the system. This processing definition is checked against the data definitions derived in stage four.

Stage six: Physical design

The complete logical design, both data and pṛocessing, is converted into a design that will run on the target environment. The initial physical design is tuned on paper before being implemented so that it will meet the performance requirements of the system. In this stage, much of the documentation required during the system implementation is produced. The implementation of the system takes place, traditionally, after this stage when the detailed program specifications are used as the basis for program design and coding, possibly using a program design method such as Jackson structured programming[4].

Structured techniques

The techniques of SSADM give standards for how each step and task is to be performed. The rules of the syntax and notation of each technique are supplemented with guidelines on how it should be applied in a particular step. The diagrammatic techniques of SSADM are data flow diagrams, logical data structuring, entity life histories and logical dialogue design. In addition, there are techniques and procedures that are not diagrammatic including:

- relational data analysis (TNF)
- first cut rules
- physical design control
- quality assurance reviewing
- project estimating

The SSADM reference material gives clear guidelines on each of the techniques and, more importantly, shows how they are interrelated and can be used to cross-check one another. The principal diagrammatic techniques and procedures are described in more detail below.

a. Logical data structure (LDS)

This is a method for describing what information should be held by the system. The approach used in SSADM is similar to entity modelling in other methods. A diagram is produced showing the entities and their relationships, this is further documented by a set of entity description forms detailing their data contents.

A logical data structure (LDS) is produced for the current system. This is extended to meet the requirements of the new system, resulting in a required system LDS. This LDS becomes the composite logical data design (CLDD) by comparison with the results of relational data analysis. The CLDD is used as the basis for the physical data design.

The major conventions of LDSs are summarized in Figure 2. These conventions are the same for the CLDD.

An entity can be thought of as either a 'thing' of significance to the system about which information will be held or a group of related data items that can be uniquely identified by a key. Which view predominates is influenced by the way in which the logical data structures are built up within SSADM; the former view is adopted when starting the whole process in Stage one and gradually the latter view is adopted so that by the time the composite logical data design is completed, the structure is thought of as 'system data'.

A relationship is a logical association between two entities. Within SSADM, only one-to-many relationships are permitted (one-to-one relationships are resolved by merging the entities and many-to-many relationships are resolved by inserting a 'link' entity). The 'crow's foot' indicates the 'many' end of the relationship. The relationships are validated by checking the assertions that, for example, an instance of 'overdrawn status' is related to many instances of 'customer' and that an instance of 'customer' will always be related to one, and only one, instance of 'overdrawn status'. If it is possible that 'customer' could exist without 'overdrawn status', then this relationship becomes optional, indicated by a small circle on the relationship. In Figure 2 the exclusive notation for relationships is illustrated, showing that instances of the two relationships to 'personal customer' and 'company' will never exist concurrently for the same 'bank account' entity.

b. Data flow diagrams (DFDs)

A data flow diagram[5,10] is a diagrammatic representation of the information flows within a system, showing how information enters and leaves the system; what changes the information; and where information is stored. Data flow diagrams are an important technique of systems analysis as a means of *boundary definition*. The diagrams clearly show the boundaries and scope of the system being represented. They also *check the completeness of analysis*. The construction of the diagrams, and their cross-comparison with the other major SSADM techniques, help ensure that all information flows, stores of information and activities within the system have been considered. DFDs denote the major functional areas of the system, and therefore the programs or program suites required. They may be

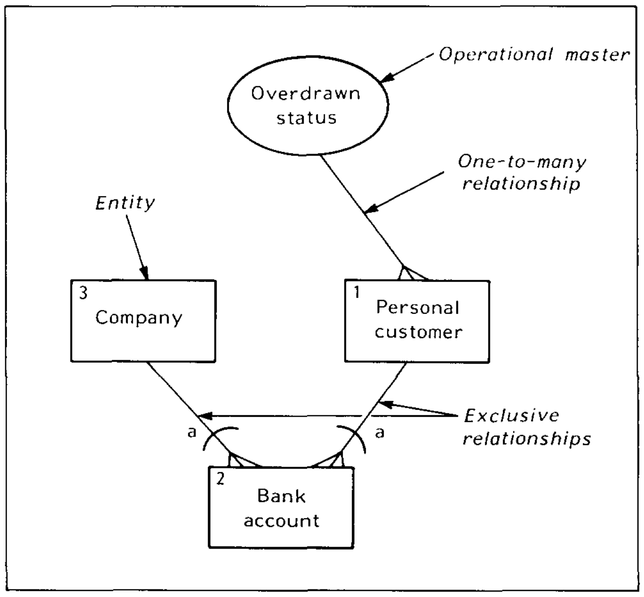


Figure 2. Major conventions of logical data structures

used to represent a physical system or a logical abstraction of a system.

In SSADM four sets of data flow diagrams are developed. First, the current physical. The current system is modelled in its present implementation. Second, the logical. The purely logical representation of the current system is extracted from the current physical DFDs. Third, the business system options. Several proposed designs are developed, each satisfying the requirements of the new system. Each of these is expressed as an overview, known as a business system option. Fourth, using the selected business system option and the logical data flow diagrams, a full set of data flow diagrams representing the new system is developed. The relationship between the different sets of data flow diagrams is represnted in Figure 3. The conventions of DFDs are illustrated in Figure 4.

External entities are sources or recipients of data, processes transform the data within the system and data stores are repositories of information. Data stores are closely related to entities on the logical data structure.

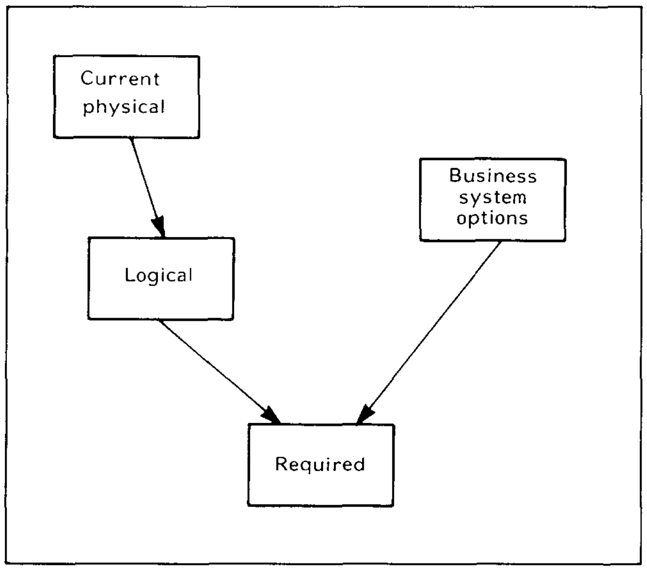


Figure 3. Data flow diagrams in SSADM

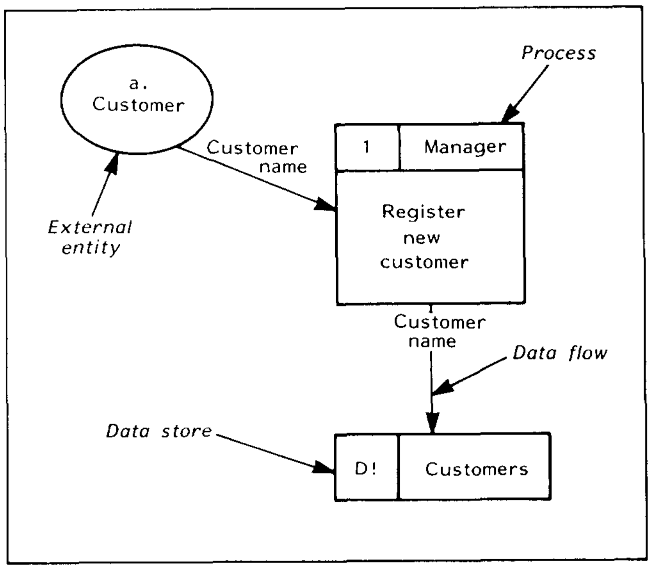


Figure 4. Conventions of data flow diagrams

Each process can be decomposed into a lower level data flow diagram, successively adding detail through each level.

c. Entity life histories (ELHs)

These are models of how the system's data is changed over time by events acting on entities. For each entity the sequence, selection and iteration of events affecting it are shown using a notation derived from Jackson[4].

An event is whatever triggers a process to update system data. As it would be too complicated to model the entire set of events for a whole system at once, the effects of the events upon each entity from the logical data structure are modelled. These individual views of the event sequences are drawn together in an entity/event matrix (ELH matrix) and process outlines. The major conventions of the entity life history technique are shown in Figure 5. The state indicators are a re-expression of the structure of the entity life history and may be used in validation in the implemented system.

d. Logical dialogue outlines

Logical dialogue outlines were introduced in version three of SSADM to allow developers to specify requirements for man-machine dialogues at an early stage in the development. The prototyping of dialogues using a screen painter, or similar rapid development software, to demonstrate the man-machine interface to users is obviously more effective in the specification of user requirements for dialogues, so dialogue outlines are designed to be used generally where prototyping

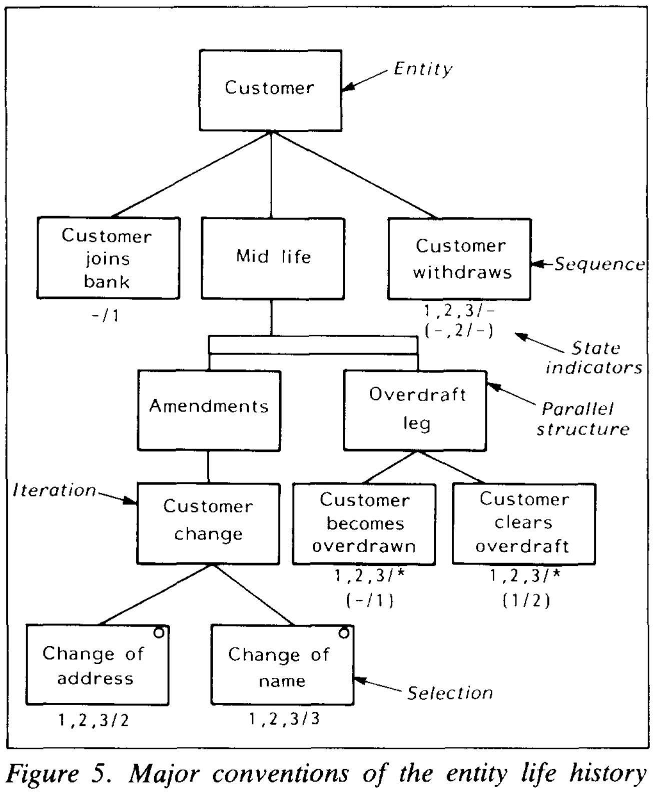


Figure 5. Major conventions of the entity life history technique

facilities are not available. A logical dialogue outline is produced for each non-trivial online event or enquiry identified during analysis. Thus, this technique is used towards the end of requirements definition in stage two. The data items flowing across the man-machine boundary are detailed, the sequence of logical 'screens' and an overview of the processing done to satisfy the dialogue are modelled using a flow-chart style notation. It is also possible to add the requirements for the time taken at each stage of the dialogue, points at which users will be required to make decisions, an indication of some messages that might be used and a cross-reference to operations on process outlines. An extract from a simple logical dialogue outline is shown in Figure 6. It is possible to create 'levels' by reflecting the context of one or more logical dialogue outlines on a higher-level outline called a logical dialogue control.

e. Relational data analysis (TNF)

Relational data analysis, based upon Codd's aproach[6], is used in the logical design stage of SSADM (Stage four) where it complements the logical data structuring done during requirements analysis. The merging of the two techniques results in the composite logical data design (CLDD) which is the basis for the physical database or file design.

Any collection of data items that have been defined without direct reference to the logical data structure can be used as an input to relational data analysis or normalization. Commonly, the input/output descriptions or screen definitions are used as inputs to this technique.

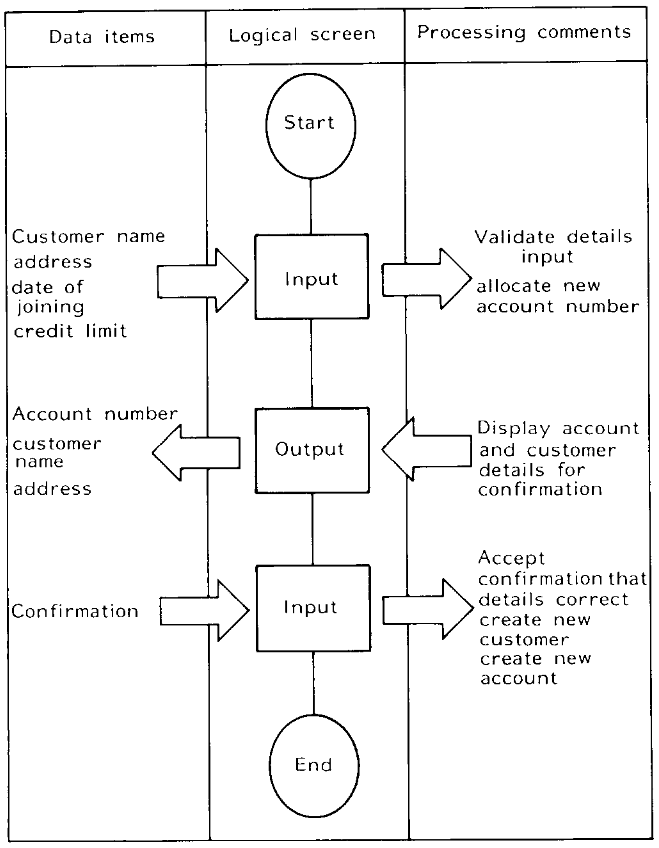


Figure 6. Example logical dialogue outline

Normalization consists of a progression from the original, unnormalized, data through several refinements (normal forms) until the data items are arranged to eliminate any repeating items or duplication. The results of performing this analysis on several different groups of data items are merged or optimized to give sets of data items that should correspond to the entities on the logical data structure. At this point, the logical data structure is merged with the results of the normalization.

The process of relational data analysis ensures that all data items required by the system are included in the system's data structure. Also, it is a good way to ensure that the data is fully understood. Although the rules of normalization appear to be mechanical, to apply them effectively the underlying relationships between data items must be well understood.

f. First cut rules and physical design control

The conversion of the logical process and data design into a workable physical design takes place in two phases. First, simple rules are applied which roughly convert the logical design into a corresponding design for the target environment. This design might work, but would probably not be efficient or exploit the features of the particular hardware or software that will be used. Therefore, the 'first cut' design is tuned using a process called physical design control. This consists of successively calculating the time taken to execute certain critical transactions, modifying the design slightly and recalculating until the performance objectives (defined in stage three) are met.

g. Quality assurance reviewing

SSADM places emphasis on holding formal quality assurance reviews at the end of each stage. It is important to ensure that the products from each stage are technically correct and that they meet the objectives of the users. The work for the second stage of SSADM has its foundations in the work done in the first stage. This principle applies throughout the project: each stage builds on the work done in the previous stage. There is a high risk that all subsequent work will be poor if the foundations are poor.

A formal sign-off by a group consisting principally of users emphasizes the joint responsibility for the project of both the users and the project team. This ensures the ongoing active interest of the users in the project and avoids the situation commonly encountered in systems analysis and design where communication between the project team and the users is minimal during the development phase leading to the implemented system not meeting the users' requirements.

Products from each stage should be reviewed by a team comprising responsible users who will have the authority to authorize the continuation of the project and at least one person with a good understanding of SSADM who will be referred to here as the 'technical reviewer'. This should be done on a formal basis to force the correction of errors identified by the reviewers before work is allowed to proceed to the subsequent stages.

The following procedures are an example of how quality assurance reviewing is undertaken within SSADM.

Before the review
All participants receive an invitation to the review meeting one week in advance of the meeting, together with a copy of all the documents they will be required to review. If any of the reviewers is unfamiliar with the conventions of the diagrams, then the analysts might arrange to explain the aspects of the diagrams that are relevant to a reviewer. This can be done on a one-to-one basis but can be achieved more efficiently when a number of people are involved. This is done by organizing a presentation to state the purpose and basic conventions of the diagrams with a more general discussion about quality assurance review procedures.

The review meeting
The actual review would not be more than one to two hours long. The chairman is either a user who has been closely involved with the project or the project team manager. The meeting should not attempt to solve the difficulties that might arise but should highlight errors for subsequent resolution away from the meeting. An analyst from the project team walks through the documentation being reviewed and invites comments from the reviewers. A list of errors is compiled by the chairman and agreed by the meeting. The reviewers may decide that the documentation contains no errors and meets its objectives in which case will sign the stage off at this meeting. More commonly, there will be a number of non-critical errors detected in which case the documentation may be signed off provided that certain follow-up action is taken and subsequently agreed by the reviewers out of the meeting. If there are numerous errors and the reviewers are not confident that the project team has met the objectives of the stage, then a date for another quality assurance review is set and the documentation failed.

After the review
Any necessary corrections are made to the documentation within a week of the review and circulated to the members of the review team. If the errors are only minor, the reviewers may sign it off individually. If the errors are more severe, the documentation is reviewed a second time at another review meeting.

The resources required to hold a quality assurance review are significant and should not be underestimated when the project plan is being prepared. At least three elaspsed weeks should be allowed for each formal review and one to two weeks for informal reviews. It is a temptation to cut this time when project timescales are tight. But compared to the weeks or months that might be wasted later in the project on trying to sort out compounded errors arising from poor quality assurance, it is time well spent.

h. Project estimating

Project estimating guidelines have been developed from experience and may be made available to project managers. They are based upon the techniques, steps and stages of SSADM. Certain factors will make timescales longer or shorter, for example the number of user areas and the complexity of the project. The estimating guidelines are applied after an initial data flow diagram and logical data structure have been drawn. The number of processes and entities on these initial diagrams give an indication of the number of diagrams that will be completed throughout the project. The results of the estimating guidelines are refined throughout the project. The estimates produced at the beginning of a project will not be accurate but will give some idea of the order of magnitude of a project.

Documents and forms

Documentation standards define how the products of this development activity should be presented. The forms are either supporting detail of the techniques or additional non-diagrammatic information. In the former category are entity descriptions and elementary function descriptions; in the latter category are the problems/requirements list and function catalogues.

In addition to forms, there are several working documents, principally matrices, which are used to help the start-up to some of the techniques. An entity matrix is used to identify relationships between entities as an initial step in the logical data structuring techniques and an entity-event (ELH) matrix is used as a basis for entity life histories.

One of the most central documents in the analysis stages of SSADM is the problem/requirement list. It is used as a checklist of all the factors that must be accounted for in the new system and can be used to measure the success of a project by checking that all the problems and requirements have a corresponding solution.

It is tempting for the analyst to accept a user requirement written by the users without any additional analysis work. Experience has shown that a statement of requirements produced by users will often include detail such as 'I need a terminal linked to a central mainframe' rather than 'I need my data to be up-to-date at all times and I will need to be able to access the data during the hours of nine to five'. It is the analyst's responsibility to make sure that the requirements and problems are stated in logical terms. It is important to have this logical statement of requirements so that the final solution does not become constrained. It must be left to the systems analyst/designer to specify the best solution to fit the users requirements not allowing the user's preconceptions to be carried through to an ill-judged implementation.

The problem/requirement list is initiated in stage one, the survey of the current system. During the analysis of requirements, the problem/requirement list is expanded to include design constraints and requirements for the system auditing, controls and security.

AUTOMATED SUPPORT FOR SSADM

One of the principle features required of the method chosen to become SSADM was that it was designed to be supported by automated support tools. A simple database tool was introduced by the CCTA soon after the introduction of the method to act as a prototype for future support tools. From experience of the use of this tool together with other tools, such as CAD and word-processing software, it was possible to define the desirable features of a software tool to support SSADM. Some of these features are summarized here, in no particular sequence:

- automatic production of documentation,
- assistance in creating and amending SSADM diagrams,
- enforcement of diagram syntax,
- enforcement and help with the rules of the method,
- consistency and completeness checking,
- traceability of specification through to logical and physical design,
- automatic generation of elements of the design,
- presentation of the information in different formats and combinations,
- integration of diagram information with data dictionary information.

There are several software tools to support SSADM and there is a growing number of other tools that can be used to support aspects of SSADM. These other tools are either designed to support other similar methods or are tailorable to a number of different methods including SSADM. They provide varying support to such techniques as data flow diagramming, entity modelling, functional decomposition, relational data analysis, action diagramming and database design. Some provide generation of database definitions and program code. These tools are generally single user, running on IBM PC/AT or compatible hardware, although some multiuser tools are becoming available.

TRENDS IN DEVELOPMENT

Developments in the area of SSADM are driven principally by user experience (ease of use) and the need for automation. Advanced state-of-the-art ideas must also always be considered to ensure better techniques are not ignored. The whole method must remain consistent through any changes made and, being a government standard method, fit in with other standards that have been set.

User experience

Some SSADM projects often develop their own interpretation of the ways in which techniques can be used successfully in their particular environment. Occasionally, these local practices can have wider applicability and developers of SSADM have been keen to introduce well-tried new ideas that have been shown to be beneficial in practice. As well as developing new practices, projects have introduced new forms or pointed out gaps in the method where inadequacies have become apparent.

It is important to take this type of experience into account because it is wasteful for different projects to start again from the beginning when an improvement is required. The benefits of standardization should not be diluted by too many local variants.

An example of a development introduced as a result of user experience is the logical dialogue design element of SSADM. Several different projects perceived a need to be able to model human-computer interactions in the analysis stage of the method and were inventing their own approaches. The experiences gained as a result of this were integrated into SSADM after several pilot uses of the techniques in projects.

The need for automation

It is generally considered to be a fact that methods will become more automated as the technology of software tools increases. The trend towards automation will determine the competitiveness of methods in the future. Methods will be determined by the tools available to support them. Eventually, the method and tool will become synonymous and the manual structured methods will fall into disuse. This means that the development of SSADM must always take into account whether particular ideas will be readily automated or whether they will make automation more difficult. This consideration is often in direct opposition to the wish to enhance the usability of the manual method as it stands currently. If a technique has strict rules of syntax associated with it, the manual use of it will seem arduous; however, a software tool needs to have a large number of such rules defined in order to give the best possible support for the technique.

As a move towards automation, the CCTA commissioned a detailed entity model of SSADM. The production of this model meant that many definitions had to be made tighter and rules had to take the place of guidelines.

COMPARISON WITH OTHER METHODS

SSADM is most readily compared with other methods that employ data flow diagrams as a major technique of analysis and design. These include the Yourdon method[7] and Arthur Young's information engineering method[8]. A brief overview of these methods and a comparison with SSADM follow.

Yourdon method

The Yourdon method is based upon the approach of DeMarco[5]. Data flow diagrams are used to build a number of models of the system required. A logical (essential) view of the required system is developed supported by an entity-relationship diagram, a data dictionary and process descriptions. An implementation-dependent view (implementation model) is developed from the logical diagram by assigning processes to processors and showing how the system will be organized. The data flow diagrams are developed down to a low level of detail. The bottom-level processes that constitute a program or module are drawn together into a program structure which becomes the program design. In addition, certain extensions have been added to the basic notation to cope with realtime control aspects of systems. A controlling process which enables, disables and triggers the transformation processes may be represented by state transition diagrams which form the basis for design. The entity-relationship diagrams are used as the basis for database design.

The main different between the Yourdon method and SSADM is that there is no structure of steps, stages and deliverables and detailed task lists defined in the Yourdon method. A sequence is implied by the way in which each model is developed but it is left to the developer to build project management and review procedures around the techniques.

Another difference is in the approach to process design. In the Yourdon method, the design is derived through successive decomposition of the data flow diagrams. Each bottom-level process is described by a detailed process description or mini-spec. In SSADM, the data flow diagrams are used mostly in the requirements specification; the process definition is taken through to design using the events identified from entity life histories: each event is expanded by a process outline which is subsequently converted into a program specification.

SSADM emphasizes the fact that three different views of the system are developed and compared in analysis whereas the principal technique that is used throughout the Yourdon method is data flow diagramming. The entity-relationship diagram developed in the Yourdon method is not given as much emphasis as the data flow diagrams. The Yourdon data dictionary is defined in terms of the contents of data flows and data stores whereas in SSADM the data is defined with reference to the logical data structure.

Arthur Young information engineering method (AY-IEM)

Arthur Young information engineering method (AY-IEM)[8] is based upon the concepts described by James Martin[9]. Within this basic framework, Arthur Young have developed a detailed method which requires the use of their software tool, information engineering workbench (IEW), to implement the concepts fully.

The method consists of a number of steps and stages leading from strategy to construction. Emphasis is placed upon the data model as the foundation for good system design. The data model developed is similar to the logical data structure of SSADM. Data flow diagrams are fully integrated with the data model. The data flow diagrams are also cross-referenced to a function decomposition diagram which effectively summarizes the hierarchy of processes within the data flow diagrams. The detail of processing is defined in terms of action diagrams.

Both AY information engineering and SSADM contain steps and stages. SSADM has detailed task lists with define inputs, outputs and activities whereas AY information engineering concentrates upon stressing the aims and objectives of each step and stage, leaving more freedom to choose the most appropriate way of achieving the objectives. Information engineering concentrates more upon providing a set of techniques and tools, together with a project framework and allowing the developer to decide upon the best way of combining them to meet the objectives. This means that there are no specified inputs and outputs of steps and no forms to fill in. The central database, or encyclopoedia, of the tools contains all the necessary information to support the developer.

Other differences between the two methods include the fact that SSADM uses a third view in analysis provided by the entity life history technique and information engineering has action diagrams and structure charts to define the structure of the processes.

CONCLUSION

SSADM has been used in a large number of projects principally in the area of government data processing systems. Several of the larger projects are now live and their implementation was considered to be a success. Experience shows that the method has improved the quality of systems analysis and design. The role of a central group in introducing, promoting, controlling and supporting SSADM has been a major contributor in ensuring its success.

References

1 **Longworth, G and Nichols, D** *The SSADM Manual* National Computer Centre, Manchester, UK (1987)

2 **Downs, Clare and Coe** *Structured Systems Analysis and Design Method – Application and Context* Prentice-Hall, (1988)

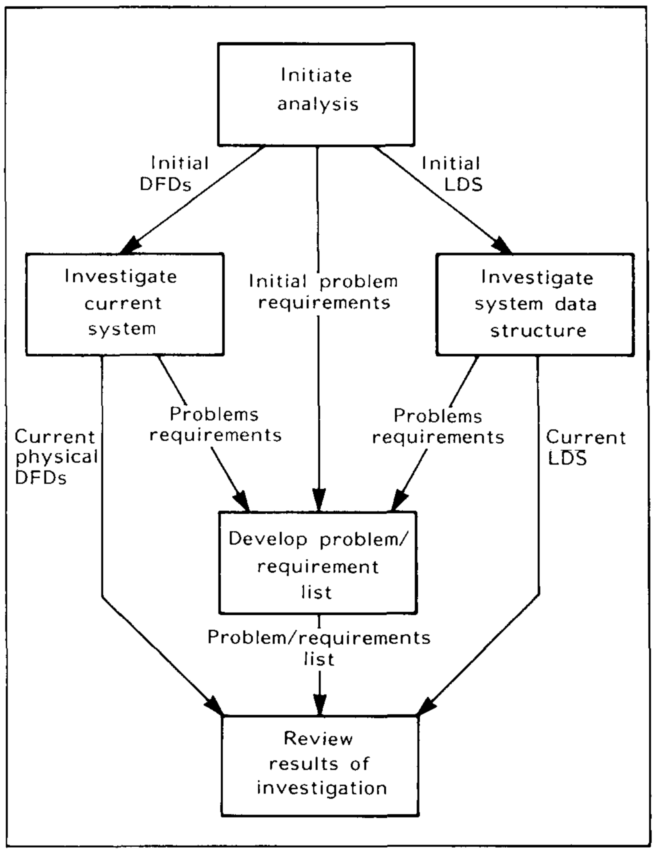
3 **Yeates, D** *Systems Project Management* Pitman Publishing Ltd, London, UK (1986)

4 **Jackson, M A** *Principles of Program Design* Academic Press, London, UK (1975)

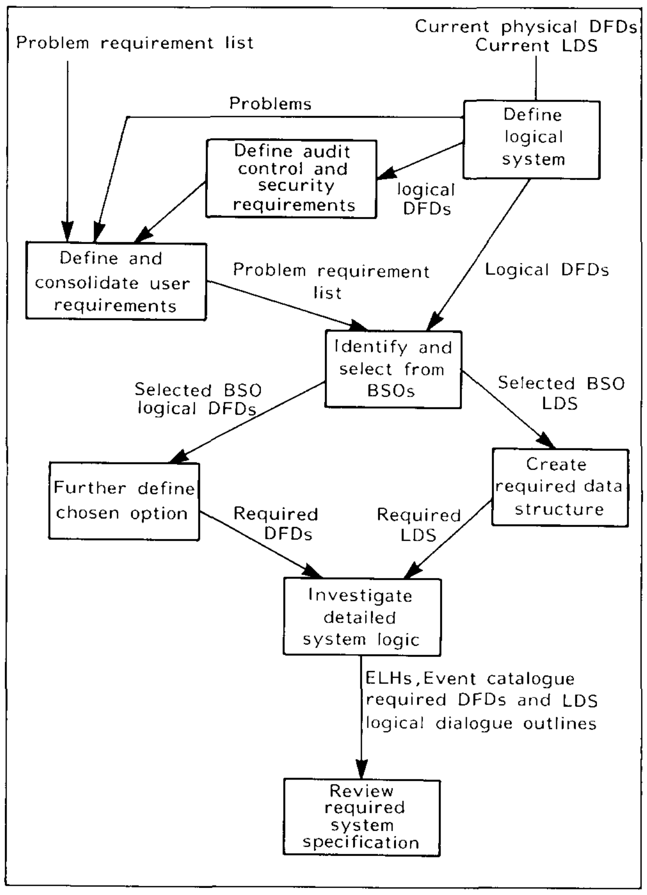
5 **DeMarco, T** *Structured Analysis and System Specification* Prentice-Hall, Englewood Cliffs, NJ, USA (1979)
6 **Codd, E R** 'A relational model of data for large shared data banks' *Commun. ACM* Vol 13 No 6 (June 1970) pp 377–387
7 *Yourdon Method,* Yourdon Europe, 15–17 Ridgmount Street London WC1 7BH, UK
8 *Arthur Young Information Engineering Method,* Arthur Young, Rolls House, 7 Rolls Buildings, Fetter Lane, London EC4A 1NH, UK
9 **Martin, J** 'Information Engineering' Savant Research Studies, 2 New Street, Carnforth, Lancs LA5 9BX, UK (1986)
10 **Gane, C and Sarson, T** *Structured Systems Analysis: Tools and Techniques* Prentice-Hall Englewood Cliffs, NJ, USA (1979)

Annex A

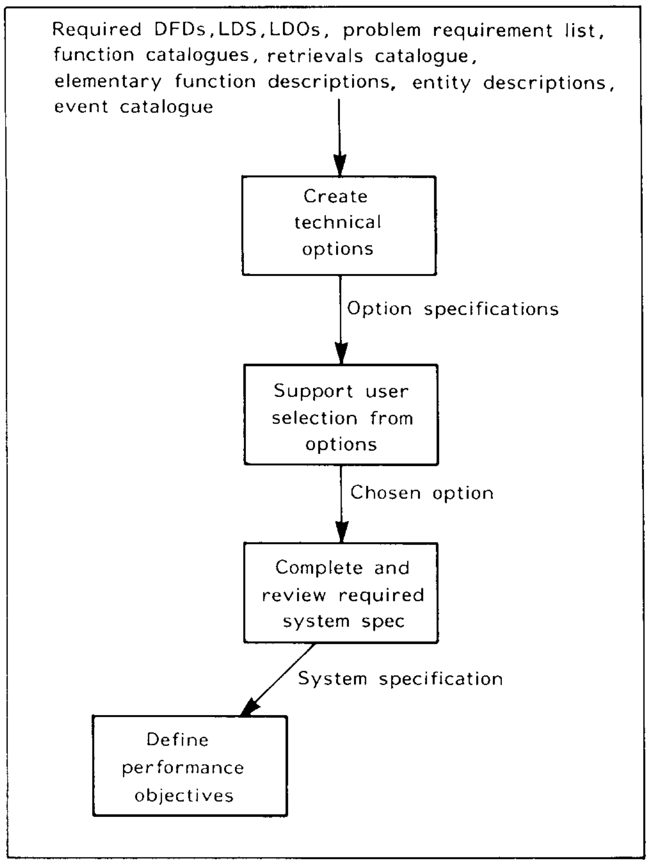
Stage 1


Initiate analysis
Initial DFDs
Initial LDS
Investigate current system
Initial problem requirements
Investigate system data structure
Current physical DFDs
Problems requirements
Problems requirements
Current LDS
Develop problem/ requirement list
Problem/requirements list
Review results of investigation

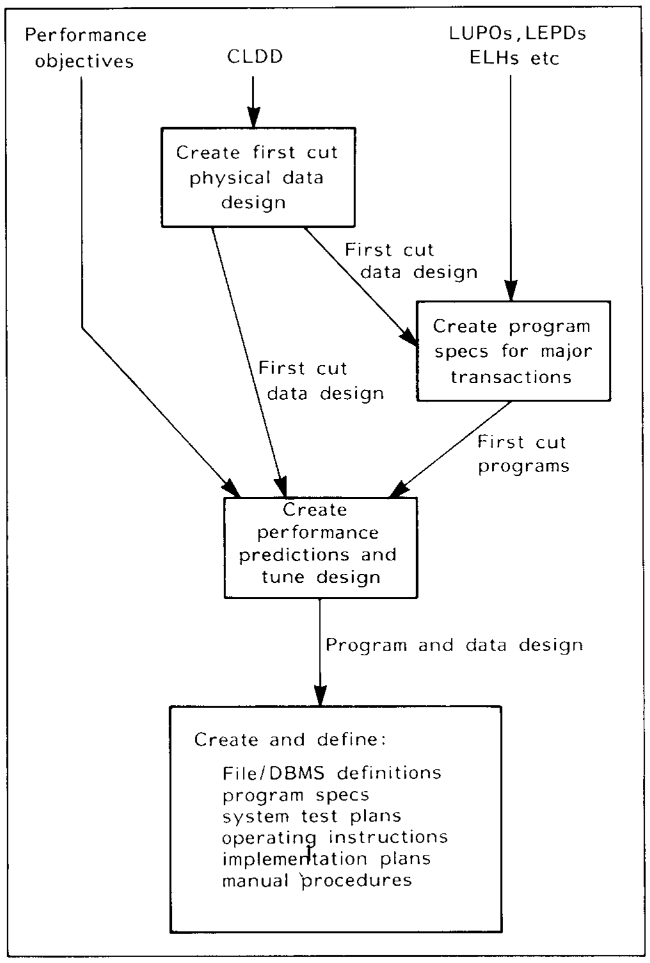
Stage 2


Problem requirement list
Current physical DFDs
Current LDS
Problems
Define logical system
Define audit control and security requirements
logical DFDs
Define and consolidate user requirements
Problem requirement list
Logical DFDs
Identify and select from BSOs
Selected BSO logical DFDs
Selected BSO LDS
Further define chosen option
Create required data structure
Required DFDs
Required LDS
Investigate detailed system logic
ELHs, Event catalogue required DFDs and LDS logical dialogue outlines
Review required system specification

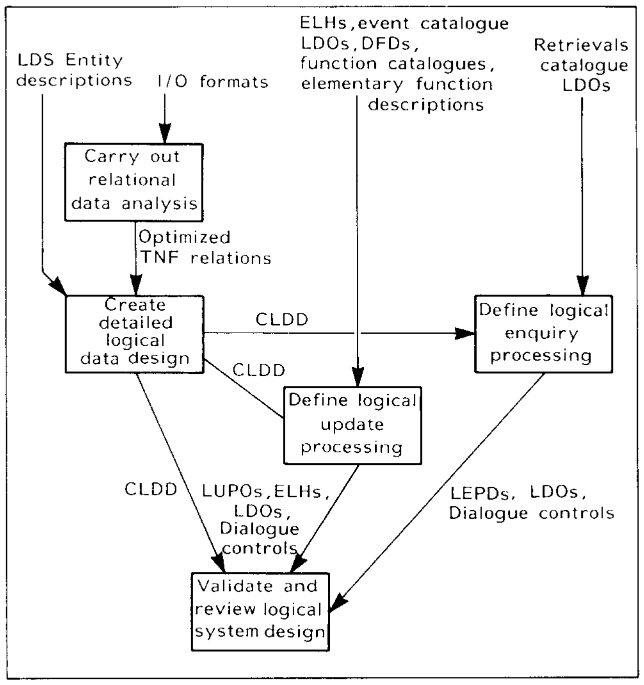
Stage 3


Required DFDs, LDS, LDOs, problem requirement list, function catalogues, retrievals catalogue, elementary function descriptions, entity descriptions, event catalogue
Create technical options
Option specifications
Support user selection from options
Chosen option
Complete and review required system spec
System specification
Define performance objectives

Stage 6



Stage 4 *Stage 5*


LDS Entity descriptions
I/O formats
ELHs, event catalogue LDOs, DFDs, function catalogues, elementary function descriptions
Retrievals catalogue LDOs
Carry out relational data analysis
Optimized TNF relations
Create detailed logical data design
CLDD
Define logical enquiry processing
CLDD
Define logical update processing
CLDD
LUPOs, ELHs, LDOs, Dialogue controls
LEPDs, LDOs, Dialogue controls
Validate and review logical system design