

8. Yourdon Systems Method (YSM).

8.1. Introduction

This chapter gives a brief outline of the 1993 version of the method. Unlike earlier versions, YSM now uses terms like "real-world" and "subject-matter" when talking about systems modelling. It is to be wondered how much these changes have been influenced by the work of Michael Jackson whose JSD method (1983) used these notions to escape from the limitations of top-down development. For a full coverage see the book *Yourdon Systems Method*, by Yourdon Inc.

Originally, YSM was very similar to STRADIS (see Chapter 9), Yourdon, Gane and Sarson all being one-time colleagues. However, unlike STRADIS, YSM in its later versions has attempted to move away from pure functional decomposition towards something called *event partitioning* which is claimed to be neither top-down, nor bottom-up but middle-out. However, because the method still uses the time-honoured techniques of data-flow diagrams and entity-relationship models it is difficult to see how the shackles of functional decomposition have been shed. YSM now claims to be a *third generation* method. However, we must note the essential difference between YSM and JSD (another 3GM) which is that unlike YSM, the Jackson approach has eschewed completely any notion of functional decomposition.

The method requires the analyst first to draw a context-diagram (top level DFD) of the problem indicating the sources, sinks and boundaries of the system. Then, after interviewing the various users, a list of the events to which the proposed system must respond is drawn up. Following this, the documentation of the system is completed using traditional techniques such as entity modelling, normalisation, DFDs, structured English etc.

YSM uses a defined set of development tools, procedures, models and structured techniques to build predominantly graphic system models. As a formal discipline it owes most to the early work of Constantine, Yourdon and DeMarco. The stages of analysis and design activity are:

8.2. Role of models in YSM

In version 3 of YSM (1993) an attempt has been made to apply a more rigorous theory underlying the definition of the various models than before. Models are said to be the foundation of Yourdon's method. In YSM, a model attempts to abstract the main features of what is under examination and then present those features in a useable way.

Two main models are made in YSM: the *enterprise model* and the *system essential model*. For these, there will likely be further models: *the essential model* and the *implementation model*. An essential model shows policy whilst an implementation model describes technological considerations.

Note, that the amount of rigour employed in a development project is very dependent on the nature of the project. Safety critical systems require a very rigorous approach to ensure the models are correct.

The various models in YSM are constructed using established techniques,

Notes

including:

- data-flow diagrams
- entity relationship models
- state transition diagrams
- mini-specifications
- structured English
- etc. etc.

8.2.1. 'Enterprise' and 'System' in YSM

An enterprise is an economic unit that is resourced and managed as a unit. Therefore, an enterprise could be an entire business organisation or a division of the company.

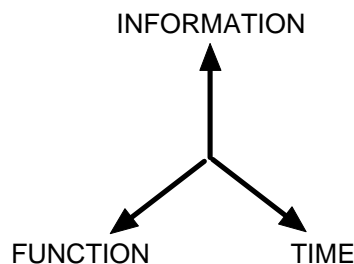
A system is said to be a collection of information and operations that are organised to meet a specific problem. Computers, programs, files, manual procedures and operations and support facilities could all be part of a system.

Whilst an enterprise has inputs and outputs that it processes, it is not a system. An enterprise has a longer life-span than a system. Within an enterprise, several systems may be developed, used and replaced. Furthermore, an enterprise usually supports several systems that are grouped to achieve a strategic goal of the business. Enterprises will provide infrastructure to allow those systems to work together.

8.2.2. Viewpoints of the system

Various graphical modelling techniques are used to address different views of the system being developed. In YSM there are three main orthogonal viewpoints of a system:

1. **Function** what the system *does*.
2. **Time** what happens and when.
3. **Information** what *information* is used by the system.



The orthogonality means that each of the questions in the three viewpoints can be considered to be independent of the others. This leads to a three-dimensional description of the system:

The event list is used to show anything that happens to which the system must respond. This occurs in the **time dimension**. The **information dimension** is described by entity relationship diagrams. DFDs are used to describe the **function dimension**.

8.3. Outline of method

Notes

A system is designed by constructing a series of models which aim to capture the relevant information about the enterprise in which the system will reside and the system itself. The three dimensions of the system are used to help identify this information.

Any model can be considered complete when it can be realised in some conceivable architecture. If the model does not allow this to be done then further work is required.

The models to be constructed are organised around the enterprise and the system itself hence:

- Enterprise models
 - Enterprise Essential Model
 - Enterprise Resource Library
- System models
 - System Essential Model
 - System Implementation Models
 - System Processor Implementation Model
 - System Software Implementation Model
 - System Code Implementation Model
 - Hardware Implementation Model
 - Manual Implementation Model

8.4. Enterprise Essential Model

This model uses some of the tools employed in the System Essential Models (e.g. E-R models). Currently, this model comprises two aspects.

1. Enterprise information aspect: this describes the information used by the enterprise.
2. Enterprise performance aspect: here we describe event frequencies and the number of occurrences of information aspect components.

Strategic planning and initiation of system development projects require this model to be in place as it is claimed that it allows the possible impact on other systems within the enterprise to be visualised and the most cost-effective solution to be identified.

Constructing this model will need

8.5. System Essential Model

The System Essential Model exists to represent the underlying policy of the system. This policy is that which must be carried out by the system regardless of the implementation chosen. Thus, this model is effectively a statement of requirements for the system. It is supposed to focus on the real-world subject matter of the system and as such should disregard technological issues. We can think of it as a logical model rather than a physical model. Therefore, a model of a payroll system would include employee numbers and hours-worked as system inputs, but would not

Notes

include time cards as these are implementation-specific it would be possible to have a payroll system without time cards.

8.6. Further reading

For more information on YSM see Yourdon Inc, *Yourdon Systems Method: Model-Driven Systems Development*, Prentice Hall International Editions, 1993 (ISBN 0-13-285818-5).

9. Gane & Sarson’s STRADIS

9.1. Overview

STRADIS stands for *Structured Analysis and Design of Information Systems*. Structured design is concerned with the selection and organisation of modules and interfaces that would solve a pre-defined problem. Gane and Sarson’s method has been developed over a number of years and attempts to provide a method of specifying requirements as a front-end to structured systems design.

In essence the methodology uses a wide variety of techniques which are found in other methods and concentrates, as do others, on functional decomposition and the use of data flow diagrams. It is concerned mainly with systems analysis, to a lesser extent with systems design and hardly at all with implementation.

The main phases of the method are:

- The initial study;
- The detailed study;
- Defining and designing alternative solutions;
- Physical design.

The main rationale or justification is:

- Wide applicability in the area of information systems;
- Most relevant to a situation in which there is a backlog of systems waiting to be developed and insufficient resources to devote to all potential new systems;
- Incorporates a wide variety of established techniques.

9.2. The Initial Study

Essentially a feasibility study which aims to elicit the monetary costs and benefits of a system proposal(s). It will usually involve the construction of an overview data flow diagram and its interfaces and an estimate of the times and costs of proceeding to a detailed investigation together with a broad-range estimate for final system development costs.

It is not as resource intensive as a traditional feasibility study, nor does it include a review of alternative approaches to the proposal. These and the decision to proceed to implementation are included in later stages of the method.

9.3. The Detailed Study

An examination of the existing system(s) in detail; containing the following stages:

- A definition of the user community for a new system, that is, the names and responsibilities of senior executives, the functions of affected departments, the relationships among affected departments, the

Notes

descriptions of clerical jobs that will be affected, and the number of people in each clerical job, hiring rates and natural wastage rates.

- Users of the system are identified at three levels:
 1. The commissioners (senior management who asked for the system proposal);
 2. Middle managers of the departments affected;
 3. End users (who will actual work with the system).

A logical model of the current system is produced The users are interviewed and a logical data flow diagram is prepared of the existing system. This DFD will be used to deduce the system boundary and hence extends beyond the current system. DFDs at various levels (similar to Yourdon) are produced and when the functions become well defined, process logic is expressed using an appropriate logic representation such as:

- decision trees;
- decision tables;
- structured English.

The detail of the DFDs (data definitions) and the process logic is entered into a data dictionary using single meaningful names for data flows and data stores;

A statement of increased revenue/avoidable cost/improved service that could be provided by an improved system is produced. The cost/benefits of the initial study are now refined. Assumptions, present and projected transaction volumes, estimated quantities of stored information and financial estimates of benefits are produced;

Account of competitive/statutory pressures (if any), including: the system cost and a firm cost/time budget for the next phase. The resultant report(s) are presented to management and a decision will be made to stop or proceed to the next stage.

9.4. Defining and Designing Alternative Solutions

The initial study elicits the organisational objectives such as increased revenue, lower cost or improved service; these are elaborated to produce system objectives which should be specific and measurable. The objectives and the DFD of the existing system are used to produce a logical DFD of the desired system.

Alternative implementation designs are produced which meet a variable selection of the identified system objectives. The alternatives cover 3 categories:

1. Low-budget, fairly quick implementation which may not initially meet all of the objectives;
2. Mid-budget, medium term version, which achieves a majority of the objectives;
3. Higher budget, more ambitious version which meets all of the objectives.

A report from this stage is presented to the relevant decision makers who would make a commitment to one of the alternatives. The report would contain:

- A DFD of the current system;

- The limitations of the current system, including cost/benefits estimates;
- The logical DFD of the new system;
- For each of the identified alternatives:
- The parts of the DFD that would be implemented;
- User interface (terminals, reports, query facilities, etc.);
- Estimated costs and benefits;
- Outline implementation schedule;
- Risks involved.

9.5. Physical Design

The chosen alternative is refined by a design team in a number of possible parallel activities:

- The detail of the DFD is produced including process logic, error and exception handling. This will lead to the contents of the data dictionary being completed together with report and screen formats;
- The physical files (database) are designed based on the data store contents defined from the logical DFD;
- Normalisation of data stores is carried out to consolidate and simplify them into logical groupings. The actual process of mapping and the design of the physical files is not defined by Gane and Sarson;
- Derivation of a modular hierarchy of functions from the DFD. Here the notions of transform centred and transaction centred systems are used. Figures 8-2 and 8-3 illustrate the basic differences between these.

Figure 8-2: Transform Centred System

Figure 8-3: Transaction Centred System

Once one of these high-level functional hierarchies is defined, then the detail of the modules in the hierarchy and the communication between them can be done. These tasks together with a definition of clerical tasks will be pursued to the level of detail when firm costs can be produced for:

- Development of the system (manpower and hardware);
- The computer configuration required;
- Data communication costs;
- Production of documentation and user training;
- The time of the users who interact with the system;
- Systems maintenance costs.

9.6. Subsequent Phases

subsequent phases are not clearly defined by Gane and Sarson in current literature, although continuing work is proceeding to develop the following tasks:

- Implementation planning, testing and user acceptance;
- Concurrent development of the application programs and the database/data communication functions;
- Loading the database;
- Testing procedures;
- Performance evaluation;
- Live running and tuning;
- Evaluation and review;
- Maintenance, change requests, analysis etc.

9.7. Exercises

1. From the above description, where would you say the major emphasis of the method is?
2. What are the advantages and disadvantages of producing DFDs of the existing system?
3. Describe how you might determine whether a system is essentially a transform centred system or a transaction centred system.
4. How might you validate a DFD?
5. Comment on the alternative solution strategy in the light of the claimed rationale for the method.
6. Describe briefly, the following process logic representations:
 - Decision tables;
 - Structured English;
 - Decision trees.

10. Information Engineering

10.1. Overview

The origin of the term Information Engineering is shrouded in mystery. A number of claims are made to its authorship (we would refer you to Finkelstein³) and a number of organisations produce documentation, tools and training under this heading. The most well-known is James Martin Associates method and this is what is briefly summarised here.

Information engineering (IE) is viewed as a framework within which a variety of techniques are used to develop good quality information systems in an efficient way.

The main stages of the method are:

- Information Strategy Planning;
- Business Area Analysis;
- Business System Design;
- Technical Design;
- Construction;
- Transition;
- Production.

Not all systems require a rigorous adherence to the above stages. The main rationale or justification for the method is:

- In order to benefit the organisation, the main problem is viewed as the provision of information, not the construction of software;
- The basis of the information system is data because is considered to be more stable than processes or procedures;
- The most appropriate means of communication is through diagrams; each deliverable is supported by diagrams. This helps to ensure that users requirements are understood;

Much of the method can be automated. There are two different tools to support IE: IEF (Information Engineering Facility) produced by JMA with Texas Instruments, and IEW (Information Engineering Workbench) produced by KnowledgeWare and Arthur Young.

Figure 9-1: Information Engineering Building Blocks

10.2. Information Strategy Planning

This is a joint activity of user management and information systems staff, and involves the performance of four basic tasks:

1. Current Situation Analysis. An overview of the strengths and weaknesses of the current systems;

³ Clive Finkelstein, *An Introduction to Information Engineering: From Strategic Planning to Information Systems*, Addison-Wesley, 1989.

2. Executive Requirements Analysis. Managers state their objectives, information needs and their perception of priorities and problems;
3. Architecture Definition. An overview of the area in terms of information architecture. What information, how it is organised and distributed, where it is used, and what technical support it has;
4. Information Strategy Plan. The determination of subject areas, business functions and their interactions, together with a business evaluation and information strategies.

The main activities and techniques to support the above are:

- **Business Strategy Analysis** techniques to identify future developments which may add to, or change, business system requirements, to establish business priorities and to recommend opportunities for exploiting information technology;
- **Structured Interviews** formal approach to planning, conducting, analysing and confirming interviews. Emphasis is on understanding the business objectives, the activities carried out to achieve these objectives and the factors which impact on their achievement. From this understanding, the information needs of the business can be derived;
- **Data Modelling** a method of representing the various types of data used within the business and the relationships between them. The objective is to combine diagrams with supporting text to provide a rigorous and unambiguous documentation common to users, development staff and computer system;
- **Functional Modelling** a method of representing the various functions carried out by the business in order to achieve the objectives. Like data modelling, diagrams and text are used to provide a common documentation vehicle;
- **Function Dependency** a method of analysing/documenting how the various business functions inter-relate through the information which flows not only between them but also from and to the outside world;
- **Cluster Analysis** can be applied to different matrices and conducted using more than one technique. In Information Strategy Planning it is used to combine functions and data into groups which fit naturally together. These groupings form the basis for establishing business areas;
- **Current Systems Analysis** In Information Strategy Planning this is limited to a brief examination of the existing systems sufficient to identify the degree to which the business objectives are currently supported and hence help determine the priorities for development. Also necessary as a start point for planning the transition path from where we are today to where we need to be to best support the business;
- **Contingency Analysis** establishes:
 - (a) the important parameters of the company (e.g. sales, profits, products, markets, locations) and
 - (b) the most important internal and external factors which will influence the development of the company. The possible impact of each of the factors on each of the parameters is then considered to help ensure flexibility and responsiveness of systems and databases;
- **Reports/Presentations** largely self explanatory but it should be noted that the emphasis with IE is on gradually adding to and improving the understanding of the business rather than the production

of a limited number of major documents.

Notes

10.3. Business Area Analysis

For an identified business area within the scope of the information systems development strategy, a detailed study is carried out of its data, its business functions and the information required to fulfil those functions. This leads to identification of entity types and of specific business processes and their information inputs and outputs. These are analysed in detail and their names, interactions, meanings, quantities, rules and business algorithms documented. An important feature is the maximum involvement of end users in the specification of requirements, priorities and facilities. From this information a detailed statement of the business requirement for information systems in the business area is produced.

By the end of business area analysis a business area description has been prepared, showing the processes performed in the area and the entity types, relationships and attributes found in the area, together with their usage patterns in the business processes. The properties of all these objects are documented. They provide greater detail for the information architecture and indicate information needs and priorities within the business area.

From these details it is possible:

- to identify the broad nature of likely computer support required for business processes;
- to define the scope of one or more design areas within which business systems can be designed;
- to prepare a work programme and resource estimates for the design areas.

All the information is present about the business and its users requirements which is necessary to select particular business processes for computer support and to design the computer systems and the data structures needed to give that support. The only discussions with users in the subsequent stage should concern the design of the user interfaces.

The tasks of Business Area Analysis may be summarised:

- Entity and Function Analysis;
- Interaction Analysis;
- Current Systems Analysis;
- Confirmation (cross-checking);
- Planning for design.

The main activities and techniques to support the above are:

- **Steering Committees/Working Committees/User Groups** as with conventional method but requiring a generally far higher level of user commitment/involvement;
- **Reports/Presentations** as per Information Strategy Planning.
- **Cluster Analysis** as in Information Strategy Planning, except that in Business Activity Analysis, processes/entity types are clustered to identify logical systems for the Business Systems Design stage;
- **Process Action Diagrams** an action diagram was originally just a simple diagram to represent the constructs of a structured program. The

same techniques can be used at higher levels within IE as a clear and concise method of documenting the steps involved in the various business processes;

- **Current Systems Analysis** the analysis of existing documentation, computer systems etc. to enable models to be built of the current system. In full IE, these models will be used merely to check the accuracy of the models built from the interviewing. In the hybrid method this analysis is the start point for the construction of business models because the scope of the interviews will be limited to the enhancement required;
- **Structured Interviews** as for Information Strategy Planning, except that in Business Area Analysis, we are concerned with the processes (what the business does) which comprise each function;
- **Entity Modelling** an entity type is something of interest to the business about which data can be stored. Diagrams and text are used (in a similar manner to data modelling in Information Strategy Planning) to document the various entity types and their relationships with each other;
- **Process Modelling** as for function modelling in Information Strategy Planning, but at the next level of detail down i.e. diagram/text to document the various business processes which are involved in each function;
- **Process Dependency** as for function dependency in Information Strategy Planning, but showing how the various business processes relate to each other and with processes, other organisations etc. external to the function. This is a very important step in the correct and complete definition of the business requirements;
- **Process Logic Analysis** improves the definition of a process by detailing its actions upon entities and relationships and ensures that the entity model can, in fact, support all the processes. Also provides a starting point for the formation of business system procedures to support the processes;
- **Entity Life Cycle Analysis** the analysis of what can happen during the life of an entity of one type i.e. from creation to termination. The objectives are to reveal undiscovered processes, improve the entity type and process definitions and to determine which processes are allowable for each state of the entity type. Also acts as a starting point for the specification of rules governing the execution of procedures in a business system;
- **Stability Analysis** techniques to identify business changes which may affect the stability of the various models, the probability of those changes and the cost of allowing for the changes versus the risk of not doing so;
- **Data Analysis** techniques to reduce data to its most basic, most flexible form to minimise the impact of business changes and reduce the duplication of data.

10.4. Business System Design

For the whole or a major part of the business area analysed, the facts gathered during analysis are used to design a system to meet the identified business requirements. The design includes all those parts of the system directly relevant to its users including transactions, dialogues and controls. It is kept as independent as possible of the technology to be employed in

implementation. Prototyping techniques, using fourth generation languages, may be used to replace many of the tasks traditionally gone through in this stage.

An important objective of this stage is that it should complete the system design to the extent possible without pre-judging technical issues. It is heavily user orientated and requires agreement of the users on the ways in which they will interact with the system.

The final product from business system design is a business system specification showing, for each business process, the consolidated documentation of information flows and user procedures and for each computer process, the dialogue design, screens, reports and other user interfaces and adjustments to the data usage patterns. From this, a detailed scoping of the intended computer systems is prepared together with a work programme and resource estimates for the next stage.

A key technique in this stage involves representing the logic and data usage of procedures in the form of structured action diagrams. These can be translated directly into fourth generation language statements. Once this is done, all aspects of the system which relate directly to the user should be defined and stable, and sufficient information should exist to finalise estimates for, and to complete, technical design.

The main activities and techniques to support the above are:

- **Preliminary Data Structure Design** a design for the organisation of all the data needed to support a business area. Although still only a logical representation, the target DBMS is considered and data is shown in the form of record types and the linkages between them;
- **Procedure Design** determining how an individual process will be carried out by one or more procedures and the design of fallback procedures;
- **Dataflow Diagrams** a method of representing the various procedures required to support a given process, the physical data which flows between them and the data stores required;
- **Code Design** deciding what fields are to be encoded, the style and format most appropriate in each case and defining who will create/maintain the codes;
- **System Structure Design** techniques to determine how the various procedures will be linked together, security, control and audit procedures required etc;
- **Prototyping** a technique primarily for building a quick and rough version of a desired system or parts of the system. The prototype illustrates the system to both users and designers, allowing them to see flaws and invent ways to improve the system. Such features as security, auditability, recoverability, ability to handle large volumes of transactions/users etc. are not normally included in a prototype. The prototype is usually regarded as a throw-away, but can be built such that it can be further enhanced to become the production version;
- **Data Structure Refining** review of the data structure to consider any additional requirements for privacy, integrity etc. Refinements are not for performance reasons at this stage.
- **Man-machine Interface Design** various considerations aimed at providing the most appropriate interface catering for differing user skills, roles, expectations etc.;

- **Dialogue Design** techniques, considerations and rules to ensure that procedures are based upon an efficient and usable set of interactions with the user;
- **Procedure Action Diagrams** a means of expressing the business system design at a level which includes: data access, significant conditional actions, reference to fields and use as a starting point for system building either in 3GL or 4GL;
- **Confirmation techniques** Various techniques aimed at ensuring the completeness, correctness and usability of the design;
- **Technical Design Planning** plan, costs/benefits, resources required, duration etc. for the next stage of development i.e. Technical Design.

10.5. Technical Design

For the computerised aspects of the business systems specified, the facts gathered during analysis are used to design those parts of the system which are dependent upon the computer technical environment. This is carried out in sufficient detail for construction and operation to be adequately costed. This design includes data storage structures, computer programs, operational procedures and interfaces. The level of detail in the design is dependant upon the selection of implementation vehicle, e.g. system generators have much of the technical structure predefined.

The aims during this stage are to define effective computer systems to support the selected business processes and to develop good (+20%) estimates of costs and timescales for construction and transition in terms of manpower and computer equipment.

The end result of technical design is a technical specification containing database designs, procedure designs in the form of action diagrams, and the technology dependant details of the system design. These include batch runs, finalised conversation flows and definition of programming work units. The specification also includes the technical architecture and standards employed by the system - the hardware and software environment selected, its mode of use and specific standards and conventions proposed. Finally it identifies the content of the construction and transition stages and gives a work programme and resource estimates for these stages.

This technical specification provides a stable design which meets the functional and performance objectives and is insensitive to likely business and technical changes.

The main activities and techniques to support the above are:

- **Physical Database Design** starting with the refined data structures and transaction volumes, adjustments are incorporated to allow for performance criteria, recovery/backup requirements, hardware constraints etc;
- **Program Design** the finalisation of dialogue flows etc. and the translation of the various Business System Design outputs into units of program work appropriate to the target hardware/software environment;
- **Operational Procedure Design** defining the archiving and recovery/backup procedures, audit trail provisions, hardware and system software implications.

10.6. Construction

For each phase identified during design, a system is put together. This includes installation of equipment, establishing files, setting up procedures and specifying, coding and testing programs. The aim in the construction stage is to develop a system, as defined in the technical specification, which meets the targets of timescale and budget, is of an acceptable quality, and which contains all necessary operating and user procedures.

This stage can be regarded as complete once the defined acceptance criteria for the application are met satisfactorily.

The two main activities of this stage are:

- **System Generation** construction of all necessary procedures, files etc.;
- **System Verification** generation of system test data, performance of system tests, generation of acceptance tests data, performance of acceptance tests and obtaining approval.

10.7. Transition

Transition is the phased replacement of existing procedures and files with the new system and data structures. It is governed by the transition plan, including a work programme and resource estimates, which is normally finalised in parallel within the construction phase, although it is not really dependant on the outcome.

Transition can be regarded as successful when the system operates for a specified period within defined tolerances as regards performance, error rate and usability, and passes its post-implementation review.

The main activities of transition are:

- **Preparation** prepare transition schedule, train users, and install new local hardware;
- **Installation of new software** perform conversion and execute trial runs;
- **Final acceptance** agree terms and fully transfer to the new system;
- **Fanout** install at appropriate location(s);
- **System variant development** identify requirements, revise analysis and design, and perform construction and transition where a particular location requires a variance from the norm.

10.8. Production

Production is the successful operation of the system, with tuning and modification as necessary, until eventually the transition stage in some other project replaces the systems built in this project.

The main objectives during production are to maintain service levels and functional performance during the lifetime of the system and to respond promptly and effectively to changes in business requirements.

As part of production we would expect:

- System evaluation, a measurement of benefits and costs, and

Software Systems Planning & Design

Notes

comparison with design objectives;

- Tuning, by monitoring performance and tuning software and databases as necessary;
- Maintenance, including bug correction and modification.

10.9. Exercises

1. From the above description, where would you say the major emphasis of the methodology is?
2. Which of the activity oriented techniques lend themselves to automation? What tools have you used for any of these techniques?

Describe the key features of the method in less than 100 words (perhaps in the form of a sales blurb).

11. Structured Systems Analysis and Design Method (SSADM)

11.1. Overview

SSADM was developed jointly by the CCTA (the IT arm of H.M. Government and formerly known as the Central Computer and Telecommunications Agency) and Learmonth and Burchett Management Systems (LBMS). It has been a mandatory Government standard since 1983, and has been adopted commercially by such diverse organisations as I.C.I. and the Metropolitan Police.

The method covers feasibility study, systems analysis and systems design. It uses established tools such as:

- Data flow diagrams;
- Logical data structures for entity modelling;
- Entity life history;
- Normalisation to third normal form (TNF).

It also concentrates very heavily on documentation and in particular the provision of pre-printed documents. The detailed rules and guidelines are found in the SSADM Manual produced by the National Computer Centre (NCC) in 1986. The method is now into version 4.0 with associated project planning and management support given by the PRINCE method.

SSADM contains three phases subdivided into stages:

1. The feasibility phase, containing:
 - the problem definition stage;
 - the project identification stage;
2. The analysis phase, containing:
 - analysis of the current system stage;
 - specification of requirements stage;
 - selection of technical options stage;
3. The design phase, containing:
 - detailed data design stage;
 - detailed procedure design stage;
 - physical design stage.

Each of the stages is subdivided into tasks using appropriate techniques and tools. These are now described.

11.2. Feasibility Phase

11.2.1. The Problem Definition Stage

The tasks for this stage are identified as:

- Initiate the feasibility study;
- Create a current system overview;
- Create an overview data structure;
- Develop a logical system overview;
- Consolidate the problem/requirements list;
- Review the problem definition.

The techniques and tools used for this stage are:

- Data flow diagrams to give a physical and logical overview.
- Logical data structure overview;
- Producing a problem/requirements list;
- System boundary agreement;
- Quality assurance review.

11.2.2. The Project Identification Stage

The tasks for this stage are identified as:

- Cost benefit analysis;
- Terms of reference;
- Identification of project options;
- Creation of outline project specifications;
- Evaluation of project options;
- Formalisation of feasibility study report.

The techniques and tools used for this stage are:

- Data flow diagrams;
- Logical data structure;
- Quality assurance review.

11.3. Analysis Phase

11.3.1. The Analysis of the Current System Stage

The current operational system (if there is one) is investigated and problems are identified. The tasks for this stage are identified as:

- Initiating the analysis;
- Carrying out the detailed investigation;
- Creating the current data flow;
- Creating the ideal current data flow;
- Identifying problems and new needs;

- Specifying current services in agreement with the user.

The techniques and tools for this stage are:

- Traditional interviewing, questionnaires, sampling, observation and studying records for information gathering;
- Data flow diagramming;
- Data structure diagramming (entity modelling).

11.3.2. The Specification of the Required System

The logical view of an ideal system is extended to include the new requirements of all users including those involved in control and audit. The tasks for this stage are identified as:

- Defining the logical system;
- Defining the security, control and audit requirements;
- Defining and consolidating the user requirements;
- Identifying and selecting from business system options;
- Defining the chosen option in detail;
- Creating the required data structure;
- Investigating the detailed system logic;

11.3.3. Reviewing the required system specification.

The techniques and tools used for this stage are:

- Data flow diagrams based on the system boundary;
- A data store/entity cross reference;
- Elementary function descriptions;
- Entity life histories (ELH) using the Jackson notation;
- ELH error handling narratives;
- Entity descriptions;
- The event catalogue;
- Function catalogues;
- Input/output description;
- The logical data structure;
- Logical design volumes;
- Logical dialogue outlines;
- Problems/requirements list;
- Screen formats.

11.3.4. Selection of Technical Options Stage

The activities here will vary according to the size of the project. It may include high level decisions such as centralised or distributed systems, or low level such as siting of terminals or other ergonomic factors. This includes the following tasks:

Notes

- The creation of technical options;
- User s selection of above;
- Completion and review of the required system specification;
- The definition of design objectives.

11.4. Design Phase

11.4.1.Detailed Data Design Stage.

The purpose of this stage is to define in detail the data and data relationships and ensure that this model supports the processes. It includes the following tasks:

- Carrying out a full data analysis;
- Creation of a detailed logical data design.

This involves the use of the following techniques:

- Normalisation to third normal form using a document driven approach (bottom up);
- Entity modelling to obtain a logical data structure (top down).
- Producing a composite logical design (CLD) of both of the above and storing the results in a data dictionary.

11.4.2.Detailed Procedure Design Stage

This stage involves the matching of the required user functions and the composite logical design to ensure that it will meet the system requirements. This stage may involve the use of prototyping. It includes the following tasks:

- Definition of logical enquiry processing;
- Definition of logical update processing;
- Validation and review logical systems design.

11.4.3.Physical Design Stage

This stage is concerned with the production of a plan for building and testing the system, program specifications, operating procedures and file layouts or database definitions. The tasks at this stage are:

- Creation of first cut physical data design;
- Creation of program descriptions for major transactions;
- Creation of performance predictions and tuning of design;
- Creation of file/database definitions;
- Completion of program specifications;
- Creation of system test plans;
- Creation of operating instructions;
- Creation of implementation plans;
- Definition of manual procedures.

11.5. Exercises

Notes

1. What would you say is the main driving force behind the method? (other than the Government big stick!)
2. There is an obvious similarity with JSD in that a similar diagramming technique is used (i.e entity life history vs. process structure diagram) - but are the methods in any way similar?
3. Why might SSADM be described as a safe method?
4. What is SSADM s appeal to management?
5. How much of the system life cycle does the method cover?

12. What method?

12.1. Introduction

Having looked at a number of methods (and there are many more out there than mentioned here) the question about which method to use naturally arises. Is there one software system analysis and design method that is truly better than all the others and which should therefore be used in all circumstances? Advocates of particular methods would certainly seem to want us to believe this. We tend to have an innate hope that the answer to life, the universe and everything can be reduced to a simple formalism which can then be applied to all cases. Why else would H.M. Government have decided that SSADM should be used in all Government IS projects? Too often we view system design methods as a car maintenance manual of we follow the instructions therein then we can solve all the problems that arise with our car. Having successfully used the rule book to solve a problem once, we then enthusiastically use it on subsequent projects.

Nietzsche once said that people tend to believe in the truth of all that is seen to be strongly believed in. Get enough people involved on the initial project and soon they all start believing that because method X worked for them once, it will work again; and indeed it might. But then again, it might not. However, by the time we have become a method enthusiast, we are starting to behave like Procrustes, the innkeeper at Eleusis. Procrustes inn only had a single guest room. To ensure that the bed was always a good fit for his guests (which it rarely was) he adjusted the guest to the bed. Shorter guests were stretched on a rack, and taller guests had parts lopped off to fit⁴.

The following sections⁵ address some of the issues surrounding choice of method and put forward some principles to help in our choice of method.

12.2. Method

Tom DeMarco said, in *Controlling Software Projects*:

As I get older and crankier, I find myself more and more exasperated with the great inflexible sets of rules that many companies try to pour into concrete and sanctify as methods. The idea that a single method should govern even two different projects is highly suspect: the differences between projects are much more important than the similarities.

Certain classes of moderately sized programs can be similar enough for one method to work on all of them; JSP can be used for all programs that consume and produce ordered streams of data. However, once the development becomes big enough to be considered a project, we will typically find that we need to combine or tailor methods.

⁴ Thanks to Michael Jackson for this delightful metaphor.

⁵ Adapted from Jackson, M.A., *Software Requirements & Specifications: a lexicon of practice, principles and prejudices*, Addison-Wesley, 1995

Where does this leave us as developers? If this is true, then it is not enough to become expert in a single method. There are larger issues involved here.

- Is method M good for this problem?
- How can we best use method M?
- What about method N?
- What are the good features of method P?
- Is method V better than method W?
- Why doesn't method X ever work for me?
- Can we combine methods Y and Z?

When you begin to think in this way, you stop being a practitioner of one method and become a *methodologist*. To be a methodologist, you have to study methods, be able to analyse and criticise them; you need to be able to see their strengths and weaknesses; above all, you must know **what types of problem they can solve**.

12.2.1. Constrained Environments

At this point we must have some sympathy for the intelligent people working on H.M. Government projects who only have one method at their disposal, SSADM. In many ways, their problem is much simpler: everything had better look like a nail because all they have is a hammer. In such cases we're back to imitating Procrustes' example.

12.2.2. Open Environments

When we do not have the stricture of enforced method adoption, then we are without excuse. But we have tended to pick up this habit of adopting a method and making it our own. Because we tend to view everything as a nail which can then be driven in with our hammer, we do not develop the skill of learning to recognise screws. Many advocates of software development methods say nothing about how to classify problems and how to recognise when the method they're advocating can be profitably used. The unspoken *assumption* is that their method is equally useful for every problem.

12.3. The Principle of Dispassionate Methodology

Don't get your method advice from a method enthusiast. The best advice comes from people who care more about your problem than about their solution.

Every method is associated with a *problem frame* (either implicit or explicit). A problem frame is simply a generalisation of a class of problem. The following development problems belong to different problem frames:

- A system to control automatic ticket machines at a roadside toll booth.
- A simple customer information system with transactions to create, maintain and delete information, and queries to make the information available to telephone sales staff.
- A signal processing program, in which input signals have to be processed and converted and then output.

Notes

- A small interactive editor that will be part of a personal information manager.
- A system to manage the calls to an emergency service.

A method that instructs you to define the user operations is assuming that in your problem you can identify *users* and the *operations* performed by them. This would be true of a word processing system, but not of a batch payroll system.

For any realistic problem (alas, not the sort generally addressed in method manuals and text books about methods), we will probably need to use more than one method. Most real-life problems are actually quite hard to describe and to solve. Unfortunately, many system methods concern themselves with telling us how to organise the solution once we've found it. They often do not tell us how to arrive at the solution itself. There is a great deal of concern with ensuring that the software we write can be proved to match the specification and that it is bug-free etc. However, the *really hard* problems are deciding what software to write in the first place. Most hard problems (ironically perhaps best addressed by so-called *soft* methods) are not easily reduced to simple formalisms.

Remember Nietzsche's aphorism when selecting methods. Just because it's a published method does not make it soundly based any more than just because a story is in a newspaper it is true. There are some general principles you can apply when critically evaluating a method.

12.4. The Principle of Close-Fitting Frames

The purpose of a problem frame is to give you a good grip on the problem. So the frame must define and constrain the problem very closely indeed.

In other words, a method that relies on a loose problem frame (e.g., step-wise refinement) can only be a loose method. It does not help to give you a good grip on your problem, so it does not give much help in solving it. This is why function decomposition is so weak; what system does not have a function, and what function can't be decomposed?

12.5. The Principle of Frame Exploitation

A method should exploit the stipulated characteristics of the principal parts of its problem frame.

A method might say: Describe the data. A better method would say: Describe the input data streams by giving a finite automaton to recognise each stream. The better method is referring to its problem frame. The input data streams are part of its frame and it is exploiting the stipulated characteristics of its parts. The problem frame states that each input data stream must be recognisable by a finite automaton. Not all sequential data streams are recognisable by finite automata. Not all problems have sequential data streams. But if your problem (or one aspect of it) does fit this frame then you can expect help from the method.

A good method must help us to distinguish between what is given — the context of the problem and the customer's requirements — from description of something new that has to be invented. Actually, it's quite easy to check

whether you have correctly described a given because there's always someone you can ask, or somewhere you can look. Further, what is given constrains what you must invent rather than the other way around.

12.6. The Principle of Deferred Invention

Invention should be delayed until description of what is already given has been completed.

Good methods place invention as late as possible in the development and follow each stage of invention with a thorough test of what has just been invented.

Finally, we need to look for the difficulties within a method. Does the method explicitly acknowledge that things may not go smoothly? Does it state stringent limitations on its applicability? Are there places at which it says, the operation you need to carry out here may prove impossible? If so, be of good cheer. Explicit limitations and difficulties are amongst the most valuable parts of a method.

12.7. The Principle of Beneficent Difficulty

Difficulties, explicitly characterised and diagnosed, are the medium in which a method captures its central insights into a problem.

A method without limitations, whose development steps are never impossible to complete, whose stages diagnose and recognise no difficulty, must be very bad indeed. This is because such a method is saying one of two things to you:

- Either, all problems are easily solved, and we know that's not true
- Or, some problems are difficult, but the method is not going to help you to recognise or overcome the difficult bits when you meet them.

Either way, this is not good news.

12.8. Summary

We have looked at several methods, one in some detail and others in passing. Couple this with lessons learnt earlier about project planning, management and estimation and you should by now have built up a picture in which it is quite obvious that software development is hard.

Methods were developed to help us tackle some of the procedural and descriptive problems we face during projects. However, when we place our faith in a particular method as the solution to all our problems, then we're actually adding to the problem. When we do this we always run the risk of making the guest fit the bed rather than referring the guest to another inn with differently shaped beds.

As professional practitioners of methodology, we must be aware that different problems tend to fit different problem frames. In actuality, an individual problem may fit several frames at once, none of them well.

Software Systems Planning & Design

Notes

Furthermore, these problems are inherently hard to solve. What we must do then is learn about methods and their underlying problem frames so that we can make rational choices about which methods to use on which projects. It will likely be a choice of methods X **and** Y **and** Z rather than X **or** Y **or** Z.

As methodologists we must be eclectic and not narrow minded, given to slavish devotion of a single method. We must also recognise that certain classes of methods (particularly those that follow the top-down paradigm) have inherent weaknesses that make life harder. In the case of top-down methods, the problem frame is so loose that we find ourselves arriving at solutions in spite of rather than by the assistance of the method.