



---

# Contents

<b>Preface</b>	xi
<b>Acknowledgements</b>	xiii
<b>1 Introduction to systems analysis</b>	1
1.1 What is a system?	1
1.2 Information systems	2
1.3 What is systems analysis?	2
1.4 Systems Methodologies	3
1.5 SSADM – Structured Systems Analysis and Design Method	4
1.6 The structure of SSADM	5
1.7 SSADM and the Systems Development Life Cycle	7
Summary	7
Exercises	8
<b>2 The current system</b>	9
2.1 The approach in this book	9
2.2 The case studies	9
2.2.1 Swillbuckets Country Club	9
2.2.2 The Medical Centre at the University of Life	11
2.3 Investigation of the current environment	14
2.3.1 Investigate and define requirements	14
2.3.2 Fact-finding techniques	16
2.3.3 Investigate current processing	17
2.3.4 Simple steps in data flow modelling	22
Summary	34
Exercises	40

<b>3</b>	<b>Modelling the data structure</b>	42
3.1	Entity modelling	42
3.1.1	Entities	42
3.1.2	Attributes	43
3.1.3	Keys	43
3.1.4	Relationships	44
3.1.5	Resolving many-to-many relationships	46
3.2	Simple steps in entity modelling	48
3.3	Entity modelling at Swillbuckets	49
3.4	Physical data store/entity cross-reference	59
	Summary	61
	Exercises	65
<b>4</b>	<b>The logical view</b>	66
4.1	Logicalization	66
4.2	Simple steps in logicalization	67
4.3	Logicalization at Swillbuckets	73
4.4	Problem and requirements catalogue	75
4.4.1	The Medical Centre	76
4.4.2	The problem and requirements catalogue for Swillbuckets	81
	Summary	82
	Exercises	83
<b>5</b>	<b>Business system options</b>	85
5.1	Business system options	85
5.2	Simple steps in creating business system options	85
5.3	BSOs at the Medical Centre	86
5.4	BSOs at Swillbuckets	89
	Summary	91
<b>6</b>	<b>Requirements specification</b>	92
6.1	Requirements specification	92
6.2	Required logical models	92
6.3	Elementary process descriptions	94
6.3.1	Structured English and decision trees	97
6.3.2	Decision tables	98
6.3.3	Simple steps in decision tables	99

6.4	Input/output design	101
6.4.1	Output design	102
6.4.2	Simple steps in output design	105
6.4.3	Input design	105
6.4.4	Simple steps in input design	108
6.4.5	User interface design	108
	Summary	109
	Exercises	109
<b>7</b>	<b>Normalization</b>	<b>110</b>
7.1	Normalization	110
7.1.1	What happens if data isn't normalized	110
7.1.2	(Not so) simple tasks in normalization	115
7.2	Rationalization	133
7.3	Rebuild the entity model	135
7.4	Entity/function matrix	136
7.4.1	Simple steps in creating an entity/function matrix	137
	Summary	138
	Exercises	138
<b>8</b>	<b>Technical and Physical Design</b>	<b>140</b>
8.1	Technical design and physical design	140
8.1.1	Design detailed user interface	141
8.1.2	Prototyping	142
8.1.3	Simple steps in prototyping	143
8.1.4	Interface flow diagrams	143
8.2	Database design	144
8.2.1	Indexes	145
8.3	Access and security	145
8.4	Volumetrics	146
8.5	Documentation	147
8.6	CASE tools	149
	Summary	150
	Exercise	150
	And finally...	151

<b>Appendix: Teaching case study – North Sea Ferries</b>	152
TITLE: NSF Project Information Document	152
Project Background	152
Project Team – Terms of Reference	153
NSF Project Briefing Document	153
Company Overview – Crossing Bookings	153
Transcript of Interview with Booking Office Manager	155
Transcript of Interview with Port Desk Staff	158
Examples of Documents used by North Sea Ferries	160
<b>Bibliography</b>	169
<b>Index</b>	171

# Introduction to systems analysis

## 1.1 | What is a system?

It is sometimes assumed that a system always refers to a computer system, but of course there are many other types of system. The human body, for example, is a complex system made up of many smaller systems: the respiratory system, the digestive system etc.

We could loosely define a system as anything with a purpose. A system must do something. If you put something into it, you should get something different out of it. So this book is a system for learning about systems analysis. If you input the time and effort required to read it, as indeed you are doing, you will gain unparalleled insights into systems analysis techniques, Swillbuckets Club, normalization and all manner of joys.

But nearly everything has a purpose. It's hard to think of something that has no purpose. Morning television, perhaps. Or museum attendants, maybe. Generally, though, we could take the view that everything is a system.

It certainly seems to be true that every system is made up of smaller systems, and also that every system is part of a larger system. The college or university you attend is a system, though it might not always seem like it. It is made up of departments, classes and students, all of which are systems themselves. Expanding upwards, the college or university is a part of the education system of the country, which in turn is part of the public services, and so on.

Sometimes, it's hard to know when one system stops and another one starts. The systems analyst has to make this decision early on: exactly what is the scope of the system being analysed? Otherwise, the analyst could be analysing away for ever.

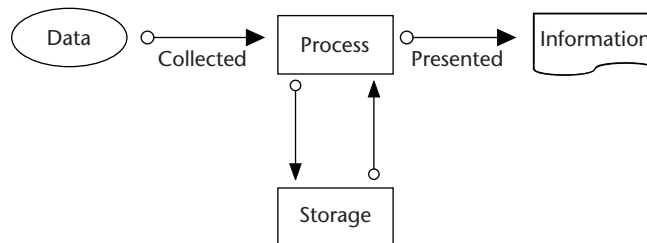
## 1.2 Information systems

Most, if not all, organizations have an information system. It might be quite primitive, like a list of names and addresses stuffed into a shoebox, or it might be hugely sophisticated. Either way, the aims will be pretty much the same: to help provide an effective customer service and to help management make the best decisions.

In order to understand how information systems can help organizations, you need to understand the difference between data and information. Data is raw facts or figures such as: 42, 12, 45, 13, 9 and 34. These numbers have no meaning until you know the context. They might be lottery numbers, map coordinates or a secret code. They need to be processed in some way to turn them into something useful. When this has been done, you have information.

Information is useful to somebody. It tells you something you didn't know before. Sometimes it's not that important and can be ignored; sometimes it's priceless. How much would advance information about the September 11th 2001 terrorist attacks have been worth? Of course, information systems in businesses won't provide that type of information, but they might make the difference between success and failure. Figure 1.1 shows how information systems work in essence.

**Figure 1.1**  
Information systems.



The raw data needs to be processed in some way to produce information. The data might be sorted into alphabetical order. Or key elements of the data might be filtered out. Or the data might just be presented in a more understandable way. It's your job as the systems analyst, in conjunction with the user, to decide what processing of data takes place and how it happens.

## 1.3 What is systems analysis?

There's a lot more to building an information system than sitting down at a PC and starting to type. In the past, before systems analysis existed, programmers went into organizations, spoke to a couple of senior managers, went away and came back a few months later with a new system. But this wasn't a very successful approach. It assumed that users knew what system they wanted. That's like an aeroplane engine manufacturer asking passengers what sort of engine they want. They just want one that works!

Equally, users of computer systems usually don't know what sort of system they want; they just want one that works. Some of them don't even want a new system. They just know that there are a few problems here and there.

So the job of the systems analyst is to find out what is good and what is bad about the system they currently have, and then design a new system that keeps the good things and gets rid of the bad things. Sounds simple, but it's not.

To begin with, there are always gaps between the user and the systems analyst. The user understands the current system, but the analyst doesn't. The analyst understands the new system, but the user doesn't. How well the analyst and user work together to bridge these gaps will determine how successful the new system is.

Effective communication between user and analyst is therefore vital. The analyst must involve the user in all stages of the systems analysis process. This will help bridge the gaps. But how will the user understand the complexities of systems analysis? After all, the user might not know much more about computers than how to type a memo or play golf.

That's one problem. Here's another. Can you believe what users tell you? Alice in Sales might have all kinds of reasons for telling you that she needs a new interactive, web-enabled, real-time, dynamic Orders system, but the real problem might be that nobody wants to buy their products.

Then again, Alice's boss might have a very different view of what an Orders system should look like (and cost!). There might be lots of 'politics' going on which causes requirements to change. The systems analyst needs a method of avoiding these pitfalls. This brings us on to methodologies.

## 1.4 Systems Methodologies

The other thing to be aware of about systems is that they are all very complicated. Usually, they are too complicated for anyone to understand without tools and techniques to help them. This is where methodologies come in. They're there to help. If they don't help, there's no point in bothering with them.

A methodology is a strategy for overcoming the problems faced by the systems analyst. It's made up of techniques, tools, conventions and documents, and it lays down the tasks to be done.

It's like cooking a meal. If you follow the recipe, you might end up with something edible. If you make it up as you go along, you usually end up with a brown mess.

One type of methodology is called *structured*. Structured methodologies are very popular with systems analysts. They are just like recipes for building computer systems. They lay down steps that the analyst should follow in a clear order. If the analyst follows these steps, then eventually a quality information systems design should be the outcome. Structured

methodologies also allow the analyst to break down complex systems into smaller, well-defined and well-documented chunks.

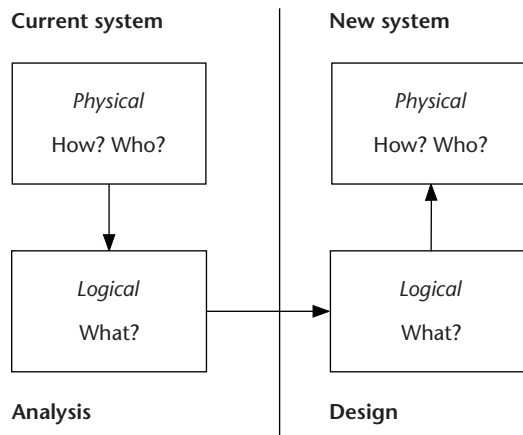
The most widely used structured methodology is SSADM.

## 1.5 SSADM – Structured Systems Analysis and Design Method

SSADM contains in it some basic principles that should help overcome the problems we've mentioned:

- 1 The first principle is the one just mentioned, about breaking down complex systems into chunks. This is called top down functional decomposition. It means that the analyst starts off just thinking about the system as a whole. Small details are ignored to begin with until the analyst has a grasp of the key features of the system. Later on, the analyst will think about the more detailed lower levels of the system.
- 2 The scope of SSADM is clearly defined. The analyst starts off by looking at the physical aspects of the current system. This means looking at *how* things are currently done and *who* does them. The analyst then moves on to look at *what* is currently done from a logical point of view. This completes the analysis phase, and then it's on to design. The analyst will consider what the new system should do and finally how it should do it. This is as far as SSADM goes. This approach might be represented as in Figure 1.2.

Figure 1.2  
SSADM approach.

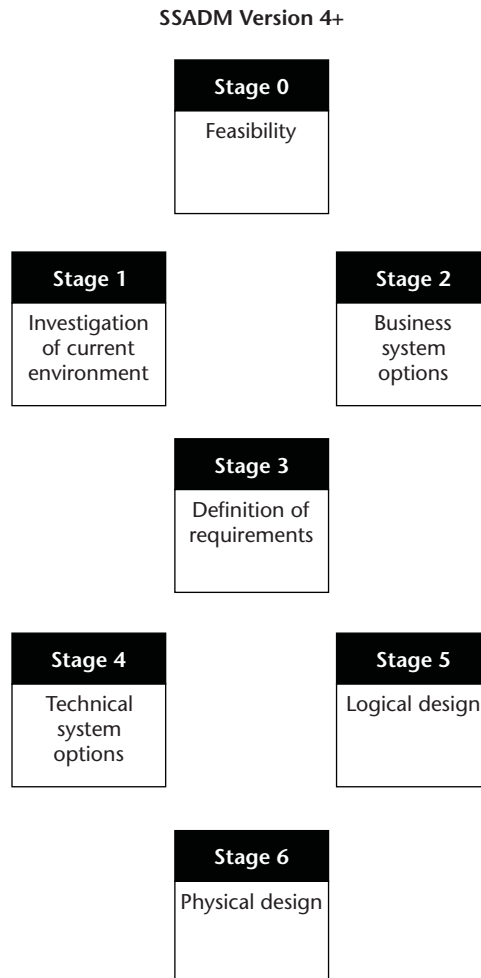


- 3 SSADM requires users to get involved from the start. This makes them more committed to the process and more likely to be happy with the new system. The analyst must meet the users regularly to sort out problems and check understanding. Incidentally, this means that the analyst should possess highly developed communication skills. These are possibly the most important skills of all in systems analysis.

- 4 SSADM makes effective use of diagrams to help both the analyst and the user understand the system. These diagrams should be simple and easy to follow, like a map of the system.
- 5 SSADM allows the analyst to see the system from different views. You can then check to see if the different views match up. This is called cross-checking.
- 6 SSADM has been around for a good many years. It's an industry standard, so most analysts have used it. If your life depended on a system being successful, you might well use SSADM as the best bet to save your skin.

## 1.6 The structure of SSADM

SSADM is made up of a number of Stages. These Stages are then divided up into Steps. Figure 1.3 shows an overview of the Stages.



**Figure 1.3**  
SSADM Stages.

### Stage 0: Feasibility

This is where the analyst and users decide if the entire project is worth pursuing. It involves the analyst considering the problems faced by the organization and producing a set of options to resolve them. The users must then decide whether the costs involved in resolving the problem are worth it. It may be that the problems are so severe that the organization simply has to resolve them. In this case, the feasibility study might be left out.

### Stage 1: Investigation of the current environment

This needs to be done so that the analyst and the users fully understand what the current system does. They need to be clear what problems they have and what they want from the new system.

### Stage 2: Business system options

This Stage allows the analyst and users to come up with some ideas about what the new system might do. Usually, a range of options, with different costs and benefits, are considered. Users will need to be clear about the objectives of the business before they can choose the option to proceed with.

### Stage 3: Definition of requirements

This involves specifying the required system. During this Stage, the analyst will want to move away from the constraints of the current system and towards a more logical, data-driven design. An overview of the underlying data structures for the required system is created.

### Stage 4: Selection of technical system options

By now, the analyst and users will have a reasonable idea of what the new system will be expected to do. This allows them to consider the technical options. For example, the key hardware components will need to be identified and costed. The users will, eventually, choose from a range of options.

### Stage 5: Logical design

This involves specifying the new system. What will the new system do? What might it look like from a user perspective?

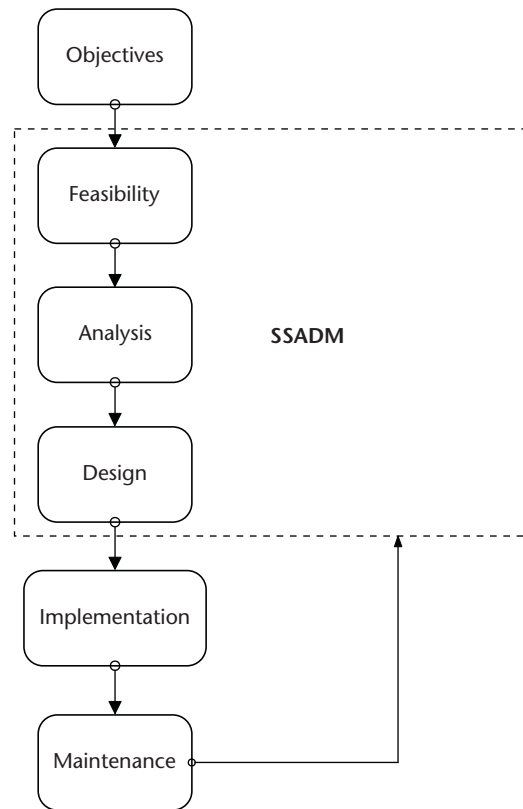
### Stage 6: Physical design

This Stage concentrates on the environment within which the new system will be running. It involves looking at storage requirements and performance issues.

At the end of each of these Stages, the analyst and users must decide whether to press on to the next Stage, abandon the project, or redo one or more Stages. All of these cost money.

## 1.7 SSADM and the Systems Development Life Cycle

SSADM isn't the end of the story – it's just part of it. The whole process of systems development goes further still. SSADM stops with the design of the new system, but systems development goes on. Most projects go through the stages outlined in Figure 1.4. This is the Systems Development Life Cycle.



**Figure 1.4**  
The Systems Development Life Cycle.

However, even this may not be the end of the process, as organizations change and it may soon be time to start the whole process again.

### Summary

We have described what systems analysis is and the need for a methodology to help the analyst. We have described the most popular methodology – SSADM – and considered its advantages and how it fits into the Systems Development Life Cycle.

## Exercises

- 1.1 What functions does the systems analyst perform during the Systems Development Life Cycle (SDLC)?
- 1.2 The SDLC is just one model for systems development. Find at least one more and describe the differences.
- 1.3 Why has SSADM become an industry standard?

# Index

- attributes 42–4, 47, 50, 53, 56–8, 67, 93, 135, 136, 145
- Business System Options 6, 85–91
- candidate keys 135
- CASE tools 149–50
- composite key 44
  - in normalization 118, 119, 120, 121, 123, 128, 131, 135
- context diagram 17, 21–4
- cross-checking 5
- database design 144–5
- database management system 144–6
- data dictionary 148
- data flow 20–1
- data flow diagrams 18–24
  - data flow modelling 17
  - decomposition 26–8
  - exercises in 40–1
  - levels 30
  - limitations 33
  - logical 66–71
  - Medical Centre examples 26, 29–33, 70–3, 88
  - Simple Steps in 22–6
  - Swillbuckets examples 34–40, 73–8
- data store 18–21, 27–9
  - different forms of 28
  - entity cross reference 58–62
  - logical 66–9
- decision table 98–100
  - exercise in 109
  - simple steps in 99
- decision tree 97–8
- dependencies 119–22
- documentation 147
- Document Flow Diagram 18, 21–4
- elementary process 27
  - descriptions 86, 94, 97
- entities 42–65
  - characteristics 42
  - database design 144
  - entity/function matrix 136–8
  - exercises in 65
  - external 19, 21, 23
  - link 55–7
  - logical 66–71
  - Medical Centre examples 63–4
  - modelling 42–9, 92–6
  - rebuilding 135
  - representation 25
  - simple steps in 48–9
  - size of 146
  - Swillbuckets examples 49–59
- fact-finding techniques 16–17
- feasibility 5–7, 14
- First Normal Form 119, 131
- foreign key 48, 58, 125, 136, 145
- functions 42, 108, 137, 148
  - entity/function matrix 137–8
- Human–Computer Interface (HCI) 106
- indexes 145
- information 2
  - gathering 16
  - management 13
- information systems 2
- input design 105–6
- interface design 141–2
- interface flow diagram 143–4
- investigation of system 5, 6, 14, 22
- keys 43–4, 47, 48, 57, 119, 123, 126
  - composite *see* composite key
  - foreign *see* foreign key
  - identification 123
  - primary *see* primary key
- logicalization 66–73
  - of DFDs 67
  - simple steps in 67–9
  - Swillbuckets example 73–5

- normalization 110–39
  - exercises in 138–9
  - First Normal Form (1NF) 119
  - rationalization *see* rationalization
  - repeating groups 118–19, 124, 126, 127
  - Second Normal Form (2NF) 120–2
  - simple steps in 115–23
  - Third Normal Form (3NF) 122, 125, 128
  - unnormalized data 110–15
- output design 102–5
- physical
  - analysis 4
  - constraints 66–7
  - data store 59
  - design 4–6, 140–50
  - DFD 25–6, 32
  - to logical 67–9
  - view 17, 26
- primary key 48–9, 58, 135, 145
- problem and requirements catalogue 73, 75–81
  - Medical Centre example 76–81
  - Swillbuckets example 81–2
- processes 2, 17, 18, 19–20, 24
  - rules 20
  - logicalization of 66–73
- prototyping 142–3, 149
- rationalization 133–5
- relationships
  - between entities 44–9, 50–5
  - resolving many to many 46, 55, 56
  - resolving one to one 54
  - type 45
- report design 103–4
- required system 6, 17, 18
  - specification 86, 92–109
- screen design 106–7
- Second Normal Form (2NF) 120–2
- security 140, 145–6
- SSADM
  - approach to 9
  - data modelling in 17
  - principles of 4, 7
  - problems with 142
  - steps in 14
  - structure of 5–6
- Structured English 97
  - exercise in 109
- systems development life cycle 7
- technical design 140–50
- Third Normal Form (3NF) 122, 125, 128
- unnormalized data 110–15
- user guide 148
- user interface design 141–2
- validation 107
- volumetrics 146–7