An alternative model for the design and implementation of records management systems

My presentation at the 2005 Records Management Society Annual Conference was concerned with identifying and analysing existing models used in the design and implementation of records management systems. These models were then compared and contrasted with models that are commonly used in the design and implementation of other information systems. The purpose of carrying out this exercise was to ascertain whether or not models used in the design and implementation of records management systems would be able to improve the quality and reduce the number of defects built into a records management system during its design and implementation. Much of the content used in my presentation was originally derived from my dissertation for the MSc in Records Management at the University of Northumbria.

So why did I suggest an alternative model for the design and implementation of records management systems? What makes me qualified to suggest an alternative? Firstly, I have an interest. I initially trained as a software engineer before completing an IT degree, so my background is to an extent in the IT world. I joined Audata 7 years ago as a technician, with little real understanding of records or information management. Over a period of time, I found myself becoming less interested in technology for technology's sake; I don't believe technology for its own sake is entirely useful. At the same time I found myself becoming more interested in the information itself, which is always useful.

Second was a coincidence. A little while into my tenure with Audata, I picked up a document on the printer, looked around the office and asked "Who does this data flow diagram belong to?" To which Jean Samuels, whom many of you will know, responded, "That's a process map I'm working on". This started me thinking about the commonality of techniques used in both software engineering and records management, which was still something of an alien world to me at that time.

Based on my experience with Jean and process maps, I decided to look at what other tools and techniques were being used in the records management world. I had this rather vague theory that many of the techniques used in the early days of data processing were probably borrowed from records managers, something, I must say, I still need to verify. That aside, I decided to look at how data processing had moved on since then, and what had happened in records management since then, in terms of the evolution of tools and techniques.

I decided to start with design and implementation models and quite quickly came across the development model in DIRKS, of course now also found in ISO 15489.

So, in essence, my talk was about the comparison of information systems development models and records management systems development models and the results of that comparison.

A Brief History of IS Models

Three key models have been used in the design and implementation of information systems, in chronological order these are the waterfall Model, the V Model and the W Model. This summary is woefully brief in terms of the number of models that exist, their acceptance or otherwise by software developers and the literature on systems development. However, these three models describe a useful evolutionary path and summarise this field.

During the early days of information systems development, usually using Cobol on mainframe computers, there was frequently an ad hoc approach to systems design and implementation, often known as ‘code and fix’, with very little documentation. This often led to systems that didn’t meet either user requirements and expectations or the technical constraints of the system. Consequently, the industry identified the need for a more structured approach. There seem to have been two main areas of study, gathering and defining

user requirements in a more logical and precise way, and defining robust, consistent techniques for the analysis and design of information systems.

The Waterfall Model

The first model I mentioned is also the first of any note, the Waterfall Model. The Waterfall Model was originally developed by Royce in 1970 and is possibly the earliest method of structured systems development. Although it has come under attack in recent years for being too rigid and unrealistic when it comes to quickly meeting customer's needs, the Waterfall Model is still widely used. It is often credited with providing the theoretical basis for other process models**,** because it provides a generic model for software development.

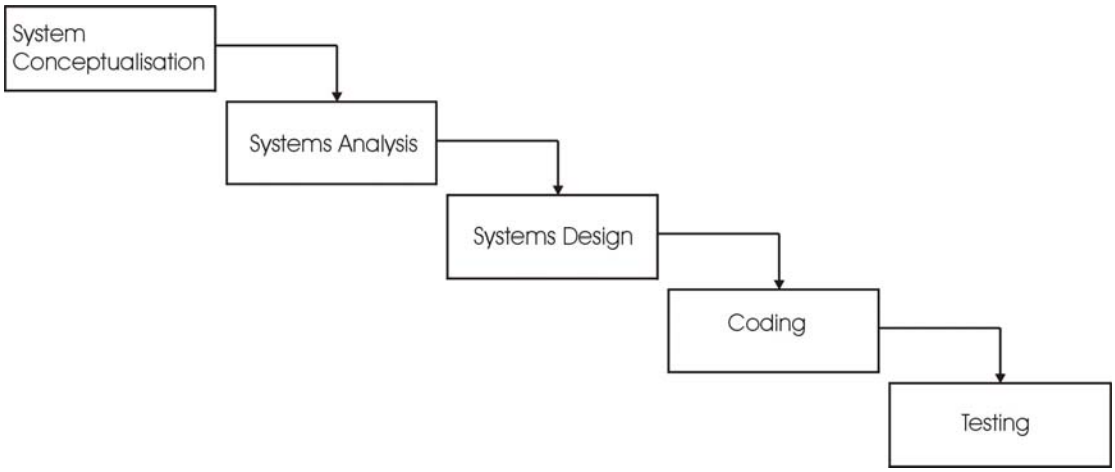

Figure 1. The Waterfall Model

The model consists of the following major steps:

- System conceptualisation,
- Systems analysis,
- System design,
- Coding,
- Testing.

The major criticism of the Waterfall Model is its assumption that the requirements will not change during the lifecycle of the project. The reality is that requirements often do change in many, if not all, projects. The failure of the Waterfall Model to recognise this could be argued to be a fundamental flaw. A mistake in the requirements phase can't be detected in the Waterfall Model until near the end, when the customer gets to see the product. This leads to a huge cost in correcting any defects. See the discussion on economics of faults below.

Whilst the demise of the Waterfall Model is predicted on an almost daily basis

"in a survey of almost 200 practitioners, accounting for several thousands of projects over the past five years, the dominant process model reported was the Waterfall, with more than a third claiming its use."

(Laplante and Neill, 2004)

Economics of Faults

The 'economics of faults, is a key concept in the development of systems – whether records management or information systems in general. Firstly, in software testing there is no such thing as a bug, there are errors, there are faults and there are failures. So a developer makes an error, which injects a fault into the code, and causes the software to fail.

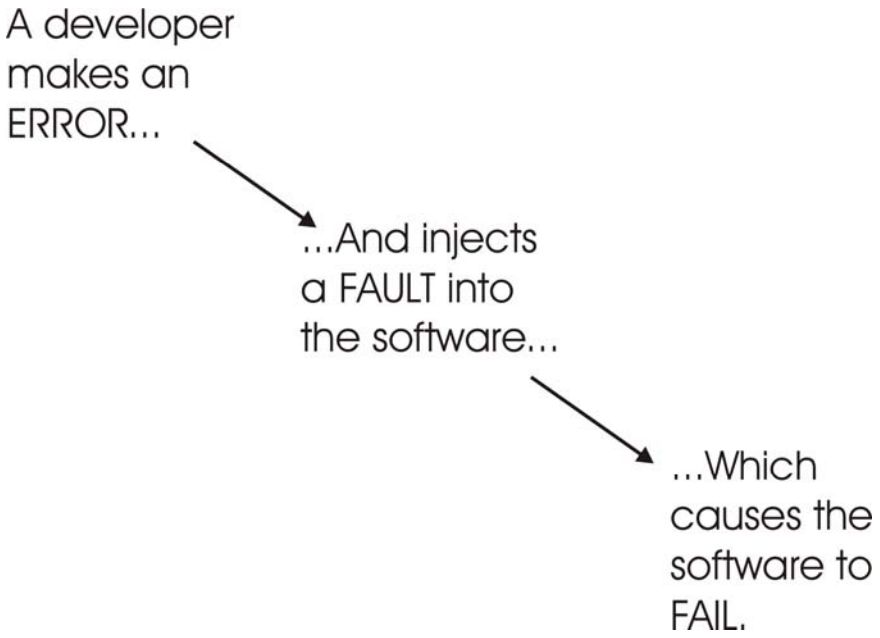

Figure 2. Errors, faults and failures

Within the information systems development and software engineering communities, it is generally accepted that faults in a systems development project become more complex, and therefore more time consuming and expensive to rectify, the later in the project it is that they are discovered. This is shown anecdotally in Figure 3.

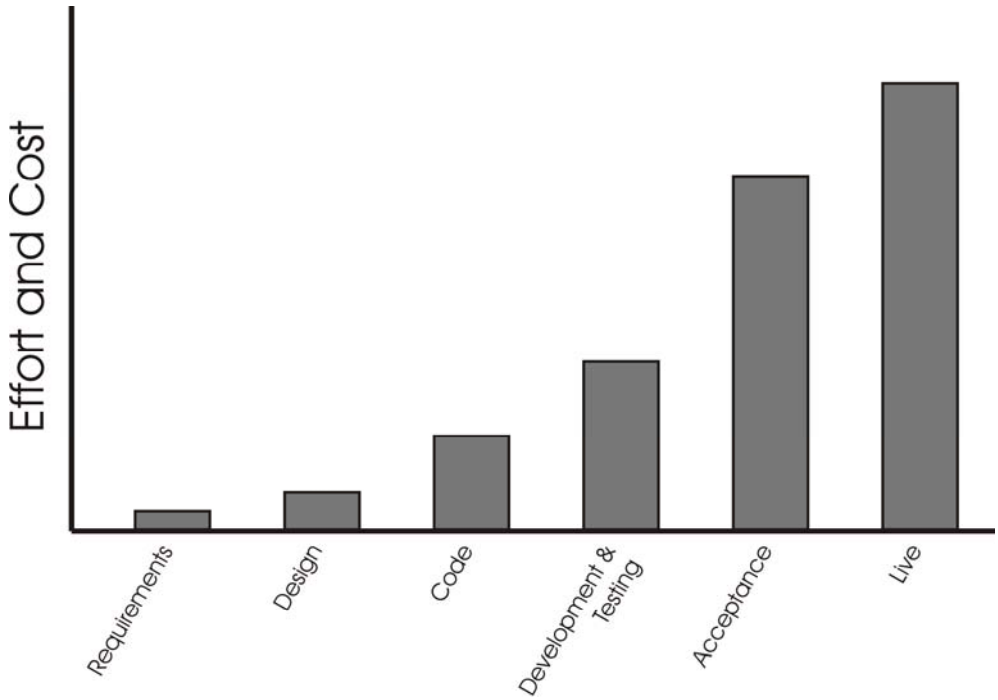

Figure 3. The Economics of Faults

So, in reality, this means that the earlier you discover faults, the easier and cheaper they are to fix. This is an important idea for everyone involved in a system – users, developers and implementers.

The V Model

As originally described by Ould and Unwin, who devised the V Model, it was known as the U Model and was laid on its side. It is an extension of the Waterfall Model. The V model associates each development activity with a test or validation at the same level of abstraction. Each development activity builds a more detailed model of the system than the one before it. Each validation tests a larger part of the system than its predecessor.

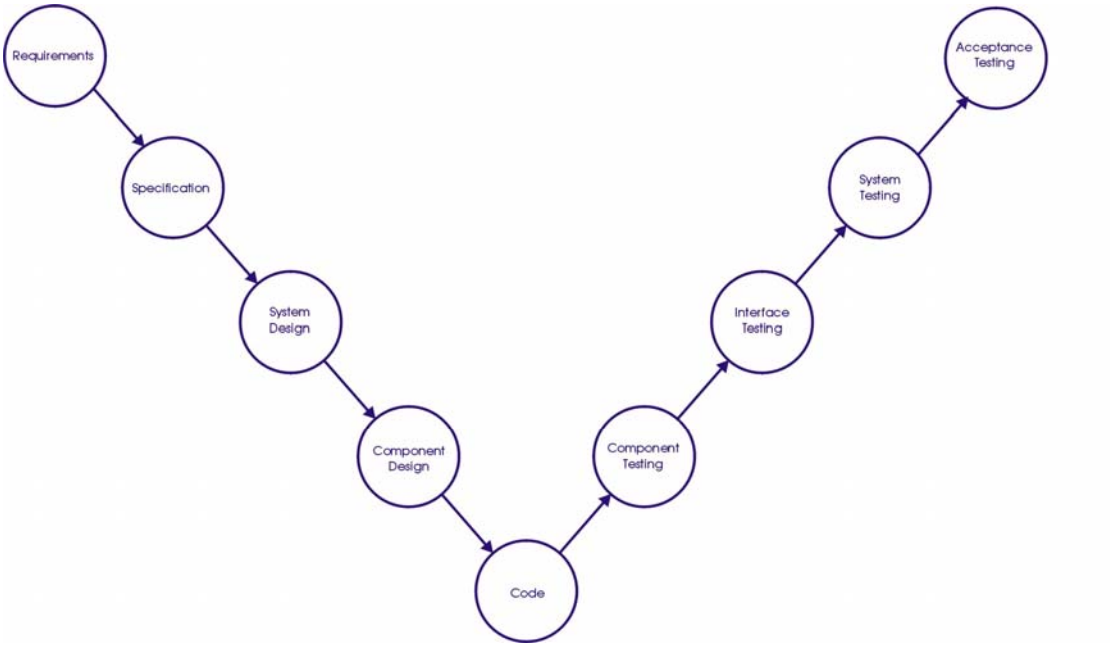

Figure 4. The V Model

The W Model

Both the Waterfall and V Model, whilst encouraging a formal process and method, which may reduce the number of faults, are not explicitly quality mechanisms. However, by extending the V Model to include an explicit quality element, the W Model is obtained as its natural heir. This quality element is expressed in the model by the mandatory use of testing throughout

the project lifecycle. Whilst originally developed for software engineering, the model can be adopted for any project as it dictates the testing of specifications, designs and any other project deliverable through a process of peer review or experiment.

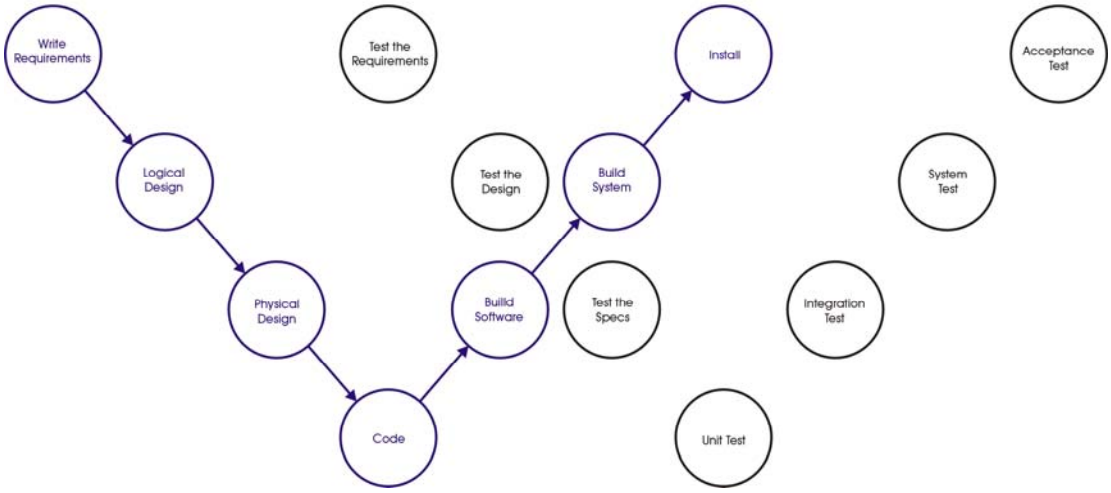

Figure 5. The W Model

Methodologies

Just to muddy the waters a little, there are also information systems development methodologies. What's the difference?

The relationship between a methodology and a model, according to Avison and Fitzgerald (1995), is that a model is the basis of the methodology's view of the world.

Or another definition:

A methodology is "a recommended collection of philosophies, phases, procedures, tools, documentation, management, and training for developers of information systems". Maddison (1983)

There are many more methodologies than models, as you would expect, and I listed a few, for example:

JSD	ISAC
SSM	YSM
SSADM	DSDM
RAD	OO
Merise	UML.

DIRKS and ISO 15489

The model from ISO 15489 is taken from DIRKS. The ISO version is easier to read than the DIRKS version. A number of colleagues and friends who work in the software engineering and information systems development industries were asked whether they thought this model covered the necessary stages for the development of a system and they all answered, without exception “No”. I They were also asked for specific comments.

The feedback can be broadly defined as positive and negative feedback. On the positive side, the following points were made:

- The model is familiar to the profession and is widely accepted.

- The model includes most of the activities that you would expect to be found in the development of a system. This shouldn’t be too surprising as the existing models make use of, to some extent, models and methodologies that are accepted in the information systems development world. The DIRKS and ISO 15 489 models are essentially based on The Waterfall Model which has the inherent weakness of not reacting to changes in the requirements. Whilst the implications of this weakness should not be underestimated, it is fair to say that at some stage the requirements must reach a point whereby they are frozen. This means everyone in the project knows what is being implemented and is a powerful tool in ensuring that the delivered system meets expectations. Freezing requirements does not necessarily mean that

they are unalterable; by incorporating a change control process into the project new requirements can be added or redundant ones removed, but within a controlled environment.

- The existing models break down the “Systems Analysis” step found in the Waterfall Model into smaller steps of “Analyse Business Activities”, “Identify Requirements”, and “Identify Strategies”. This is useful in that it provides guidance on what is actually involved in this step at a practical level.

- The model, and supporting documentation, mention the use of tools, techniques or standards but do not do not explicitly define which ones. This gives people a degree of flexibility and allows the model to be adapted to local circumstances.

The negative points were:

- The implementation of a system is a major task. Within the existing models it is shown as one step, Step G; this could be described or shown to include sub-steps to ensure, for instance, that the business processes are adequately implemented. Also, the implementation of a new system is generally carried out in a staged approach as opposed to a ‘big bang’. This could involve a pilot implementation for a small group of key users before being expanded to a department and finally organisation wide. The current models do not reflect this common approach.

- There is no build step in the models. The existing models jump from designing the system to implementing it without discussing the fact that in between design and implement, the system has to be built. This could be viewed as reasonable in that the system development, if a bespoke one, is likely to be carried out by specialist software developers. However, their omission could also be viewed as a weakness in that

these development activities must take place, and should therefore be covered by the model. Indeed these development activities are likely to represent a significant proportion of the overall project both in terms of time and cost. There are existing methods that will be appropriate for the development of these steps.

- It is not a good idea to have multiple lines in each direction on a model. It makes it easy for users to miss steps as the order of flow is not explicit.

- The model is based, as previously stated, on the Waterfall Model; however, it has been modified to a small degree to provide feedback loops that allow for the model to be regarded as cyclical. To an extent, these feedback loops give the impression that the models are edging towards the Spiral Model of software development, but do not go far enough to complete this transition from Waterfall to Spiral as the feedback loops are one offs and not part of a continuous iterative process. [The Spiral Model is an advance on the W Model in which the system iterates towards a final design.]

- The last step, the Post Implementation Review, feeds back into the first, allowing for potential modifications of the next version of the system based on lessons learnt during the implementation of the first. This is likely to negate the need for a feedback path from Step H to Step C. This is not because the information making up this feedback is without use, but because it is best fed back into the start of the lifecycle rather than the middle.

- Other of the feedback paths do not make as much sense. The feedback path from Step G to Step F is not useful. Whilst gathering feedback post implementation is useful to guide future versions, the feedback should be presented as a series of requirements, whilst altering the design of

the system during implementation, as suggested by the models, should be discouraged.

- The feedback path from Step C to Step B is not useful either. The analysis of business activities is just that, an analysis. If the business activities undertaken do not conform to the records management requirements then there might well be a case to alter the activities, but this should be included as part of the overall systems design. Therefore the gaps identified between the records management requirements and the practise should be documented as non-functional requirements.

- The existing models show a primary path from Step B to Step F. Whilst the business processes will influence the design to an extent, this should be stated in the requirements. The inclusion of this primary path is confusing at best and plain wrong at worst.

- The last point was around quality. There are too few explicit quality steps in the models. The models do suggest a review of the business processes identified as a result of Step B to validate them and also to review the design documentation produced as part of Step F. These are good things to do, but do not go far enough in themselves. In terms of the weaknesses identified, this lack of quality is potentially the greatest.

- Testing is a quality activity, but in my opinion testing is often viewed as a quality control exercise and not a quality assurance exercise. The difference is fundamental It has been estimated that some 85% of software defects can trace their roots to the requirements phase of the project (Young, 2001). This implies the need for a quality review of the requirements. This need should be expressed in the development model as an explicit activity. The more errors that are trapped and resolved at this stage of the project the better. Not only will the finished system cause fewer failures to the user, but if you think back to the

economics of faults, it is far more likely to have been delivered on time and within budget.

The W Model for Records Management

Bearing in mind the evolutionary path taken by information systems development, I believe that the best way to capture the plus points in the current models whilst addressing their weaknesses is to develop a version of the W Model for records management.

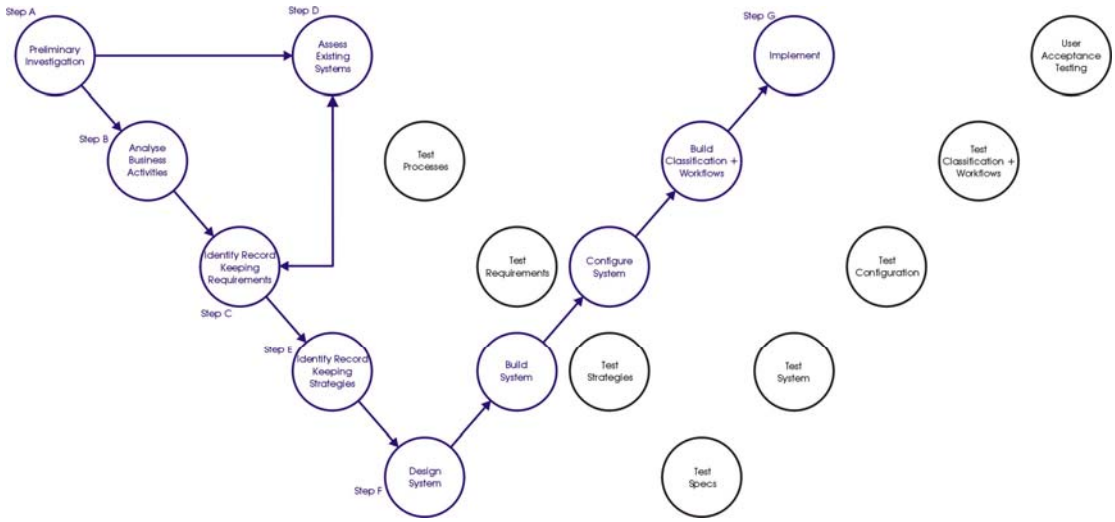

Figure 6. The W Model for Records Management

My alternative model is based on the W Model. It has taken the steps defined in DIRKS and ISO 15 489, retained the naming conventions and presented them as a W Model. To do this it has been necessary to introduce new steps. The introduction of these new steps also serves to describe the implementation phase of the project in greater detail, therefore addressing another one of the weaknesses identified in the existing models. These new steps introduced are:

- Build system,
- Configure system,
- Build and configure workflows.

Each of these additional steps correlates to a design step, which gives the first 'V' of the model, shown in blue. Each of these additional steps has a corresponding quality step; these are shown in black, which gives the second 'V' in the model. The emphasis of the prototype model is to build quality into the system from the outset as opposed to finding defects at the end which, as previously mentioned, is more time consuming and expensive to rectify.

Build System. This new step is concerned with the development of the system itself. This is a specialist activity and is best undertaken by suitably experienced software developers. If the system being developed is a commercial off the shelf product (COTS), this step will be skipped, although its quality activities won't be. Whilst it might be unreasonable to expect records managers to have an in depth understanding of software development, at least one member of the project team should. It would also be useful for the developers to have their development procedures assessed prior to the project starting.

Configure System. This is a different step to building the system, not just in the semantic differences in its name, but in what the step actually involves. During the build stage of the project, pre-defined requirements will be met. If, for instance, the system needs to be able to incorporate retention scheduling, then during the system build step this functionality will be implemented within the system, but it would be foolish to hardwire it. It would be much more sensible and useful to build the system in such a way as to allow new retention schedules to be added as required. This is an example of what is meant by configuring the system. In other words, the ability to customise or tailor the system to meet the specific needs of an organisation at a specific time.

The same process will be applied to the classification scheme, this is something that will be implemented dynamically and although it could be hardwired, it wouldn't actually leave any flexibility to react to changes.

This configuration step will largely be governed by the requirements.

Workflow. Once again it would be foolish to hardwire any workflows into the system as they are based on business processes that are liable to change.

Addressing The Gaps

The W Model for records management addresses the gaps previously identified.

Implementation. The prototype model is based on the W Model instead of the Waterfall Model. It has associated development activities with design activities and simultaneously introduced explicit quality activities. It has expanded the implementation phase and removed the primary path between Step B and Step F.

Tools and Techniques. The model has not specified the use of any tools and techniques but the quality steps do imply their use. Testing, for example, will require specialist testing software, whereas reviewing the requirements might use the Fagan Method. Similarly, the audit of the software developers might be against the Capability Maturity Model. Individuals and organisations might have their preferred methods. The prototype model only suggests that quality activities are a useful thing to do, not how to carry them out.

Practise. The major difference between the theory and practise in the existing models is the lack of testing. The Waterfall Model has a clearly defined testing step, but this is not present in the existing models. The prototype model has introduced testing at every step, thus also addressing the lack of explicit quality activities.

Testing can essentially be thought of as falling into two categories, dynamic and static. Dynamic testing is that with which most people are familiar and involves testing a system and its code (its programming). In the case of User Acceptance Testing, this involves testing that the requirements have been met, which has an evident dependency on the quality of the initial

Requirements Specification. Dynamic testing will also include testing individual modules of code, testing these modules together, and testing the system as a whole.

Static testing is the testing of documentation, specifications, designs and so on. It is the type of testing that will be used throughout much of the design and implementation of a records management system using the W Model. There are a number of techniques in use to aid with testing or reviewing documentation. Two commonly used techniques are the Fagan Method and the Delphi Method. Either would be applicable in the design and implementation of records management systems.

Feedback Loops. I suggested above that some of the feedback loops were either not necessary or not useful. To address this, the prototype model does not include two feedback paths that are present in the existing models. The paths from Step G to Step F and from Step F to Step C have been removed.

Conclusion

This paper has discussed the concept of building quality into the design and implementation of records management systems. It discussed three models used in information systems development,; looked at the models in DIRKS and ISO 15489 and went on to identify strengths and weaknesses in those models. Last, an alternative model was presented that should cause a reduction in the number of defects in the design and implementation of records management systems.

A full copy of the original dissertation can be found at www.audata.co.uk.