

**DIVISION OF COMPUTER SCIENCE**

**Notations for Modelling Multimedia Systems**

**Carol Britton  
Sara Jones  
Margaret Myers  
Mitra Sharif**

**Technical Report No. 256**

**September 1996**

# Notations for modelling multimedia systems.

Carol Britton<sup>1</sup>, Sara Jones<sup>1</sup>, Margaret Myers<sup>2</sup>, Mitra Sharif<sup>1</sup>

<sup>1</sup>Faculty of Information Sciences,  
University of Hertfordshire, Hatfield, AL10 9AB, UK.  
Tel: +44-(0)1707-284321, Fax: +44-(0)1707-284303,  
Email: C.Britton, S.Jones, M.Sharif@herts.ac.uk

<sup>2</sup>American International University in London,  
Richmond College, Queens Road, Richmond. TW10 6JP  
Tel: +44-(0)181-455-7921, Email: myersm@staff.richmond.ac.uk.

## Abstract

This report is one of the deliverables from the M3 (Modelling MultiMedia) project, which was funded by the EPSRC under the ROPA initiative. The other four reports from the project can be found in the references section under [BRI96 b - e].

The aim of this report is to provide some guidance for developers in choosing notations for the modelling of multimedia systems. The report focuses both on work that has already been carried out on modelling notations for traditional computer systems, and on the distinctive features of multimedia systems themselves. By examining multimedia in relation to the work that has already been carried out on modelling notations, we have been able to identify the characteristics of notations that make them especially appropriate for modelling multimedia systems. Developers, faced with the task of choosing a modelling notation for a multimedia system, can use the characteristics suggested here to evaluate different notations and to predict which notation will be the most effective as a modelling technique.

## 1 Introduction

Modelling plays a crucial part in the development of systems. It is important not only because it produces deliverables such as the requirements specification, documentation and prototypes, but also because it is through the process of modelling that the developer comes to understand the problem and to visualize a solution.

The effectiveness of a particular model is influenced by a number of different factors, including the nature and complexity of the problem to be modelled, the expertise and experience of the modeller, the purpose for which the model is built, the modelling notations used, and the tools used to implement the notations. The number and variety of variables that affect the model make it very difficult to compare different models in terms of overall quality and fitness for purpose. A model which is effective in eliciting and refining detailed user requirements may be woefully inadequate as a basis for design and implementation; or a model which is not particularly informative may be the result of a number of factors, such as modeller inexperience or a highly complex problem domain. Because of the difficulties inherent in evaluating models in general, the M3 project [BRI96] restricted its focus to look at notations which may be used to model multimedia systems, in particular those systems which are intended for use in education and training.

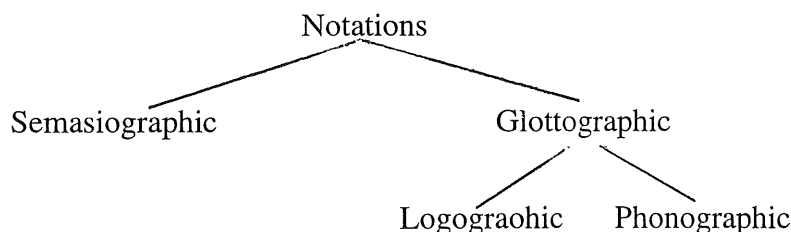
The importance of using an effective notation has long been recognized. We only need to think of performing long multiplication and division using Roman numerals to realize the enormous benefits of the Arabic numeric system. Our problem in the case of new and rapidly evolving technology, such as multimedia, is how to have confidence in our choice of notation, since we have no way of knowing from experience which notation is the most effective. The literature on modelling computer systems contains little in the way of either theoretical guidance or empirical case studies relating to the specification of requirements for interactive multimedia systems. Where advice does exist for multimedia systems, it is geared mainly to their design and production, rather than to the modelling of their requirements. The growing number of multimedia projects and the specific problems associated with multimedia development [BRI96c] mean that there is a real need for some form of guidance and for useful notations to facilitate modelling. In this report we discuss the distinctive features of multimedia systems and use these features to identify useful characteristics of modelling notations for this type of system.

Section 2 of the report defines what we mean by the term 'notation' and gives a brief overview of three classifications that have been suggested for notations in general. In Section 3 we identify notations that are used to produce models in disciplines related to multimedia system development and in Section 4 we discuss criteria which have been suggested for modelling notations for traditional computer-based systems. Section 5 outlines the distinctive features of multimedia development, which are then used as the basis for identifying the criteria for modelling notations which will be effective in modelling multimedia systems, as suggested in Section 6. Section 7 summarizes the findings of the report and their implications for modelling multimedia systems.

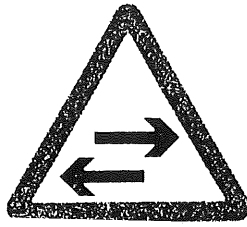
## 2 Notations in general.

Notations, also referred to as scripts or writing systems, have been defined as " a given set of written marks, together with a particular set of conventions for their use" [SAM85]. During the twentieth century, linguists both in Europe and in America have concentrated on spoken rather than written languages, with the result that there is little recent research that can be used as the basis for work on notations in their own right. However, Geoffrey Sampson's book 'Writing Systems' provides a notable exception.

Sampson describes three classifications for notations. The first of these is shown, in simplified form, in the diagram below:




The initial distinction is between semasiographic and glottographic notations. Semasiographic writing systems are means of visible communication which are independent of any particular spoken language. Semasiographic systems can be found in areas of public communication, such as road signs and cleaning instructions. Although it is not possible to read aloud a sign such as the one shown below, motorists are fully aware that they are approaching a road with traffic in both directions.



Well-developed semasiographic writing systems are often associated with primitive peoples, but this is certainly not always the case. Maths is a highly sophisticated form of semasiography. As an example, we can consider the mathematical notation 91,000, which translates as ninety-one thousand in English, and quatre-vingt-onze mille (four twenties and eleven thousand) in French. All 3 representations (maths, English and French) have different structures; there is repetition in the maths, but not in the other languages since 000 translates into one word in each. Mathematical symbolism is not tied to a specific spoken language. It is a semasiographic notation that articulates thought directly and independently rather than merely standing for its spoken articulation.

In contrast to semasiographic, glottographic notations provide visible representations of spoken-language utterances. Glottographic writing systems are further divided into logographic (based on meaningful units) and phonographic (based on units of sound). Examples of logographic notation are %, meaning 'per cent', and & meaning 'and'. It is clear that the logographic symbol & bears no relation to the phonographic a, n, d, since it is not meaningful to replace words such as land or candy with l& or c&y. One of the most common examples of a phonographic notation (based on units of sound) is the international phonetic alphabet.

The second classification described by Sampson is that of motivated and arbitrary notations. A notation may be considered to be motivated if there exists a natural relationship between the elements of the notation and objects or ideas that they represent. Many of the characters in Chinese script are motivated, for example:

  
meaning tree

and

  
meaning forest

In an arbitrary notation there is no natural relationship between the object and the representation; the symbol +, for example, bears no obvious relation to the notion of conjunction. On the whole the elements of semasiographic notations tend to be more motivated than elements of glottographic notations, as in the example of the road sign shown above. However, there are always exceptions: who would know without studying the Highway Code that the sign below, which looks like a motor cycle jumping over a car in a red circle, actually means 'No motor vehicles'?



Sampson's final classification for notations refers to 'completeness': the extent to which a notation provides representations for the whole range of units that are relevant in the language concerned. Completeness in a notation is desirable because it allows the greatest number of thought distinctions to be transferred from speech to paper. In this sense, written English is not fully complete because it fails to provide the means to represent tonal differences that are present in speech and which can radically alter the meaning of what is said.

Pictogram and ideogram, two terms that are frequently used to describe notations, are not included in Sampson's classifications. They are not considered as helpful in this context because they may refer to an element of a notation which is either semasiographic, or motivated, or both. Although semasiographic notations tend to be motivated, this is not always the case and it is useful to make the distinction.

### **3 Notations used in disciplines related to multimedia**

Our aim in this project is not to invent new notations, but to carry out a survey of current notations in the light of their effectiveness for modelling multimedia systems. We therefore need to establish what the current practice is as regards multimedia system development and what notations are commonly used in the modelling process. This subject is discussed fully in [BRI96c] which describes the survey of developers working on multimedia systems for education and training which was carried out as part of the project.

A separate approach to identifying notations for modelling multimedia systems is to look at notations that are used successfully in areas closely related to multimedia development, in particular the development of traditional computer-based systems.

In traditional software systems various forms of structured diagrammatic notation, such as data flow and entity relationship diagrams are widely used. Formal notations, based on maths and logic, are frequently found in requirements specifications for safety and security-critical systems; while highly interactive systems are usually modelled, at least in part, using prototyping. More recent modelling techniques include those which have been developed as part of an object-oriented approach to software development, and techniques which have been designed primarily to model the user interface, such as Wasserman's USE [WAS86]. Among the Requirements Engineering community, RML (Requirements Modelling Language) [GRE 86 and 94] is a technique which has been designed specifically for capturing software system requirements.

Two other disciplines which have features in common with multimedia system development are engineering and film production. Modelling techniques used in engineering practice according to Kaposi [KAP95] include technical drawings, 3-dimensional models, prototypes, verbal descriptions, maths, logic and directed graphs.

The process of film production is similar to multimedia system development in that several different types of media are involved and all have their own modelling notations. In the production of films voice content is usually represented by a script, audio effects by a time-line and video by storyboards. However, the film industry does not claim to use modelling techniques as such. In the majority of cases, a model of what is required is held in the director's head and subsequently refined during constant retakes in a way that is similar to evolutionary prototyping.

It is beyond the scope of this report to examine in detail all the notations used in disciplines related to multimedia system development, but it is nonetheless useful to be aware of notations that have been found to be effective for modelling these types of systems. In the following section we focus on traditional computer-based systems and consider work that has been carried out on establishing criteria for effective modelling notations for this type of system.

#### 4 Criteria for modelling notations for computer based systems

Work on criteria for modelling notations has been carried out by authors from both the academic and industrial communities. In general, the emphasis is on criteria either for the completed model or the software requirements specification as a whole, although some work does exist on criteria for choosing effective modelling notations. Farbey [FAR93] includes both criteria for notations, such as readability, modifiability and lack of ambiguity, and criteria relating to the notation in use, such as the production of a well-presented specification, the cost in time to produce the model, and the amount of support available. Green [GRE89] suggests that a notation should be able to support what he terms 'opportunistic planning', where high-and low-level decisions may be mingled, work may frequently be re-evaluated and modified and the developer's commitment to different decisions may be strong or weak. Although Green is writing about notations for programming, his point is equally relevant to the study of notations which are used earlier in the development process. In an article on hypermedia design [GAR95], Garzotto evaluates notations in terms of what can be described; a useful notation should be able to model information content and presentation, system structure, and interaction with the user.

Davis [DAV88 and 93] has suggested a list of criteria relating to the effectiveness of models and the choice of notations. Several of Davis' criteria are expressed in terms of the software requirements specification, such as the criterion that "proper use of the technique should result in an SRS that is understandable by customers and users who are not computer scientists". As mentioned in the first section of this report, it is difficult to assess modelling notations against criteria which are expressed in terms of the completed model or software requirements specification, since the model may be influenced by a number of factors apart from the notation used. If we want the software requirements specification to be understandable by non computer scientists, we need to look further to identify the characteristics of modelling notations that can achieve this aim. In [GRE80] Green makes the point that programming languages with a large number of features are more difficult to learn and understand than languages with fewer features. If we apply this to modelling notations, we can deduce that understandability is supported by using a notation which has a relatively small number of different symbols. This means that, to achieve ease of understanding, a relevant criterion of a modelling notation is to have relatively few symbols. As well as criteria expressed in terms of the completed model, Davis' list also includes some criteria which relate directly to modelling notations, such as that the notation should permit annotation and traceability, facilitate modification, and provide a basis for automated checking and generation of prototypes and system tests.

Among publications from authors in industry, criteria for modelling notations in the STARTS Guide [STA87] incorporate qualities such as rigour, suitability for agreement with the end-user and assistance with structuring the requirements. Rigour comprises four separate features: how precisely the syntax of the notation is defined, the extent to which it is underpinned by maths and logic, whether the meaning of individual symbols is defined and the extent to which the notation supports consistency checking of the requirements themselves. Suitability for end-user agreement refers to ease of understandability of the notation by an untrained user, and assistance with structuring the requirements assesses the extent to which the notation supports hierarchical decomposition and separation of concerns in the model. The STARTS Guide also regards the range of requirements covered by the notation as important, including functional, performance, interface, system development and process requirements.

Admiral Training's [ADM95] guide to multimedia design is of particular interest in the M3 project, since Admiral is a successful producer of educational multimedia systems. The Admiral guide is similar to the STARTS Guide in placing emphasis on

what the notation should be able to model. Effective notations, according to Admiral, should be able to describe the current situation, the target audience, the actual and required level of user performance, the overall aim of the system, the environment, possible constraints and details of specific functions.

In developing a set of criteria for modelling notations for multimedia systems, we recognize that there is no single 'best' notation. A notation which rates well against a particular criterion will almost certainly score badly against others. There is frequently a trade-off, for example, between the rigour and precision of a particular notation and the ease with which a novice user can understand it. We have already mentioned the difficulty of comparing completed models in terms of quality. We need to be wary, therefore, of selecting criteria that apply to the models built using particular notations, rather than to the notations themselves. This means that certain criteria that one might expect to find on our list, such as correctness and traceability of requirements, are in fact not part of this investigation, since these criteria relate to the way the notations are used to build models, rather than to the intrinsic properties of the notations themselves. We have assumed in the M3 project that notations are used by well-qualified practitioners, so that models produced are the best that can be produced using the notations, and we have attempted to identify criteria that relate specifically to notations rather than to the completed model.

In the following section we discuss distinctive features of multimedia development which influence the choice of notation used for modelling multimedia systems.

## **5 The distinctive features of multimedia system development**

The elicitation, specification and validation of requirements for interactive multimedia systems present the developer with a number of problems. Some of these problems are the same as those experienced by developers of traditional software systems, but others are more specific to the development of multimedia systems. In this section we briefly discuss some of the more distinctive features of multimedia projects that have a material effect on the multimedia requirements process. A more detailed account can be found in [JON96].

**5.1 Requirements for information presentation, rather than processing:** Many multimedia information systems, especially those for education and training which have been the focus of the M3 project, are better characterised as systems for information presentation, rather than information processing. The data stored in such systems does not change during the running of the system, so this means that many 'traditional' notations for modelling data processing requirements, such as data flow diagrams or entity life histories will be inappropriate. It also means that more emphasis must be placed on what have been called 'non-functional' requirements (relating, for example, to information content, media and usability), than on input-output specifications of required functionality. Requirements of this kind are not unique to the development of multimedia systems, but are likely to have more influence on the development process than in traditional systems.

**5.2 Need for integrated modelling techniques:** Multimedia systems will normally, by definition, integrate many different media, requirements for each of which might be modelled in a different way: for example, story boards have traditionally been used to define sequences of video and animation, voice clips might typically be scripted in natural language, music might follow a score, and graphics and screen designs might be mapped out in free-form diagrams, whereas the connectivity of the underlying hypertext network might be modelled as a connection map, and the interactivity might be represented in terms of state transition diagrams. There is a need for a common framework within which requirements of all these kinds can be integrated. If particular kinds of requirements are omitted from this framework, there is a danger

that the technology they relate to will be under-exploited: for example, if developers cannot see how to model the relations and connections between different screens, they may impose an overly simplistic structure on the system and thereby restrict its functionality.

This kind of problem is encountered in developing many kinds of system where requirements on different aspects or views of the system need to be integrated into a single specification, but it is particularly acute in the case of multimedia systems, owing to the large number and wide range of elements that need to be combined.

**5.3 Confusion between requirements specification and detailed design:** In multimedia system development, it may be especially difficult to distinguish between requirements and design. Clients may have requirements that particular parts of the system should be built in very specific ways: for example, they may specify that particular video clips should be used, or a particular piece of music should be played. Precise specifications of videos or music may be viewed by some as being an essential part of the requirements specification, and by others as part of design. For other parts of the system the client may have a much more abstract notion of what is required with no particular idea of how this is to be achieved. The different levels of detail in the requirements specification can be problematic to the project manager as it makes it difficult to establish identifiable project milestones.

**5.4 The Implications of Using New Technologies:** Any development employing new or emerging technologies is vulnerable to problems associated with uncertainty. Lack of experience with the relevant technology means that members of the development team are likely both to over- and under-estimate the capabilities which the system might feasibly provide. Different members of the team start the project with different expectations, depending on their previous experience, and each member's expectations evolve during the course of the project, probably at a different rate from those of others. All these factors make communication both more necessary and at the same time more difficult.

The development of experience and understanding of new technologies through the course of a project can also lead to radical re-thinking of the domain, as new ways of doing things become apparent. Methods for requirements definition must therefore cater not only for the evolution of requirements, but also for evolution in the domain itself as new possibilities are discovered by the domain experts.

**5.5 A fast-moving market:** Related to the rapid development and evolution of multimedia technology is the fact that the market for multimedia systems is extremely fast-moving. Often companies involved in multimedia development find themselves developing to market (rather than developing a bespoke system for a particular client) so that they can capture a particular market niche before any of their competitors. Developments obviously need to progress quickly in this context. We must therefore acknowledge the need for requirements practices and procedures which are flexible and efficient in delivering effective requirements with a minimum of cost, time and effort.

**5.6 Disparity of User Groups and Development Teams:** An important group of information systems are those which provide information to members of the general public in an easily accessible form. For these systems there is no easily identifiable 'user' from whom to elicit requirements. Educational information systems may be directed at a particular subset of the general population, but still have to cater for considerable differences in ethnic and social backgrounds, educational attainment, knowledge of the domain and experience or confidence with the use of computers. Methods for defining requirements for such systems need to be able to record and integrate requirements from representatives of many kinds of system users. Of course this is true for both mono- and multimedia system development.



A multimedia development project may also involve a large and multidisciplinary team of system developers. In multimedia system development, it is not possible to simply pass a software specification to a team of programmers; the development must, for example, integrate material provided by audio-visual technicians and specialists in computer graphics and animation. Writers, actors, directors, editors, composers and artists may also be involved, if less directly. Each of the team members will bring their own views and experiences to the project, which can enrich the development process, but also lead to problems of management and communication. Although all team members play a part in the development of the overall system, it is important to differentiate between different areas of expertise and to delineate the roles played by each team member. A computer scientist, for example, may have exciting ideas for music and sound effects, but should be prepared to be overruled if the audio expert is not in agreement. In the same way, all team members should recognise the computer scientist's expertise in issues relating to the overall design and structure of the system. Work on the M3 project has found that, in the case of many multimedia developments the main developer of the system does not have a computer science background. It is beyond the scope of the M3 project to assess the effect of this on the final multimedia system, but it does mean that layers of knowledge and experience in computer science which can be assumed in the developers of a 'traditional' system do not exist in multimedia development.

## **6 Implications of the distinctive features of multimedia systems for criteria for effective modelling notations.**

As discussed in section 4 of this report, there is a considerable body of work on the characteristics of a good software requirements specification, some of which relates particularly to the notations used for modelling computer-based systems. In this section of the report we examine the distinctive features of multimedia systems, discussed in section 5, to identify the effect that these have on the choice of modelling notation.

**6.1 Requirements for information presentation, rather than processing:** In section 5.1 we note that the emphasis on information presentation, rather than processing, in multimedia systems means that many notations commonly used in developing traditional software systems are inappropriate to model multimedia. However, certain established notations can make an important contribution to a model of a multimedia system. These are notations which allow the developer to produce a clear model of the structure of the intended system. Such a model provides a picture of how the information is to be organised and presented to the user. With presentation of information, the crucial relations are those of timing and sequencing of different items; this means that notations such as state diagrams [DAV93] will be useful, since they allow the developer to produce a model of different routes through the system, showing the various ways in which the information can be presented to the user. In a special edition of the Communications of the ACM, Thüning [THU95] notes that "It is not sufficient to simply impose a coherent structure; . . . it is also necessary to convey that structure to the reader. This can be accomplished most efficiently by providing a comprehensive overview of the components and their relations in terms of graphical maps or browsers."

A high-level model of the system structure provides an overview and framework for discussions between developer and client and serves as a basis for a common understanding of the system requirements. Such a model can also reflect the purpose of the system in terms of the effect this has on the overall architecture. The article referred to above [THU95] distinguishes between systems where users are allowed to wander freely, accessing information in any order they choose, and systems where users are guided through the information in a predefined order and random

exploration is curtailed. An example of the first type is a public information system and an example of the second is a teaching or training system. Although both types of system present information to the user, this is for two very different purposes. A model of a public information system should reflect the fact that users can browse the information presented in any order they choose, whereas a model of an educational system should show the prescribed routes that users must follow through the information. A modelling notation that can reflect the system's overall purpose in terms of its structure is a valuable technique for the developer's toolbox.

The second implication of the role of the multimedia system as presenter of information is the importance of modelling non-functional requirements. These can be defined as the attributes of the system as it performs its job and include properties of the system such as usability, performance and reliability. It is unlikely that any single notation will be able to express effectively all these different types of requirements. As Davis [DAV93b] says: "The ability of a requirements writer to specify a system correctly is primarily a function of the number of techniques that writer knows how and when to apply." In the absence of established notations for modelling non-functional requirements, developers generally rely on natural language and on disposable prototypes.

**6.2 Need for integrated modelling techniques:** Section 5.2 identifies the need for a common framework which allows integration of the different modelling techniques that are used for the various media that make up a multimedia system. A notation is needed which can not only model the overall architecture of the system, as discussed in section 6.1, but which can also give a high-level view of how the different media components fit into the complete system. The notation should allow the developer to indicate specifically which type of media are to be used for which parts of the system, or to show that the type of medium has not yet been decided, where that is the case.

**6.3 Confusion between requirements specification and detailed design:** In the opinion of Davis [DAV93b] "The most difficult aspect of requirements specification is avoiding the tendency to design." This problem is particularly acute in multimedia system development, since requirements for multimedia systems are often expressed with a degree of detail that may be regarded as pre-empting the overall design of the system (see section 5.3). Different levels of detail in requirements should be mirrored by different levels in the model of the system. Therefore, a notation is needed that can express a range of requirements, according to the level of abstraction or detail in which they are initially described. It is likely that the more detailed requirements will be expressed in terms of specific media, and so will be modelled in the appropriate notation. In modelling the more abstract requirements, we find, once again, the need for a notation that can express the overall structure of the required system, but which can also flag parts of the system where more detailed requirements exist.

**6.4 The Implications of Using New Technologies:** In section 5.4 we note that working with new technologies needs both fluent communication between team members and between developers and clients, and ease of modification as requirements and the problem domain itself evolve during the course of the project.

A notation which is to be used to build a model that is to be the basis for discussion and communication must be readily understandable by all parties concerned. The concept of ease of understanding will always be ultimately subjective; there are so many factors which influence any given situation that any attempt at generalization will be of little practical use. However, in trying to identify what ease of understanding means in terms of a modelling notation, it is useful to distinguish certain properties of notations that may be used to support a subjective assessment. Building on the work of Green [GRE80], Fitter and Green [FIT81] and Sampson [SAM85], we can identify four properties that contribute to ease of understanding of a notation: the number of symbols in the notation [GRE80], the

discriminability of the symbols [GRE80], whether the symbols are perceptual or symbolic [FIT81] and the degree of motivation of the symbols [SAM85]. Each of these properties is discussed below.

Among notations measured in the course of this project [BRI96b] the numbers of symbols vary between 2 (in the storyboard notation) and 76 (in the Z specification language). While it is no great revelation that a storyboard model is considerably easier for novices to grasp than a specification in Z, the number of symbols in a new notation may give some indication of how easy it will be for non computer scientists to understand.

Discriminability of symbols [GRE80] refers to the ease with which different symbols in a notation can be distinguished from each other. As an example, we can consider the Z symbols for partial function and maplet, shown below:



These are difficult to distinguish from each other and so are a potential cause of confusion for anyone trying to understand a specification written in Z.

In state transition and in data flow diagrams the boxes and arrows are clearly distinguishable from each other, although confusion may arise in some data flow notations between the symbols for process, data store and external entity. Discriminability is not currently a measurable property of a notation, but can be an indicator of how easy an unknown notation will be for novices to understand.

Perceptual representations [FIT81] are those in which we perceive meaning directly without having to reason about them, for example use of colour in electricity cables, or diagrams in various contexts. In practice both perceptual and symbolic notations frequently mix in the same model. Graphs are perceptual in that they are diagrammatic, but often contain labels (symbols) for the various nodes. on the other hand, symbolic notation is frequently laid out to show certain aspects perceptually: Z's use of schemas, formatting of program code and introduction of 'white space' to aid comprehension are examples of this.

Perceptual and symbolic notations each have advantages and disadvantages for ease of understanding. Models in diagrammatic notation often embody ideas such as connectedness (as used in state transition diagrams) and inclusiveness (as used in Venn diagrams), which most people understand. However, without elaborate 'extras' the amount of information they can contain is restricted. Models in symbolic notations are able to hold much more information, but are frequently beyond the understanding of people not specially trained in the relevant notations.

The concept of motivated and arbitrary notations [SAM85] has already been introduced in section 2 of this report. Modelling notations for traditional software systems tend to be arbitrary, although some diagrammatic notations do include motivated symbols, such as a line representing a link in entity relationship diagrams, and an arrow signifying direction in data flow diagrams. Connected (such as state transition) and inclusive (such as Venn) diagrams may each be considered as motivated in certain situations. For example, a diagrammatic-inclusive notation can represent the positions of different elements on a screen, but for specifying possible routes through a system a diagrammatic-connected notation will be more effective.

We have already said in this section that ease of understanding will always be a subjective concept. However, for developers faced with a raft of modelling notations

to choose from, many of which have been largely untried on 'real' problems, it is helpful to have some idea of whether a particular notation is likely to produce a model which non computer scientists can readily understand. Each of the properties identified above is, on its own, a fairly crude indicator of the overall understandability of a notation, but taken together they do give some idea of how quickly and well a novice reader will be able to understand a model developed in a particular notation.

The second implication of working with new technologies is that the notation used should allow the model to be modified as requirements evolve. Very little has been written about the properties of notations that make models produced using them easy or difficult to modify. Green in [GRE89] describes a notation that is hard to modify as 'viscous' and one that is easy to modify as 'fluid'. A notation is viscous if it is resistant to change, in that an apparently small modification has a knock-on effect that entails further changes throughout the model. An example of viscosity is a text that is labelled with section and sub-section numbers or letters. If a section is deleted, or a new one added, the whole numbering schema will have to be changed. Among the structured techniques, the hierarchical structure of data flow diagrams makes them viscous; the addition or deletion of a process may necessitate changes at several different levels. A fluid notation is one which encourages a clear overall structure and where individual components are clearly separated from each other. It may not be possible to avoid the knock-on effect of a desired change, but developers should be able to predict that this will happen and anticipate the extent of the modifications involved.

The most widely used and most effective technique for coping with changes in requirements is prototyping. In a survey carried out during the M3 project [BRI96c] all the multimedia developers questioned used prototyping as one, or the only, modelling technique. However, almost all developers also felt the need for modelling notations to express ideas and requirements prior to the first prototype; it is these notations that are the concern of the M3 project.

**6.5 A fast-moving market:** Developing systems in a fast-moving market means getting in before the opposition. If developers are to be persuaded to spend any time on modelling before embarking on the initial prototype, notations must be available that are quick to learn, easy to use and cost effective. Notations which have automated support may produce models more quickly than their pencil and paper equivalents, but extra time is needed to learn how to use and exploit the tools, and buying and maintaining them is an added development cost. The M3 project survey found that, where speed of development is crucial, developers tend to skip modelling altogether, or to use storyboards to draft initial ideas.

**6.6 Disparity of User Groups and Development Teams:** The wide range of potential users of a multimedia information system and the varied composition of the development team both mean that clear communication is essential in modelling requirements for the system. Ease of understanding of a notation by non computer scientists has already been identified in section 6.4 as a key element in fluent communication. We have also tried to dig deeper into some of the characteristics that make one notation easier to understand than another.

In section 5.6 we also identified the problem that arises from disparity of team members in differentiating between the various areas of expertise. In order to delineate clearly the roles played by each team member in the development of a multimedia system, a model is needed that distinguishes the different components of the system in terms of the media involved. The notation used should be able to produce a model which not only shows the overall architecture of the system (including the ways in which it will interact with users), but which also indicates the parts of the system where the various media play a prominent part. It should be

obvious to any member of the development team from looking at the model where his or her particular areas of responsibility lie and how these contribute to the system as a whole. The need to identify how the various media fit into the overall system was also identified in sections 5.2 and 6.2, which discuss the problem of the number of different modelling techniques used in a multimedia system.

## 7 Summary of findings and final comments.

The table below summarizes the discussion in section 6 on the implications of multimedia for the choice of modelling notation. We can see from the table that the principal criteria for an effective modelling notation for a multimedia system are that it is able to illustrate overall structure and that it is readily understandable by non computer scientists.

<b>Distinctive features of multimedia system development</b>	<b>Relevant characteristics of modelling notations</b>
Requirements for information presentation, rather than processing.	Able to illustrate overall structure, with different routes through the system and interactions with user.
Need for integrated modelling techniques.	Able to illustrate overall structure, with indications of where various media components fit in.
Confusion between requirements specification and detailed design.	Able to illustrate overall structure, reflecting the different levels of detail in given requirements.
Implications of using new technologies.	Readily understandable by non computer scientists. Simple structure that is easy to modify.
A fast-moving market	Fast and cheap to use.
Disparity of User Groups and Development Teams	Readily understandable by non computer scientists. Able to illustrate overall structure, with indications of where various media components fit in.

In terms of structure, three separate demands are placed on a modelling notation by multimedia system development. First, the notation should be able to illustrate the overall structure of the system, how information will be presented and how user interaction will take place. As mentioned already in section 6.1, this criterion suggests that a notation such as state diagrams [DAV93] would be the most suitable choice, since this allows the developer to produce a model of different routes through the system, showing the various ways in which the information can be presented to the user. Secondly, the notation should be able to indicate where different media are used and how they contribute to the system as a whole. This is not a characteristic of any established modelling notations and the addition of markers for different media will add an extra layer of complexity to models. However, the varied backgrounds of multimedia team members and the associated diversity in modelling techniques means that there is a potential source of confusion if the balance of different media in the system is not carefully monitored.

The second criterion that was identified in section 6 of this report as particularly important for multimedia development is that a notation should be easy to understand by non computer scientists. This is supported by the fact that the most widely used modelling techniques in multimedia development are storyboards, prototyping and natural language [BRI96c]. While prototyping is, in itself, undoubtedly a highly

successful modelling technique, it is even more effective when preceded by some form of modelling to record and guide initial thoughts on the system. The M3 survey of multimedia developers [BRI96c] found that, where this type of modelling does take place, the main notation used is storyboarding. Storyboards are the most basic of models, they have only two symbols (the box and the line) and they are intuitively easy to grasp. They are, however, restricted in what they can show and, as extra features are added to express more requirements, they tend to lose their initial simplicity.

Natural language has frequently been derided as a notation by computer scientists, who claim that it is imprecise, ambiguous, inconsistent and verbose. The 'picture is worth a thousand words' dictum and innumerable examples of poorly written requirements specifications have meant that natural language has been largely ignored in the increasingly frenetic search for new modelling notations. Yet no software development can take place without a basic reliance on spoken language and extensive use of its written form. This is particularly the case with multimedia system development where natural language is often the only notation that clients and the various team members have in common. The nature and novelty of multimedia development give us a compelling reason and an excellent opportunity to reassess the role of natural language in system development, particularly in the early stages, and to acknowledge that past problems have frequently arisen, not because of the language itself, but because most system developers and requirements engineers are not trained to use it properly.

It would be absurd to suggest that natural language should be the only, or even the principal notation used to model multimedia systems. The refinement of existing notations that have proved useful and the search for new and better ways of modelling systems must continue, but alongside these activities it is time for us to realize that one of our most valuable resources in modelling systems is natural language. If we are looking for ways in which to improve our requirements specifications, there is no better place to start than with the quality of our written English.

The aim of this report has been to provide some guidance for developers on choosing notations for the modelling of multimedia systems. The speed of change in multimedia development and the overwhelming number of new notations for modelling and design mean that it is neither sensible nor feasible to conduct empirical experiments on the effectiveness and appropriateness of different notations. Instead, the M3 project has focused both on work carried out on modelling notations for traditional computer systems and on the distinctive features of multimedia systems. By examining multimedia in relation to the work that has already been carried out on modelling notations, we have been able to identify the characteristics of notations that make them especially appropriate for modelling multimedia systems. Findings from our work indicate that a notation for modelling multimedia should:

- be able to illustrate the overall structure of the system with different routes through the information and interactions with the user;
- be able to illustrate the overall structure with indications of where the various media components fit in;
- be able to illustrate the overall structure, reflecting the different levels of details in the given requirements;
- be readily understandable by non computer scientists.

Future work in this area will concentrate on verifying our findings through contacts with multimedia developers in industry [BRI96c], and through using a range of notations to model case studies [BRI96e].

## References

- [ADM95] Admiral Training Ltd. (1995) *Multimedia Design - A Newcomer's Guide* The Department of Employment, Sheffield.
- [BRI96] C.E.Britton (1996) *The Evaluation of tools and techniques suitable for modelling interactive multimedia systems used in teaching and training* EPSRC grant reference GR/K62439
- [BRI96b] C.Britton, S.Jones, M.Myers and M.Sharif (1996) *An investigation into the measurement of modelling notations* Technical Report No. 257, Faculty of Information Sciences, University of Hertfordshire.
- [BRI96c] C.Britton, S.Jones, M.Myers and M.Sharif (1996) *A survey of multimedia developers* Technical Report No. 258, Faculty of Information Sciences, University of Hertfordshire.
- [BRI96d] C.Britton, S.Jones, M.Myers and M.Sharif (1996) *A survey of tools for modelling multimedia* Technical Report No. 259, Faculty of Information Sciences, University of Hertfordshire.
- [BRI96e] C.Britton, S.Jones, M.Myers and M.Sharif (1996) *A new notation for modelling multimedia with three case studies* Technical Report No. 260, Faculty of Information Sciences, University of Hertfordshire.
- [DAV88] A. Davis (1988) *A Comparison of techniques for the specification of external system behaviour* Communications of the ACM, Volume 31, Number 9, September 1988
- [DAV93] A.M.Davis (1993) *Software Requirements. Objects, Functions and States* Prentice Hall International.
- [FAR93] B. Farbey (1993) *Software quality metrics: considerations about requirements and requirement specifications* . in *Software Engineering: A European Perspective*, R. Thayer and A. McGetterick (Eds.), IEEE C.S.Press
- [FIT81] M.J. Fitter and T.R.G. Green (1981) *When do diagrams make good computer languages ?* from J. Alty and M. Coombs (Eds.) *Computing Skills and the User Interface* Academic Press.
- [GAR95] F. Garzotto, L. Mainetti, P. Paolini (1995) *Hypermedia Design, Analysis and Evaluation Issues*. Communications of the ACM Vol 38, No.8, August 1995
- [GRE80] T.R.G. Green *Programming as a Cognitive Activity* in H.T. Smith and T.R.G. Green (Eds.) (1980) *Human Interaction with Computers*. Academic Press
- [GRE89] T.R.G. Green (1989) *Cognitive Dimensions of Notations* in *People and Computers (HCI 89)* Sutcliffe and McCauley (Eds.) C.U.P.
- [GRE86] S.J.Greenspan, A.Borgida, J.Mylopoulos *A Requirements Modelling Language and its Logic*. Information Systems Vol. 11, No. 1, pp.9-23, 1986
- [GRE94] S. Greenspan, J.Mylopoulos and A. Borgida *On Formal requirements Modelling Languages: RML revisited*. Invited plenary talk at ICSE (16th International Conference of Software Engineering) Sorrento 1994.

- [JON96] S. Jones and C. Britton (1996) *Early Elicitation and Definition of Requirements for an Interactive Multimedia Information System* Proceedings of the Second International Conference on Requirements Engineering, IEEE Computer Society Press.
- [KAP95] Kaposi Associates (1995) *Quality Assurance and Quality Standards* Notes for MSc / Diploma in Quality Improvement and System Reliability Module 2.13, City University
- [SAM85] G. Sampson (1985) *Writing Systems*. Hutchinson.
- [[STA87] Department for Trade and Industry and National Computing Centre (1987) *The STARTS Guide*, second edition, volume 1, NCC Publications
- [THU95] M.Thüring, J. Hannemann and J.M.Haake *Hypermedia and Cognition; Designing for Comprehension* Communications of the ACM Vol 38, No.8, August 1995
- [WAS86] A.Wasserman et al. *Developing Interactive Information Systems with the User Software Engineering Methodology* IEEE Transactions on Software Engineering 12, 2 (February 1986) pp 326-45