Modelling a Canal System using Definitive Principles

Meurig Beynon
Mike Joy

Department of Computer Science
University of Warwick
Coventry
CV4 7AL

email: {wmb,msj}@dcs.warwick.ac.uk

Abstract

This paper investigates the application of new programming principles to the development of a computer-based Geographical Information System (GIS). Our aim is to develop and demonstrate a method of modelling a geographic region that involves creating environments in which a user can emulate different modes of real-world observation and interaction.

Introduction

The development of effective GISs is a major challenge in knowledge representation. There is little doubt that new approaches to data representation and processing will be required. Relevant issues have been addressed in part by a model-based approach to programming ("definitive[1] programming") that represents system state by sets of definitions[1]. This paper assesses the prospects for applying these techniques to GIS design.

The computer model to be constructed using our approach is not as restricted as a conventional computer program – the user can interact with it in many different ways, including many that are not preconceived in the model-building process. Such a model can be adapted to serve specific functions that would normally require several different special-purpose programs. It can also deal with empirical information that is encountered whilst the model is in use, rather than known in advance. In these respects, it is quite different in character from a conventional map, in which all the information is preprocessed and statically presented, and it is up to the user to interpret the data to suit a particular purpose.

The paper has two main sections: the first reviews fundamental problems in GIS design and development in relation to definitive programming. The second explains and illustrates how we propose to apply our methods in developing a GIS for the canal system in the West Midlands.

[1] throughout this paper, the term definition is used in a technical sense, and definitive to mean definition-based.

1 Programming Principles for GIS Development

1.1 Fundamental Issues for Geographical Information Systems

The design and development of GISs raises some fundamental issues:

- *we don't know what problem a GIS may be used to solve*

  Geographical data can be processed in many ways. A map-user may be planning a scenic route, identifying the location of an aerial photograph, or investigating geological history. The data required for these applications is diverse and cannot easily be preconceived.

- *complex graphical images are difficult to modify*

  A GIS differs from a conventional database in that the external presentation is generally graphical and iconic. The size of geographical data bases demands enormous computation when change is required. Referencing components of an image that is to be modified is typically difficult. It is hard to maintain the integrity of images in change.

- *graphical presentation presumes conventions for interpretation*

  Interpreting a conventional map is a complex process. The map reader has to associate graphical symbols with geographical features in a manner that must first to be learnt, and typically has to be reinforced by experience. Consider e.g. recognising symbols that are drawn out of scale, distinguishing rivers from roads, or interpreting contours. The difficulties of interpretation put further constraints on modifiability.

- *a GIS models analogue data: exact geometric locations are significant*

  It is hard to capture the concept of "X marks the spot" in a discrete computer-based model. Every geographical location is a point of reference with which values and attributes may be associated.

- *data changes randomly, as roads are built or geological events occur*

  Natural processes of change over time are outside the scope of most data-processing applications. Though we might need to adapt a library database to take account of books that disappear or disintegrate, such activities occur in a controlled environment. In contrast, geography is shaped by continuous and unpredictable processes.

1.2 Basic Principles of Definitive Programming

The demands that GIS development make upon programming methods are hard to meet with current tools. Definitive programming is a new programming paradigm that has been developed at Warwick over the last few years. Its potential has been demonstrated in several contexts where similar challenges arise, in particular, in computer-aided design [1, 3] and in modelling and simulation of concurrent systems [4].

Definitive programming entails transforming an appropriate description of a system of interacting agents into a prescription for agent operation. Of particular relevance in the context of GIS development is the fact that the system description can – to a large extent – be developed without concern for the purpose of the programming task. There is a useful and significant analogy with engineering practice, where experimental knowledge of the application domain informs the solution of a design task.

Developing definitive programs involves three complementary activities:

- agent-oriented analysis: analysing the interaction between the state-changing agents of a system so as to take account of how each agent *observes* changes of state in other agents and is conditionally *privileged* to change the state of other agents.
- representing experimental knowledge: this involves constructing sets of definitions ("definitive scripts") in which the variables correpond to experimental observations and the definitions record our expectations about how changes to observations are correlated when state-changing actions are performed. By implication, experiments are concerned with correlation between observations that admits no interference from external agents i.e. with atomic or indivisible change.
- prescribing agent actions: this entails specifying agent actions as permissible sequences of redefinitions of variables, or invocations of other agents, performed in the context of a definitive script.

Of these three activities, only the latter is centrally concerned with the programming goal to be attained. The programming task is greatly simplified by the constraints on agent actions imposed by the other activities. The prescription of an agent action must take account of what the agent observes of the system state, and its privileges to change the state. The effects of an action must be consistent with experimental knowledge.

The next section explains the relevance of definitive programming principles to the basic GIS issues introduced above, and our reasons for optimism about their successful application in the longer term.

1.3 Definitive Programming and GIS issues

- *we don't know what problem a GIS may be used to solve*

  Many approaches to computer system development focus on *the function the system has to serve*. This favours the design of an encapsulated system, where the user-interface is preconceived for a particular mode of use. The user cannot adapt such systems to serve new functions.

  Definitive programming uses a development method based on *modelling the application-domain* – an approach that creates software systems more readily adapted to new roles. Unlike other model-oriented methods, such as JSD [14] or object-oriented design [12], it focuses on identifying and faithfully representing the agents and observations in the application-domain. This focus leads to a much greater degree of flexibility; a particular choice of mode and nature of observation is consistent with models serving many different functions.

- *complex graphical images are difficult to modify*

  Definitive programming principles were first applied to interactive graphics by Wyvill [15]. They are well-suited to CAD [3], scientific visualisation [5] and animation [6]. They rely upon constructing an internal representation of a graphical image in which all the significant features can be independently referenced whether or not they are easy to distinguish visually. This representation records dependencies between the components of a graphical image in a way that allows an image to be selectively and consistently updated.

- *graphical presentation presumes conventions for interpretation*

  Our internal representation of a graphical display attaches variables to the picture elements that correspond to observations. The dependencies between the values of these variables represent relationships between real-world observations in much the same way that the defining formulae of cells of a spreadsheet reflect an external interpretation. As explained in [5], this enables the user – through experiment – to relate an abstractly defined geometric symbol to its intended interpretation.

- *a GIS models analogue data: exact geometric locations are significant*

  Any representation of a geographical region is incomplete and approximate. This limitation cannot be overcome in a static document, such as a map. It can be addressed in principle in an interactive environment, provided that the user can adapt the representation as new information is obtained.

  Definitive principles are well-adapted to deal with incomplete information. They operate in such a way that there is no distinction between design and use of the information model except in respect of interpretation [6]. For instance, a user can introduce new attributes or features interactively.

  The problem of exact modelling of analogue data has no solution within the context of the formal representation system itself. There is nevertheless a role for refining a geometric representation where useful information can be derived by interpolation from existing data. The application of definitive principles to modelling analogue data in this manner has been demonstrated in [2].

- *data changes randomly, as roads are built or geological events occur*

  In the conventional process of producing a map, there has been a clear separation between surveying and map specification. In principle, technological advances can reduce or even eliminate the need for such separation. In the future, by linking sophisticated surveying devices, such as geosatellites, to a computer-based GIS we can hope to record many geographical changes as they occur. For this purpose, programming principles that establish an intimate connection between external observations of the world and a computer model are a prerequisite.

2 Applying Definitive Programming Principles to a GIS: a Case Study

This section describes an initial study, involving some problem analysis and prototyping, aimed at assessing issues of feasibility and identifying the technical challenges to be solved.

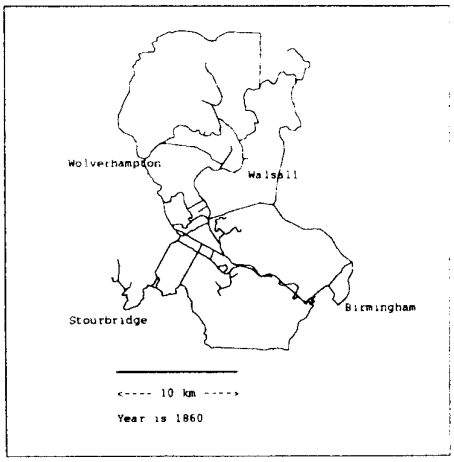

Figure 1.1

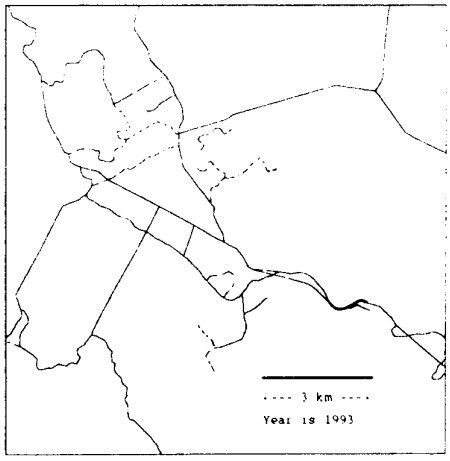

Figure 1.2

Our chosen case-study is the development of a GIS for the BCN ("Birmingham Canal Navigations") network [8, 13, 11]. This canal system was developed after the Industrial Revolution of the 18th century to provide a transport infrastructure for what was at that time one of the principal manufacturing regions in the UK. In its heyday, the network served an area of about 500 square kilometres and consisted of nearly 300 kilometres of

navigable waterway, of which 200 are still in use today. The BCN was built to handle narrowboats (dimensions no longer than 2.1m wide by 18m long), and could not compete with the railways for commercial traffic in the 19th century. Today, the canals are mainly used by pleasure craft, and the towpaths as useful green arteries for walkers and cyclists.

Our GIS is to be designed to represent both contemporary and historical data. Figure 1.1 depicts the BCN network as it was in 1860. Figure 1.2 shows the contemporary state of the canal system in the Sandwell district; closed sections of canal are indicated by dashed lines. As these figures illustrate, the canal system is sufficiently complicated for the representation problem to be non-trivial.

2.1 Analysing the Application

The first step towards designing our system is to identify the agents with an interest in the canal system, and the observations of the canal system they make. We distinguish three types of agent, each with a different motive for knowing about the canal system:

- the canal *enthusiast*, who studies the canal system abstractly,
- the canal *user*, who travels about the system,
- the canal *authority* official, who polices the system.

The roles of these agents invoke many kinds of observation of the system. Recording these entails modelling a large and diverse volume of data:

- relevant map data, to include significant present-day and historical canal features, such as locks, tunnels, boatyards and junctions, as well as nearby landmarks, such as historic buildings, public houses, and centres of population,
- information about the current status of the canal system, as is needed for route planning: e.g. restrictions due to maintenance or temporary closure of sections, approximate journey times, toll and licence requirements,
- registration information concerning each canal boat, to include name, authorised owner, place of permanent mooring, dimensions, licences issued and current location.

In applying definitive programming principles, we are led to consider how these observations are organised into states, and how these states are affected by actions on the part of agents in the system. In effect, we view the canal system and the agents that affect and interact with it as a concurrent system. In comparison with other contexts in which we have addressed concurrent systems modelling, the canal system is intrinsically relatively static. On different time-scales, canal boats move about the system, licences are issued, sections of canal are closed for maintenance, canals come and go. On the other hand, the nature of the real-world interaction and interfaces that the agents establish in performing their roles is exceptionally volatile. The dynamic aspects of the model relate primarily to the variety and subtlety of the views of the agents, who are forever re-organising the data for different patterns of use.

The fundamental principle we wish to exploit in developing a GIS is that of creating a computer model of the canal system with which we can interact in ways that faithfully reflect real-world observation and action. In using the GIS, the human agents in the canal system need to be able to reconstruct within the computer model faithful images of the states and privileges they observe in the real system, whilst the GIS manager

has to be able to emulate the effect of other transforming agents. To this end, we use definitive scripts to model real-world states. In these scripts, the values of variables represent current observations, and redefinitions of variables represent possible actions. Faithfulness to observation in such models derives from the fact that a change in the value of a variable automatically propagates changes to the values of all dependent variables (as in a spreadsheet).

2.2 Definitive Scripts and the Modelling Process

Many kinds of state-changing activity have to be modelled in our GIS. In this section, we consider some of the ways in which definitive scripts can in principle be applied. At the present stage of development of our system, these can be illustrated by modifying or augmenting the definitions currently in the model interactively, but this demands interaction with the model at a rather low-level of abstraction. The prospects for more appropriate methods of representing, developing and managing scripts are discussed in section 2.3 below.

Example 1: Informing the Canal Enthusiast

The history of the canal system is well-documented [7]. The status of the canal system at any point in its 200 year history can be inferred from the dates when individual sections were in operation. This information is represented in our GIS so that the display is automatically updated on "redefining the current year" (cf Figures 1.2 and 2.2).

Example 2: Assisting the Canal User

There are many ways in which a definitive script can be used to generate maps customised for a particular canal user or journey. Figure 2.1 depicts a map that might be generated for a journey from Horseleyfields to Old Turn Junction:

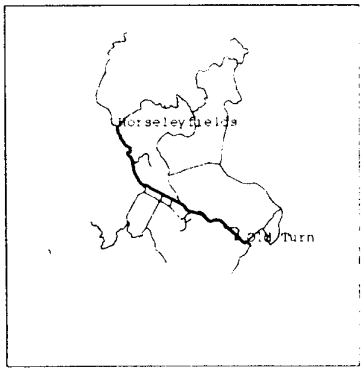

Figure 2.1

The definitive script used to specify the map is given below. In the script, `from` and `to` are the points representing the endpoints of the journey. The

map is centred on the midpoint, as given by the variable `centre`. The map is square, with side twice the distance between the endpoints. The shortest path between the endpoints is represented as a sequence of canal segments; the width of these segments is redefined so as to highlight the optimal route.

```
> from is location("Horseleyfields");
> to is location("Old Turn");
> centre is midpoint(from, to);
> side is 2*distance(from, to);
> width of shortest_path(from, to) is thick;
> year is 1992;
```

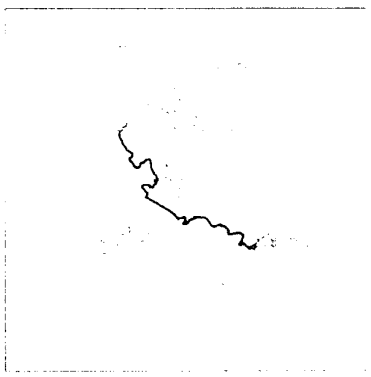

Figure 2.2

Had we been making the journey in 1800, we would have followed a different route, as depicted in Figure 2.2. To derive Figure 2.2 from Figure 2.1 it is only necessary to make a single redefinition:

```
> year is 1800;
```

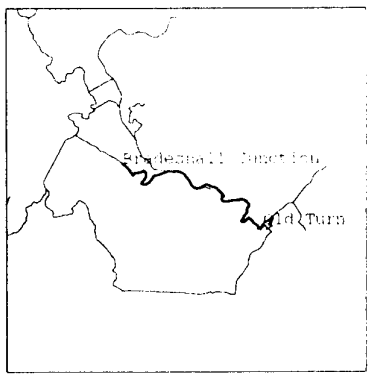

Figure 2.3

On reaching Bradeshall Junction we could generate a higher-resolution map of the remainder of the route, as in Figure 2.3. Again, Figure 2.3 can be derived by a single redefinition:

```
> from is location("Bradeshall Junction");
```

Many sections of canal are subject to periodic restrictions, and all are closed for maintenance at various times. Definitive scripts provide a mechanism by which we can take account of current information on navigability in journey planning. Closures scheduled in advance can be recorded and registered on the display by the same method that is used to distinguish active from defunct canal branches in Figure 1.2. An unexpected temporary closure can be recorded when it occurs.

Example 3: An interface for a Canal Authority agent

The duty of a lengthsman is to check that a boat B in a given region R is licensed under the appropriate authority. A suitable definition can express the condition under which B is in breach of regulations:

$$legal(B) = in\_region(R, B) \text{ and not } licensed(B, authority(R))$$

A complementary definition can relate the boat location to the region:

$$in\_region(R, B) = is\_in\_sect(sect\_no, loc(B)) \text{ and } is\_in(R, sect\_no)$$

To display this information, the position of an icon to represent the boat B can be defined as a function of the location of B, and an attribute of the icon, such as its colour, defined to reflect the value of *legal(B)*.

A lengthsman patrolling a stretch of canal might enter the location of a boat B by keying in its identification code and pointing at a map on the screen. The effect of such an action would be to define *loc(B)*, and to display a suitably coloured icon on the map. Note that the information displayed would still be valid after movement of the boat, acquisition of a licence or a change of responsible authority.

Example 2 illustrates an important distinction between information that can be inferred entirely from "knowing the time" and information that cannot possibly be preconceived when setting up the model. By applying definitive principles, both types of knowledge can be represented in a convenient and flexible manner. Definitive scripts have advantages in common with constraints and logical specification where maintaining consistency between data items is concerned. Unlike these specification techniques, a definitive script represents particular states of knowledge that can be readily modified to reflect new observations. For instance, the historical data about canal locations is potentially inaccurate, and subject to revision in the light of archaeological findings – where such inaccuracies are identified, they can be interactively resolved through redefinition.

2.3 Technical Challenges to be met

The virtues of definitive programming are not adequately represented by the illustrative examples above. The most significant advantage of a definitive approach is that we can readily program interactions that were not preconceived when setting up the model. This is possible because the model implicit in the system is already rich enough to simplify the programming task. Whether the naive user of a GIS could be trained to apply definitive principles is a controversial issue, but it is arguably no more difficult to construct a definitive script than to conceive the data relationship it captures. Our practical

experience proves that even novice programmers can adapt existing scripts that perform a recognisably similar function, just as engineers exploit catalogues of mechanical linkages.

More fundamental challenges for user-interface design are illustrated by contrasting Examples 1 and 2 above. Information about the temporary closure of canal sections is accessible and topical only in respect of recent past, present or future time. We represent this by introducing a notion of *current time* into the GIS, measured by days and weeks rather than years. Contemplating the canal system in the historical time perspective is conceptually distinct from a day-to-day view. Both views can be supported by specifying an appropriate set of definitions, but there is an essential need for management as well as mere representation of alternative views.

Developing a suitable user-interface involves representing an enormous number of modes of observation and privileges to intervene. Relevant issues include: observing a single object at different levels of abstraction, dealing with references to observations and recording histories of observation. Different views are not necessarily compatible with a single environment – a boat cannot travel on a canal that is under construction. It seems probable that geometric representations, perhaps resembling Harel's statecharts [9], will be necessary to convey the subtle ways in which who and where an agent is determines what it can see and do.

The data intensive nature of the application poses other problems not previously encountered in applying definitive programming principles to reactive systems [2, 4]. In interfacing with an existing geographical database, we have to overcome the limitations of a conventional graphical database, where the organisation of data is haphazard and the form and identity of objects is inadequately represented. Our experience so far shows the advantages of a definitive approach to incremental development of a model, where data can be introduced as it is acquired, and geometric inconsistencies resolved as they are detected.

The complexity of geographical images poses technical problems in efficiently updating the display. When a value is changed, our interpreter selectively updates dependent variables, but this may involve much more computation than is required to make the display consistent. For instance, redefining the year in the canal model potentially affects the display status of every section of the canal, but only those sections currently being displayed need to be updated immediately. This points to a need for mechanisms of lazy evaluation [10] where maintaining up-to-date values of visible data takes priority.

Conclusions

Definitive programming has great potential as the basis for GIS development. There are difficult challenges to be addressed, but the long-term benefits will repay investment in developing concepts and tools.

Successful application of definitive principles would transform the processes of data capture and incremental system design, simplify the process of integrating map data with animated sequences, both graphics and video, and enhance the potential for special-purpose programming and customisation of the interface to a particular task.

The major technical issues to be addressed are: understanding and formally describing the levels of reference required, avoiding the syntactic convolutions that these generate in our present framework, and ultimately making the package accessible to the ordinary user.

References

[1] W.M. Beynon, *Definitive Principles for Interactive Graphics*, NATO ASI Series F, Vol 40, Springer-Verlag 1988, pp. 1083-1097, 1988.

[2] W.M. Beynon, I. Bridge and Y.P. Yung, *Agent-oriented modelling for a Vehicle Cruise Controller*, in Proc. Eng. Sys. Design and Analysis, ASME PD-47-4:159-165, 1992.

[3] W.M. Beynon and A.J. Cartwright, *A Definitive Programming Approach to the Implementation of CAD Software* in Intelligent CAD Systems II: Implementation Issues, Springer-Verlag, pp. 456-468, 1989.

[4] W.M. Beynon, M.T. Norris, R.A. Orr and M.D. Slade, *Definitive Specification for Concurrent Systems*, in Proc. UKIT'90, IEE Conference Publications 316, pp. 52-57, 1990.

[5] W.M. Beynon, Y.P. Yung, A.J. Cartwright and P.J. Horgan, *Scientific Visualisation: Observations and Experiments*, in Proc. Eurographics Workshop on Scientific Visualisation, Viareggio, 1992.

[6] M. Chmilar and B. Wyvill, *A Software Architecture for Integrated Modelling and Animation*, New Advances in Computer Graphics: Proc of CGI'89, 257-276, 1989.

[7] R. Dean, *Historical Map of the Birmingham Canals*, M&M Baldwin, Cleobury Mortimer, 1989.

[8] L.A. Edwards, *Inland Waterways of Great Britain*, Imray, Laurie, Norie and Wilson, St. Ives, Cambs., 1985.

[9] D. Harel, On Visual Formalisms, *CACM*, 31(5):514-530, 1988.

[10] P. Henderson and J.H. Morris, *A Lazy Evaluator*, Conference Record of the Third Annual ACM Symposium on Principles of Programming Languages, Atlanta, GA, 1976.

[11] Inland Waterways Association, *Birmingham Canal Navigations: A Cruising and Walking Guide*, London, 1984.

[12] B. Meyer, *Object-oriented Software Construction*, Prentice-Hall, 1988.

[13] J.M. Pearson, *Canal Companion: Birmingham Canal Navigations*, J.M. Pearson and Associates, Burton-on-Trent,1989.

[14] A. Sutcliffe, *Jackson System Development*, Prentice-Hall, 1988.

[15] B. Wyvill, *An interactive Graphics Language*, PhD Thesis, Bradford University, 1975.